

Addressing Data Scarcity in Computer Vision

Professor: Bernard
Ghanem

Teacher Assistants:

- Carlos Hinojosa
- Fida Thoker

2024

Table of Contents



- | | | | |
|---|--------------------------|---|-----------------------|
| 1 | Imbalanced Learning | 5 | Unsupervised Learning |
| 2 | Semi-supervised learning | | |
| 3 | Few-shot Learning | | |
| 4 | Zero-shot Learning | | |

Recap: Supervised Learning



- The size of the dataset matters! The bigger the better.
- **What costs** when building a dataset for supervised learning?

Recap: Supervised Learning



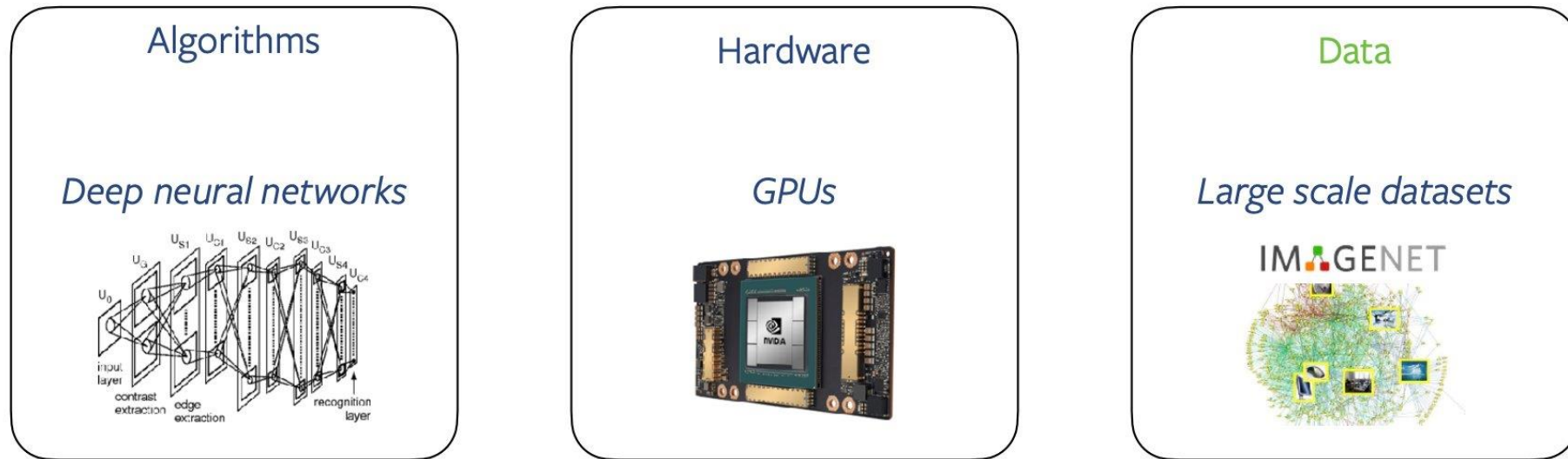
- The size of the dataset matters! The bigger the better.
- **What costs** when building a dataset for supervised learning?
- Usually semi-automatic labelling process.
- Still requires human intervention to check the labelling.

Do we really need all these labels?



- **Could we** remove, or at least **decrease**, the need for annotations?

Data Scarcity in AI

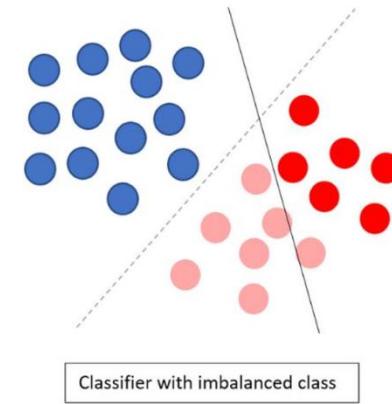
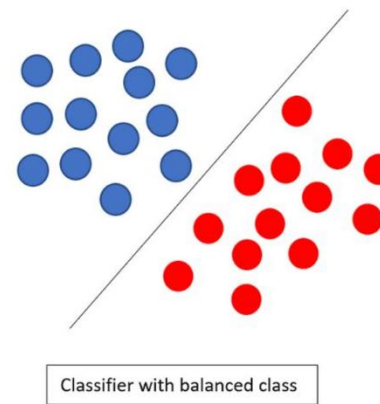


Data is expensive despite petabytes of data is generated everyday, only the few tech giants control the flow of it.

It is hard to obtain, and often insufficient, especially quality data with annotations

Imbalanced Learning

Imbalanced learning deals with datasets in which the distribution of classes is significantly imbalanced.



Imagine teaching a child to recognize animals by showing them a hundred pictures of dogs and just one of a cat.


Naturally, the child might start assuming that every four-legged furry creature is a dog.

This analogy mirrors the challenge in machine learning models trained on imbalanced datasets.

Such models are prone to bias, often mistaking rare/minority instances (the 'cats') for the prevalent/majority ones (the 'dogs').

Imbalanced Data - Techniques



- To get good predictive performance on imbalanced datasets, you can do the following
 - Data pre-processing. This modifies the dataset to reduce the imbalance before training the model.
 - Algorithm-specific. Depending on the algorithm used, weighting can be used to increase the penalty for mis-classifying a minority example.
 - Ensemble. By training multiple models and combining the results, performance can be increased over a single model. This is not just true for imbalanced datasets.
- 

Handling Imbalanced Data

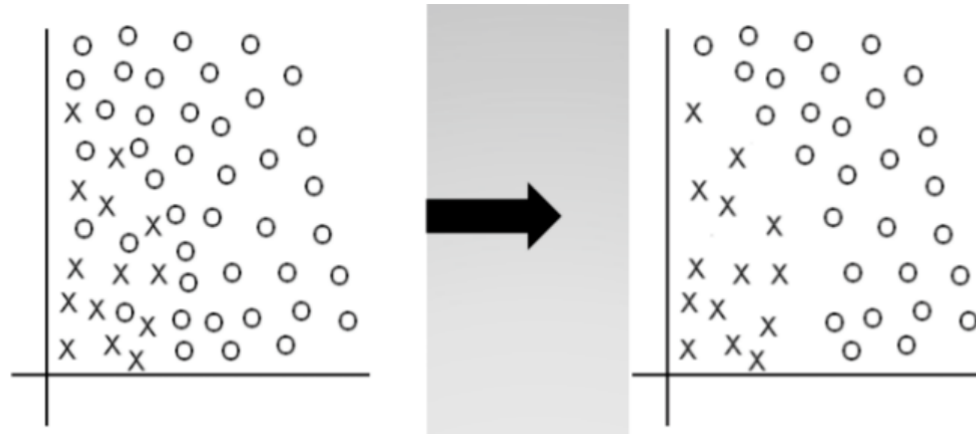
Data Resampling

- There are two options to generate a 50:50 balanced dataset from an imbalanced one
 - Oversampling: Generate new minority class examples.
 - Undersampling: Remove majority class examples.
- These can be done at random, but there are issues
 - Random oversampling gives issues with overfitting, as there will be many identical duplicated minority examples.
 - Random undersampling means you throw away good majority data.
- To improve on the performance of random under and oversampling, kNN-based methods selectively add or remove data.

Handling Imbalanced Data - Undersampling

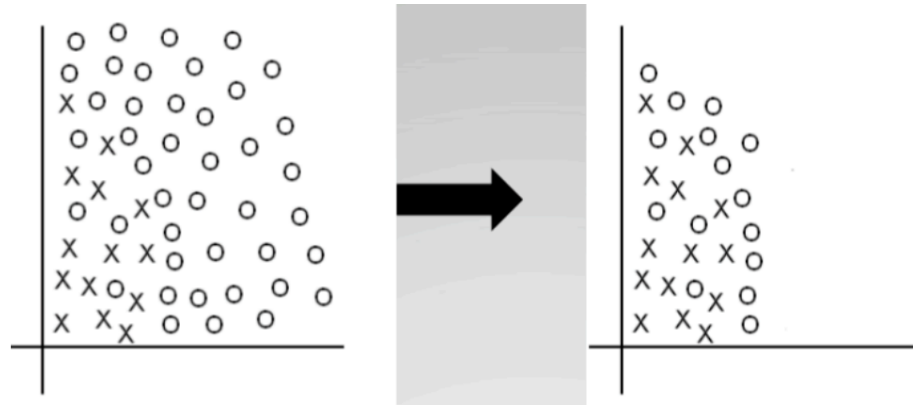
- Tomek Links

- Find pairs of points that are different classes, they form a Tomek link if there is no closer example to either point.
- Remove majority example in the pair.
- Effectively widens the decision boundary between majority and minority.



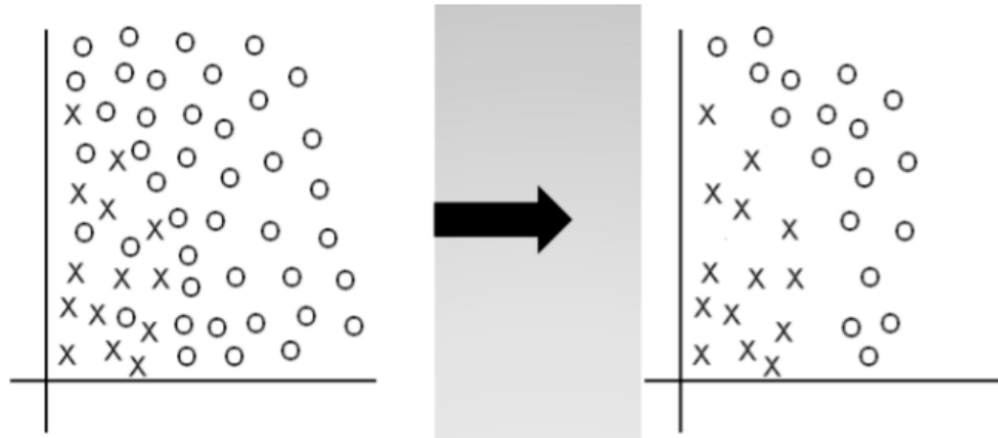
Handling Imbalanced Data - Undersampling

- **Condensed Nearest Neighbor (CNN)**
 - Removes all points, and adds them back in as required to correctly predict the examples with a kNN classification where $k=1$.
 - CNN removes majority class examples that are distant from the decision border.



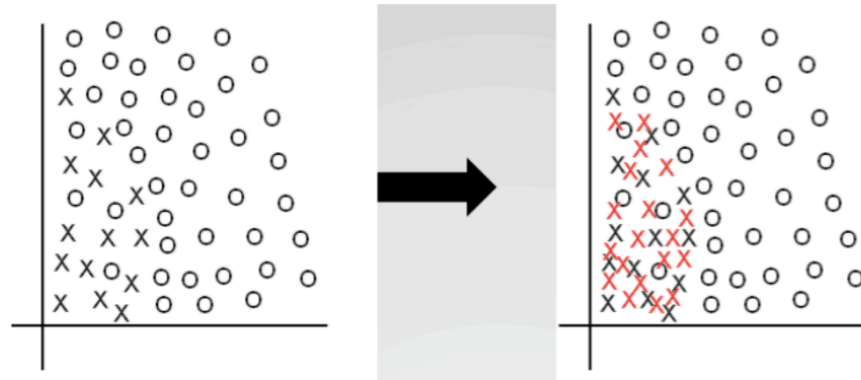
Handling Imbalanced Data - Undersampling

- **One-Sided Selection (Tomek links followed by Condensed Nearest Neighbor Rule)**
 - Removing majority-class examples from Tomek links increases the separation between majority and minority class feature spaces.
 - Condensed Nearest Neighbor Rule removes majority examples far from the class borderline.



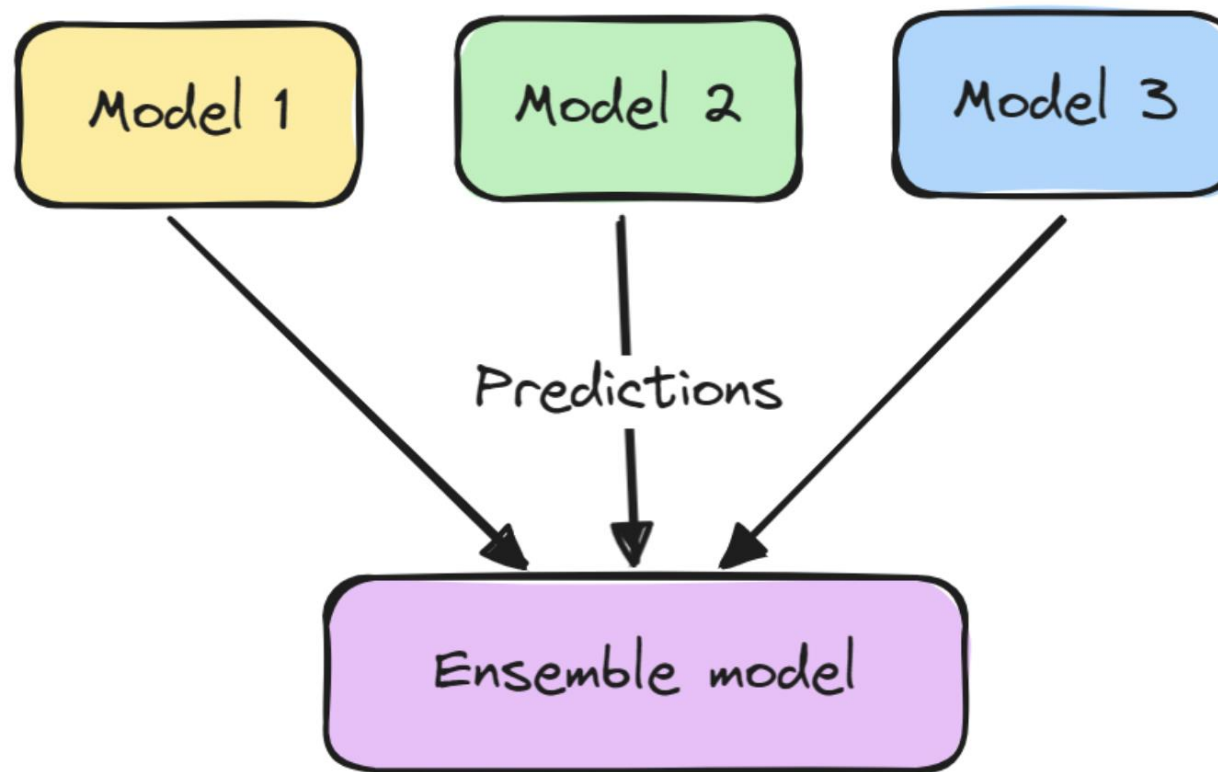
Handling Imbalanced Data - Oversampling

- Synthetic Minority Oversampling Technique (SMOTE)
- For each minority example
 - Randomly chose one of the k nearest neighbours (majority or minority)
 - Create a new sample a random distance between the minority example and the nearest neighbour.
 - This allows the minority example class space to be expanded.



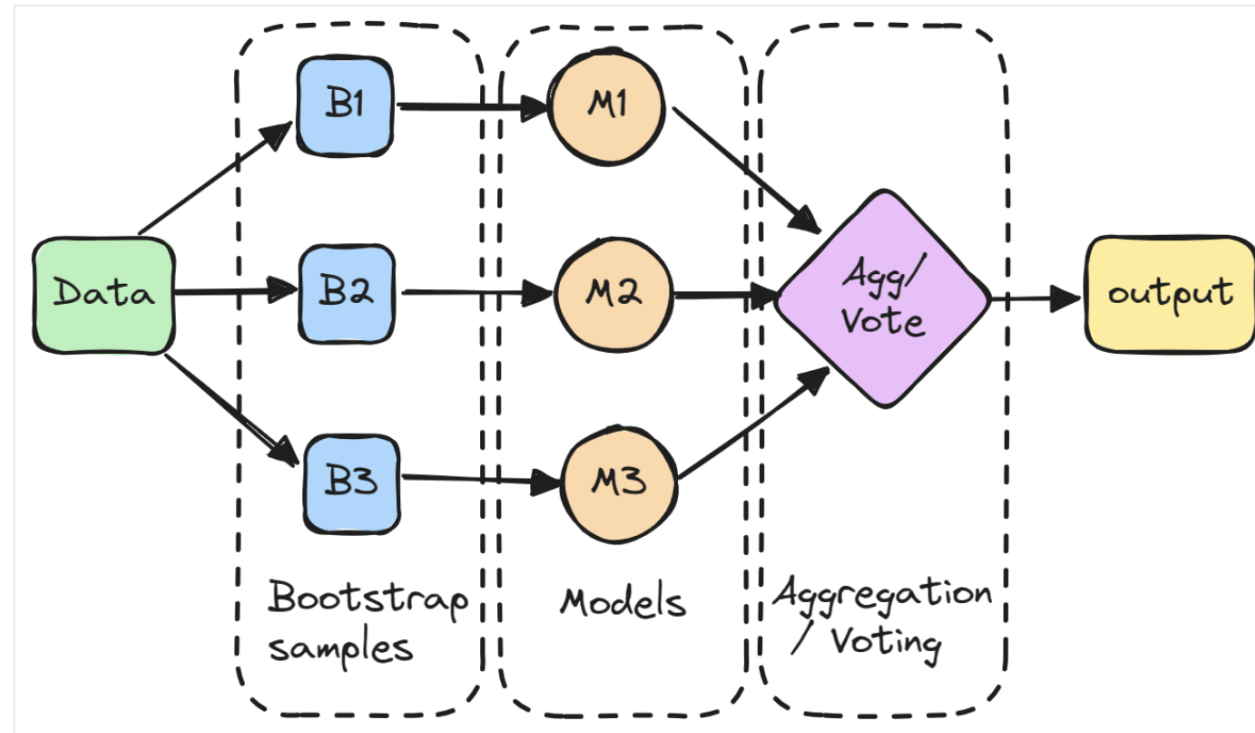
Handling Imbalanced Data – Ensemble Methods

Ensemble techniques combine multiple models to improve the overall performance of a classification task.



Handling Imbalanced Data – Ensemble Methods

Bagging



Bagging is an ensemble method that involves training multiple model independently on random subsets of the data, and aggregating their predictions through voting or averaging.

Handling Imbalanced Data – Ensemble Methods



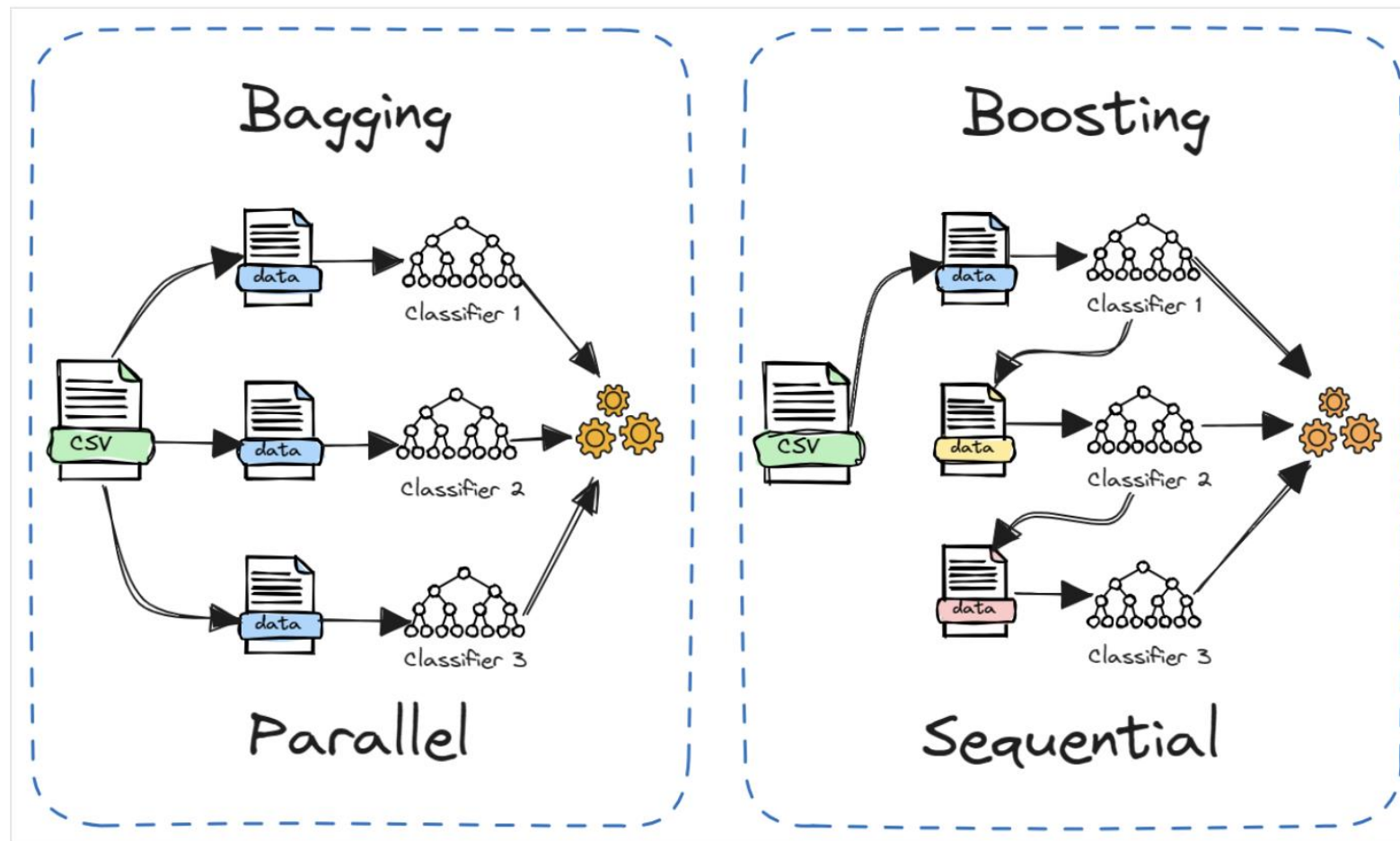
Boosting is another popular ensemble method.

- The main difference lies in how the constituent models are trained.
- In bagging, models are trained independently in parallel on different random subsets of the data.
- In boosting, models are trained sequentially, with each model learning from the errors of the previous one.



Handling Imbalanced Data – Ensemble Methods

Boosting is another popular ensemble method.



Semi-supervised learning



- Large dataset of images with some missing labels
- How to deal with missing labels?

Semi-supervised learning



- Large dataset of images with some missing labels
- How to deal with missing labels?
 - → Pseudo-labelling

Semi-supervised learning



- Large dataset of images with some missing labels
- How to deal with missing labels?
 - → Pseudo-labelling



All classes must have annotated images!

Pseudo-labels

Dataset of labelled images

$$D_1 \text{ (database icon)} = \left\{ \left[\boxed{I_1}, c_1 \right], \left[\boxed{I_3}, c_2 \right], \dots \right\}$$

Dataset of unlabelled images

$$D' \text{ (database icon)} = \left\{ \left[\boxed{I_2}, ? \right], \left[\boxed{I_4}, ? \right], \dots \right\}$$

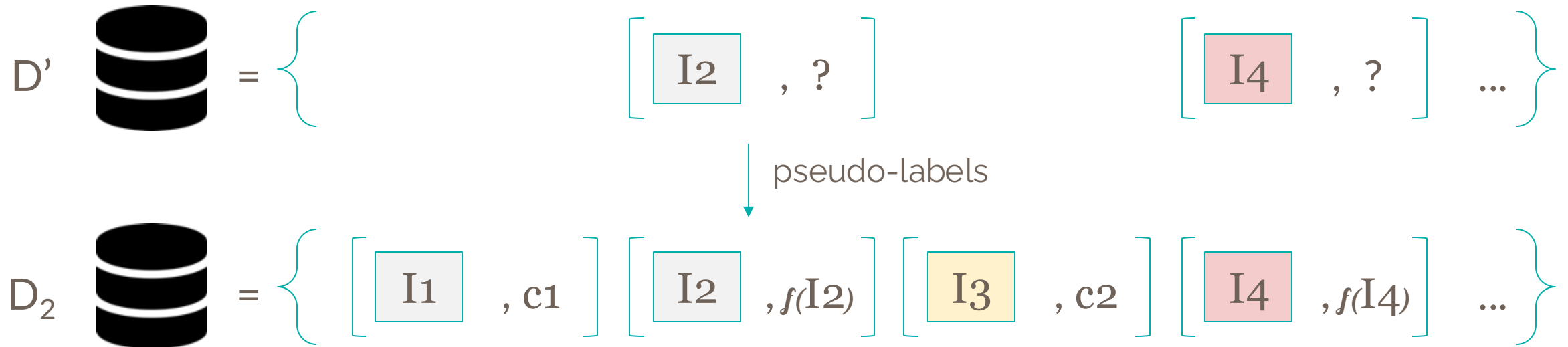
Pseudo-labels



- Iterative algorithm:
 - a. Train a classifier f on D_1



Pseudo-labels



- Iterative algorithm:
 - a. Train a classifier f on D_1
 - b. Pseudo-labelling: build dataset D_2 where we used f to label the images of D' (without labels)

Semi-supervised learning

$$D_3 \text{ (Database Icon)} = \left\{ \left[\boxed{I_1}, c_1 \right] \left[\boxed{I_2}, f_2(I_2) \right] \left[\boxed{I_3}, c_2 \right] \left[\boxed{I_4}, f_2(I_4) \right] \dots \right\}$$

- Iterative algorithm:
 - a. Train a classifier f on D_1
 - b. Pseudo-labelling: build dataset D_2 where we used f to label the images of D' (without labels)
 - c. Train another classifier f_2 on D_2
 - d. We can build D_3 where we use f_2 to label the images of D' → improve the pseudo-labelling
 - e. ...

When to use semi-supervised learning?

- Not all images are labelled
- **Every class must have some labelled images**
 - What to do if this is not the case? → zero-shot learning, unsupervised learning
- Still requires many labelled examples for each class to function properly.

When to use few-shot learning?

- Dataset completely annotated...

When to use few-shot learning?



- Dataset completely annotated..
- ...but all/some classes have very few training examples.



When to use few-shot learning?

- Dataset completely annotated..
- ...but all/some classes have very few training examples.
- Notation: when only **N** examples per class is available → N-shot learning
 - Examples: 5-shot learning, 1-shot learning

Few-shot Learning - Methods

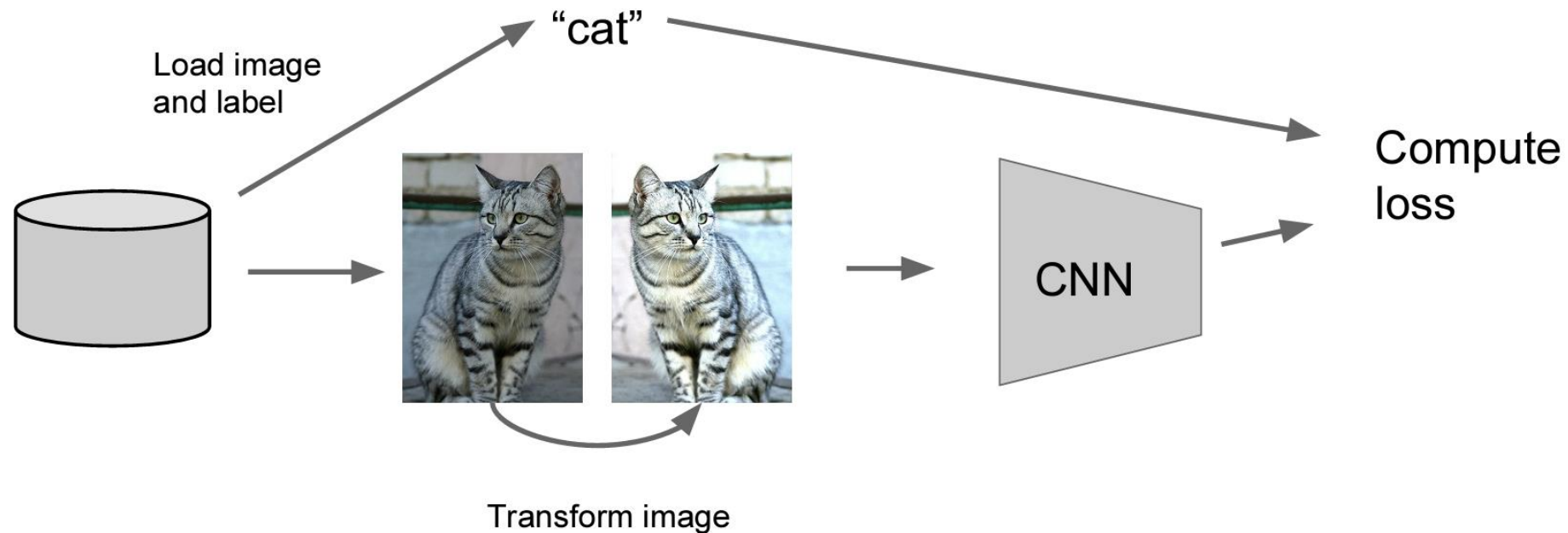


1. Data-level approach
2. Metric learning approach
3. *Parameter-level approach* (not covered in this course)



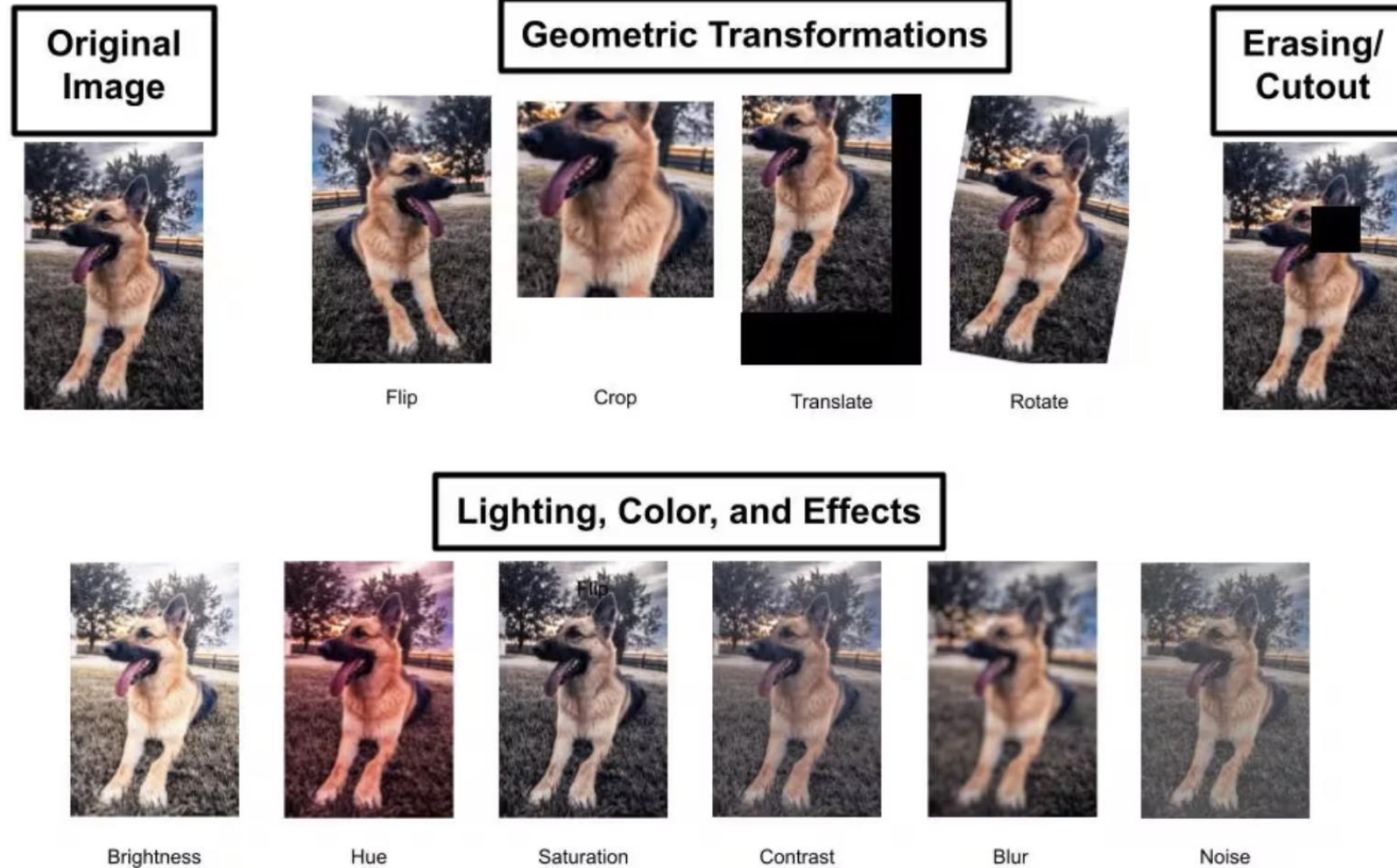
Data-level approach - Data Augmentation

- One image can yield many images!
- Artificially increase the dataset size by randomly transforming the images



Data Augmentation is a very powerful method to reduce overfitting by providing a more comprehensive set of possible data points to minimize the distance between the training and testing sets.

Data Augmentation - Examples



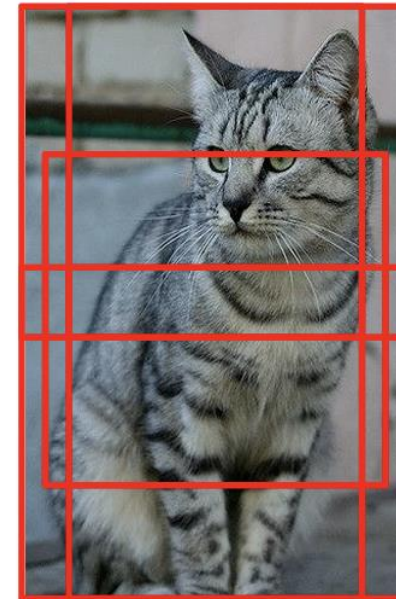
Data Augmentation

Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range $[256, 480]$
2. Resize training image, short side = L
3. Sample random 224×224 patch



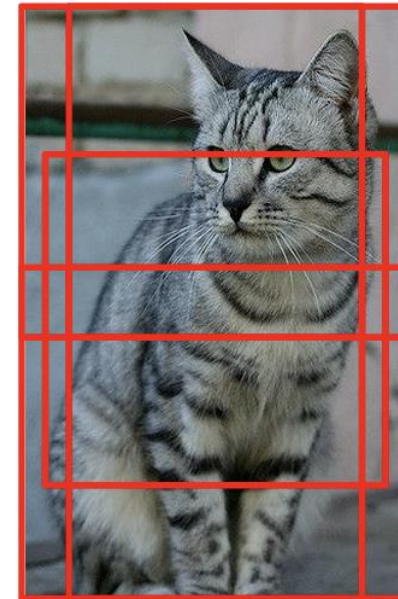
Data Augmentation

Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch



Testing: average a fixed set of crops

ResNet:

1. Resize image at 5 scales: {224, 256, 384, 480, 640}
2. For each size, use 10 224 x 224 crops: 4 corners + center, + flips

Data Augmentation

Color Jitter

Simple: Randomize
contrast and brightness

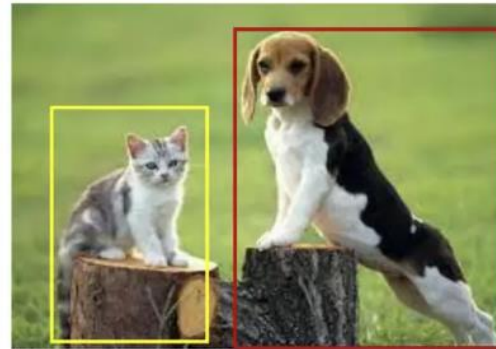


More Complex:

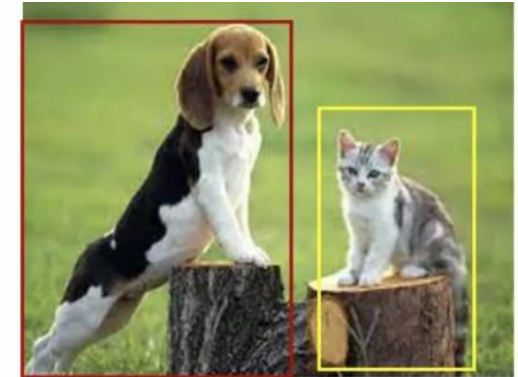
1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a “color offset” along principal component directions
3. Add offset to all pixels of a training image

Data Augmentation

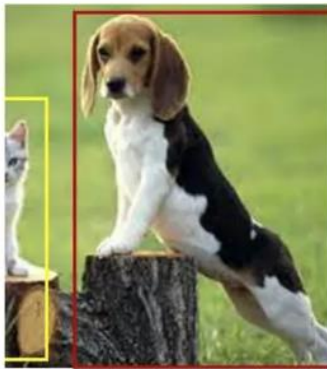
Object Detection



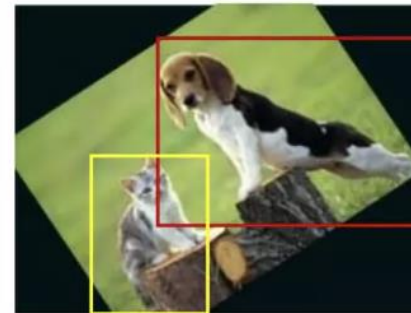
Resize



Flip



Crop



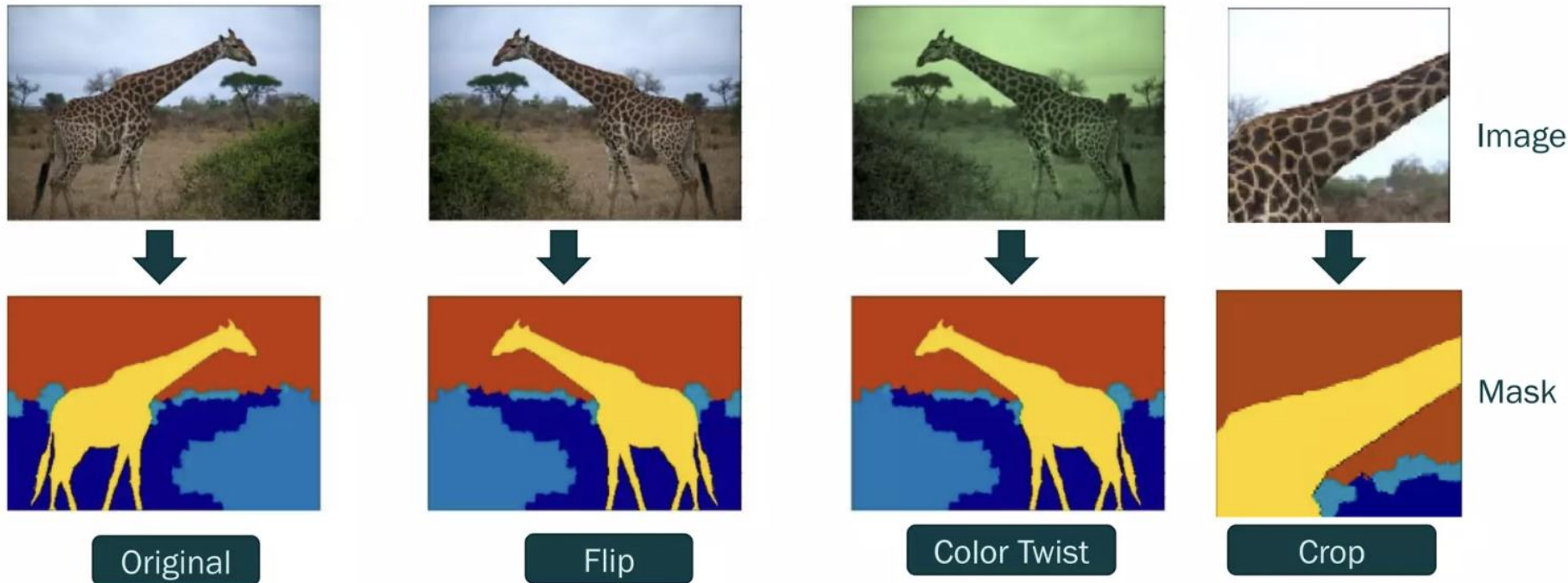
Rotate

Color data augmentation doesn't impact bounding boxes but geometric operations do.

For the object detection task, make sure to transform the bounding box locations accordingly.

Data Augmentation

Image Segmentation



For segmentation, we apply the same transform to the image and the mask.

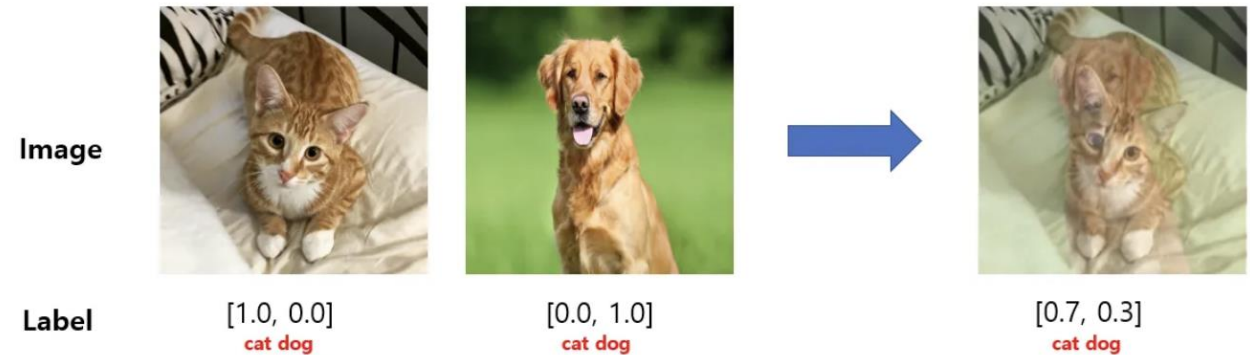
Data Augmentation

Mixup:

mix the input samples and their labels

$$\begin{aligned}\hat{x} &= \lambda x_i + (1 - \lambda)x_j, \\ \hat{y} &= \lambda y_i + (1 - \lambda)y_j,\end{aligned}$$

where $\lambda \in [0, 1]$ is a random number

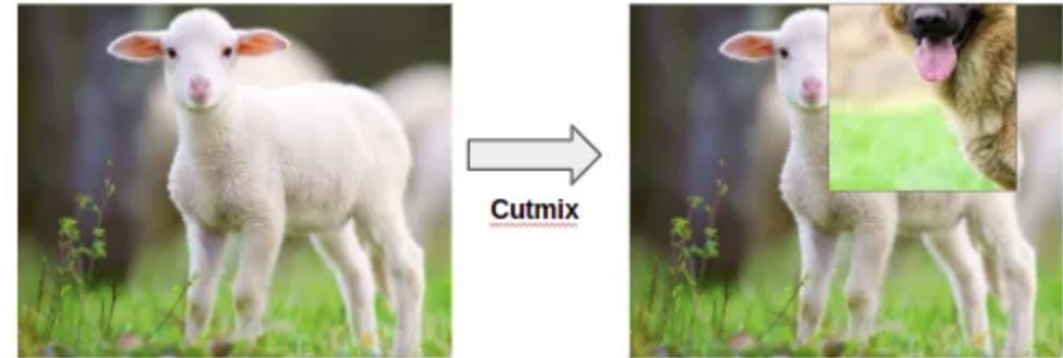


Note: the lambda values are values with the [0, 1] range and are sampled from the Beta distribution.

It extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets.

Data Augmentation

CutMix: a square region of an input image is replaced with a patch of similar dimensions from another image.



The virtual examples thus generated is given by,

$$\tilde{x} = M x_i + (1 - M) x_j$$

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

where M is a binary mask (often square) indicating the Cutout and the fill-in regions from the two randomly drawn images. Just like mixup, λ is drawn from $\text{Beta}(\alpha, \alpha)$ distribution and $\lambda \in [0, 1]$.

Data Augmentation

AugMix

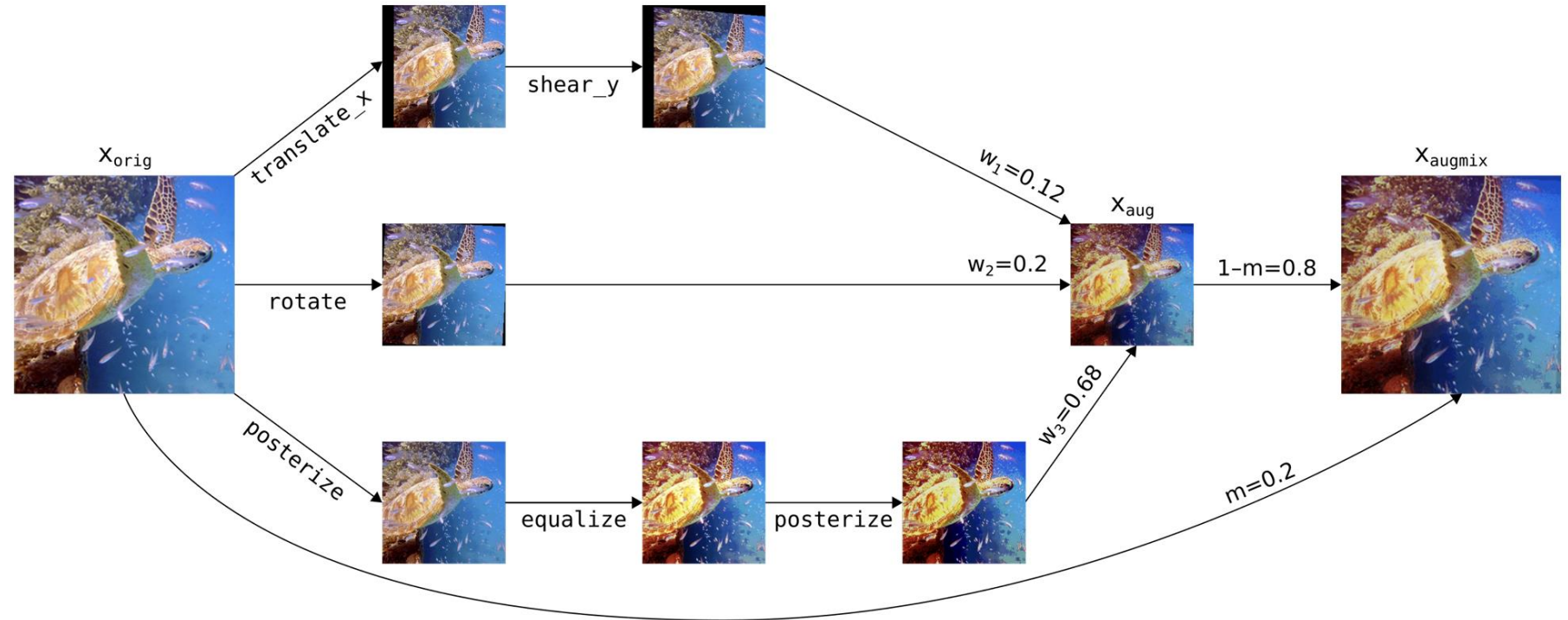


Figure 4: A realization of AUGMIX. Augmentation operations such as `translate_x` and weights such as m are randomly sampled. Randomly sampled operations and their compositions allow us to explore the semantically equivalent input space around an image. Mixing these images together produces a new image without veering too far from the original.

Metric-learning approach

- Endow the image space with a distance.
- How to compare two images?



d

↔



Metric-learning approach

- Endow the image space with a distance.
- How to compare two images?



d
↔
?

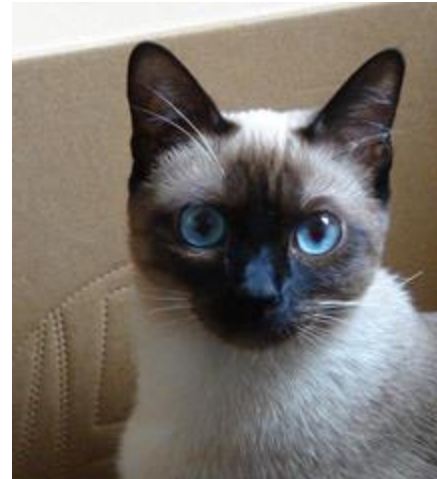


Metric-learning approach

- Endow the image space with a distance.
- How to compare two images?



d
↔
?



Metric-learning approach

- Endow the image space with a distance.
- How to compare two images?
We need an abstract representation of the images! Also called “feature vector”.



$$\begin{array}{c} ? \\ \longleftrightarrow \end{array} f \in \mathbb{R}^m$$
$$[0.234, 0.987, -0.735, 1.983, \dots, -0.736]$$

Metric-learning approach

How to obtain the feature vector?

- Help of **Transfer Learning**.
- Use a network trained on another dataset.

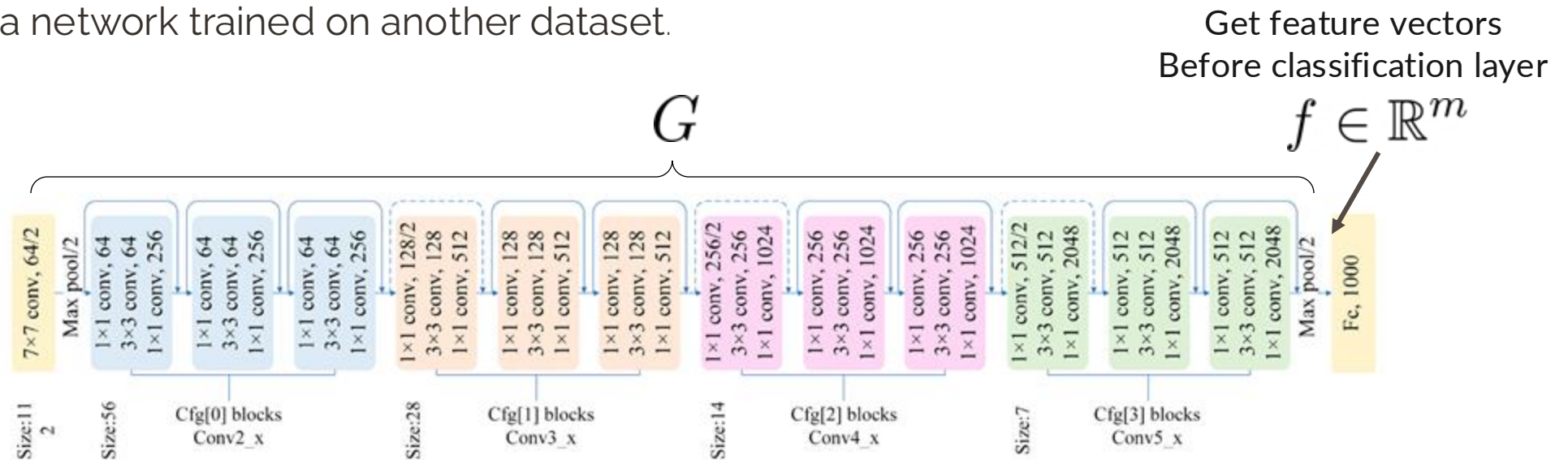
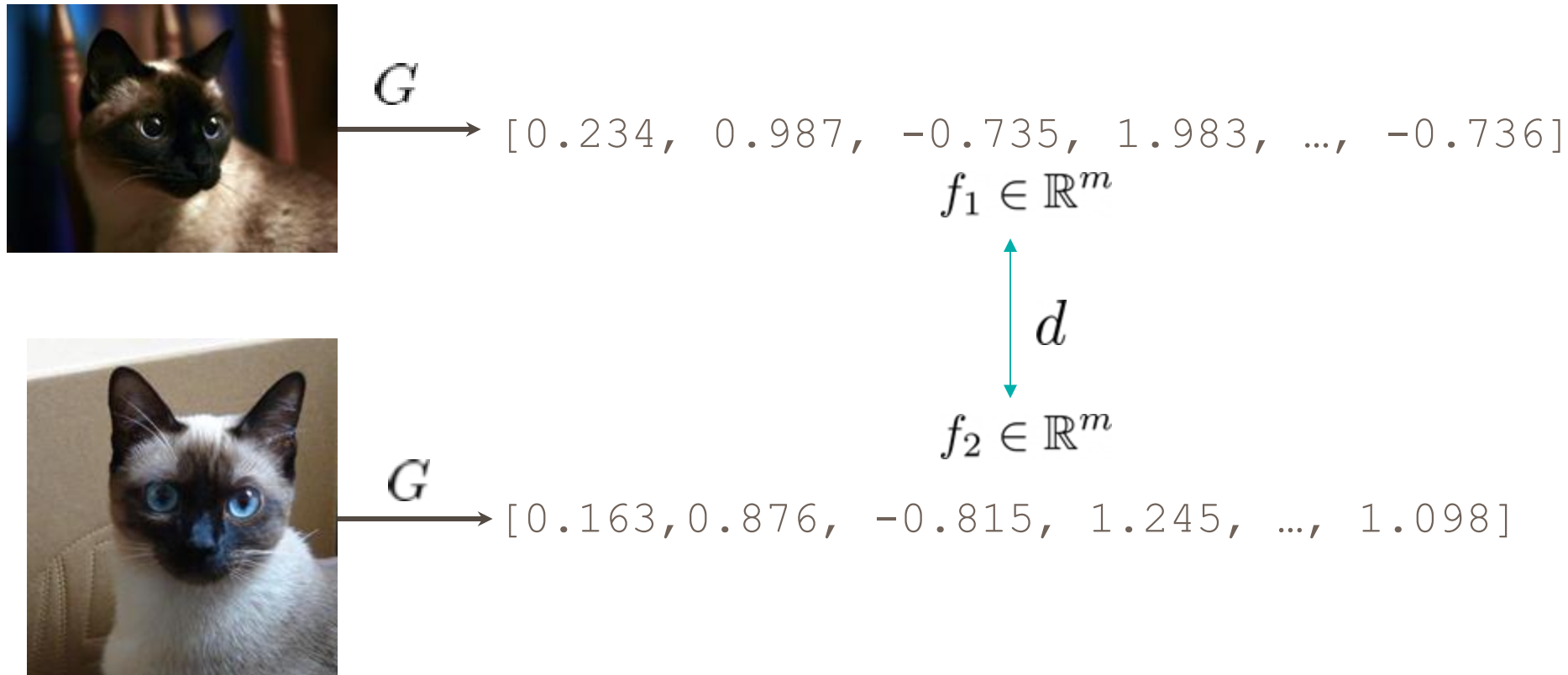


Image source: Deng et al., 2020

Metric-learning approach

Then, we can compare the feature vectors.



Metric-learning approach

Then, we can compare the feature vectors.



G

$[0.234, 0.987, -0.735, 1.983, \dots, -0.736]$

$f_1 \in \mathbb{R}^m$

d

$f_2 \in \mathbb{R}^m$



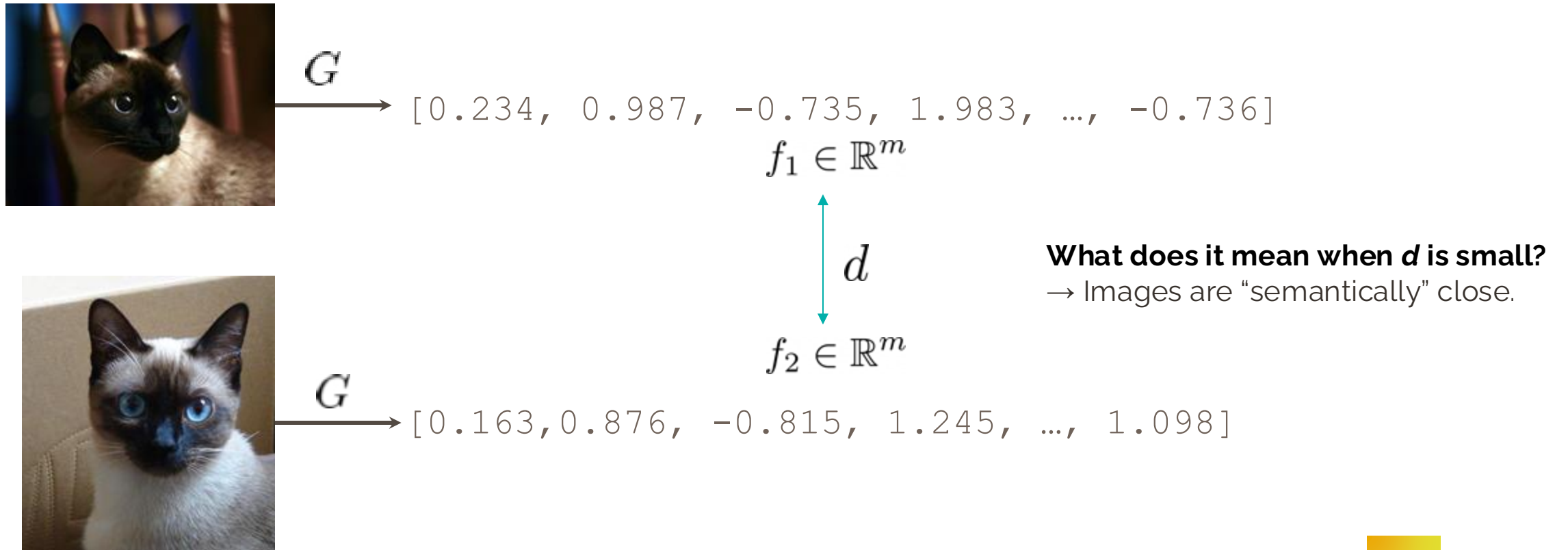
G

$[0.163, 0.876, -0.815, 1.245, \dots, 1.098]$

What does it mean when d is small?

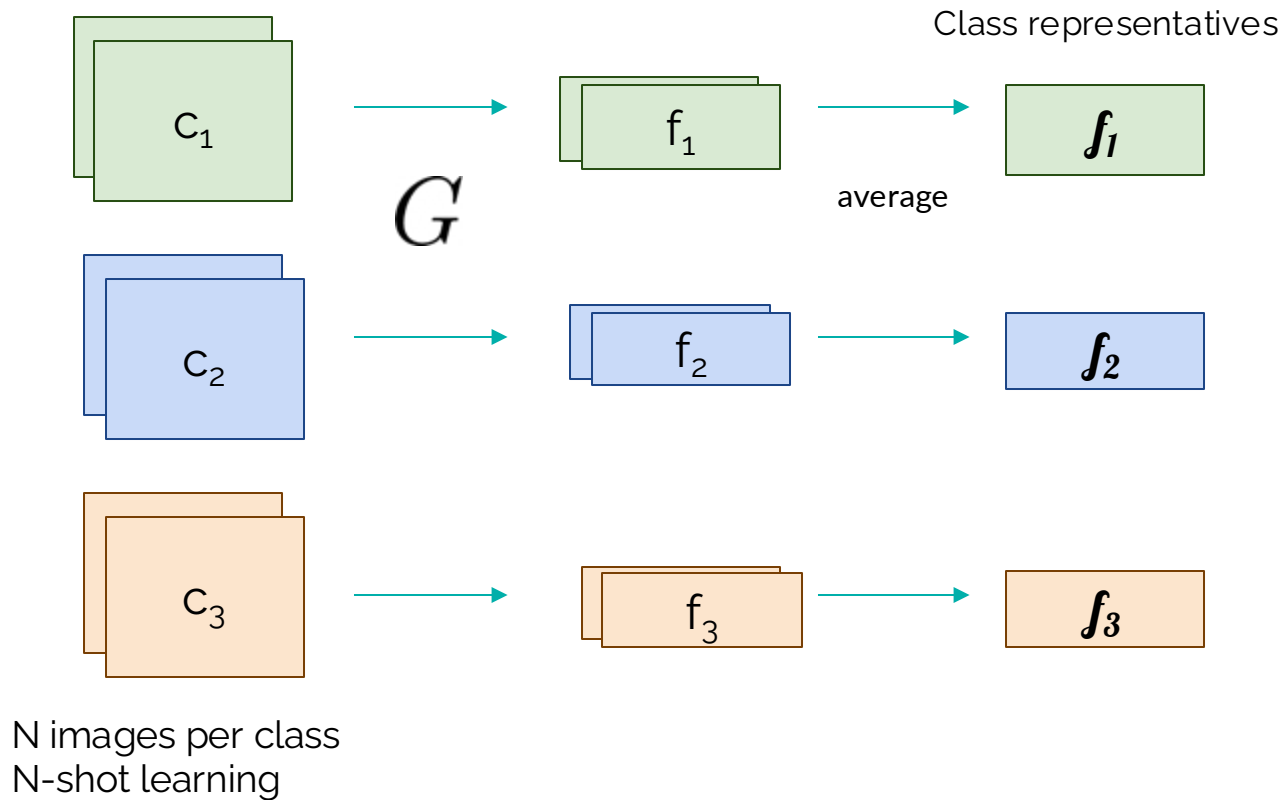
Metric-learning approach

Then, we can compare the feature vectors.



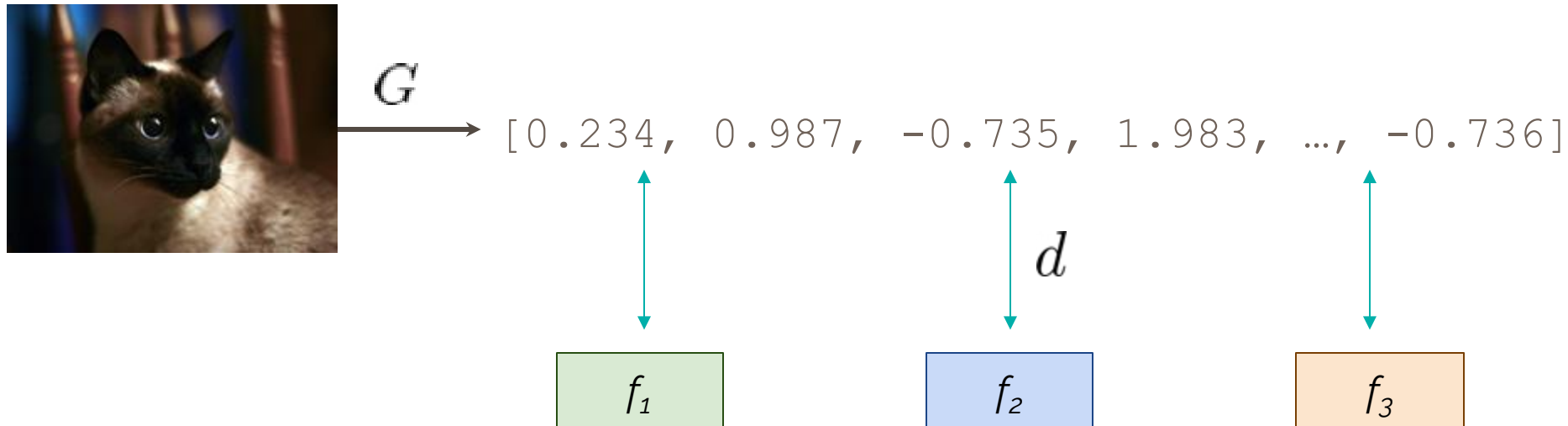
Metric-learning approach

How to classify images?



Metric-learning approach

How to classify images?



Compare the feature vector of the image to the class representatives.

d is called the "compatibility function".

Recap



- Dataset completely annotated...
- ...but all/some classes have very few training examples.
- 3 main methods
 - Data-driven approaches: augment the effective size of the dataset
 - Metric-learning approach: learn a distance between images
 - Parametric-approach: not covered here (using meta-learning)



Going further



- We just scratched the surface!
- Parameter-level approach using meta learning...
- Less than 1-shot learning...
- And if some classes don't even images or labels?
 - Zero-shot learning



Zero-shot learning

What is



?

- Gnamakoudji?
- Mendole?
- Manchu?

Zero-shot learning

What is



?

- Gnamakoudji: a traditional West-african beverage containing ginger.
- Mendole: a kind of fish that has a **dark spot on the side**.
- Manchu: an east-asian Tungusic language.

Zero-shot learning

What is



?

- Gnamakoudji: a traditional West-african beverage containing ginger.
- ~~Mendole: a kind of fish that has a **dark spot on the side**.~~
- ~~Manchu: an east-asian Tungusic language.~~


Zero-shot learning



Recognize images from new **unseen** classes...

...Having additional information on how the unseen classes look like.

Vocabulary:

- “**Seen**” class: we have labelled images of this class in the training dataset.
 - “**Unseen**” class: there are no images of this class in the training dataset.
 - “**Semantic vector**”: vector containing the extra information.
 - 1 semantic vector per class.
- 

Dataset - AwA2: Animals with Attributes 2

Images of 50 different animals.

- 40 seen classes (40 races of animals)
- 10 unseen classes
- Semantic vectors:
 - 85 visual attribute information (colors, textures, abilities, ...)

zebra

black:	yes
white:	yes
brown:	no
stripes:	yes
water:	no
eats fish:	no



Toy example

- 4 different animals:
 - Seen classes: horse, tiger, skunk
 - Unseen class: zebra
- We'll use attributes to distinguish them:
 - white, black, orange, stripes, big, small

Class	white	black	orange	stripes	big	small
horse	✓	✓			✓	
tiger	✓	✓	✓	✓	✓	
skunk	✓	✓		✓		✓
zebra	✓	✓		✓	✓	

Toy example

- 4 different animals:
 - Seen classes: horse, tiger, skunk
 - Unseen class: zebra
- We'll use attributes to distinguish them:
 - white, black, orange, stripes, big, small

Not very useful attributes...
Do not discriminate anything.

Class	white	black	orange	stripes	big	small
horse	✓	✓			✓	
tiger	✓	✓	✓	✓	✓	
skunk	✓	✓		✓		✓
zebra	✓	✓		✓	✓	



Toy example

- 4 different animals:
 - Seen classes: horse, tiger, skunk
 - Unseen class: zebra
- We'll use attributes to distinguish them:
 - white, black, orange, stripes, big, small

Not very useful attributes...
Too discriminative.

Class	white	black	orange	stripes	big	small
horse	✓	✓			✓	
tiger	✓	✓	✓	✓	✓	
skunk	✓	✓		✓		✓
zebra	✓	✓		✓	✓	

Toy example

Better (if possible):
proportion of images of one class to have a certain attribute.

→ AWA2 contains this amount of detail!

Class	white	black	orange	stripes	big	small
horse	0.6	0.4	0	0	0.7	0
tiger	0.6	0.8	0.9	0.99	0.8	0
skunk	0.99	0.99	0	1	0	0.9
zebra	1	1	0	1	0.7	0

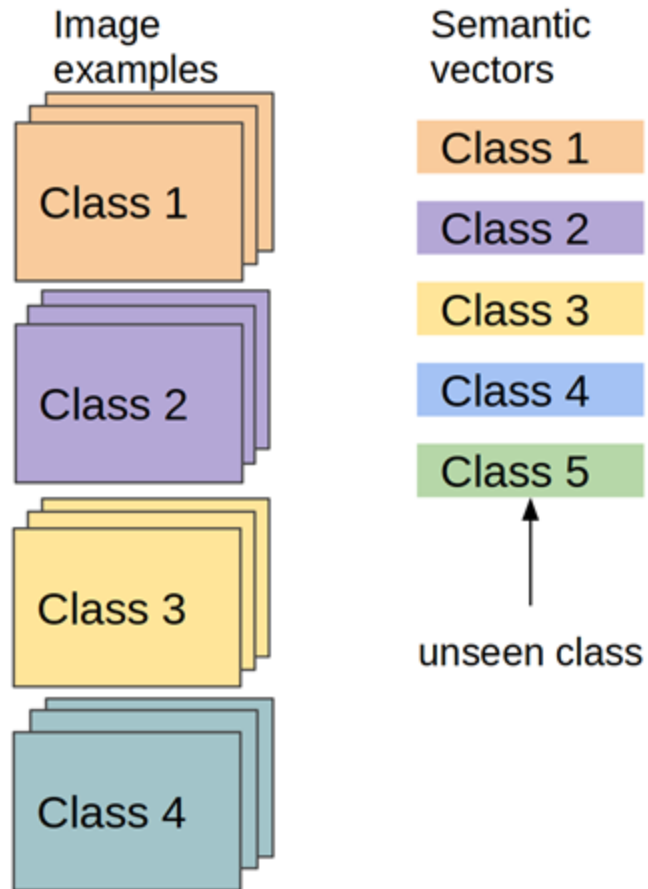
AwA2 - PCA of semantic vectors

Grazer, Vegetation, Hooves

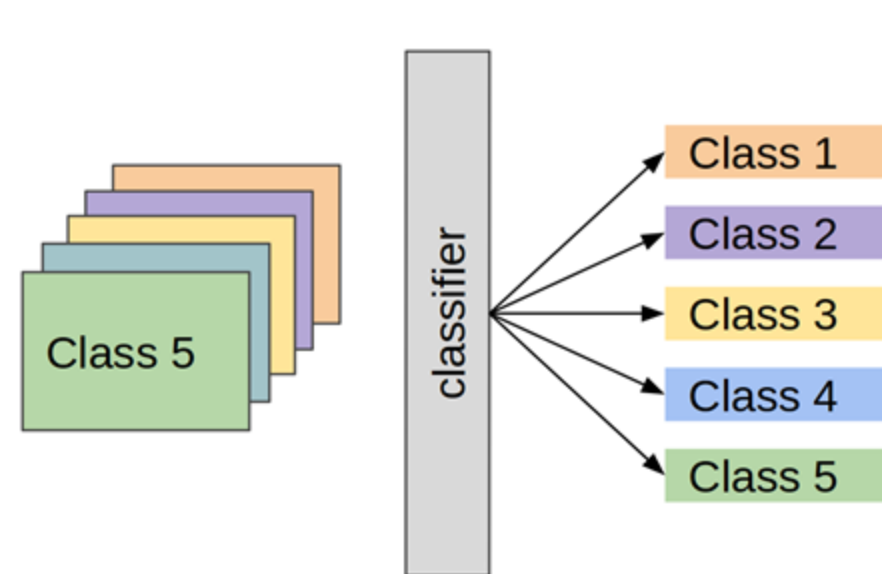


Recap

During training, we have...

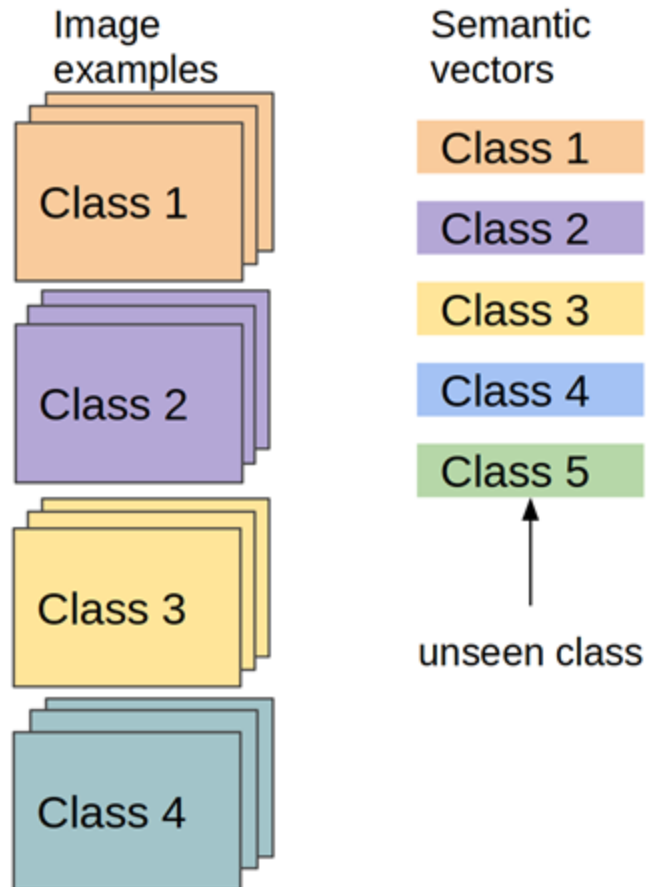


And when we evaluate/test...

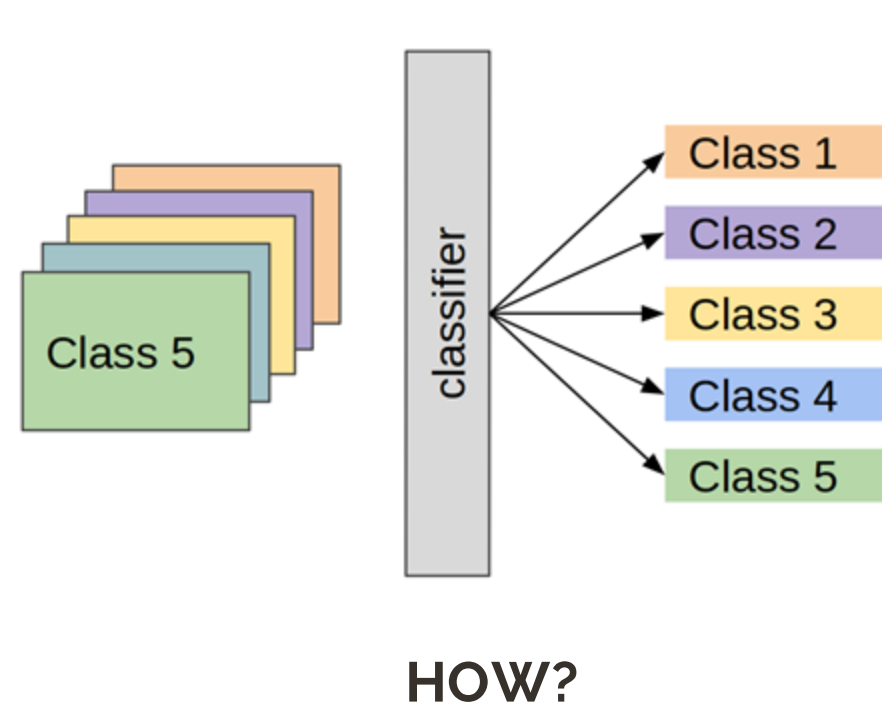


Recap

During training, we have...

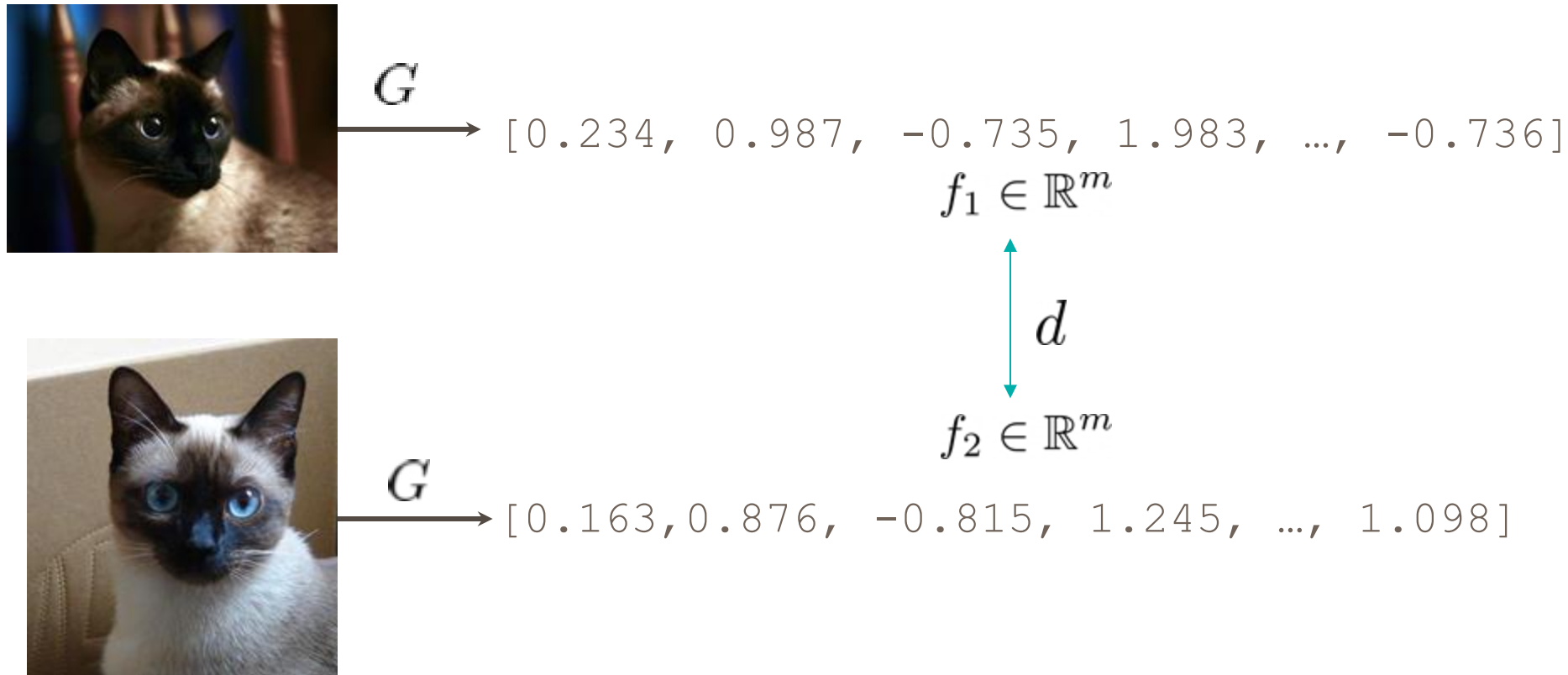


And when we evaluate/test...



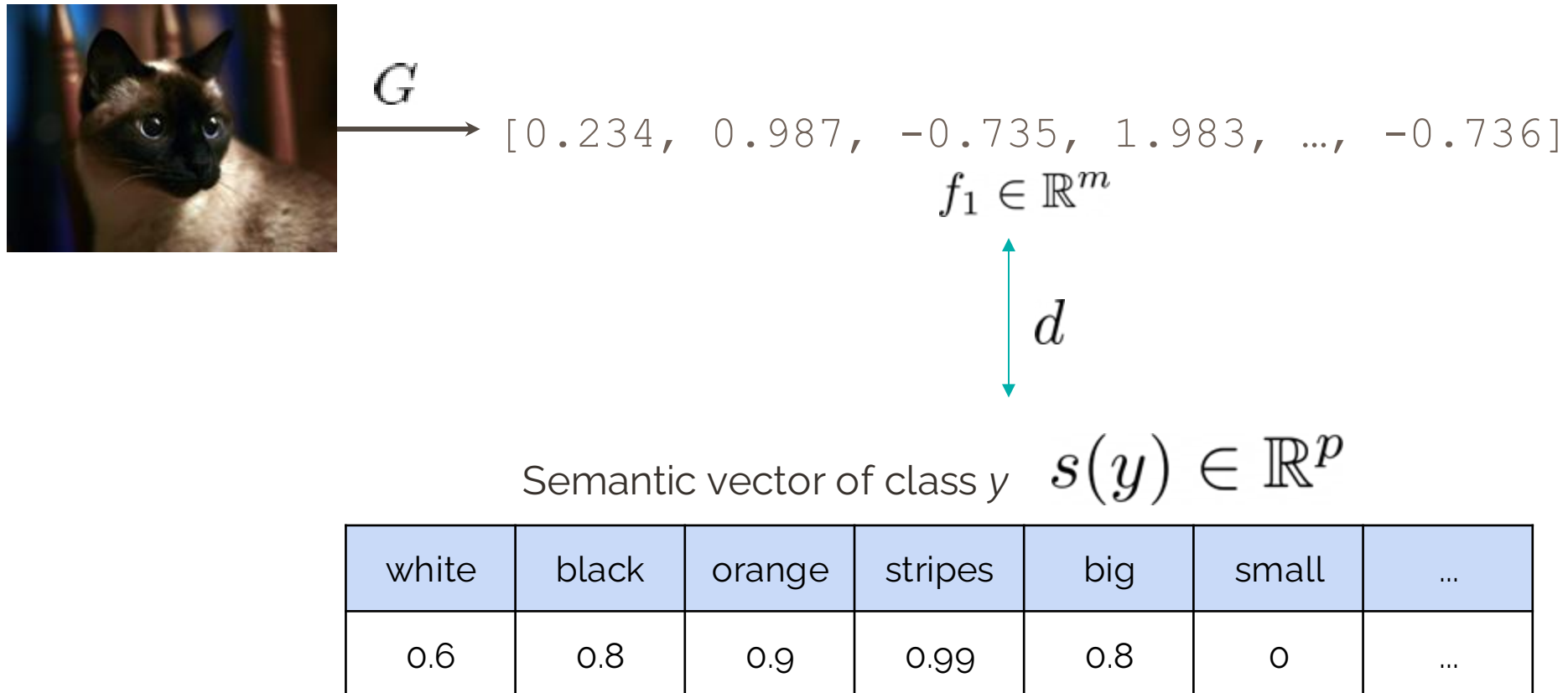
First method: compatibility function

Remember in few-shot learning...



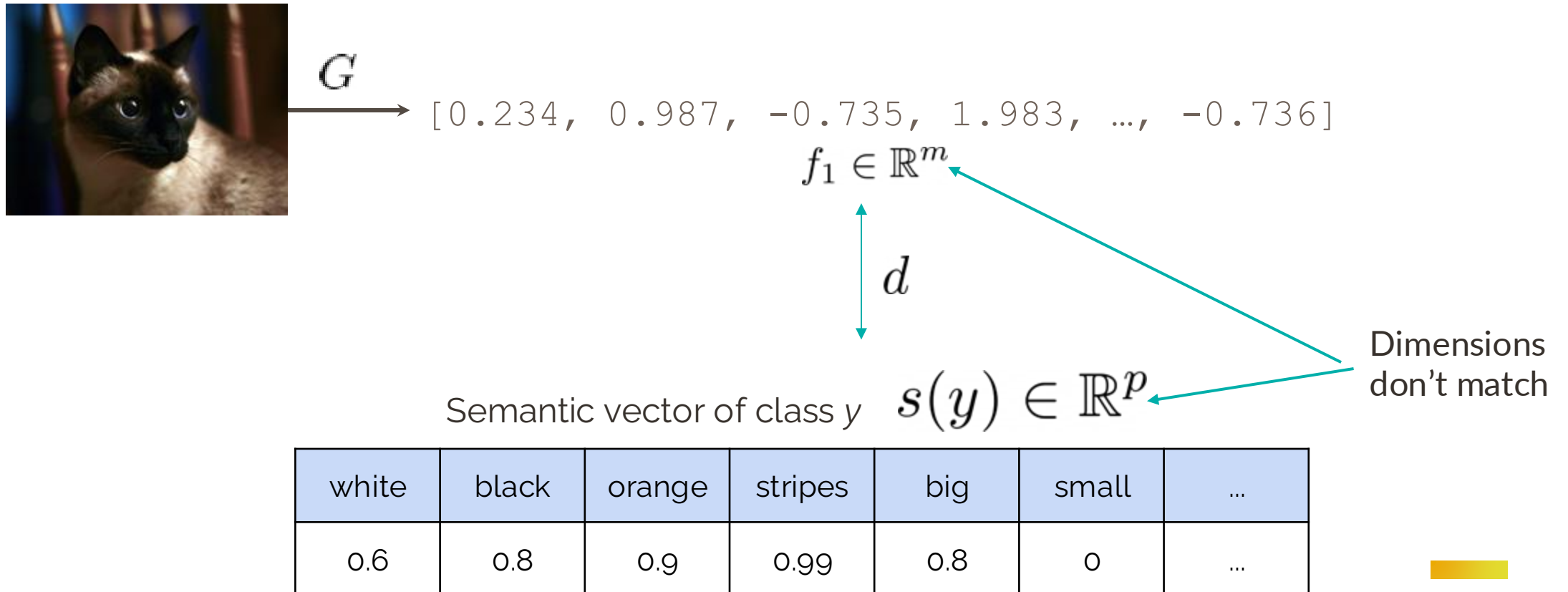
First method: compatibility function

Now, we actually want:

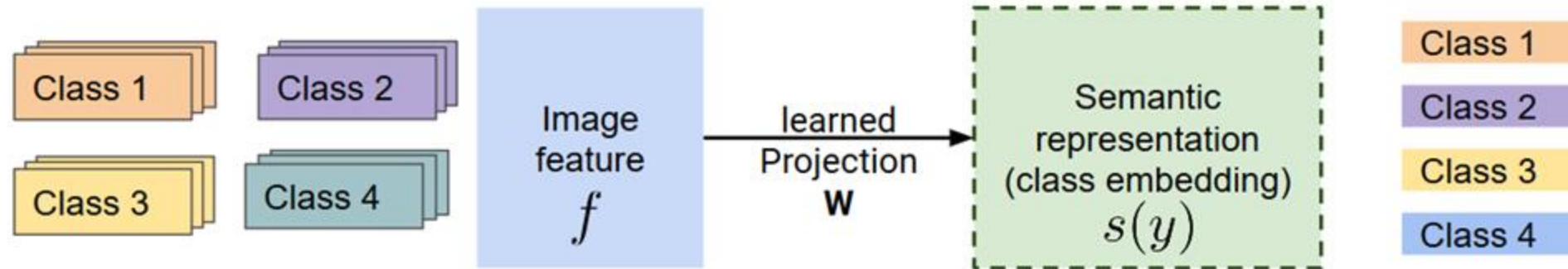


First method: compatibility function

Now, we actually want:

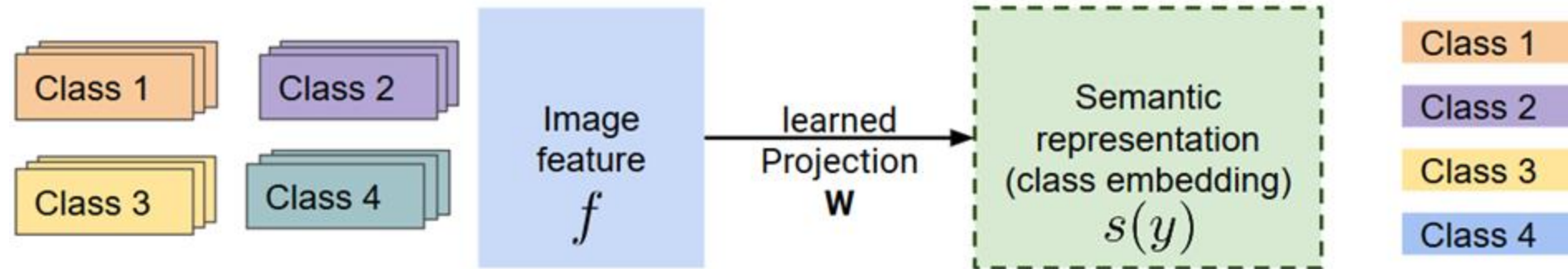


First method: compatibility function



$$d(f, y) = \langle Wf, s(y) \rangle$$

First method: compatibility function



$$d(f, y) = \langle Wf, s(y) \rangle$$

Compatibility function.
How compatible is f with class
 y ?

Projection mapping
(learned with seen classes)

Semantic vector

$\langle \cdot, \cdot \rangle$ Scalar product notation

DeViSE



[A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In NIPS, 2013.]

What loss function do we use?



DeViSE

[\[A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In NIPS, 2013.\]](#)

What loss function do we use? Pairwise ranking objective.

$$\sum_{y \in \mathcal{Y}^{tr}} \max(0, 1_{y_n \neq y} + d(f_n, y) - d(f_n, y_n))$$

For each seen class

DeViSE

$$\sum_{y \in \mathcal{Y}^{tr}} \max(0, 1_{y_n \neq y} + d(f_n, y) - d(f_n, y_n))$$

Actual class
of image n

Feature of
image n

DeViSE

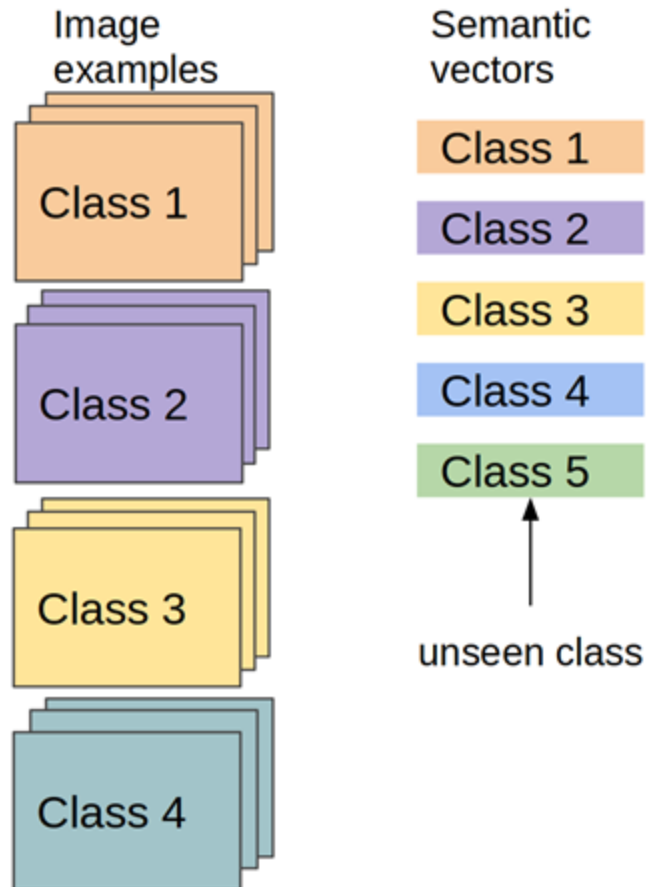
$$\sum_{y \in \mathcal{Y}^{tr}} \max(0, 1_{y_n \neq y} + d(f_n, y) - d(f_n, y_n))$$

Recall: we want to minimize this function.

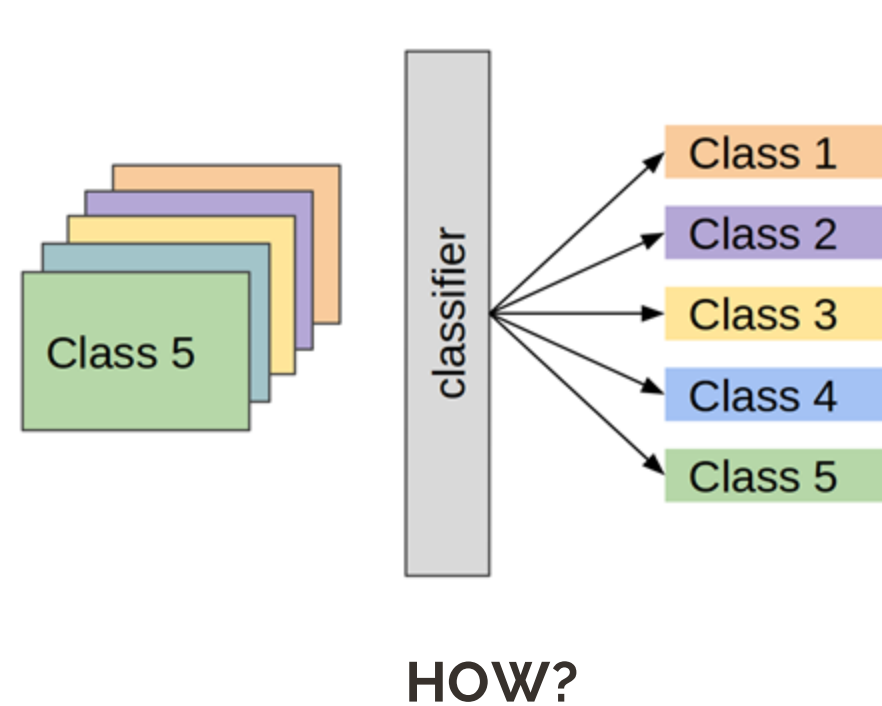
- If $y = y_n$, this is 0
- Else, if $d(f_n, y) > d(f_n, y_n)$, it's > 1
- If $d(f_n, y) < d(f_n, y_n)$, between 0 and 1

Recap

During training, we have...



And when we evaluate/test...



HOW?

Another paradigm: generative approach

Remember our definitions?

- Gnamakoudji: a traditional West-african **beverage** containing **ginger**.
- Mendole: a kind of **fish** that has a **dark spot on the side**.
- Manchu: an east-asian Tungusic **language**.

Another paradigm: generative approach

Remember our definitions?

- Gnamakoudji: a traditional West-african **beverage** containing **ginger**.
- Mendole: a kind of **fish** that has a **dark spot on the side**.
- Manchu: an east-asian Tungusic **language**.

What if I ask you to imagine **what an image of the class “mendole” could look like?**

Another paradigm: generative approach

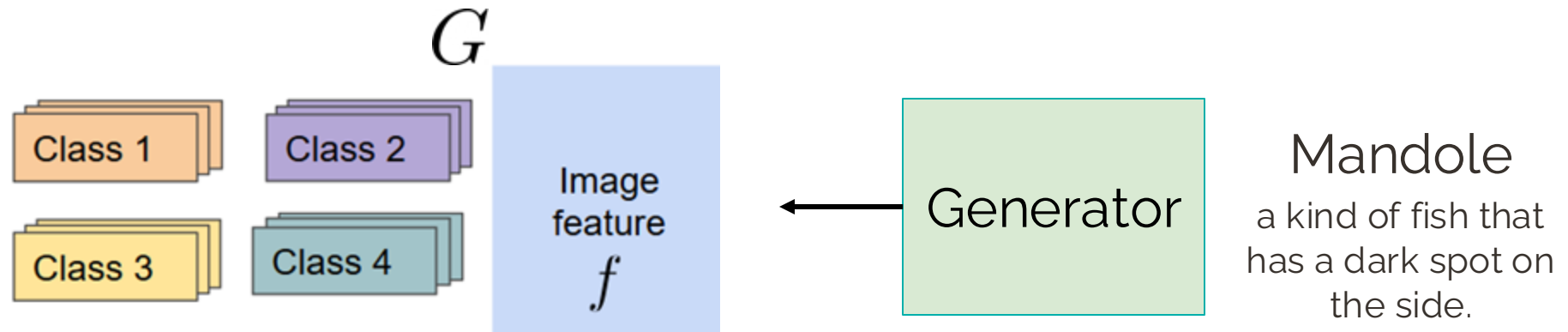
Mendole
a kind of fish that
has a dark spot on
the side.

Generator



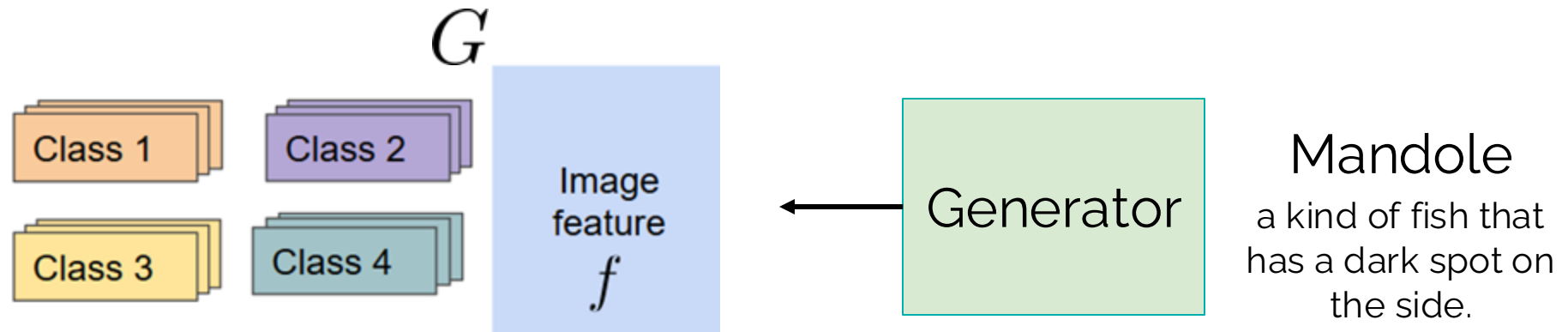
Very hard!

Another paradigm: generative approach



Easier to generate the feature vectors of the image!

Another paradigm: generative approach



Easier to generate the feature vectors of the image!
We will later learn more about generative models.


Recap



Zero-shot learning is:

to recognize images from new **unseen** classes...
...having additional information on how the unseen classes look like.

Two main approaches:

- Discriminative, using a compatibility function
 - Generative
- 

Recap



But what if we don't have labels at all?



Unsupervised learning



Learning without annotation!

Very broad subject:

- Clustering (~ classification),
- Anomaly detection,
- Image generation,
- Image restoration,
- ...



Unsupervised learning

Idea:

1. Find a new surrogate task that does not require the data to be labelled.
2. Train a feature extractor (previously denoted G) using this task.
3. Either
 - a. Use a clustering algorithm of the learned features (k-means, spectral clustering, ...)
 - b. Annotate one example per class and use few-shot learning methods

Common surrogate tasks

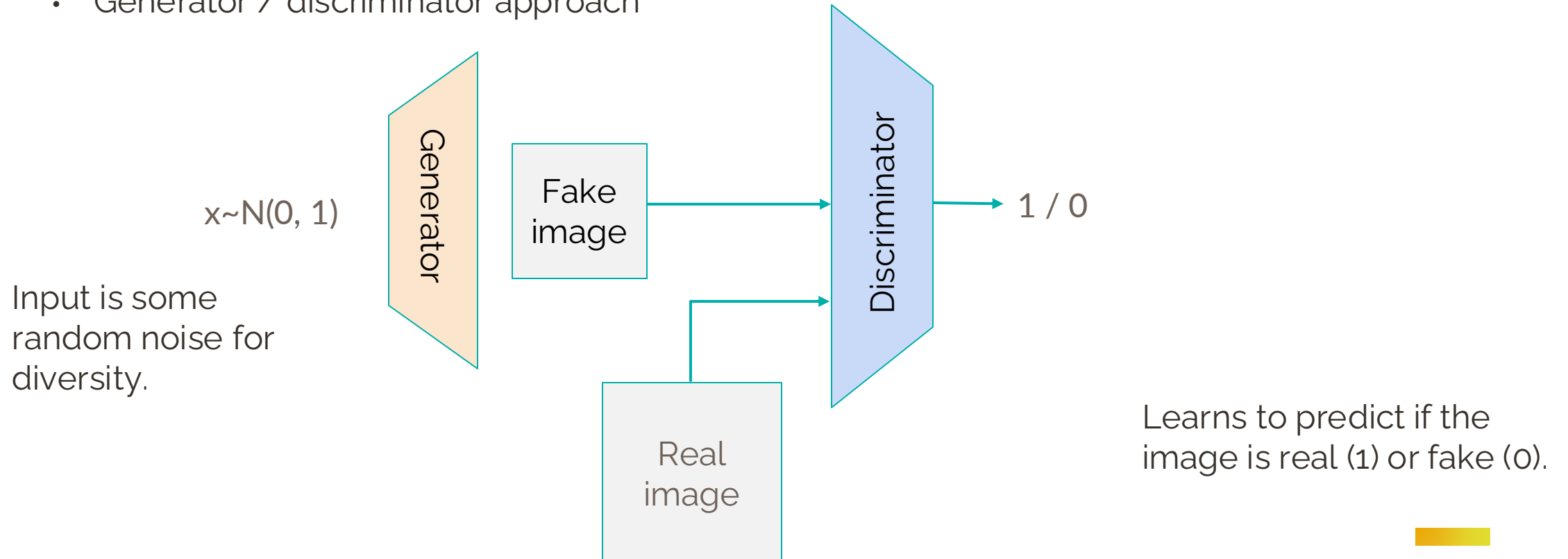
- Image reconstruction
- Image denoising
- Patch from same image / different image
- Matching image - text-description (readily available online, instagrams with hashtags, image descriptions, ...)

auto-encoders

Contrastive learning

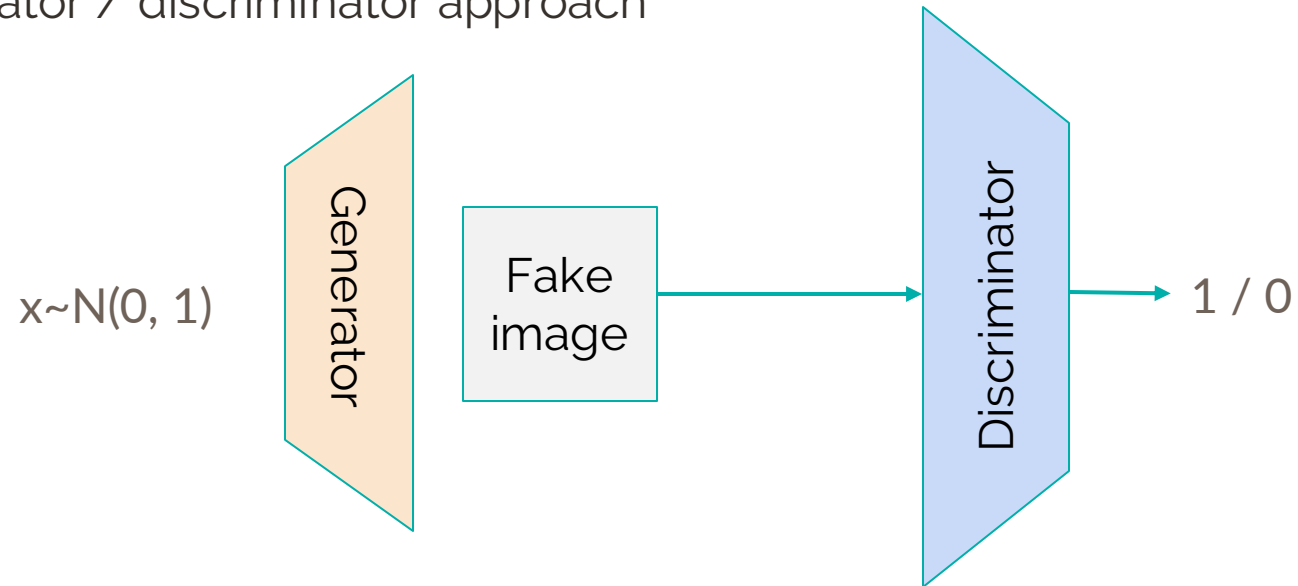
Generative Adversarial Networks - GANs

- Not an auto-encoder
- Generator / discriminator approach



Generative Adversarial Networks - GANs

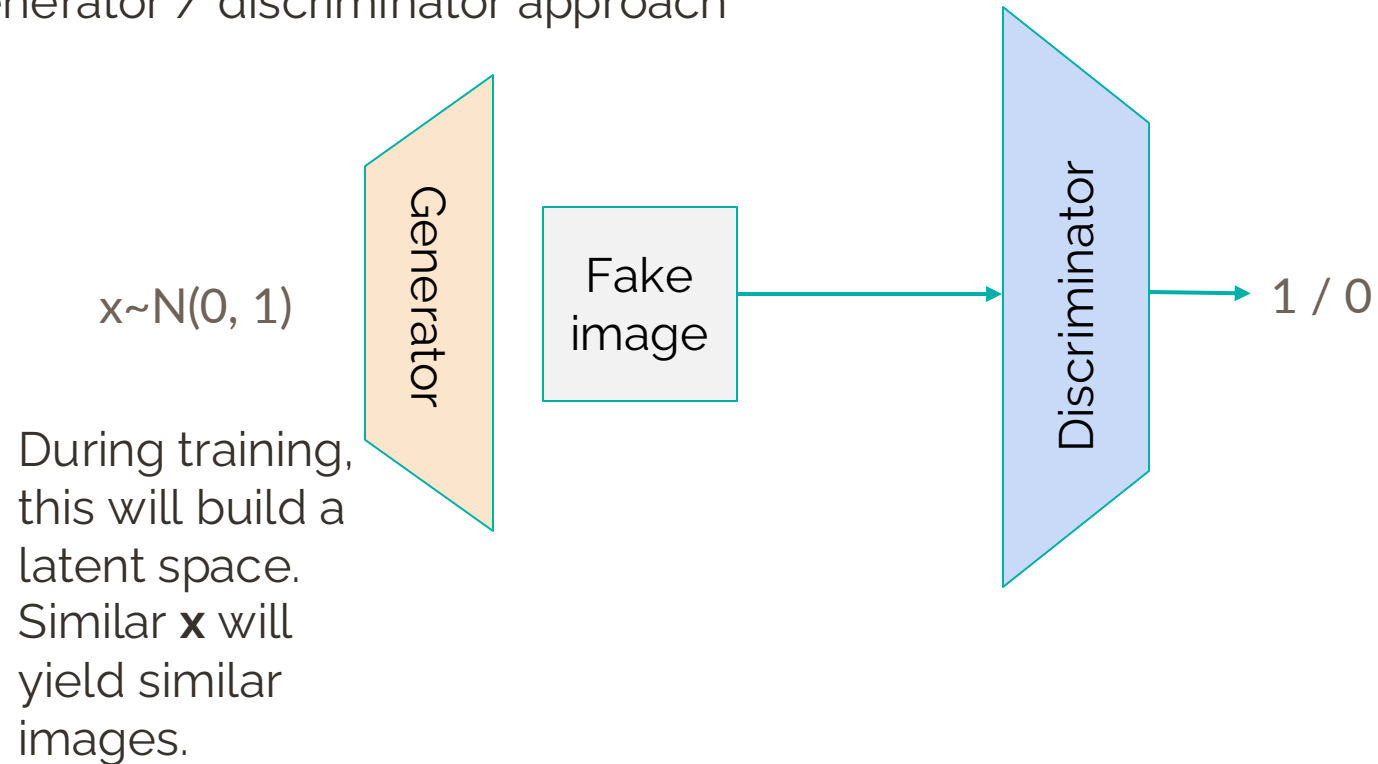
- Not an auto-encoder
- Generator / discriminator approach



Learns to fool the discriminator.

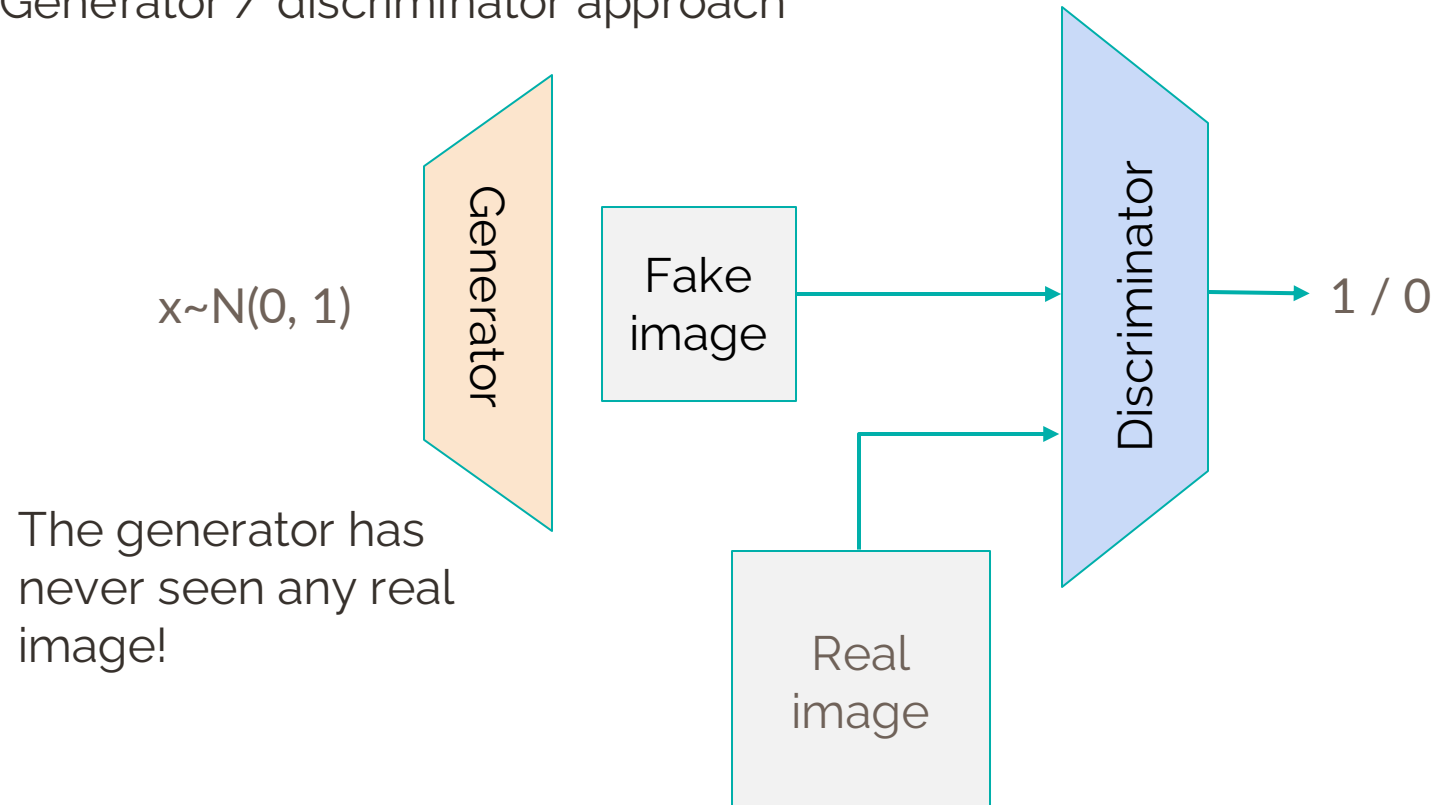
Generative Adversarial Networks - GANs

- Not an auto-encoder
- Generator / discriminator approach



Generative Adversarial Networks - GANs

- Not an auto-encoder
- Generator / discriminator approach



Evolution of GAN quality over the years



2014



2015



2016



2017



2018