UNIVERSITÄT BERN

Beyond Supervised Learning

Paolo Favaro

Computer Vision Group
Computer Science Department - University of Bern

Overview

- Goal: Representation learning for transfer learning
- Part I: Beyond human annotation
  - Splitting data in two parts
  - Known relations between multiple parts
  - Learning relations between multiple parts
- Part II: Beyond human annotation
  - Generative models

Part I is about SSL.

Representation Learning

- Suppose to be given data $x \sim p_x$ and a task
- Representation learning is about transforming x into a z that can be used to solve the task with simple functions (eg a linear classifier)

In transfer learning we consider two data domains that have common factors of variations and use concepts that were learned in one domain to help learn concepts in the other domain.

Example

- **Dataset** of cars and non-cars

x




$y = 1$ (cars)

$y = 0$ (non-cars)

Features

Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

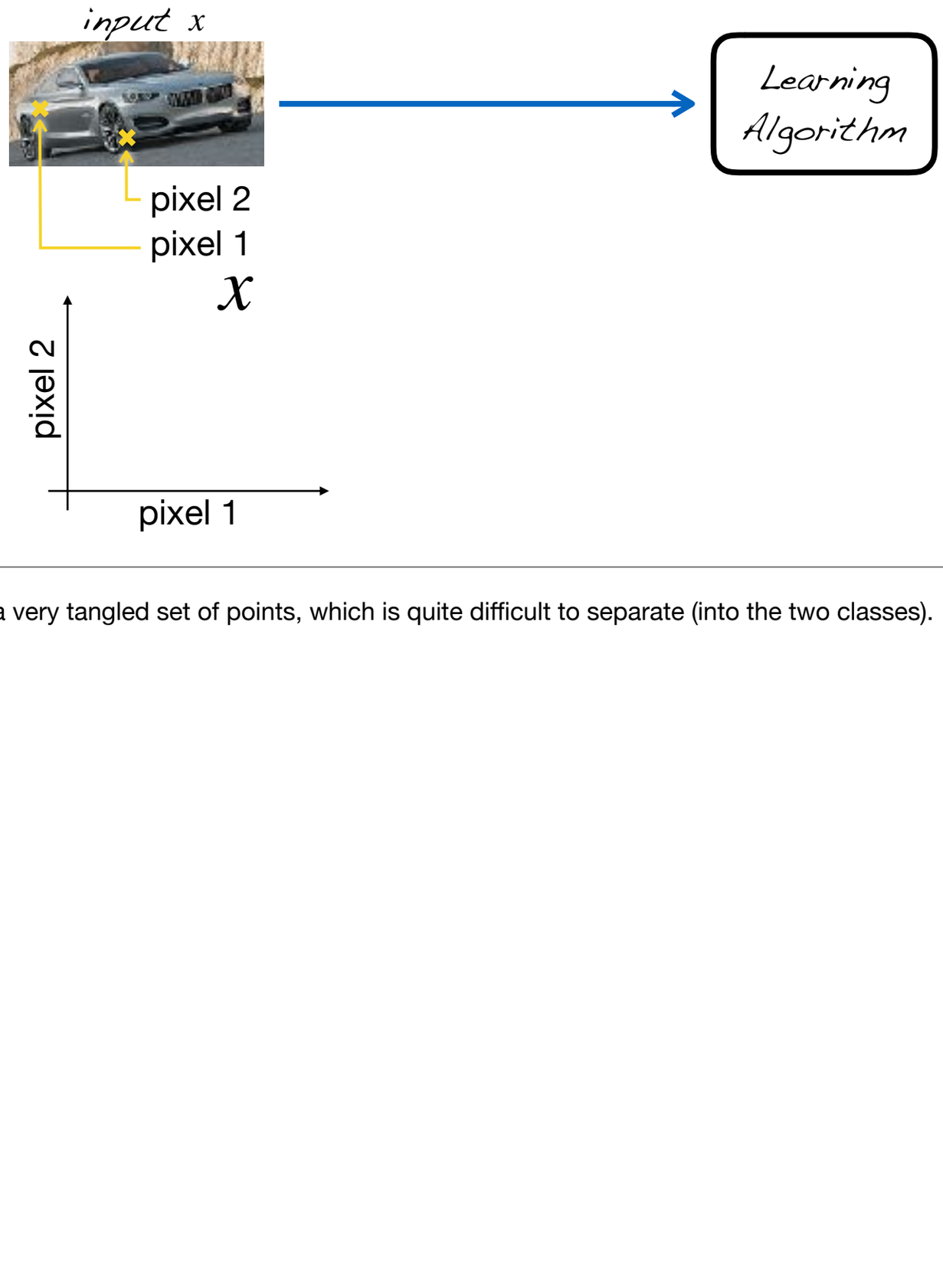
Features

Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features

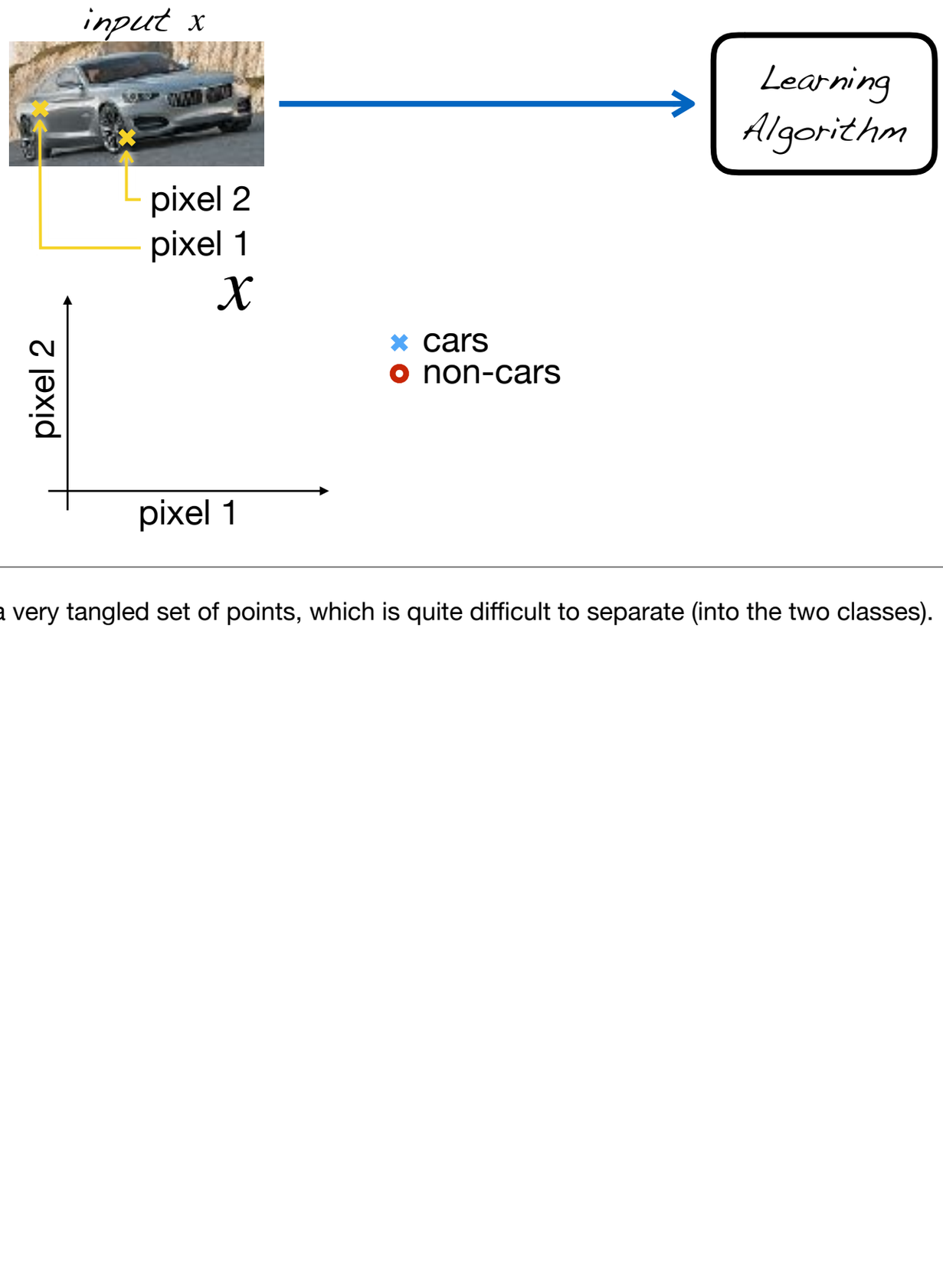
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



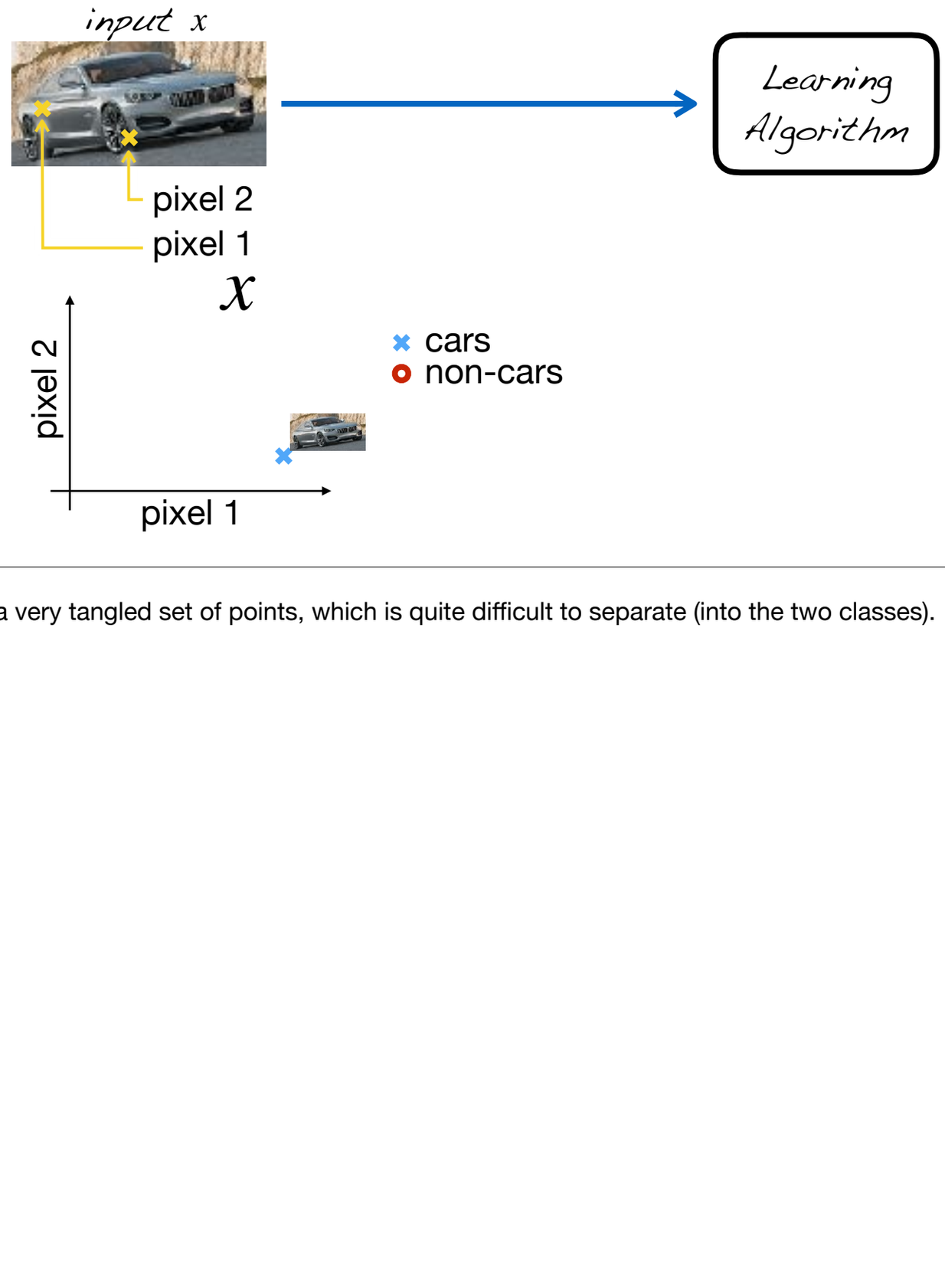
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



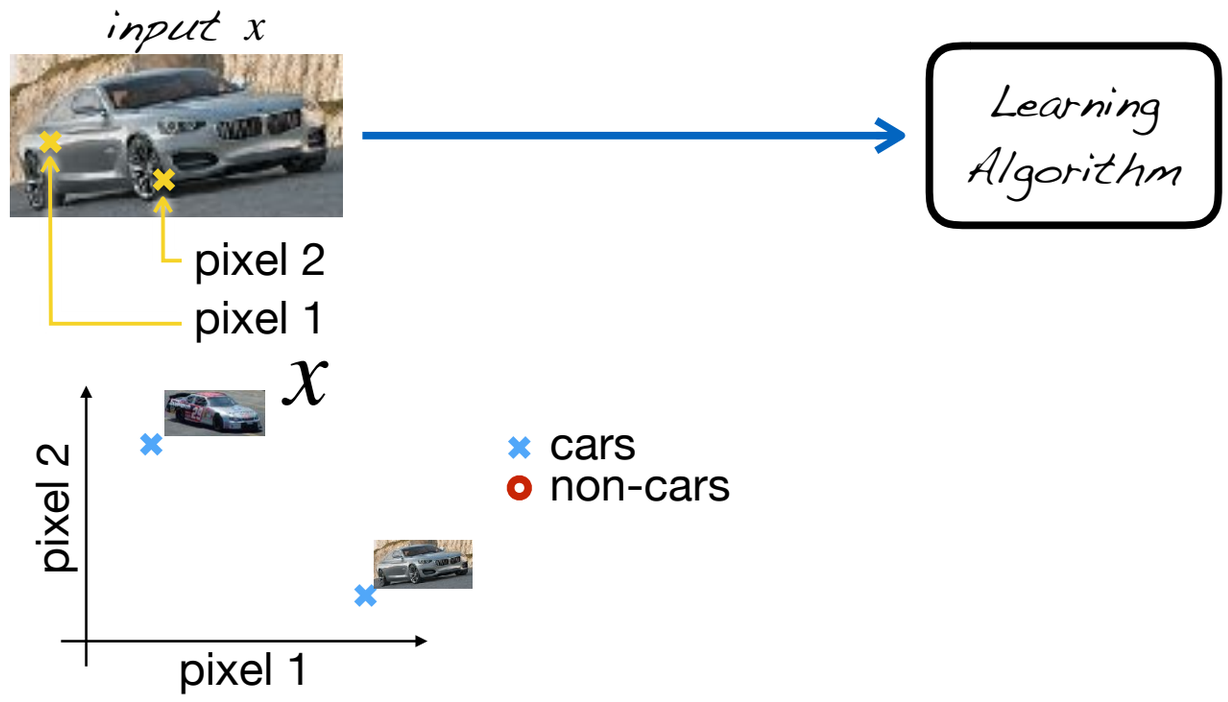
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



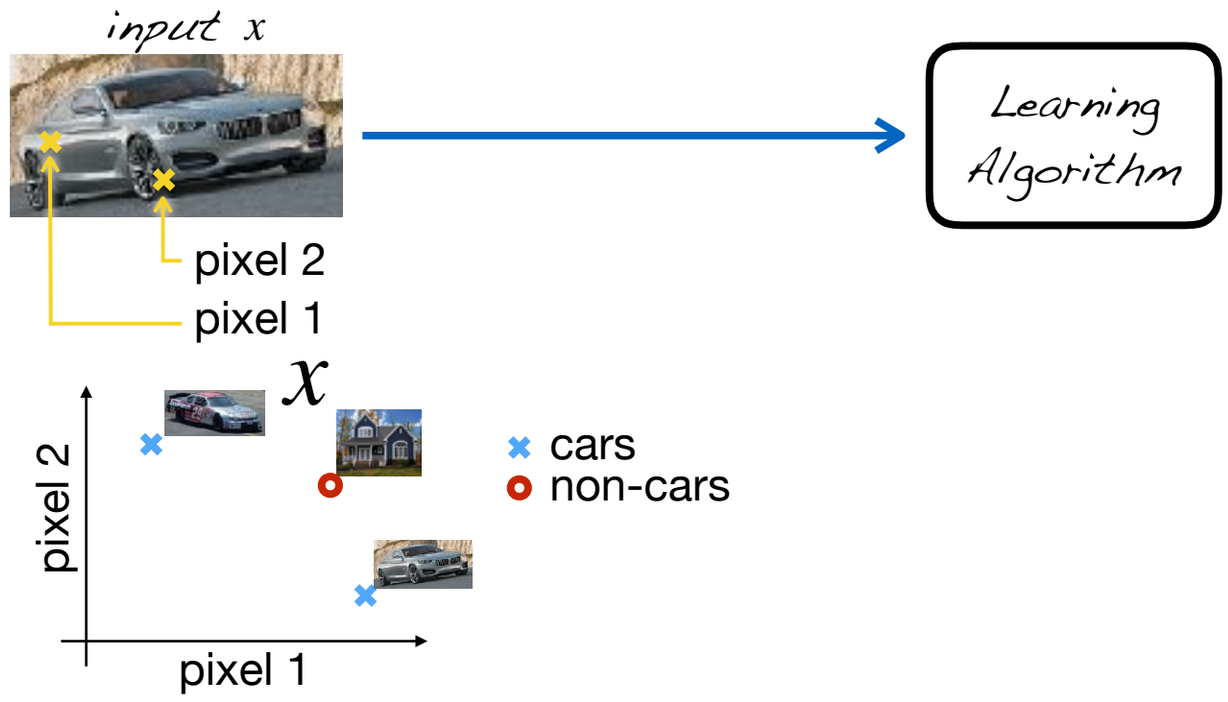
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



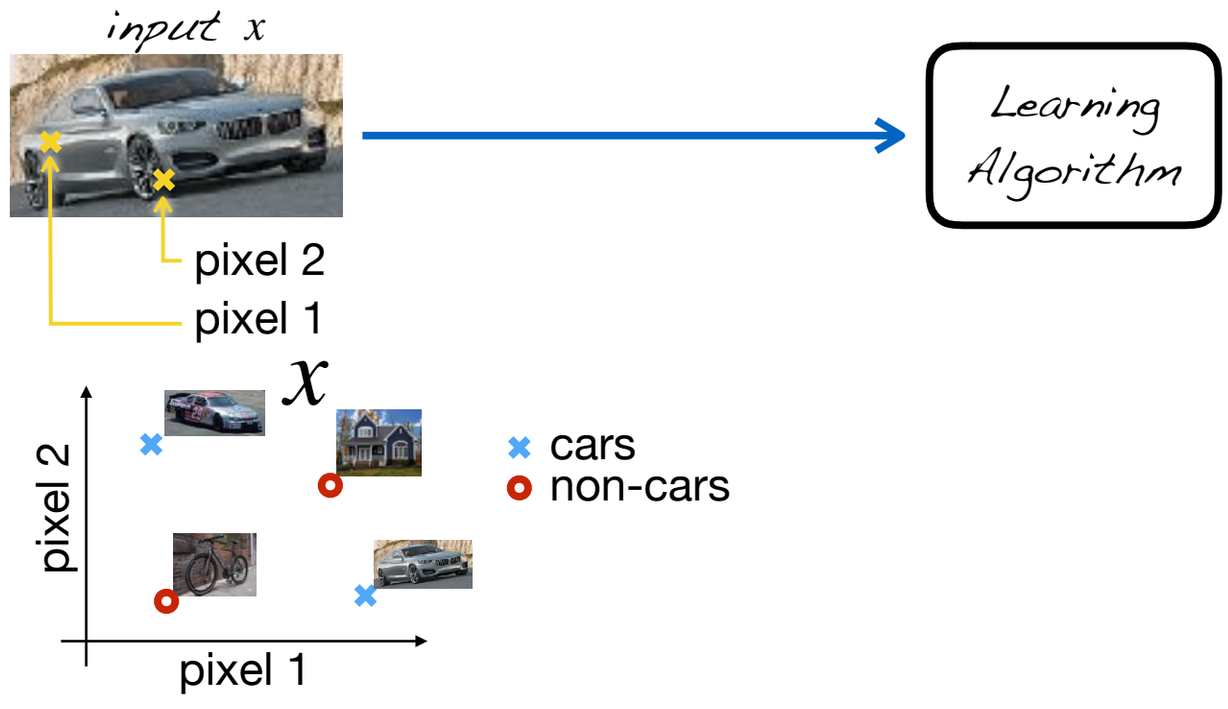
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



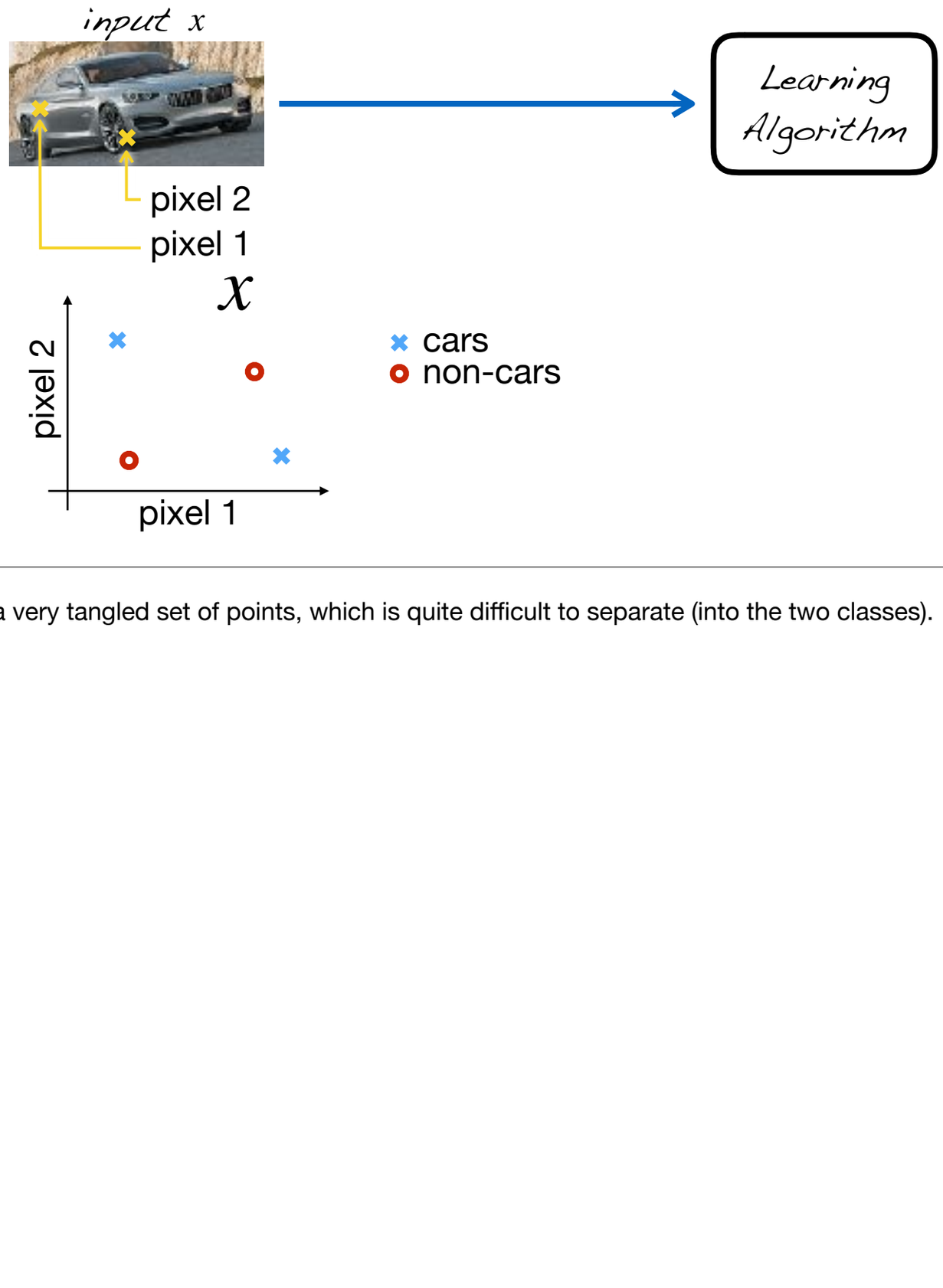
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



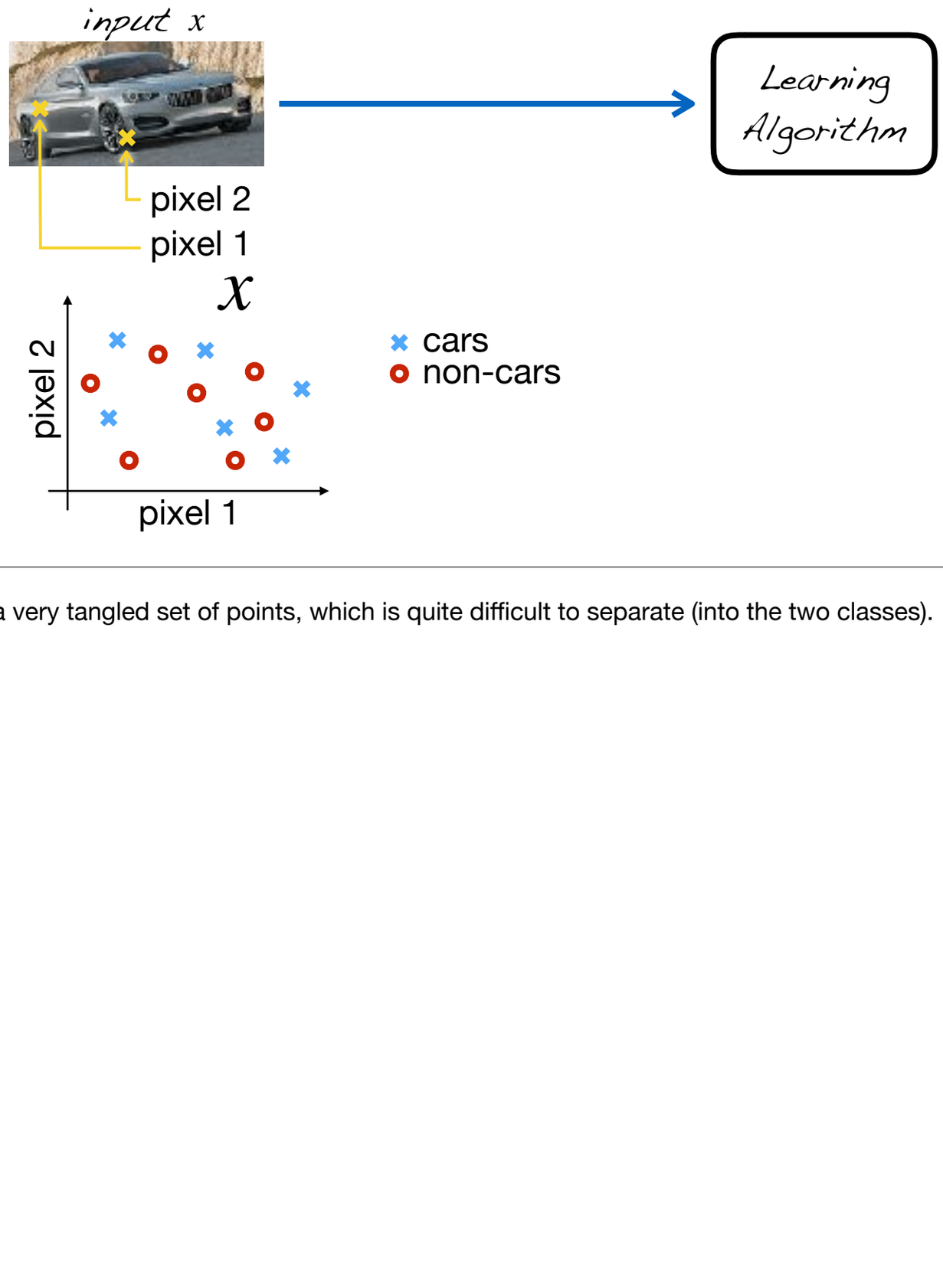
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



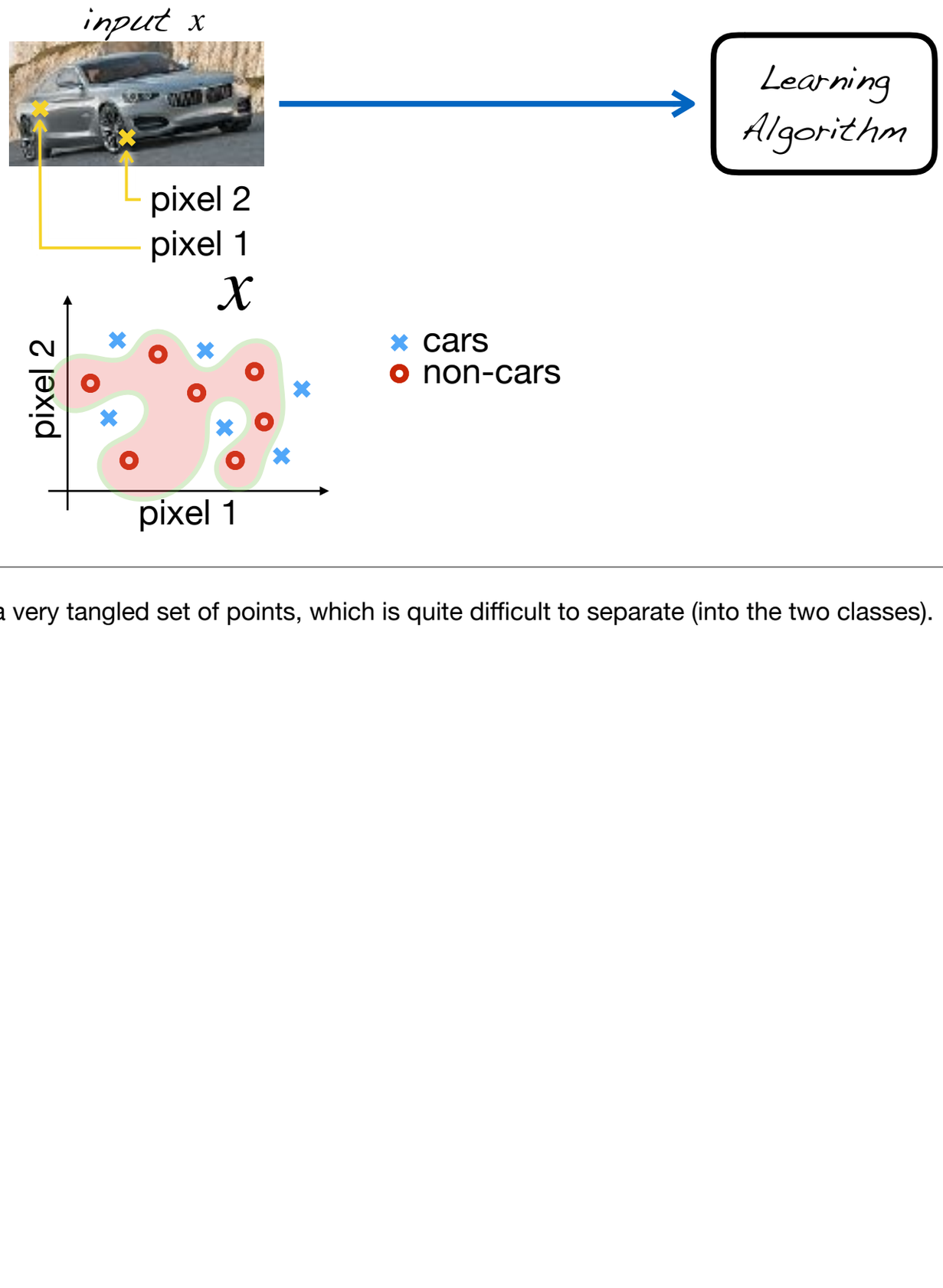
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



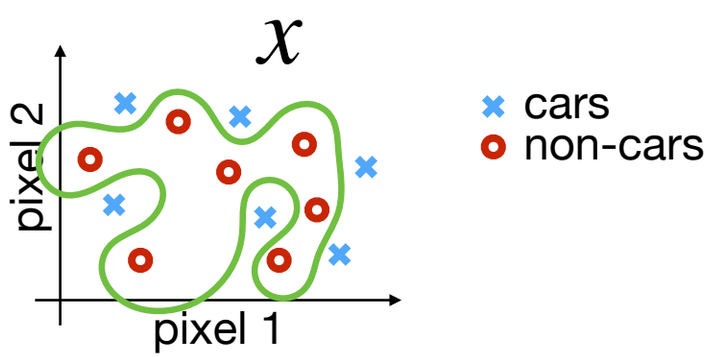
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



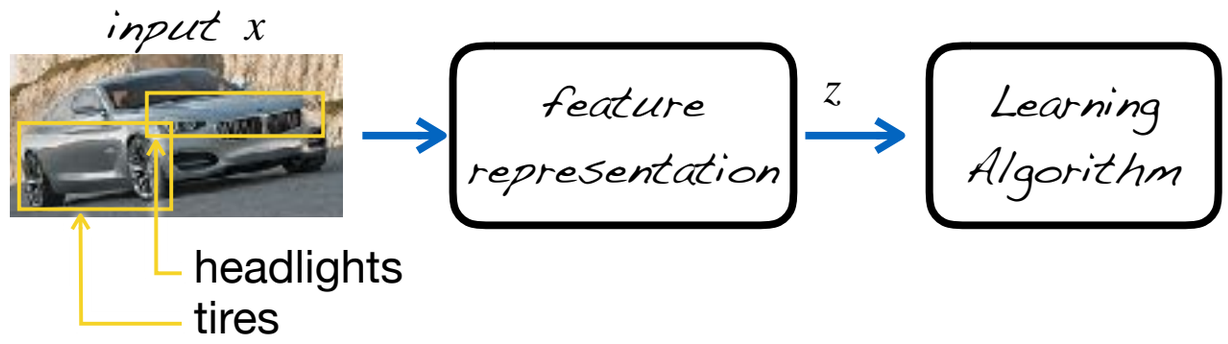
Working directly on the raw data leads to a very tangled set of points, which is quite difficult to separate (into the two classes).

Features



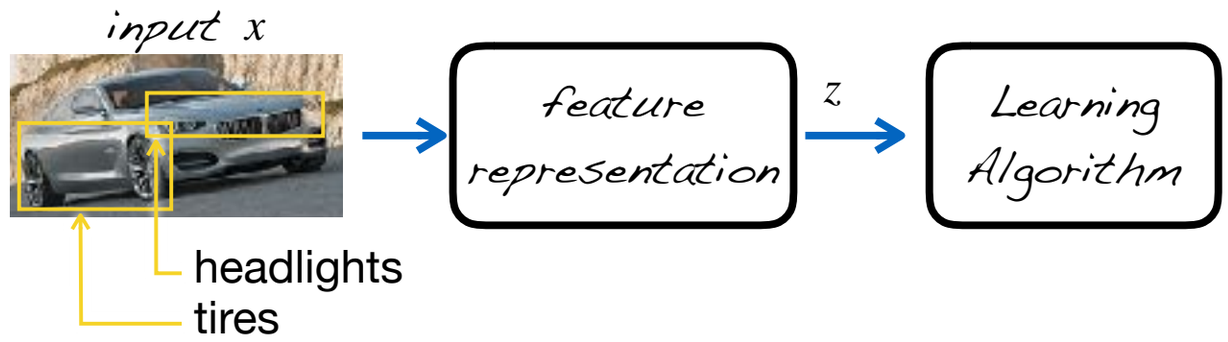
The separation may become easier if we use some intermediate representation (features).
This shows then two problems: one is the identification of suitable features (so that important factors of variation are separated — in the example the category car) and the other is the classification of the features of the input data. The classification can be achieved, for example, with SVM (see next slide).
Unfortunately, the above features are as difficult to detect as much as the original problem (car detection). Therefore, typically the features are much less ambitious (low-level) statistics of the image.

Features



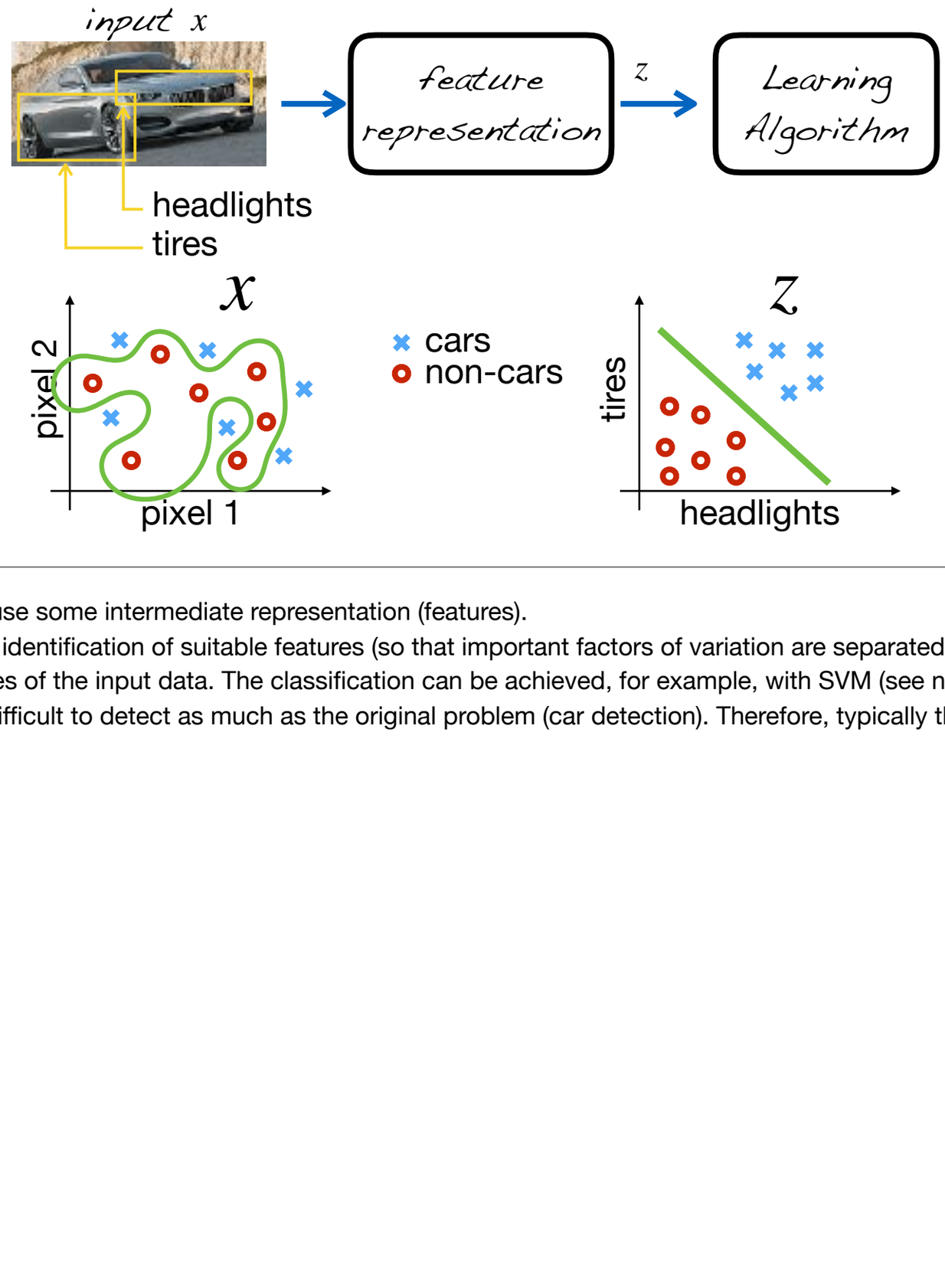
The separation may become easier if we use some intermediate representation (features).
This shows then two problems: one is the identification of suitable features (so that important factors of variation are separated — in the example the category car) and the other is the classification of the features of the input data. The classification can be achieved, for example, with SVM (see next slide).
Unfortunately, the above features are as difficult to detect as much as the original problem (car detection). Therefore, typically the features are much less ambitious (low-level) statistics of the image.

Features



The separation may become easier if we use some intermediate representation (features).
This shows then two problems: one is the identification of suitable features (so that important factors of variation are separated — in the example the category car) and the other is the classification of the features of the input data. The classification can be achieved, for example, with SVM (see next slide).
Unfortunately, the above features are as difficult to detect as much as the original problem (car detection). Therefore, typically the features are much less ambitious (low-level) statistics of the image.

Features



The separation may become easier if we use some intermediate representation (features).
This shows then two problems: one is the identification of suitable features (so that important factors of variation are separated — in the example the category car) and the other is the classification of the features of the input data. The classification can be achieved, for example, with SVM (see next slide).
Unfortunately, the above features are as difficult to detect as much as the original problem (car detection). Therefore, typically the features are much less ambitious (low-level) statistics of the image.

Transfer Learning

- Learn a representation for $x \sim p_x$ and then use it to solve a problem in a different, but related, domain with data $w \sim p_w$
- We assume that many factors of variations in the data x and w are **shared**
- **Example**

  x are images of cats and dogs

  w are images of ants and wasps

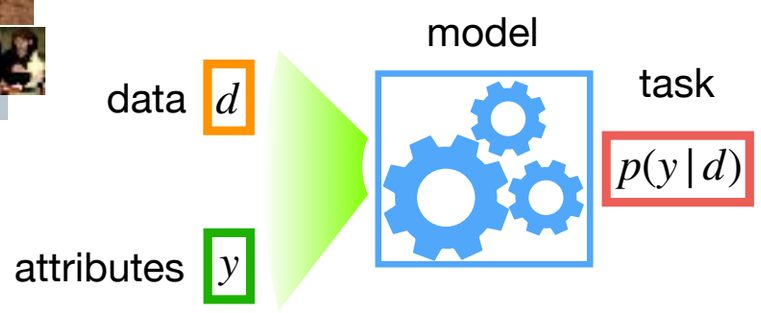
  They share low-level texture details, geometrical and illumination effects

In transfer learning we consider two data domains that have common factors of variations and use concepts that were learned in one domain to help learn concepts in the other domain.

Transfer Learning

- There are a **source** domain and task where we perform the initial learning of a representation
- The source task is typically rich of data so that one can build a general representation
- The **target** task has typically not enough data
- Learning the target task is our main priority

Supervised Learning

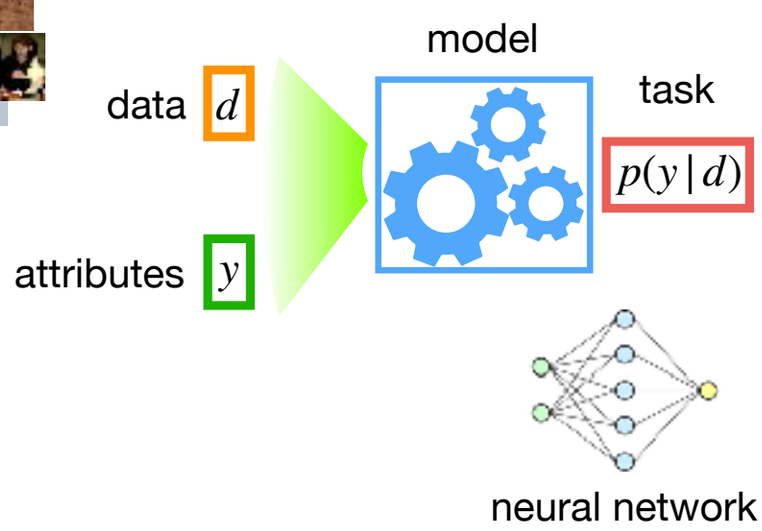


The source task is often defined as a supervised learning problem.
It is well defined. We take data and attributes and try to approximate the conditional probability that describes their dependency.
More explicitly, we look for redundancy in x to identify y.

Supervised Learning

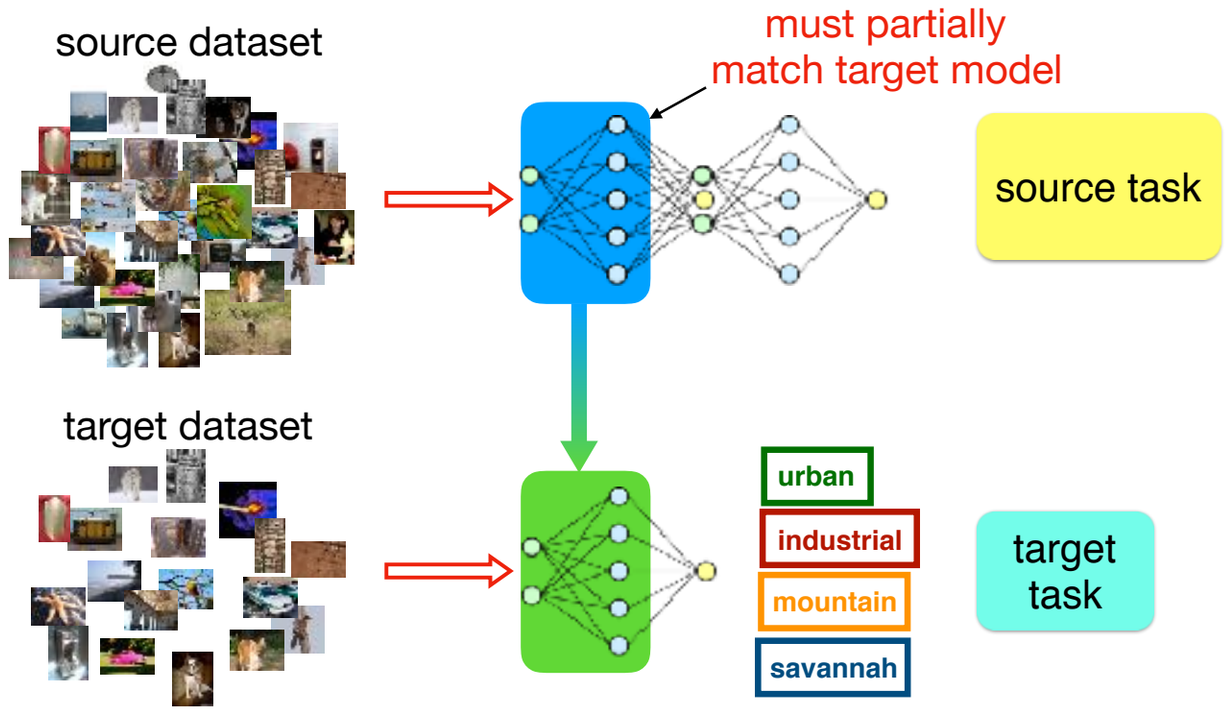
book car

shield dog



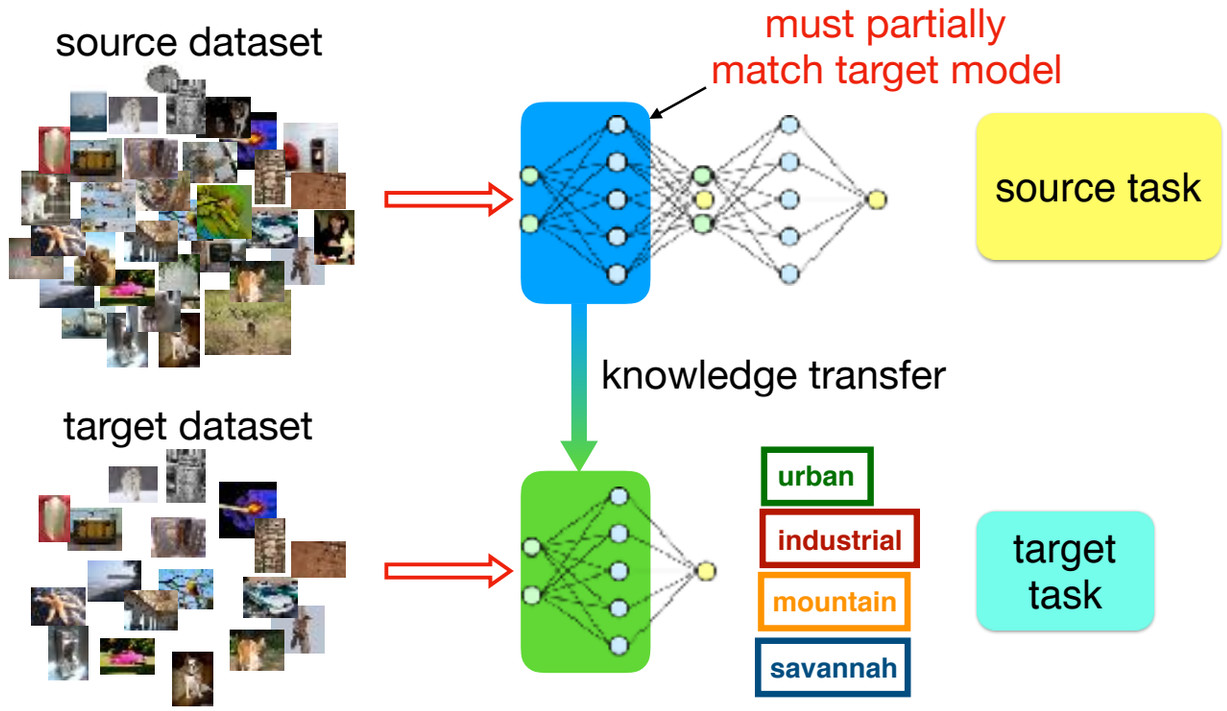
The source task is often defined as a supervised learning problem.
It is well defined. We take data and attributes and try to approximate the conditional probability that describes their dependency.
More explicitly, we look for redundancy in x to identify y.

Transfer via Fine-Tuning



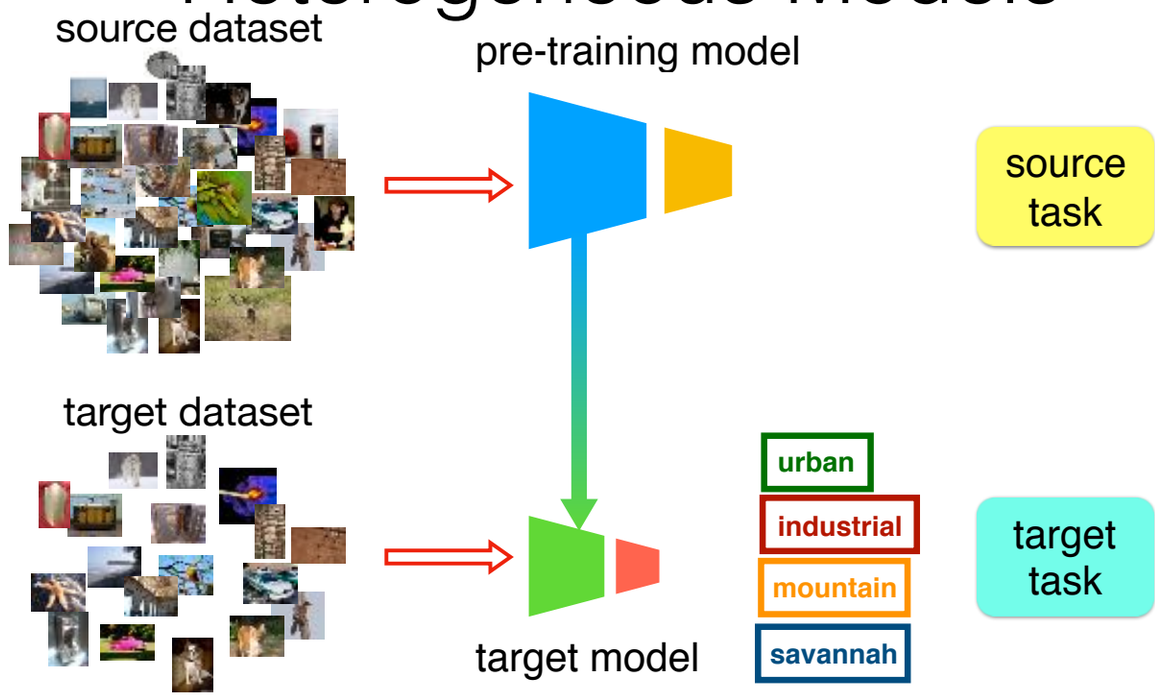
Once trained our model on the source task, we can transfer the learned representation
to the target model.
The copy/paste of the parameters is not ideal as it forces the two models to partially match in their architecture.
An alternative method is to transfer the knowledge by copying the feature space.
This method is more general as it allows to transfer knowledge between different models.

Transfer via Fine-Tuning



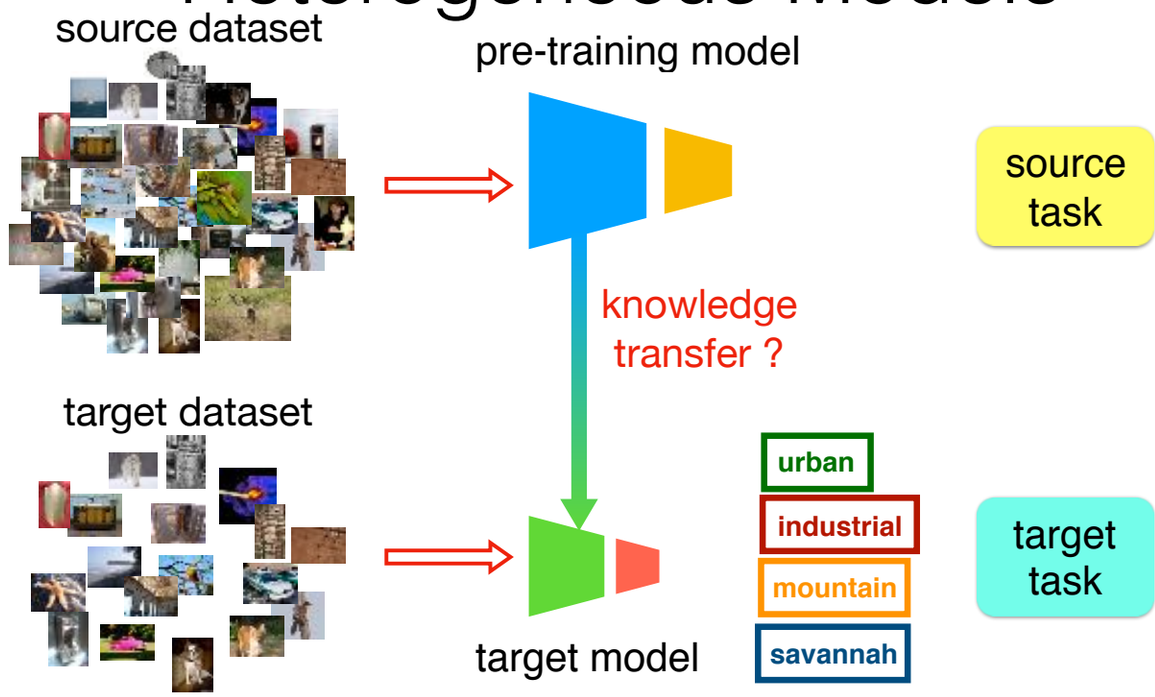
Once trained our model on the source task, we can transfer the learned representation
to the target model.
The copy/paste of the parameters is not ideal as it forces the two models to partially match in their architecture.
An alternative method is to transfer the knowledge by copying the feature space.
This method is more general as it allows to transfer knowledge between different models.

Knowledge Transfer between Heterogeneous Models



Boosting Self-Supervised Learning via Knowledge Transfer, M. Noroozi et al, CVPR 2018

Knowledge Transfer between Heterogeneous Models



Boosting Self-Supervised Learning via Knowledge Transfer, M. Noroozi et al, CVPR 2018

Knowledge in a Model

- (Task) knowledge in a model is represented by the **parameters** estimated through training

Note

Knowledge in a Model

- (Task) knowledge in a model is represented by the **parameters** estimated through training

- A dual interpretation of task knowledge is given by the **feature space** induced by the parameters

Note

Knowledge in a Model

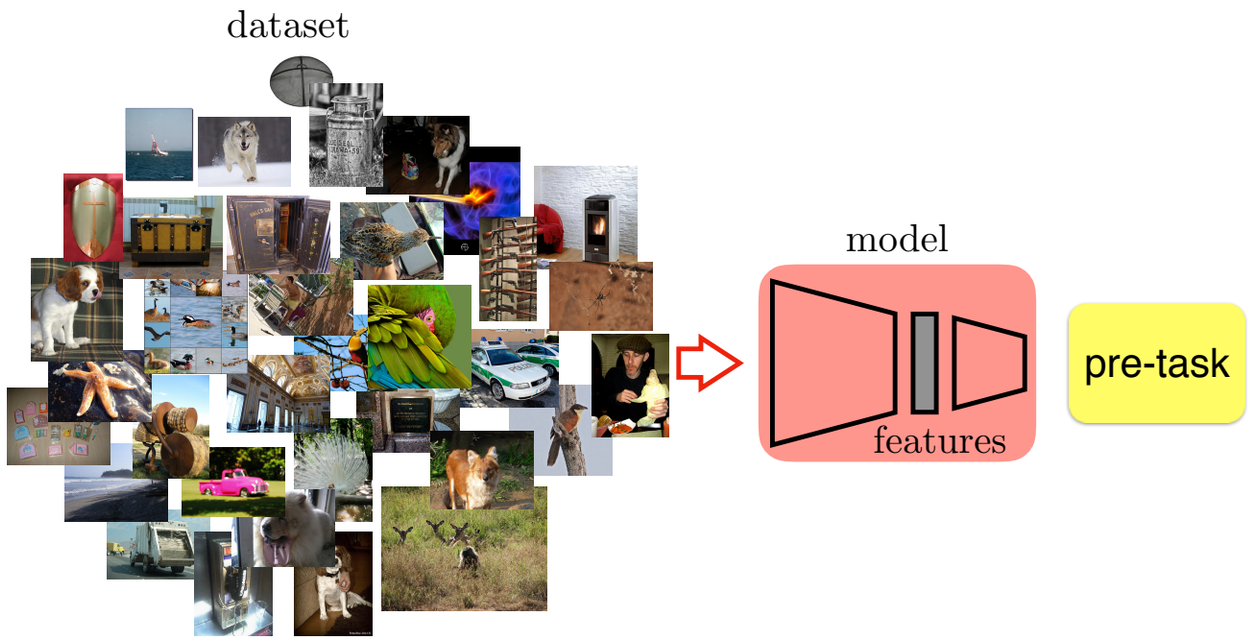
- (Task) knowledge in a model is represented by the **parameters** estimated through training

- A dual interpretation of task knowledge is given by the **feature space** induced by the parameters

  - i.e., which samples are close and which far apart

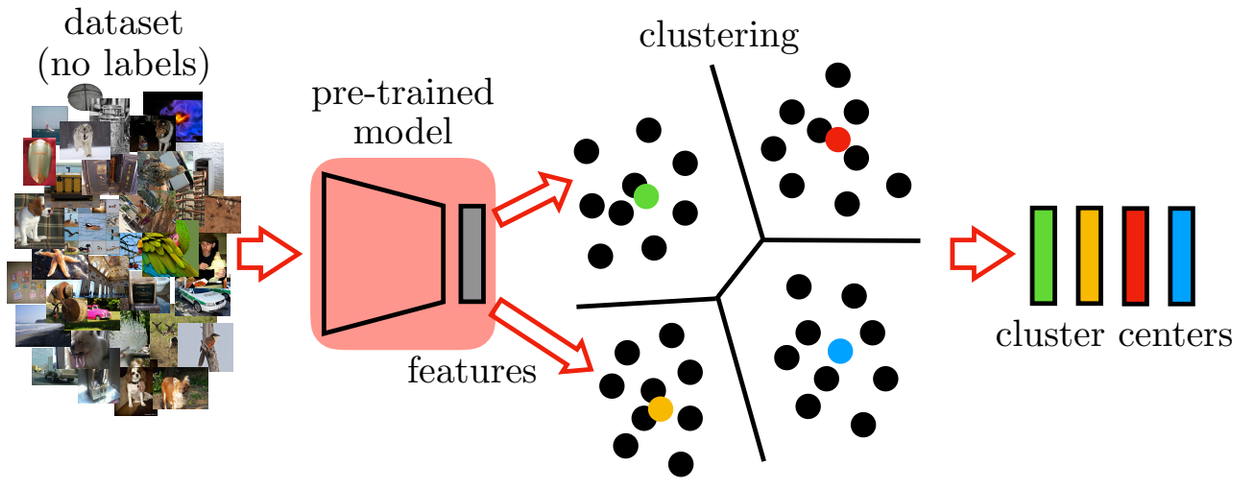
Note

Pre-Training

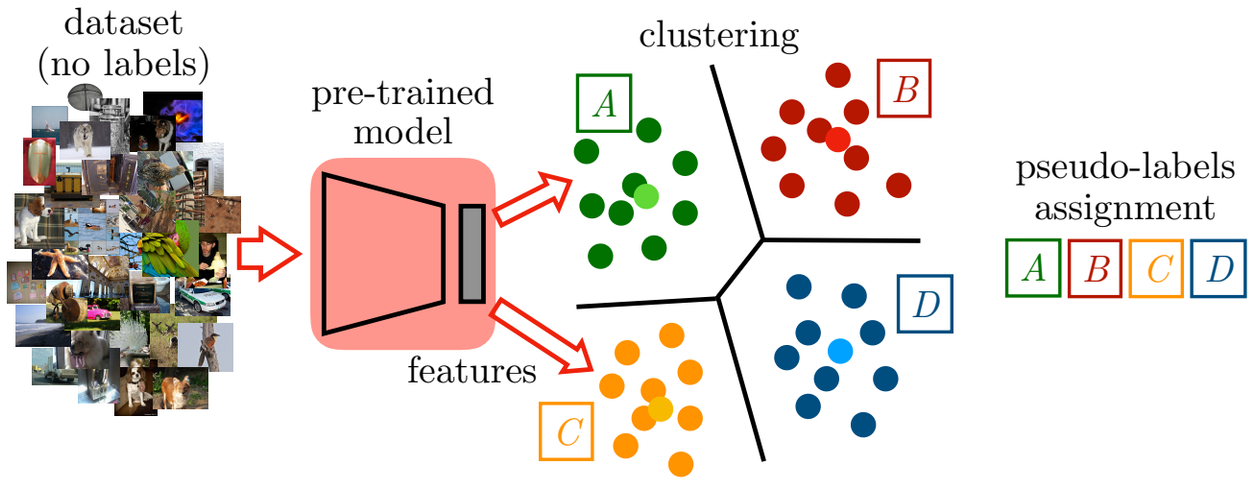


This is split in 4 steps: pre-training, clustering, pseudo-labels extraction, cluster classification.

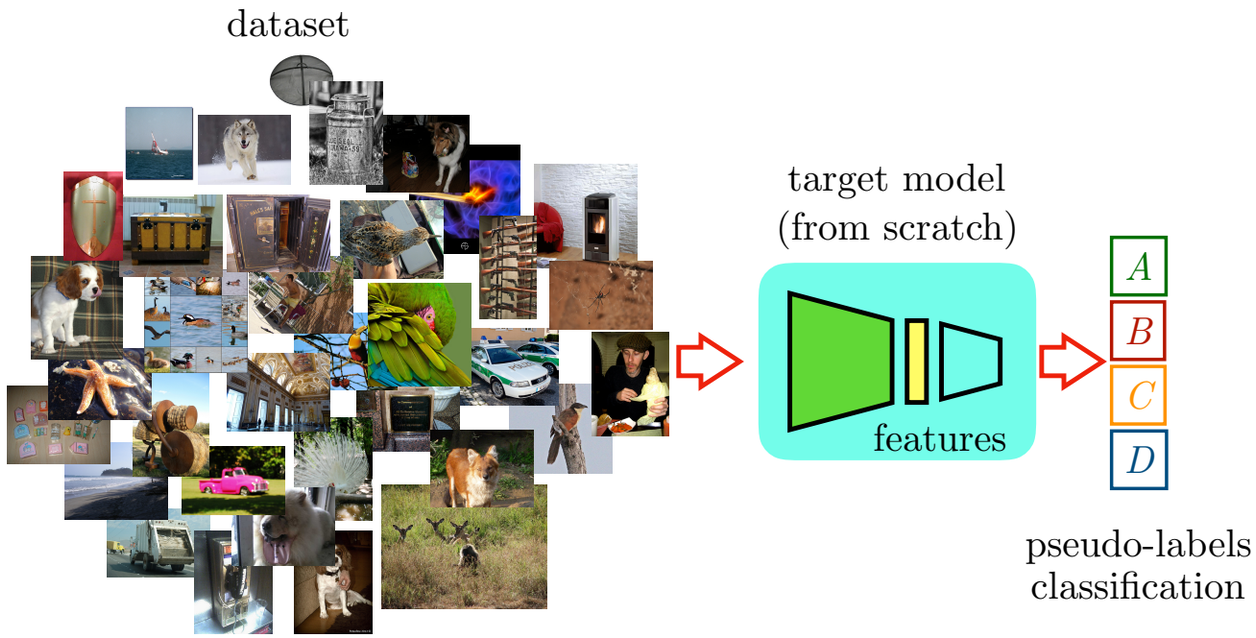
Clustering


dataset
(no labels)
pre-trained
model
features
clustering
cluster centers

Extracting Pseudo-Labels



Knowledge Transfer via Pseudo-Labels Classification



So now we have 2 effective ways to transfer knowledge from some pre-trained model to another.

Manual Annotation

- Costly
- Time-consuming
- Error-prone
- Not scalable to large datasets
- Might require uncommon expertise (eg. medical imaging)
- Might not be well-defined (eg. annotating actions in videos)

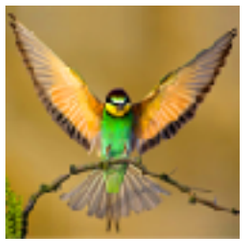
Microsoft COCO
Common Objects
in Context

Now that we solved how we would transfer the knowledge from one dataset and model to another, we discuss briefly the issues with manual annotation that appear in supervised learning.
This part is about the first training step which generates the knowledge that we want to transfer.

Part I
Beyond Human Annotation

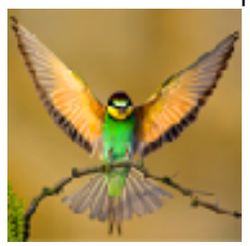
Self-Supervised Tasks

- 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)




Self-Supervised Tasks

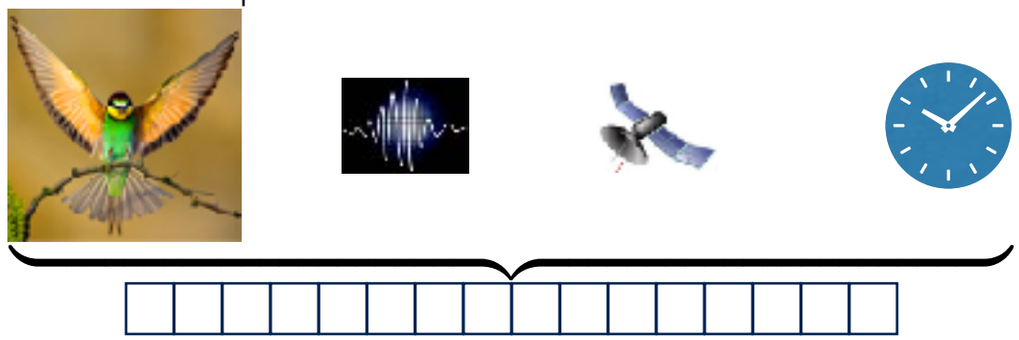
- Collect each piece of information into a vector




This works only because there is lots of redundancy in the data, even within each homogeneous type

Self-Supervised Tasks

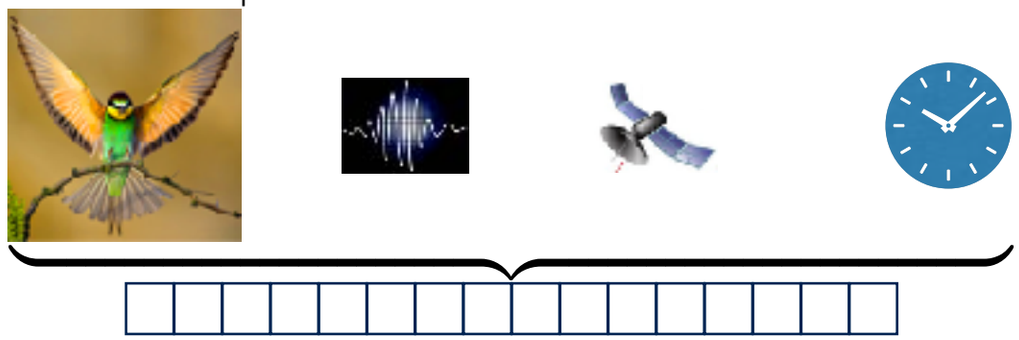
- Collect each piece of information into a vector



This works only because there is lots of redundancy in the data, even within each homogeneous type

Self-Supervised Tasks

- Collect each piece of information into a vector

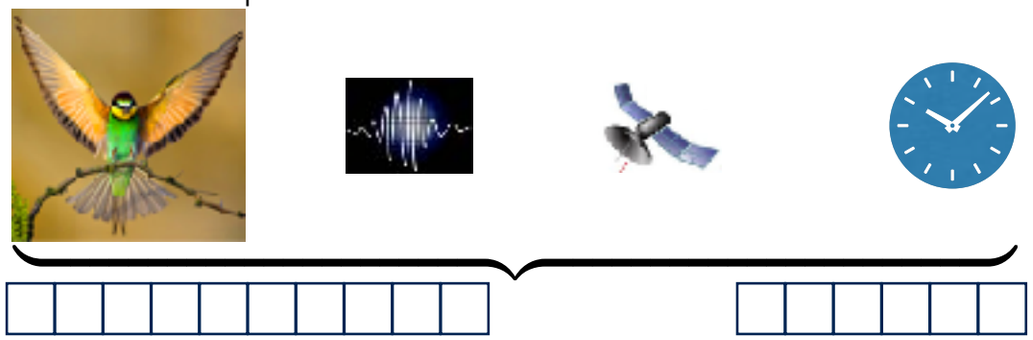


- Transform the vector into two parts: data and targets

This works only because there is lots of redundancy in the data, even within each homogeneous type

Self-Supervised Tasks

- Collect each piece of information into a vector

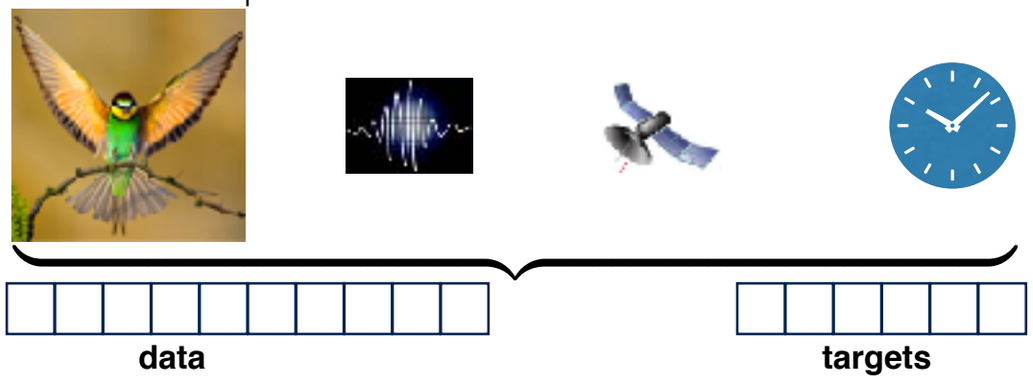


- Transform the vector into two parts: data and targets

This works only because there is lots of redundancy in the data, even within each homogeneous type

Self-Supervised Tasks

- Collect each piece of information into a vector

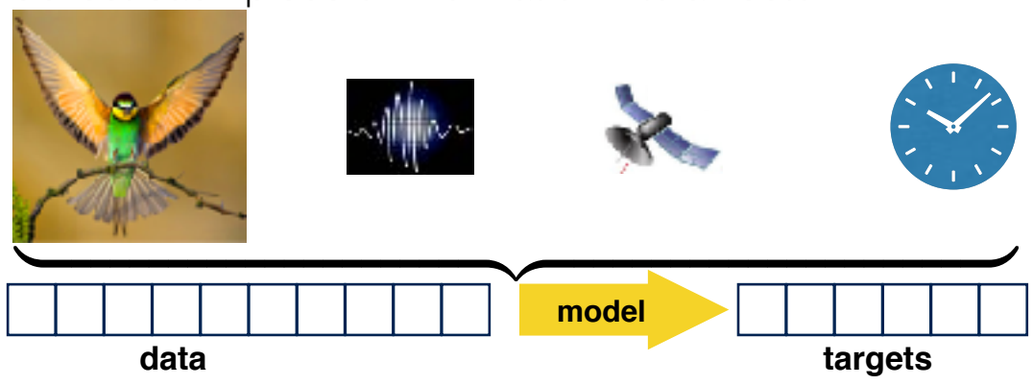


- Transform the vector into two parts: data and targets

This works only because there is lots of redundancy in the data, even within each homogeneous type

Self-Supervised Tasks

- Collect each piece of information into a vector



- Transform the vector into two parts: data and targets
- Train a model to retrieve the **missing** observations (the targets) in the data

This works only because there is lots of redundancy in the data, even within each homogeneous type

Prior Work

- 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

Virginia De Sa, Learning classification from unlabeled data, NIPS 1994

Self-Supervised Learning vs Unsupervised Learning

- In UL we build an approximate model of $p(x)$
- In SSL split x into x_1 and x_2 and train a model for

$$\phi(x_1) = x_2 \qquad \text{or} \qquad \phi(x_1) = p\left(x_2 \,|\, x_1\right)$$

- In the SSL formulation we can therefore write

$$p(x) = p\left(x_2 \,|\, x_1\right) p\left(x_1\right) = \phi(x_1) p\left(x_1\right)$$

Learning a Self-Supervised Task

- The previous relations define a **supervision signal**
- The conditional probability $\phi(x_1) = p\left(x_2 \,|\, x_1\right)$ is 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

$$\phi\left(T_1(x)\right) = T_2(x) \quad \text{or} \quad \phi(T_1(x)) = p\left(T_2(x) \mid T_1(x)\right)$$

- The loss function is therefore either

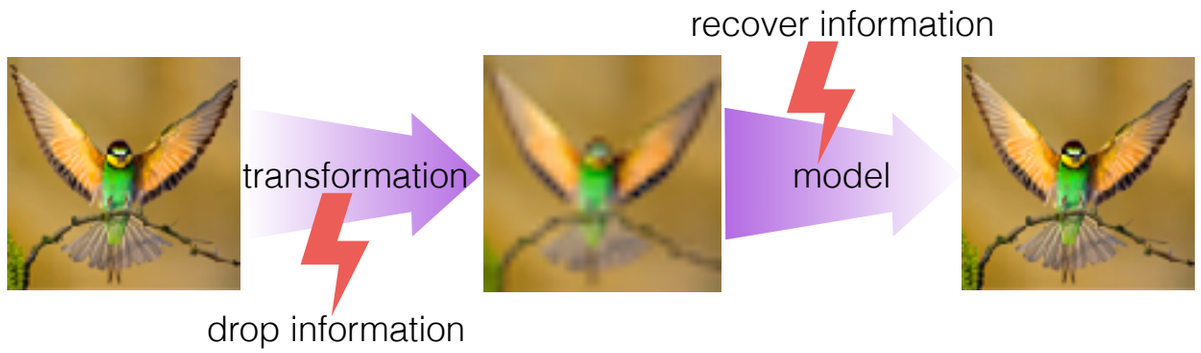
$$\mathscr{L}(\phi) = \frac{1}{m}\sum_{i=1}^{m}\left|\phi\left(T_1\left(x^{(i)}\right)\right) - T_2\left(x^{(i)}\right)\right|$$

or

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

Learning by Dropping & Retrieving Information

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

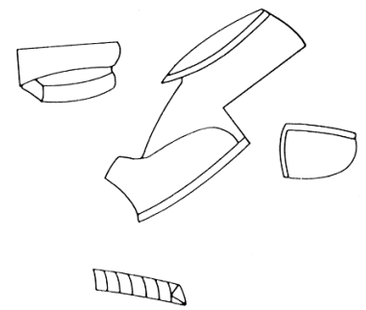


- Thus, we can recover dropped information only if it **depends** on the transformed data (we call this **redundancy**)

Thus we should only drop redundant information.
What information should we drop and retrieve to learn a representation that can generalize to new tasks?

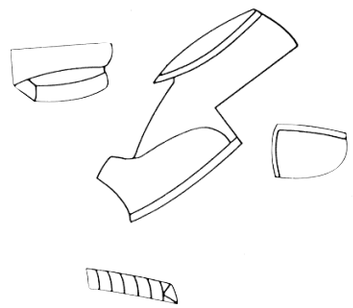
Self-Supervised Learning

- **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?



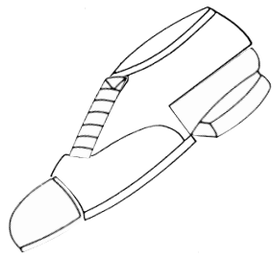
Self-Supervised Learning

- **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?



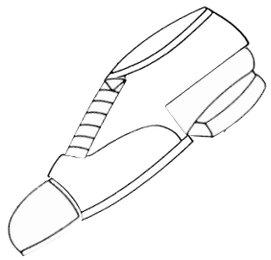
Self-Supervised Learning

- **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?

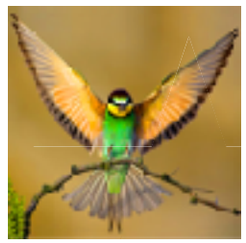


- Is it **necessary** to know **how objects are made?**

Perhaps also just looking at boundaries will suffice.

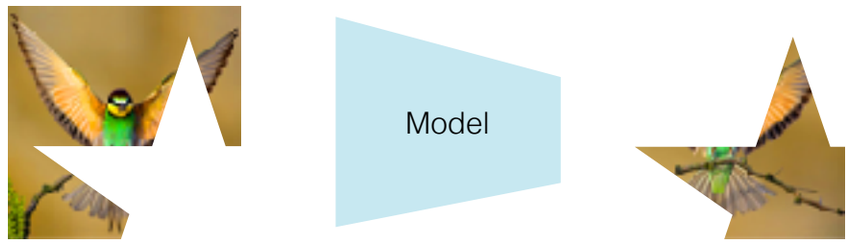
Restrictions on the Data Transformations

- 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



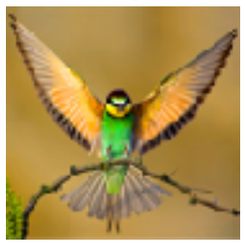
Restrictions on the Data Transformations

- 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



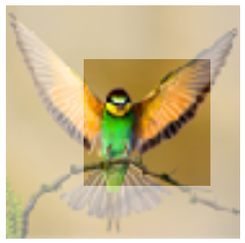
Restrictions on the Data Transformations

- 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



Restrictions on the Data Transformations

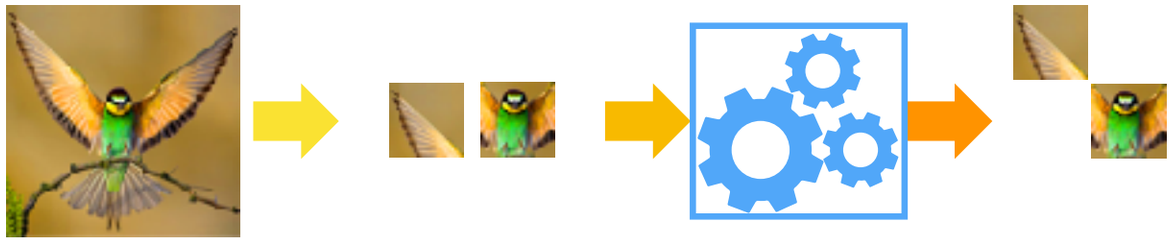
- After training the SSL model how do we use it?
- The way we split the data determines the format of the new data and this is what we need to use also later
- To feed an image to the model later on, choose the split so that the data term is an image or the cropping of an image



Context Prediction

- **Example #1**
- Drop the relative location of two tiles

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



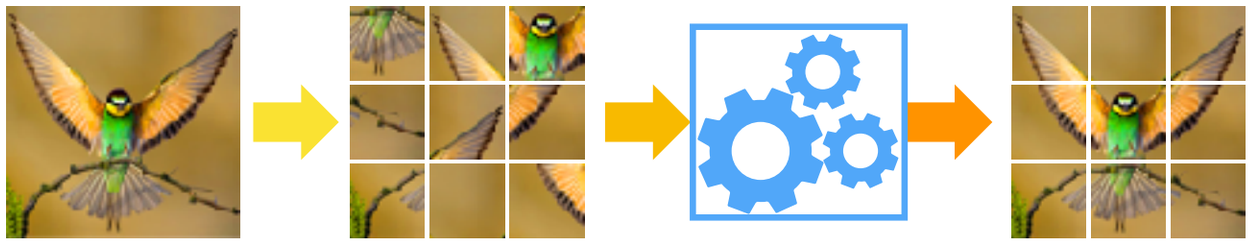
1. image **2. tiles** **3. learning** **4. location**

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

Jigsaw Puzzles

- **Example #2**
- Drop the ordering of all the puzzle tiles

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



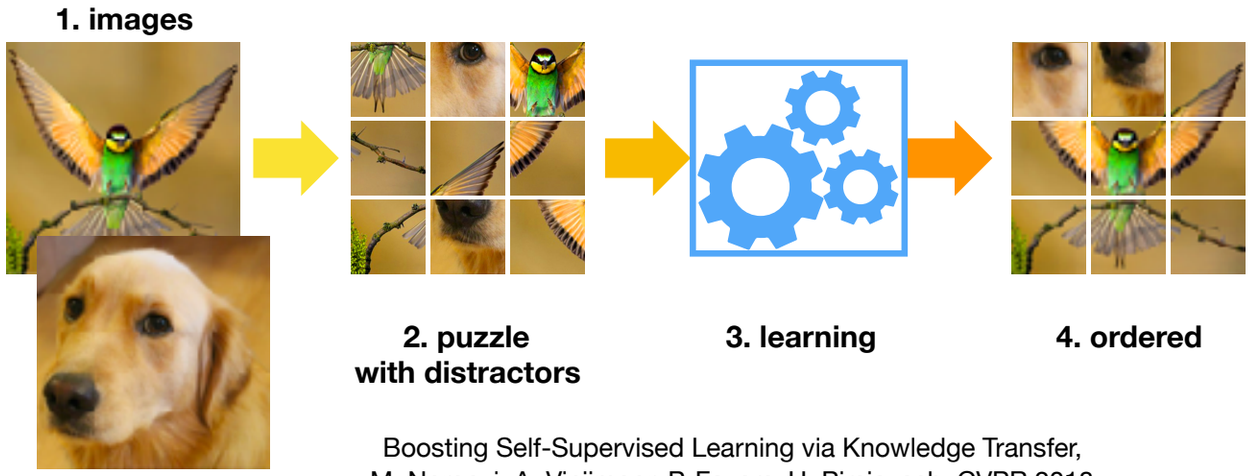
1. image **2. puzzle** **3. learning** **4. ordered**

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

Jigsaw++

- **Example #2’**
- Drop the ordering of all the puzzle tiles

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

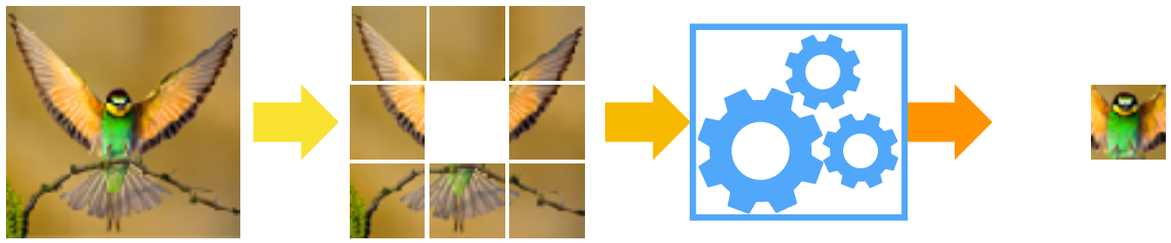


Boosting Self-Supervised Learning via Knowledge Transfer,
M. Noroozi, A. Vinjimoor, P. Favaro, H. Pirsiavash. CVPR 2018

Context Encoders

- **Example #3**
- Drop an entire image tile

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

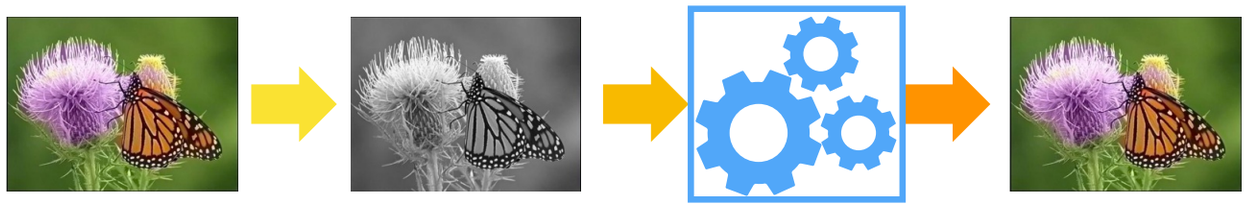


1. image **2. context** **3. learning** **4. missing tile**

Context encoders: Feature learning by inpainting
D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros, CVPR 2016

Colorization

- **Example #4**
- Drop the data colors



1. image **2. de-color** **3. learning** **4. color**

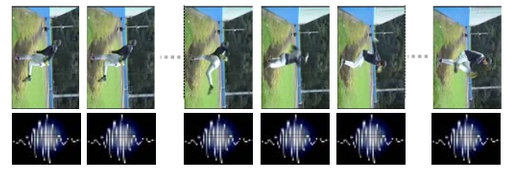
Learning representations for automatic colorization
G. Larsson, M. Maire, and G. Shakhnarovich, ECCV 2016

Heterogeneous Data

- Data can be **heterogeneous**
- Data can be collected synchronously from multiple sensors
- For example

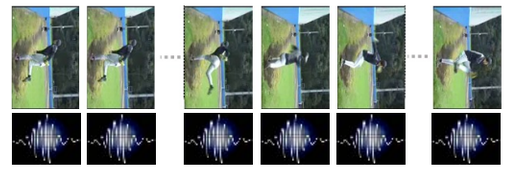
Heterogeneous Data

- Data can be **heterogeneous**
- Data can be collected synchronously from multiple sensors
- For example
  - Videos have images and audio



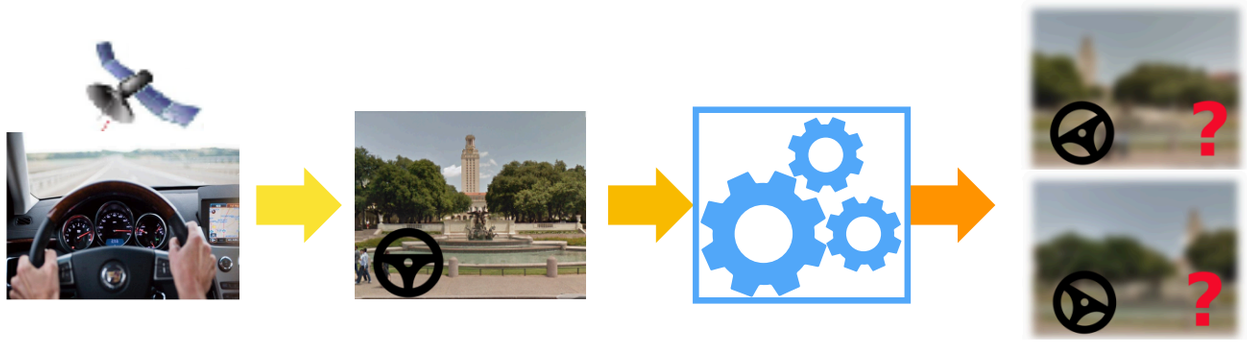
Heterogeneous Data

- Data can be **heterogeneous**
- Data can be collected synchronously from multiple sensors
- For example
  - Videos have images and audio
  - Photos/videos with GPS tagging



Ego Motion

- **Example #5**
- Predict motion from images



1. video + motion **2. video** **3. learning** **4. motion**

Learning image representations tied to ego-motion
Jayaraman and Grauman, ICCV 2015

Towards General Self-Supervised Learning

- We used transformations of the input x

$$\phi\left(T_1(x)\right) = T_2(x) \quad \text{or} \quad \phi(T_1(x)) = p\left(T_2(x) \,|\, T_1(x)\right)$$

- Can we further generalize it?

Towards General Self-Supervised Learning

- We used transformations of the input x

$$\phi\left(T_1(x)\right) = T_2(x) \quad \text{or} \quad \phi(T_1(x)) = p\left(T_2(x) \,|\, T_1(x)\right)$$

- Can we further generalize it?

- We could relax the constraint to

$$G\left(\phi\left(T_1(x)\right) - T_2(x)\right) < G\left(\phi\left(T_1(x)\right) - T_2(z)\right)$$

with $G(w)$ some metric

Towards General Self-Supervised Learning

- We used transformations of the input x

$$\phi\left(T_1(x)\right) = T_2(x) \quad \text{or} \quad \phi(T_1(x)) = p\left(T_2(x) \,|\, T_1(x)\right)$$

- Can we further generalize it?

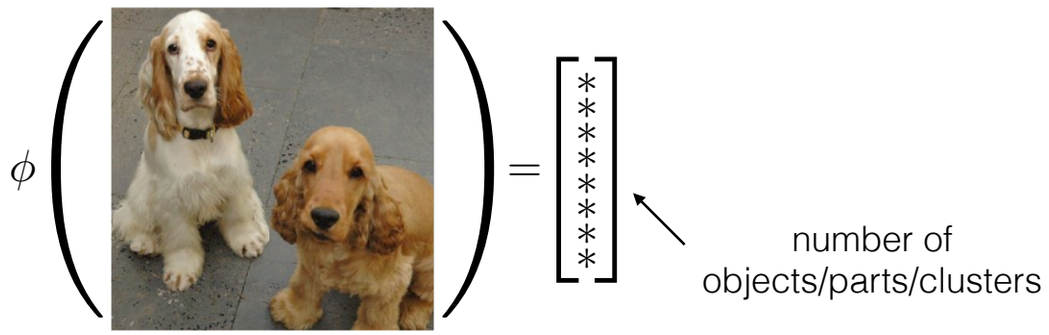
- We could relax the constraint to

$$G\left(\phi\left(T_1(x)\right) - T_2(x)\right) < G\left(\phi\left(T_1(x)\right) - T_2(z)\right)$$

with $G(w)$ some metric

Towards General Self-Supervised Learning

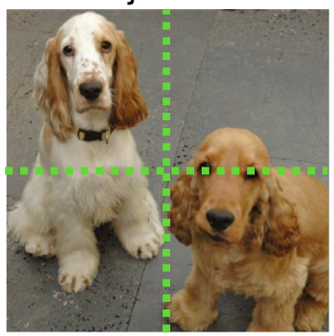
- Introduce **known relations** between the features of transformed images
- For example, let's build a model that learns to **count** objects (without specifying what objects are)



Define what feature transformations the model should satisfy given known image transformations.

Counting Objects

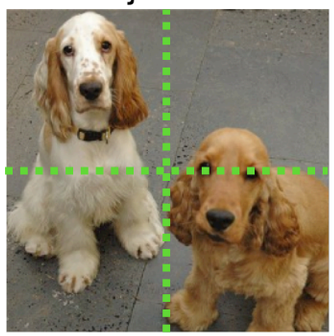
- If the feature counts objects and we split an image



Define what feature transformations the model should satisfy given known image transformations.

Counting 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 (downsampled) image

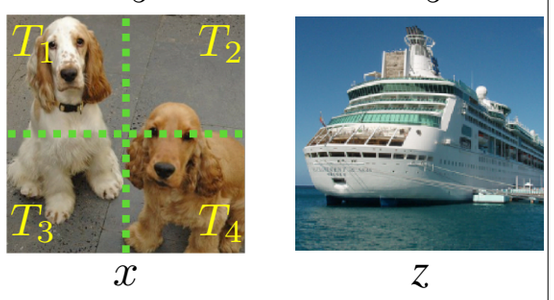
$$\phi\left(\text{}\right) + \phi\left(\text{}\right) + \phi\left(\text{}\right) + \phi\left(\text{}\right) = \phi\left(\text{}\right)$$

Define what feature transformations the model should satisfy given known image transformations.

A More General Self-Supervised Learning

- In general, the self-supervision signal is a **relation**

$$G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(x)\right)\right) < G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(z)\right)\right)$$



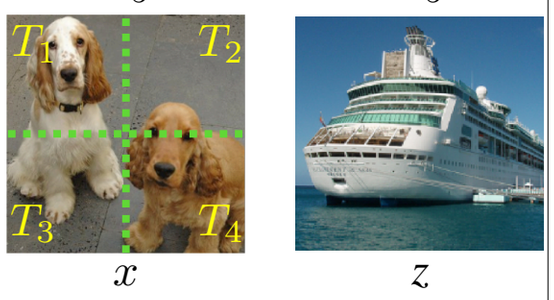
In this example the transformations are image cropping (T1 to T4) and downsampling (T5).
The relation is a sum of all the features from the croppings and a difference with the features from the downsampled image.

A More General Self-Supervised Learning

- In general, the self-supervision signal is a **relation**

$$G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(x)\right)\right) < G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(z)\right)\right)$$

- Define a set of **known** transformations $T_1, \ldots, T_k$ applied to the input data (x)



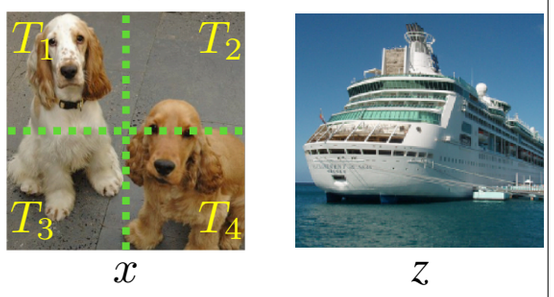
In this example the transformations are image cropping (T1 to T4) and downsampling (T5).
The relation is a sum of all the features from the croppings and a difference with the features from the downsampled image.

A More General Self-Supervised Learning

- In general, the self-supervision signal is a **relation**

$$G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(x)\right)\right) < G\left(\phi\left(T_1(x)\right),\ldots,\phi\left(T_k(z)\right)\right)$$

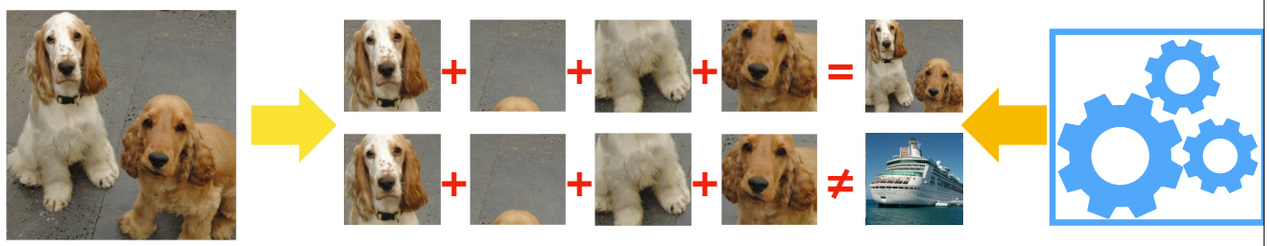
- Define a set of **known** transformations $T_1, \ldots, T_k$ applied to the input data (x)
- The relation G is **known**



In this example the transformations are image cropping (T1 to T4) and downsampling (T5).
The relation is a sum of all the features from the croppings and a difference with the features from the downsampled image.

Counting Objects

- **Example #7:** Learning to count objects



1. image **2. coherence/incoherence** **3. learning**

Representation learning by learning to count
M. Noroozi, H. Pirsiavash, and P. Favaro, ICCV 2017

Towards General Self-Supervised Learning

- What transformations $T_1, \ldots, T_k$ and relations G achieve the best learned representation?
- It is unclear how to define them explicitly
- **Learn** both the transformations $T_1, \ldots, T_k$ and the relation G, and consider **inequalities** as constraints

Towards General Self-Supervised Learning

- **Two transformations**:
  - an identity transformation (e.g., an AE)
  - a learned transformation (T), which drops only high-level redundancy about x
- **Learn** G, the probability of real data

$$G(T(x)) < G(x)$$

Dropping Information

real

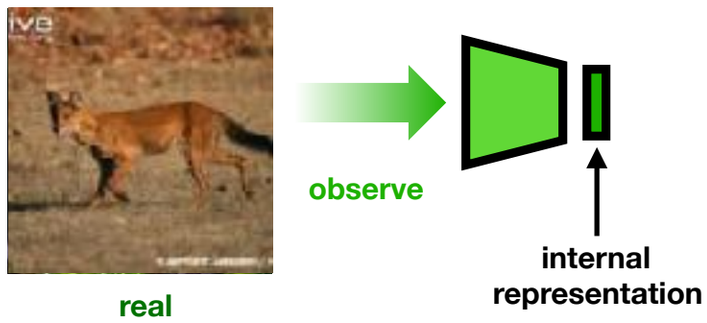
Dropping Information

- Let us consider a model that observes and synthesizes data

real

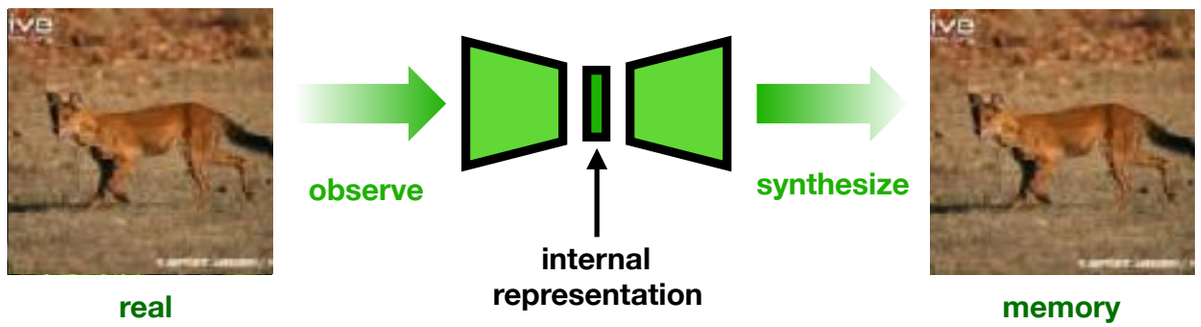
Dropping Information

- Let us consider a model that observes and synthesizes data



Dropping Information

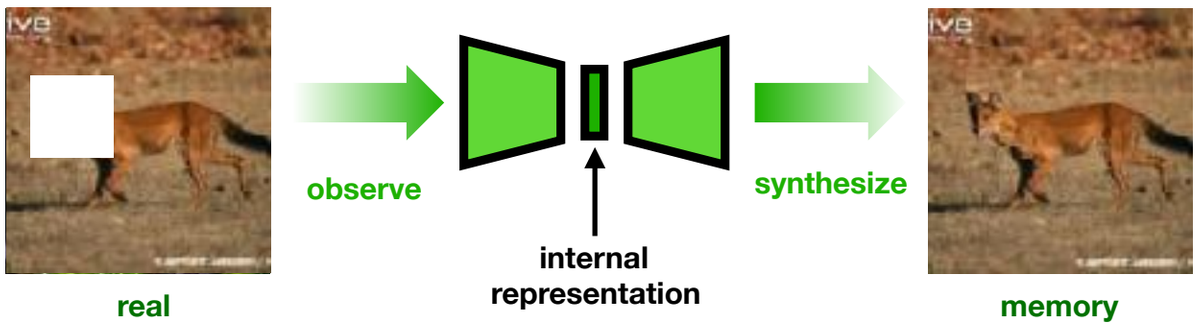
- Let us consider a model that observes and synthesizes data



Dropping Information

- Let us consider a model that observes and synthesizes data

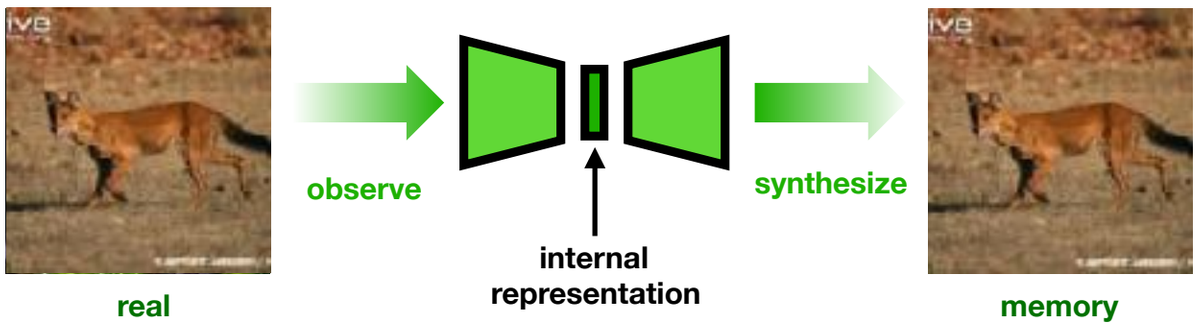
Instead of dropping
part of the data …



Dropping Information

- Let us consider a model that observes and synthesizes data

Instead of dropping
part of the data …

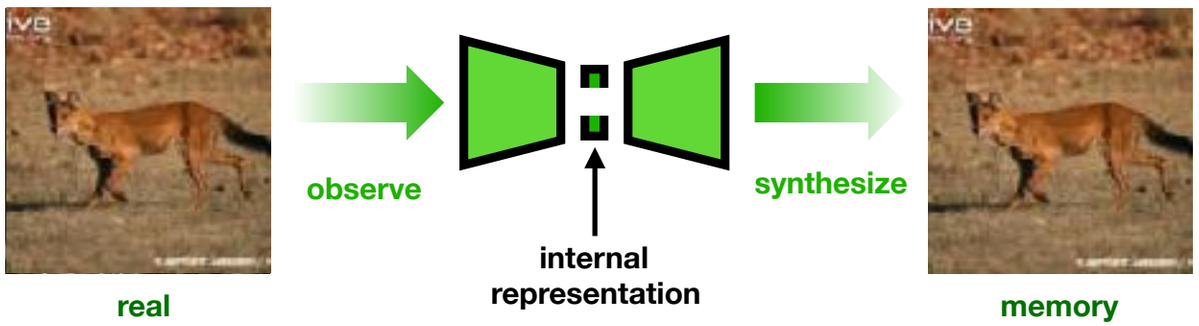


Dropping Information

- Let us consider a model that observes and synthesizes data

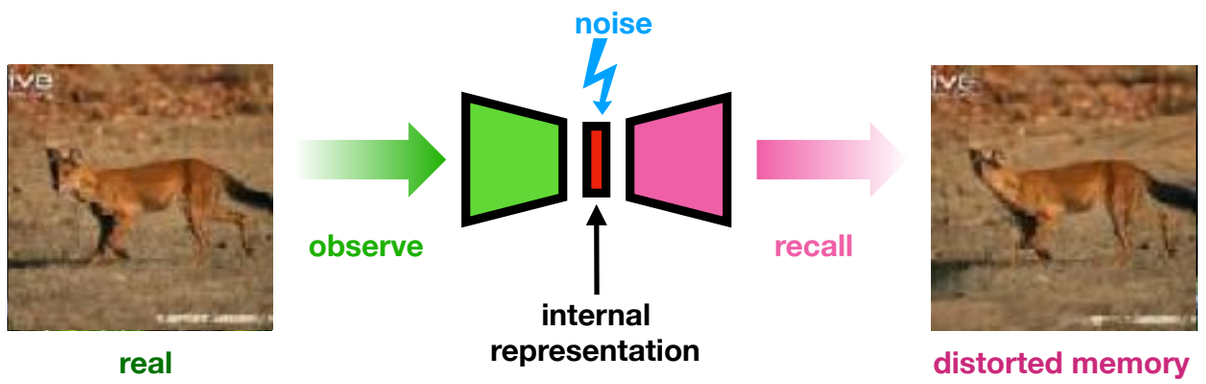
Instead of dropping part of the data …

… we could drop part of the **internal representation**



Using Noise to Drop Information

- Let us consider a model that observes and synthesizes data
- **Dropping information by injecting noise**: The model forgets some details and then recalls a distorted image



Distinguishing Reality from Dreams

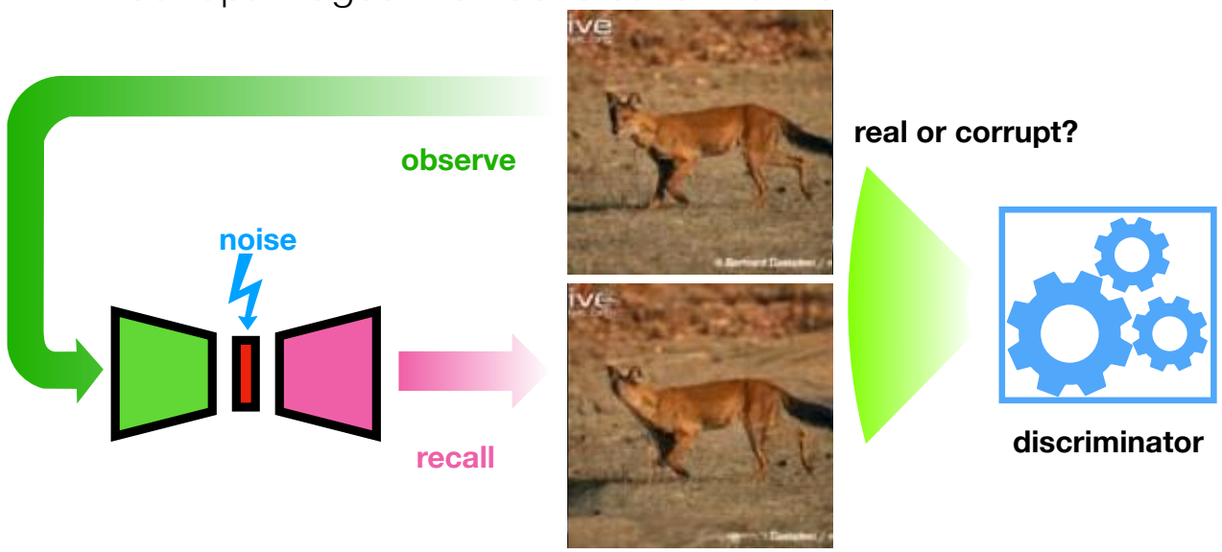
- A second model (the discriminator) learns to distinguish **real** from **corrupt** images

real

corrupt

Learning to Detect Artifacts

- **Example #8:** A discriminator learns to distinguish real from corrupt images in an adversarial manner

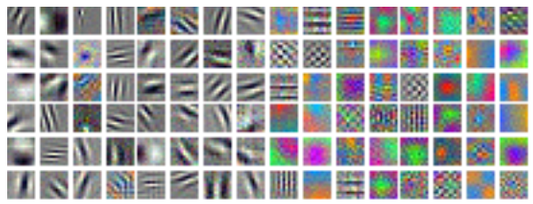


Self-Supervised Feature Learning by Learning to Spot Artifacts
S. Jenni and P. Favaro, CVPR 2018

Evaluation of Self-Supervised Learning Methods

Learned AlexNet CONV1 Filters

random init + cluster transfer

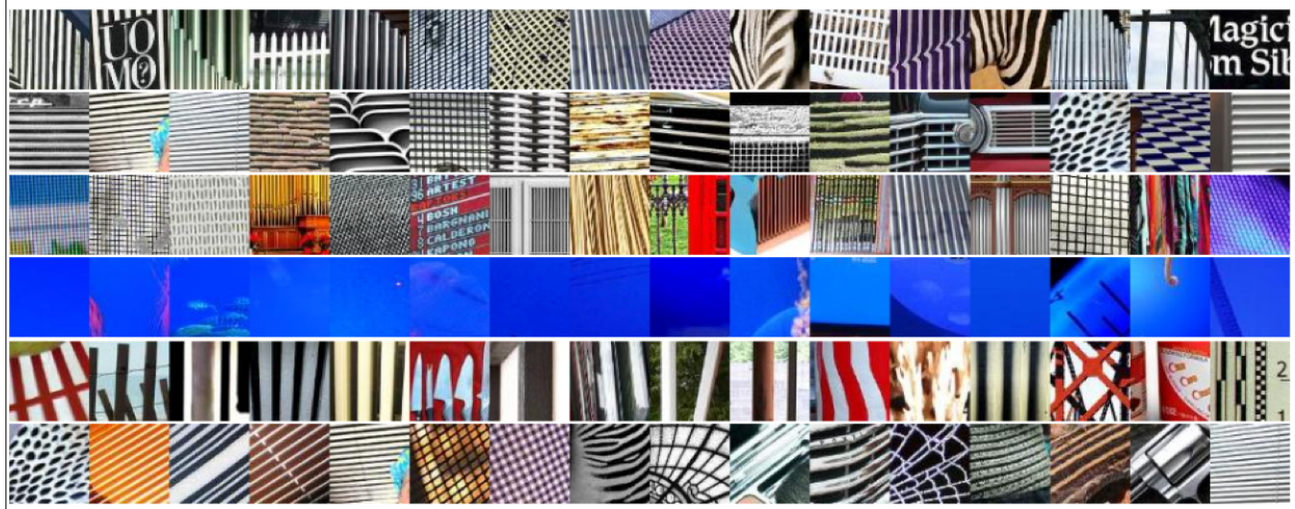


spotting artifacts

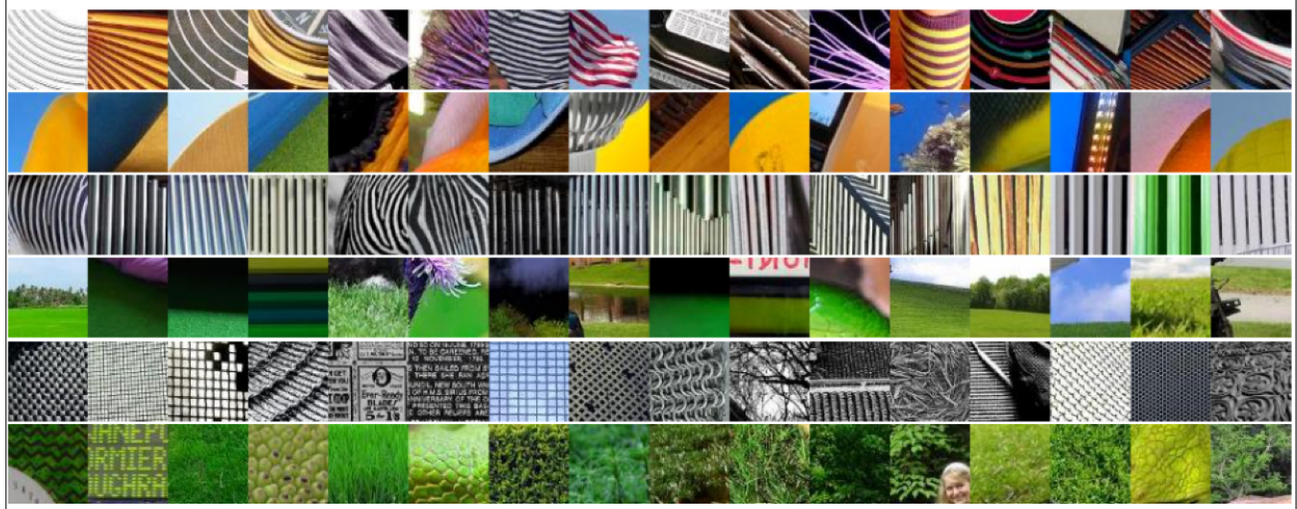
HOG + cluster transfer

context + cluster transfer

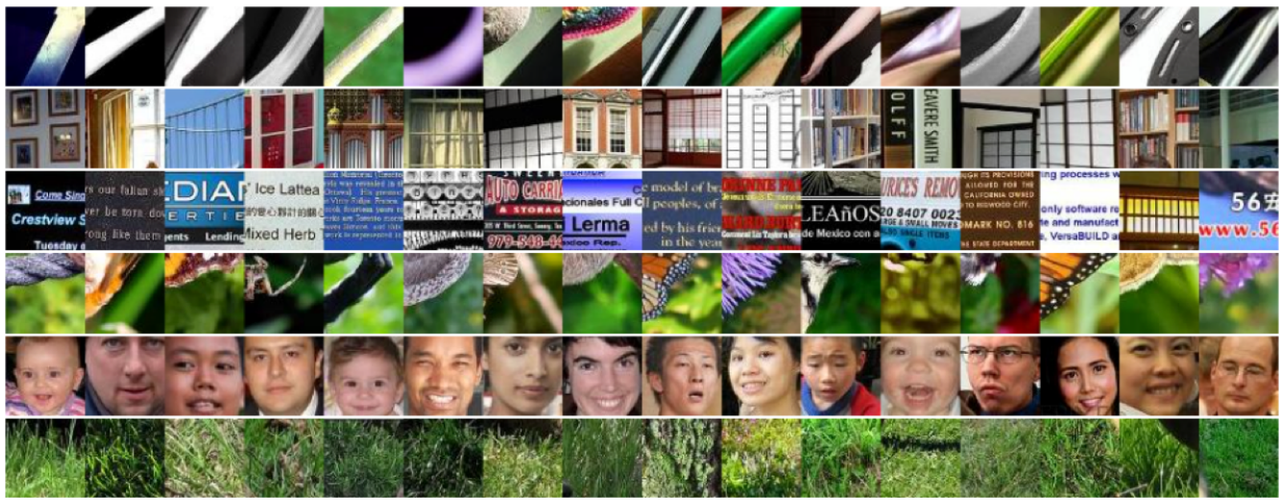
Jigsaw CONV1
Strongest Responses



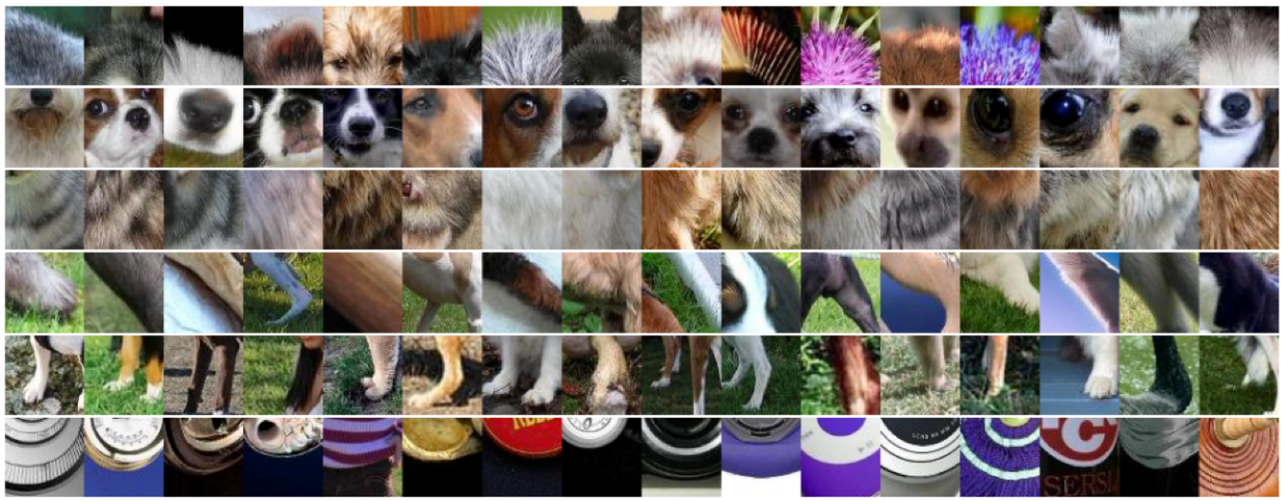
Jigsaw CONV2
Strongest Responses



Jigsaw CONV3
Strongest Responses



Jigsaw CONV4
Strongest Responses



Jigsaw CONV5
Strongest Responses

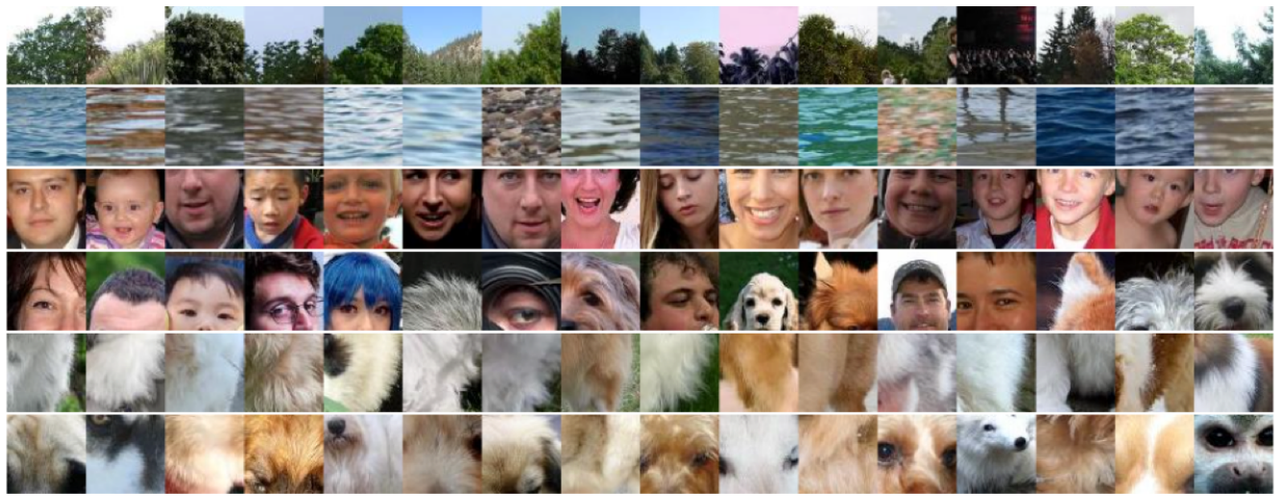


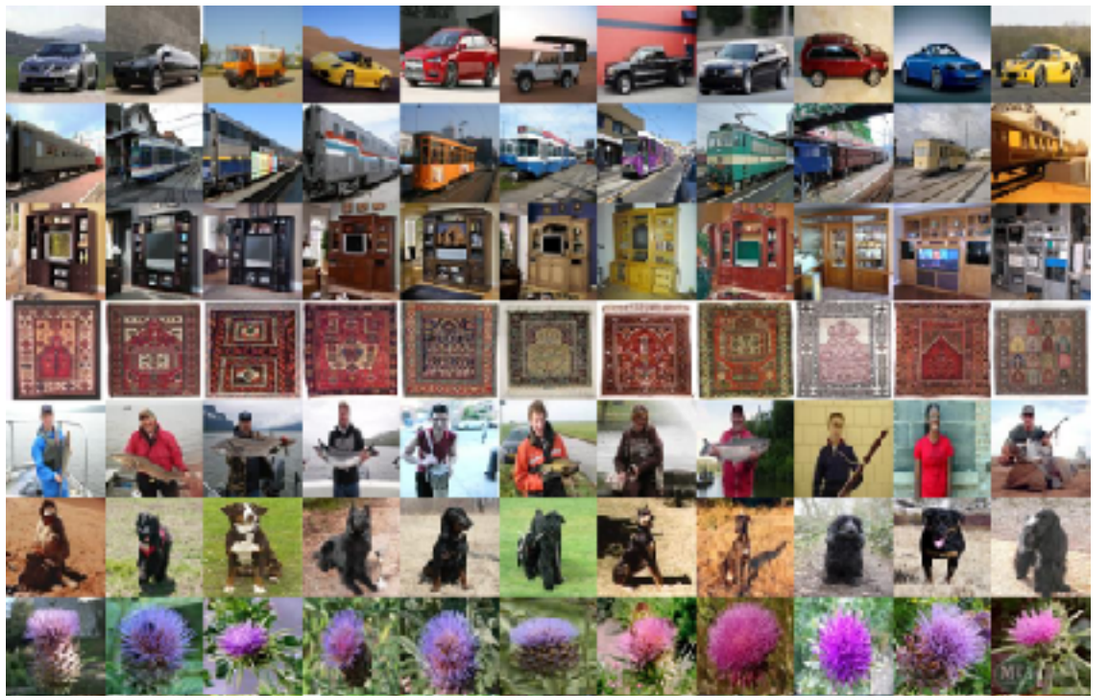
Image Retrieval Evaluation

- To see what feature space we have learned, we can use image retrieval
- We compute features for a whole dataset of images (eg 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

Spotting Artifacts — NN

The effectiveness of these methods can be measured quantitatively and at the moment our group has developed algorithms that allow us to get closer to the performance of supervised learning.
Perhaps it is more compelling to show batches of images that one of these algorithms considers similar.
Let's remember that these algorithms were not told anything about the content of the images.

Jigsaw++ — Image Clusters



Quantitative Evaluations

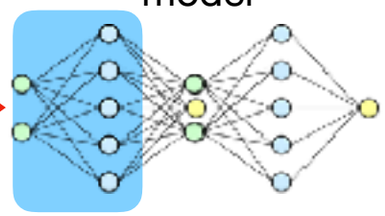
- Two quantitative evaluations
  - Transfer the knowledge learned with a SSL task via fine-tuning to Pascal VOC on the classification, detection and segmentation tasks
  - Freeze features and train a linear* classifier on top of them

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

Fine-Tuning on Pascal VOC

IMAGENET without labels

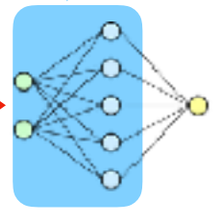
model



self-supervised learning pretext task

knowledge transfer

PASCAL VOC with labels



urban

industrial

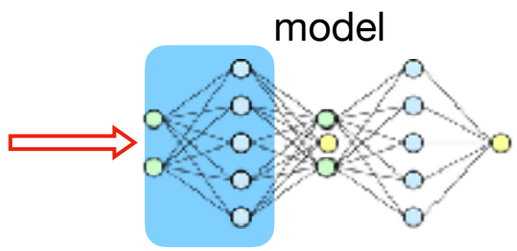
mountain

savannah

attributes

Fine-Tuning on Pascal VOC

IMAGENET without labels



self-supervised learning pretext task

knowledge transfer

PASCAL VOC with labels

urban

industrial

mountain

savannah

attributes

Fine-Tuning on Pascal VOC

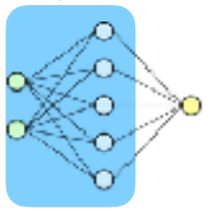
IMAGENET without labels

model

self-supervised
learning
pretext task

knowledge transfer

PASCAL VOC with labels



urban

industrial

mountain

savannah

attributes

Pascal VOC Transfer via Fine-Tuning

tasks

Model	Classification (mAP)	Detection (mAP)	Segmentation (mIU)
Krizhevsky *et al.*	79.9%	56.8%	48.0%
Random	53.3%	43.4%	19.8%
Agrawal *et al.*	54.2%	43.9%	-
Bojanowski *et al.*	65.3%	49.4%	-
Doersch *et al.*	65.3%	51.1%	-
Donahue *et al.*	60.1%	46.9%	35.2%
Jayaraman & Grauman	-	41.7%	-
Krähenbühl *et al.*	56.6%	45.6%	32.6%
Larsson *et al.*	65.9%	-	38.0%
Noroozi & Favaro	67.6%	**53.2%**	37.6%
Noroozi *et al.*	67.7%	51.4%	36.6%
Owens *et al.*	61.3%	44.0%	-
Pathak *et al.*	56.5%	44.5%	29.7%
Pathak *et al.*	61.0%	52.2%	-
Wang & Gupta	63.1%	47.4%	-
Zhang *et al.*	65.9%	46.9%	35.6%
Zhang *et al.*	67.1%	46.7%	36.0%
Jenni & Favaro	**69.8%**	52.5%	**38.1%**

AlexNet transfer

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

AlexNet transfer

Method	Class.	Det.		Segm.
		SS	MS	
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop [voc]	67.9	52.8	53.4	-
Context-ColorProjection [voc]	66.7	51.5	51.8	-
Context [voc]	68.0	53.0	53.5	-
HOG [voc]	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++ [voc]	69.9	55.0	55.8	40.0
Jigsaw++ [VOC]	**72.5**	**56.5**	**57.4**	**42.6**

tasks

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

AlexNet transfer

Method	Class.	Det. (tasks)		Segm.
		SS	MS	
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop	67.9	52.8	53.4	-
Context-ColorProjection	66.7	51.5	51.8	-
Context	68.0	53.0	53.5	-
HOG	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++	69.9	55.0	55.8	40.0
Jigsaw++	**72.5**	**56.5**	**57.4**	**42.6**

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

AlexNet transfer

tasks

Method	Class.	Det. SS	Det. MS	Segm.
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop [ICCV]	67.9	52.8	53.4	-
Context-ColorProjection [ICCV]	66.7	51.5	51.8	-
Context [ICCV]	68.0	53.0	53.5	-
HOG [PC]	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++[ICCV]	69.9	55.0	55.8	40.0
Jigsaw++[VOC]	**72.5**	**56.5**	**57.4**	**42.6**

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

AlexNet transfer

Method	Class.	Det. (tasks)		Segm.
		SS	MS	
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop [voc]	67.9	52.8	53.4	-
Context-ColorProjection [voc]	66.7	51.5	51.8	-
Context [voc]	68.0	53.0	53.5	-
HOG [voc]	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++ [voc]	69.9	55.0	55.8	40.0
Jigsaw++ [VOC]	**72.5**	**56.5**	**57.4**	**42.6**

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

AlexNet transfer

tasks

Method	Class.	Det. SS	Det. MS	Segm.
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop [voc]	67.9	52.8	53.4	-
Context-ColorProjection [voc]	66.7	51.5	51.8	-
Context [voc]	68.0	53.0	53.5	-
HOG [voc]	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++ [voc]	69.9	55.0	55.8	40.0
Jigsaw++ [VOC]	**72.5**	**56.5**	**57.4**	**42.6**

SSL on ImageNet and tasks on the Pascal VOC dataset

Pascal VOC Transfer via Fine-Tuning

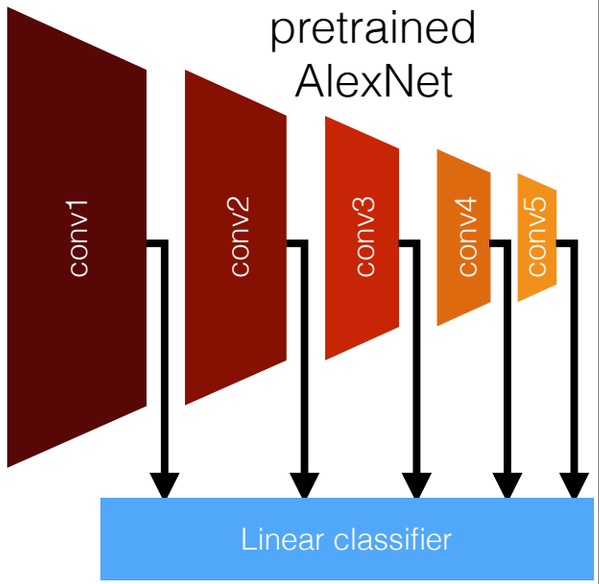
AlexNet transfer

Method	Class.	Det. (tasks)		Segm.
		SS	MS	
Supervised	79.9	59.1	59.9	48.0
Random	53.3	43.4	-	19.8
ego-motion	54.2	43.9	-	-
BiGAN	58.6	46.2	-	34.9
ContextEncoder	56.5	44.5	-	29.7
Context-ColorDrop [voc]	67.9	52.8	53.4	-
Context-ColorProjection [voc]	66.7	51.5	51.8	-
Context [voc]	68.0	53.0	53.5	-
HOG [voc]	70.2	53.2	53.5	39.2
Jigsaw	67.7	53.2	-	-
Jigsaw++	69.8	55.5	55.7	38.1
Jigsaw++ [voc]	69.9	55.0	55.8	40.0
Jigsaw++ [VOC]	**72.5**	**56.5**	**57.4**	**42.6**

SSL on ImageNet and tasks on the Pascal VOC dataset

Transfer Learning via a Linear Classifier

- After 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



Transfer Learning via Linear Classifiers

SSL on ImageNet and classification on **ImageNet**

Method	conv1	conv2	conv3	conv4	conv5
Supervised	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Context	16.2	23.3	30.2	31.7	29.6
ContextEncoder	14.1	20.7	21.0	19.8	15.5
BiGAN	17.7	24.5	31.0	29.9	28.0
Colorization	12.5	24.5	30.4	31.5	30.3
Split-Brain	17.7	29.3	35.4	35.2	32.8
Counting	18.0	30.6	34.3	32.5	25.7
HOG [ee]	16.8	27.4	20.7	32.0	29.1
Jigsaw++	18.2	28.7	34.1	33.2	28.0
Jigsaw++[a,c,e]	18.9	30.5	35.7	35.4	32.2
Jigsaw++[t,c,c]	**19.2**	**32.0**	**37.3**	**37.1**	**34.6**
Spotting artifacts	**19.5**	**33.3**	**37.9**	**38.9**	**34.9**

Transfer Learning via Linear Classifiers

SSL on **ImageNet** and classification on **Places**

	Method	conv1	conv2	conv3	conv4	conv5
supervised	Places labels	22.1	35.1	40.2	43.3	44.6
	ImageNet labels	22.7	34.8	38.4	39.4	38.7
	Random	15.7	20.3	19.8	19.1	17.5
SSL	Spotting artifacts	**23.3**	**34.3**	**36.9**	**37.3**	**34.4**
	Jigsaw	23.0	31.9	35.0	34.2	29.3
	Context encoder	18.2	23.2	23.4	21.9	18.4
	Sound	19.9	29.3	32.1	28.8	29.8
	BiGAN	22.0	28.7	31.8	31.3	29.7
	Colorization	16.0	25.7	29.6	30.3	29.7
	Split-Brain	21.3	30.7	34.0	34.1	32.5
	Counting	**23.3**	33.9	36.3	34.7	29.6
	HOG [cc]	20.3	30.0	31.8	32.5	29.8
	Jigsaw++	22.0	31.2	34.3	33.9	22.9
	Jigsaw++^{aoc}	22.5	33.0	36.2	36.1	34.0
	Jigsaw++^{voc}	22.9	**34.2**	**37.5**	**37.1**	**34.4**

Data Redundancy

Learning Useful Data Redundancy

- Our target tasks need features related to objects

Learning Useful Data Redundancy

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

Learning Useful Data Redundancy

- Our target tasks need features related to objects
- Is all the data redundancy useful to the target tasks?
  - Some redundancy is about low-level statistics (such as edges and corners)

Learning Useful Data Redundancy

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

Learning Useful Redundancy

- 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

Recall Context Prediction

- **Example #1**
- Drop the relative location of two tiles

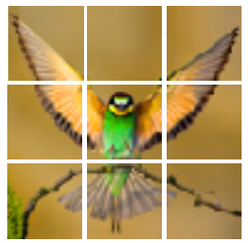
$$x_i = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$

1. image **2. tiles** **3. learning** **4. location**

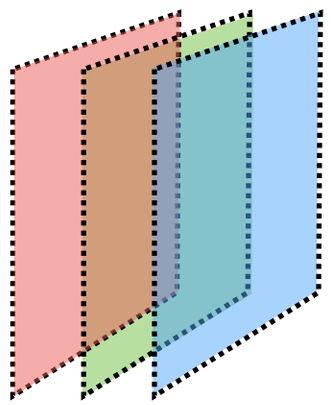
Unsupervised Visual Representation Learning by Context Prediction.
C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

Recall Context Prediction

- What redundancy is needed to solve the task?
  - Geometric information (location)
- Possible dependencies
  - **No need to look at other tiles**:
    The absolute location (relative to the image) is embedded in each tile
  - **Need to look at other tiles**: The relative location is obtained through tile comparisons

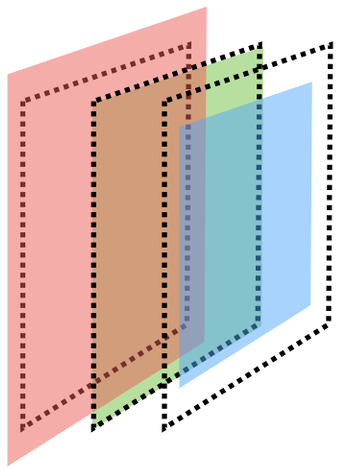


Chromatic Aberration



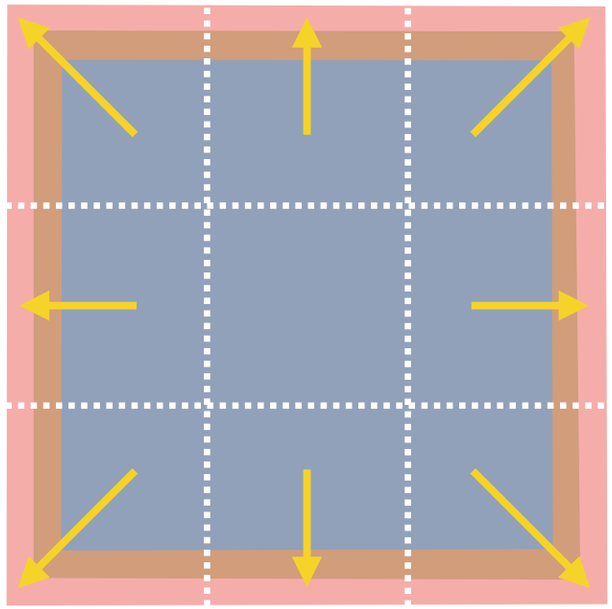
Chromatic aberration introduces a relative shift (or scaling or blur) between the color channels. This artifact changes relative to the image center.

Chromatic Aberration



Chromatic aberration introduces a relative shift (or scaling or blur) between the color channels. This artifact changes relative to the image center.

Chromatic Aberration



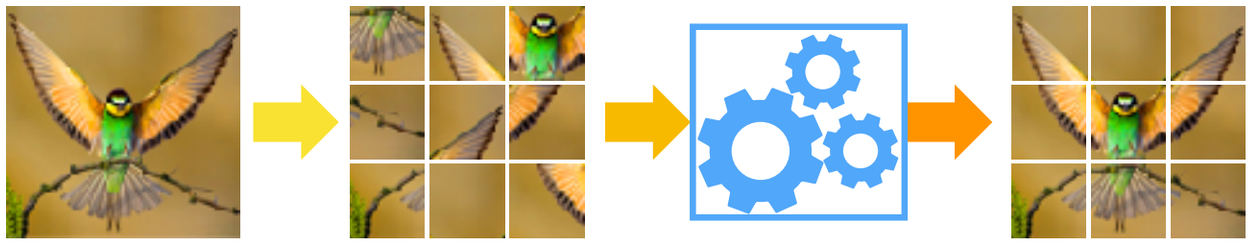
chromatic aberration encodes absolute location

Chromatic aberration introduces a relative shift (or scaling or blur) between the color channels. This artifact changes relative to the image center.

Recall Jigsaw Puzzles

- **Example #2**
- Drop the ordering of all the puzzle tiles

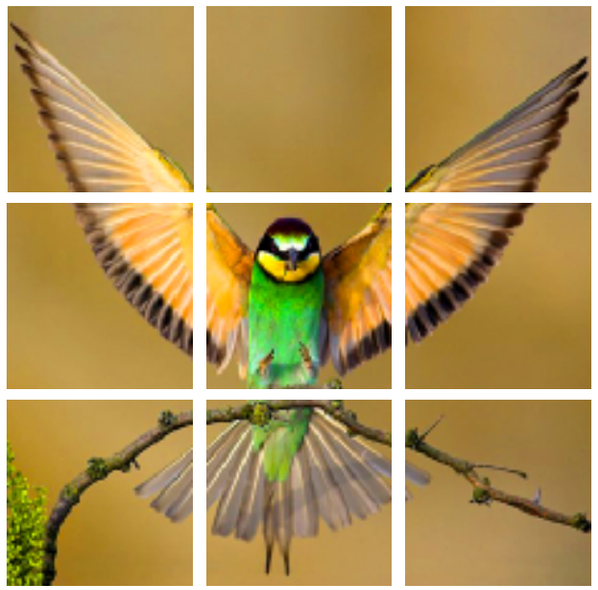
$$x_i = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$



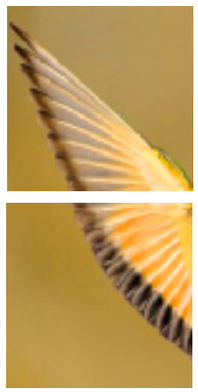
1. image **2. puzzle** **3. learning** **4. ordered**

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

Comparing Tiles

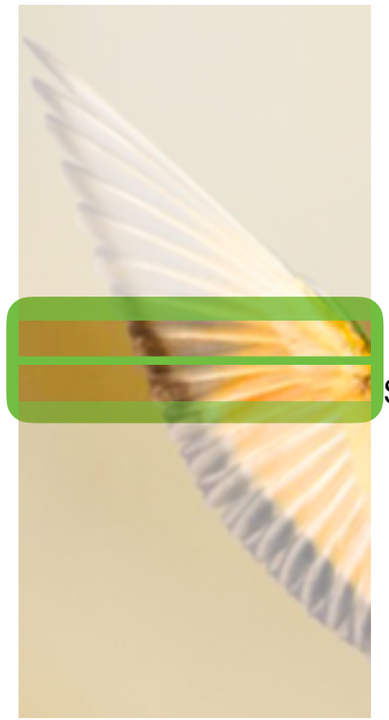


Comparing Tiles



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

Comparing Tiles



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

Comparing Tiles

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

Recall Counting Objects

- We train a model to count objects by enforcing

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

Define what feature transformations the model should satisfy given known image transformations.

Recall Counting Objects

- We train a model to count objects by enforcing

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

- If the model can distinguish tiles from downsampled images, then it could learn two different families of features

Define what feature transformations the model should satisfy given known image transformations.

Recall Counting Objects

- We train a model to count objects by enforcing

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

- If the model can distinguish tiles from downsampled images, then it could learn two different families of features

- How can it tell the difference?

Define what feature transformations the model should satisfy given known image transformations.

Recall Counting Objects

- We train a model to count objects by enforcing

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

- If the model can distinguish tiles from downsampled images, then it could learn two different families of features
- How can it tell the difference?
  - Downsampling method leaves a **footprint**

Define what feature transformations the model should satisfy given known image transformations.

Recall Counting Objects

- We train a model to count objects by enforcing

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

- If the model can distinguish tiles from downsampled images, then it could learn two different families of features
- How can it tell the difference?
  - Downsampling method leaves a **footprint**
  - Less noise in the downsampled image

Define what feature transformations the model should satisfy given known image transformations.

Recall Counting Objects

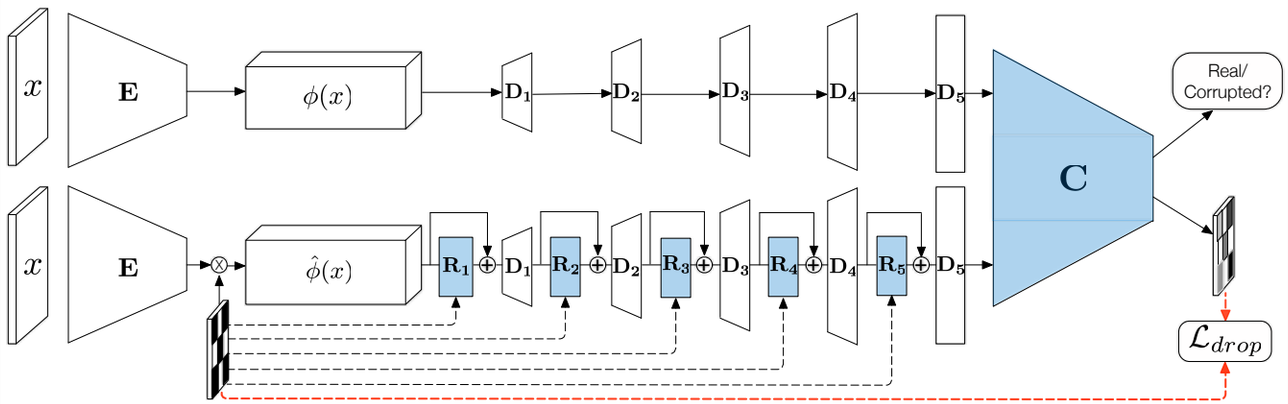
- We train a model to count objects by enforcing

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

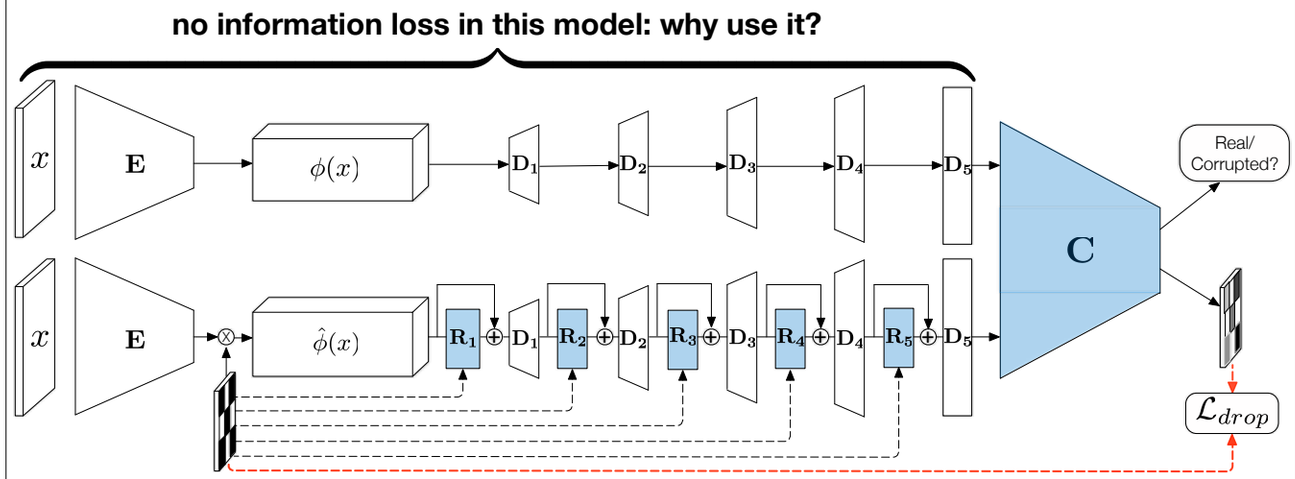
- If the model can distinguish tiles from downsampled images, then it could learn two different families of features
- How can it tell the difference?
  - Downsampling method leaves a **footprint**
  - Less noise in the downsampled image
  - Chromatic aberration (again)

Define what feature transformations the model should satisfy given known image transformations.

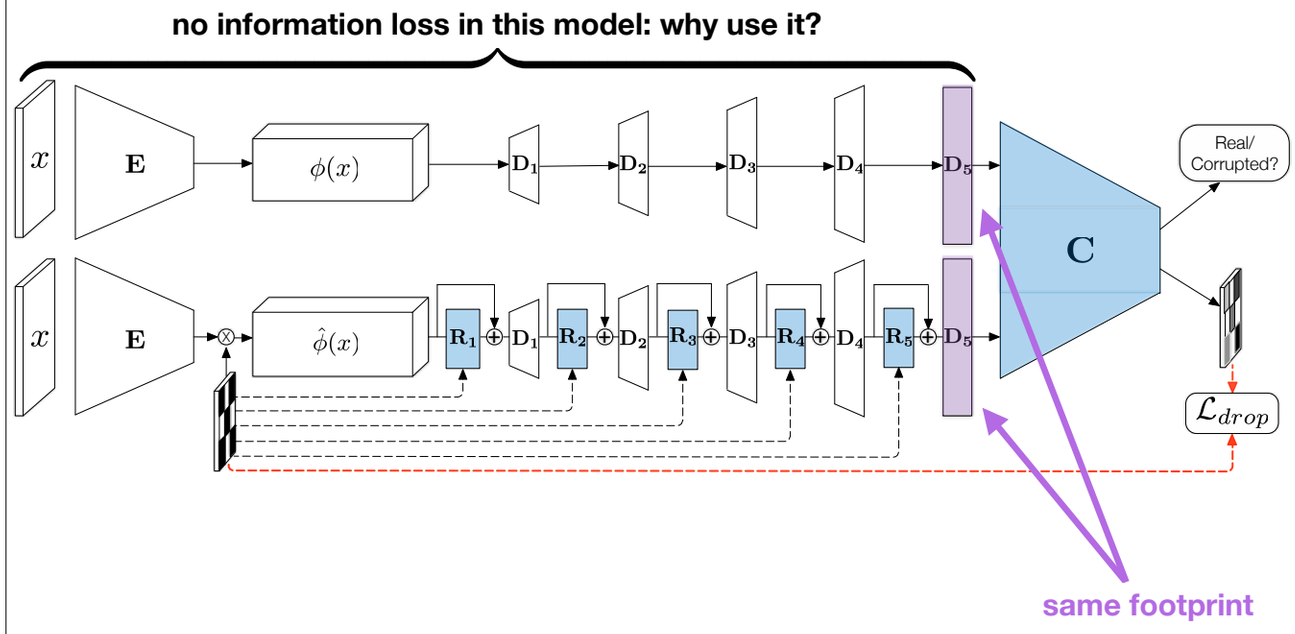
Recall Spotting Artifacts



Recall Spotting Artifacts



Recall Spotting Artifacts

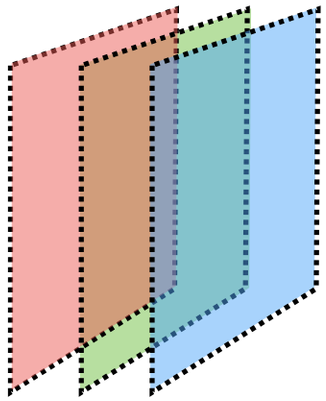


Learning Shortcuts

- These undesired learning behaviors are called **shortcuts**
- They are **specific** to each SSL task
  - E.g.: Context/puzzle (chrom. aber.), counting (downsampling, 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

Dealing with Shortcuts

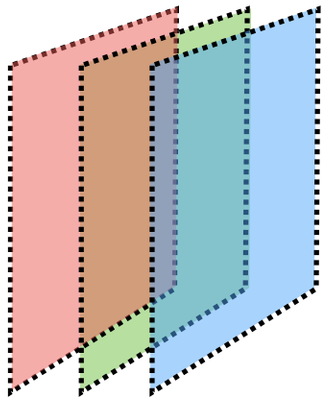
- The general strategy is to **drop the unwanted redundancy** (through noise) that allows solutions through shortcuts
- **Example**
  - To reduce chromatic aberration generate new images by randomly and independently scaling and shifting (by very small amounts) the color channels of each image
  - This will randomize chromatic aberration and not allow a model to learn from it



Note

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



Note

Part II
Generative Models for Unsupervised Learning