UNIVERSITÄT KLAGENFURT

Unlocking the Full Potential of Neural NILM: On Automation, Hyperparameters & Modular Pipelines

Motivations

Best practices in ML development

NILM toolkits

Deep-NILMtk

Hands-on session

Limitations and outlook

Hands-On Tutorial
(NILM Workshop 2022)

Hafsa Bousbiat
Hafsa.Bousbiat@gmail.com

DECIDE Doctoral College
Digital Age Research Center (D!ARC)
University of Klagenfurt, Austria

Co-authors:

- Anthony Faustine, a data scientist in Ireland.
- Christoph Klemenjak, PhD student in Smart Grid Research Group, University of Klagenfurt.
- Lucas Pereira, Research fellow at Técnico Lisboa, data science lead at prsma.com prsma.com Univ.-Prof. Dr.techn.
- Prof. Wilfried Elmenreich, head of Smart Grid Research Group, University of Klagenfurt

11.11.2022

Motivations

Motivations

Best practices in ML development

NILM toolkits

Deep-NILMtk

Hands-on session

Limitations and outlook

*“**Despite all progress** made concerning disaggregation techniques, **performance evaluation and comparability remains an open research question.** The **lack of standardisation** and **consensus on evaluation procedures** makes **reproducibility and comparability extremely difficult**.”[1]*

[1] Klemenjak, C., Makonin, S., & Elmenreich, W. (2020, February). Towards comparability in non-intrusive load monitoring: On data and performance evaluation. In 2020 IEEE power & energy society innovative smart grid technologies conference (ISGT) (pp. 1-5). IEEE.

Comparability and reproducibility as a RG in NILM

1. An Exploding number of models recently proposed
2. Details of the methodology are missing in some publications.
3. No sharing of the code in some cases, or code that is not compatible.
4. A gap between code and results
   - Can not get the code to give the same results
   - No pre-trained models
   - No training scripts.

Reproducing results and comparisons becomes hard

Motivations

Motivations

Best practices in ML development

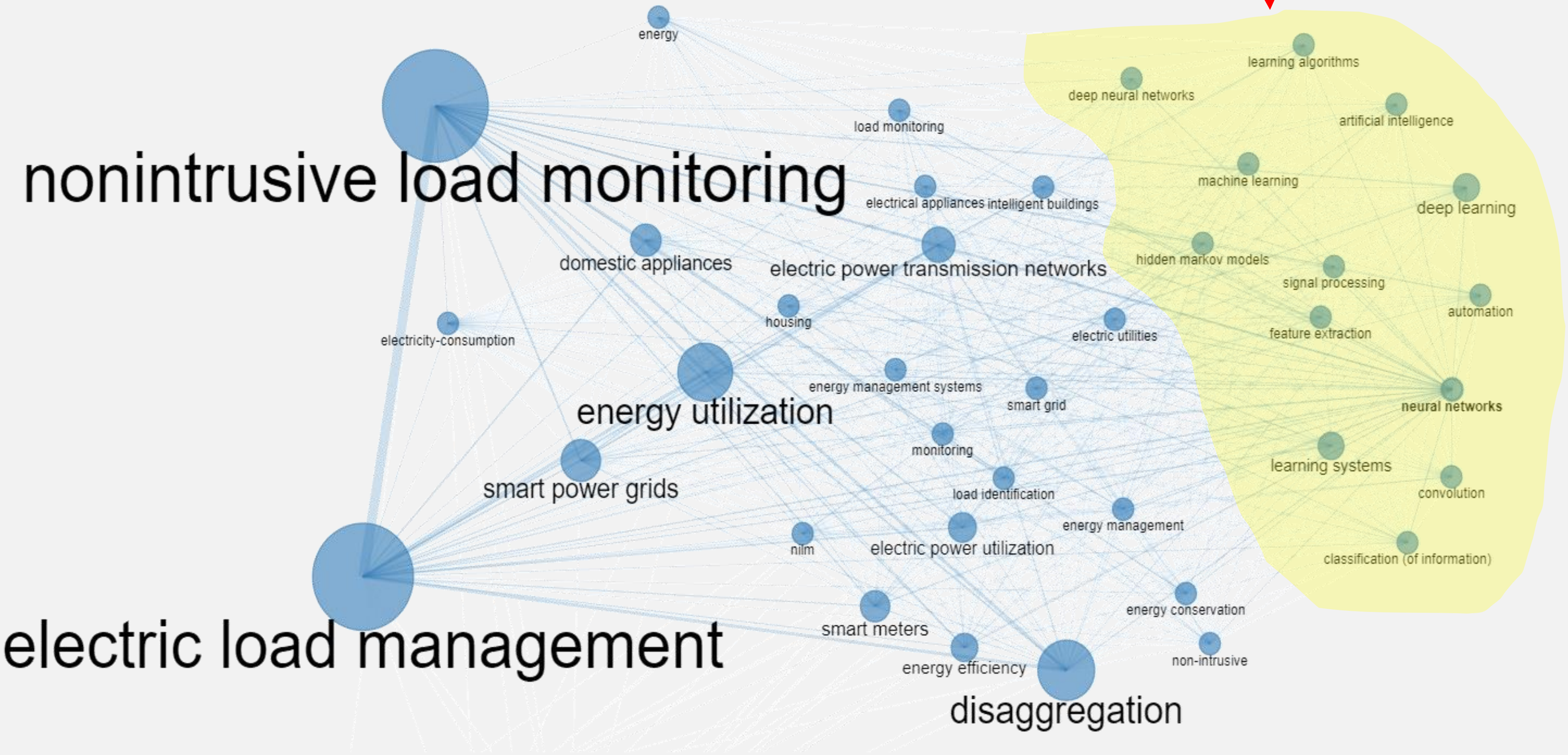
NILM toolkits

Deep-NILMtk

Hands-on session

Limitations and outlook

Most NILM models are ML models



Co-occurrence network generated with the bibliometrics R package for NILM contributions published between 2019-2023.

Best practices in ML development

1. Several best practices catalogues exist.

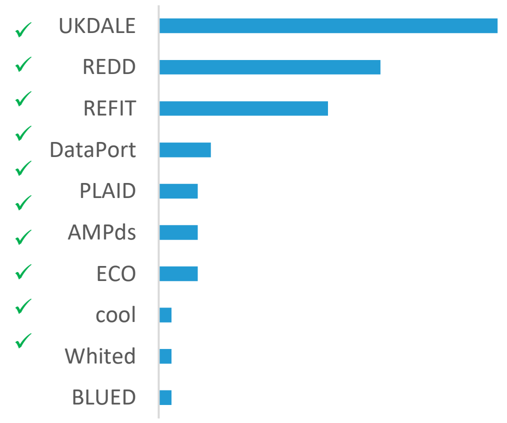
2. The main aim of these catalogues is to provide a tool for scientists to improve their efficiency and enhance reproducibility, replicability and transparency.

3. There are even new fields, such as MLOps, to achieve the previous goals.

4. Many of these practices are relevant to the NILM research.

NILM toolkits: The good and the bad

Data Parsers for NILM datasets



Advanced functions for analysis

NILMtk contains a set of appliances that allow to efficiently analyze the energy data (e.g., generate states and check data quality).

- ✓ Meters connections
- ✓ Available appliances per building
- ✓ Statistical summary of available data

The new API

Model development API allowing fast and easy experimental design of NILM models offering full compatibility with the toolkit functions.

- ✓ Decoupled model development from toolkit specific data structures.
- ✓ Implementation of deep baselines

Facilitate exploratory data analysis of the available NILM datasets and the evaluation NILM models.

NILM toolkits: The good and the bad

1 The NILM pipeline

In its current version the toolkit, the available toolkit allows implementing a static NILM pipeline that must be re-implemented each time a new model is introduced.

2 Non-availability of pre-designed experiments

A main problem in the NILM scholarship is the comparability problem due to different experimental setups relying on different parts of a dataset.

3 Lack of DL best practices

The toolkit does not offer any features allowing to ease the use of best practices leading to an efficient research process.

4 Poor compatibility with DL frameworks

The current version of the toolkit is only compatible with Tensorflow and Keras preventing scholars from using other frameworks to benefit from the toolkit.

NILM toolkits: The good and the bad

	Keras & TF	PyTorch	Hyper-parameter Optimisation	Cross-validation	Experiment tracking & results logging	Possibility of direct testing of a new model
NILMtk-contrib	•					
Torch-NILM		•	•	•		
Deep-NILMtk	•	•	•	•	•	•

Features of NILM toolkits developed in Python

Deep-NILMtk: The toolkit

1. Deep-NILMtk is an open-source package designed specifically for deep models.

2. It implements the general NILM pipeline independently of the deep learning backend. In its current version, the toolkit considers two of the most popular deep-learning pipelines.

3. The training and testing phases are fully compatible with NILMtk.

4. Several MLOps tools are included that aim to support the evaluation and the validation of those models (e.g., cross-validation, hyper-params optimisation) in the case of NILM.

Deep-NILMtk: The implemented pipeline

Motivations

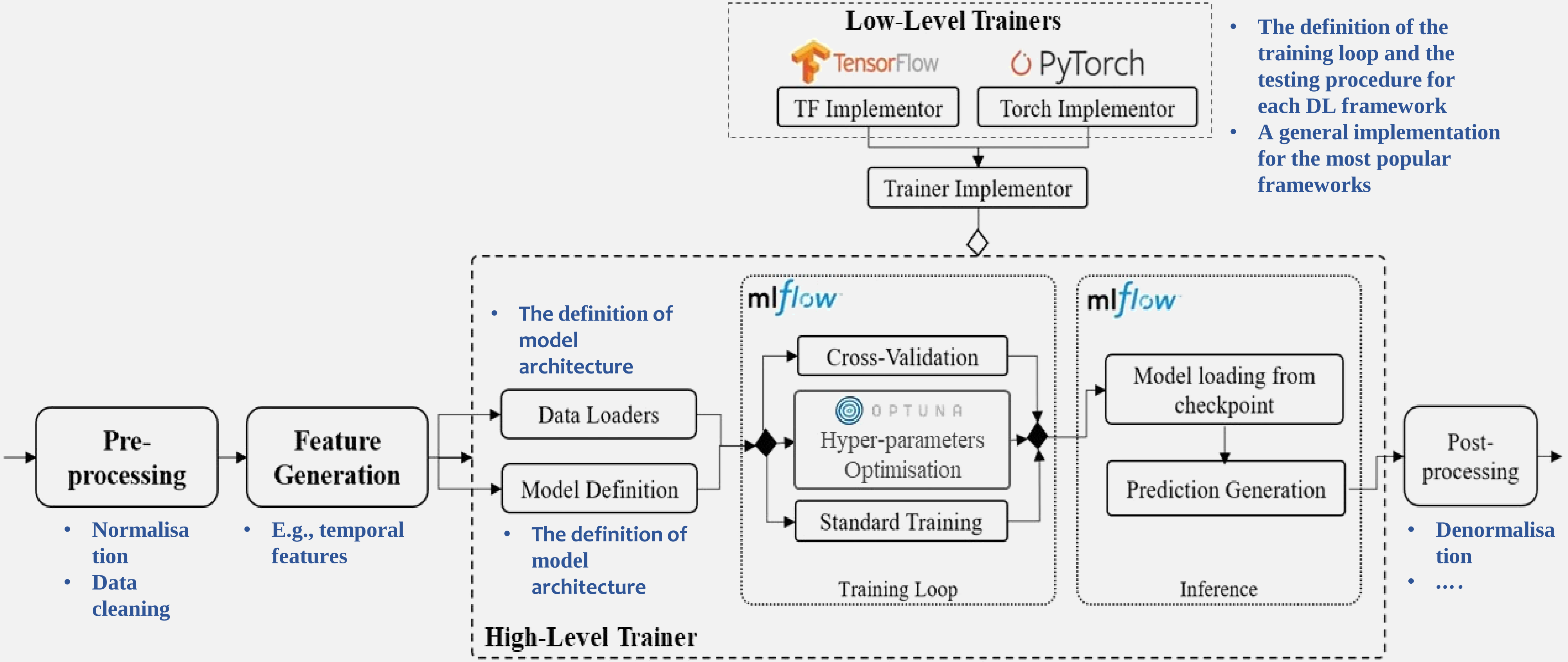
Best practices in ML development

NILM toolkits

Deep-NILMtk

Hands-on session

Limitations and outlook



- The definition of the training loop and the testing procedure for each DL framework
- A general implementation for the most popular frameworks

- The definition of model architecture

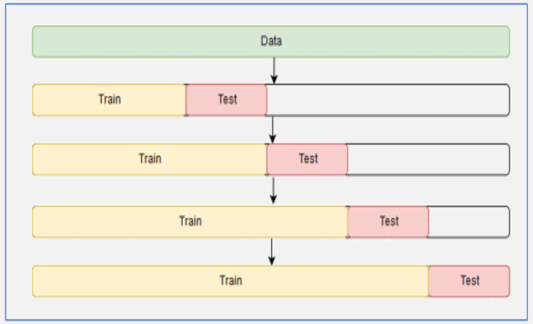
- Normalisation
- Data cleaning

- E.g., temporal features

- The definition of model architecture

- Denormalisation
- ….

Deep-NILMtk: The implemented pipeline

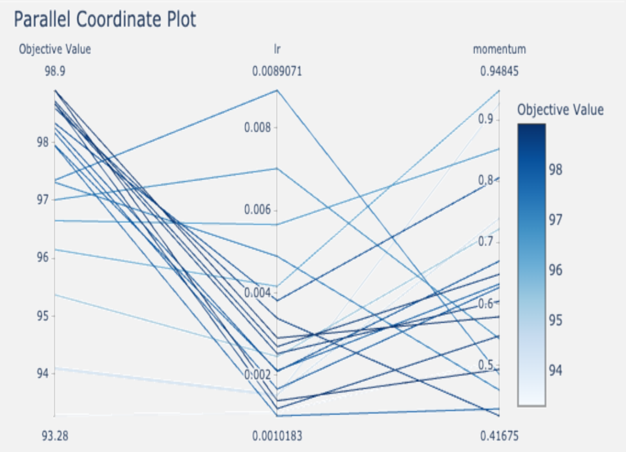


Timesplit cross-validation

01

Time-Split Cross-validation

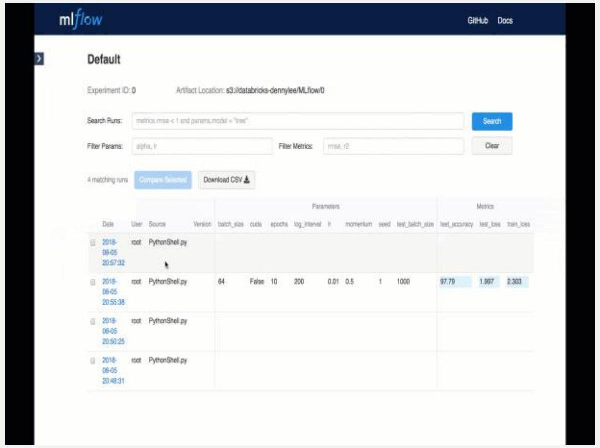
Deep-NILMtk implements k-folds time-split cross-validation. The performance results are obtained considering all the models to provide a clearer picture of the performance variation over different sets



02

Hyper-parameters Optimization with Optuna

Deep-NILMtk implements automatic hyper-parameter optimization leveraging the Optuna Framework. Unpromising trials are stopped at the early stages for faster exploration of the considered space.



03

Experiment with Managing and Tracking

Deep-NILMtk use MLflow API to trace training metrics. The experiments are organized according to appliances, with each experiment gathering multiple runs related to the different trained models.

Deep-NILMtk: Experiment-templating

1. What sampling rate to choose for evaluation and comparison?
2. What time periods to select for training and testing periods?
3. Is there a possibility to test only new models and perform a direct comparison?

```
experiment = {
  'power': {'mains': ['active'], 'appliance': ['active']},
  'sample_rate': 60, 'artificial_aggregate':False,
  'appliances': ['fridge', 'washing machine'],
  'methods': {'CO': {},
    'FHMM_EXACT': {'num_of_states': 2},
    'Seq2Point': {
      'n_epochs': 1,
      'pre-processing': {
        'appliance_params': {
          'washing machine': {
            'mean': 400,'std': 700},
          'fridge': {
            'mean': 200,'std': 400 },
        }
      },
    }
    },
  'train': {
    'datasets': {
      'REDD': {
        'path': '/data/REDD/redd.h5',
        'buildings': {
          1: {'start_time': '2011-04-01','end_time': '2011-04-30'},
          2: {'start_time': '2011-04-01','end_time': '2011-04-30'}
        }
      }
    }
  },
  'test': {
    'datasets': {
      'IAWE': {
        'path': '/data/IAWE/iawe.h5',
        'buildings': {
          1: {'start_time': '2015-08-05','end_time': '2015-08-10'}
        }
      }
    },
    'metrics': ['mae', 'rmse']
  }
}
```

UNIVERSITÄT KLAGENFURT

Deep-NILMtk: Experiment-templating

"Predesigned set of NILM experiments usable with baselines and new models"

```
template = ExperimentTemplate( data_path=DATA_PATH,
                               template_name='ukdale',
                               list_appliances=['kettle'],
                               list_baselines_backends=[('Seq2Pointbaseline', 'pytorch')],
                               in_sequence=480,
                               out_sequence=1,
                               max_epochs=MAX_EPOCHS)
```

```
new_model =  NILMExperiment({...})
template.extend_experiment({
    'new_model':new_model
})
template.run_template(EXPERIMENT_NAME,
                      RESULTS_PATH,
                      f'{RESULTS_PATH}/mlflow')
```

<u>Advantages:</u>

1. Allows direct comparison and saves resources (time and comparison).
2. Allows fast prototyping.
3. Allows to build standard evaluation snapshot datasets for DNN-NILM models.
4. Allows to join forces of scholars working on energy data and scholars working model development.

Break ☺ !!

UNIVERSITÄT KLAGENFURT

Limitations & Outlook

- **Limitations**

1. The models from NILMtk-contrib have not yet been integrated. Only the sequence-to-point baseline is available.
2. Deep-NILMtk focuses only on eventless DNN-NILM models.
3. Deep-NILMtk focuses only on centralised training.

- **Outlook**

1. Augment Deep-NILMtk with a pipeline for event-based NILM.
2. Augment Deep-NILMtk with a decentralised trainer allowing the simulation of actual smart grid setups.
3. Further meta-research work aiming at improving transparency and reproducibility in NILM industry.

Extra Slides

Best practices in ML development

Motivations

Best practices in ML development

NILM toolkits

Deep-NILMtk

Hands-on session

Limitations and outlook

Lindauer, M., & Hutter, F. (2020). Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, *21*(243), 1-18.

Best practices for releasing code	Best practices for comparisons	Best practices for reporting
• Release code for the training pipeline • Release code of the model • Release code asap	• Evaluate using same protocol • Evaluate performance as a function of computational resources. • Use same benchmarks (e.g., NAS-Bench-301) • Multiple runs • Run ablation studies	• Report hyper-parameters used • Report resources required • Report details of experimental setup

Best practices in ML development

Serban, A., van der Blom, K., Hoos, H., & Visser, J. (2020, October). Adoption and effects of software engineering best practices in machine learning. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-12).

Best practices for data	Best practices for training
• Use sanity checks for all data • Use public datasets • Check that input data is complete • Ensure data labeling is performed in a strictly controlled process • Reusable scripts for data cleaning and merging • Use data versioning tools	• Share a clear definition of the training objective • Capture the training objective in a metric that is easy to measure and understand • Test all feature extraction code • Enable parallel training of experiments • Automate hyper-parameters optimization and model selection The least adopted as they require more effort and knowledge