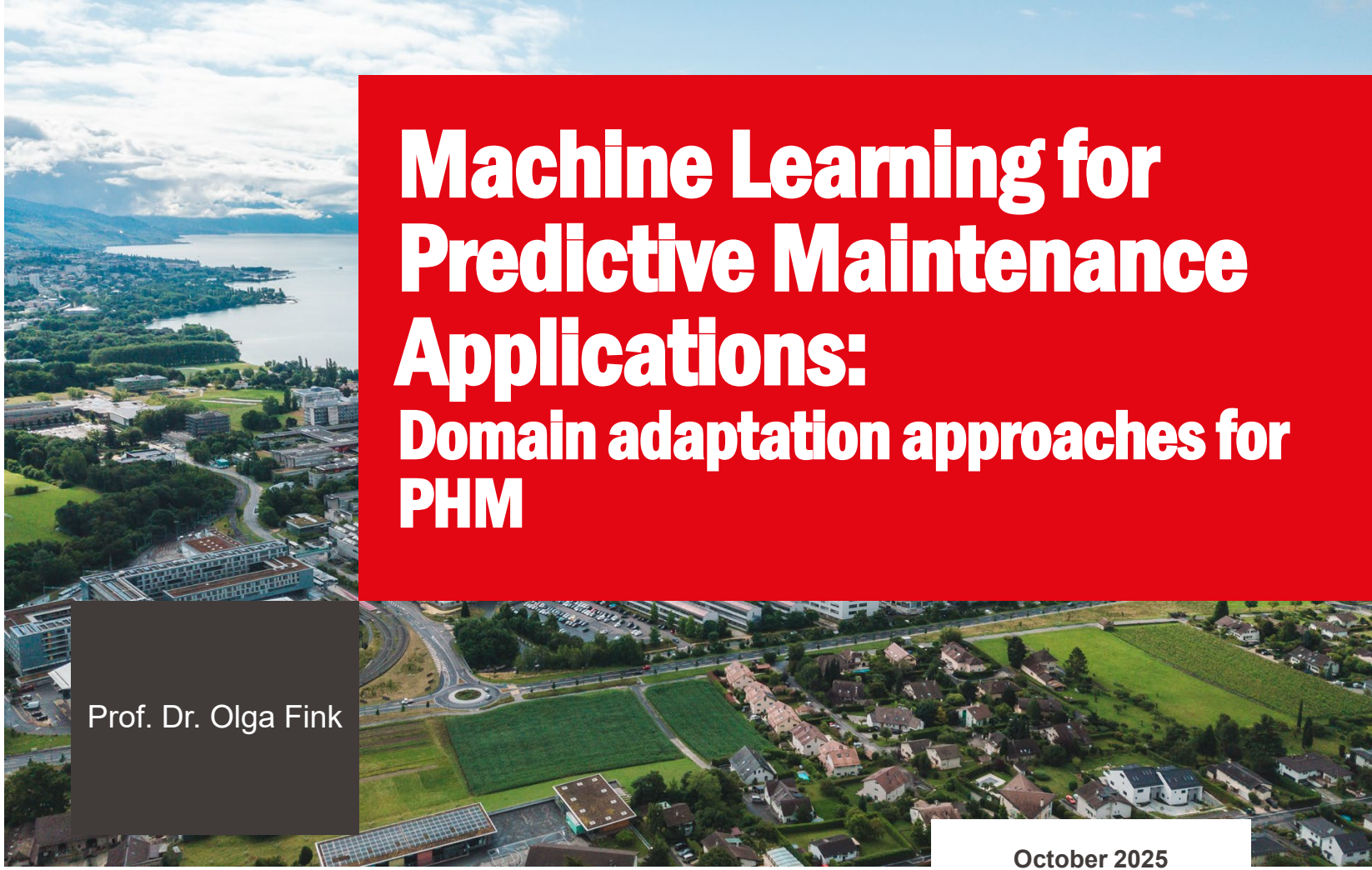


Machine Learning for Predictive Maintenance Applications:
Domain adaptation approaches for PHM

Prof. Dr. Olga Fink

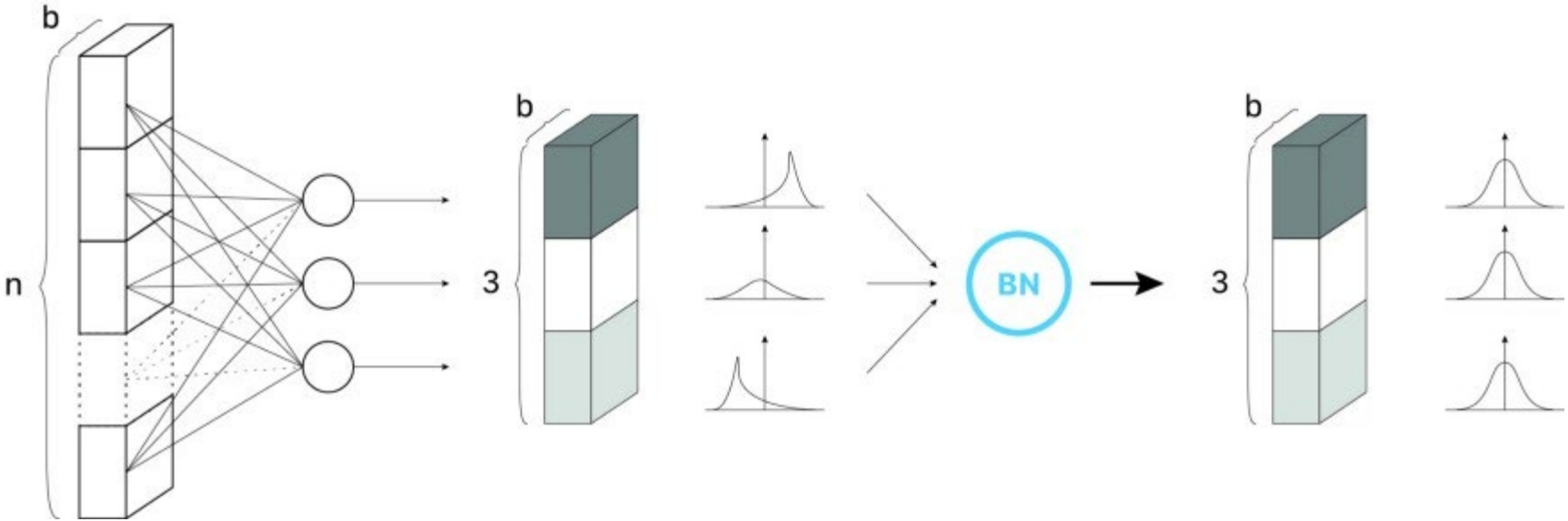
École polytechnique fédérale de Lausanne

October 2025

Recap: Batch Normalization

Basic principle Batch Normalization

- Makes the training faster and more stable



Source: Huber, 2020

Basic principle Normalization

$$\mu = \frac{1}{n}\sum_i x^{(i)}$$

$$\sigma^2 = \frac{1}{n}\sum_i (x^{(i)} - \mu)^2$$

$$x^{(i)}_{\text{norm}} = \frac{x^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\tilde{x}^{(i)} = \gamma \cdot x^{(i)}_{\text{norm}} + \beta$$

$$\tilde{x}^{(i)} = \gamma \cdot \frac{x^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

$x^{(i)}$ input activation (scalar or vector component)

n number of elements included in normalization

μ mean of activations within the normalization group

σ^2 variance of activations within the normalization group

ϵ small constant added for numerical stability

$x^{(i)}_{\text{norm}}$ standardized activation (zero mean, unit variance)

γ learnable scale parameter

β learnable shift parameter

$\tilde{x}^{(i)}$ final normalized and re-scaled output

- Allows the model to choose the optimum distribution for each hidden layer, by adjusting those two parameters :
- γ allows to adjust the standard deviation
- β allows to adjust the bias, shifting the curve on the right or on the left side

Different normalizations

Normalization Type	Normalized Across	Typical Use	Depends on Batch Size?
Batch Normalization (BatchNorm)	Across the batch dimension (all samples in a mini-batch)	CNNs	Yes
Layer Normalization (LayerNorm)	Across features of a single sample	Transformers, GNNs	No
Instance Normalization (InstanceNorm)	Across each feature map per sample	Style transfer, image generation	No
Group Normalization (GroupNorm)	Across groups of channels per sample	CNNs (small batch)	No

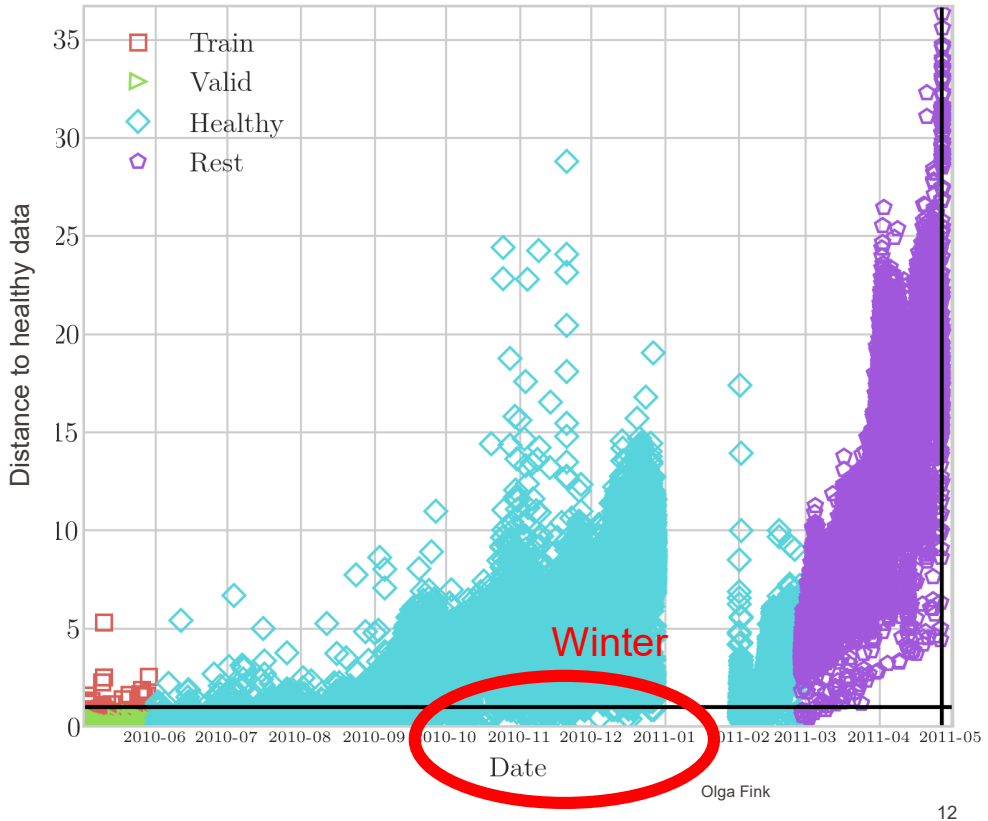
Why domain adaptation / transfer learning?

Some Challenges in Predictive Maintenance

- Varying and evolving operating conditions → Even healthy system conditions are not always representative due to limited observation time period
- Algorithms also for systems required that are newly taken into operation

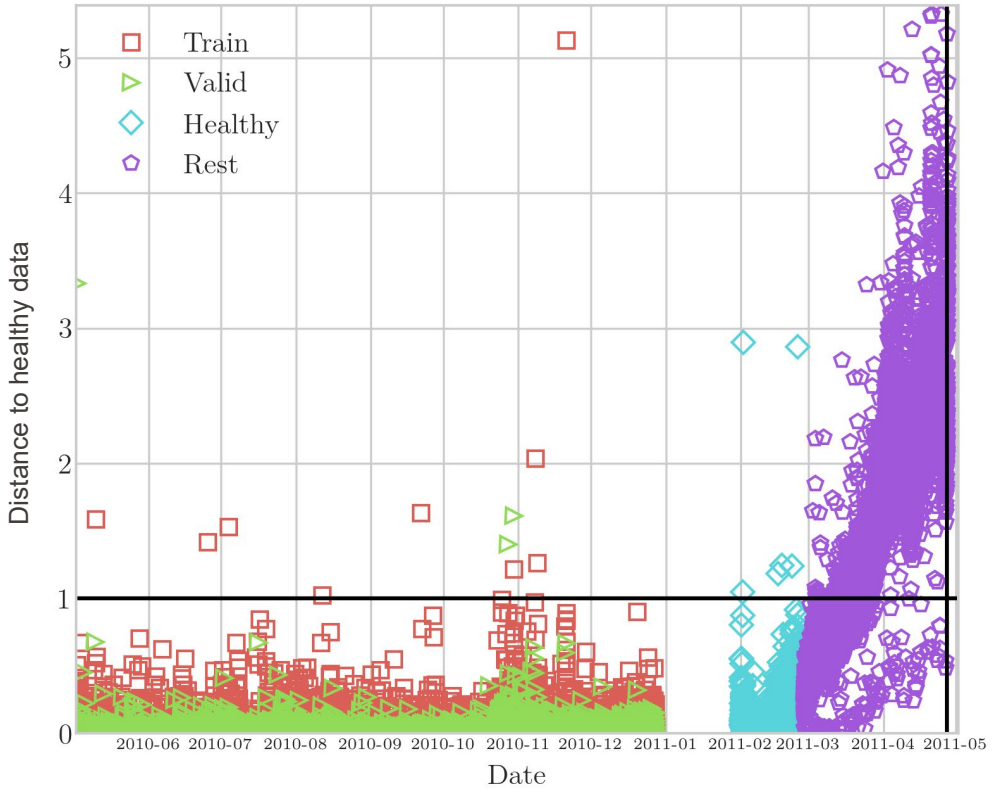
Example Gas Turbines

Only a short observation period

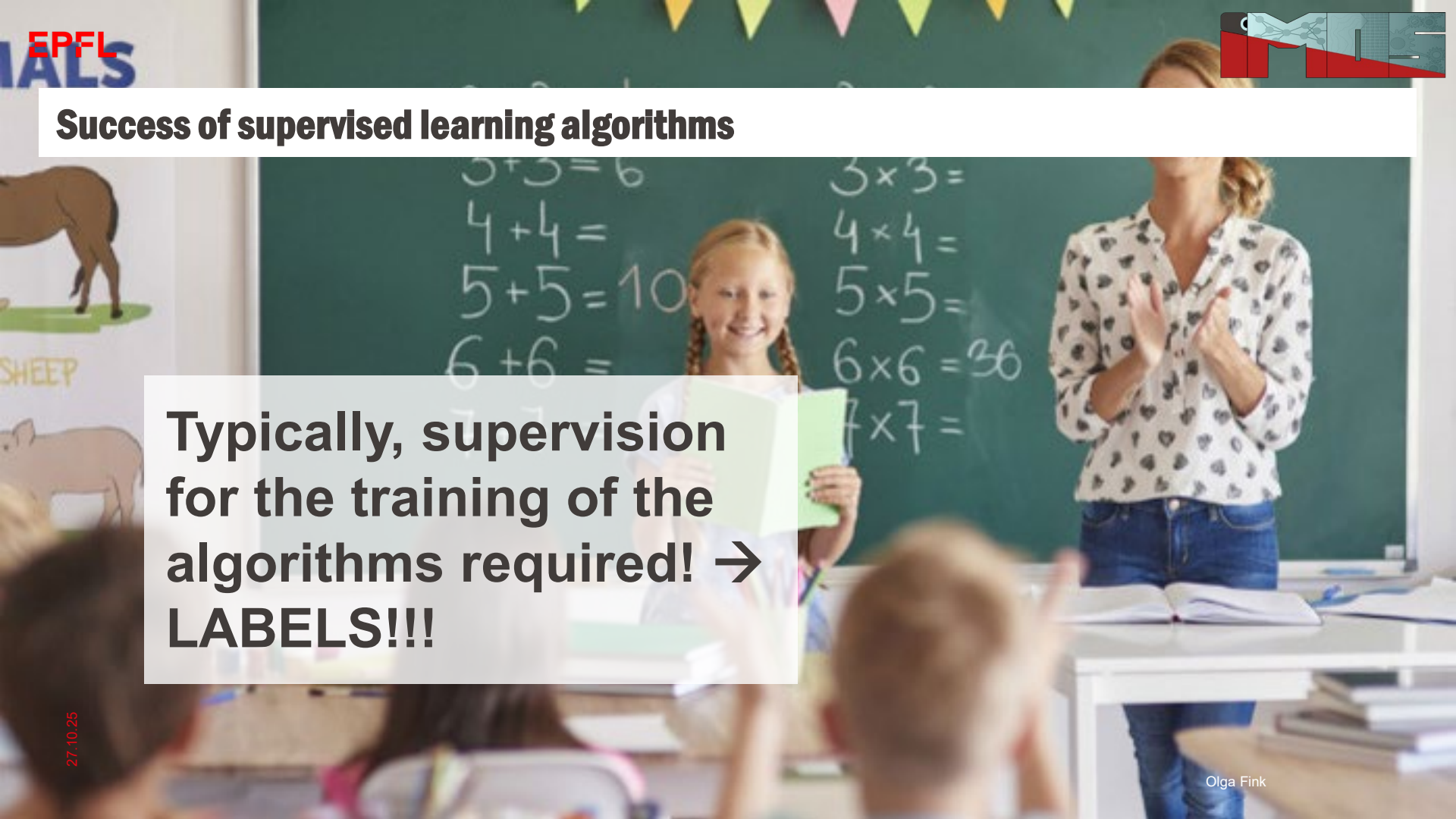


Example Gas Turbines

Extend the observation period



Success of supervised learning algorithms



Typically, supervision for the training of the algorithms required! → LABELS!!!

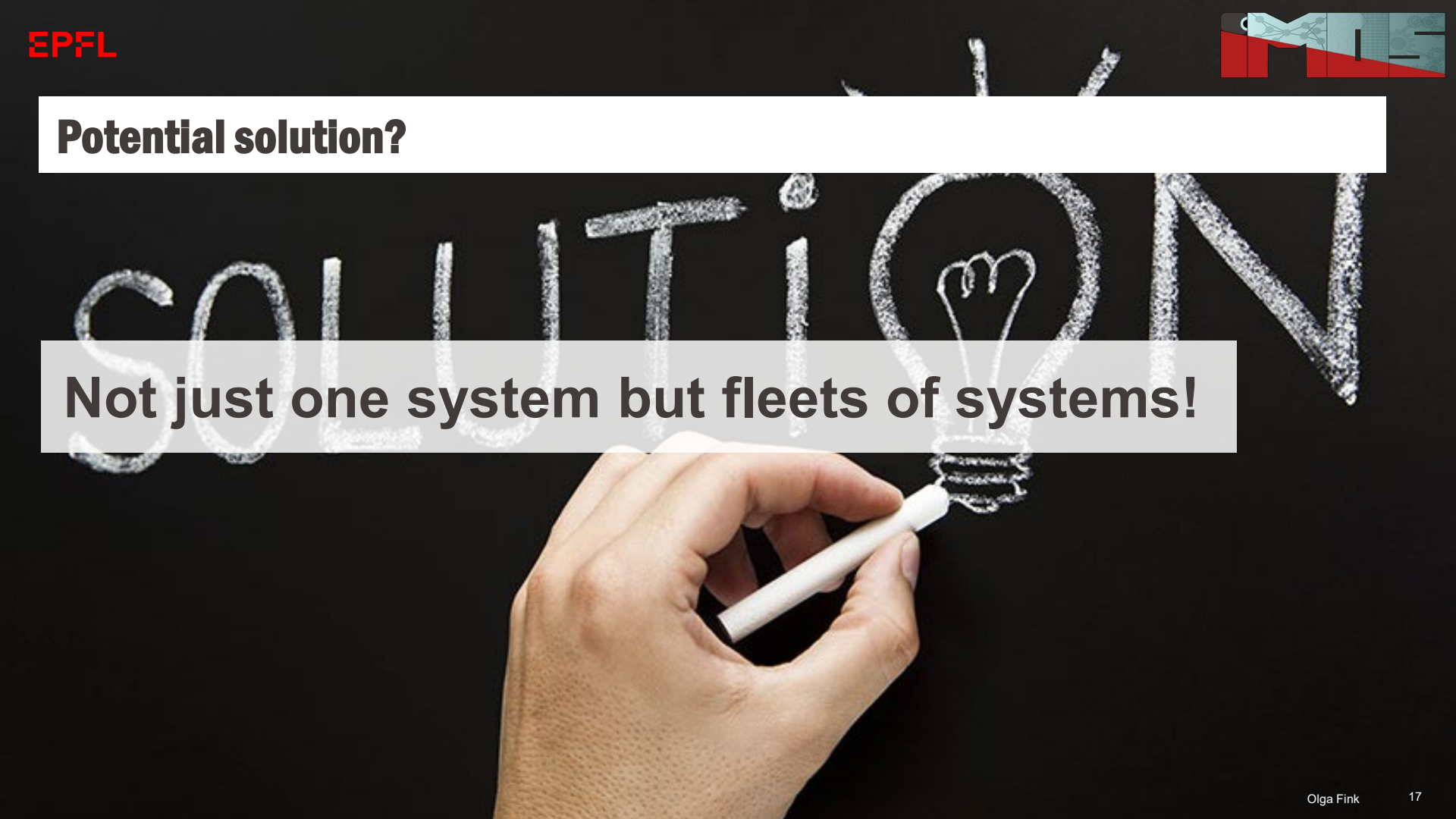
Faults are equivalent to labels in intelligent maintenance

Faults in critical systems are rare → not possible to learn a good representation of faulty conditions

How about a new system?

How long do we have to wait until we get reliable fault detection (diagnostics)?

Potential solution?



Not just one system but fleets of systems!


Fleets of complex systems?


Fleets of complex systems?

What do we start with?

- Limited number of faults (labels)
- Large variety of condition monitoring data under different operating conditions
- Several units of the same fleet (but units have variability in their configurations and operating conditions)
- Heterogenous operating conditions and configurations of the fleet units
- Limited observation time periods
- **Limited representativeness** of the collected data for the expected operating conditions

What are we trying to achieve?

- Compile representative training datasets that are valid for the specific units under the specific operating conditions (homogeneous datasets)
- Using labeled and unlabeled data as efficiently as possible at the level of an entire fleet
- Develop also algorithms for new units
- Transferring knowledge (on operating conditions and faults) between the single units of a fleet
- Learn robust features that are invariant to different operating conditions

Transfer learning / unsupervised Domain Adaptation

Different types of domain changes (images)

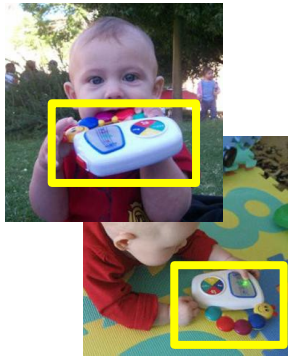
high quality

low quality

daylight

sunset

posed



“in the wild”

art

surveillance

Source: Saenko, 2012

Transfer learning in DL

- **Myth**: you can't do deep learning unless you have a million labelled examples for your problem.
- **Reality**
  - You can learn useful representations from **unlabelled data**
  - You can train on a nearby **surrogate objective** for which it is easy to generate labels → self-supervised learning
  - You can **transfer** learned representations from a related task

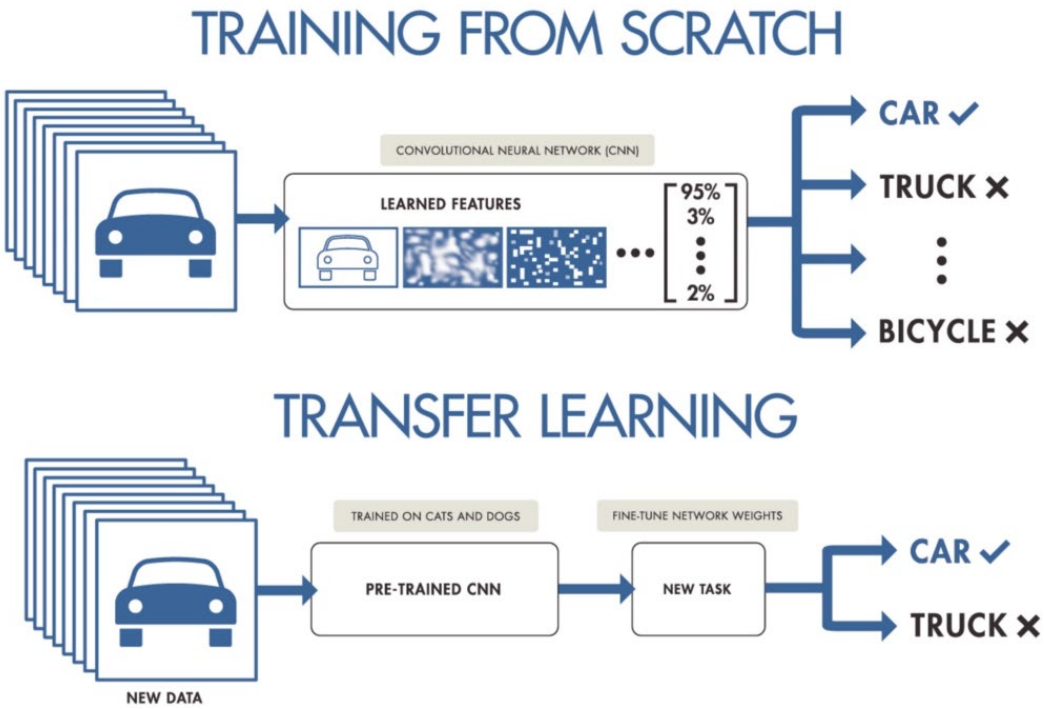
Source: Morros, 2017

Transfer Learning

- **The ability to apply knowledge learned in previous tasks to novel tasks**
  - Similar on human learning.
  - People can often transfer knowledge learnt previously to novel situations
    - Play classic piano →Play jazz piano
    - Maths →Machine Learning
    - Ride motorbike →Drive a car

Source: Morros, 2017

Making use of trained models



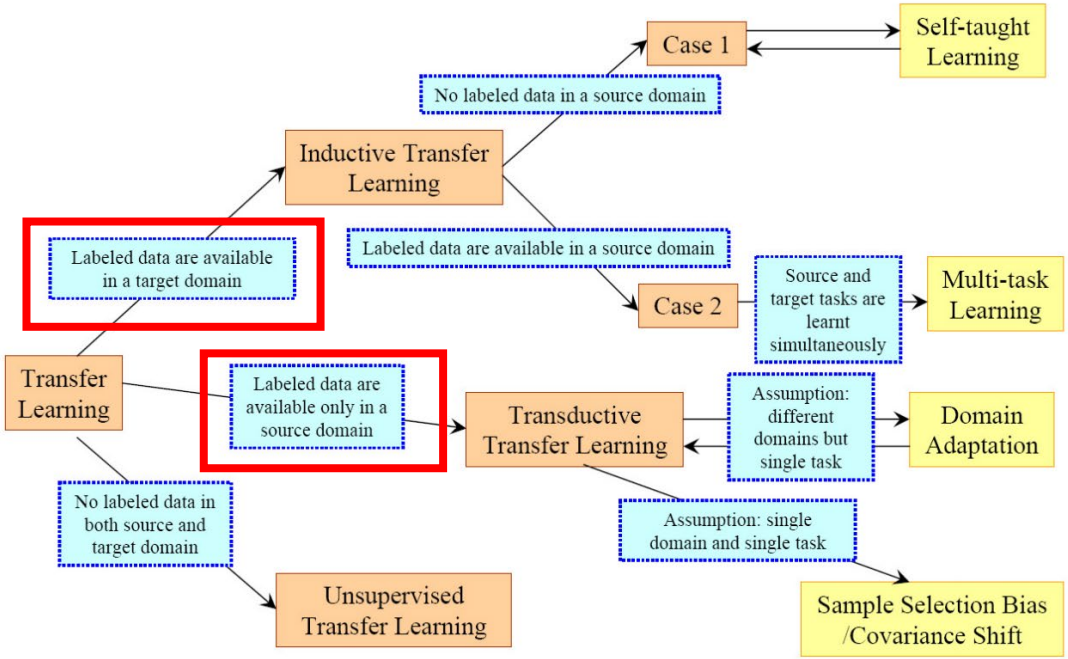
Source: H. Davidson Regua, 2020

Relationship between traditional ML and various transfer learning settings

Learning Settings		Source and Target Domains	Source and Target Tasks
Traditional Machine Learning		the same	the same
Transfer Learning	*Inductive Transfer Learning / Unsupervised Transfer Learning*	the same	different but related
		different but related	different but related
	Transductive Transfer Learning	different but related	the same

Source: S. J. Pan & Q. Yang, 2009

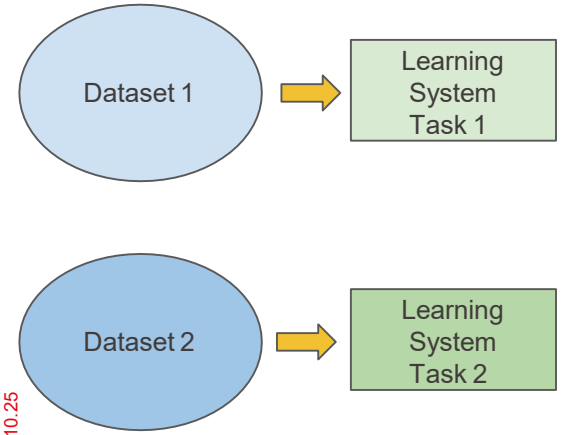
Different types of transfer learning



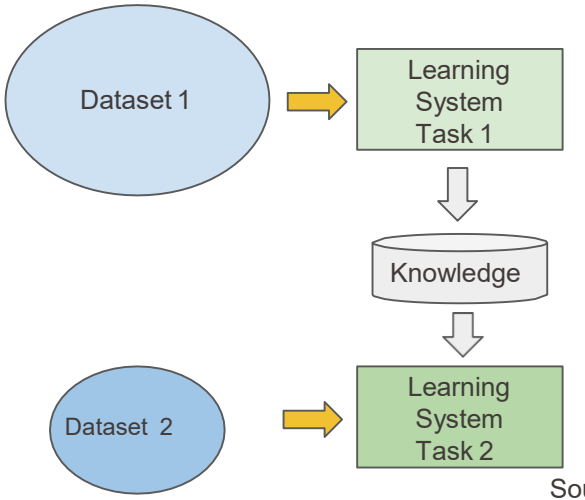
Source: S. J. Pan & Q. Yang, 2009

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed
  - w.o. considering past learned knowledge in other tasks



- Learning of a new task relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



Source: Morros, 2017

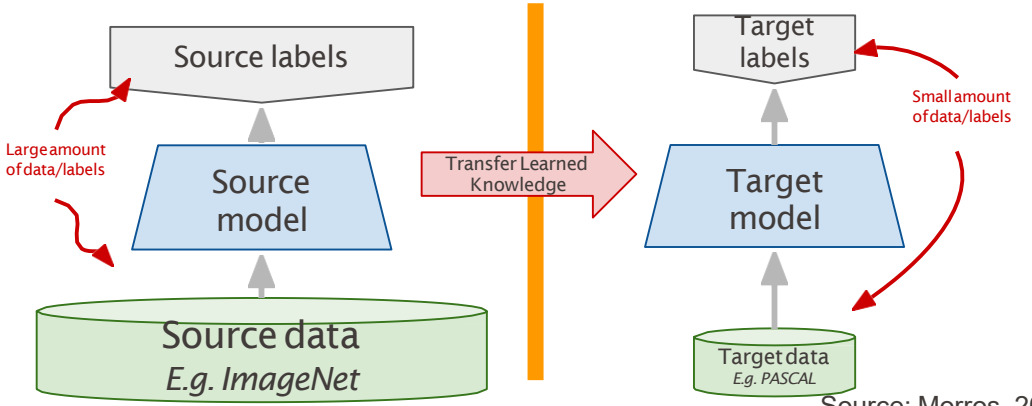
Transfer learning: idea

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

Variations:

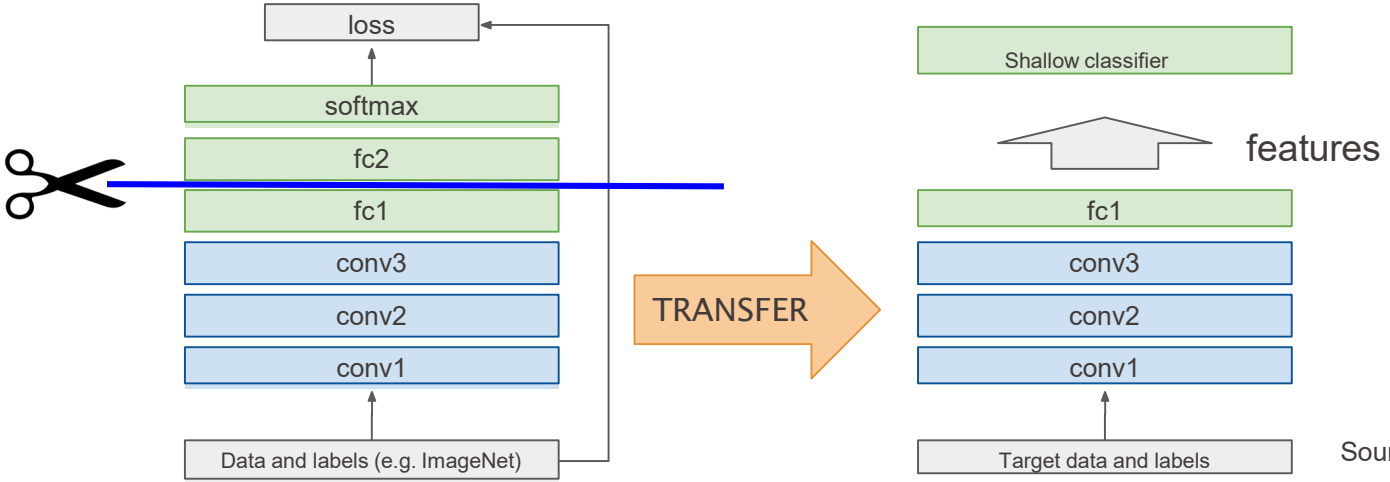
- Same domain, different task
- Different domain, same task



Source: Morros, 2017

“Off-the-shelf”

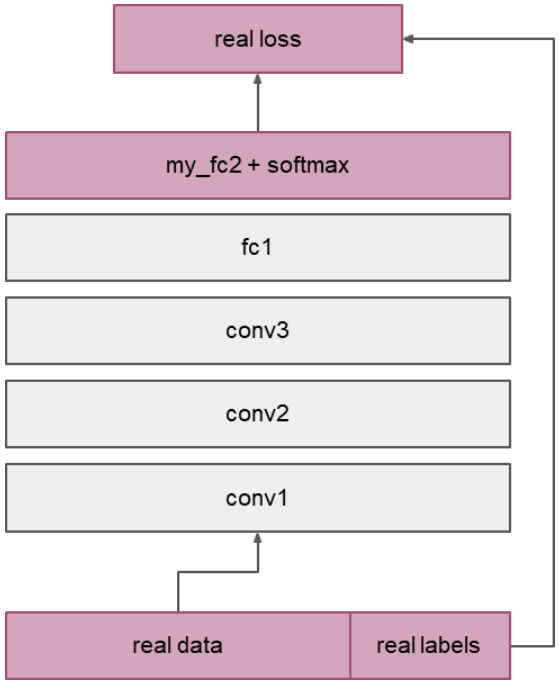
Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.



Source: Morros, 2017

Fine-tuning: transfer learning

- Train deep net on “nearby” task for which it is easy to get labels using standard backprop
  - E.g. ImageNet classification
  - Pseudo classes from augmented data
- Cut off top layer(s) of network and replace with supervised objective for target domain
- **Fine-tune** network using backprop with labels for target domain until validation loss starts to increase



Source: Morros, 2017

Freeze or fine-tune?

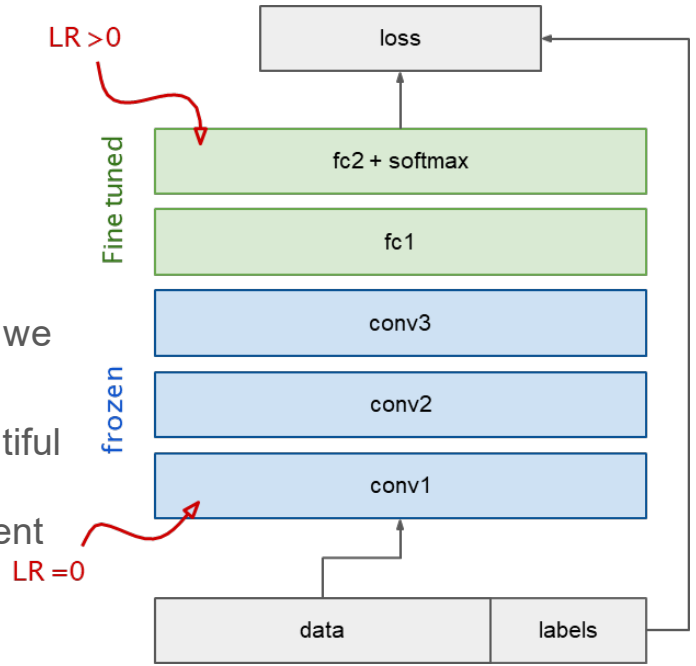
Bottom *n* layers can be frozen or fine tuned.

- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



Source: Morros, 2017

Unsupervised domain adaptation

- Labeled training data from a source domain

$$\mathcal{D}_s = \{(\mathbf{x}_s^1, \mathbf{y}_s^1), ..., (\mathbf{x}_s^m, \mathbf{y}_s^m)\}, \mathbf{y}_s^i \in Y.$$

- Unlabeled data from a target domain

$$\mathcal{D}_t = \{\mathbf{x}_t^1, ..., \mathbf{x}_t^n\}.$$

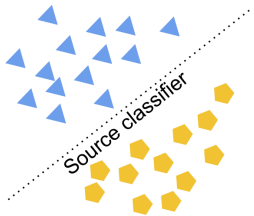
- Test data from the same target domain

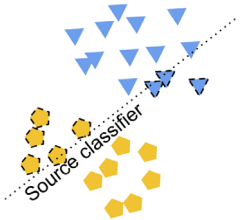
$$\mathcal{D}_{test} = \{\mathbf{x}_{test}^1, ..., \mathbf{x}_{test}^o\},$$

Domain Shift - Target domain different from Source domain

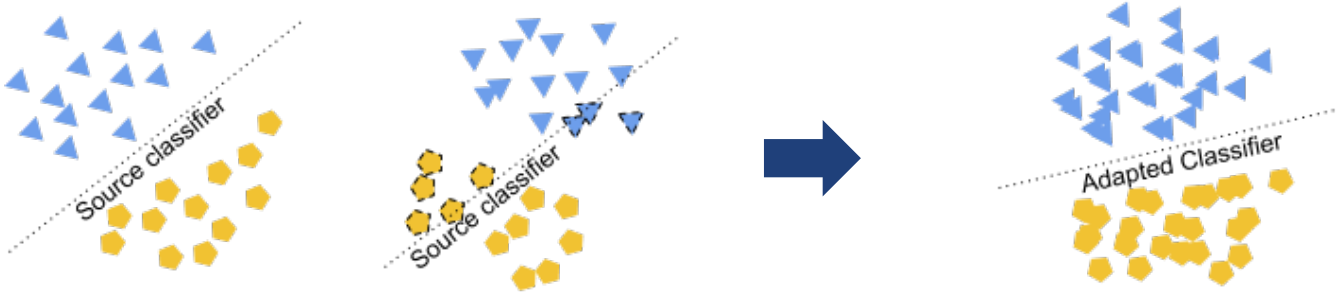
Source

Target

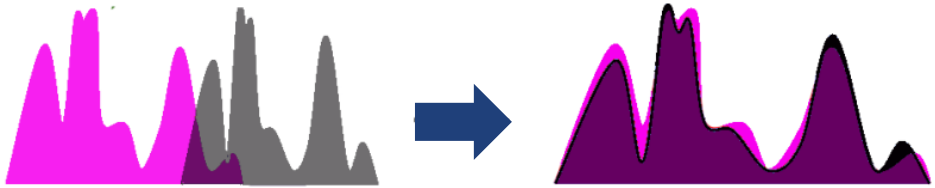




Unsupervised Domain Adaptation

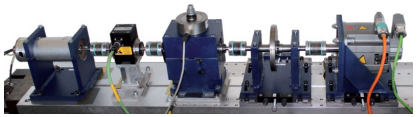


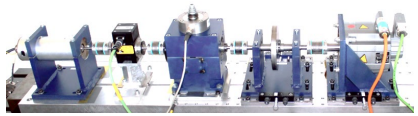
Labeled Source Domain **Unlabeled Target Domain** **Adapted Domain**



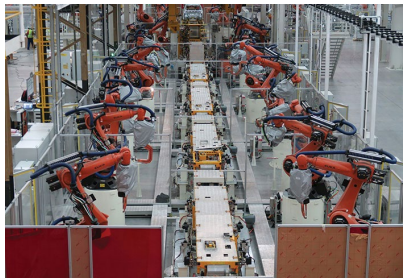
Transferability / Generalizability

- Between different operating conditions

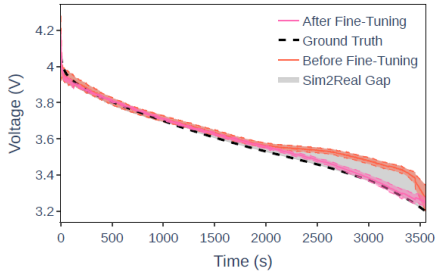




- Between different units of a fleet

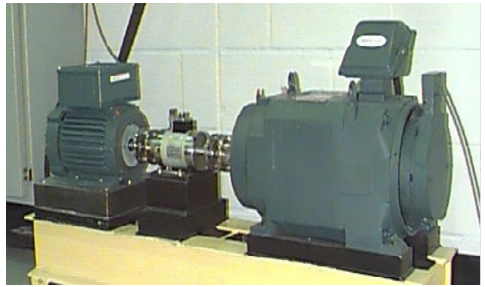


- From simulated data to real conditions



Wang, Qin, Gabriel Michau, and **Olga Fink** (2021): "Missing-Class-Robust Domain Adaptation by Unilateral Alignment", *IEEE Transactions on Industrial Electronics, 68 (1), 663-671.*

Case Study on CWRU (Case Western Reserve University) Bearing Dataset



- Dataset information
  - Ten Classes classification

Fault	Class Label									
	0	1	2	3	4	5	6	7	8	9
Loc	NA[1]	IF	IF	IF	BF	BF	BF	OF	OF	OF
Size	0	7	14	21	7	14	21	7	14	21

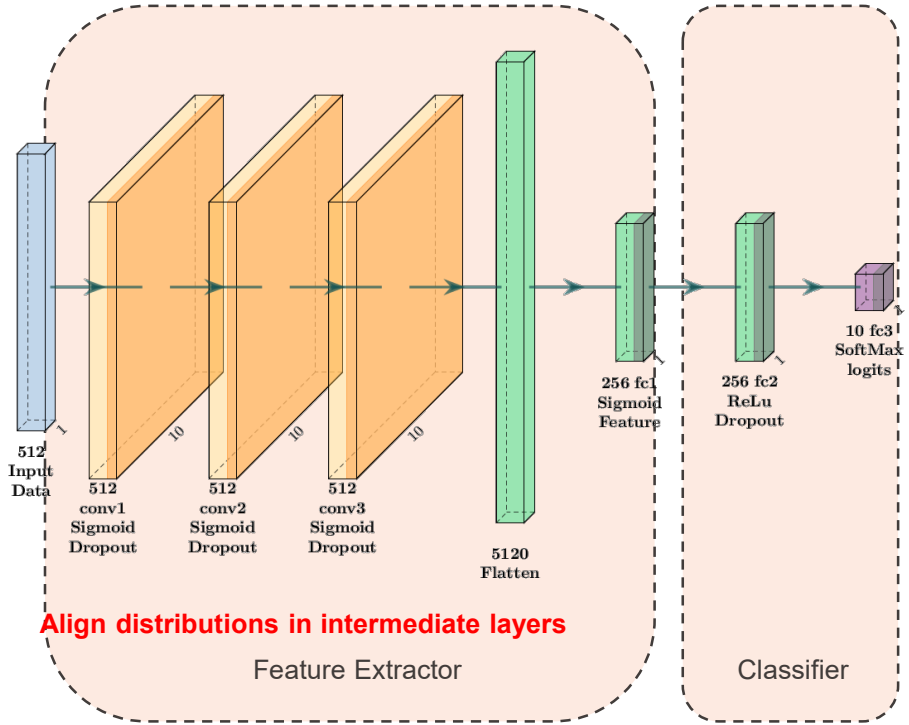
  - Four loads for domain adaptation

    Labeled source load, Unlabeled target load

Motor Load (HP)	Approx. Motor Speed (rpm)
0	1797
1	1772
2	1750
3	1730

- Data Preprocessing
  - 200 sequences of 1024 points for each recording
  - each converted to 512D Fourier coefficients by FFT

Basic Network

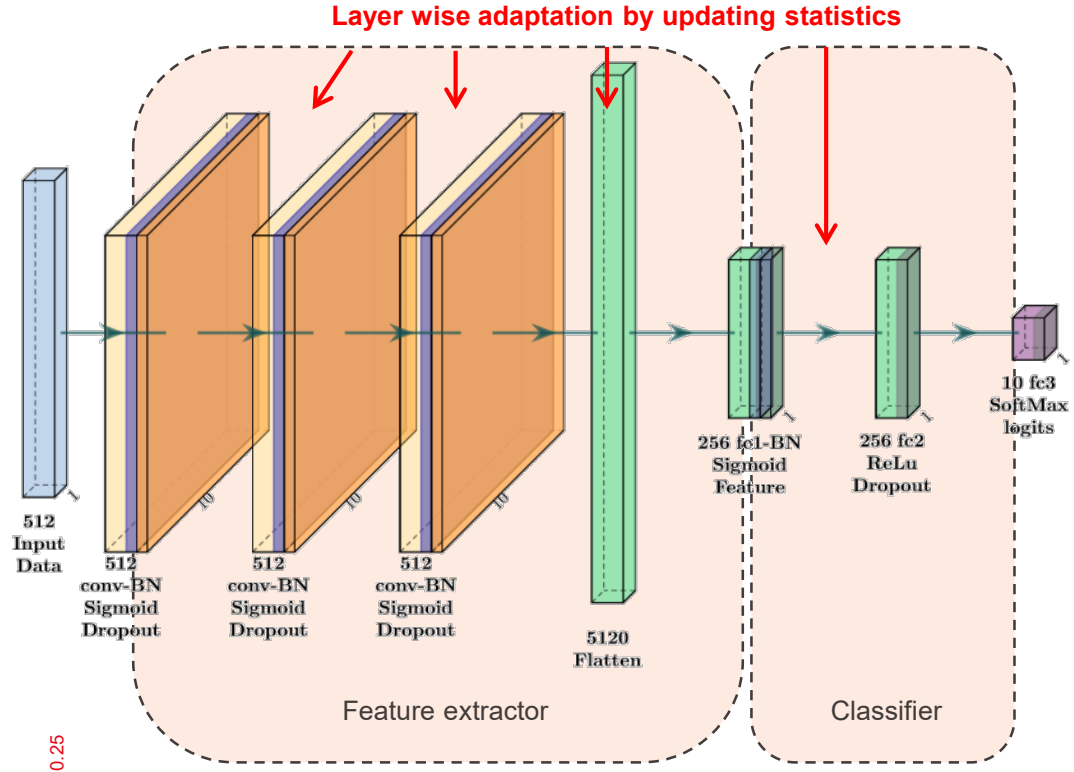


- Feature Extractor
  - Three convolutional layers
  - Flatten and dense layer
- Classifier
  - Fully-connected Layer

- How should we add domain adaptation ability to this network?

[14] X. Li, W. Zhang, and Q. Ding, "Cross-domain fault diagnosis of rolling element bearings using deep generative neural networks," IEEE Transactions on Industrial Electronics, 2018.

Background: Adaptive Batch Normalization (AdaBN)

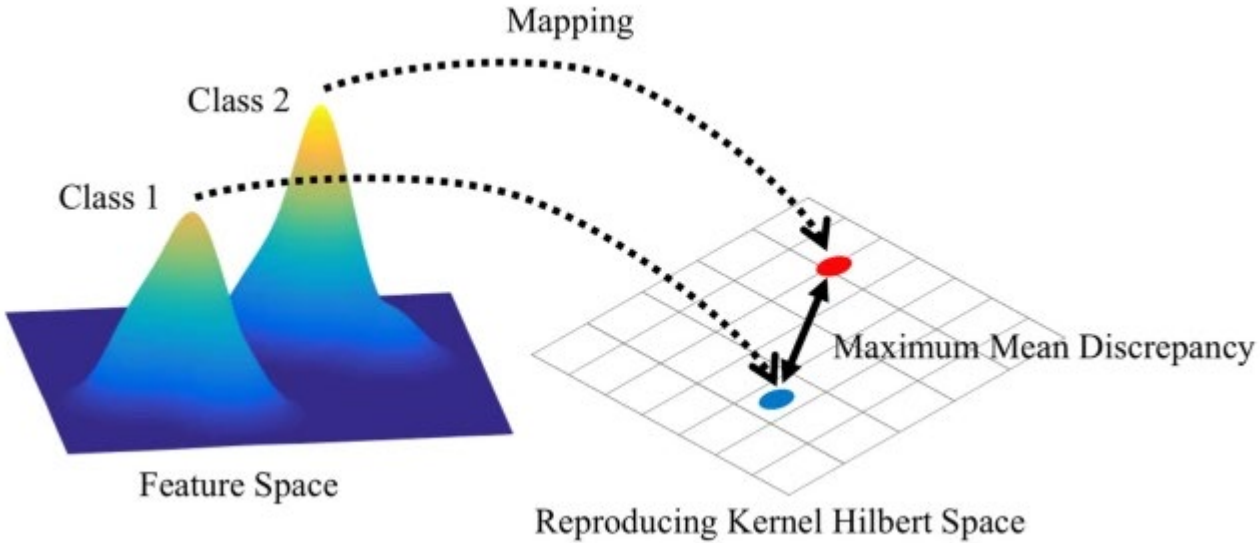


$$\hat{x_j} = \frac{x_j - \mathbb{E}[x_{.j}]}{\sqrt{Var[x_{.j}]}}$$

$$y_j = \gamma_j \hat{x_j} + \beta_j,$$

- Idea
  Keep different batch norm statistics for source and target.
- Pros
  Simplicity. Very little computational resource required.
- Cons
  Performance not optimized

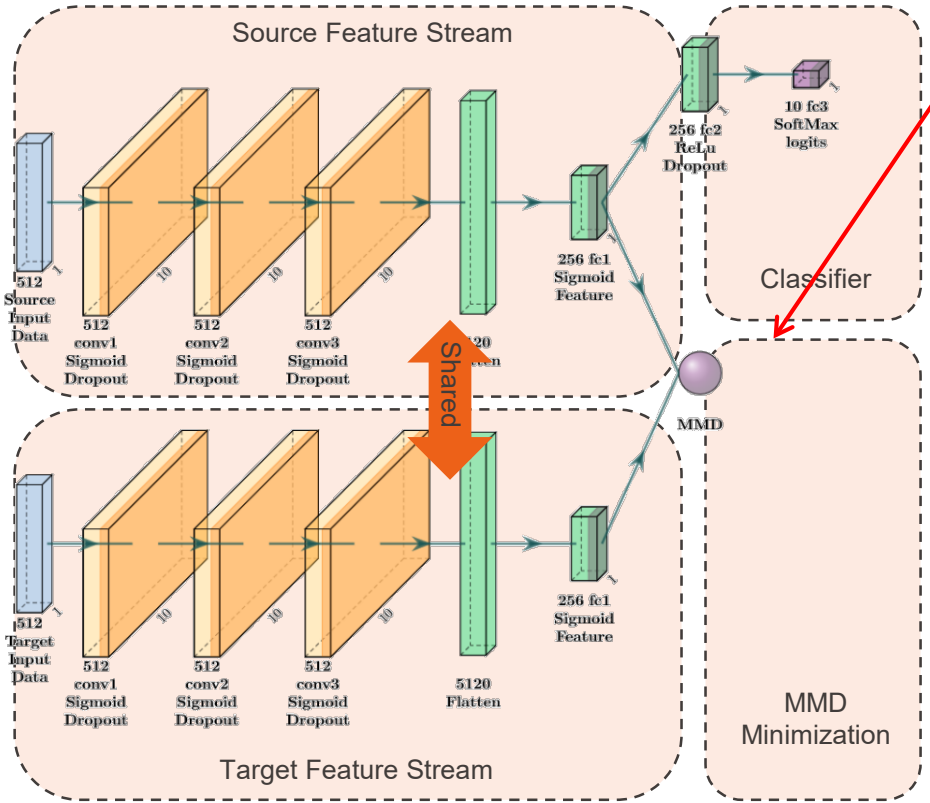
Maximum Mean Discrepancy



$$\mathcal{L}_{\mathrm{D}}(\mathbf{X}^s, \mathbf{X}^t) = \mathrm{MMD}(\mathbf{X}^s, \mathbf{X}^t) = \left\| \frac{1}{n^s} \sum_{i=1}^{n^s} \phi(\mathbf{x}_i^s) - \frac{1}{n^t} \sum_{j=1}^{n^t} \phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2$$

ϕ: mapping function
$\boldsymbol{X}^s$: feature matrix in source domain
$\boldsymbol{X}^t$: feature matrix in target domain

Background: Maximum Mean Discrepancy Minization

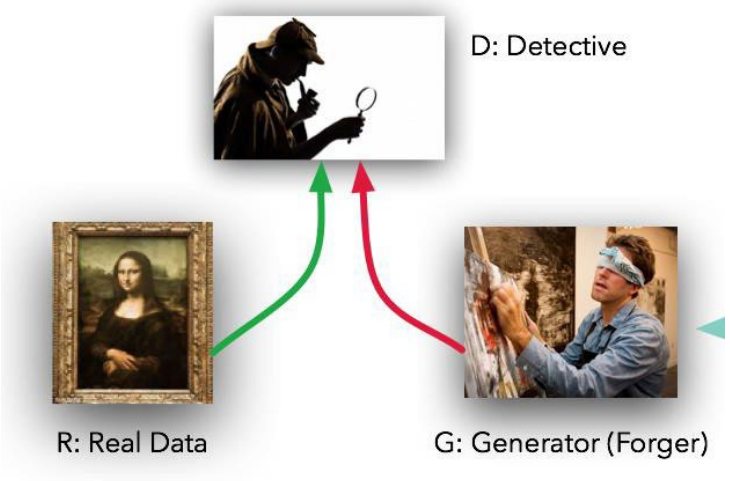


Estimate distribution difference between source and target features by MMD.

- Idea
  - MMD as an additional loss to minimize.
- Pros
  - MMD **directly estimates the distance** between source and target distributions.
- Cons
  - Multiple kernels are needed in reality, leads to dramatically **increased model complexity**

X. Li, W. Zhang, Q. Ding, and J.-Q. Sun, "Multi-layer domain adaptation method for rolling bearing fault diagnosis," Signal Processing, vol. 157, pp. 180–197, 2019.

Adversarial Training



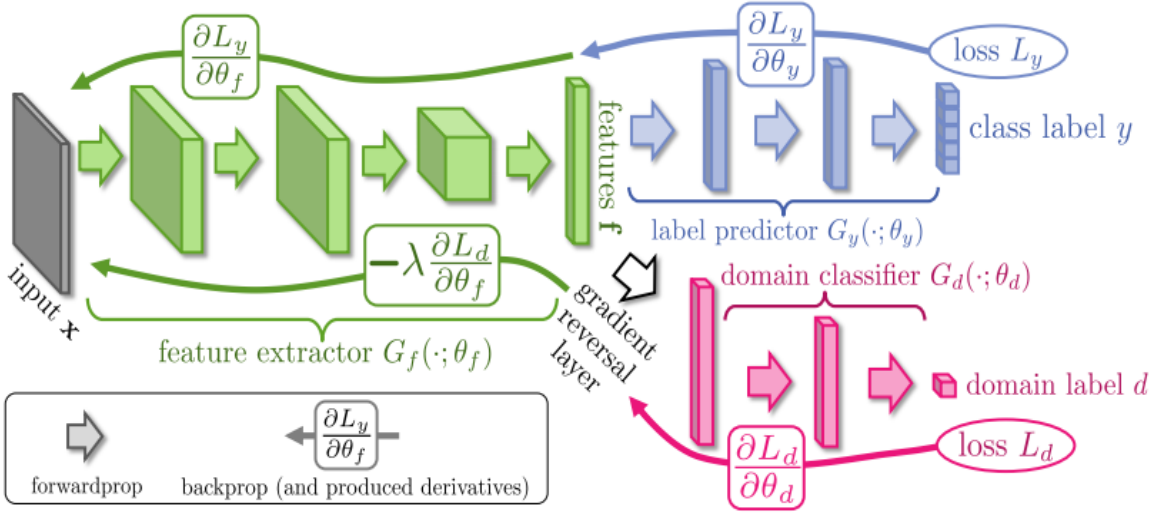
Generative Adversarial Networks (GAN)

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

Adversarial Networks in DA

Domain adaptation for deep learning , Kate Saenko

Domain-Adversarial Training (DANN) in detail



$$E(\theta_f,\theta_y,\theta_d) = \sum_{\substack{i=1..N\\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) -$$

$$\lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), y_i\right) =$$

$$= \sum_{\substack{i=1..N\\ d_i=0}} L_y^i(\theta_f,\theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f,\theta_d)$$

$$(\hat{\theta}_f,\hat{\theta}_y) = \arg\min_{\theta_f,\theta_y} E(\theta_f,\theta_y,\hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f,\hat{\theta}_y,\theta_d)\,.$$

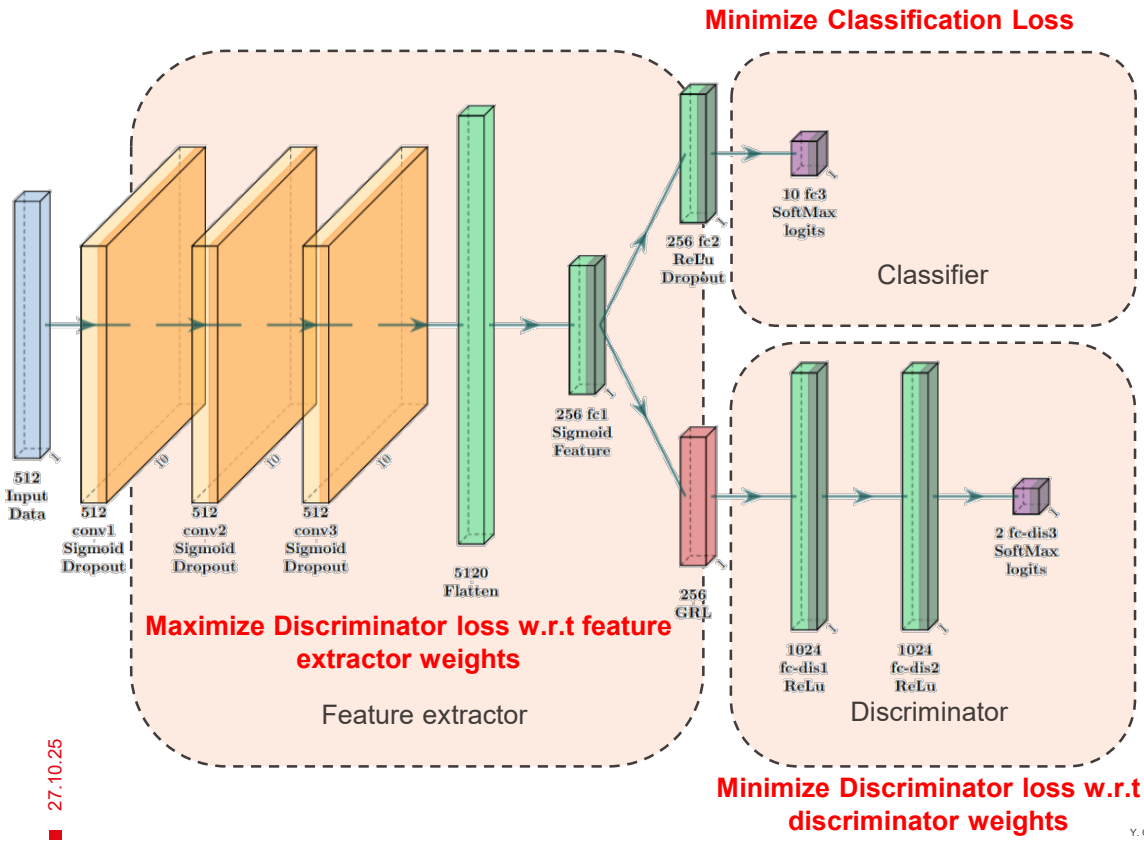
f: feature extractor

y: classifier

d: discriminator

Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," International Conference on Machine Learning. 2015.

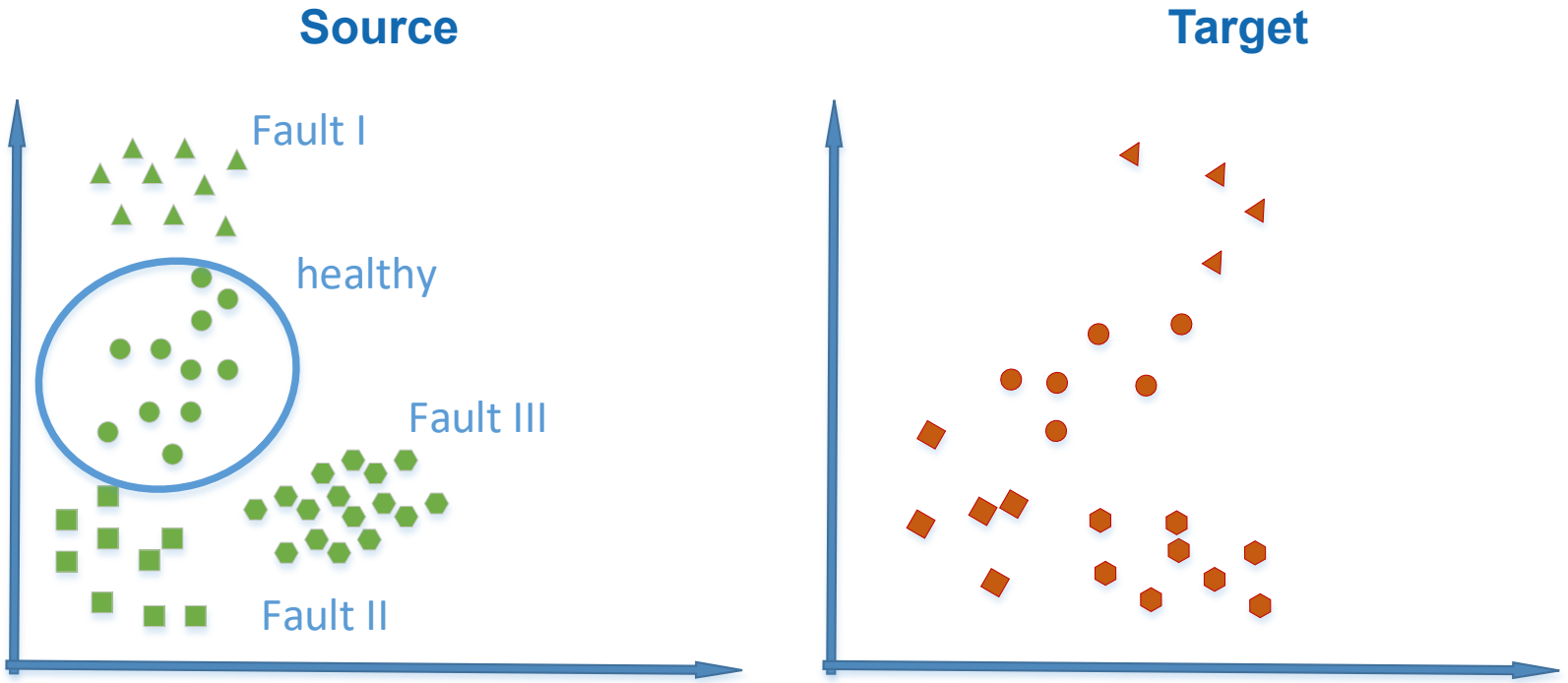
Domain-Adversarial Training (DANN)



- Strategy: Discriminator
  - Train a discriminator to distinguish between target and source
  - Force the feature extractor to generate unbiased features

Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," arXiv preprint arXiv:1409.7495, 2014

Feature alignment: all faults known



Model Performance on CWRU Dataset

- Same basic architecture
- Same budget for hyper-parameter tuning
- Same optimizer and learning rate

	Baseline	AdaBN	MMD	DANN
3-1	89.42%	94.42%	98.53%	98.41%
0-2	93.65%	99.30%	99.98%	99.96%
...	...	…	…	…
Mean Accuracy	94.99%	97.82%	99.42%	99.07%

Average over **5** runs, trained on a NVIDIA GTX 1080

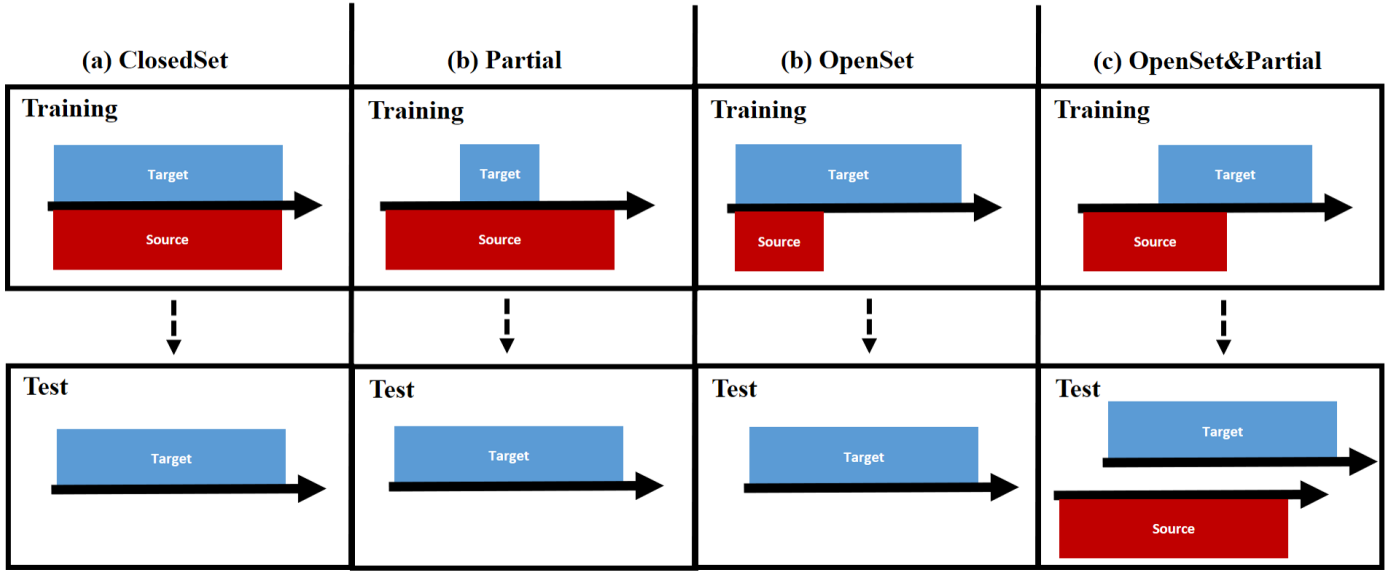
Model Efficiency

	Baseline	AdaBN	MMD	DANN
Mean Accuracy	94.99%	97.82%	99.42%	99.07%
Training Time	84 s	133 s	266 s	177 s
	Linear	Linear	Quadratic	Linear

w.r.t. # training samples

Partial / Openset Domain adaptation

Four DA configurations according to label space discrepancies



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

Standard DA

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

Partial DA (Cao et al. 2018)

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

Partial DA under «extreme» setup

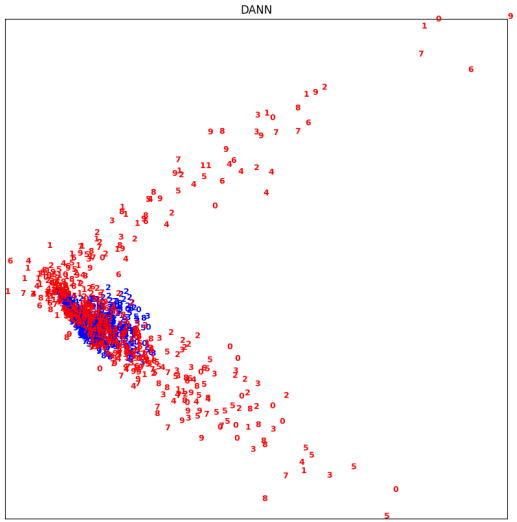
Source - Train	Target - Train	Target - Test
Healthy	Healthy	Healthy
Fault 1		Fault 1
Fault 2		Fault 2
Fault 3		Fault 3
Fault x		Fault x

Visualization on Feature Space

DA in Real Life

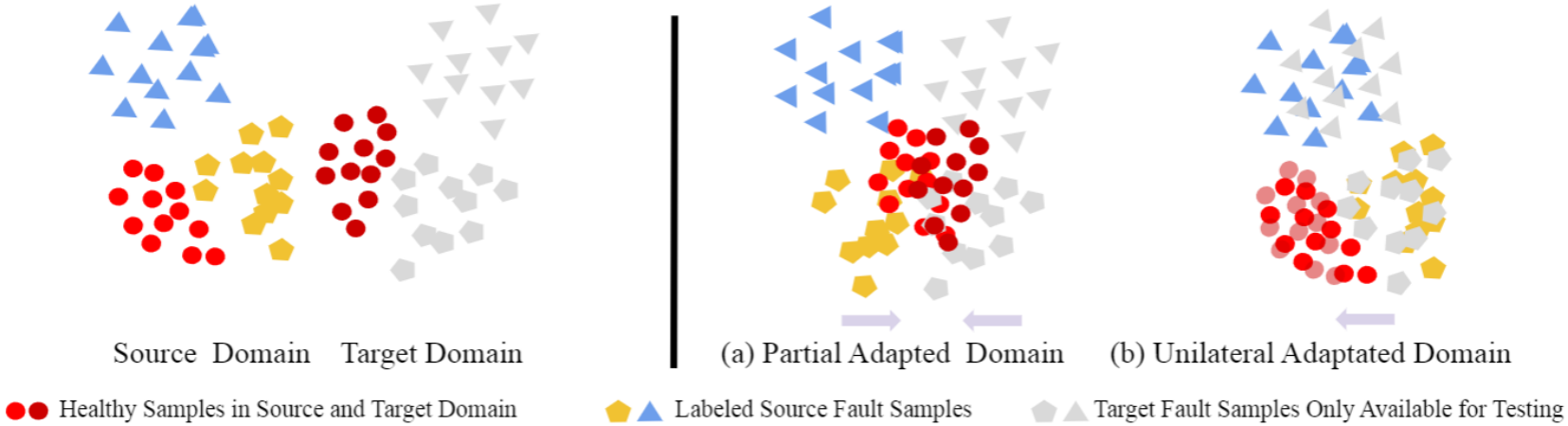
Source - Train	Target - Train	Target - Test
Healthy	Unknown	Healthy
Fault 1	Unknown	Fault 1
Fault 2	Unknown	Fault 2
Fault 3		Fault 3
Fault x		Fault x

Aligning Complete Source with Partial Target is Wrong



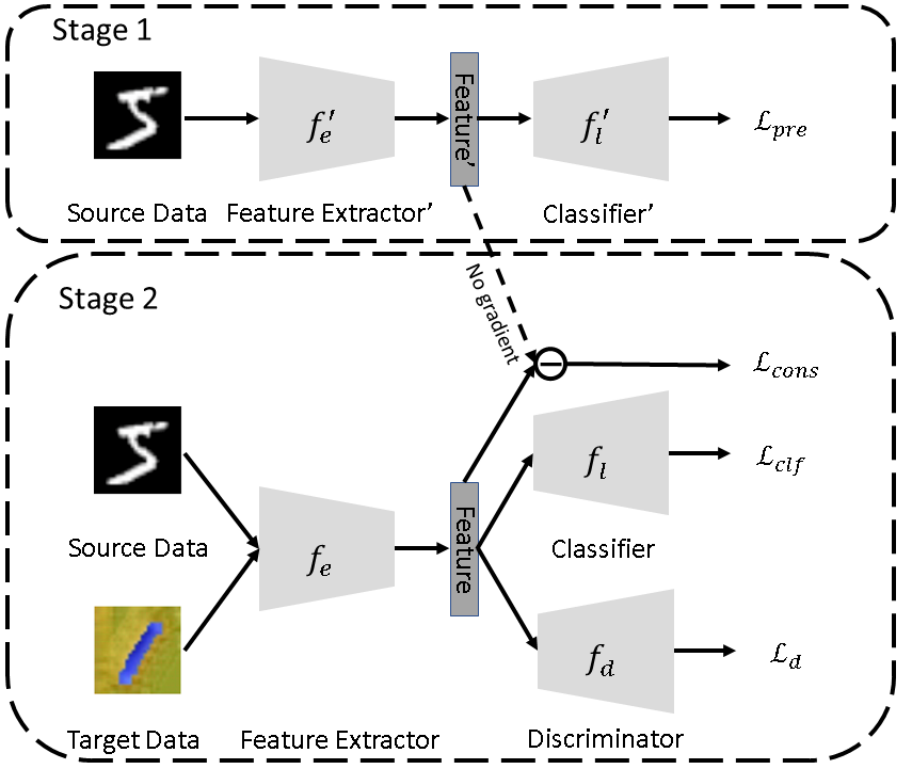
Feature Visualization on DANN Method
Part of target features (Red) are aligned with complete source data (Blue)

Our Proposed Method: Bilateral vs Unilateral



Wang, Qin, Gabriel Michau, and **Olga Fink** (2021): "Missing-Class-Robust Domain Adaptation by Unilateral Alignment", *IEEE Transactions on Industrial Electronics, 68 (1), 663-671.*

Our proposed approach



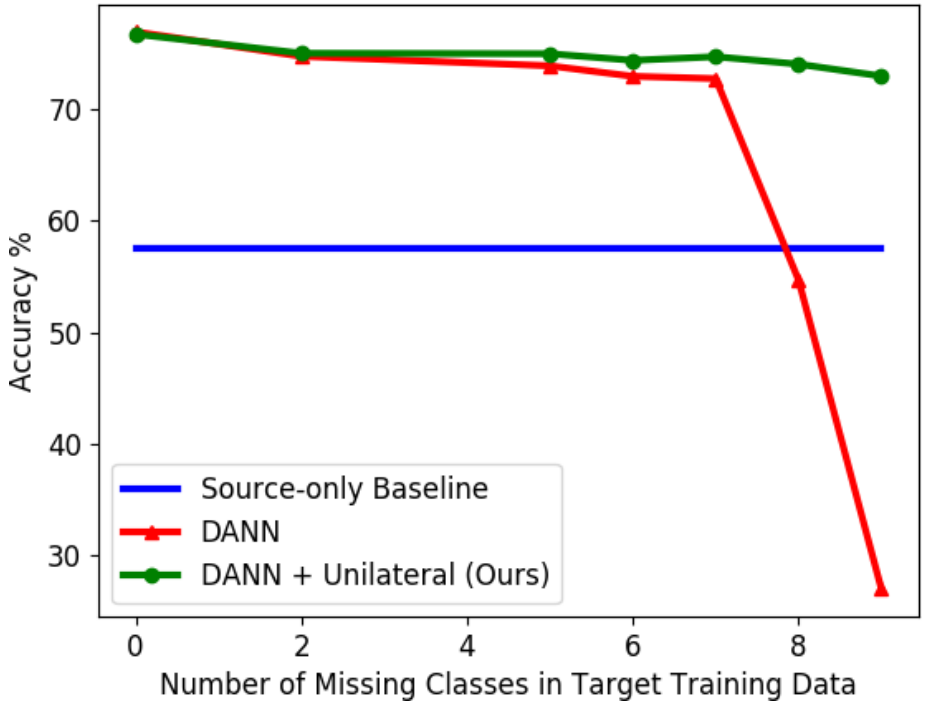
- Intuition
  - Preserve discriminability learned from source.
- Stage 1
  - Train a source feature extractor
- Stage 2
  - Train the main net
  - Make use of this additional info to construct a

$$\mathcal{L}_{cons} = \frac{1}{K}\sum_{j=1}^{K} ||f(x_s)_j - f'(x_s)_j||_1,$$

- $\mathcal{L} = \mathcal{L}_{clf} + \mathcal{L}_d + \lambda_{cons}\mathcal{L}_{cons}$. parts.

Wang, Qin, Gabriel Michau, and Olga Fink (2021): "Missing-Class-Robust Domain Adaptation by Unilateral Alignment", *IEEE Transactions on Industrial Electronics, 68 (1), 663-671.*

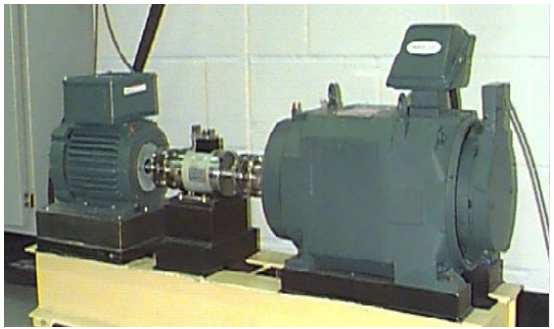
Results



Wang, Qin, Gabriel Michau, and Olga Fink (2021): "Missing-Class-Robust Domain Adaptation by Unilateral Alignment", *IEEE Transactions on Industrial Electronics, 68 (1), 663-671.*

Feature alignment: only healthy condition in target known

Transfer between operating conditions just knowing the health!



Bearing Case Study (CWRU)

- Ten Classes
  One healthy class, Nine fault classes
- Four Loads (Domains)
- Transfer between different load conditions
- Vibration Signals

- Same basic architecture
- Same budget for hyper-parameter tuning
- Same optimizer and learning rate

	Baseline	Unilateral alignment
Mean Accuracy	94.99%	**98.07%**

Wang, Qin, Gabriel Michau, and Olga Fink (2021): "Missing-Class-Robust Domain Adaptation by Unilateral Alignment", *IEEE Transactions on Industrial Electronics, 68 (1), 663-671.*