SimProcess: High Fidelity Simulation of Noisy ICS Physical Processes

Denis Donadel[1], Gabriele Crestanello[2], Giulio Morandini[2], Daniele Antonioli[3], Mauro Conti[2], Massimo Merro[1]

[1] University of Verona, Italy, [2] University of Padua, Italy, [3] EURECOM, France

denis.donadel@univr.it donadelden.github.io

CPSS 2025

Index

Introduction

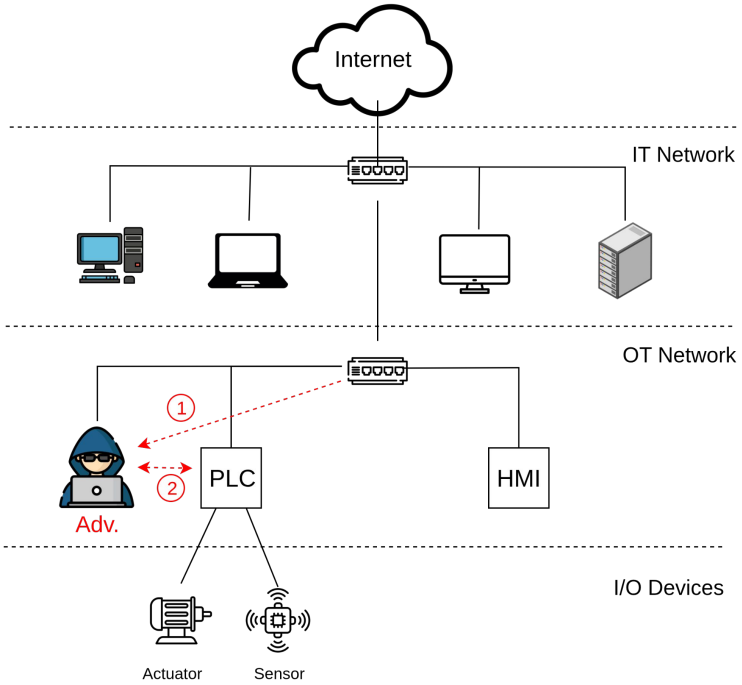
- Industrial Control System (ICS) security research
  - Simulators and emulators are essential
  - Can be used as decoy systems
- Honeypots
  - Trick attackers and make them loose time
  - Learn new attacker behaviors

Motivation

- Effective honeypot is based on many factors, such as:
  - Exposed in “clean” and realistic IP addresses
  - Correctly emulate network devices and communications
  - **Exhibit a realistic physical process**
- Fidelity of the physical process can be compromised by the noise
  - Various noises influence an ICS physical process (sensor noise, environment, …)
- How to introduce a realistic noise in our simulation?
- How to understand if and how much the noise is realistic?

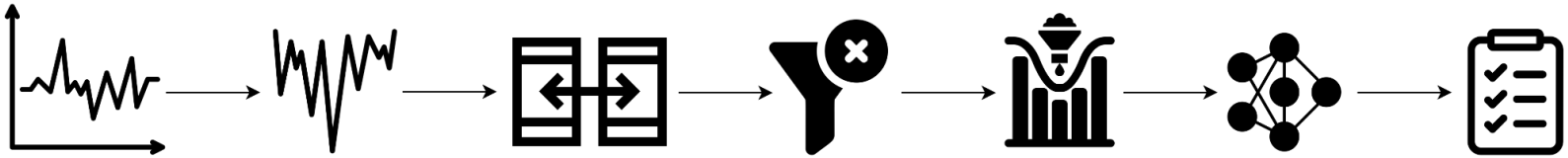
System and Attacker Model

- System model
  - A developer build a honeypot to (partially) mimic a system he controls
- Threat model
  - Remote attacker which collects data from sensors
    1. Passively, sniffing the network
    2. Actively, by quering components

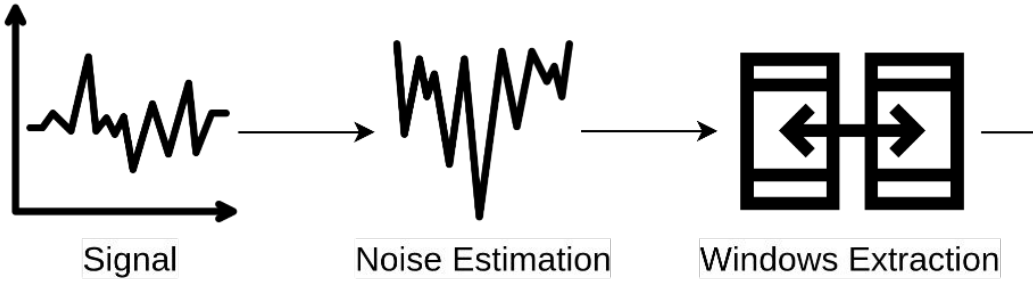


SimProcess

Framework that allows the identification of the simulations with the best fidelity to replicate a real-world physical process

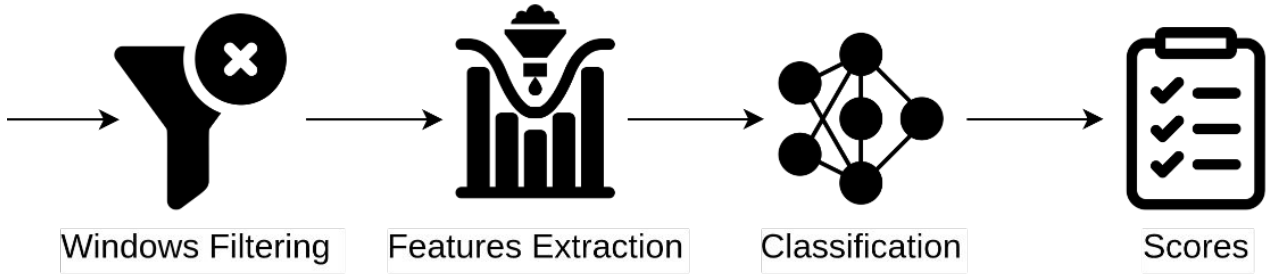


Pipeline



1. Collect **signals** from the real and simulated systems $\hat{x}(t) = x(t) + n(t)$
   - Composed of the real signal and a noise
   - Need to be collected for real system and evey different noise you want to test
2. Estimate the **noise** through a filter $\tilde{n}(t) = \hat{x}(t) - f(\hat{x}(t))$
3. Sliding **window** approach to divide the signal into smaller batches

Pipeline



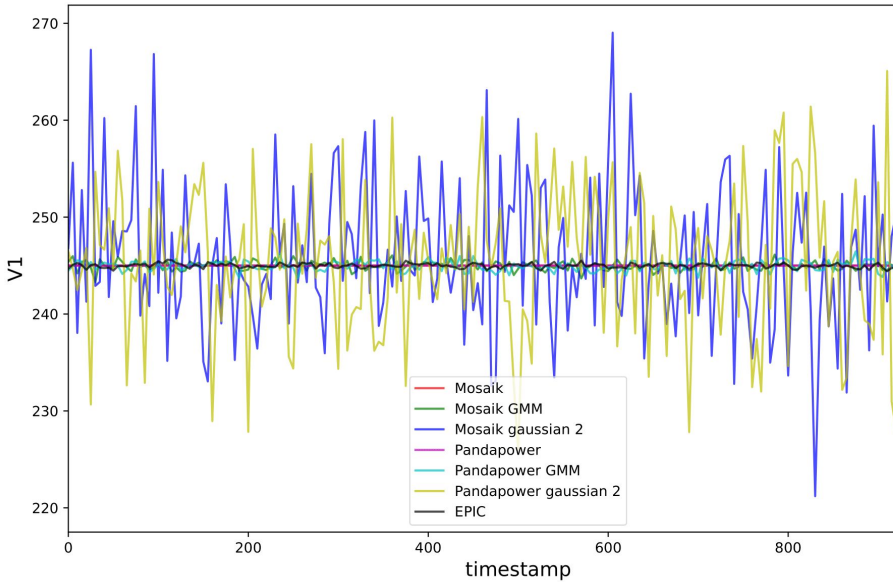
4. **Window filtering** by removing samples with high variation
   - May be due to change to *real* changes in the physical process, while we want to profile only the noise
5. **Feature extraction** using tsfresh
6. **Classification** using ML models
   - Binary classificaton problem (real vs simulated)
7. **Scoring** the results to identify the one closer to the real system
   - Looking for the “probability of a *simulated* process to be misclassified as *real*”

Case study: EPIC Testbed

- Electric Power and Intelligent Control
- Developed by iTrust in Singapore
- Four stages:
  - Generation
  - Transmission
  - Micro-grid
  - Smart Home
- We used its available dataset as ground-truth

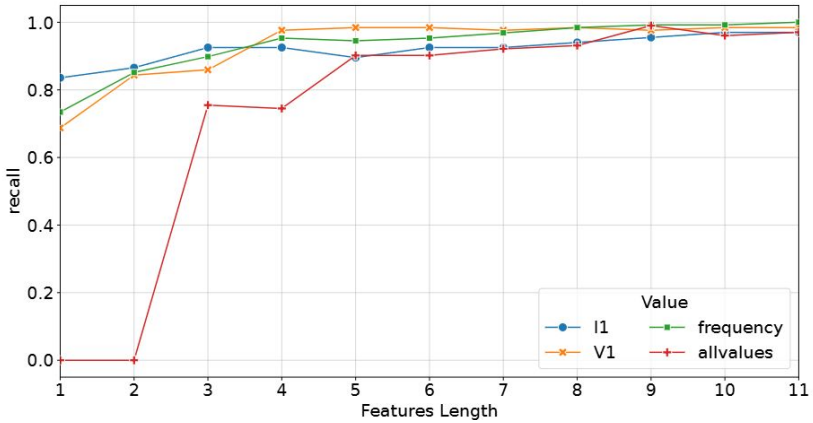
Partial simulations with different noises

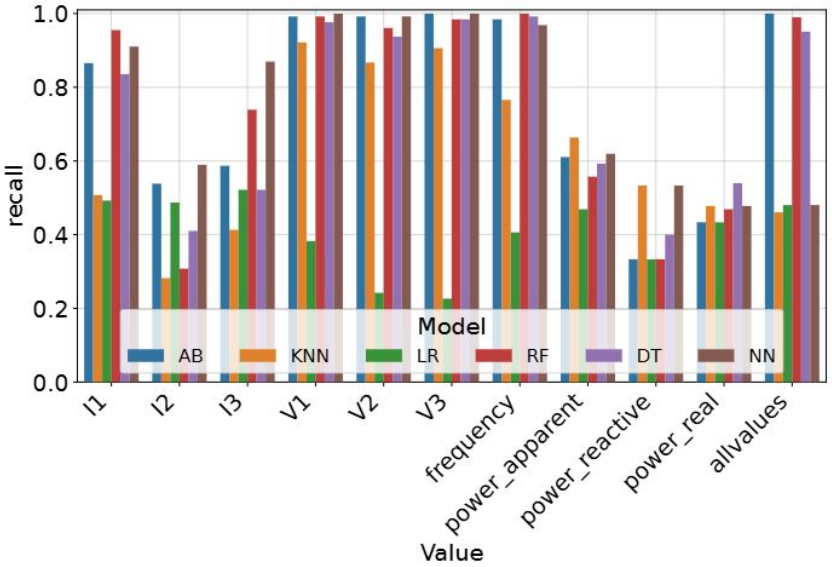
Source	Real	Noise	Parameters
EPIC [1]	✔	✔	-
Plain Simulation	✘	✘	-
Sim. + uniform	✘	✔	σ_u=0.01
Sim. + gaussian1	✘	✔	σ_g=0.01
Sim. + gaussian2	✘	✔	σ_g=0.05
Sim. + poisson	✘	✔	σ_p=0.01, λ_p=1.5
Sim. + laplace	✘	✔	σ_l=0.01
Sim. + pink	✘	✔	σ_p=0.01
Sim. + GMM	✘	✔	σ_g=0.02
Sim. + gaussian + uniform	✘	✔	σ_g=0.01, σ_u=0.01
Sim. + laplace + uniform	✘	✔	σ_l=0.01, σ_u=0.01
Sim. + laplace + poisson	✘	✔	σ_l=0.01, σ_u=0.01, λ_p=1.5
Sim. + VRAE	✘	✔	$input\_weight$ = 0.99



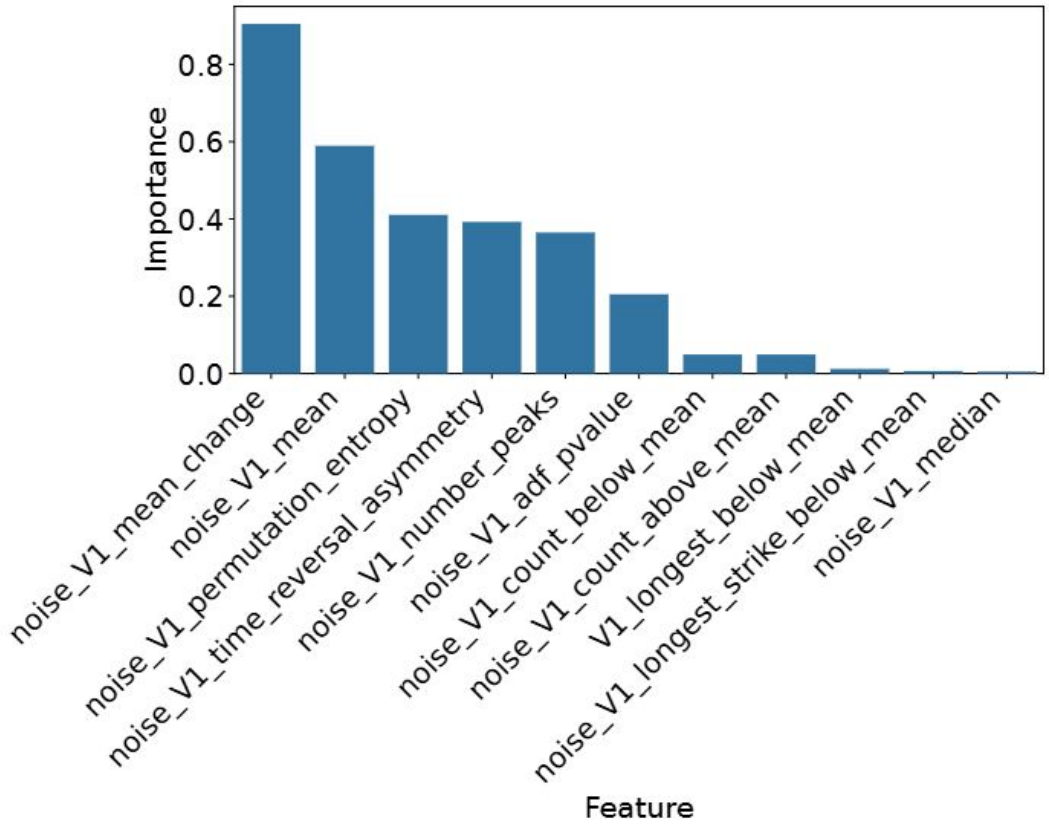
Fine Tuning

- Dataset balancing
  - Data augmentation with SMOTE
- Window length testing
  - 7 different window sizes
- Feature reduction
  - Trained adding one feature at the time
  - Tested on V1, I1, frequency, allvalues
- Model selection
  - **DT**, LR, kNN, **RF**, **AB**, NN



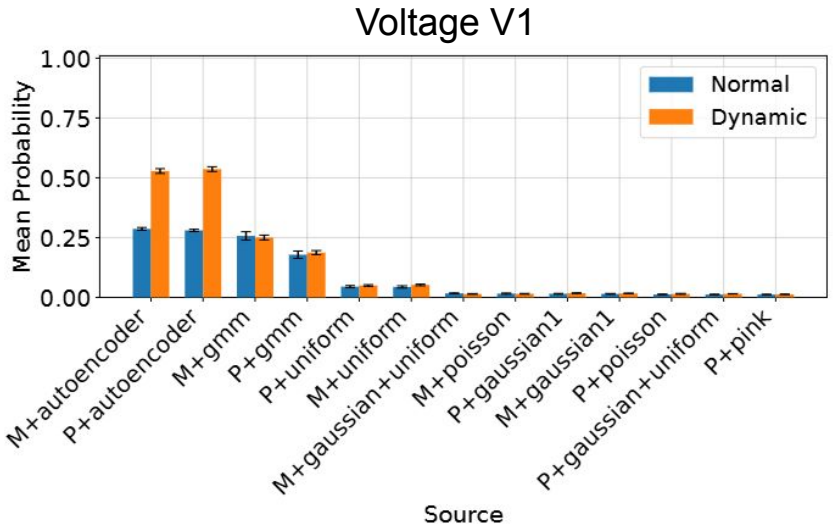


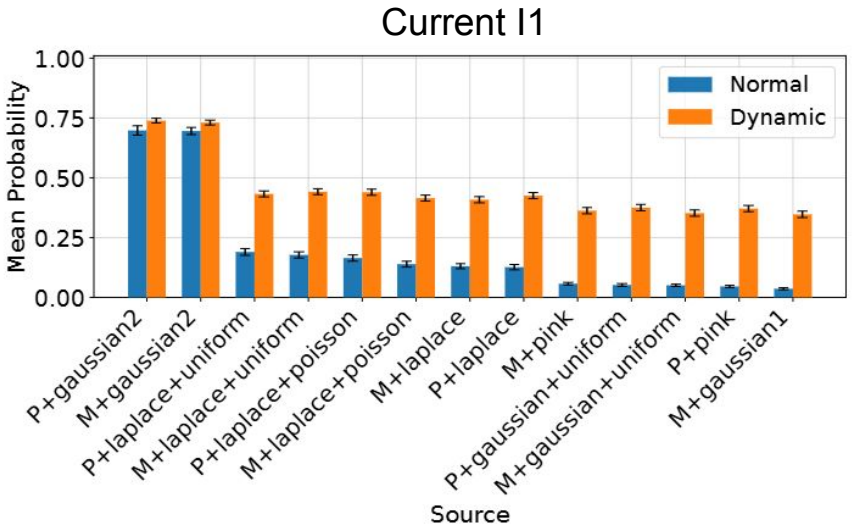
Most Important Features



Results

M: Mosaik; P: Panapower





Normal: normal behavior without significant changes

Dynamic: something happened in the process (e.g., peak of energy requested due to a switch triggered)

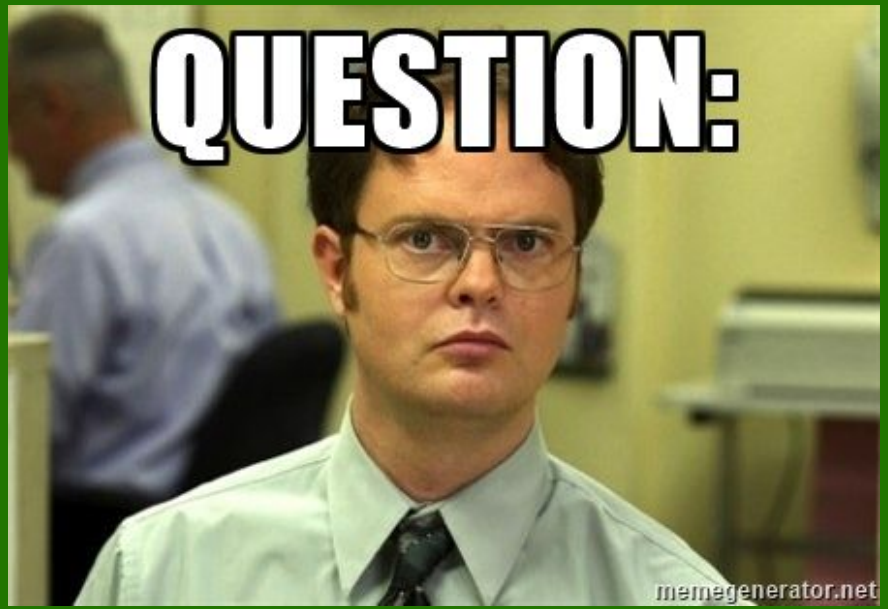
Conclusions

- Tackling the problem of measuring the fidelity of a physical process simulation
- Different measures exhibit different noises
  - GMM and Gaussian are the most similar to real scenarios
  - The usage of autoencoder is, as well, promising
- SimProcess is general and applicable in every domain
  - Future works should apply it in other fields, such as water systems

That’s all folks!

Denis Donadel
denis.donadel@univr.it

Project Github Page:
https://github.com/donadelden/SimProcess


QUESTION:
memegenerator.net

Other results?

- Prob:
  - probability of a simulated process to be misclassified as real
- Delta:
  - difference with the prob. In the dynamic system
  - Low deltas -> applicable also in dynamic systems
- Different measures, different noises

Value	Best Noise	Prob.	Delta
voltage V1	GMM	0.267	0.064
voltage V2	Autoencoder	0.397	0.078
voltage V3	Autoencoder	0.356	0.076
current I1	Gaussian2	0.688	0.052
current I2	Gaussian2	0.365	0.074
current I3	Gaussian2	0.525	0.065
frequency	GMM	0.331	0.233
power_apparent	Gaussian2	0.605	0.039
power_reactive	Laplace+Uniform	0.133	0.045
power_real	Gaussian2	0.707	0.073
all values	GMM	0.418	0.119

Why recall is important?

- True Positive Rate: Recall = TP/(TP + FN)
  - True Positive (TP) counts real samples correctly identified as real
  - False Negative (FN) identifies real samples classified as simulated
- We want to *maximize* the chances that our system classifies real samples correctly to then employ the classification probability as a *distance metric* between real and simulated samples
  - If a simulated sample sample is classified as real, it is a good simulation!

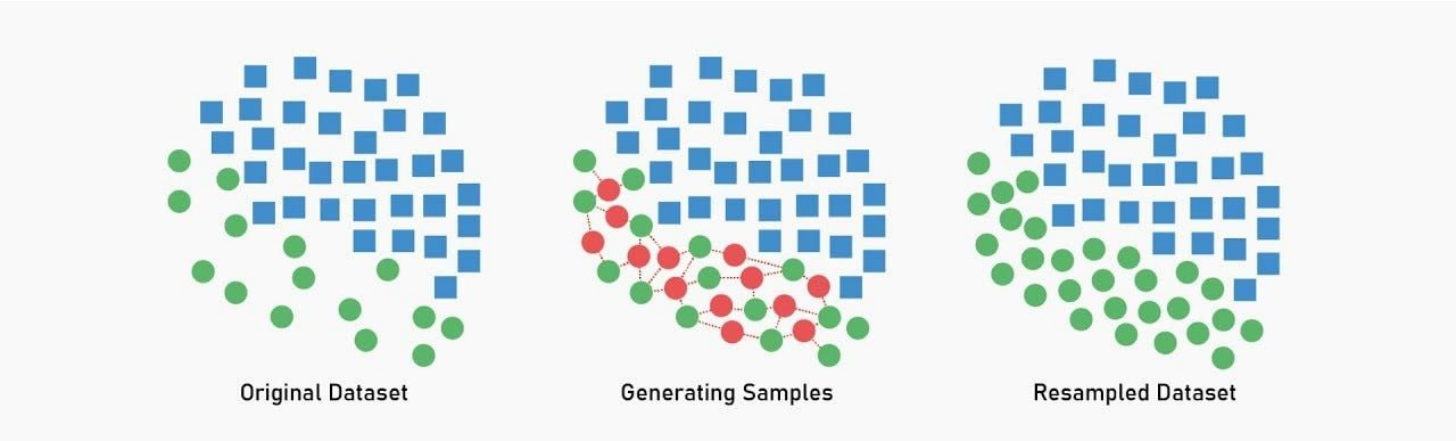
Autoencoder to generate noise?

- Use an Autoencoder to learn the noise pattern and generate it given a clean (i.e., without noise) input
- We used a Variational Recurrent Auto Encoder (VRAE)
  - Handle causality that allows the generating samples temporarily dependent on previous elements in the time series
- Trained on the noise pattern
- We used the architecture by [1]

[1] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. Advances in neural information processing systems 28 (2015).

What is SMOTE for data augmentation?

- SMOTE - Synthetic Minority Over-sampling Technique [2]
- Use KNN and interpolation to generate new realistic samples

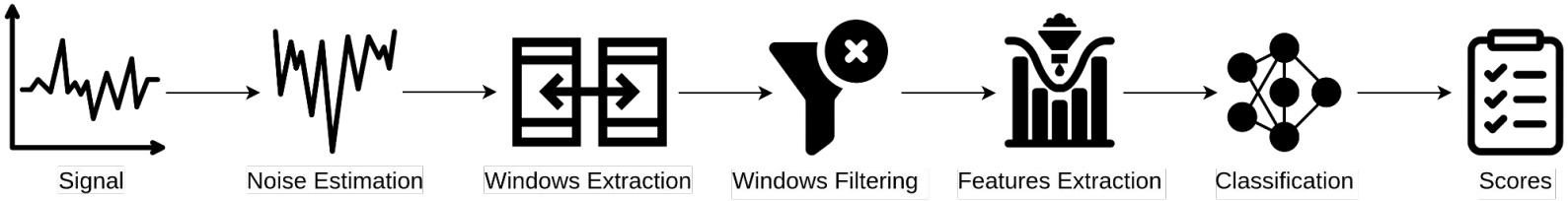


[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research 16 (2002), 321–357.

Which filter to get the noise?

- Different possibilities based on the system dynamic
- We employed a Kalman filter
- Similar results can be obtained with simpler filters (e.g., moving median)
- Coefficient for window pruning is also important based on the filter
  - High ε alleviate filter effects and reduce amount of deleted data
  - Low ε removed many samples containing even small variations

Pipeline


Signal
Noise Estimation
Windows Extraction
Windows Filtering
Features Extraction
Classification
Scores

Pipeline

1. Collect **signals** from the real and simulated systems $\hat{x}(t) = x(t) + n(t)$
   - Composed of the real signal and a noise
2. Estimate the **noise** through a filter $\tilde{n}(t) = \hat{x}(t) - f(\hat{x}(t)$
3. Sliding **window** approach to divide the signal into smaller samples
4. **Window filtering** by removing samples with high variation
   - May be due to change to *real* changes in the physical process, while we want to profile only the noise
5. **Feature extraction** using tsfresh
6. **Classification** using ML models
7. **Scoring** the results to identify the one closer to the real system
   - Looking for the "probability of a *simulated* process to be misclassified as *real*"