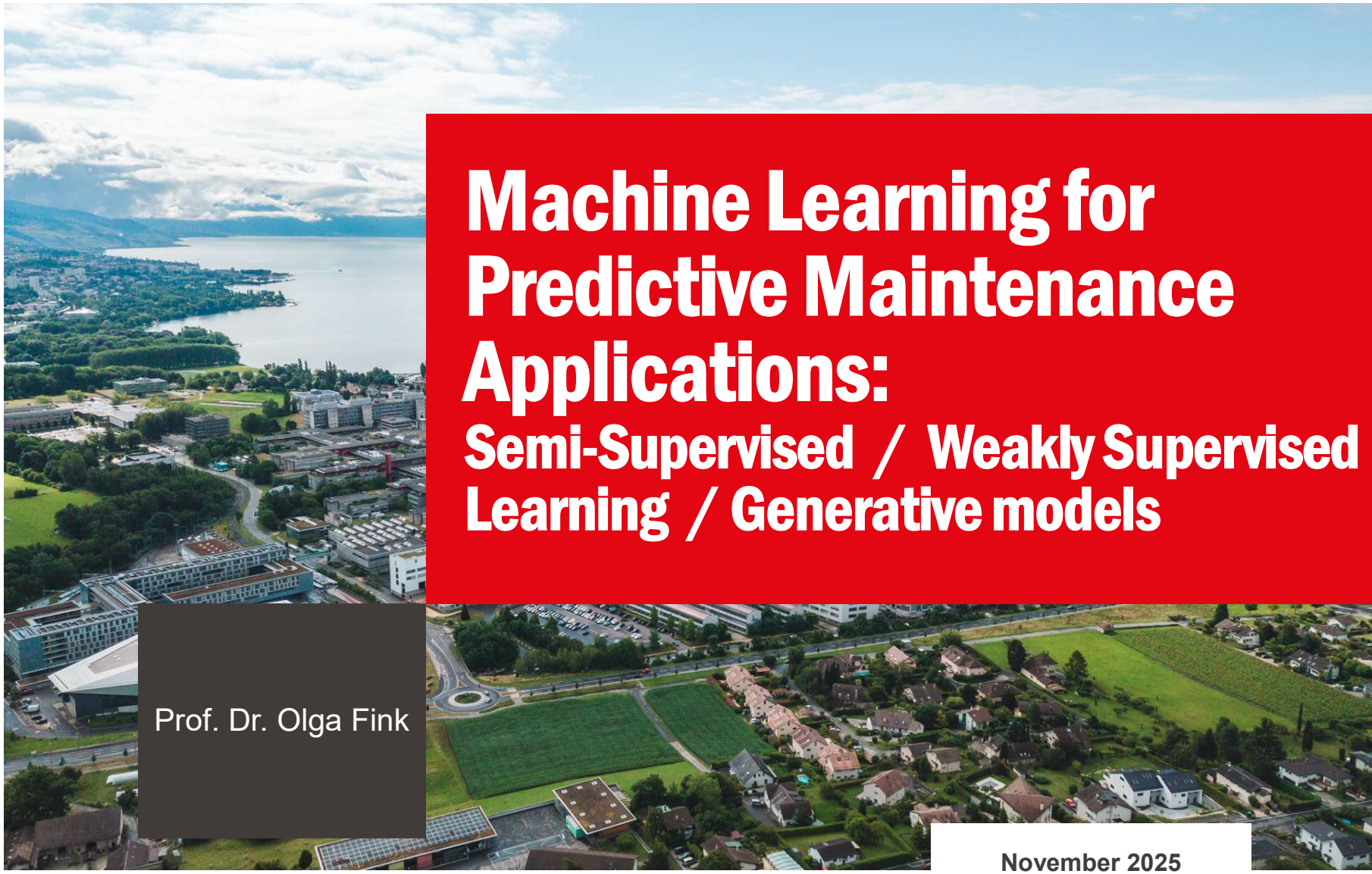


Machine Learning for Predictive Maintenance Applications:

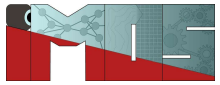
Semi-Supervised / Weakly Supervised Learning / Generative models

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

November 2025

Inductive vs. Transductive Learning



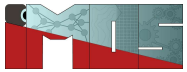
Inductive Learning

- Learns a **generalizable model** from the training data.
- The model can be applied to **unseen test instances**.
- Focuses on learning a **mapping** from inputs → outputs.
- Assumes training and test data come from the **same distribution** (unless domain adaptation is applied).
- **Examples:**
  - Standard supervised learning
  - Neural networks, random forests
  - Pretrained models used for downstream tasks

Transductive Learning

- Makes predictions **only for the given test instances**, not for unseen future data.
- Learns from both **labeled training data** and **unlabeled test data**.
- Does **not** aim to learn a fully generalizable function.
- Exploits the **structure of the test set** (e.g., clusters, manifolds).
- **Examples:**
  - Label propagation, graph-based semi-supervised learning
  - Transductive SVM
  - Test-time adaptation methods

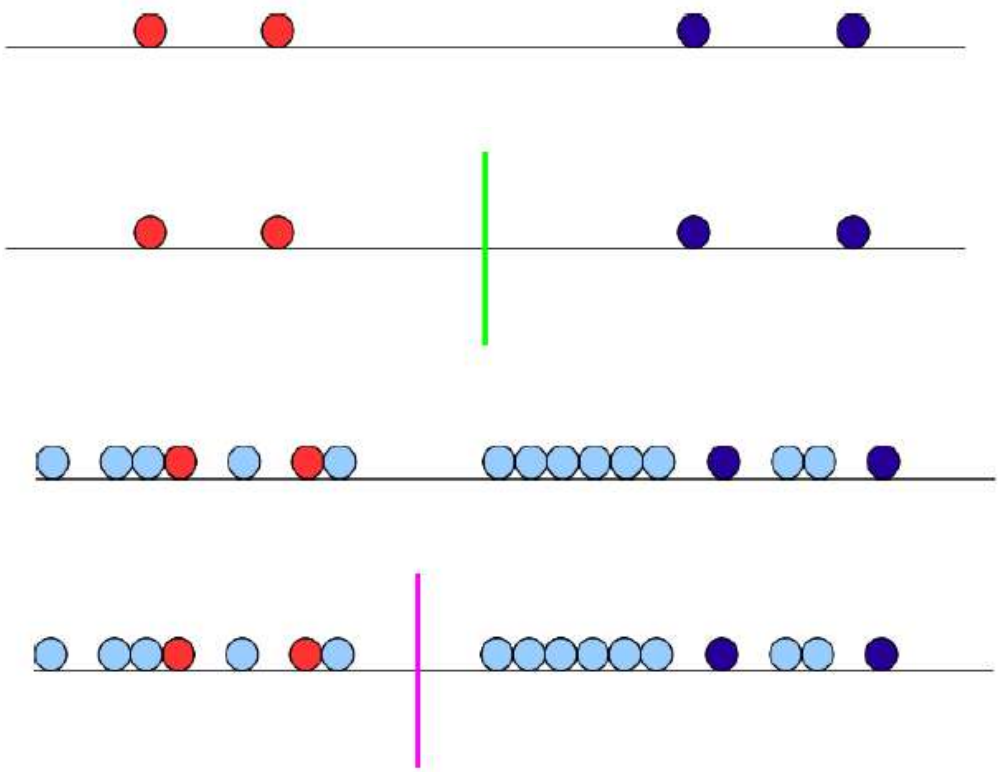
EPFL



Semi-supervised learning

Olga Fink

Why/How Might Unlabeled Data Help?

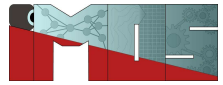


Source: Piyush Rai 2011

Semi-supervised learning

- Semi-supervised learning leverages the available unlabeled data to improve the performance of the supervised learning task.
- Different concepts have been proposed for semi-supervised learning tasks
  - generative models
  - graph-based methods
  - transductive methods
- A further possibility to distinguish the different semi-supervised learning approaches is to differentiate between those based on
  - consistency regularization
  - entropy minimization
  - traditional regularization

Benefits of Semi-Supervised Learning:



- **Cost-effective:** Collecting labeled data can be time-consuming and expensive, especially for large datasets. Semi-supervised learning allows for the use of large amounts of unlabeled data, which can be collected relatively easily and inexpensively, to train the model.
- **Improved accuracy:** By incorporating both labeled and unlabeled data into the training process, those models can learn patterns and relationships within the data that may not be easily visible from the labeled data alone. This can lead to improved accuracy compared to models trained solely on labeled data.
- **Better generalization:** They tend to generalize better to new data compared to models trained solely on labeled data. This is because the models are able to learn more about the underlying structure of the data by incorporating both labeled and unlabeled data into the training process.
- **Flexibility**: It is a flexible approach that can be used in a variety of different applications, including image classification, natural language processing, and more.

Limitations of Semi-Supervised Learning

- **Labeled data limitations:** The effectiveness of semi-supervised learning models is dependent on the quality and quantity of the labeled data available. If the labeled data is limited or of poor quality, the model may not perform as well.
- **Model selection:** Selecting the right model for a semi-supervised learning problem can be challenging, as different models may perform better or worse depending on the specific problem and dataset.
- **Evaluation difficulty:** Evaluating the performance of that kind of model can be challenging, as the available labeled data may be limited and it can be difficult to determine the effectiveness of the model in making predictions for new data.

Assumptions for different methods

- **Continuity assumption** → objects near each other are likely to share the same group or label +data points that are part of the same cluster are more likely to share the same label
- **Cluster assumptions** →data points that are part of the same cluster are more likely to share the same label
- **Manifold assumptions** → high-dimensional data lie on a low-dimensional manifold → the learning algorithm should respect the manifold structure → learning should primarily happen on the manifold
- **Smoothness assumption:** if two points in a high-dimensional space are close to each other, then so should be their outputs

The Learning Problem

- Using both labeled and unlabeled data to build better learners, than using each one alone

input instance x, label y

learner $f : \mathcal{X} \mapsto \mathcal{Y}$

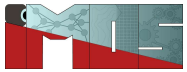
labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$

unlabeled data $X_u = \{x_{l+1:n}\}$, available during training

usually $l \ll n$

test data $X_{test} = \{x_{n+1:}\}$, not available during training

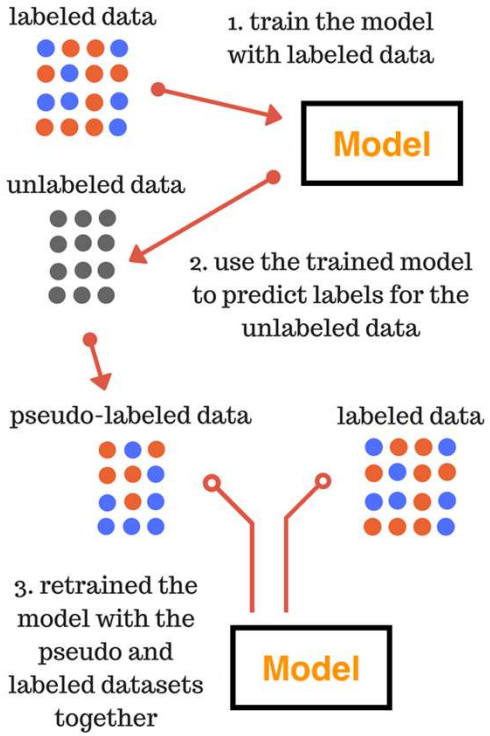
Source: Piyush Rai 2011



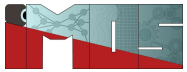
Pseudo-labeling

Olga Fink

Pseudo-labeling



Source: Potrimba, 2022



Self-Training

Olga Fink

Self-Training

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

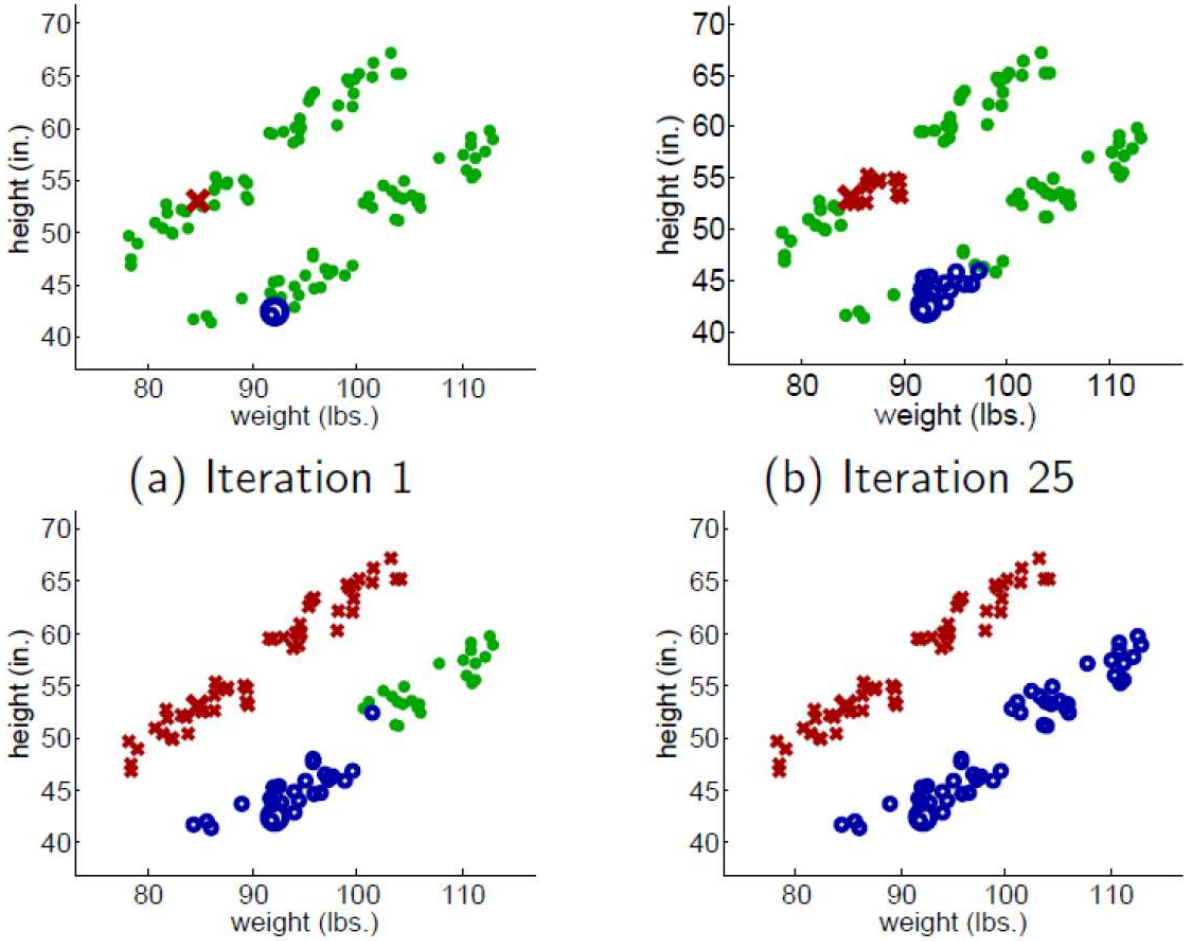
1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat:
3. Train f from L using supervised learning.
4. Apply f to the unlabeled instances in U.
5. Remove a subset S from U; add $\{(\mathbf{x}, f(\mathbf{x}))|\mathbf{x} \in S\}$ to L.

Self-training is a wrapper method

- the choice of learner for *f* in step 3 is left completely open
- good for many real-world tasks like natural language processing
- but mistakes by *f* can reinforce itself

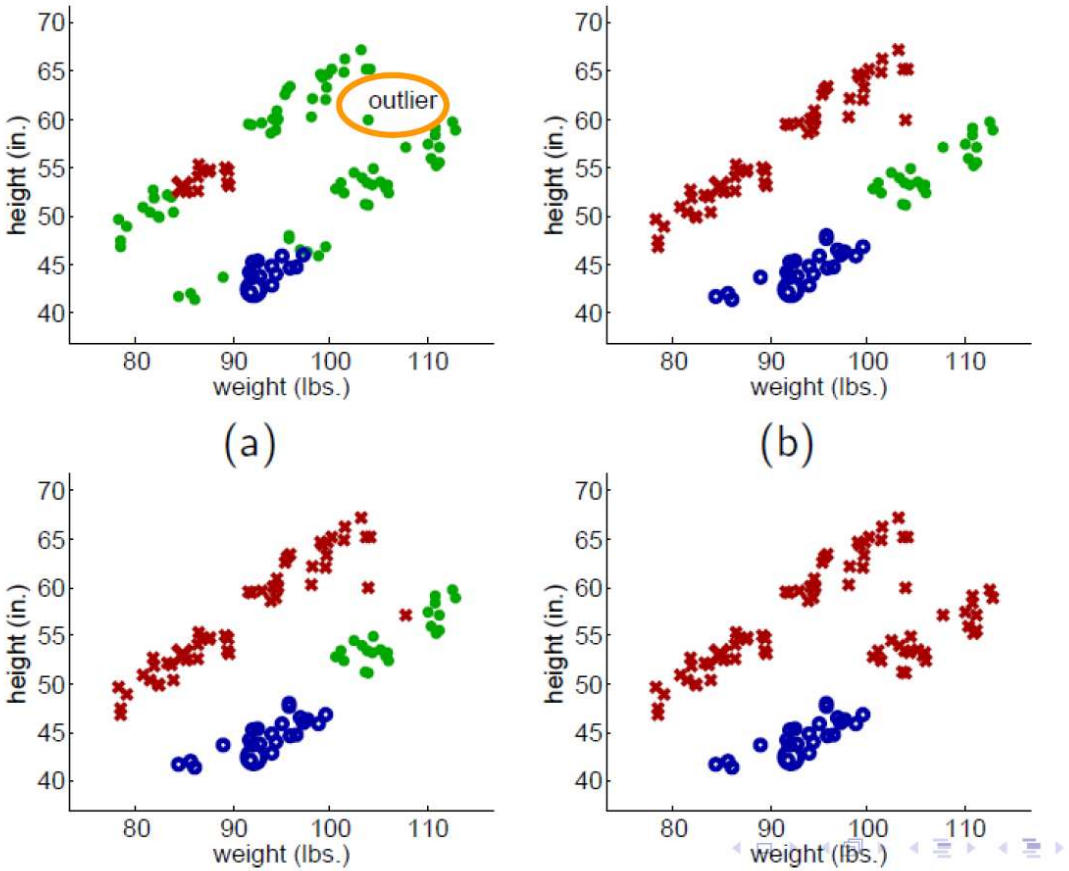
Source: Piyush Rai 2011

Self-Training: A Good Case

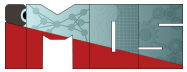


Source: Piyush Rai 2011

Self-Training: A bad Case



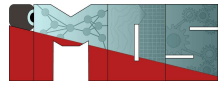
Source: Piyush Rai 2011



Cluster-and-Label Approach

Olga Fink

Cluster-and-Label Approach



Step 1: Clustering the Data

- **Data Preparation**: Gather both labeled and unlabeled data and preprocess it as necessary (scaling, normalization, etc.).
- **Clustering Algorithm**: Apply a clustering algorithm to the entire dataset (both labeled and unlabeled). Algorithms like K-means, DBSCAN, or hierarchical clustering are commonly used depending on the nature of the data and the specific requirements of the task.

Step 2: Assigning Labels to Clusters

- Once the data is clustered, there are several strategies to label the unlabeled samples

Step 3: Evaluation and Refinement

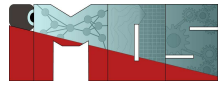
- **Evaluate the Accuracy**: Assess the accuracy of the labels assigned to the unlabeled data using various metrics (if additional labeled data is available for validation).
- **Refine Clustering and Labeling**: Based on the evaluation, refine the clustering approach, choose a different labeling strategy, or adjust the parameters of the used algorithms

➢ Finally train a supervised learner on the entire labeled data

➢ Assumption: Clusters coincide with decision boundaries

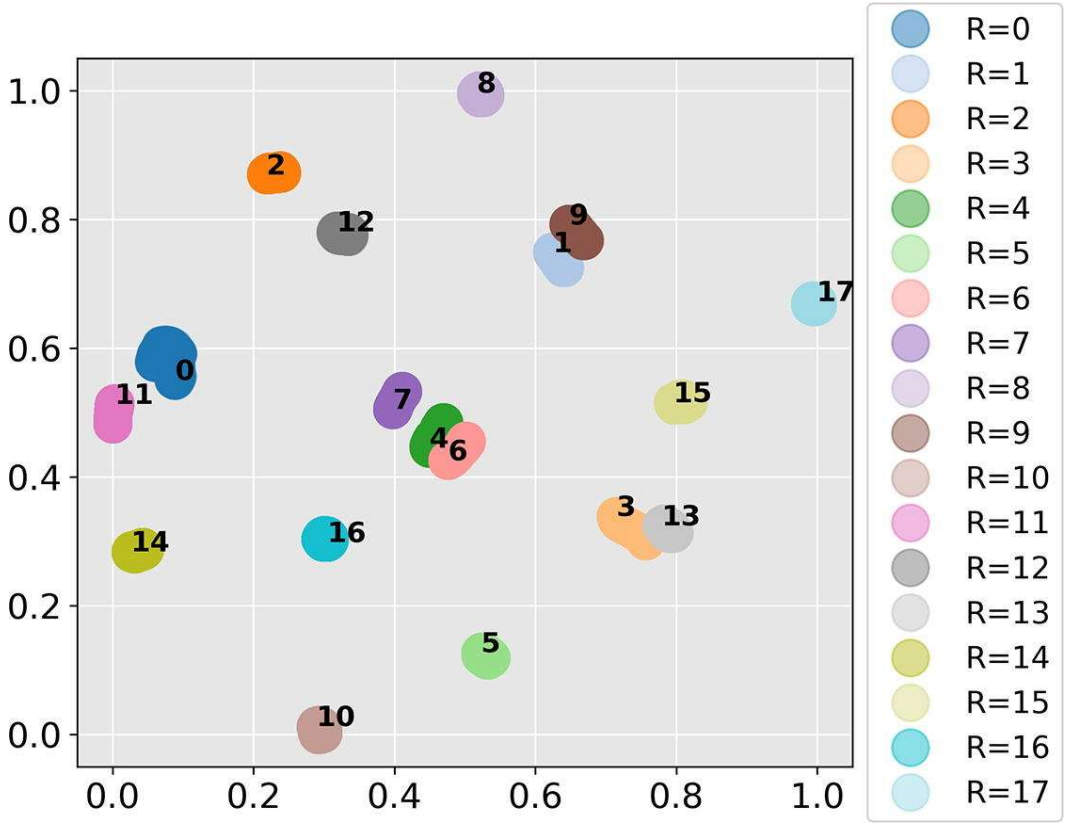
  - Poor results if this assumption is wrong

Strategies to label the unlabeled samples

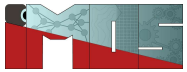


- **Majority Voting (Label Propagation within Clusters)**:
  - **Method**: For each cluster, look at the labels of the labeled data points (if any) within that cluster.
  - **Assign Labels**: Assign the most frequent label among the labeled data in the cluster to all the unlabeled points in the same cluster.
  - **Consideration**: This method assumes that the clusters are homogeneous enough that most of the data points in a single cluster belong to the same class. This might not always hold, especially with poor clustering results.
- **Nearest Neighbors**:
  - **Method**: For each unlabeled point in a cluster, find the nearest labeled data points (possibly in the same or different clusters) and use their labels to determine the label of the unlabeled point.
  - **Assign Labels**: This could be based on the nearest single neighbor or a majority vote among several nearest neighbors.
  - **Consideration**: This method assumes that similar data points share the same label, which is typical in many real-world scenarios but can be computationally expensive.
- **Model-Based**:
  - **Method**: Train a classification model on the labeled data points.
  - **Prediction**: Use this model to predict the labels of the unlabeled data points.
  - **Assign Labels**: The model uses the features of the data points to assign labels, potentially allowing for more flexibility and accuracy compared to simpler methods like majority voting.
  - **Consideration**: This method can be effective if the labeled dataset is representative of the entire dataset.
- **Cluster Purity**:
  - **Method**: Evaluate the purity of clusters by checking if labeled data points in each cluster predominantly belong to one class.
  - **Decision Making**: If a cluster contains labeled samples from multiple classes, further analysis or refined clustering might be necessary to improve the quality of the labeling.

If labeled samples available, label unlabeled samples, e.g. based on majority voting…



Arias Chao, M., Adey, B. T., & Fink, O. (2021). Implicit supervision for fault detection and segmentation of emerging fault types with Deep Variational Autoencoders. *Neurocomputing*, *454*, 324-338.

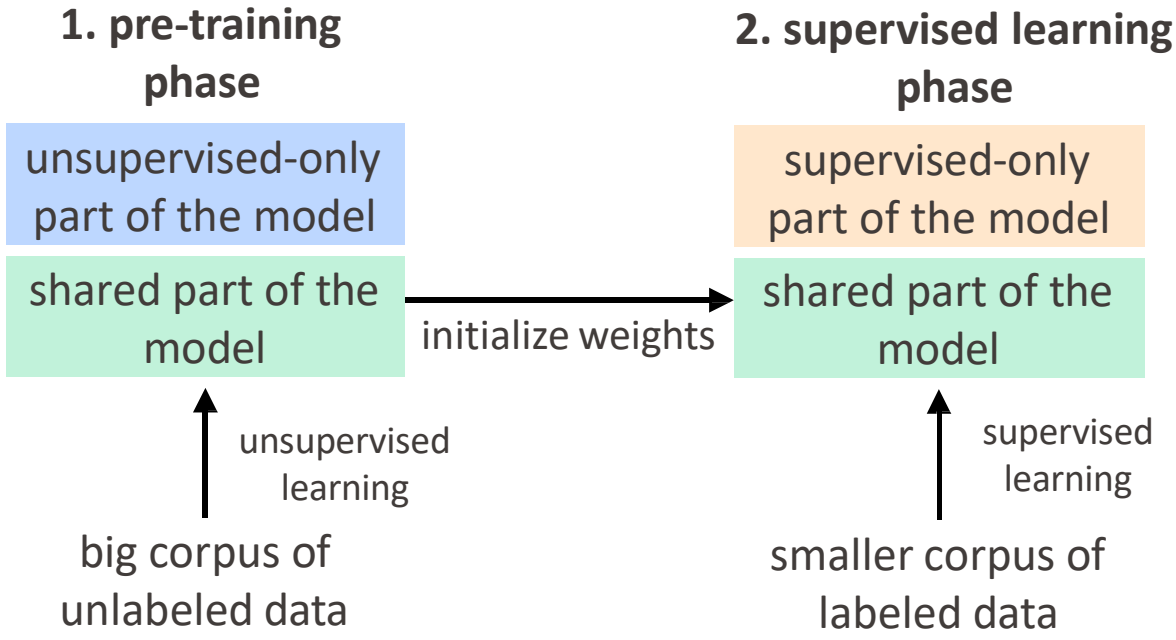


Pre-training

Olga Fink

Pre-training

- First train an unsupervised model on unlabeled data
- Then incorporate the model's learned weights into a supervised model and train it on the labeled data
  - Optional: continue fine-tuning the unsupervised weights.

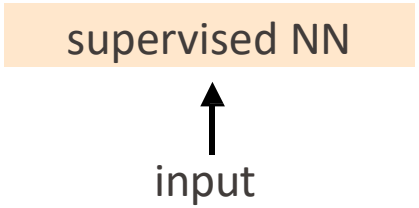


Source: Clark.2019

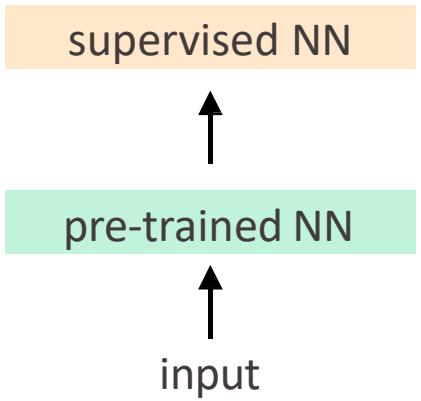
Why does pre-training work?

- ”Smart” initialization for the model
- More meaningful representations in the model

Supervised learning: have to learn everything from “raw” input



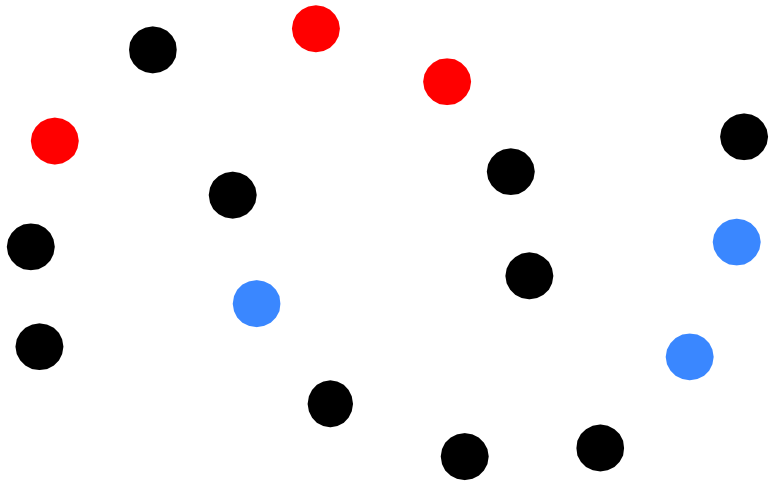
Pre-training: supervised part gets more useful representations as input



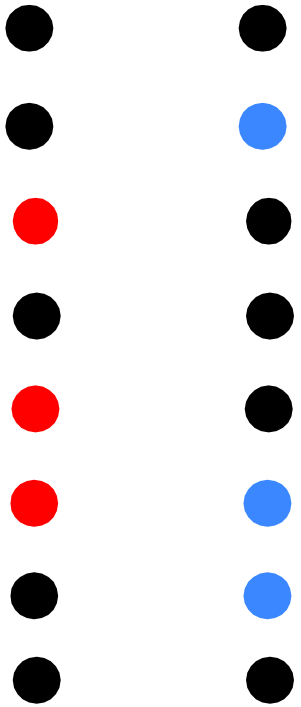
Source: Clark.2019

Why does pre-training work?

original representation space



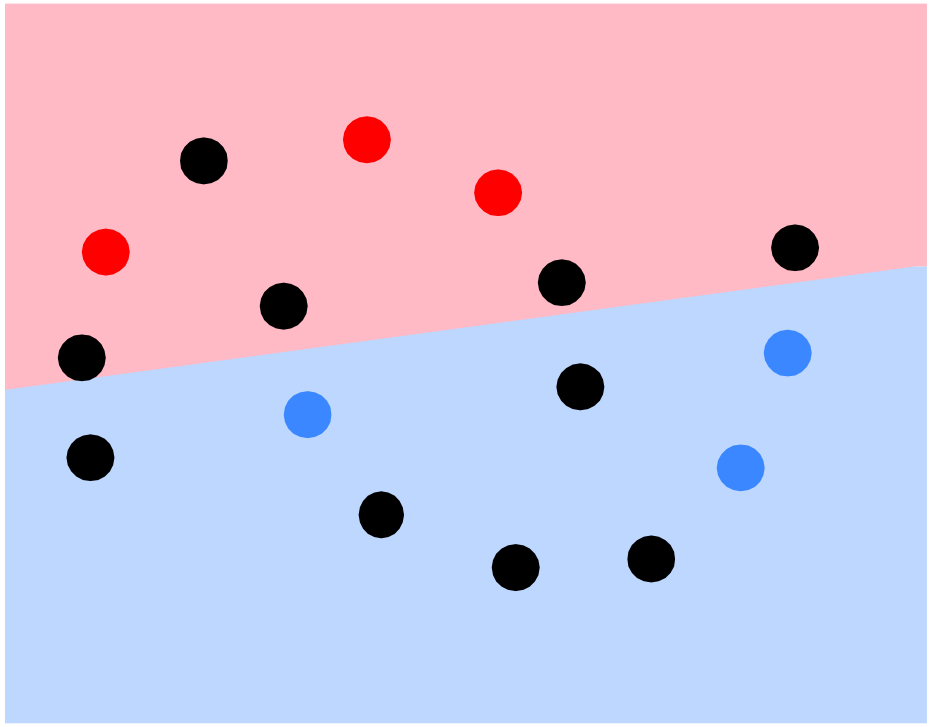
learned representation space after pre-training



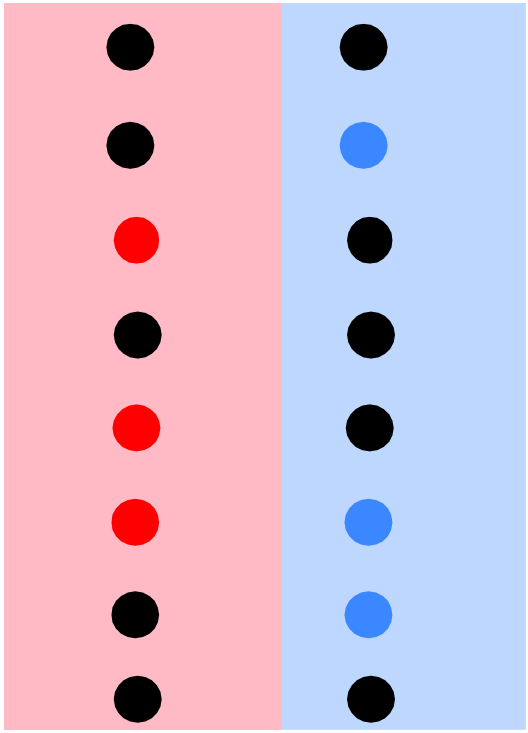
Source: Clark.2019

Why does pre-training work?

original representation space

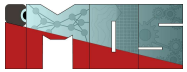


learned representation space after pre-training



Supervised part of the model has a much easier job after pre-training

Source: Clark.2019



Consistency regularization

Olga Fink

Consistency Regularization

- Add noise to the student's inputs

$$J(\theta) = CE(p(y|x_j, \theta), p(y|x_j + \eta, \theta))$$

Soft target

Model learns to produce target even when noise is added to its input

Where η is a vector with a random direction and a small magnitude ϵ

Source: Clark.2019

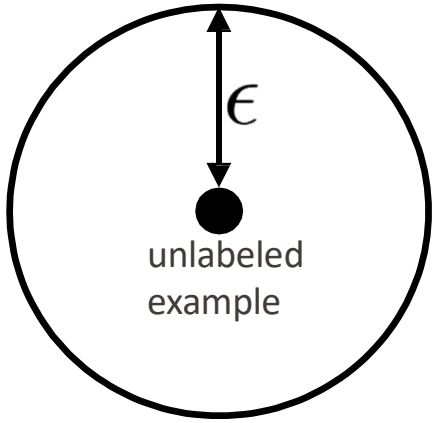
Consistency Regularization

- Add noise to the student's inputs

$$J(\theta) = CE(p(y|x_j, \theta), p(y|x_j + \eta, \theta))$$

Where η is a vector with a random direction and a small magnitude ϵ

- Train the model so a bit of noise doesn't mess up its predictions
- Equivalently, the model must give consistent predictions to nearby data points

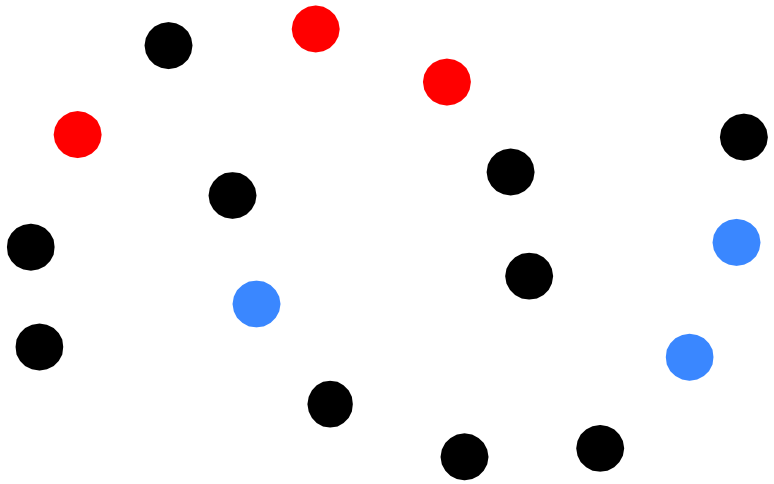


The model is trained to give the same prediction for any point in the circle

"distributional smoothing"

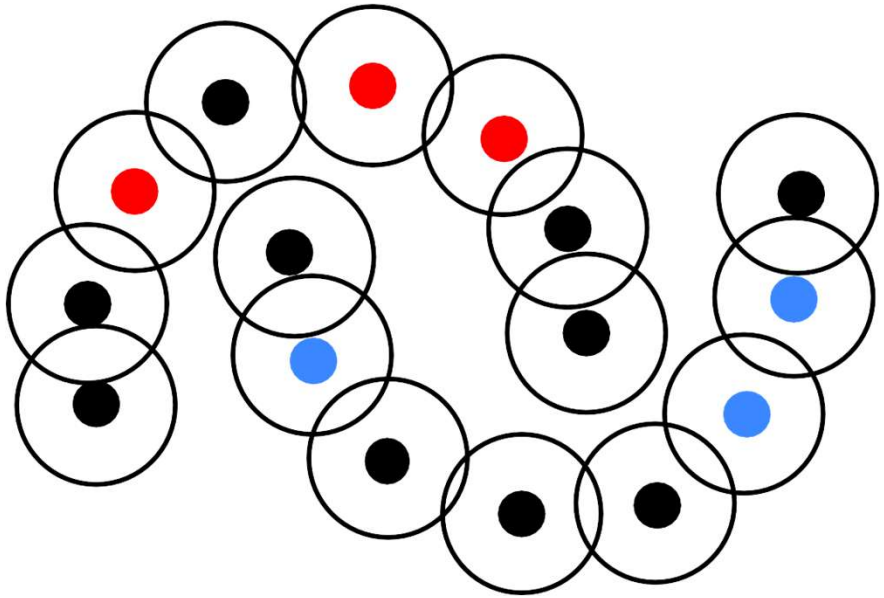
Source: Clark.2019

Consistency Regularization: Example



Source: Clark.2019

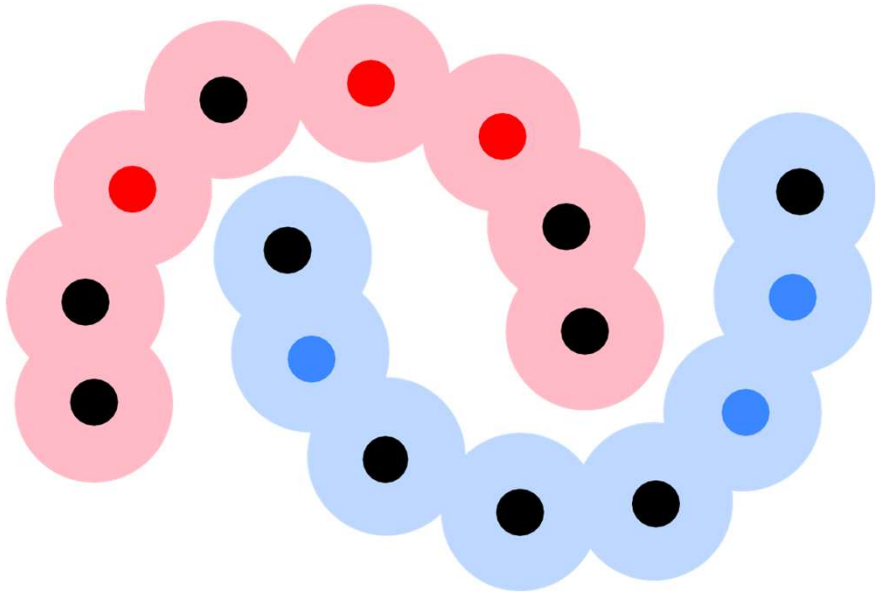
Consistency Regularization: Example



Model should produce the same predictions everywhere in the circles -> overlapping circles should have the same prediction

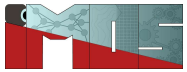
Source: Clark.2019

Consistency Regularization: Example



Decision boundary will look like this

Source: Clark.2019

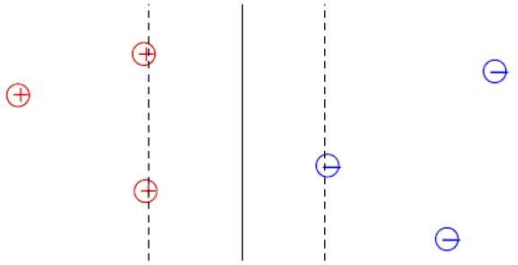


Semi-supervised SVMs(S3VMs)

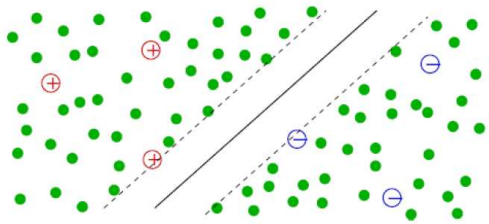
Olga Fink

Semi-supervised SVMs (S3VMs)

- SVMs



- Semi-supervised SVMs(S3VMs) = Transductive SVMs (TSVMs)



- Assumption: unlabeled data from different classes are separated by large margin
- Idea: The decision boundary shouldn't lie in the regions of high density

Source: Xiaojin Zhu 2007

S3VM objective function

How to incorporate unlabeled points?

Assign putative labels sign$(f(x))$ to $x \in X_u$

sign$(f(x))f(x) = |f(x)|$

The hinge loss on unlabeled points becomes

$$(1 - y_i f(x_i))_+ = (1 - |f(x_i)|)_+$$

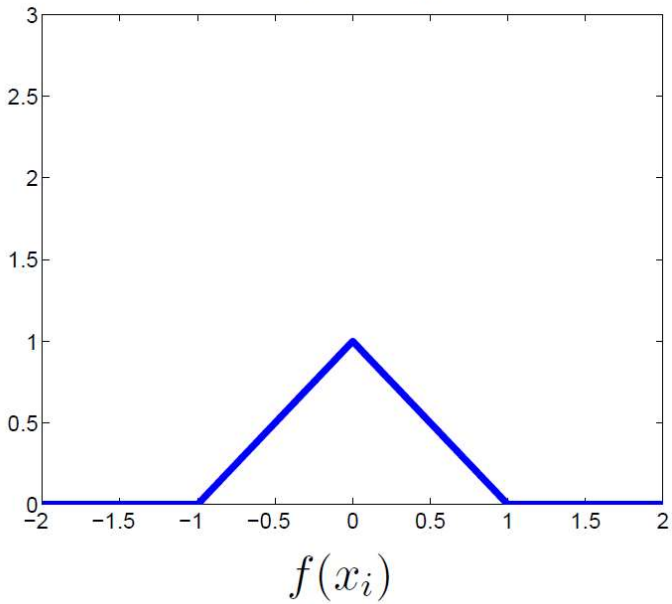
S3VM objective:

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

Source: Xiaojin Zhu 2007

The hat loss on unlabeled data

$$(1 - |f(x_i)|)_+$$



Prefers $f(x) \geq 1$ or $f(x) \leq -1$, i.e., unlabeled instance away from decision boundary $f(x) = 0$.

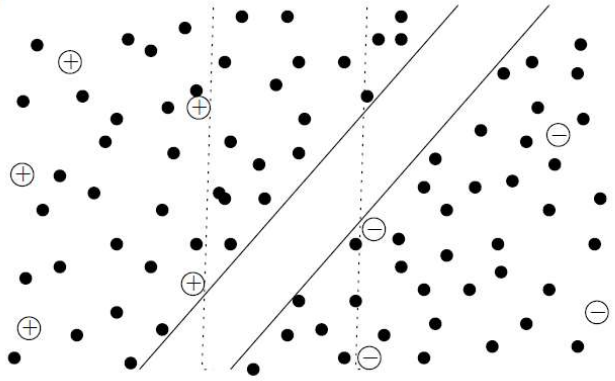
Source: Xiaojin Zhu 2007

Avoiding unlabeled data in the margin

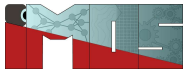
S3VM objective:

$$\min_f \sum_{i=1}^{l}(1-y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n}(1-|f(x_i)|)_+$$

the third term prefers unlabeled points outside the margin. Equivalently, the decision boundary $f = 0$ wants to be placed so that there is few unlabeled data near it.



Source: Xiaojin Zhu 2007

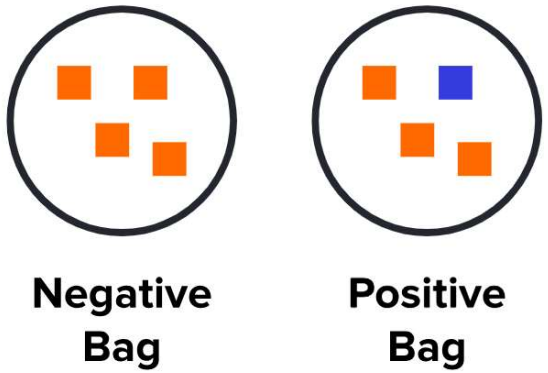


Weakly Supervised Learning: Multiple Instance Learning

Olga Fink

Multiple Instance Learning

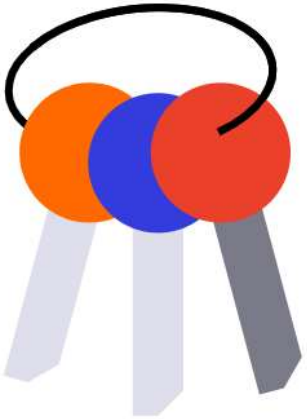
- Multiple Instance Learning (MIL) is a form of weakly supervised learning where the learning algorithm receives a set of labeled bags, each containing multiple instances. In this paradigm, the label is provided at the bag level rather than the instance level.
- In the standard MIL assumption, negative bags are said to contain **only negative instances**, while positive bags **contain at least one positive instance**. Positive instances are labeled in the literature as **witnesses.**



Source: https://nilg.ai/

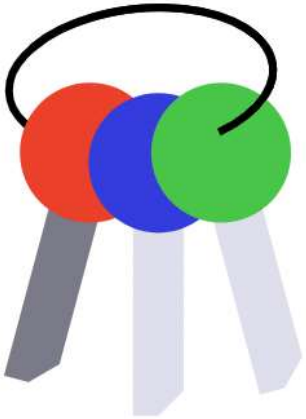
Basic idea

Serge's key-chain



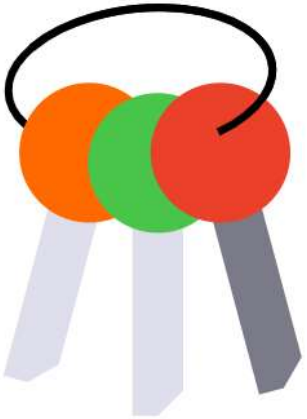
Serge **cannot** enter the Secret Room

Sanjoy's key-chain



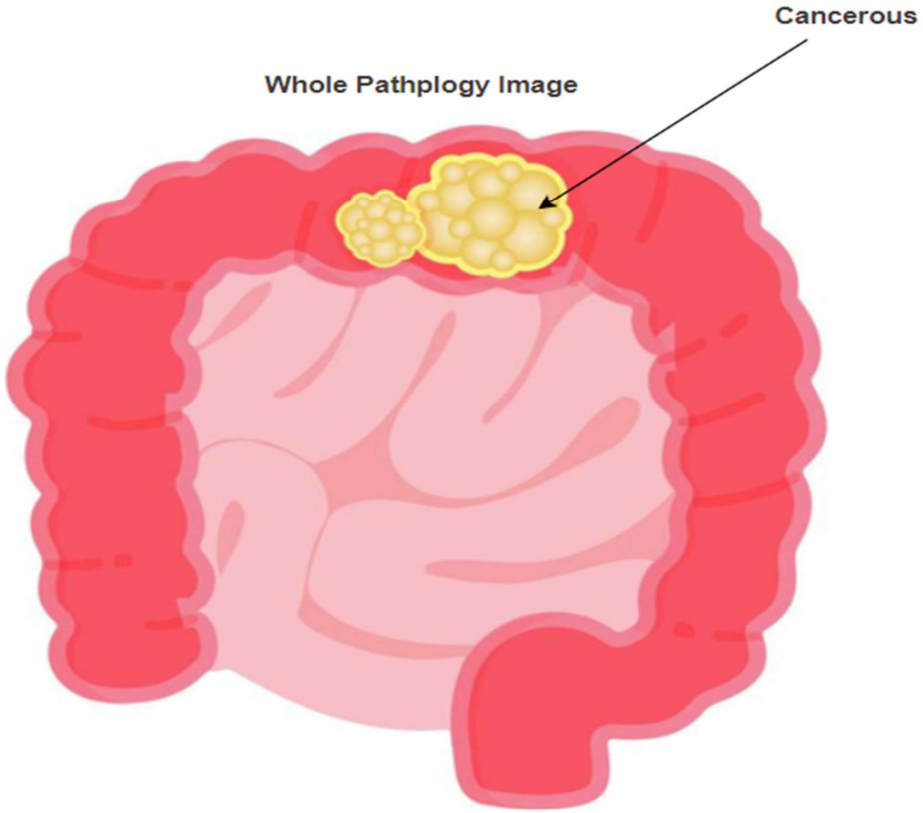
Sanjoy **can** enter the Secret Room

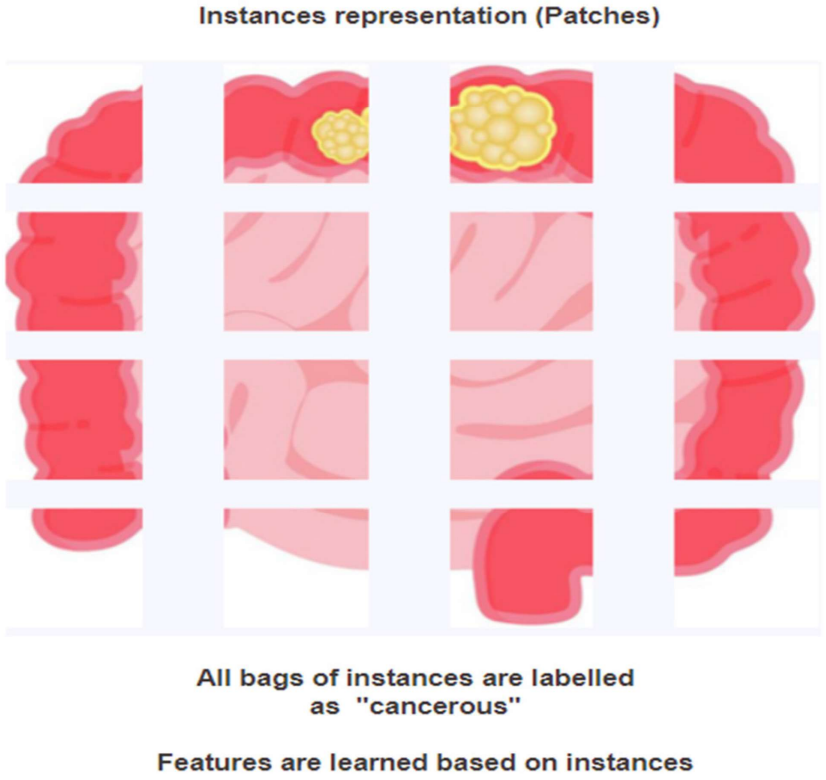
Lawrence's key-chain



Lawrence **can** enter the Secret Room

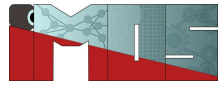
Example





Fatima, Samman, Sikandar Ali, and Hee-Cheol Kim. "A comprehensive review on multiple instance learning." *Electronics* 12.20 (2023): 4323.

MIL: Basics



1. Bags and Instances

- **Bags**: In MIL, a "bag" is a collection of instances. Each bag is labeled as a whole, without specifying the labels of the individual instances within it.
- **Instances**: Each bag contains multiple instances, which are individual data points. The instances can be anything from images, segments of text, or features extracted from a set of data.

2. Bag Labels

- The label of a bag is typically binary (positive or negative). The most common assumption in MIL is the "standard MIL assumption":
  - **Positive Bags**: If a bag is labeled positive, it implies that at least one instance in the bag is positive.
  - **Negative Bags**: If a bag is labeled negative, it implies that all instances in the bag are negative.
- This labeling scheme simplifies the task of learning from complex data structures where only partial knowledge of the correct labels is available.

3. Learning Objective

- The goal in MIL is to learn a model that can accurately predict the labels of unseen bags based on the patterns learned from the training set. This often involves identifying which instances in positive bags are contributing to the bag's positive label.

MIL: Basics

4. Applications of MIL

- **Medical Imaging**: In tasks like cancer detection, a bag may represent a patient's set of medical images, and the bag is labeled positive if any image shows signs of cancer.
- **Document Classification**: A document can be considered a bag, and sentences or paragraphs can be instances. The document is classified based on the content of its parts.
- **Object Detection**: Images can be bags with patches as instances. An image is labeled as containing an object (positive) if at least one patch contains the object.

5. Challenges in MIL

- **Ambiguity in Instance Labels**: Determining which instances are responsible for the label of the bag can be challenging, especially in the case of positive bags.
- **Variability of Instances**: Instances within the same bag can vary significantly, adding complexity to the learning process.

6. Approaches to MIL

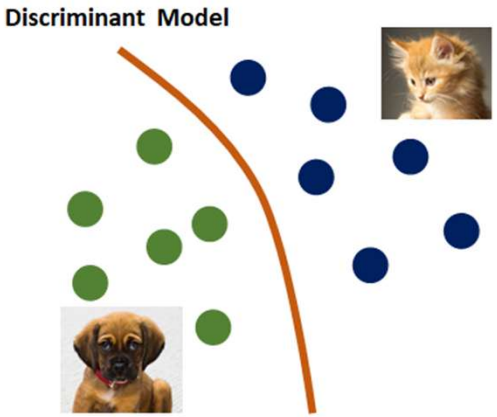
- **Instance-Based Approaches**: These methods focus on estimating labels for individual instances and then aggregate these to make a bag-level prediction.
- **Bag-Based Approaches**: These methods derive features from the entire bag and use these features to predict the bag's label directly.
- **Embedded Space Approaches**: These techniques transform instances into a new feature space where traditional machine learning algorithms can be applied more effectively.

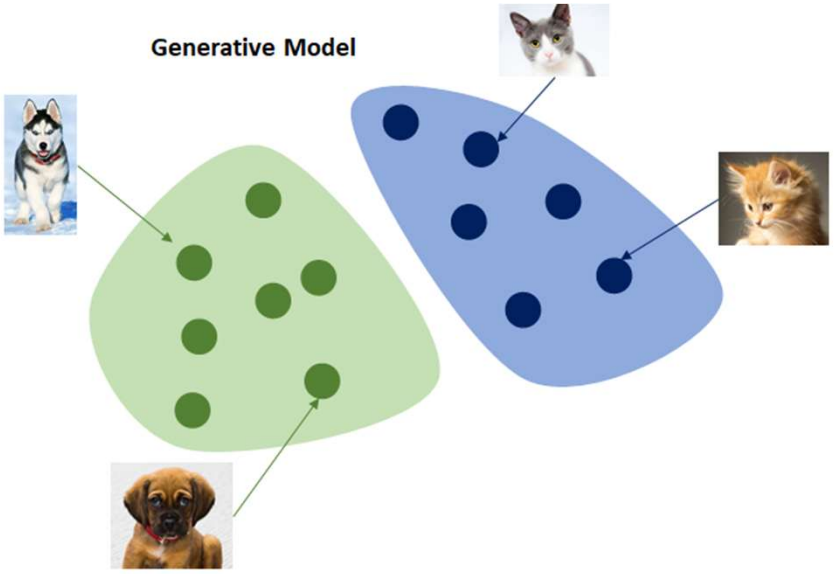
Generative ML models

Olga Fink

Discriminative VS Generative

- Discriminative
  - Model P(y|x)
  - Learn the boundary between classes (in classifiers)
  - Usually better performance in low-data regimes
- Generative
  - Model P(x,y)
  - Model the distribution of classes
  - Usually needs more data
  - Can be used to generate new samples

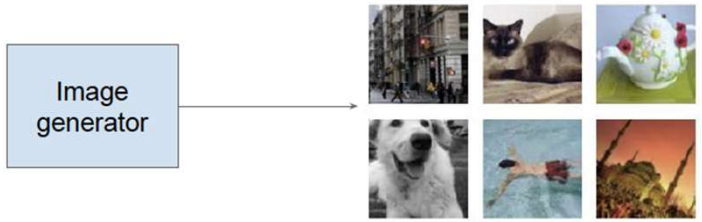




Why generative models?

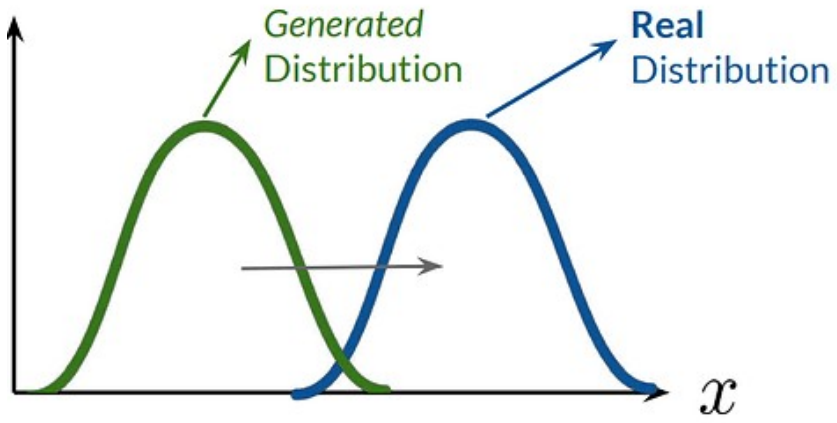
- Model complex and high-dimensional distributions
- Generate realistic synthetic samples
  - Data augmentation
  - Simulation scenarios for learning algorithms
- Fill the blanks in the data
  - Manipulate real samples
- Learn a latent representation useful for other tasks

original enhanced



Generative Modeling

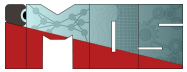
- **Goal of generative modeling**
  - **Learning the underlying distribution of data**



Input ~ $P_{data}(x)$

Output~$P_{model}(x)$

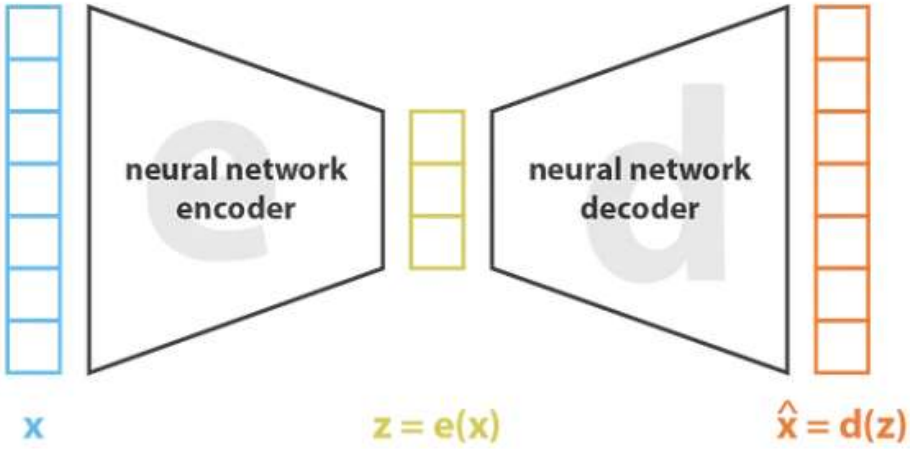
Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.



Variational Autoencoders

Olga Fink

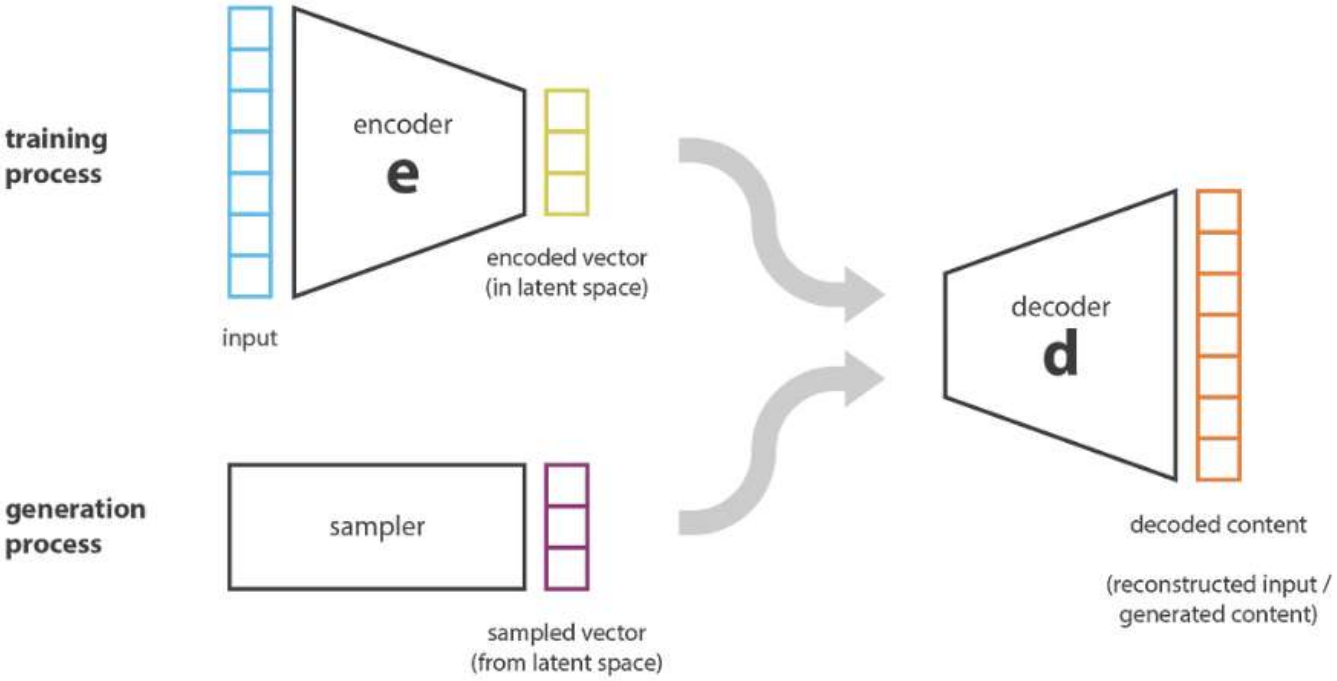
Recap: Autoencoders



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

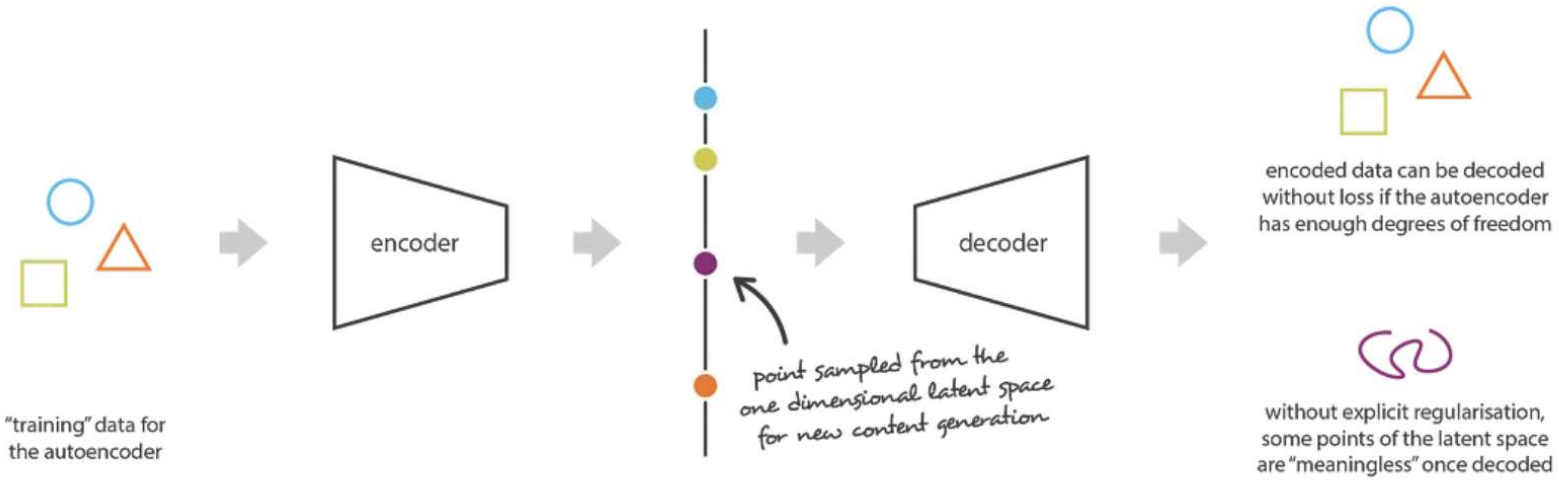
Source: https://towardsdatascience.com

Potential generation process



Source: https://towardsdatascience.com

AE for data generation

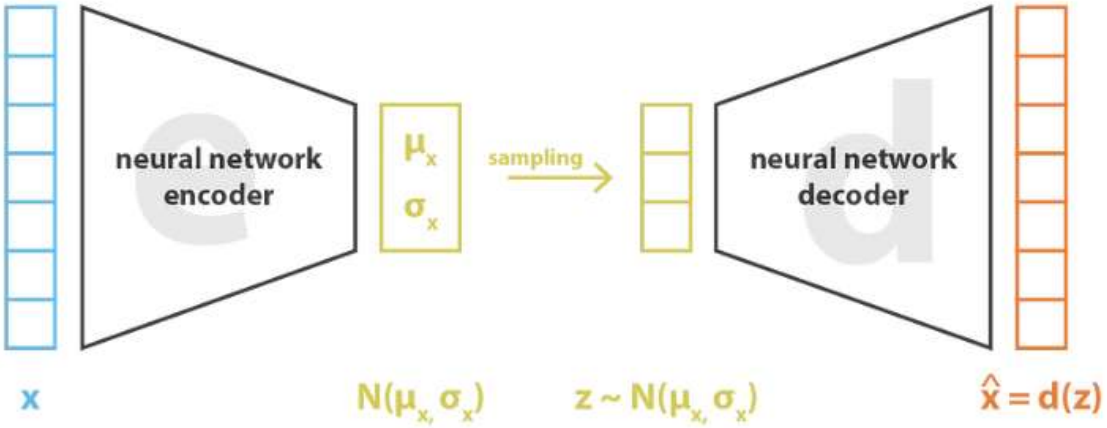


Source: https://towardsdatascience.com

Variational Autoencoders

- Definition: Variational Autoencoders are a type of generative model that use machine learning to produce new data points that are statistically similar to a given dataset.
- Key Concept: VAEs are based on the principles of probability and statistics, aiming to model the underlying data distribution.
- Purpose of VAEs
  - Data Generation: Generate new data instances that mimic the original data.
  - Dimensionality Reduction: Compress data into a more manageable latent space.

General structure of VAE



$$\text{loss} = \| x - \hat{x} \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,] = \| x - d(z) \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

Source: https://towardsdatascience.com

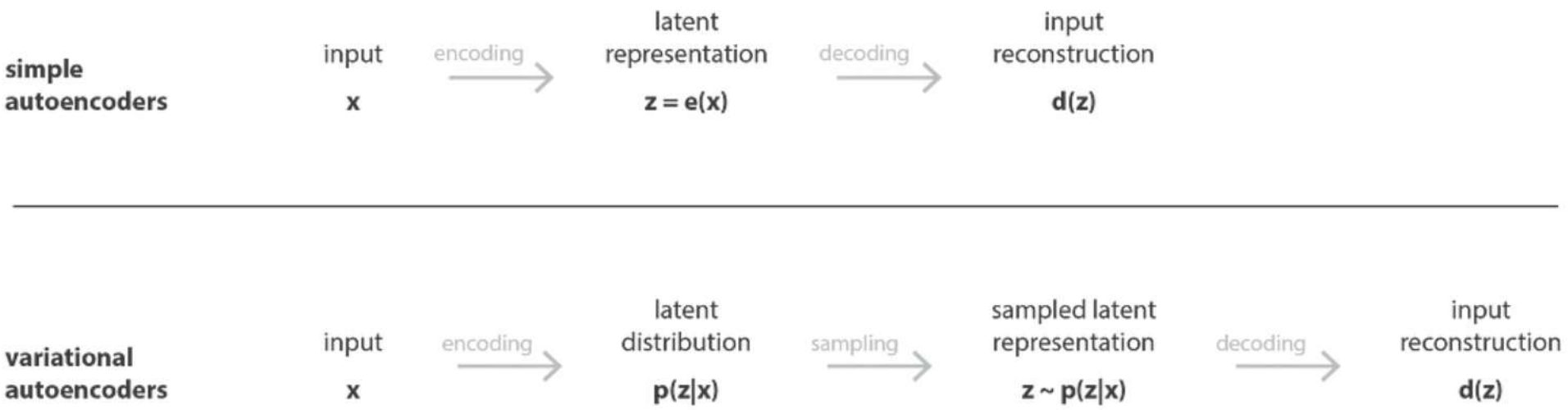
Objective function of VAE

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

Reconstruction loss

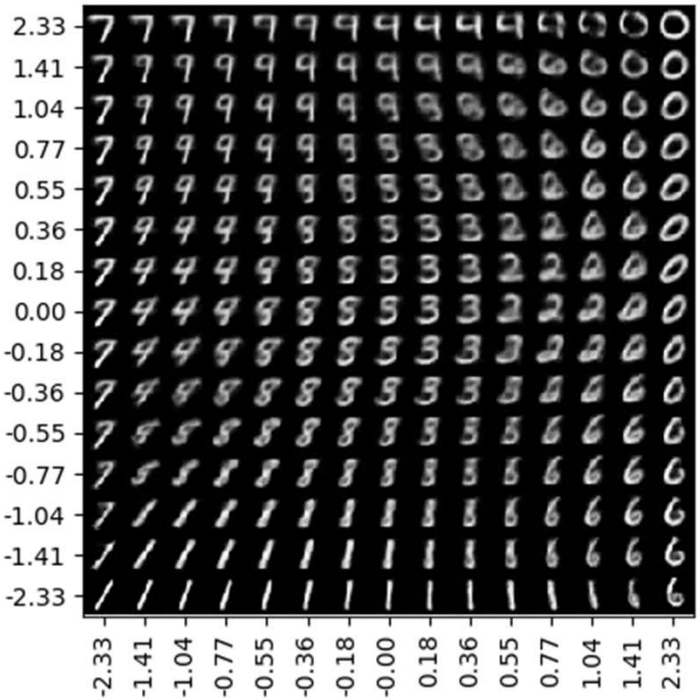
KL Divergence Loss

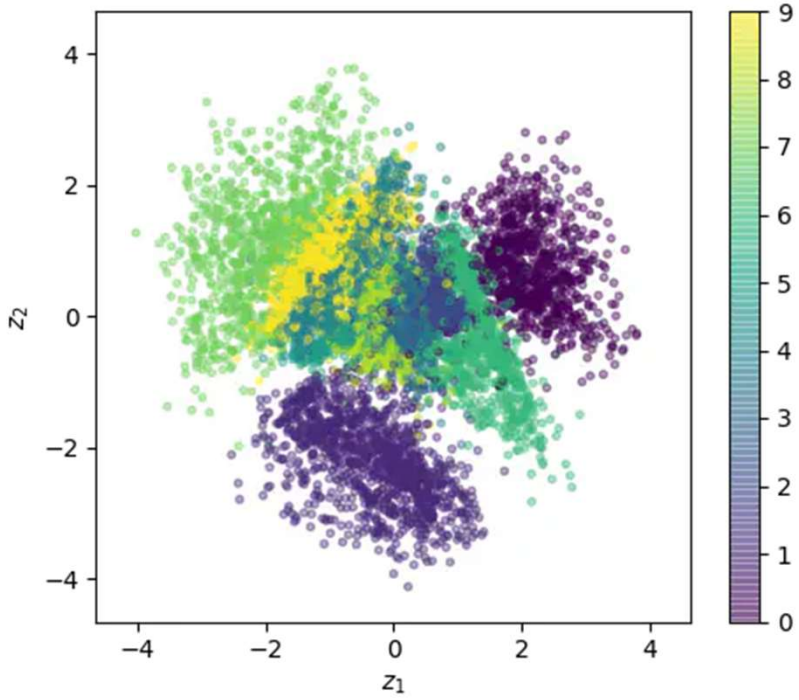
Simple AE vs. VAE



Source: https://towardsdatascience.com

VAE on MNIST

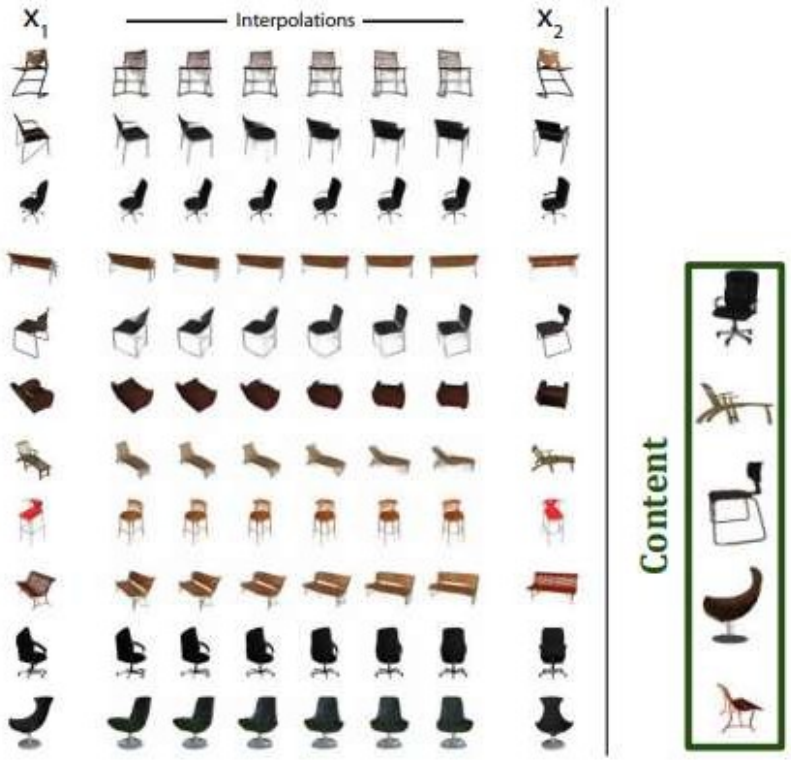

2.33
1.41
1.04
0.77
0.55
0.36
0.18
0.00
-0.18
-0.36
-0.55
-0.77
-1.04
-1.41
-2.33
-2.33
-1.41
-1.04
-0.77
-0.55
-0.36
-0.18
-0.00
0.18
0.36
0.55
0.77
1.04
1.41
2.33

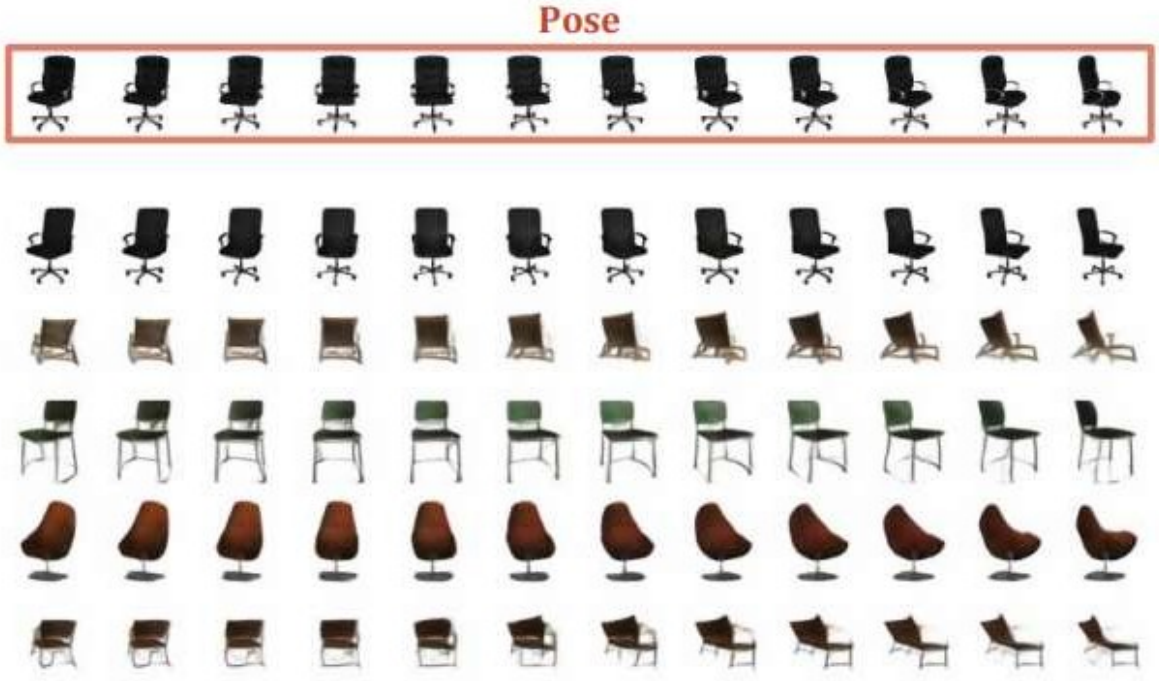

z_2
z_1
4
2
0
−2
−4
−4
−2
0
2
4
9
8
7
6
5
4
3
2
1
0

Idea of dientanglement

- The concept of disentanglement is based on the hypothesis that real-world data is generated by a few independent explanatory factors of variation
- Can be sued for controlled data generation:
  - learn a disentangled feature representation of the data
  - use these disentangled features representing independent factors of variation to generate data samples with desired characteristics in controlled ways

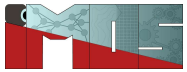
Disentangled features: generation





Denton, Emily L. "Unsupervised learning of disentangled representations from video." *Advances in neural information processing systems* 30 (2017).

EPFL



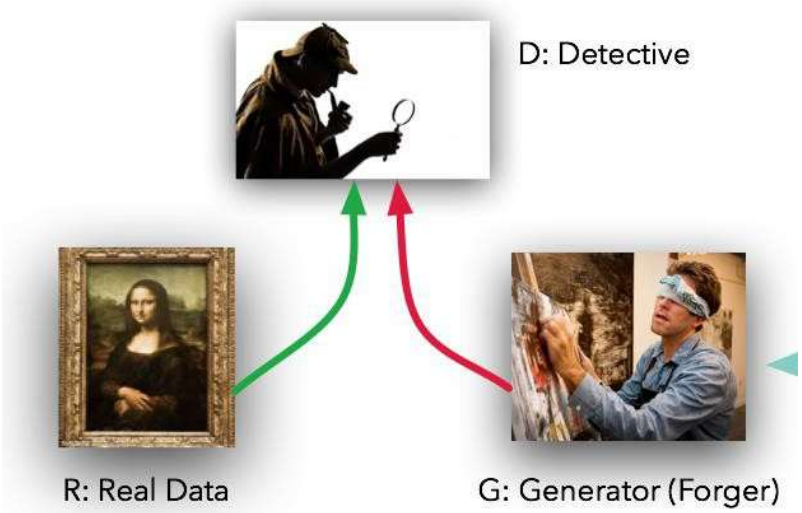
Generative Adversarial Networks (GANs)

Olga Fink

GANs

- 2014: a major milestone in generative models
- GAN: Generative Adversarial Network
- Take a look at http://thispersondoesnotexist.com by Style-GAN

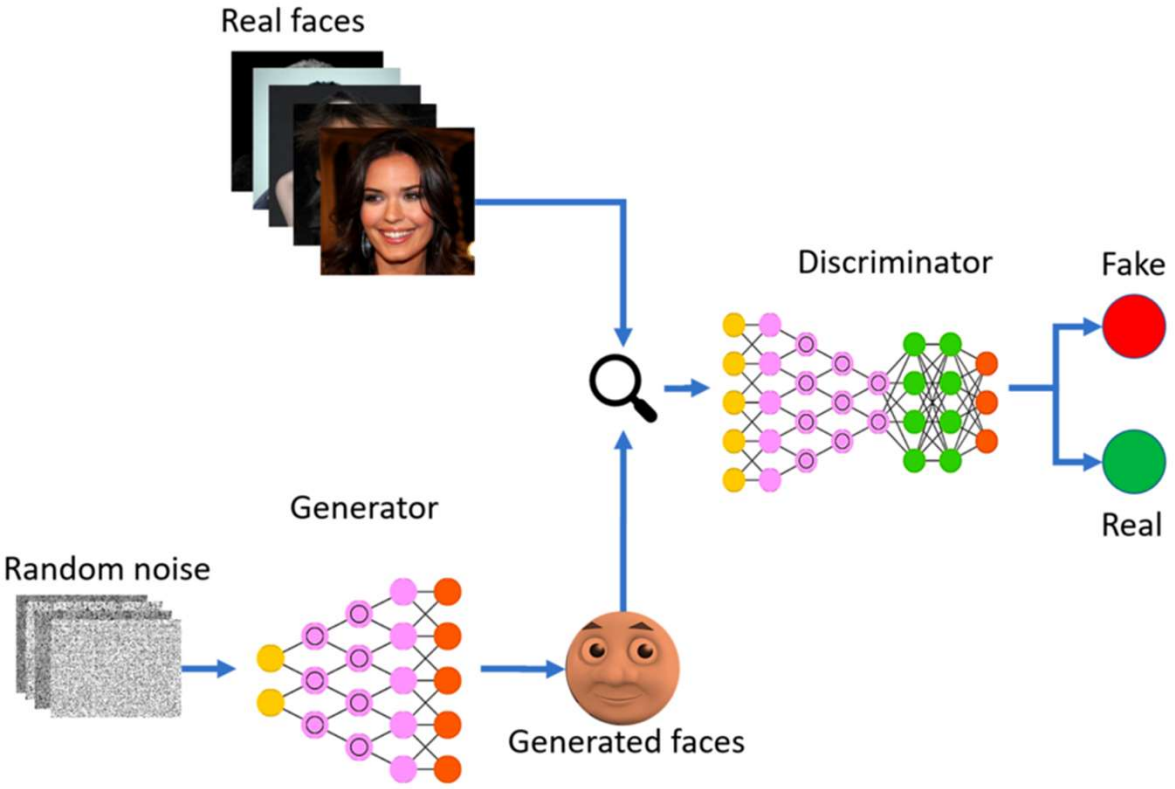
Adversarial Training



Generative Adversarial Networks (GAN)

Figure 1 https://github.com/devnag/pytorch-generative-adversarial-networks

GANs



Source: https://medium.com/sigmoid/a-brief-introduction-to-gans-and-how-to-code-them-2620ee465c30

GAN loss functions

Min-Max Loss:

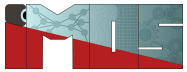
$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

Discriminator Loss:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

Generator Loss:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$



Example

Olga Fink

Different DA setups

Partial DA under «extreme» setup

Source - Train	Target - Train	Target - Test
Healthy	Healthy	Healthy
Fault 1		Fault 1
Fault 2		Fault 2
Fault 3		Fault 3
Fault x		Fault x

Open-Partial DA for Fault Diagnosis

Source - Train	Target - Train	Target - Test	Source - Test
Healthy	Healthy	Healthy	Healthy
Fault 1	---	Fault 1	Fault 1
Fault 2	---	Fault 2	Fault 2
----	Fault 3	Fault 3	Fault 3
----	Fault 4	Fault 4	Fault 4

Challenges / requirements for synthetic faults

- Need to be physically plausible / interpretable
- Need to be specific to the considered system and specific to the operating conditions
- Usually, samples of all fault types in one system cannot be assumed! 

Controlled Generation of Unseen Faults

Generate unseen Faults in a new domain (new unit or new operating condition) and “exchange fault patterns between different domains”

→ Open-Partial Domain adaptation

General Idea

- Hypothesis: faulty signal can be represented by combining
  - domain-specific variations of the normal steady state operation (represented by healthy data)
  - and a domain-independent signal representing solely the characteristic from a fault.
- Hypothesis in the Fourier domain:
  - Fourier spectrum of fault data can be expressed as the sum of

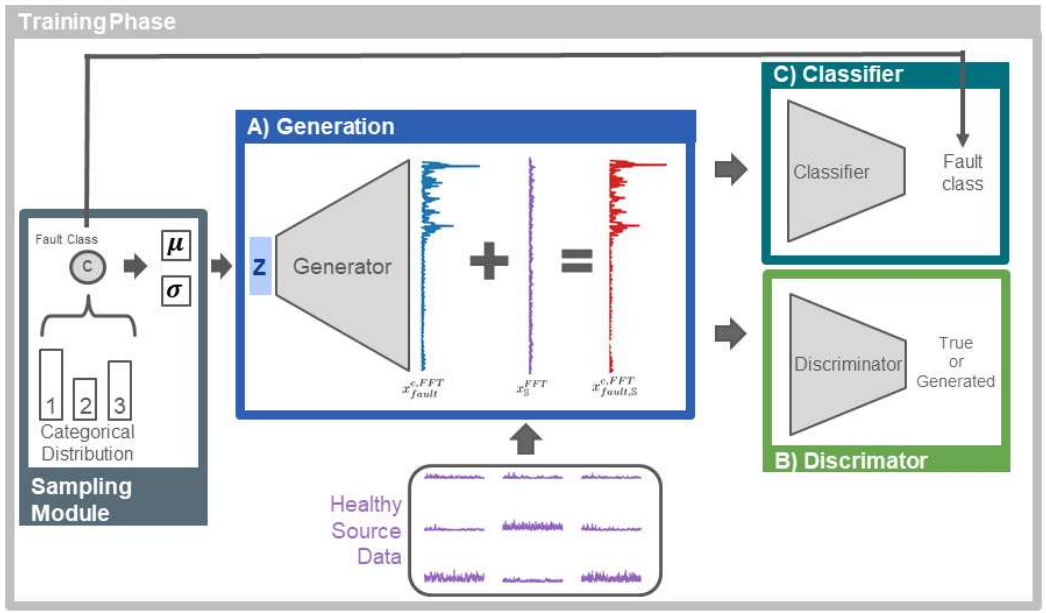
  1) the spectrum of a signal from normal operation and

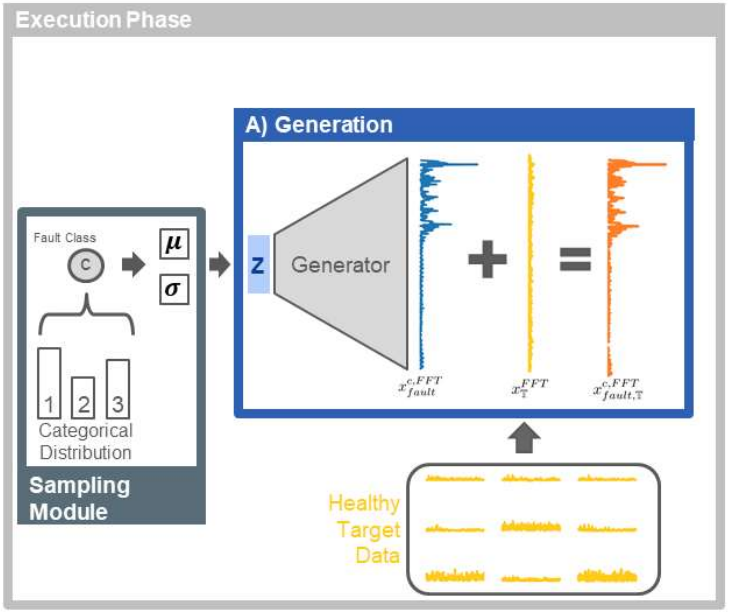
  2) the spectrum of a signal representing the domain-independent faulty condition.

$$x_{fault,\mathbb{X}}^{c,FFT} = x_{\mathbb{X}}^{FFT} + w * x_{fault}^{c,FFT}$$

Rombach, K., G. Michau & O. Fink: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

FaultSignatureGAN





Training Phase:

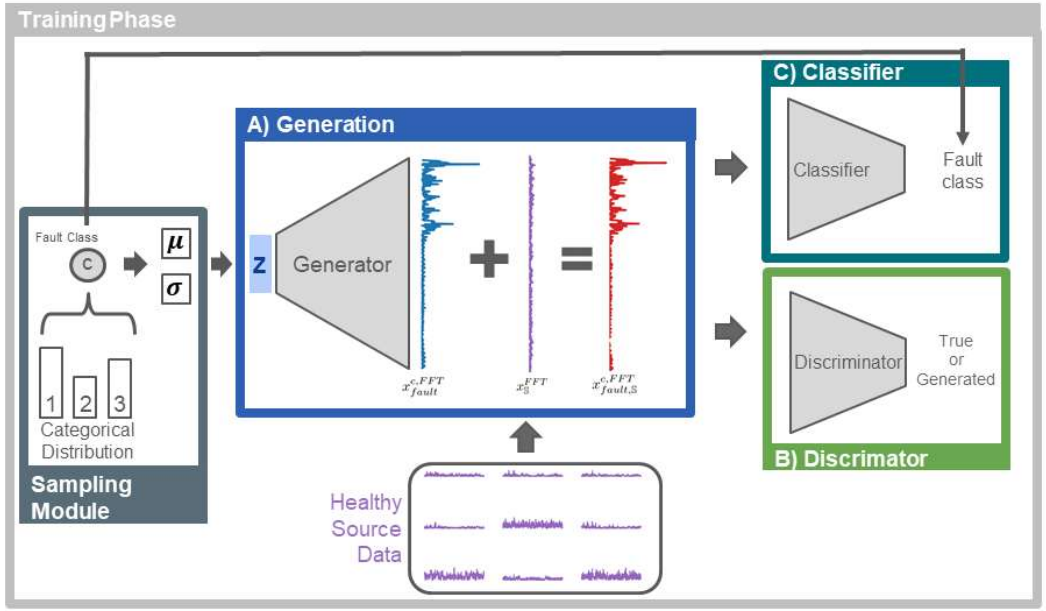
→ Training the A) generative model to generate domain independent fault characteristics while imposing B) plausibility with the discriminator in the source domain and C) semantic consistency with the classifier.

Execution Phase

→ generation of unseen target data

Rombach, K., G. Michau & O. Fink: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

FaultSignatureGAN



Training Phase:

→ Training the A) generative model to generate domain independent fault characteristics while imposing B) plausibility with the discriminator in the source domain and C) semantic consistency with the classifier.

Execution Phase

→ generation of unseen target data

Rombach, K., G. Michau & O. Fink: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

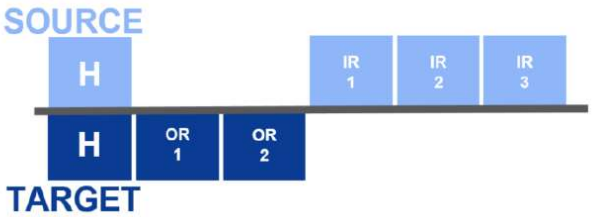
Dataset: Paderborn Bearing

Domain	Rotational speed [rpm]	Load Torque [Nm]	Radial Force [N]
0	1500	0.7	1000
1	900	0.7	1000
2	1500	0.1	1000
3	1500	0.7	400

- Six Classes
  One healthy class, two OR fault severities and three IR fault severities
- Four Domains
- Generate faulty data in different domains
- Domain 1: differs significantly from the other conditions

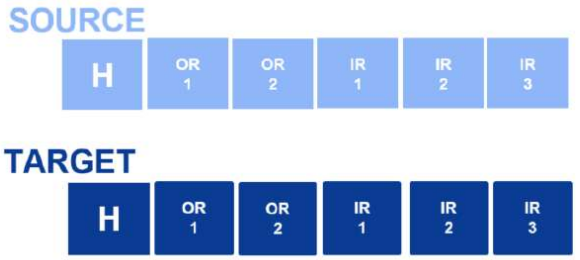
Open-Partial Setup for the Paderborn Dataset

Real datasets



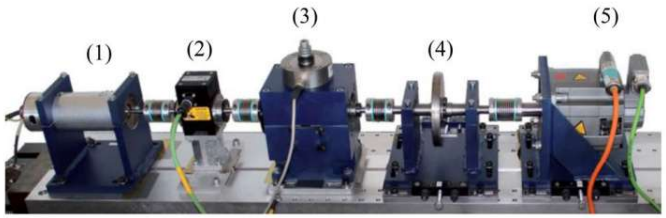
Real and generated training datasets

Real test datasets



Rombach, K., G. Michau & **O. Fink**: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

Transfer between OCs knowing the health + 1 OC in each of the datasets (without OC1)



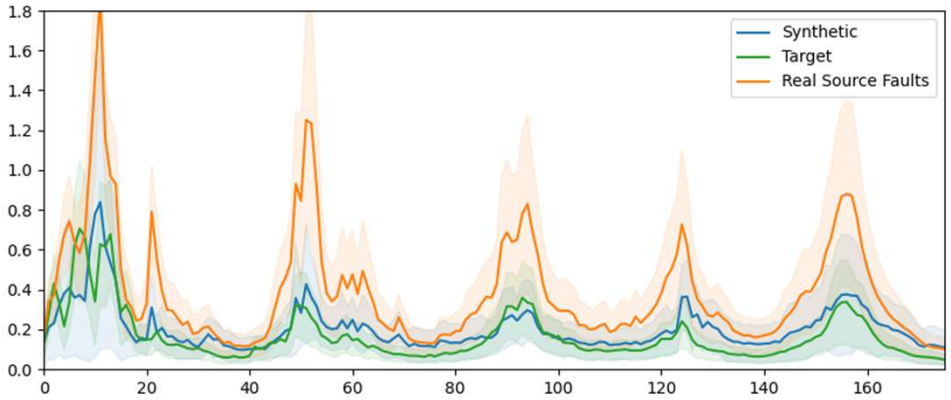
Paderborn Dataset

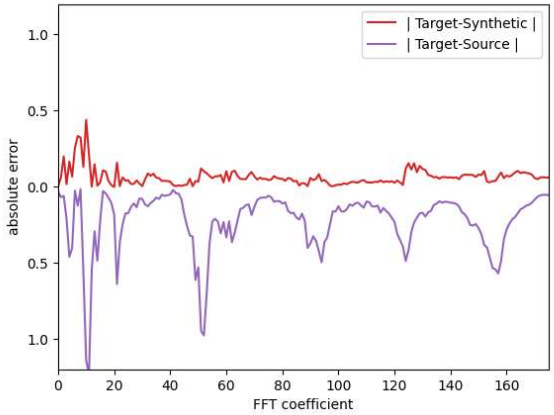
	Baseline	FaultSignature GAN
Mean Accuracy (over all test datasets and transfer directions)	83.4%	**96.0%**

*The test datasets comprise the real missing fault data as well as of a 30% of known health conditions

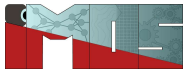
Rombach, K., G. Michau & **O. Fink**: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

Paderborn data visualization of the OR severity 1 fault comparing real fault data with generated fault data





Rombach, K., G. Michau & **O. Fink**: Controlled Generation of Unseen Faults for Partial and Open-Partial Domain Adaptation, Reliability Engineering and System Safety

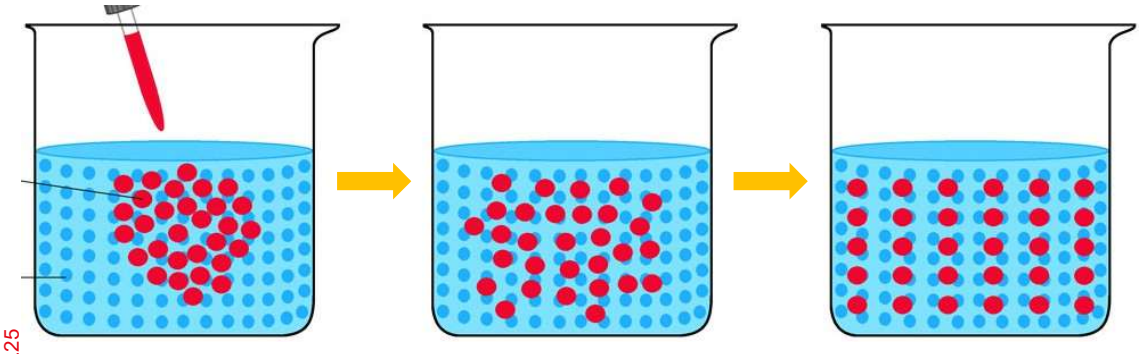


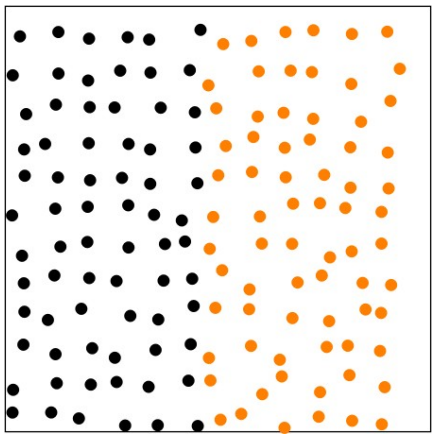
Diffusion models

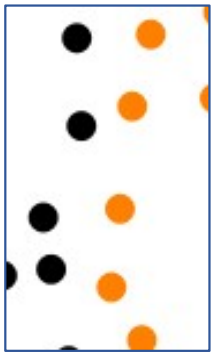
Olga Fink

Diffusion Process

- Diffusion models are inspired by non-equilibrium thermodynamics.
- For a small fraction of the time, it is difficult to determine whether particles are moving in the direction of mixing or in the opposite direction.

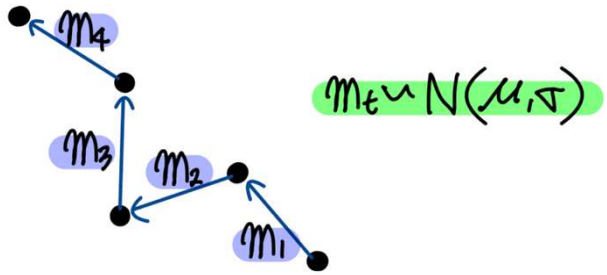




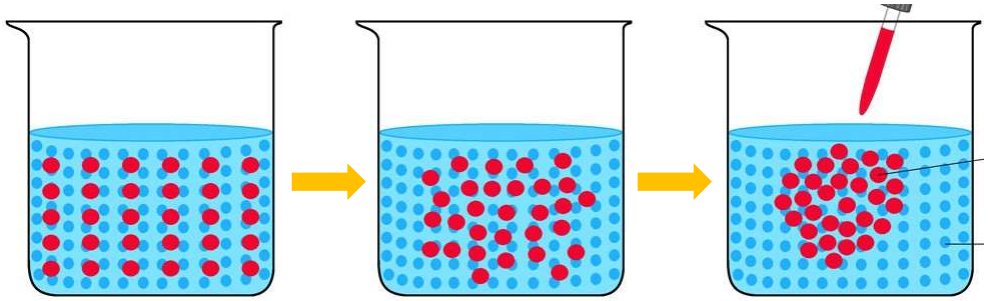


Source: Jumin Lee

Diffusion Process



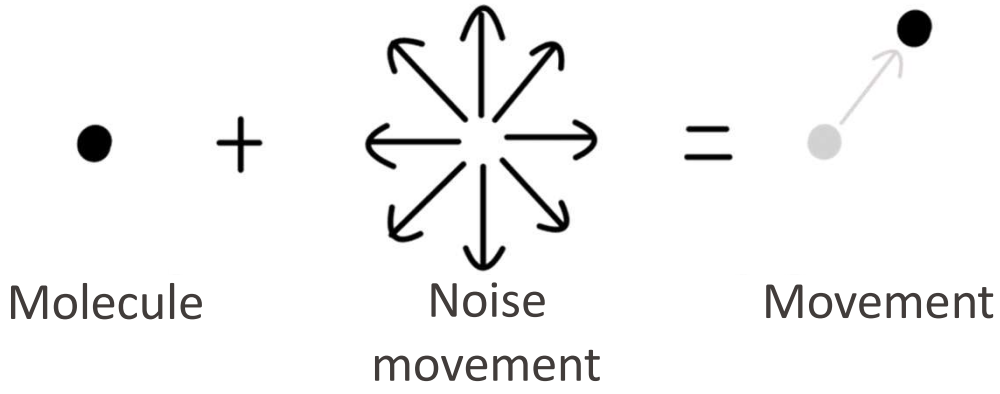
- If we look at the movement of a single molecule on a very short time scale, it follows a Gaussian distribution.
- Since the direction of mixing and the opposite direction are the same in a very short time, the opposite direction also follows a Gaussian distribution.

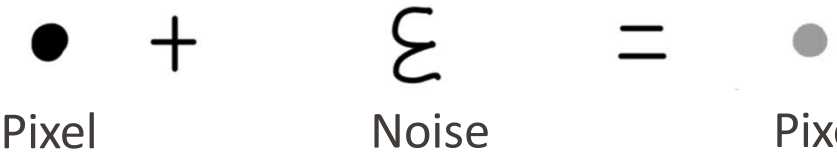


Source: Jumin Lee

Diffusion Process

- Just as we viewed the molecule's motion as a Gaussian-distributed noise, we add a Gaussian-distributed noise to the pixel.





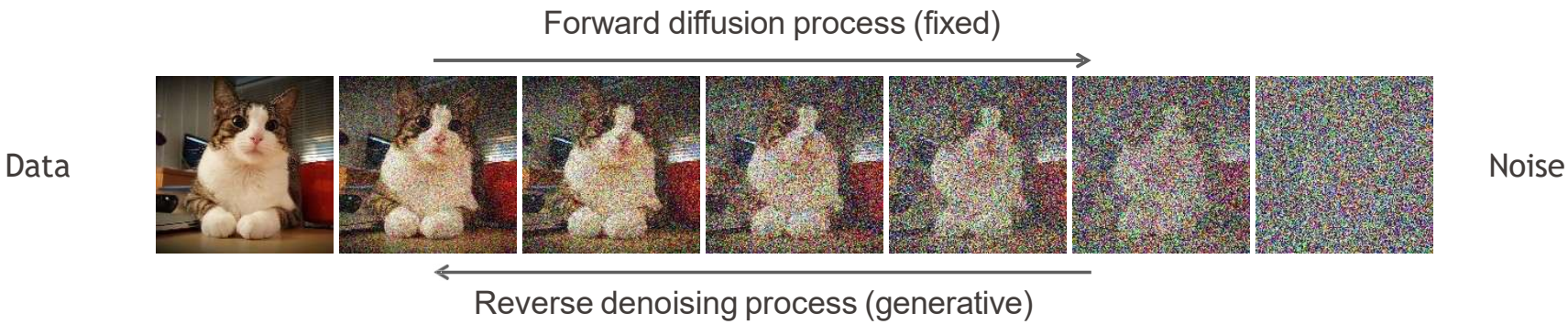
Source: Jumin Lee

Olga Fink

Denoising Diffusion Models

Denoising diffusion models consist of two processes:

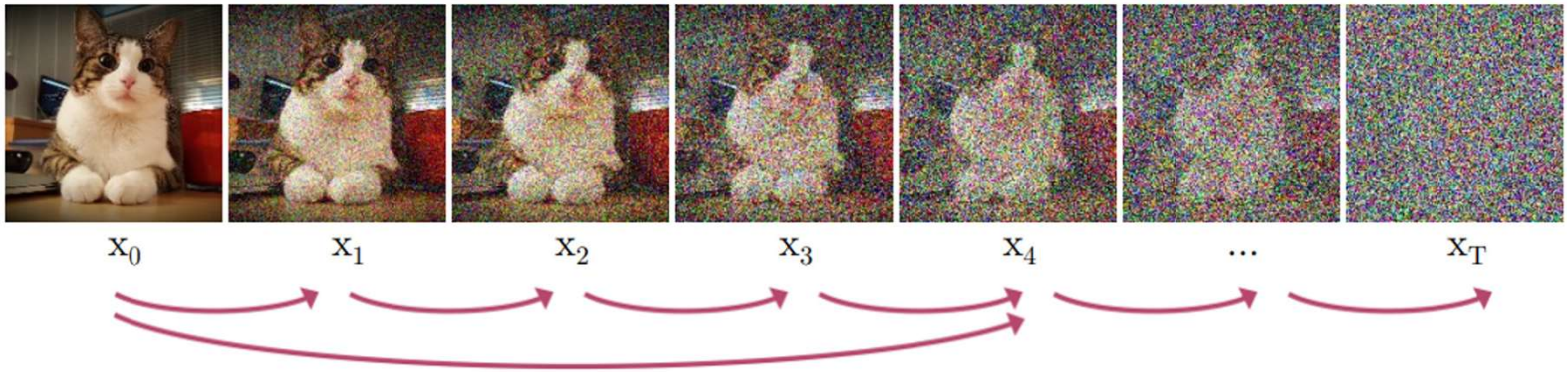
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Source: Jumin Lee

Forward Diffusion Process

The formal definition of the forward process in T steps:



Markov Property

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$ ← **Diffusion Kernel**

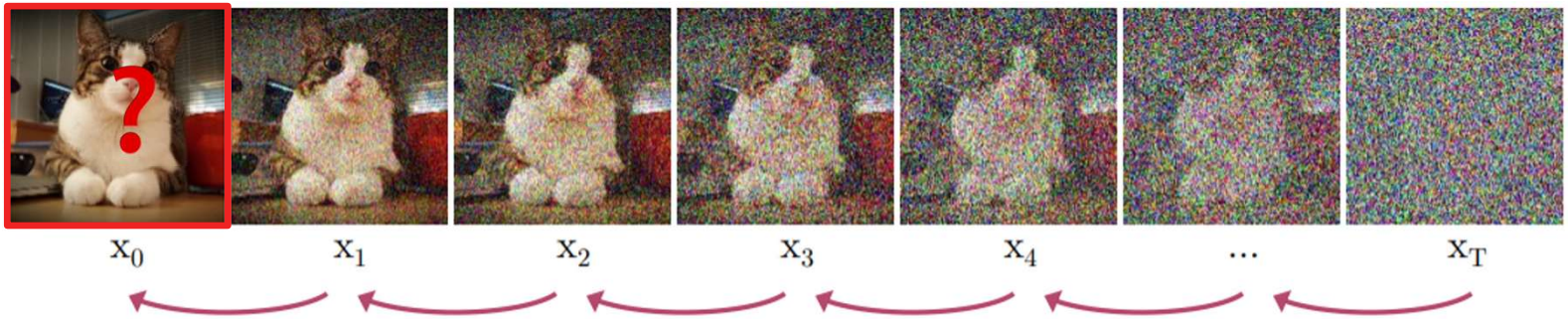
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\alpha_t := 1 - \beta_t \text{ and } \bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$$

Source: Jumin Lee

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

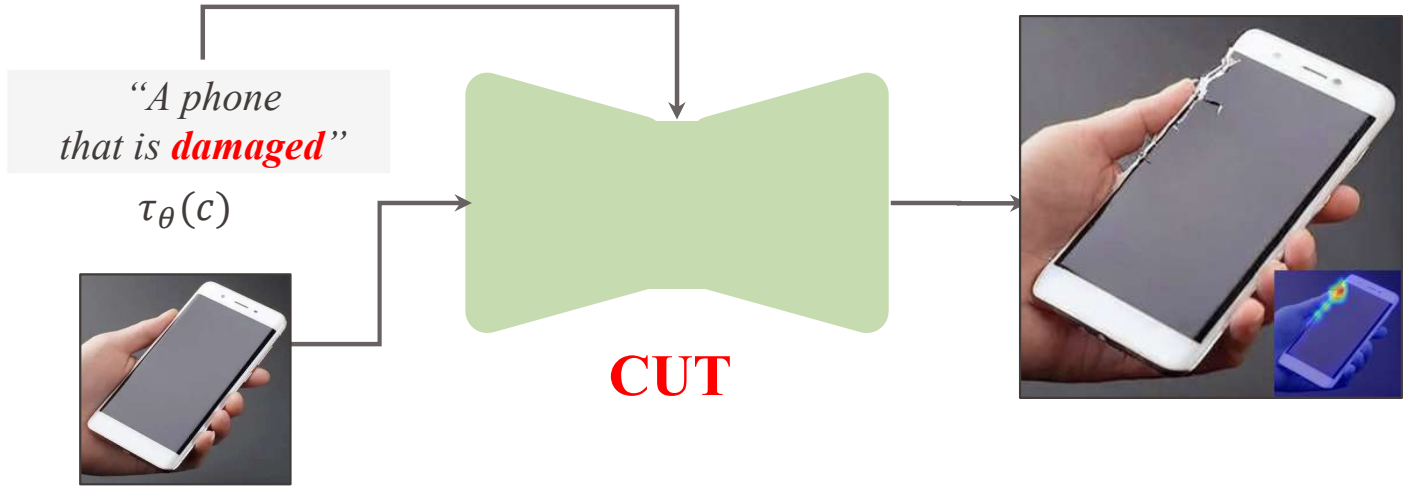
Model

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Source: Jumin Lee

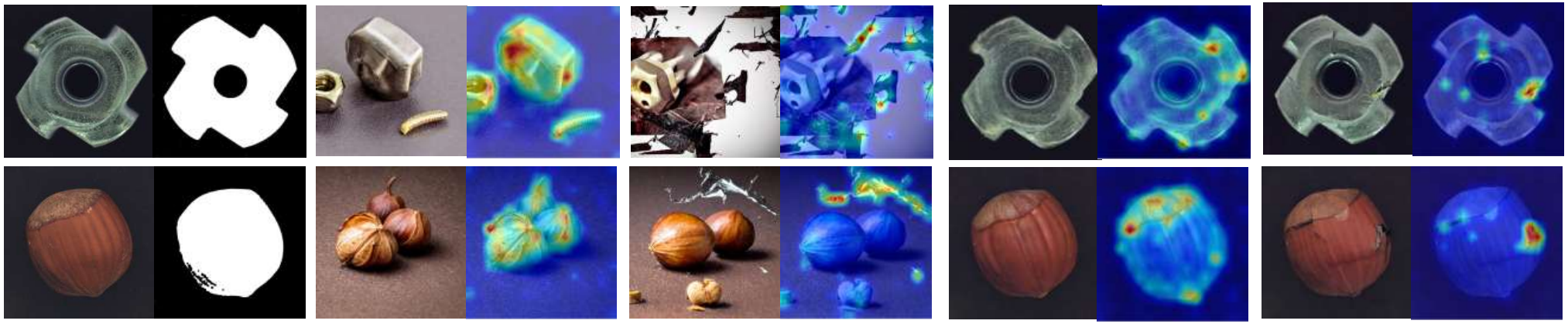
CUT

- A **C**ontrollable, **U**niversal, and **T**raining-Free Visual Anomaly Generation Framework
  - Controlled by sample image and text description
  - Applicable to any objects and anomaly types
  - No model training is required



Sun, H., Cao, Y., Dong, H. and Fink, O., 2025. Unseen Visual Anomaly Generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (pp. 25508-25517).

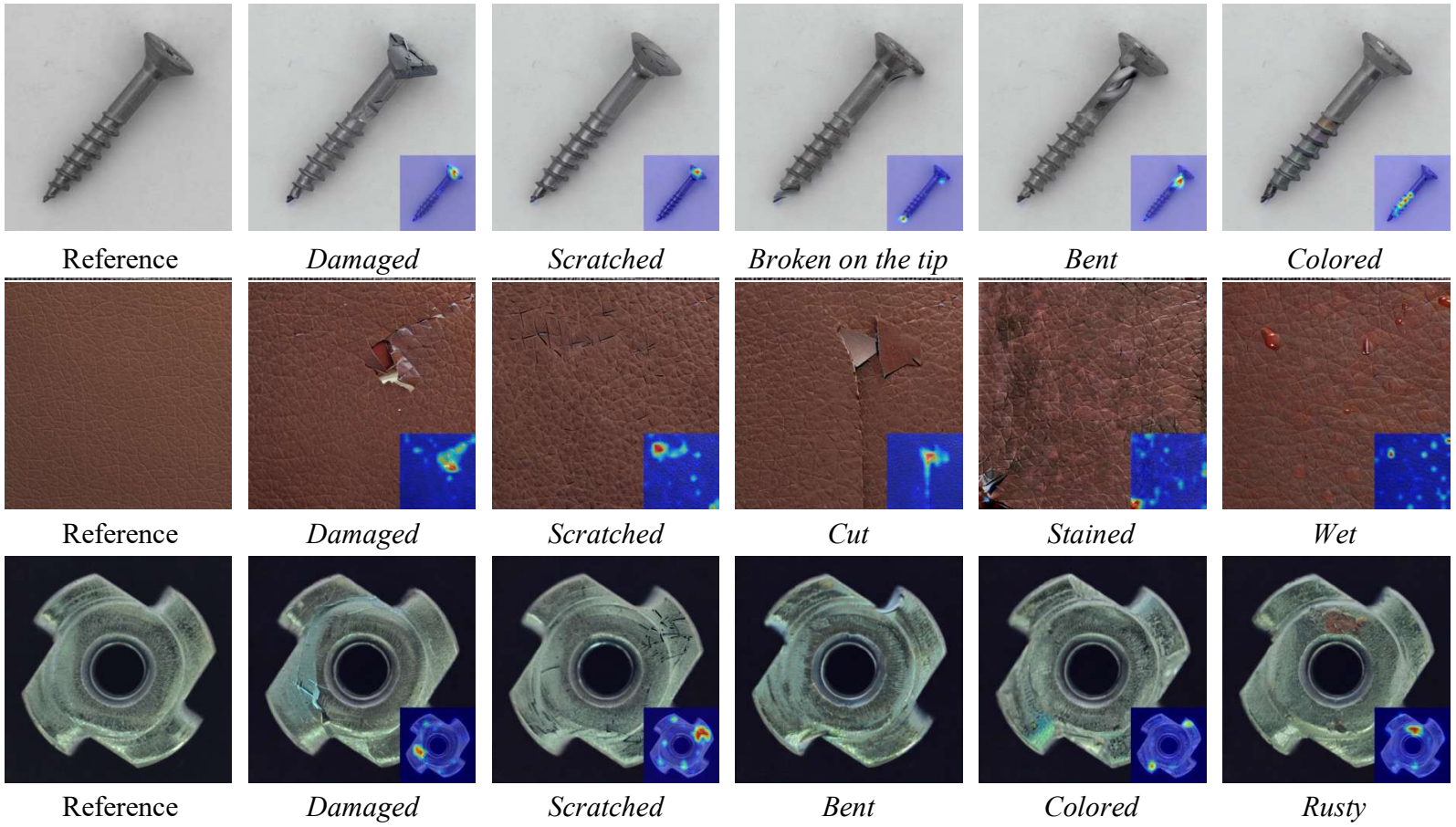
Generation with desired object and anomaly type



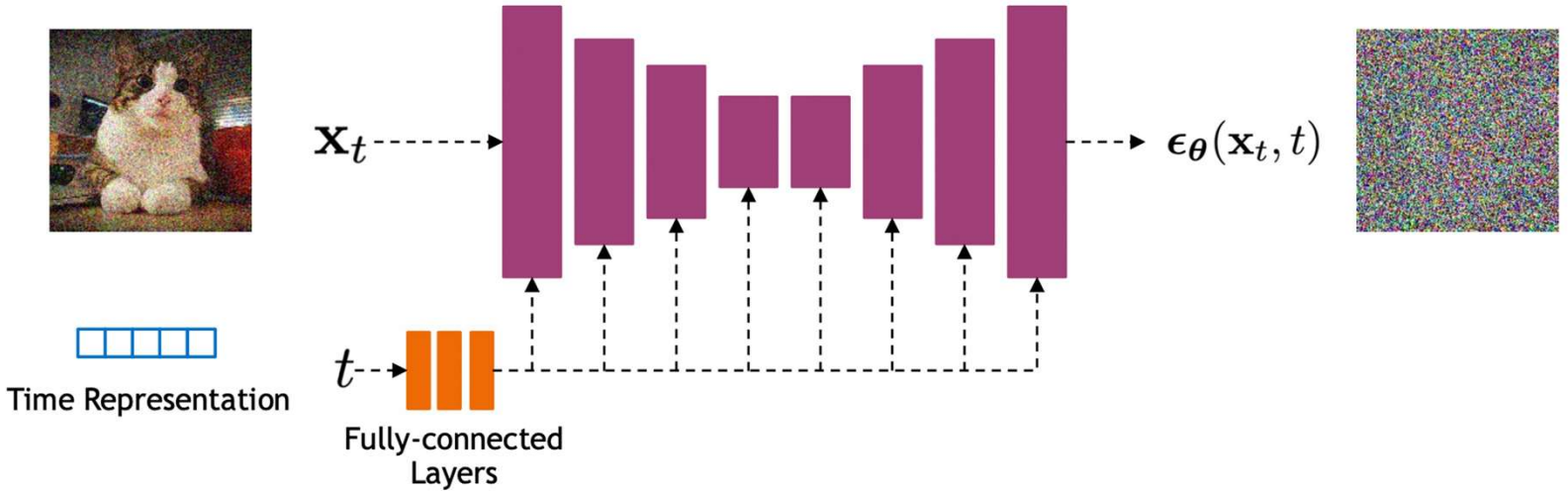
(a) Normal image and mask | (b) Stable Diffusion | (c) w/o image guidance | (d) w/o attention optimization | (e) ours

Sun, H., Cao, Y., Dong, H. and Fink, O., 2025. Unseen Visual Anomaly Generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (pp. 25508-25517).

Generation with desired object and anomaly type



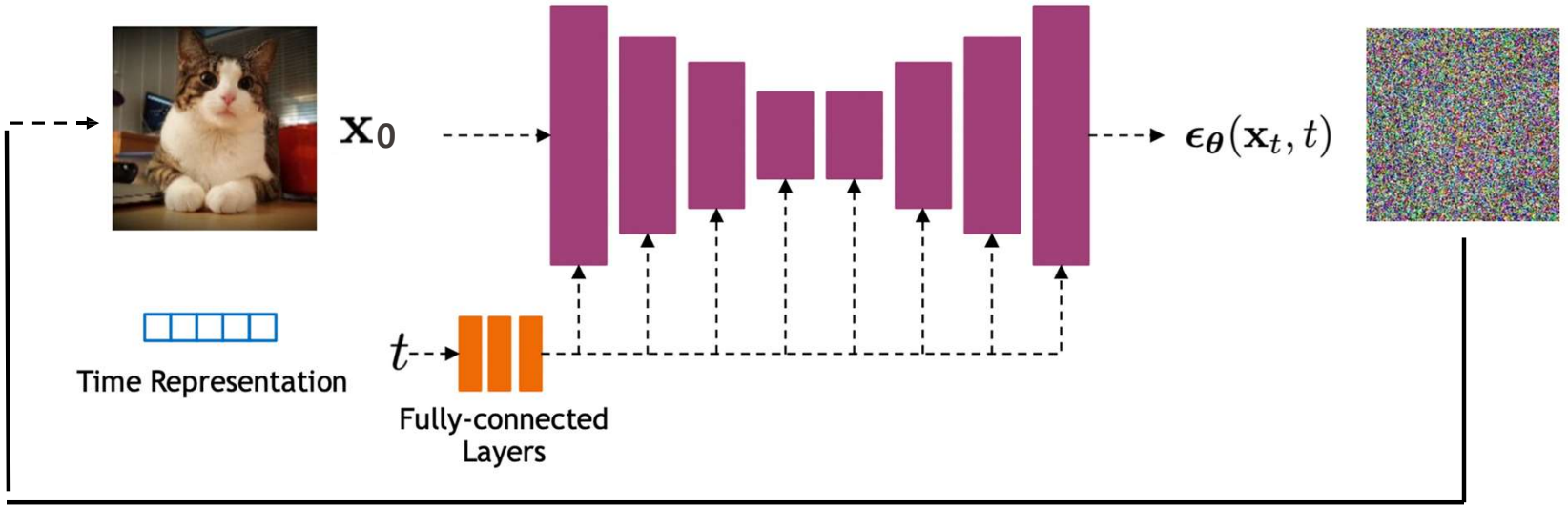
Denoising Diffusion Models : Training



Algorithm 1 Training

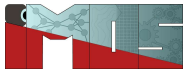
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \mathbf{z}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Denoising Diffusion Models : Sampling



Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

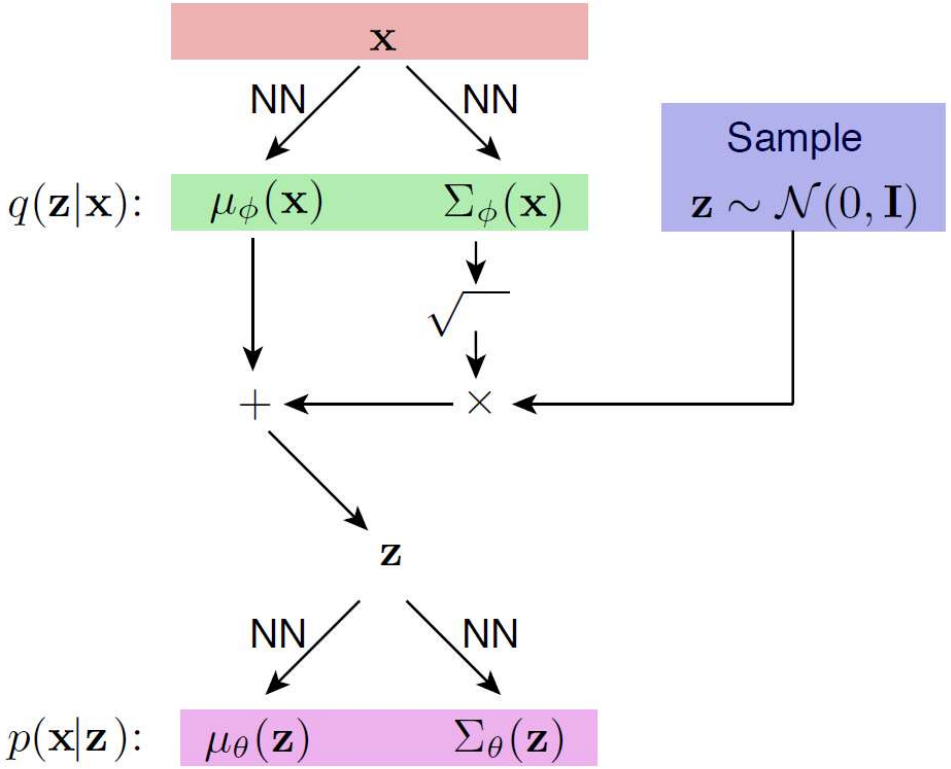


Artificial Intelligence Act

Olga Fink

Essential elements of VAE

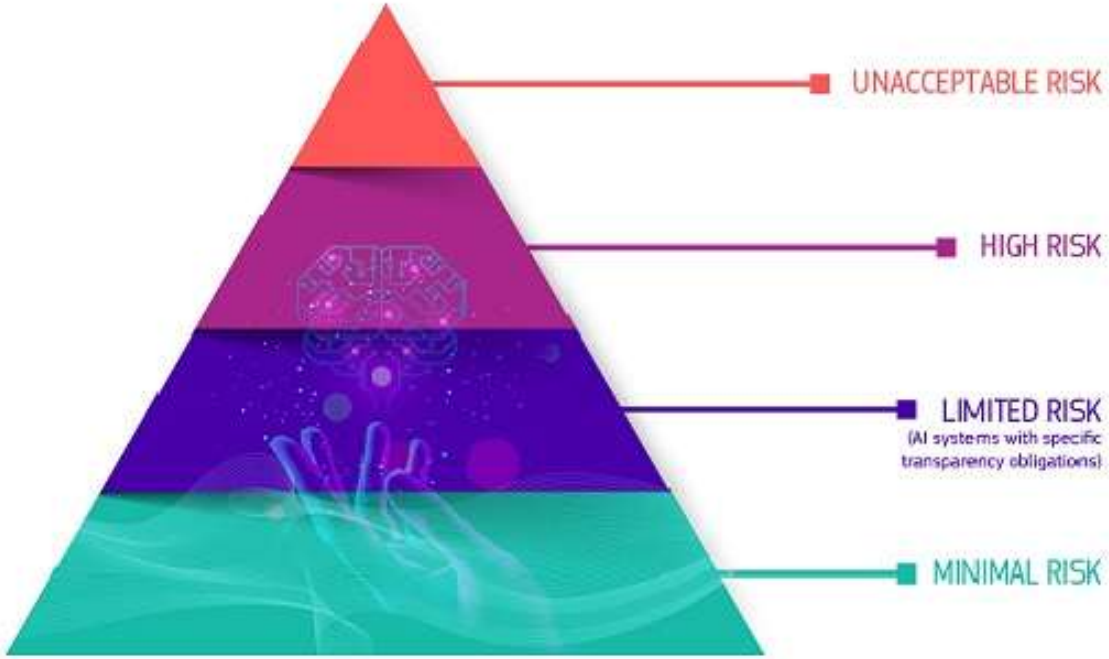
With the reparametrization trick, we can now backpropagate to calculate the gradient with respect to all parameters.



Source: J.C. Kao, UCLA

Artificial Intelligence Act

- Risk-based approach



Unacceptable risk

- All AI systems considered a clear threat to the safety, livelihoods and rights of people
- will be banned,
- Examples: social scoring by governments, toys using voice assistance that encourages dangerous behavior…

High risk (1/2)

- AI systems identified as high-risk include AI technology used in:
  - critical infrastructures (e.g. transport), that could put the life and health of citizens at risk;
  - educational or vocational training, that may determine the access to education and professional course of someone's life (e.g. scoring of exams);
  - safety components of products (e.g. AI application in robot-assisted surgery);
  - employment, management of workers and access to self-employment (e.g. CV-sorting software for recruitment procedures);
  - essential private and public services (e.g. credit scoring denying citizens opportunity to obtain a loan);
  - law enforcement that may interfere with people's fundamental rights (e.g. evaluation of the reliability of evidence);
  - migration, asylum and border control management (e.g. verification of authenticity of travel documents);
  - administration of justice and democratic processes (e.g. applying the law to a concrete set of facts).

High risk (2/2)

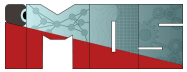
- High-risk AI systems will be subject to strict obligations before they can be put on the market:
  - adequate risk assessment and mitigation systems;
  - high quality of the datasets feeding the system to minimise risks and discriminatory outcomes;
  - logging of activity to ensure traceability of results;
  - detailed documentation providing all information necessary on the system and its purpose for authorities to assess its compliance;
  - clear and adequate information to the user;
  - appropriate human oversight measures to minimise risk;
  - high level of robustness, security and accuracy.

Limited risk

- Limited risk refers to AI systems with **specific transparency obligations**.
- When using AI systems such as chatbots, users should be aware that they are interacting with a machine so they can take an informed decision to continue or step back.

Minimal or no risk

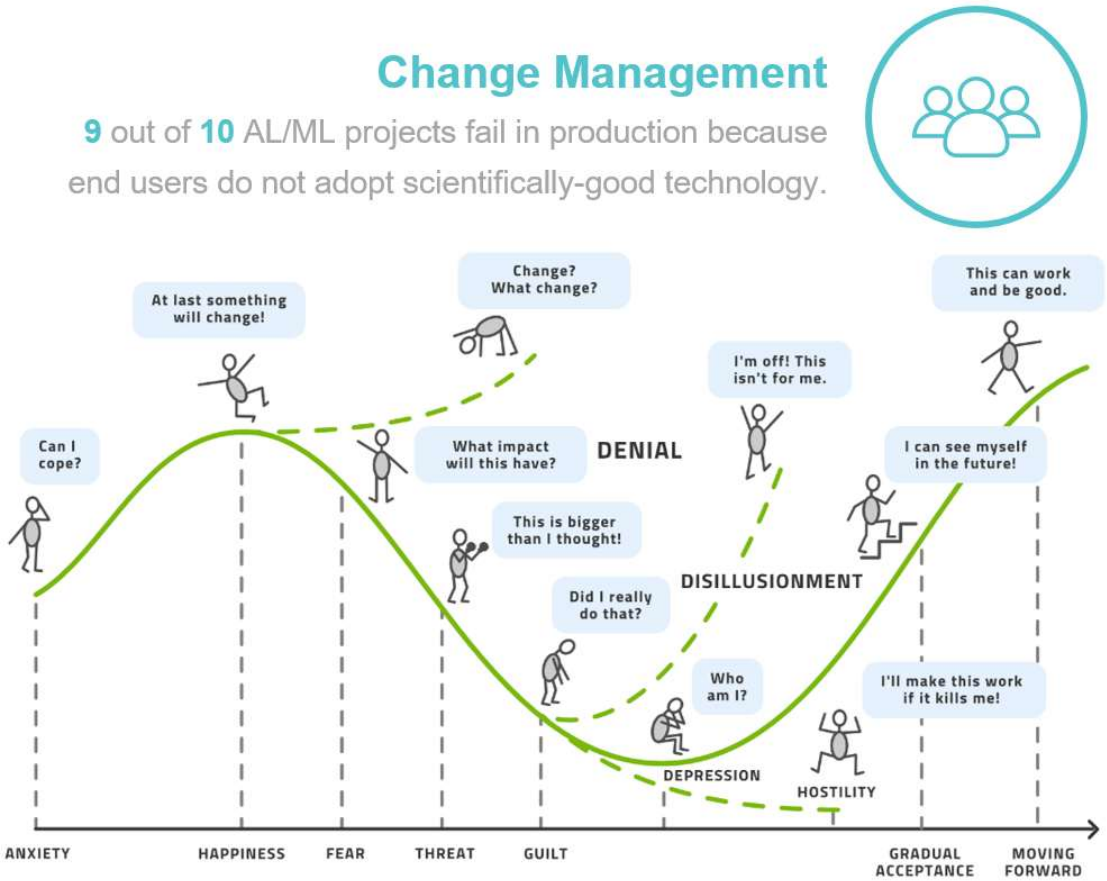
- They neither use personal data nor make any predictions that influence human beings.



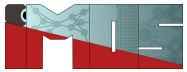
Quote

Olga Fink

Quote (from Samsung)



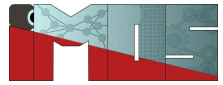
Source: P. Bangert, Samsung



MLOps

Olga Fink

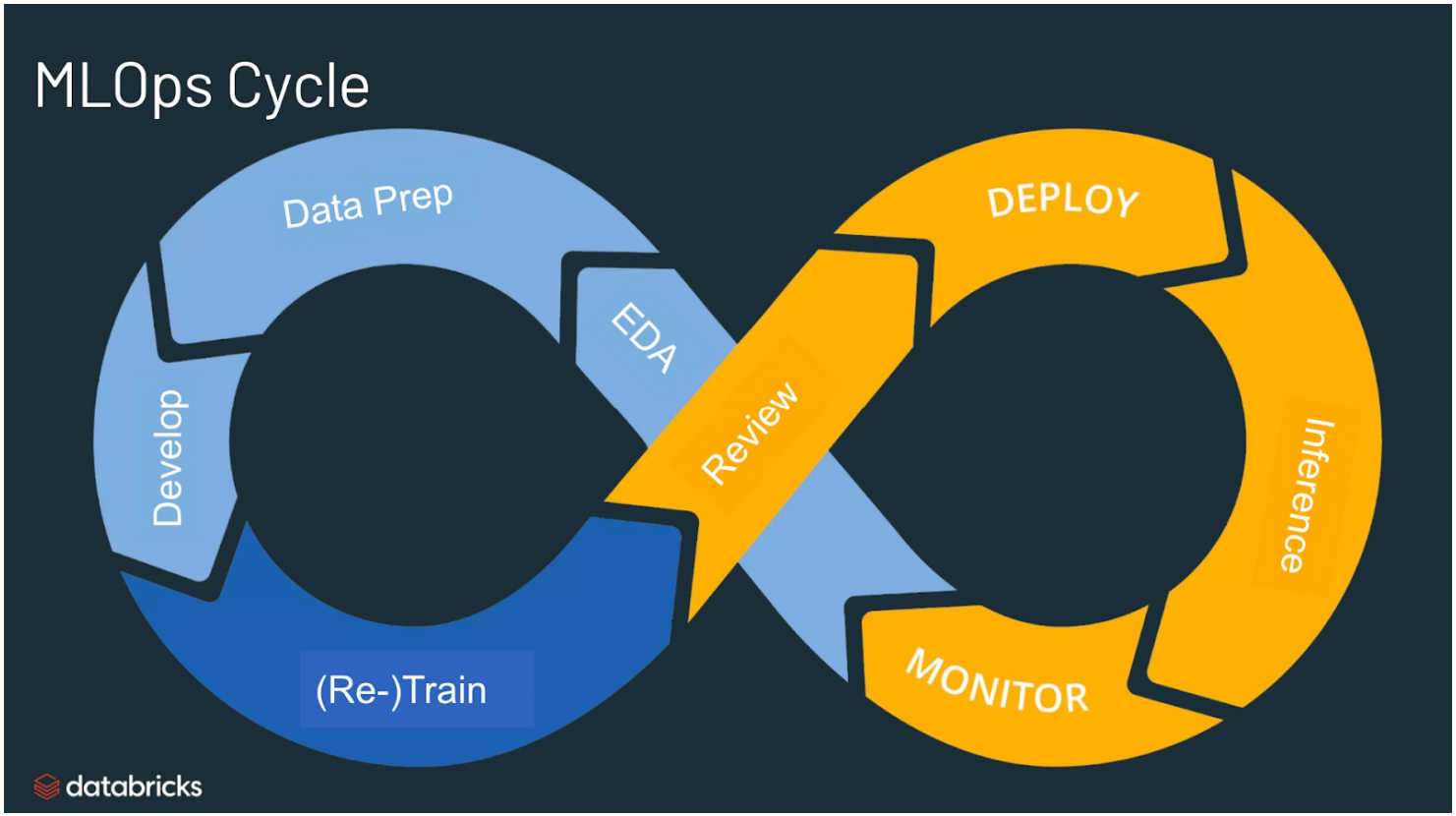
What is MLOps



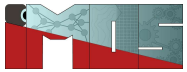
- Machine Learning Operations
- core function of Machine Learning engineering
- focused on streamlining the process of taking machine learning models to production

→then maintaining and monitoring them.

→a collaborative function, often comprising data scientists, development and operations engineers, and IT



Source: databricks



LLMs

Olga Fink

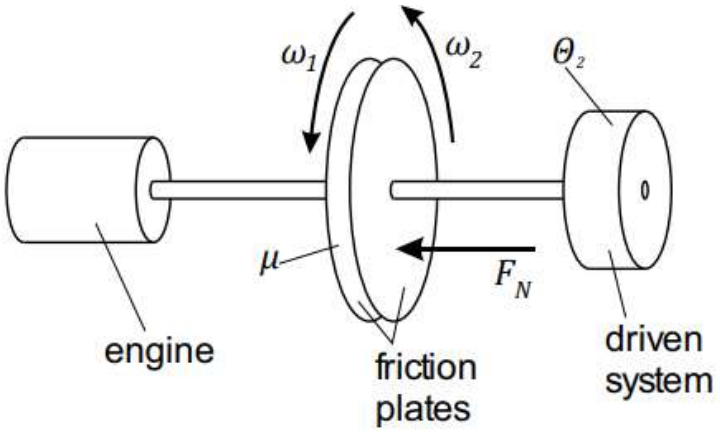
Some perspectives on LLMs

- Natural Language Interfaces
- Data Analysis and Decision Support
- Knowledge Management
- Virtual Assistance and Training
- Quality Assurance and Inspection
- Automation and Optimization
- Collaboration and Communication

Possible means of condition monitoring

- Thermography
- Measurement of compressed air consumption
- Ultrasonic testing technology
- Oil quality monitoring
- Current consumption measurement
- Vibration diagnosis
- Acoustic emission monitoring

Controlling the Remaining Useful Lifetime using Self-Optimization



Two objective functions :

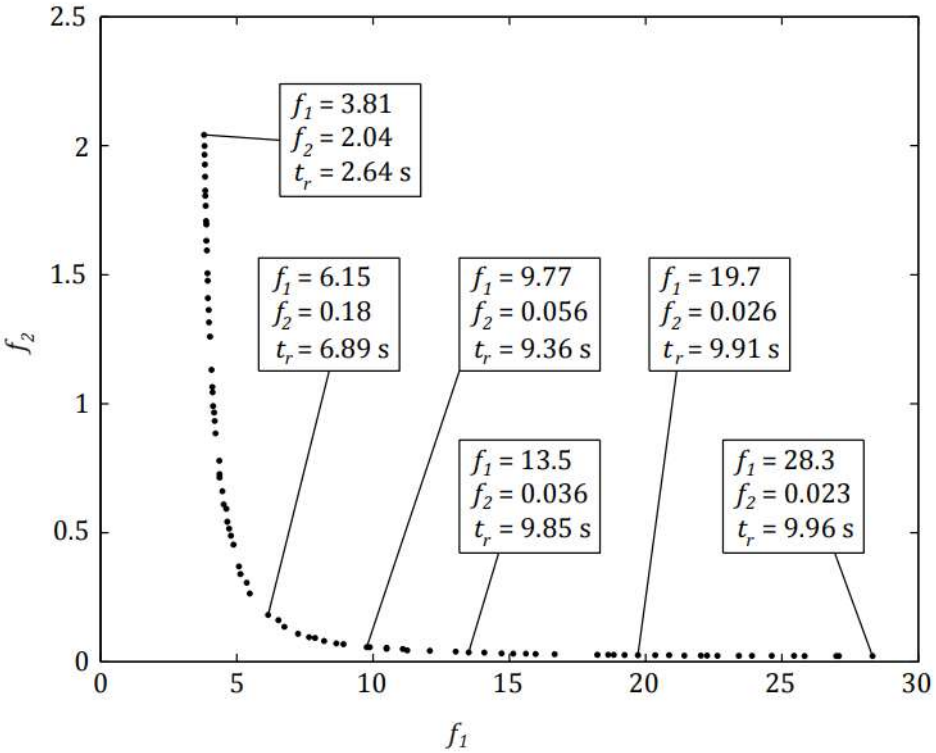
$$f_1 = \int_{t_0}^{t_0+t_r} P_f(t)^2 \mathrm{d}t = \int_{t_0}^{t_0+t_r} \left(T_F(t) \cdot \Delta\omega(t)\right)^2 \mathrm{d}t$$

$$f_2 = \int_{t_0}^{t_0+t_r} \left(\dot{\omega}_2(t)\right)^2 \mathrm{d}t$$

f_1 represents the power loss in the clutch
f_2 represents e.g. comfort of vehicle passengers

Meyer, Tobias, et al. "Controlling the remaining useful lifetime using self-optimization." *Chemical Engineering Transactions* 33 (2013).

Pareto front of the clutch system



Meyer, Tobias, et al. "Controlling the remaining useful lifetime using self-optimization." *Chemical Engineering Transactions* 33 (2013).

Controlling the remaining useful lifetime of a friction drive system

- Based on the assumption that the system deterioration is a consequence of the motion control actions.
- Motion control actions have short-term objectives

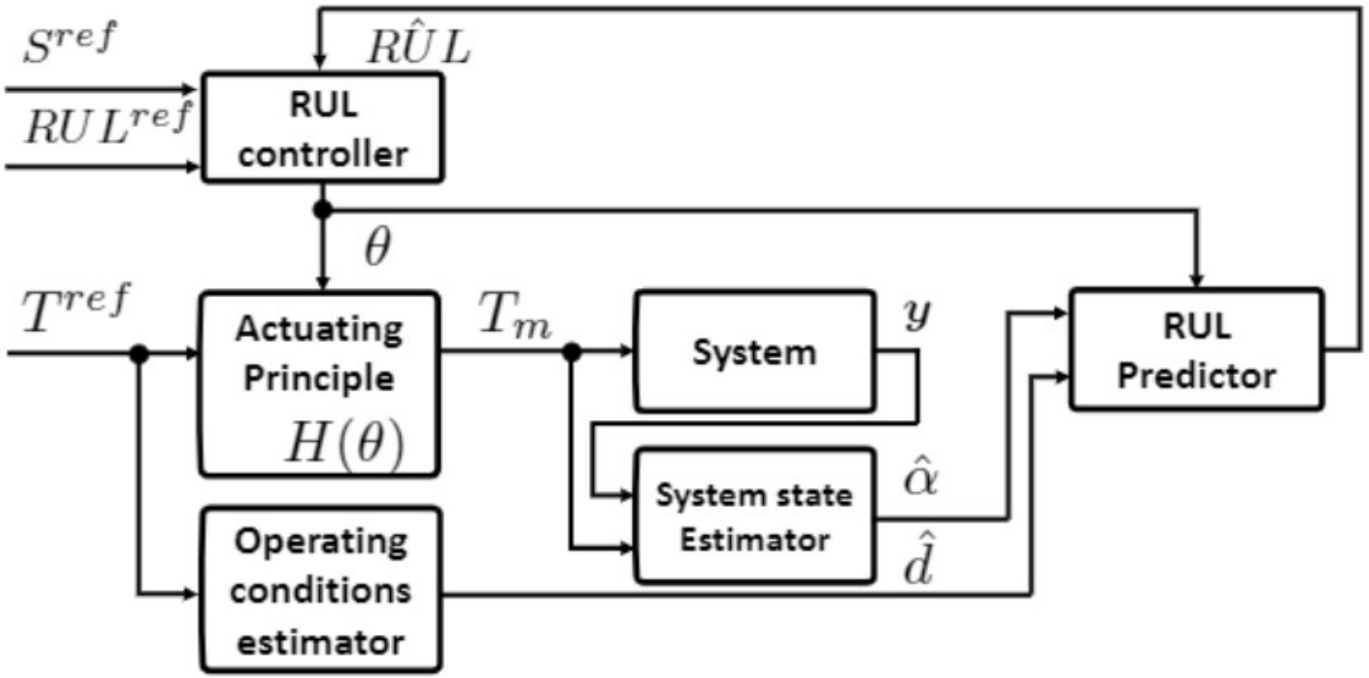
→ have to be modified to be compatible with the required/desired RUL

→ RUL actuating principle is proposed in order to control the RUL

→ The proposed RUL actuating principle is based on a parametric varying filter which modifies the motion control realization based on the available information about the expected RUL

Rodriguez, Diego J., John J. Martinez, and Christophe Berenguer. "An architecture for controlling the remaining useful lifetime of a friction drive system." *IFAC-PapersOnLine* 51.24 (2018): 861-866.

Controlling the remaining useful lifetime of a friction drive system



Rodriguez, Diego J., John J. Martinez, and Christophe Berenguer. "An architecture for controlling the remaining useful lifetime of a friction drive system." *IFAC-PapersOnLine* 51.24 (2018): 861-866.

Different levels of presriptive operation

- Objective: Given UAV state information, the agent performs the optimal routing and then do optimal control to accomplish delivery tasks with lowest energy consumption.

1. **Mission planning : long term**
2. **Motion planning: short term**
3. **Energy management: short term**
4. **System calibration: short term**