Machine Learning for Predictive Maintenance Applications:
Prognostics

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

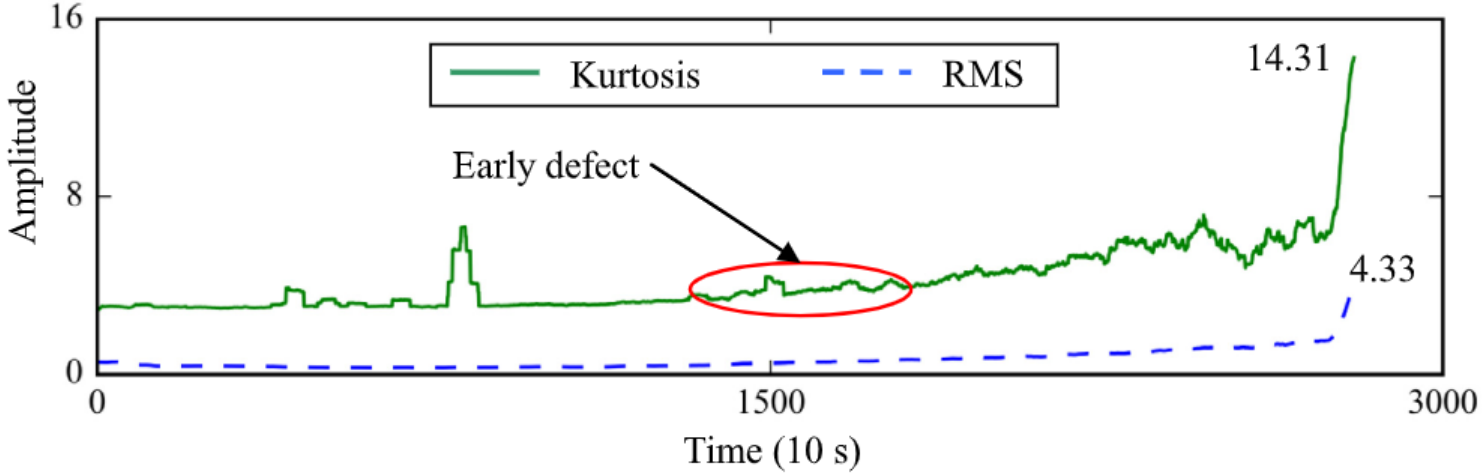
October 2024

Condition / Health indicators

Condition indicators

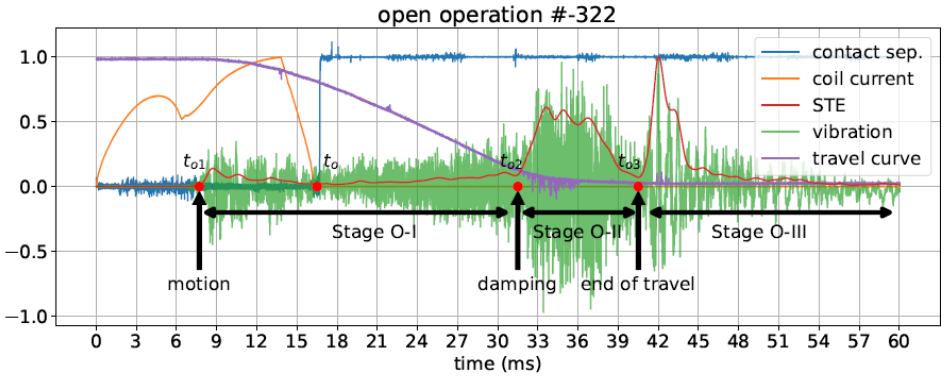
- A *condition indicator* is a feature of system data whose behavior changes in a predictable way as the system degrades or operates in different operational modes.
- A condition indicator can be any feature that is useful for distinguishing normal from faulty operation or for predicting remaining useful life.

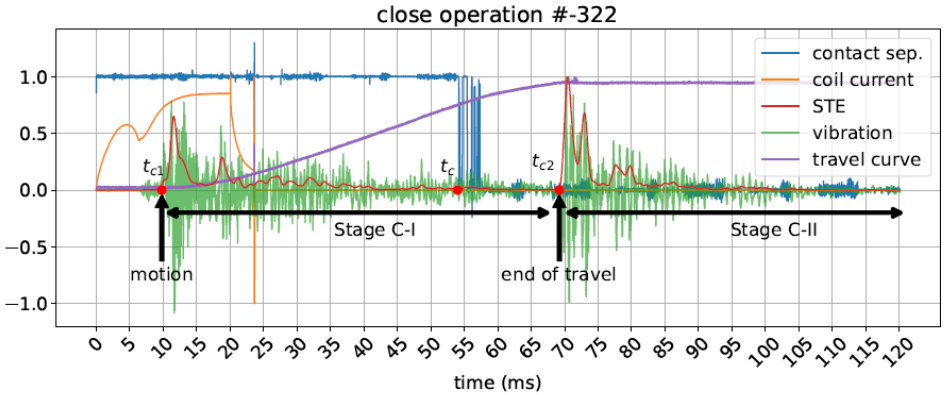
Condition indicators: example



Source: Guo et. al 2017

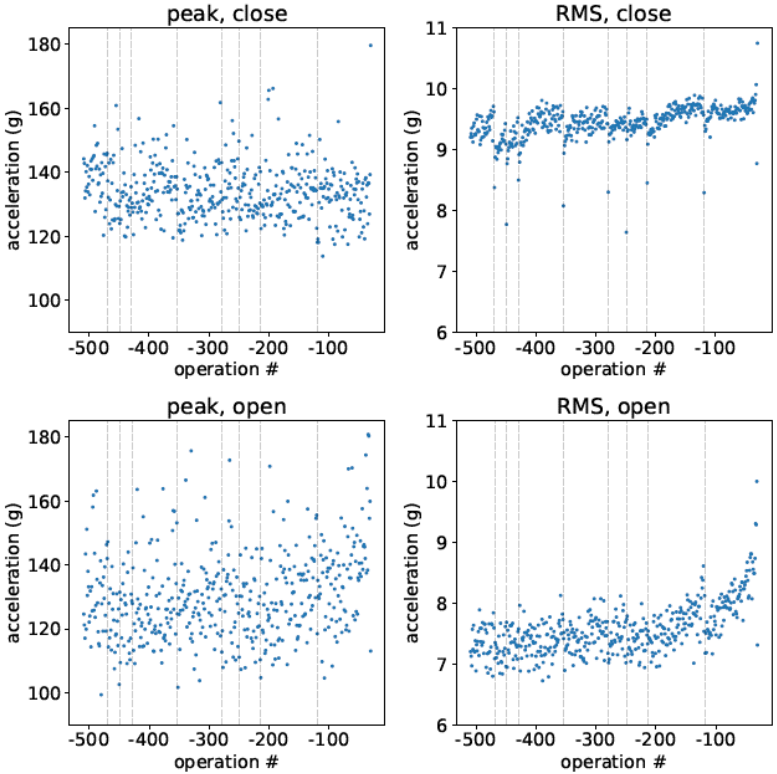
Example signals (normalised) collected during open and close operation of a circuit breaker





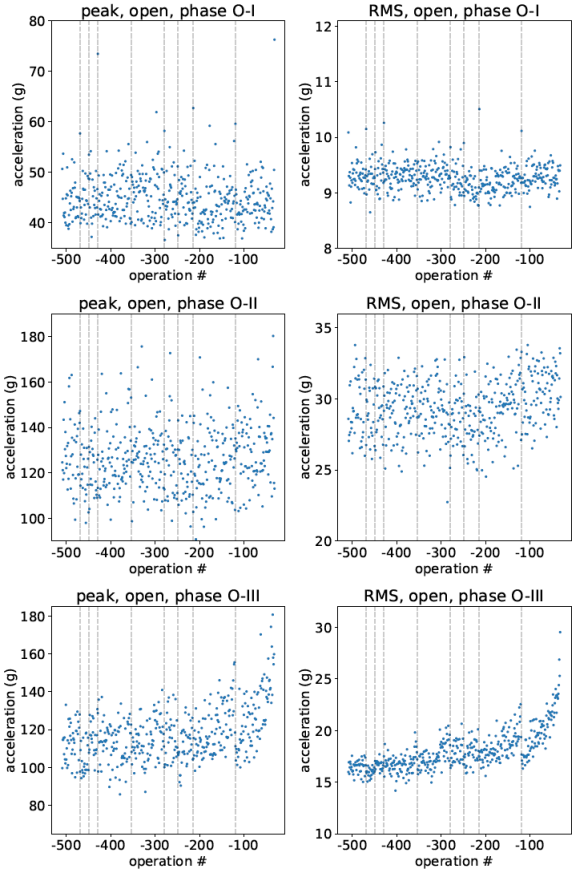
C.-C. Hsu , G. Frusque , O. Fink , & C. M. Franck: Condition Monitoring on a High Voltage Gas Circuit Breaker using Vibration Signals in a Run-to-Failure Experiment

Condition indicators of a circuit breaker



C.-C. Hsu , G. Frusque , O. Fink , & C. M. Franck: Condition Monitoring on a High Voltage Gas Circuit Breaker using Vibration Signals in a Run-to-Failure Experiment

Condition indicators of a circuit breaker



C.-C. Hsu , G. Frusque , O. Fink , & C. M. Franck: Condition Monitoring on a High Voltage Gas Circuit Breaker using Vibration Signals in a Run-to-Failure Experiment

Condition vs. Health Indicators

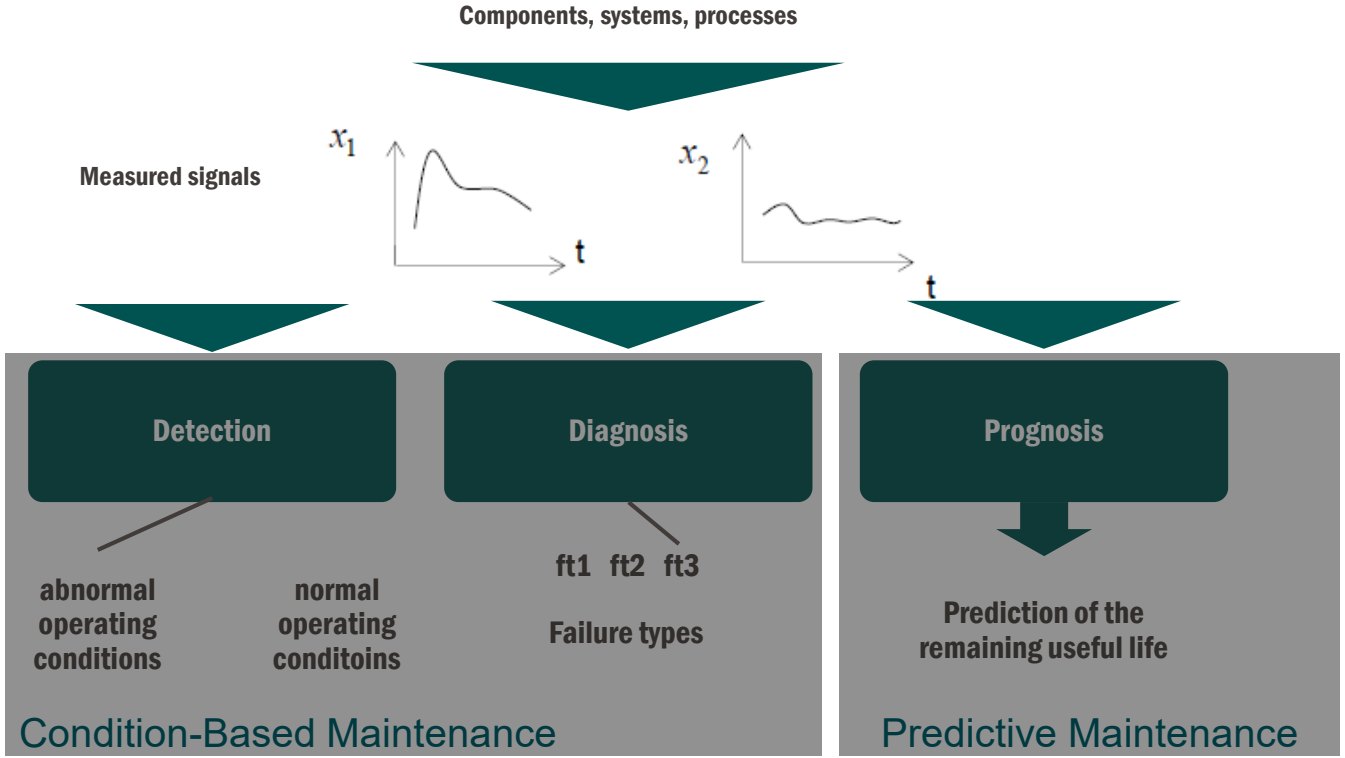
- Health indicators consist of the integration of **several condition indicators** into one value that provides the health status of the component to the end user.

Desired characteristics of health / condition indicators

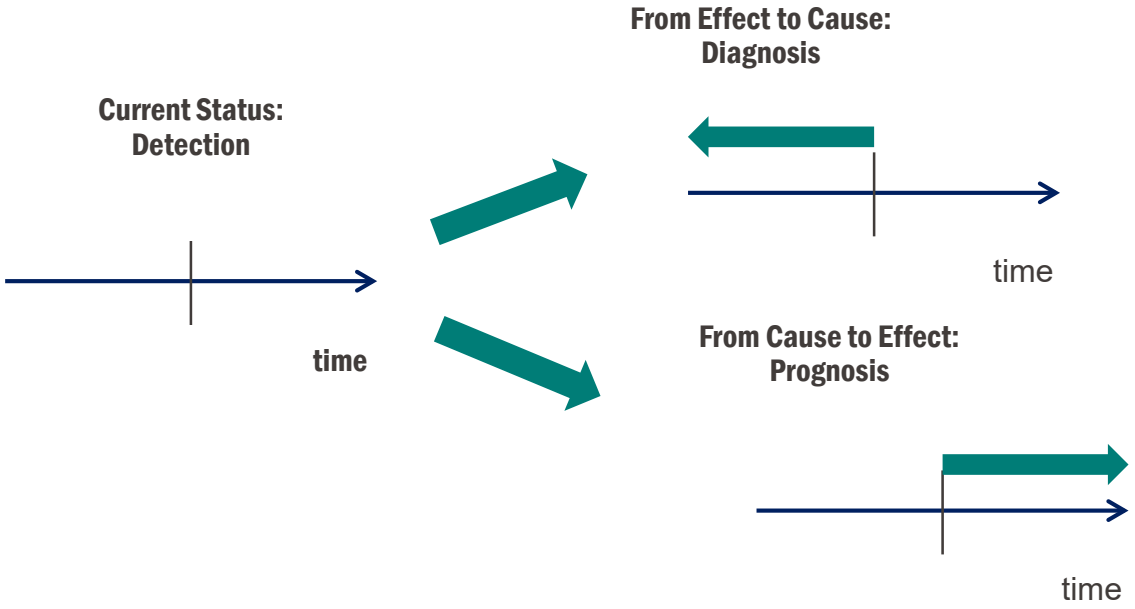
- Monotonicity
- Robusstness
- Adaptability

Prognostics

Prognostics and health management (PHM)

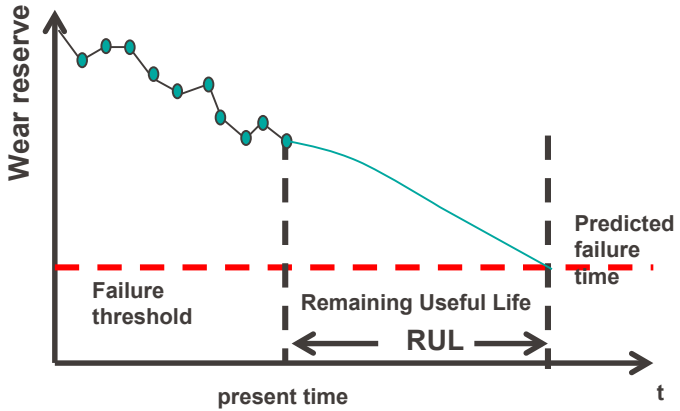


Detection, Diagnosis, Prognosis



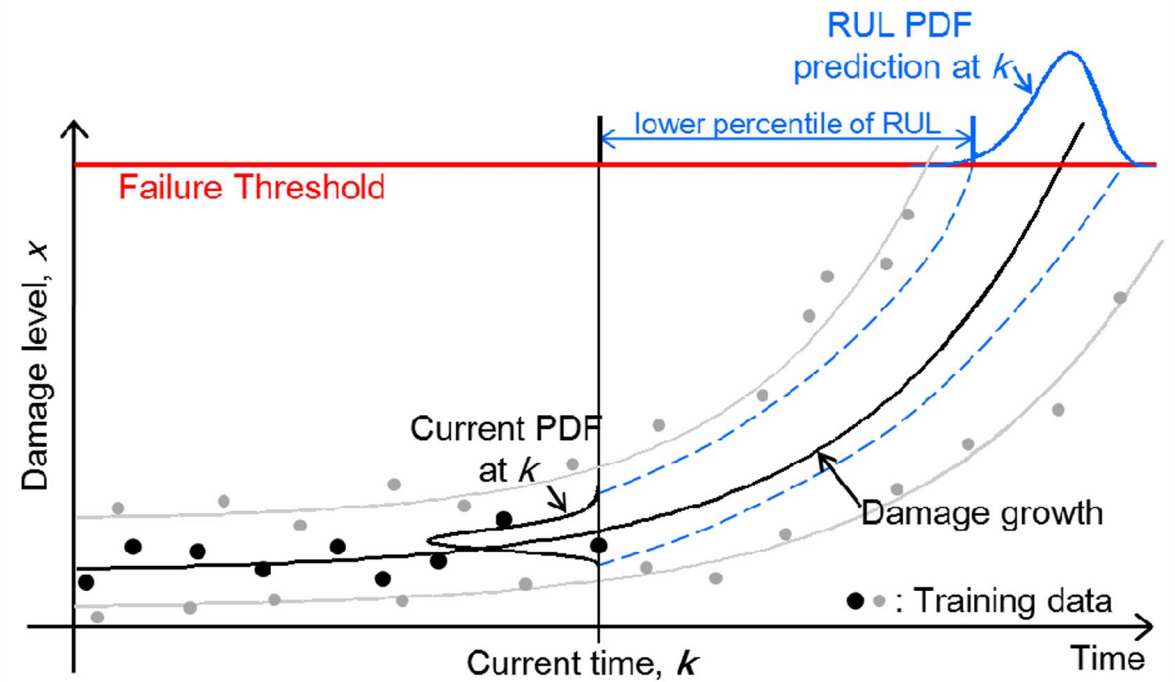
Prognostic Term Definitions

- Remaining Useful Life (RUL): the amount of time, in terms of operating hours, cycles, or other measures the component will continue to meet its design specification
- Time of Failure (ToF): the time a component is expected to fail (no longer meet its design specifications).
- Probability of Failure (PoF): the failure probability distribution of the component.
- End of Life (EoL) refers to a failure of the component as defined by its functional specifications



Source: Coble, 2016

RUL prediction



An, D., Choi, J. H., & Kim, N. H. (2018). Prediction of remaining useful life under different conditions using accelerated life testing data. *Journal of Mechanical Science and Technology*, *32*, 2497-2507.

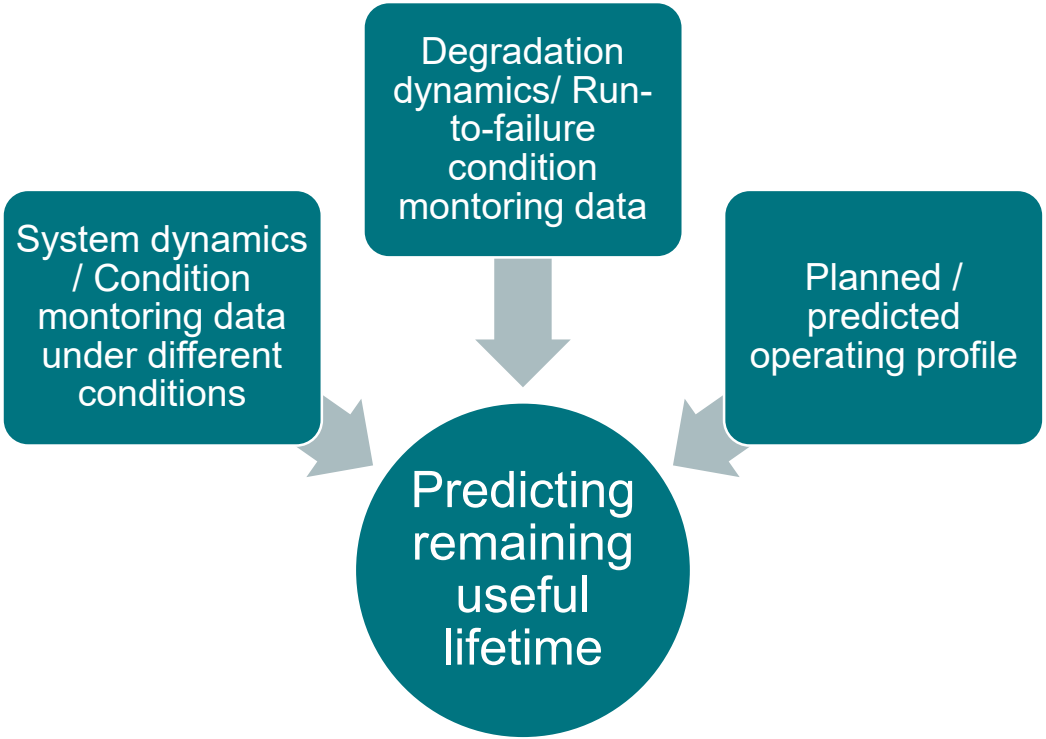
Types of failures (End of Life)

- Soft failures:
  - Soft failures refer to conditions where a system or component experiences performance degradation or intermittent issues without complete loss of functionality. These failures often manifest as a gradual decline in performance, reliability, or efficiency.
  - Intermittent Performance Issues: Systems may function sporadically, with performance fluctuating between optimal and suboptimal states.
  - Degradation Over Time: Gradual wear and tear lead to diminishing performance metrics.
  - Early Warning Signs: Often preceded by detectable anomalies or deviations in health indicators.
  - Non-Catastrophic: Do not cause immediate or severe damage but can lead to significant issues if unaddressed.
- Hard Failures
  - Hard failures occur when a system or component ceases to function entirely or experiences a catastrophic breakdown. These failures result in the immediate and complete loss of functionality, often requiring significant repairs or replacements.
  - Sudden and Complete Failure: Systems stop working abruptly without prior warning.
  - Catastrophic Impact: Can cause extensive damage to equipment, infrastructure, or even pose safety hazards.
  - High Severity: Often lead to significant operational disruptions and financial losses.
  - Immediate Action Required: Necessitate prompt intervention to restore functionality and prevent further damage.
- Other terms used in the context of system failures:
  - Warning
  - Alarm

Definition prognostics

- *Predict progression of the system health state based on current and future operational and environmental conditions to estimate the time at which a system no longer fulfils its function within desired specifications ("Remaining Useful Life")*

Required ingredients for successful prefiction of remaining useful lifetime (RUL)



Prognostics vs Trending vs Predictive Diagnostics

- **Prognostics:** Predict the remaining useful life or time to failure of a failing component/system
- **Trending:** Trend or linearly project/regress a current measurement until it reaches a predefined threshold
- **Predictive Diagnostics**: Find precursors to failure

Algorithm selection

End-of-Life predictions

Event predictions → RUL prediction

Decay /degradation prediction → trajectory prediction

No/little history data → mainly model-based

History data –> data-driven methods can be applied

Nominal data only

Nominal and failure data

Prognostic Algorithm Categories

- Type I: Reliability Data-based
  - Use population based statistical model
  - These methods consider historical time to failure data which are used to model the failure distribution. They estimate the life of a typical component under nominal usage conditions.
  - Example: Weibull Analysis

- Type II: Stress-based
  - Use population based fault growth model – learned from accumulated knowledge
  - These methods also consider the environmental stresses (temperature, load, vibration, etc.) on the component. They estimate the life of an average component under specific usage conditions.
  - Example: Proportional Hazards Model

- Type III: Condition-based
  - Individual component based data-driven model
  - These methods also consider the measured or inferred component degradation. They estimate the life of a specific component under specific usage and degradation conditions.
  - Example: Cumulative Damage Model, Filtering and State Estimation

Source: Goebel, 2012

Reliability and Prognostics

- Reliability analysis gives us information about the failure of a population of similar systems or components
- Prognostics extends this to a specific system or component
- When will it fail?
- What's the probability that it will fail in the next 5 minutes?
- What's the probability that we can complete the mission before something fails?

Source: Coble, 2016

Type III – Degradation-based Prediction

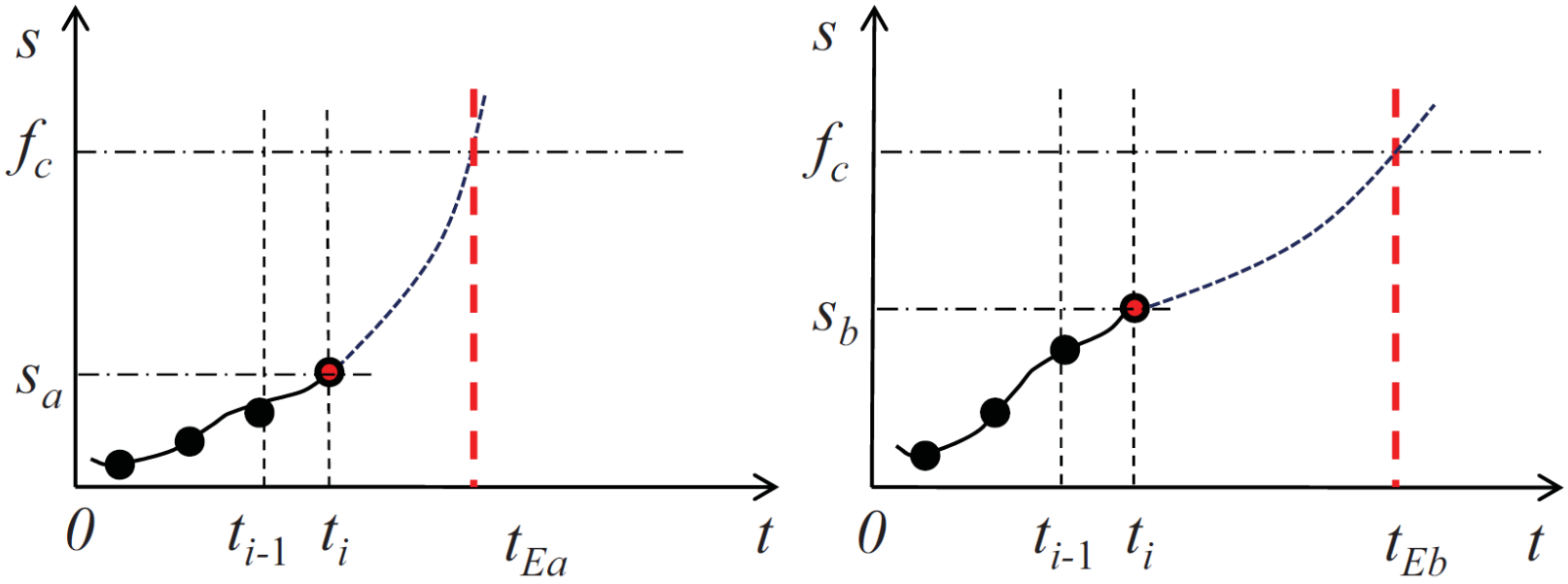
- Type III prognostics estimate the lifetime of the specific component in its specific operating environment
- Type III algorithms track the degradation (damage) as a function of time and predict when the total damage will exceed a predefined threshold that defines failure
- Damage is generally assumed to be cumulative (irreversible)

Degradation-Based Prognostics

- A degradation measure is a scalar or vector quantity that numerically reflects the current ability of the system to perform its designated functions properly. It is a quantity that is correlated with the probability of failure at a given moment.
- A degradation path is a trajectory along which the degradation measure is evolving in time towards the critical level corresponding to a failure event.

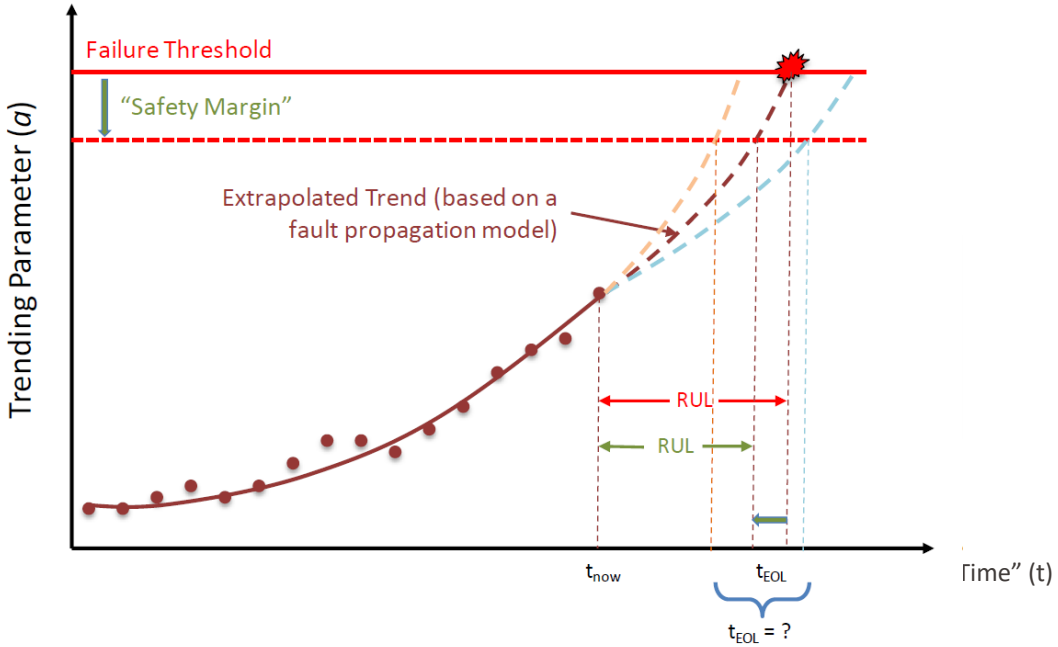
Source: Coble, 2016

Health indicator vs. RUL



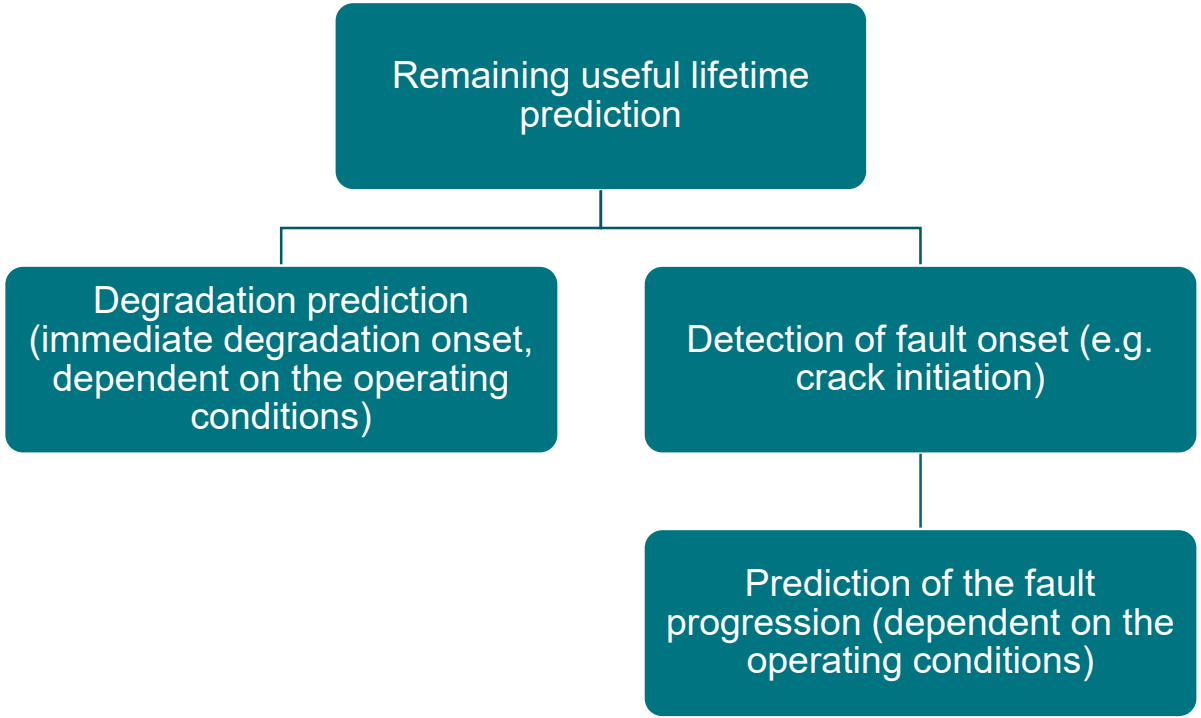
Wang, T., Trajectory Similarity Based Prediction for Remaining Useful Life Estimation, 2010

Trending the degradation parameter

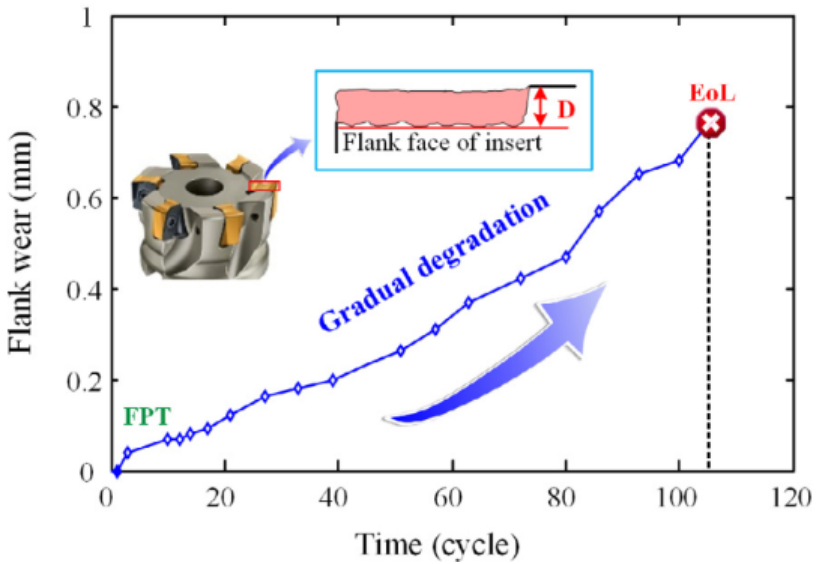


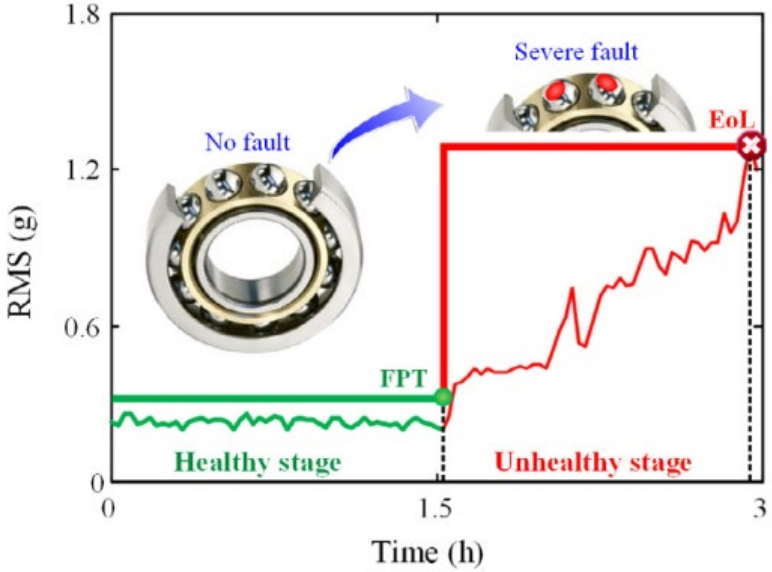
Source: Goebel, 2012

Prognostics: degradation and fault progression



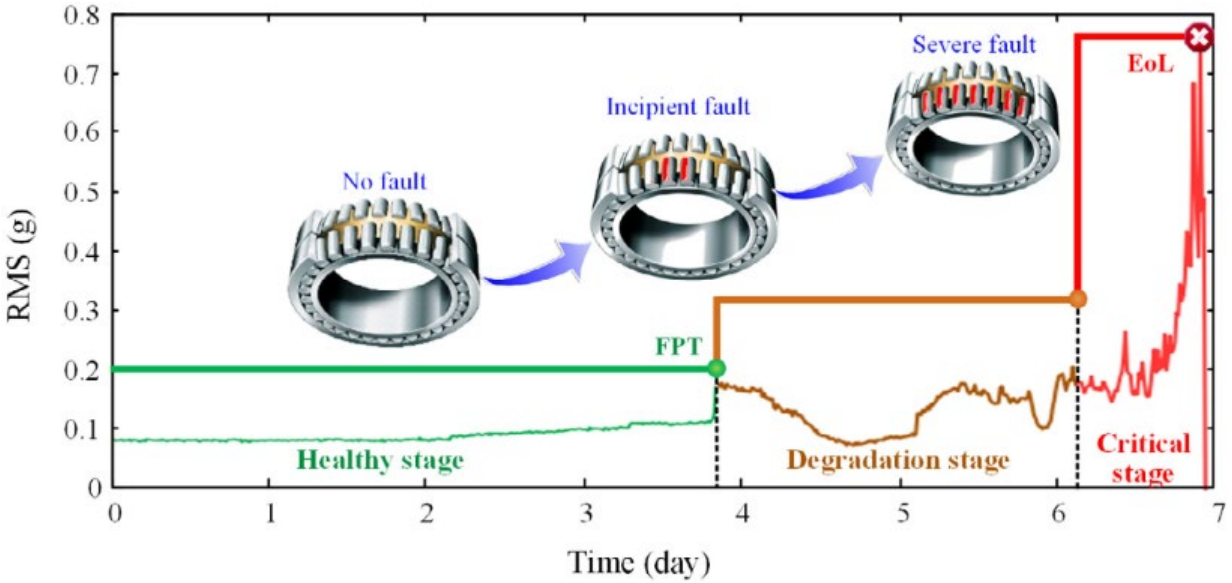
Different degradation trajectories





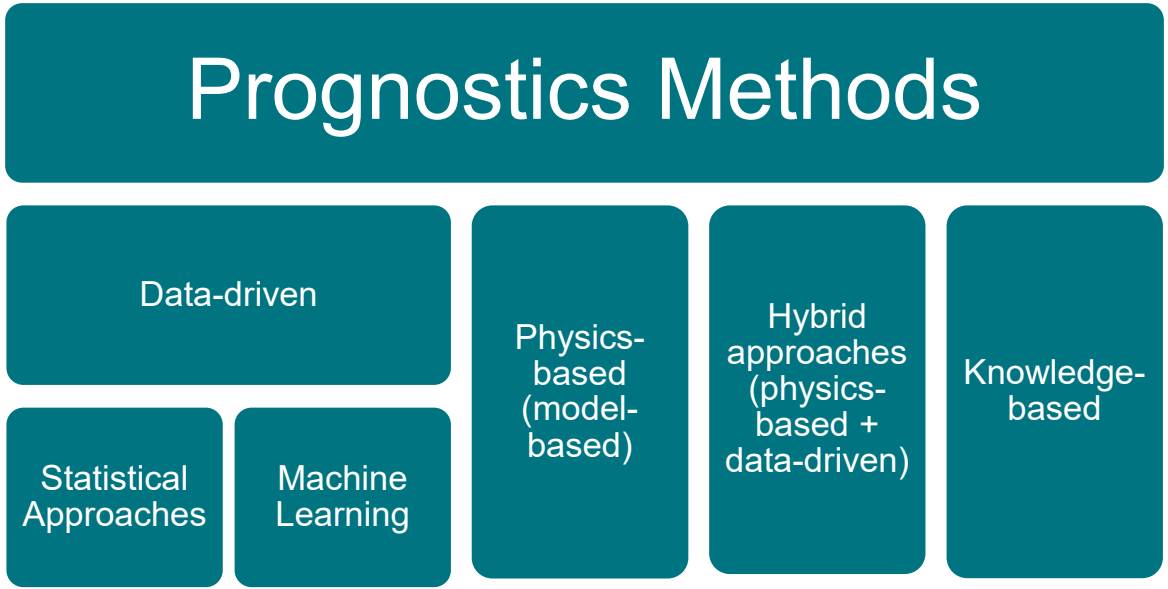
Source: Lei et al, 2018

Different degradation trajectories

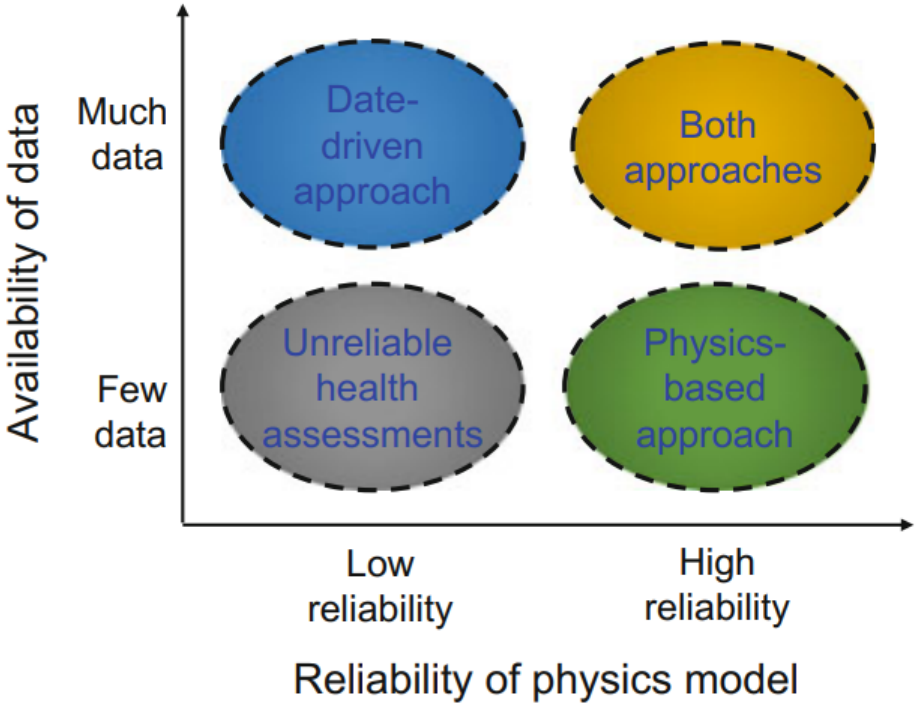


Source: Lei et al, 2018

Prognostics Methods

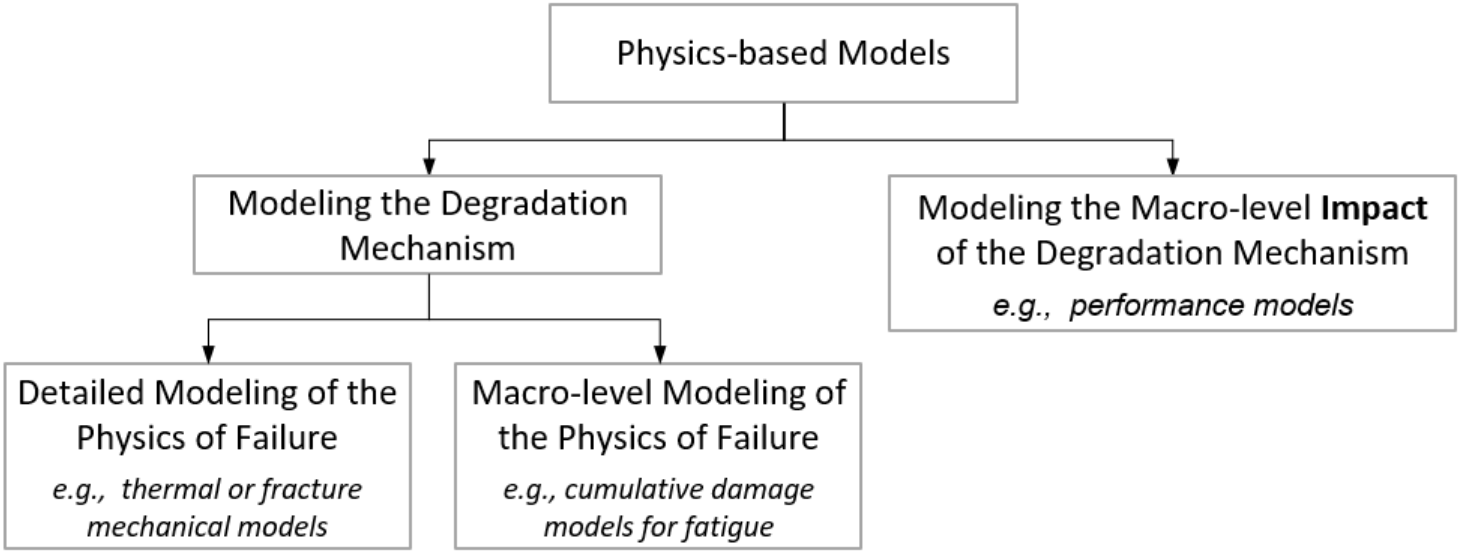


Choice of Methods



Source: Prognostics and Health Management of Engineering Systems

Physics-based models



M. Arias, Algorithm for Fleet Diagnostics and Prognostics Combining Deep Learning and Physics-based Performance Models (2021)

Physics-Based Methods

- Description of a system's underlying physics using suitable representation
- Some examples:
  - Model derived from "First Principles"
    - Encapsulate fundamental laws of physics
      - partial differential equations (PDEs)
      - Euler-Lagrange Equations
  - Empirical model chosen based on an understanding of the dynamics of a system
    - Lumped Parameter Model
    - Classical 1st (or higher) order response curves
  - Mappings of stressors onto damage accumulation
    - Finite Element Model
    - High-fidelity Simulation Model
- Something in the model correlates to the failure mode(s) of interest

Source: Goebel, 2012

Steps for physics-based prognostics

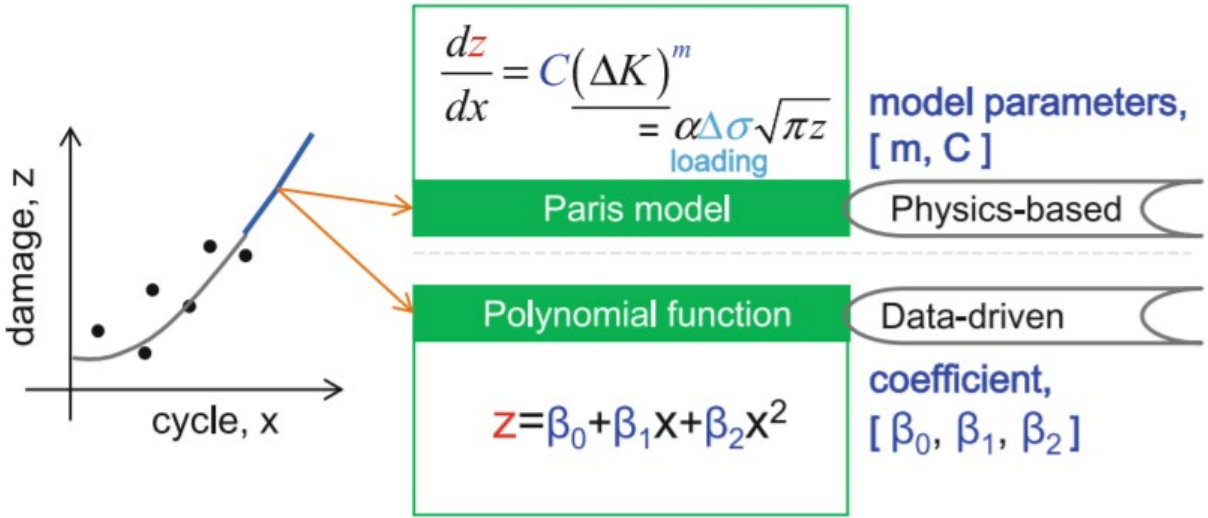
- Model underlying physics of a component/subsystem
- Model physics of damage propagation mechanisms
- Determine criteria for End-of-Life threshold
- Develop algorithms to propagate damage into future
- Deal with uncertainty

Data-Driven Methods

- Model is based solely on data collected from the system
- Some system knowledge may still be handy:
  - What the system ‘is’
  - What the failure modes are
  - What sensor information is available
  - Which sensors may contain indicators of fault progression (and how those signals may ‘grow’)
- *General* steps:
  - Gather what information you can (if any)
  - Determine which sensors give good trends
  - Process the data to “clean it up” – try to get nice, monotonic trends
  - Determine threshold(s) either from experience (data) or requirements
  - Use the model to predict RUL
    - Regression / trending
    - Mapping (e.g., using a neural network)
    - Statistics

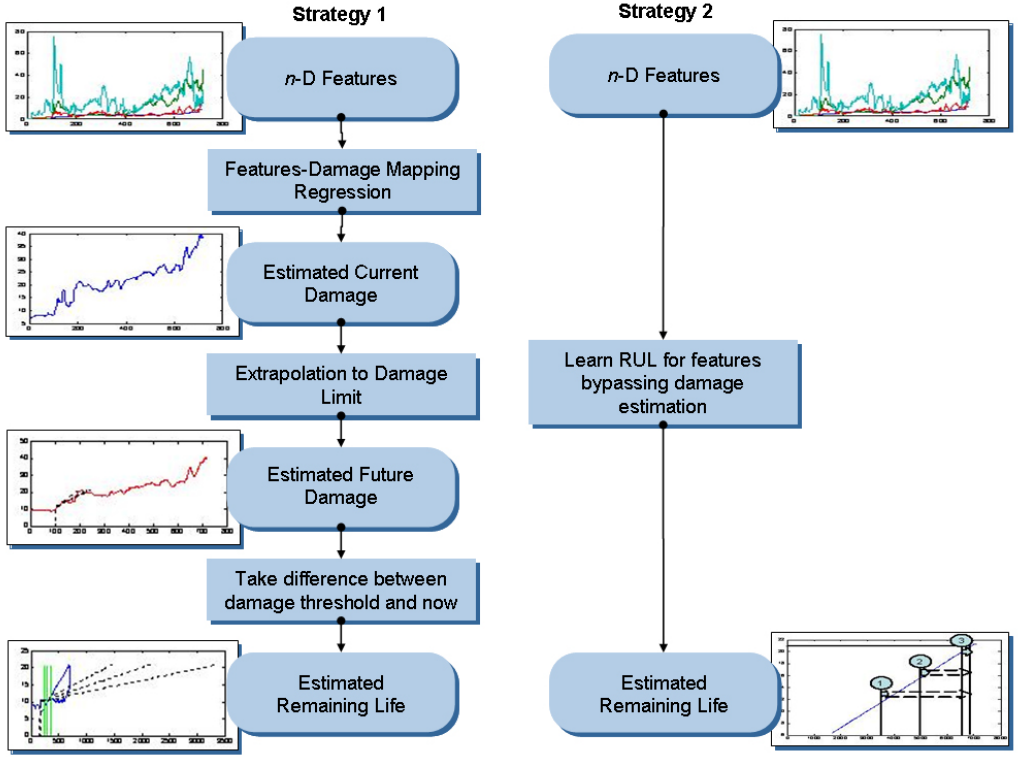
Source: Goebel, 2012

Simple example of data-driven vs. physics-based (crack growth model)



Source: Integrated Vehicle Health Management

Different approaches to data-driven RUL prediction



Source: NASA

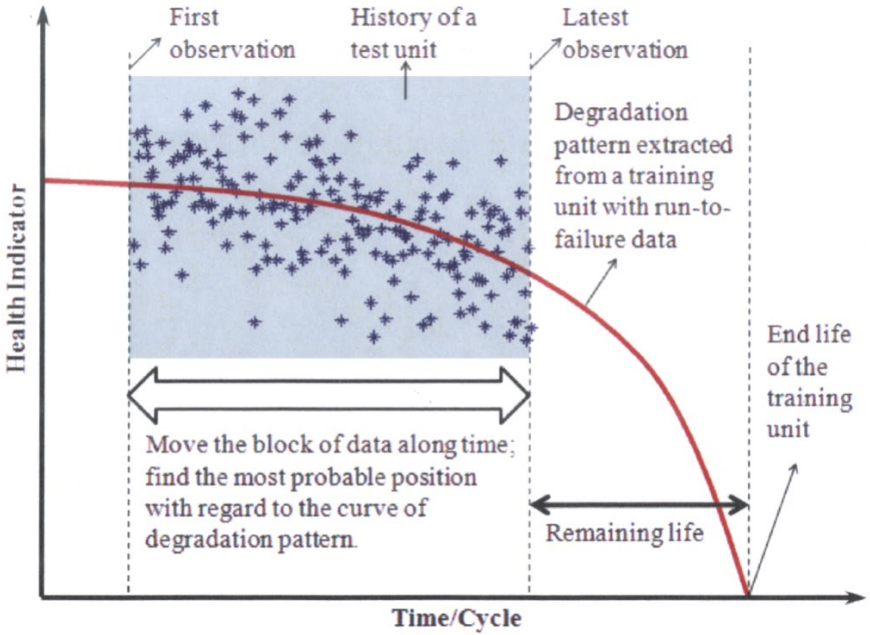
Similarity-based RUL prediction

- Leverages the concept of comparing the current state of a system with historical data to estimate the remaining useful life.
- Underlying assumption is that systems with similar degradation patterns will exhibit similar RUL trajectories.
- Similarity-based approaches generally involve the following steps:
- Data Collection and Preprocessing:
  - Historical Data: Gather comprehensive historical operational and failure data.
  - Feature Extraction: Identify and extract relevant health indicators or features that represent the system's state.
  - Normalization: Standardize data to ensure consistency across different datasets.
- Similarity Measurement:
  - Distance Metrics: Utilize metrics such as Euclidean distance, Dynamic Time Warping (DTW), or Mahalanobis distance to quantify similarity between current and historical states.
  - Feature Space Comparison: Assess similarity in a multidimensional feature space where each dimension corresponds to a health indicator.
- RUL Estimation:
  - Weighted Averaging: Compute RUL based on the weighted average of RULs from the most similar historical instances.
  - Nearest Neighbors: Identify the 'k' most similar historical cases and use their RULs to predict the current RUL.

Examples of possible approaches

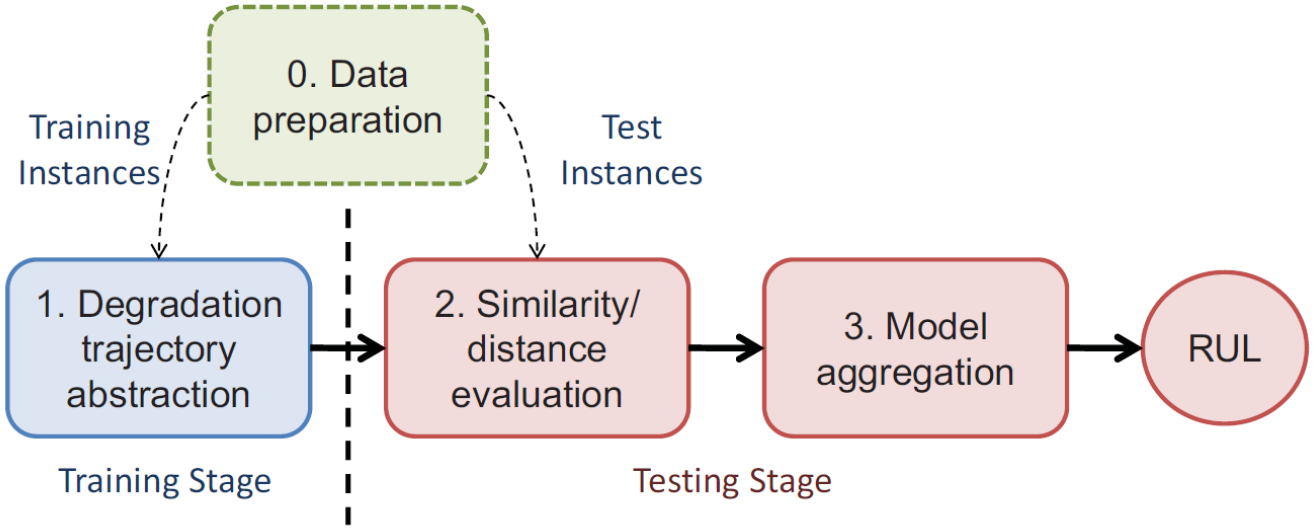
- **K-Nearest Neighbors (K-NN):**
  - Description: Identifies the 'k' most similar historical instances to the current state and averages their RULs.
  - Advantages: Simple to implement and interpret.
  - Limitations: Sensitive to the choice of 'k' and distance metrics; may not capture complex degradation patterns.
- **Pattern Matching and Template Matching:**
  - Description: Compares current degradation patterns with predefined templates representing typical failure modes.
  - Advantages: Can effectively recognize known failure patterns.
  - Limitations: Limited to detecting only those failure modes represented by templates; lacks flexibility for novel failures.
- **Cluster-Based Similarity:**
  - Description: Groups historical data into clusters and predicts RUL based on the cluster membership of the current state.
  - Advantages: Reduces computational complexity by limiting comparisons within relevant clusters.
  - Limitations: Requires effective clustering algorithms; may not handle overlapping or dynamic clusters well.

Similarity based RUL prediction



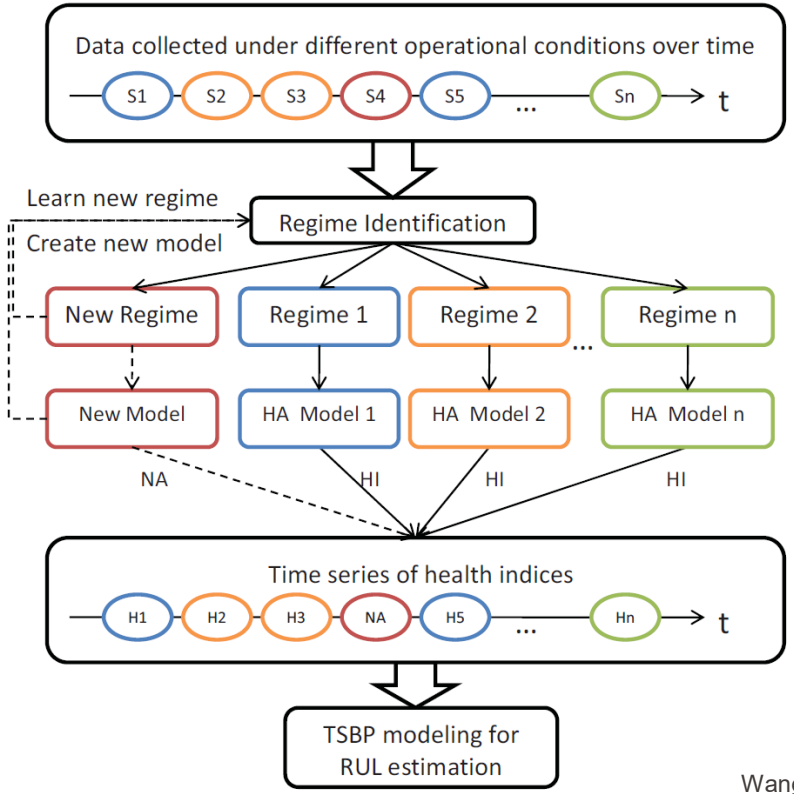
Source: T. Wang, et al. 2008

Similarity based RUL predictions



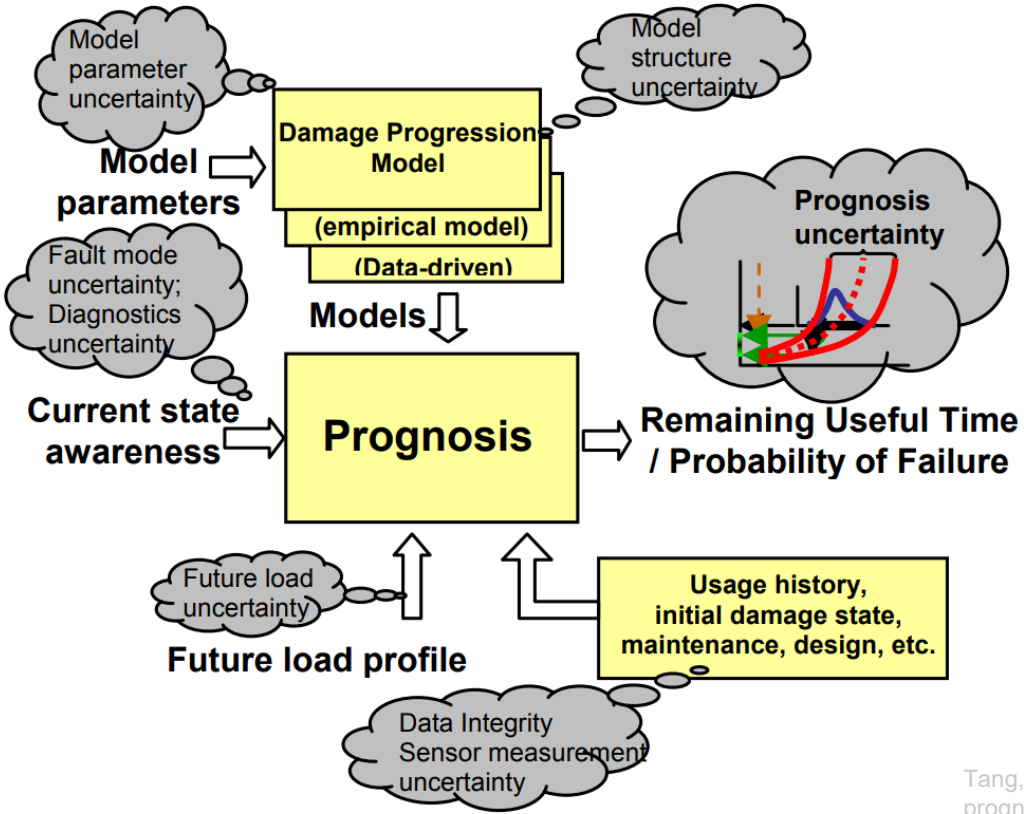
Wang, T., Trajectory Similarity Based Prediction for Remaining Useful Life Estimation, 2010

Similarity based RUL predictions



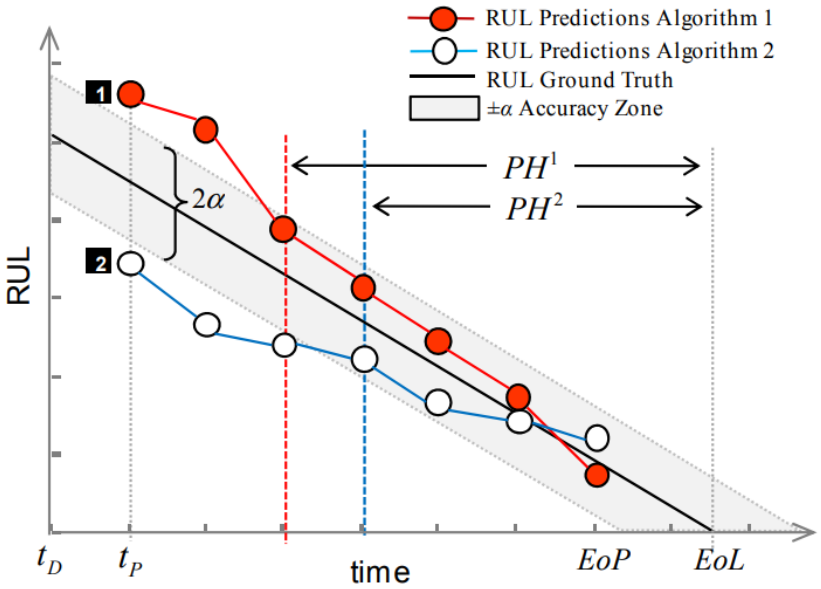
Wang, T., Trajectory Similarity Based Prediction for Remaining Useful Life Estimation, 2010

Sources of uncertainty in prognostics



Tang, Liang, et al. "Methodologies for uncertainty management in prognostics." *2009 IEEE Aerospace conference*. IEEE, 2009.

Prognostic Horizon



$$PH = EoL - i$$

where:

$i = \min\{j \mid (j \in \ell) \wedge (r_* - EoL^*\alpha) \le r^l(j) \le r_* + EoL^*\alpha)\}$ is the first time index when predictions satisfy α-bounds

ℓ is the set of all time indexes when a prediction is made

l is the index for l^{th} unit under test (UUT)

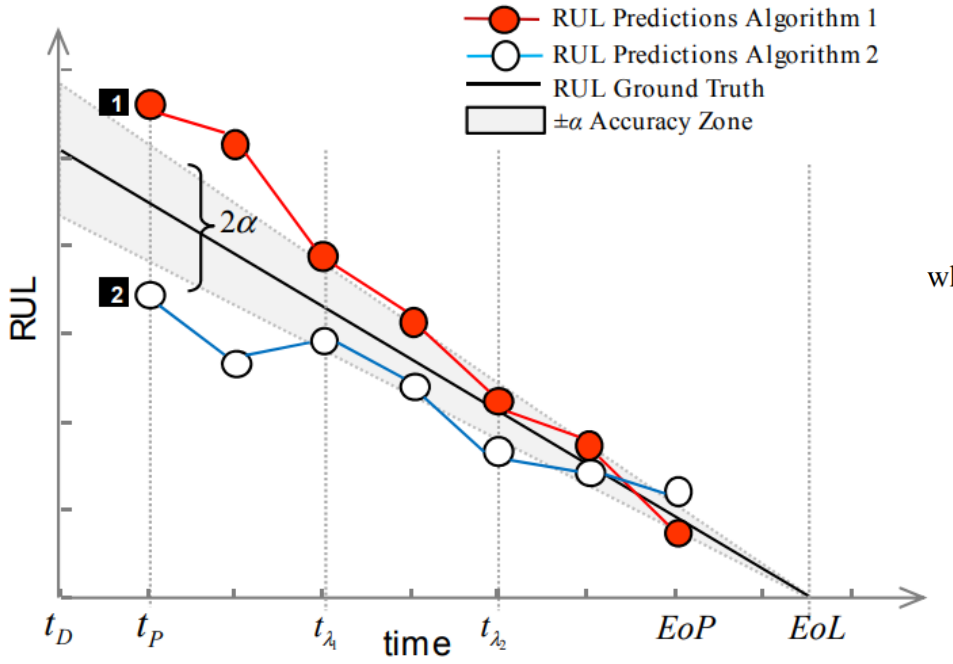
r_* is the ground truth RUL

$r(j)$ is the predicted RUL at time j

EoL is the ground truth End-of-Life (actual failure)

Saxena, A., Celaya, J., Saha, B., Saha, S. and Goebel, K., 2009. On applying the prognostic performance metrics. In *Annual Conference of the PHM Society* (Vol. 1, No. 1).

α- λ Accuracy



$$(1-\alpha)\cdot r_*(t) \le r^l(t_\lambda) \le (1+\alpha)\cdot r_*(t)$$

where:

α is the accuracy modifier

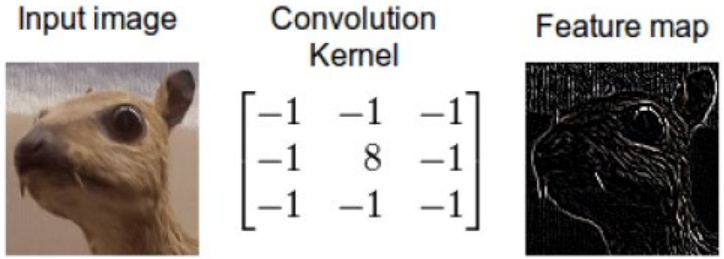
λ is a time window modifier such that $t_\lambda = t_P + \lambda(EoL - t_P)$

Saxena, A., Celaya, J., Saha, B., Saha, S. and Goebel, K., 2009. On applying the prognostic performance metrics. In *Annual Conference of the PHM Society* (Vol. 1, No. 1).

Recap: Convolutional neural networks

General intuition behind unsing convolutional filters

- Image filters can enhance image attributes
- Convolutional neural networks are similar to conventional image filtering
- Filter kernels are learnt



Source: UIO, 2017

Filters

0	0	0	0	0	0	0
0	2	4	9	1	4	0
0	2	1	4	4	6	0
0	1	1	2	9	2	0
0	7	3	5	1	3	0
0	2	3	4	8	5	0
0	0	0	0	0	0	0

Image

X

1	2	3
-4	7	4
2	-5	1

Filter / Kernel

=

21	59	37	-19	2
30	51	66	20	43
-14	31	49	101	-19
59	15	53	-2	21
49	57	64	76	10

Feature

$$O = \frac{n - f + 2p}{s} + 1$$

Where O is the output height/length, n is input height / length, f is filter size, p is the padding, and s is the stride

Pooling

6	8	6	3	1	0
9	13	10	5	2	0
9	14	11	6	3	0
9	13	11	6	2	0
8	13	10	5	3	0
6	7	5	3	1	0

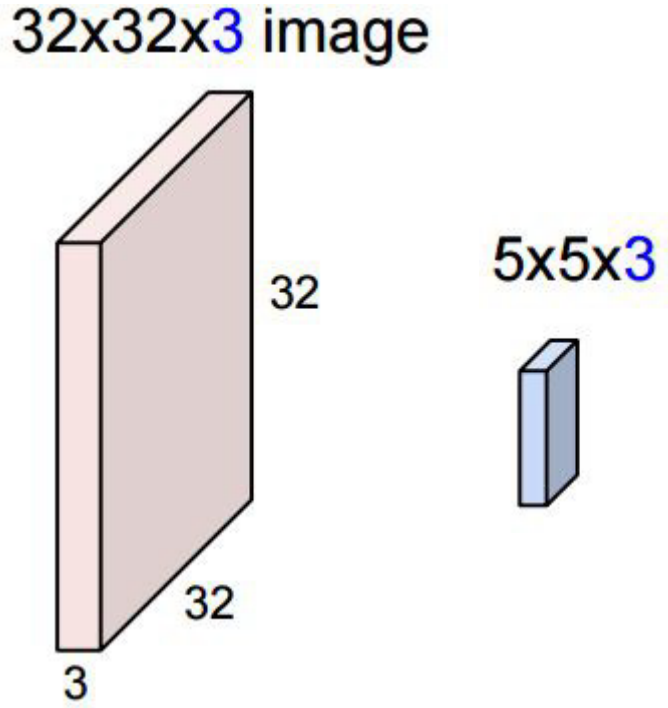
Feature map

13	10	2
14	11	3
13	10	3

Max pooling

Filters and channels (Standard method)

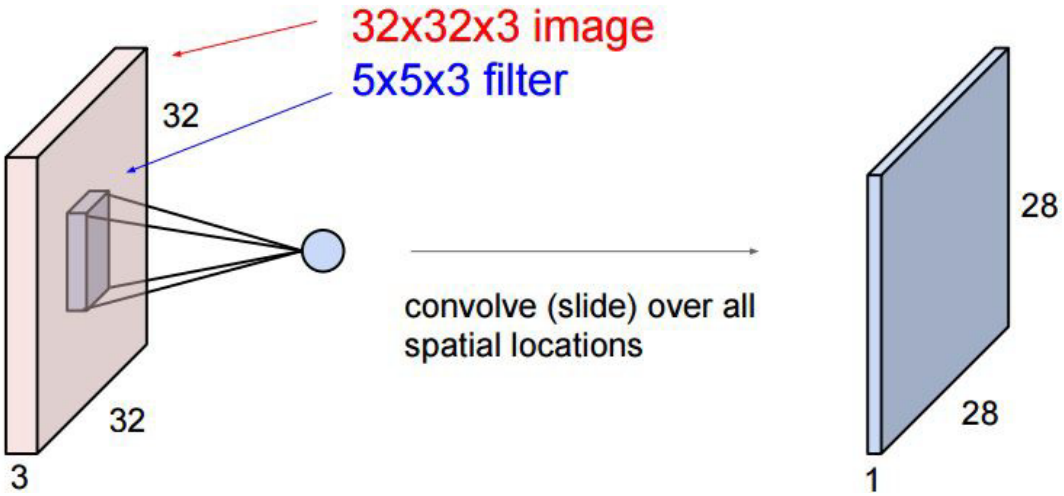
- An input image has a third dimension (say RGB)
- A filter/kernel always has the same third dimension



Source: UIO, 2017

Filters and channels

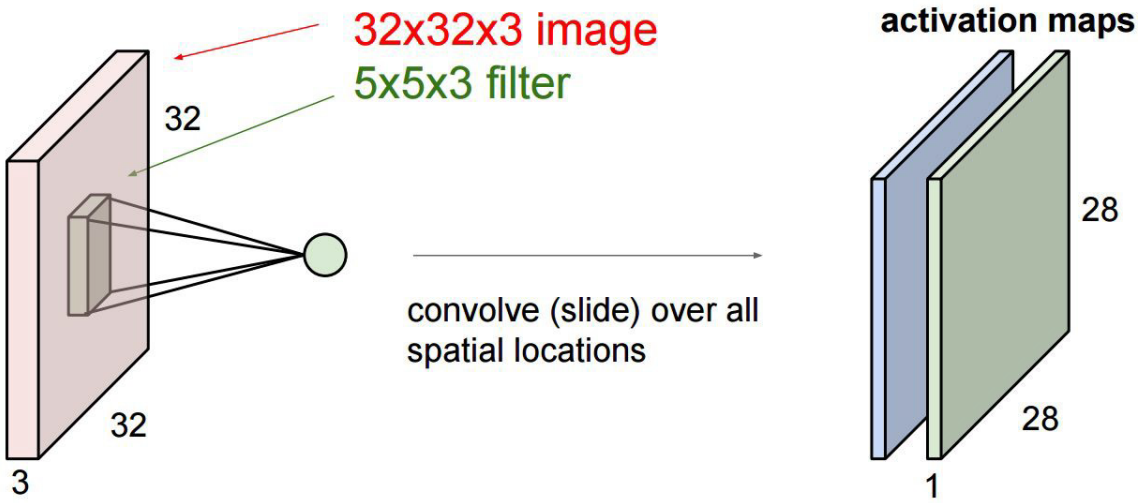
- Overlapping area is multiplied then summed (dot product)
- With sliding you get 28x28x1 output



Source: UIO, 2017

Usually we use multiple filters per layer

- A new kernel/filter slides over the same image
- Create a new filtered image

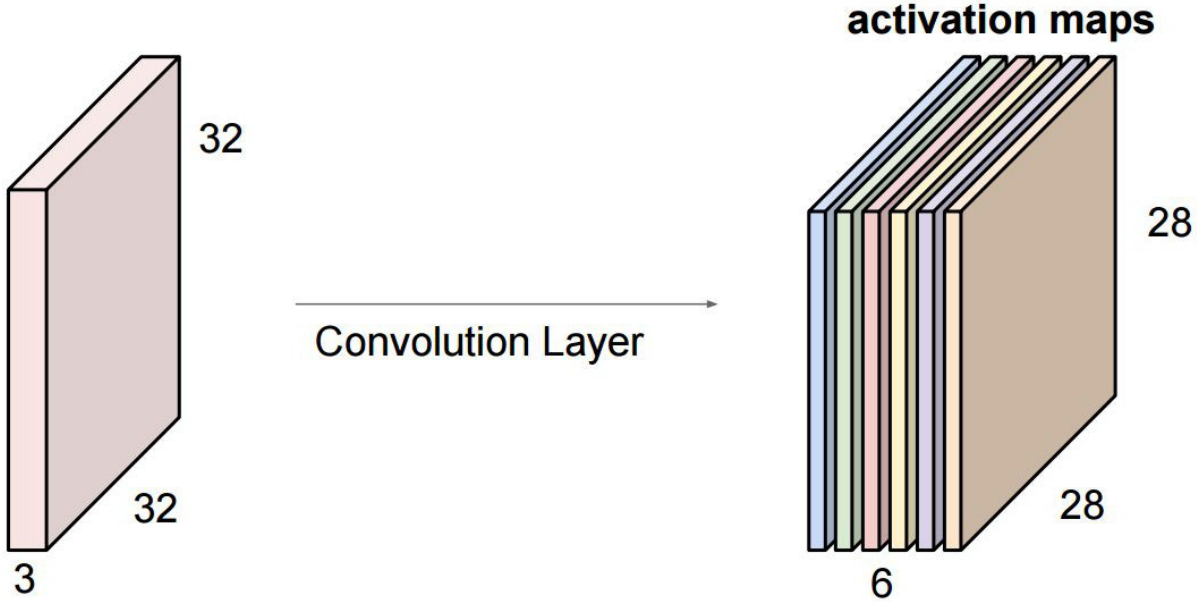


Source: UIO, 2017

Applying filters

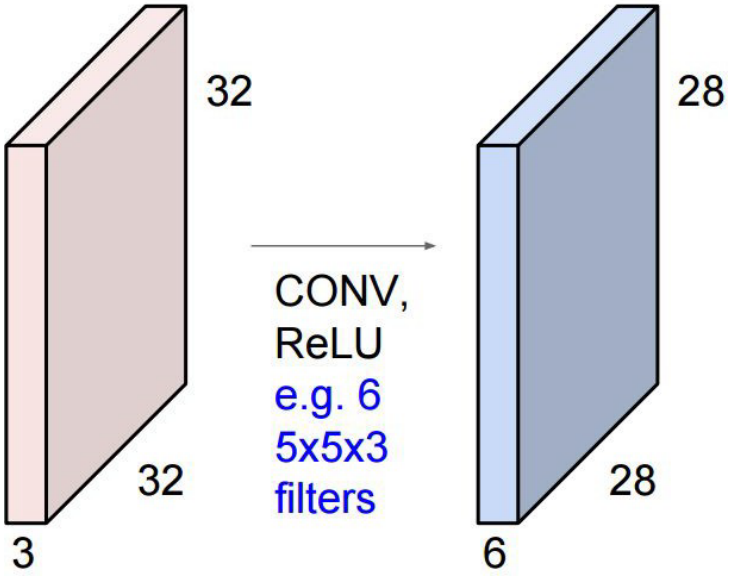
Many activation maps create a new “image”

If we filter the image 6 times, we get a new image with 6 channels.



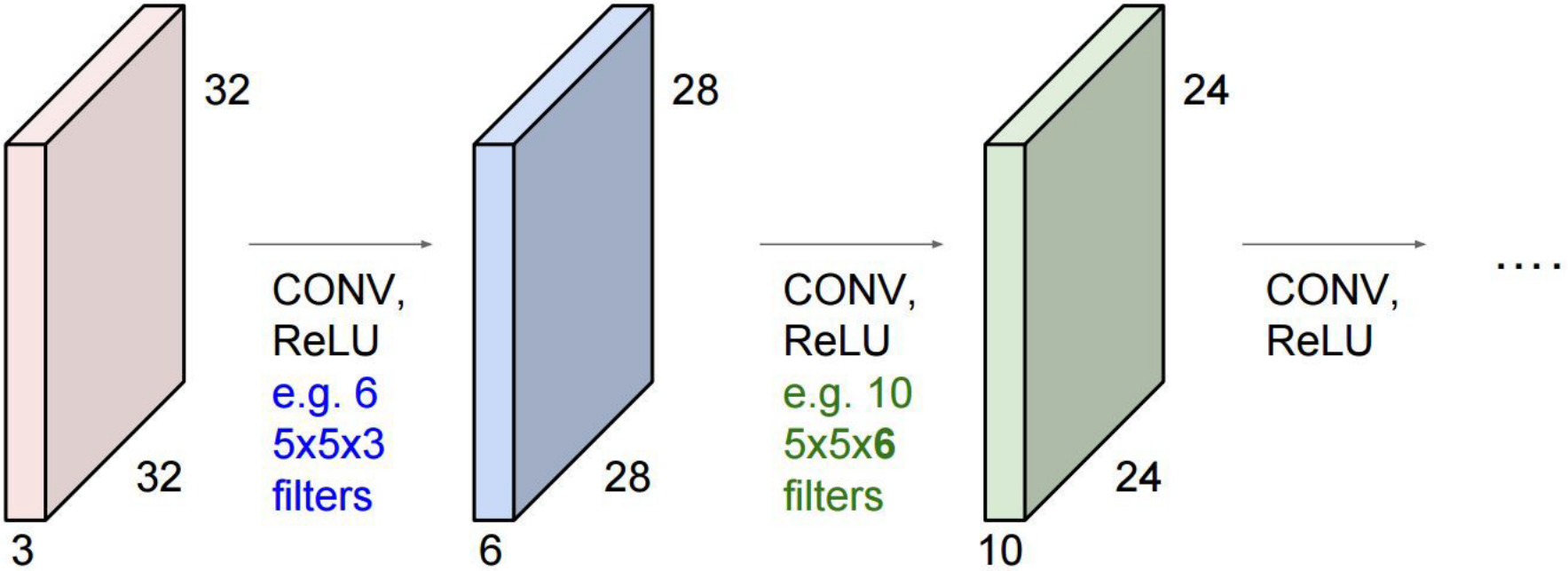
Source: UIO, 2017

Convolutional neural network consist of multiple layers



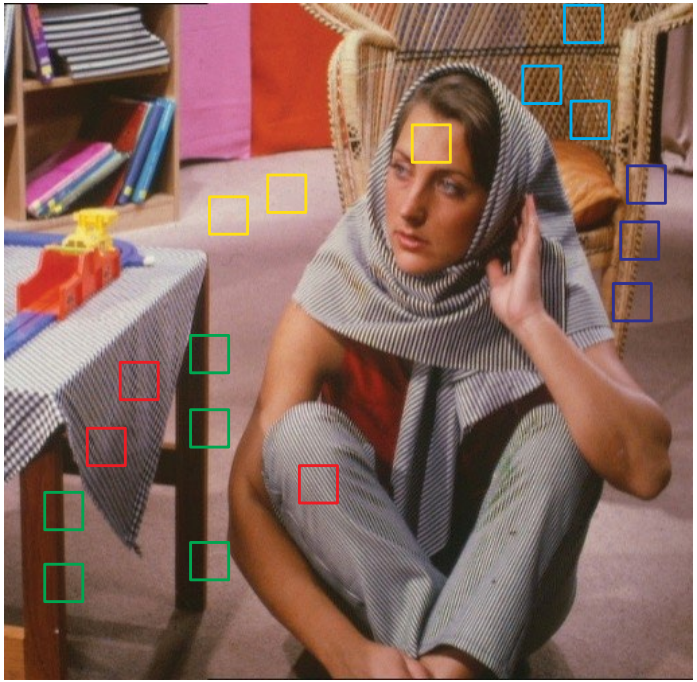
Source: UIO, 2017

Convolutional neural network consist of multiple layers



Source: UIO, 2017

Stationarity and Self-similarity



Data is self-similar across the domain

Translation invariance (image classification tasks)

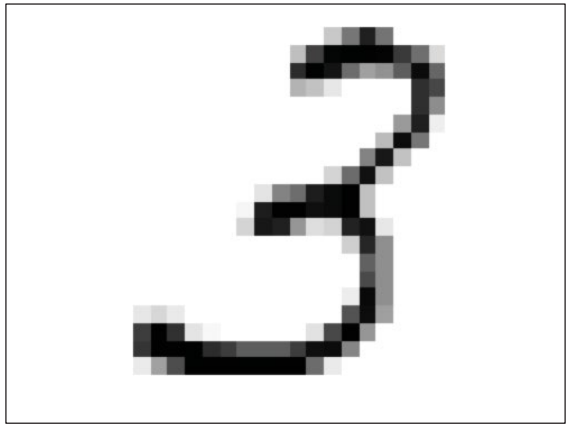
$$f(\mathcal{T}_{\mathbf{v}}x) = f(x) \quad \forall x, \mathbf{v}$$

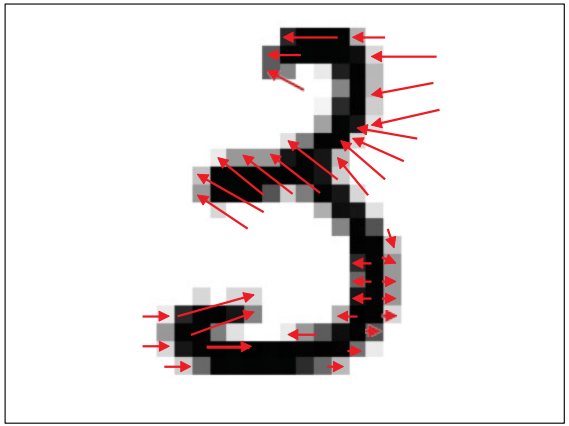
where

- image is modeled as a function $x \in L^2([0,1]^2)$
- $\mathcal{T}_{\mathbf{v}}x(\mathbf{u}) = x(\mathbf{u} - \mathbf{v})$ is a translation operator
- $\mathbf{v} \in [0,1]^2$ is a translation vector
- $f : L^2([0,1]^2) \to \{1, \dots, L\}$ is a classification functional

Bruna, Mallat 2012

Deformation invariance (image classification tasks)





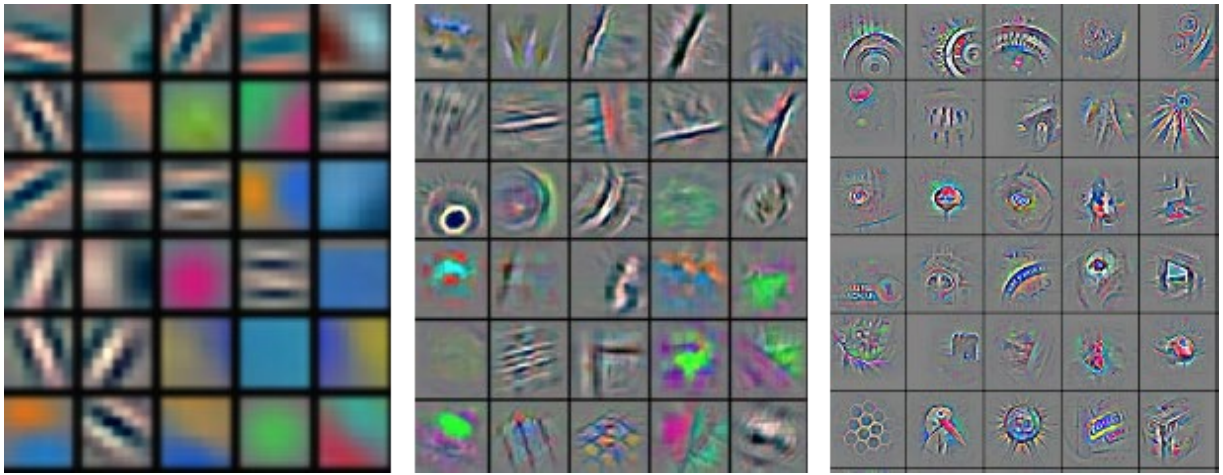
$$|f(\mathcal{L}_\tau x) - f(x)| \approx \|\nabla \tau\| \quad \forall f, \tau$$

where

- image is modeled as a function $x \in L^2([0,1]^2)$
- $\mathcal{L}_\tau x(\mathbf{u}) = x(\mathbf{u} - \tau(\mathbf{u}))$ is a warping operator
- $\tau : [0,1]^2 \to [0,1]^2$ is a smooth deformation field
- $f : L^2([0,1]^2) \to \{1, \ldots, L\}$ is a classification functional

Bruna, Mallat 2012

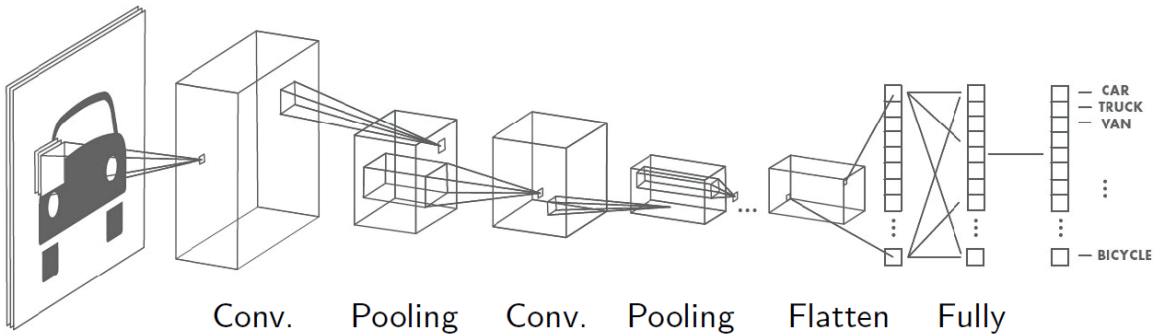
Hierarchy and Compositionality



Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

Convolutional Neural Networks (CNN)



Conv. layer $$x_{\ell'}^{(l+1)}(\mathbf{u}) = \xi\left(\sum_{\ell=1}^{d^{(l)}}(w_{\ell'\ell}^{(l+1)} \star x_{\ell}^{(l)})(\mathbf{u})\right)$$

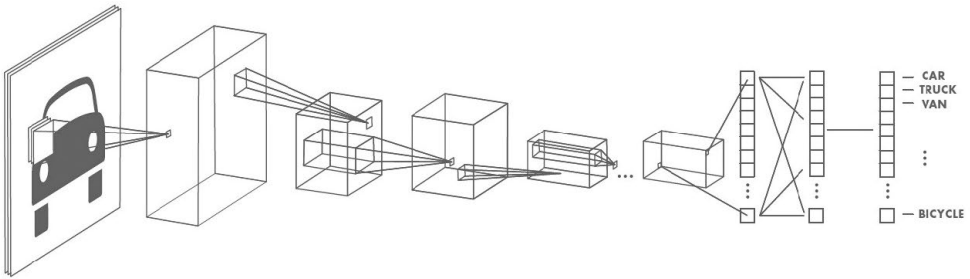
Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters $W^{(1)}, \ldots, W^{(L)}$

Pooling $$x_{\ell}^{(l+1)}(\mathbf{u}) = \|x_{\ell}^{(l)}(\mathbf{u}') : \mathbf{u}' \in \mathcal{N}(\mathbf{u})\|_p \quad p = 1, 2, \text{ or } \infty$$

LeCun et al. 1989 (Image: Debarko De)

Key properties of CNNs

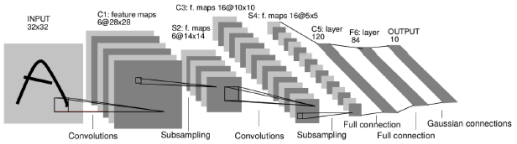


- ☺ Convolutional filters **(Translation invariance+Self-similarity)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)
- ☺ $\mathcal{O}(\log n)$ layers in classification tasks

LeCun et al. 1989

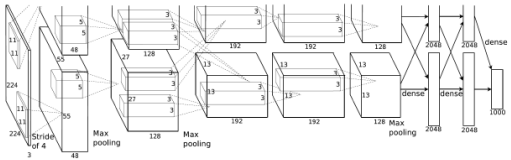
Convolutional Neural Networks (historical perspective)

1989



- 3 convolutional + 1 fully connected layer
- 1M parameters
- Trained on MNIST 70K
- CPU-based
- $\tanh$ non-linearity

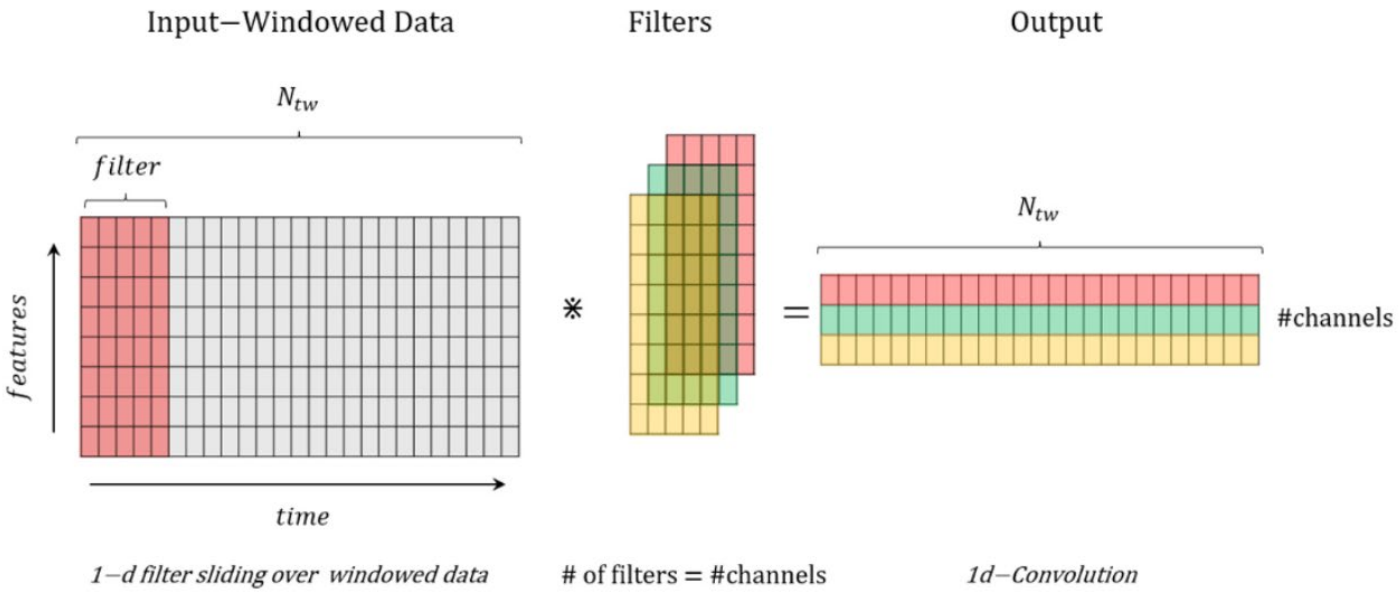
2012



- 5 convolutional + 3 fully connected layers
- 60M parameters
- Trained on ImageNet 1.5M
- GPU-based
- ReLU, Dropout

LeCun et al. 1998; Krizhevsky, Sutskever, Hinton 2012

1D CNN


Input—Windowed Data
Filters
Output
N_{tw}
filter
features
time
*
=
N_{tw}
#channels
1—d filter sliding over windowed data
# of filters = #channels
1d—Convolution

Hyperparameter search

Hyperparameter Tuning in Machine Learning

- **Definition**: Hyperparameter tuning involves adjusting the parameters of a machine learning model that are set prior to the start of the learning process. These parameters influence model training and can significantly affect performance.

Why It's Important

- Hyperparameters control the learning process and directly impact the effectiveness and efficiency of the model.
- Proper tuning can prevent overfitting, underfitting, and can improve model accuracy.

Formalizing hyperparameter tuning

- In a general sense, tuning involves these components:
  - a learning algorithm A, parameterized by hyperparameters λ
  - training and validation data $\mathbf{X}^{(tr)}$, $\mathbf{X}^{(vl)}$
  - a model $\mathcal{M} = \mathcal{A}(\mathbf{X}^{(tr)} \mid \lambda)$
  - loss function $\mathcal{L}$ to assess quality of $\mathcal{M}$, typically using $\mathbf{X}^{(vl)}$:

$$\mathcal{L}(\mathcal{M} \mid \mathbf{X}^{(te)})$$

- In optimization terms, we aim to find λ^* (assuming minimization):

$$\lambda^* = \arg\min_{\lambda} \mathcal{L}\left(\mathcal{A}(\mathbf{X}^{(tr)} \mid \lambda) \mid \mathbf{X}^{(vl)}\right) = \arg\min_{\lambda} \underbrace{\mathcal{F}(\lambda \mid \mathcal{A}, \mathbf{X}^{(tr)}, \mathbf{X}^{(vl)}, \mathcal{L})}_{\textit{objective function}}$$

Source: Claesen, de Moor 2015

Tuning in practice

- Most often done using a combination of grid and manual search:
  - grid search suffers from the curse of dimensionality
  - manual tuning leads to poor reproducibility

Abstraction of hyperparameter tuning

- We need to define the hyperparameters we want to optimize:
  - Architecture (#layers, #kernels, stride, kernel size)
  - Learning rate, optimizer (momentum)
  - Regularizations (weight decay rate, dropout probability)
  - Batchnorm / no batchnorm
  - Number of epochs: The number of times the learning algorithm will work through the entire training dataset.
  - Batch size: The number of training examples utilized in one iteration.
- Our diagnostic statistics:
  - Loss curves
  - Gradient norms
  - Accuracy / Visual output (generative models)
  - Performance on training vs validation set
  - Other abnormal behaviors

Source: Li, Johnson, Yeung, 2018

Example Hyperparameters SVM

$$\min_{\alpha,\xi,b} \frac{1}{2} \sum_{i \in SV} \sum_{j \in SV} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + C \sum_{i=1}^{n} \xi_i,$$

$$\text{subject to } y_i \Big(\sum_{j \in SV} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + b \Big) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i.$$

Tuning Techniques

- Grid Search: Exhaustive search over a specified parameter range.
- Random Search: Randomly samples the search space and evaluates sets from a specified probability distribution.
- Bayesian Optimization: Uses a probabilistic model to predict which hyperparameters might lead to better performances.
- Gradient-based Optimization: Adjusts hyperparameters using gradient descent to minimize a predefined loss function.
- Evolutionary Algorithms: Uses mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection.

Fast iteration

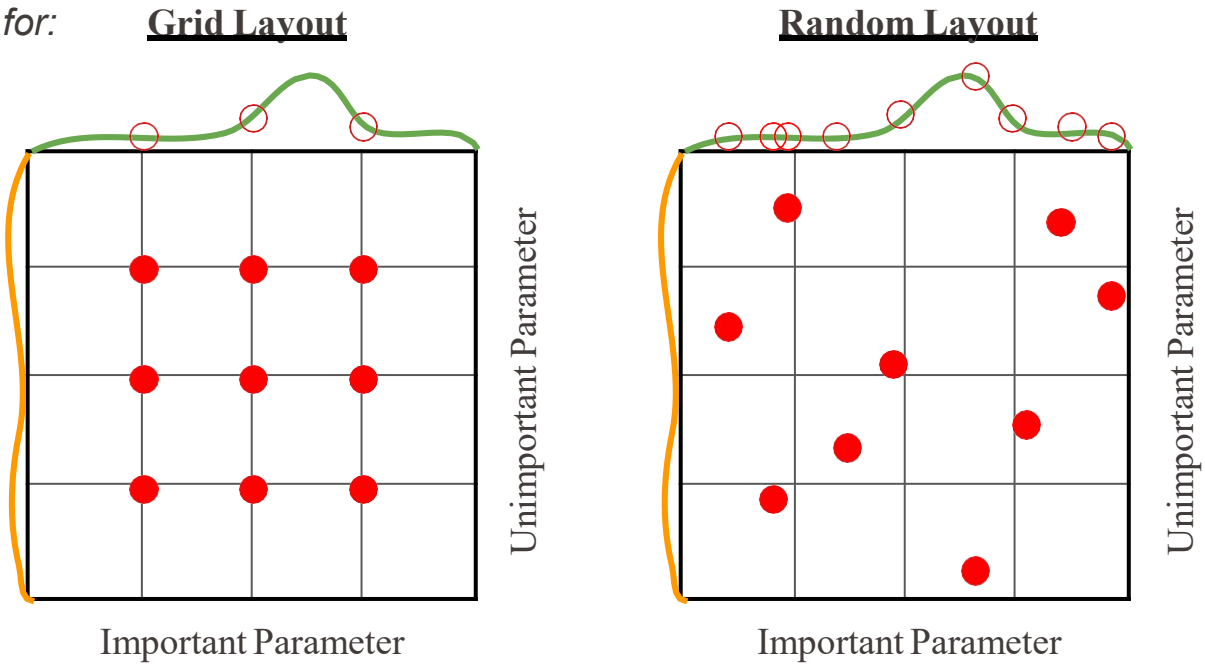
- When tuning parameters, only use a small portion of the dataset for fast iteration.
- Start from a larger learning rate for fast prototyping.

- Cross-validation strategy:
  - coarse → fine: cross-validation in stages
  - First stage: only a few epochs to get rough idea of what parameters work
  - Second stage: longer running time, finer search
  - … (repeat as necessary)

Source: Li, Johnson, Yeung, 2018

Random Search vs. Grid Search

Random Search for:



Hyper-Parameter Optimization
Bergstra and Bengio, 2012

Can we use Machine Learning instead?

- To predict regions of the hyperparameter space that might give better results.

- To predict how well a new combination of hyperparameters will do and also model the uncertainty of that prediction

Source: Snoek et. al., 2012

Can We Do Better? Bayesian Optimization

- Build a probabilistic model for the objective.
- Include hierarchical structure about units, etc.
- Compute the posterior predictive distribution.
- Integrate out all the possible true functions.
- Gaussian process regression is often used.
- Optimize a cheap proxy function instead.
- The model is much cheaper than the true objective.
- The main insight:
  - Make the proxy function exploit uncertainty to balance exploration against exploitation.

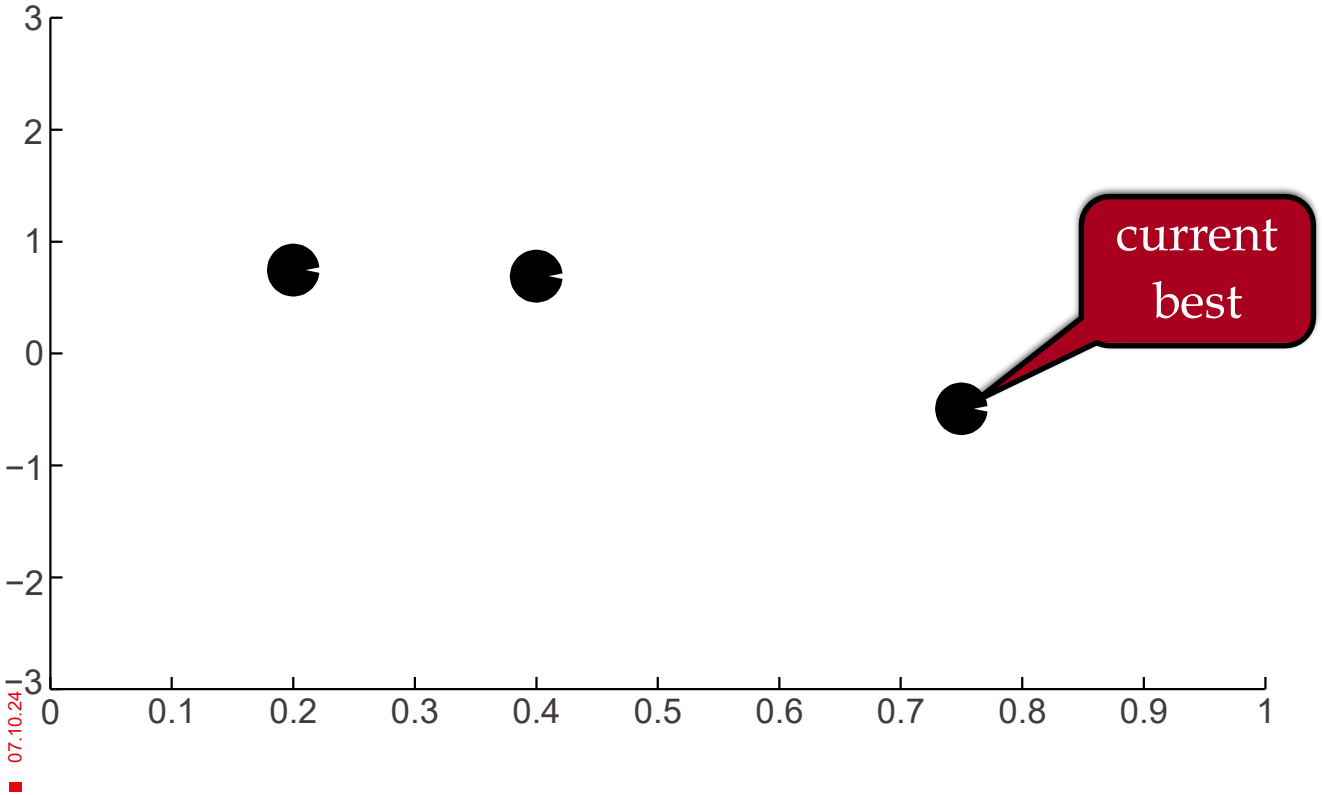
Source: R. Adams, 2017

Bayesian Optimization

- Frame Hyperparameter Search as an Optimization Problem
- Model the estimation of the function from hyperparameters to the error metric as a regression problem
- Use Gaussian Process Prior: “Similar inputs have similar outputs” to build a statistical model of the function. Prior is weak but general and effective.
- Use statistics to tell us:
  - Location of expected minimum of the function
  - Expected Improvement of trying other parameters

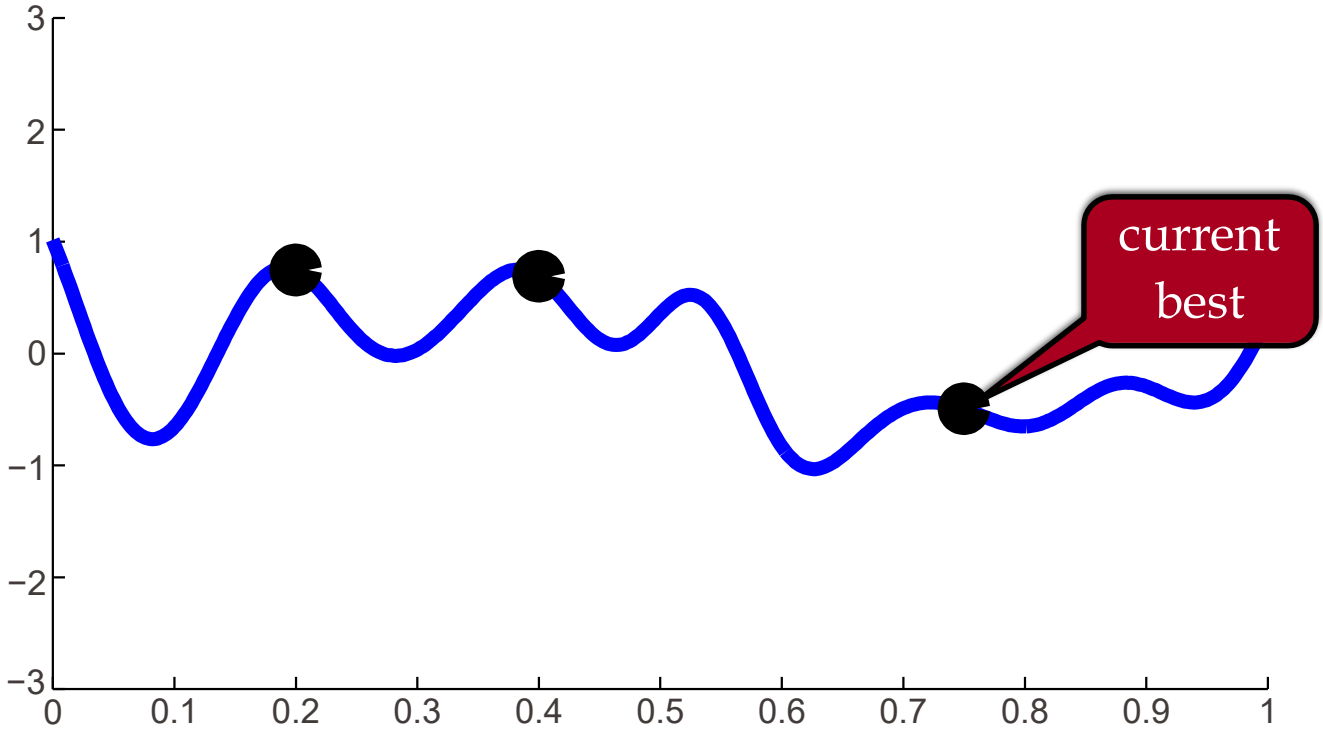
Source: Snoek et. al., 2012

The Bayesian Optimization Idea



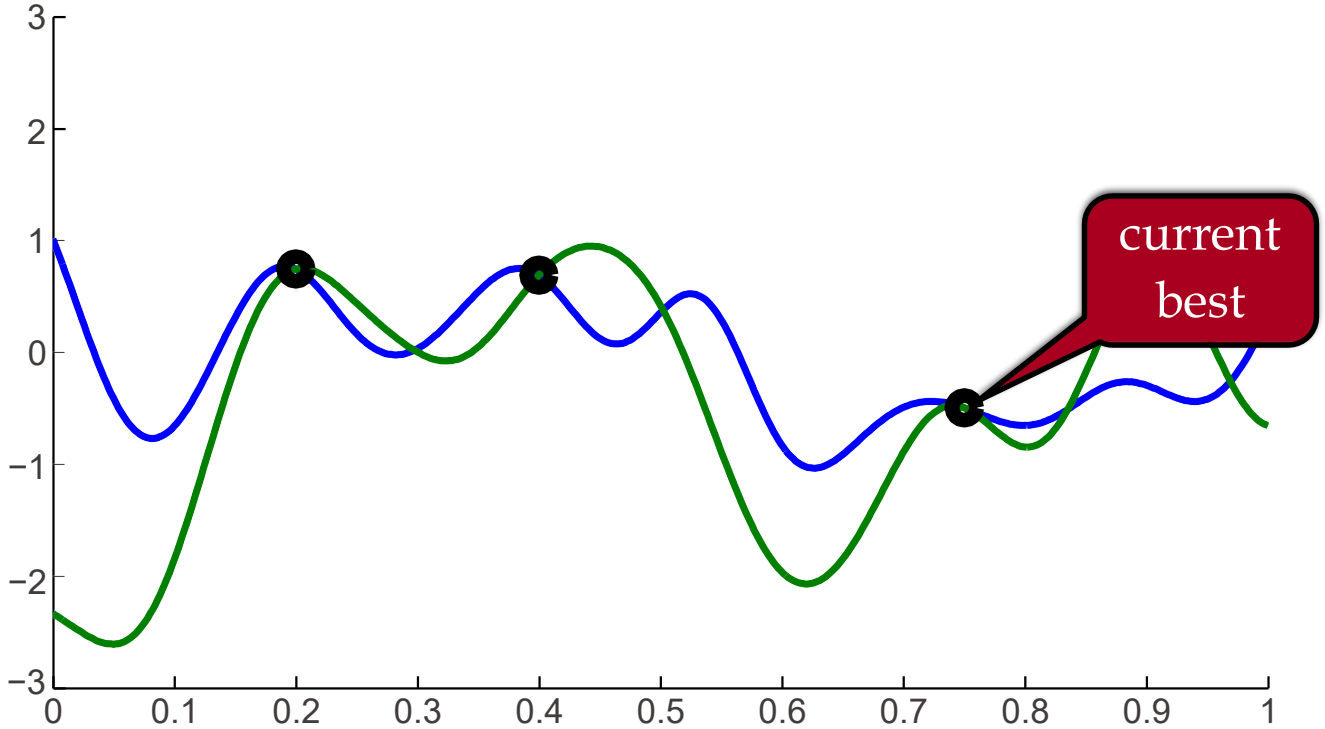
Source: R. Adams, 2017

The Bayesian Optimization Idea



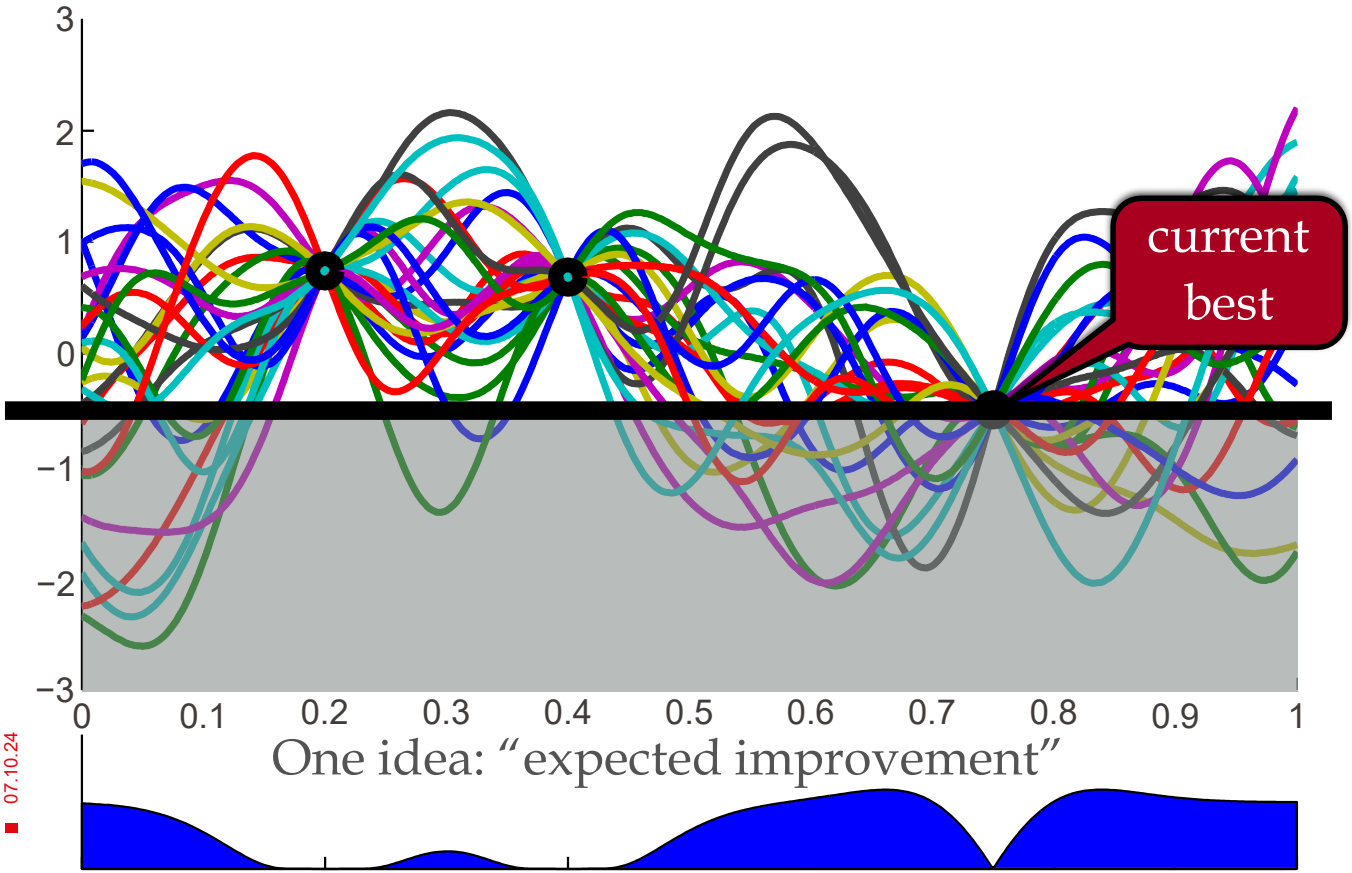
Source: R. Adams, 2017

The Bayesian Optimization Idea



Source: R. Adams, 2017

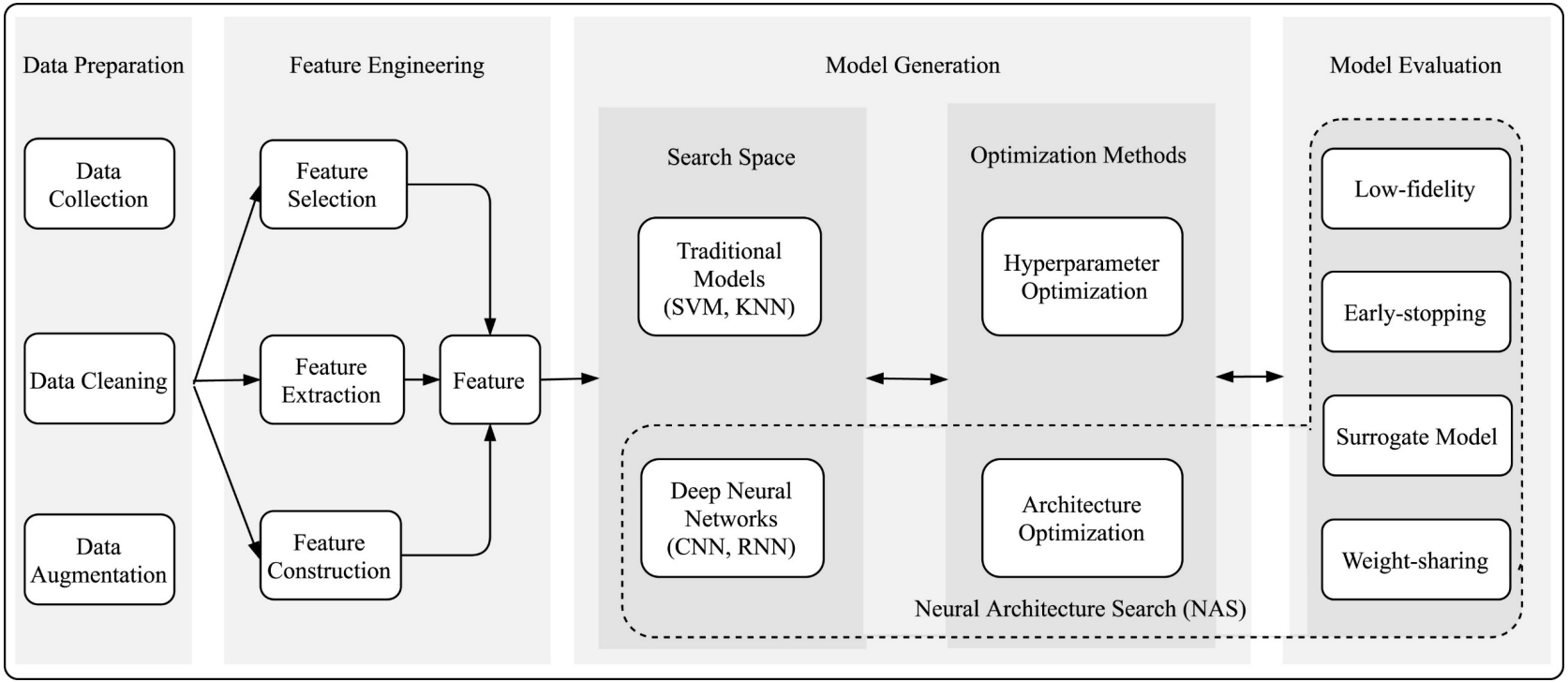
The Bayesian Optimization Idea



Where should we evaluate next in order to improve the most?

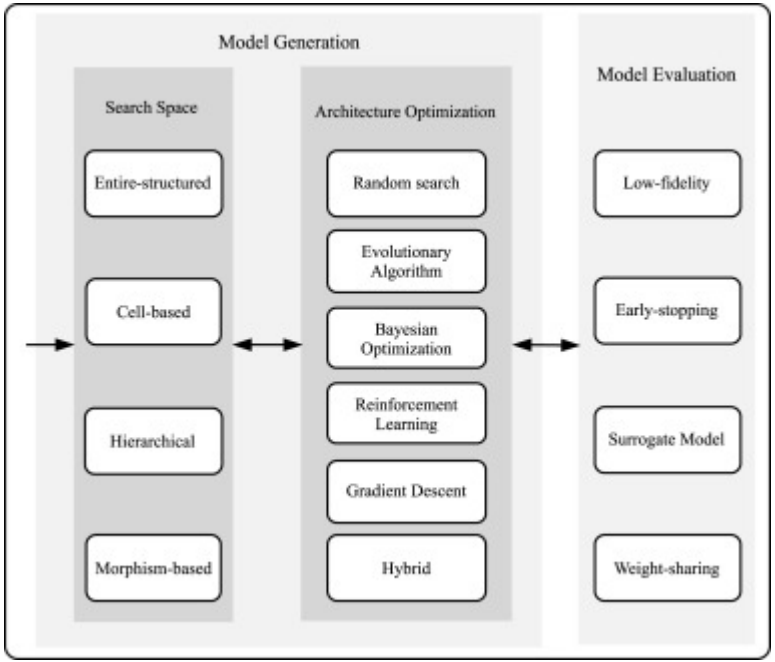
Source: R. Adams, 2017

From hyperparameter search to Neural Architecture Search



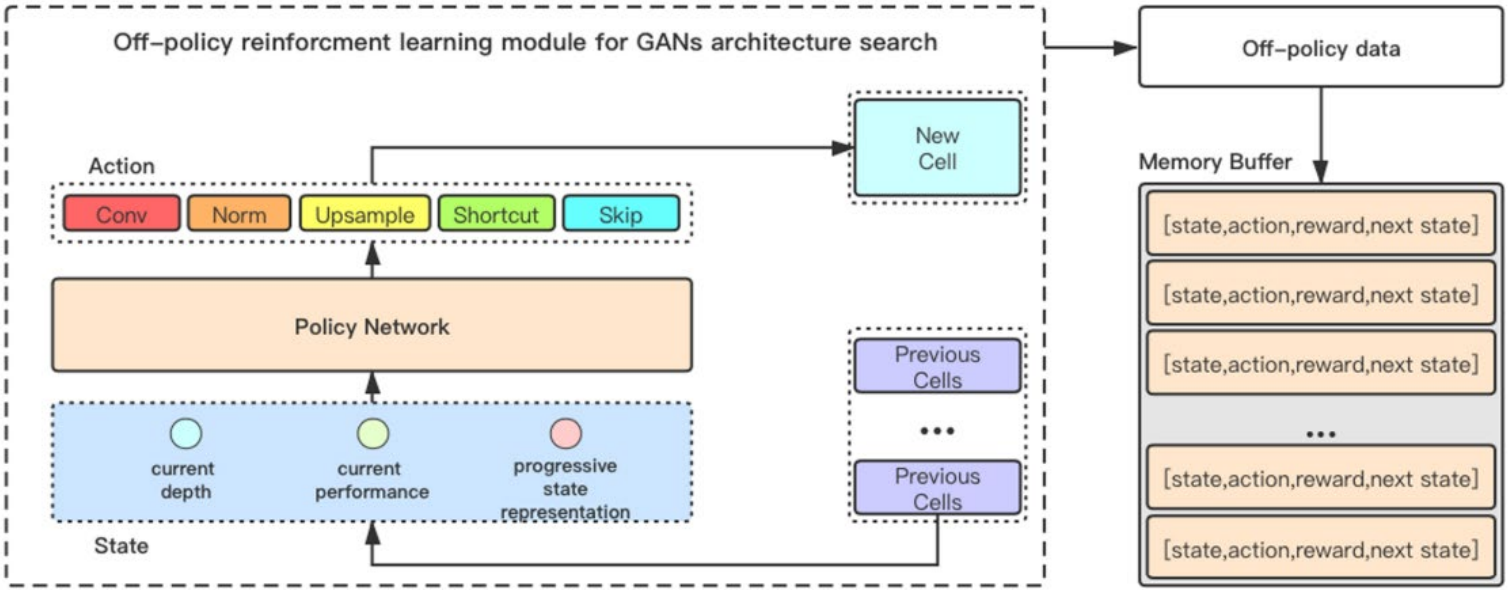
He, Xin, Kaiyong Zhao, and Xiaowen Chu. "AutoML: A Survey of the State-of-the-Art." *Knowledge-Based Systems* 212 (2021): 106622.

An overview of neural architecture search pipeline



He, Xin, Kaiyong Zhao, and Xiaowen Chu. "AutoML: A Survey of the State-of-the-Art." *Knowledge-Based Systems* 212 (2021): 106622.

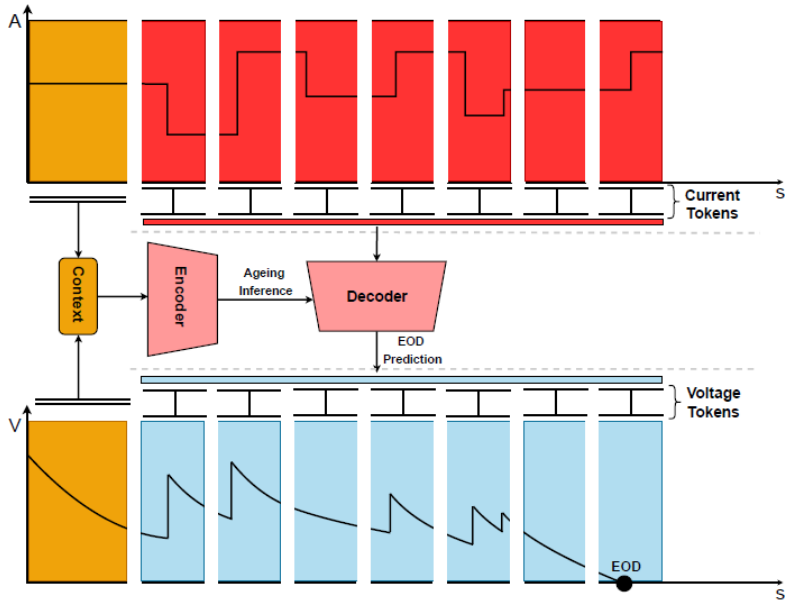
Reinforcement learning for GAN architecture search



Tian, Y., Wang, Q., Huang, Z., Li, W., Dai, D., Yang, M., ... & Fink, O. (2020, August). Off-policy reinforcement learning for efficient and effective gan architecture search. In *European Conference on Computer Vision* (pp. 175-192). Springer, Cham.

Tools and libraries

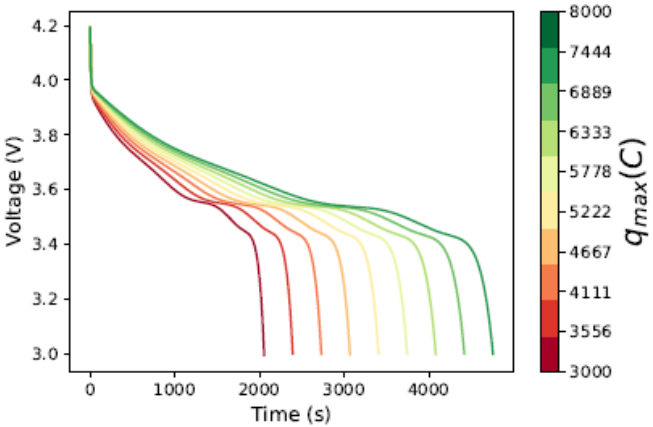
- **Scikit-learn**: Offers grid and random search capabilities.
- **Hyperopt**: Advanced optimization, including Bayesian and random.
- **Optuna**: Framework for automatic hyperparameter optimization, supporting sophisticated algorithms.

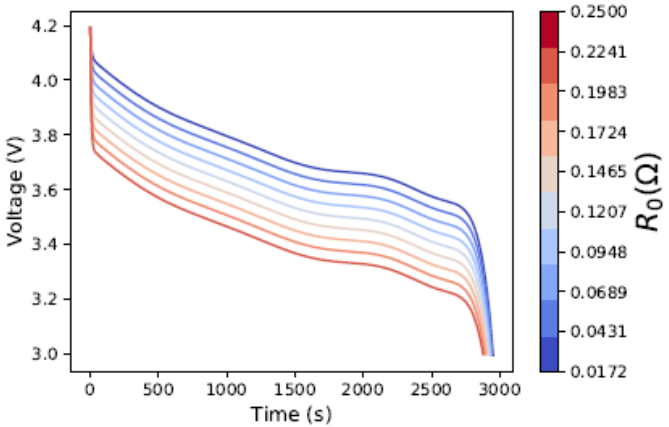


Ageing-aware Battery Discharge Prediction

Degradation of Li-Ion batteries → importance of precise planning

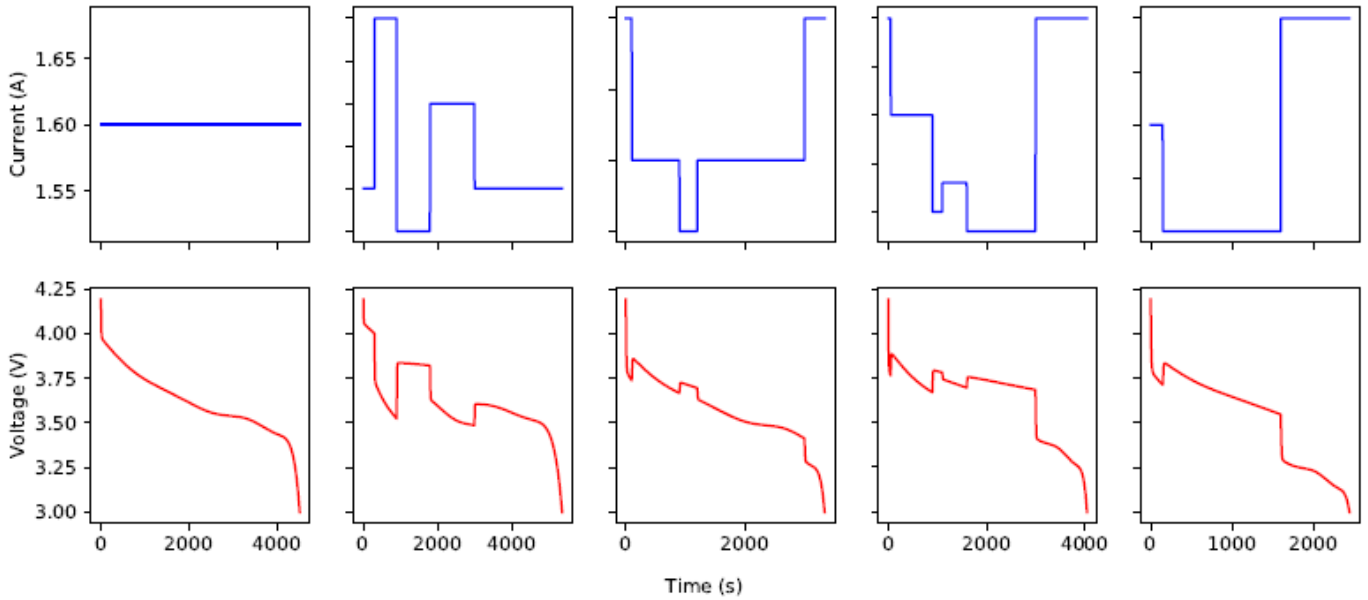
Effect of varying the degradation parameters on the voltage discharge curve of a Li-ion battery





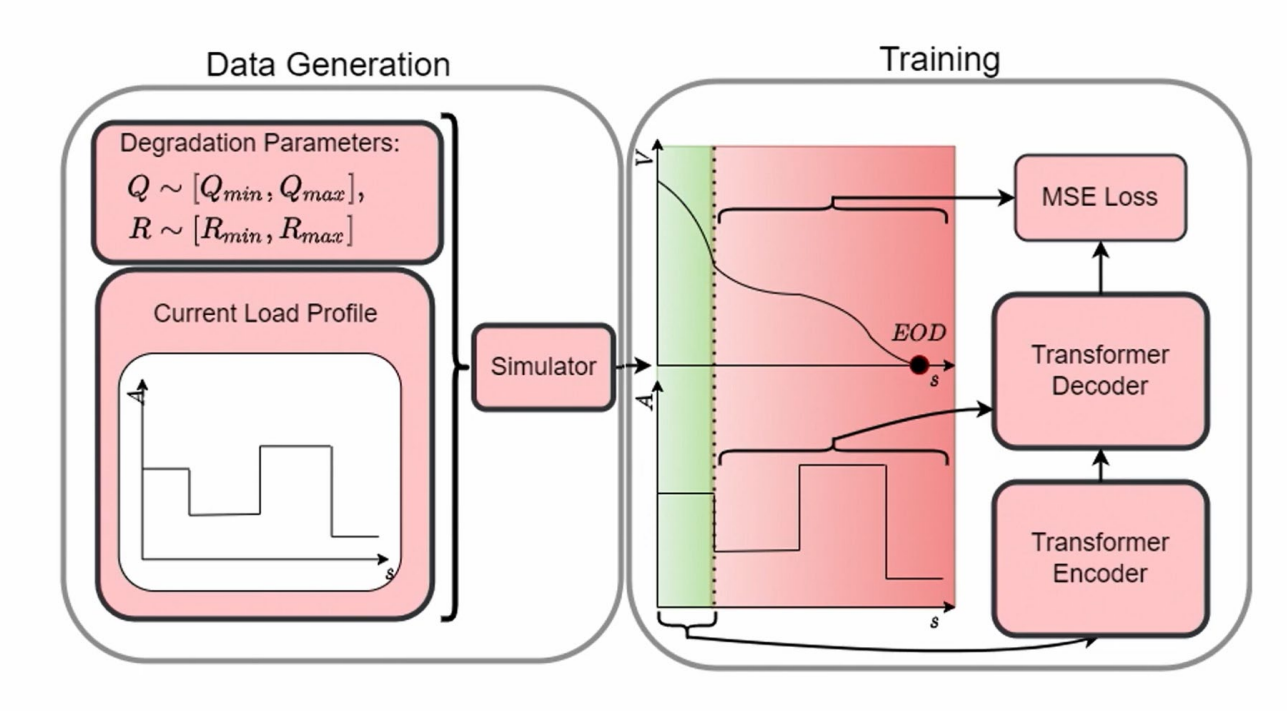
Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Discharge behaviors with respect to the different load profiles



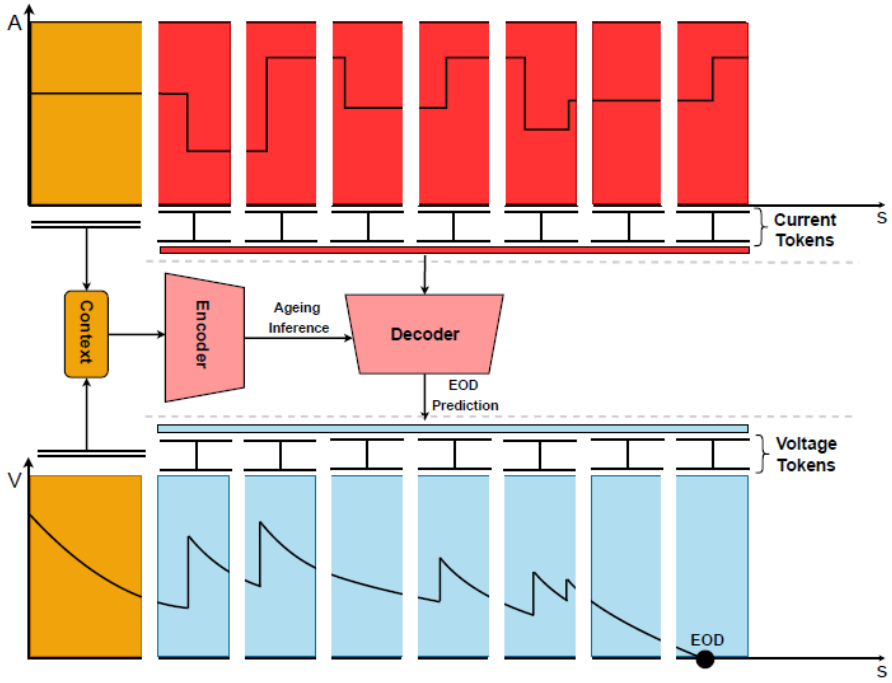
Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Data generation + training



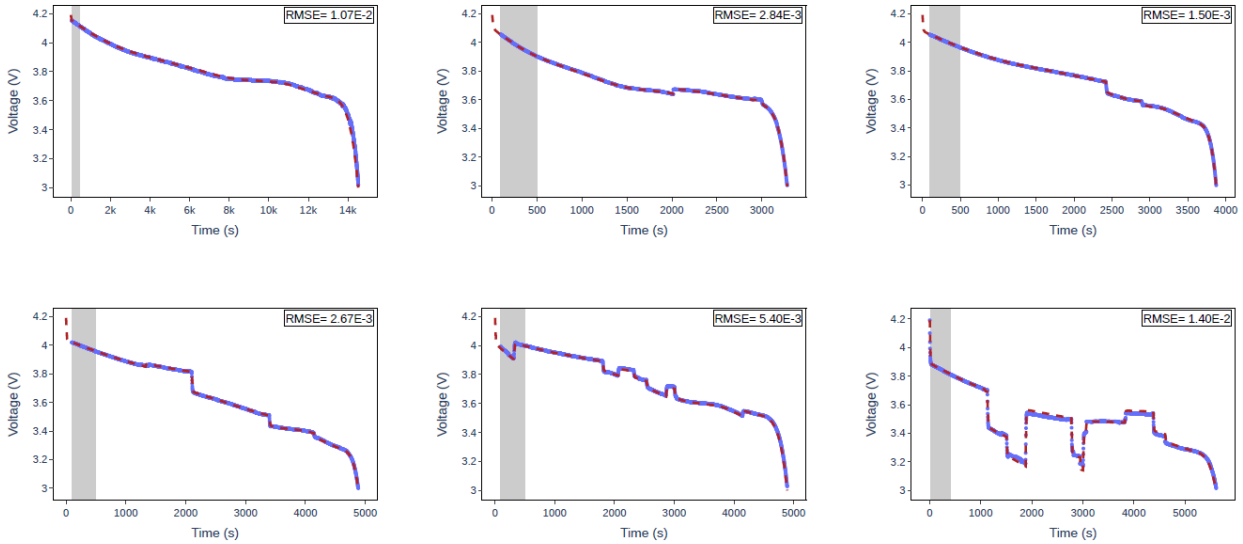
Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Transformer-based long-term battery discharge prediction → Dynaformer



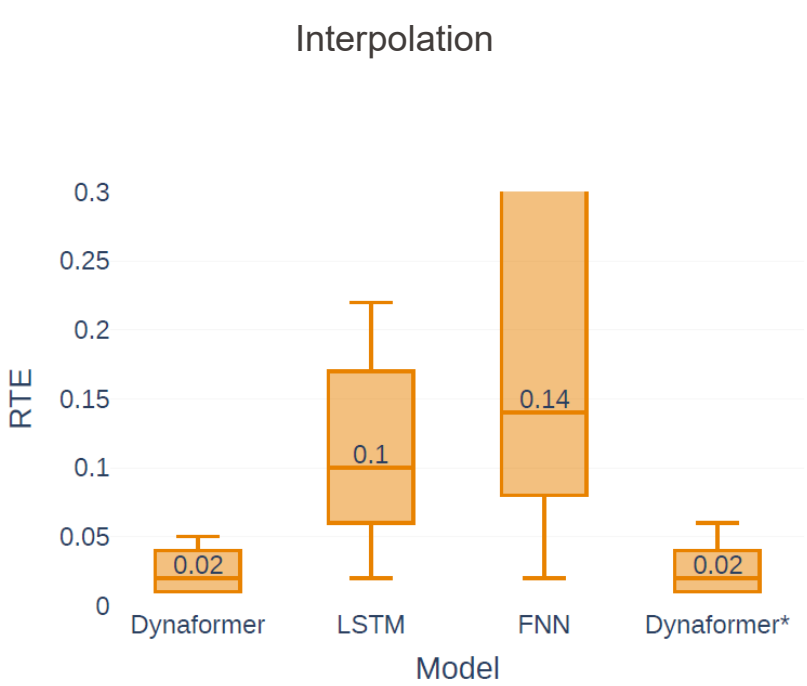
Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

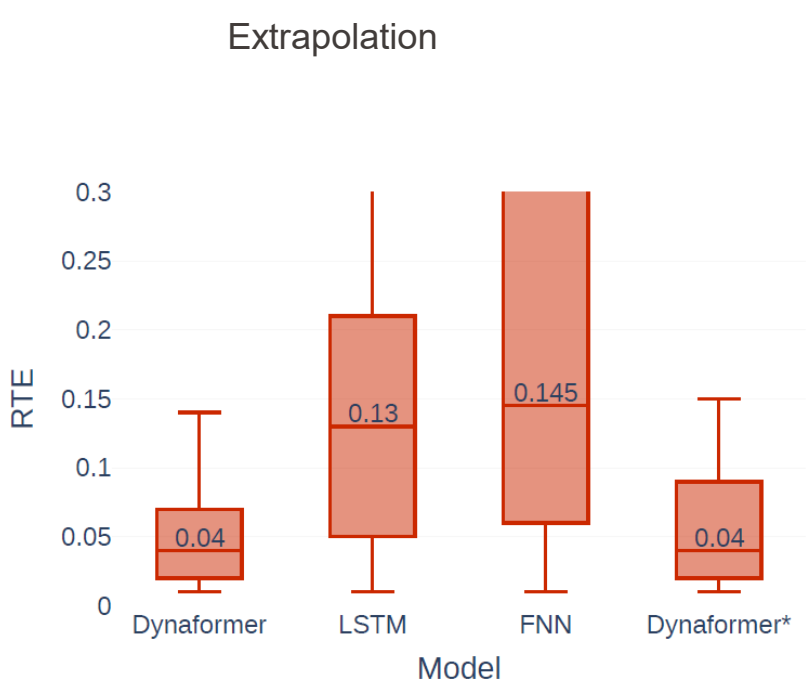
Selected results



Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Performance without fine-tuning



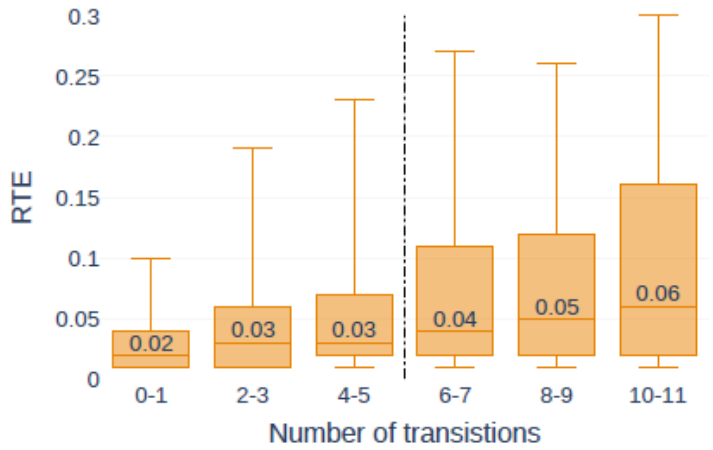


Dynaformer* → trained with variable current profiles

*RTE = relative temporal error

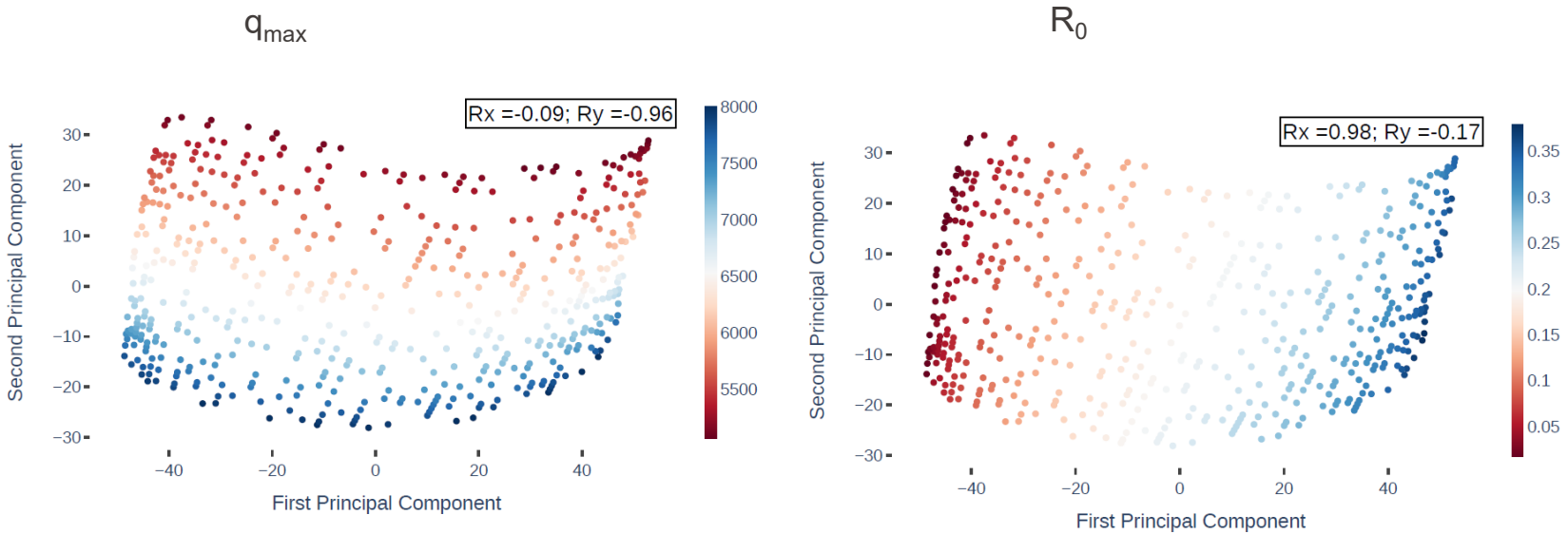
Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Performance dependent on the complexity of the the load profiles



Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy

Implicit learning of the degradation parameters in the latent space



Biggio, L., Bendinelli, T., Kulkarni, C., & Fink, O. (2023). Ageing-aware battery discharge prediction with deep learning, Applied Energy