BaseNILM: A Simple Toolkit for Energy Disaggregation (V.0.0)

1. Introduction

BaseNILM is a simple toolkit for Non-Intrusive Load Monitoring (NILM) and Energy Disaggregation. The aim is to provide a baseline system for researchers entering the area of NILM to enable them to contribute with new ideas without having to build their own NILM toolkit. The aim is not to include all best performing approaches that have been published in the literature so far, but to provide a set of reference approaches that can be used for comparison of new ideas. As failure and mistakes are inextricably linked to human nature, the toolkit is obviously not perfect, thus suggestions and constructive feedback are always welcome.

1.1. Publication

The BaseNILM toolkit is part of the following NILM survey paper and tries to replicate the presented architectures and disaggregation approaches. Please cite the following paper when using the BaseNILM toolkit:

P. A. Schirmer and I. Mporas, Non-Intrusive Load Monitoring: A Review

Furthermore, please do also cite the corresponding publicly available datasets. For a complete list of all publicly available datasets please see the NILM survey paper.

1.2. Dependencies

The BaseNILM Toolkit was implemented using the following dependencies:

- Python 3.8
- Tensorflow 2.5.0
- Keras 2.4.3

For GPU based calculations CUDA in combination cuDNN has been used, utilizing the Nvidia RTX 3000 series for calculation. The following versions have been tested and proven to work with the BaseNILM tookit:

- CUDA 11.4
- cuDNN 8.2.4
- Driver 472.39

1.3. Folder Structure

The folder structure of the BaseNILM system can be found below:

Table 1: BaseNILM folder structure.

BaseNILM	**Folder**	**Subfolder**	**Content**
|--	data		Contains all datafiles
|--	docu		Contains the documentation
|--	lib		Contains all functions
	|--	fnc	Contains all help function
	|--	mdl	Contains the regression models
|--	mdl		Contains the model weights
|--	results		Contains the results
|--	setup		Contains setup files

2. Architecture

The Architecture described in the NILM survey paper is presented in Fig. 1. The aim of the BaseNILM toolkit is it to replicate the architecture as close as possible to enable new researchers to enter the area of energy disaggregation with ease.

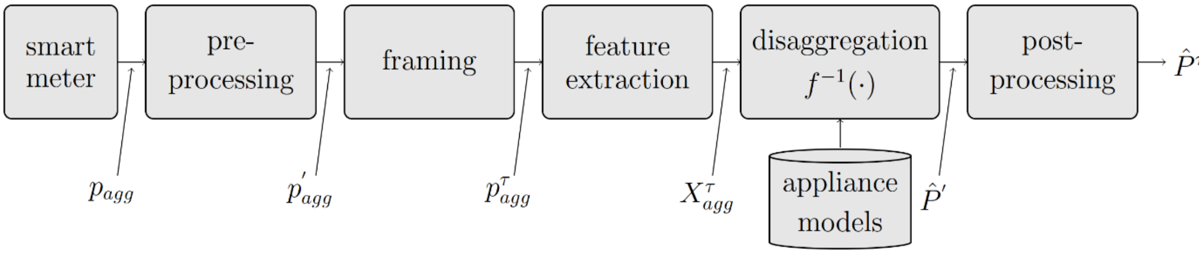

Fig. 1: Architecture implemented in the BaseNILM system.

The architecture presented in Fig. 1 is implemented in the BaseNILM toolkit. The aim was that each of the six steps, namely smart meter, pre-processing, framing, feature extraction, disaggregation, and post-processing, is exactly reproduced in the implementation. However, due to the necessity of having a training and testing process and the fact that transfer learning on different datasets should also be realized this was only partially achieved. The following table gives a mapping of the six steps and the corresponding functions in the implementation:

Table 2: I/O relations for the main functions of the BaseNILM tool.

Name	Functions	Inputs	Outputs
Smart Meter	loadData.py	data path, data name	raw data
Pre-Processing	preprocessing.py	raw data	pre-processed data
Framing	framing.py	raw data	time frames
Feature Extraction	features.py	time frames	features
Disaggregation	trainXXX.py / testXXX.py	features, model setup	trained model
Post-Processing	postprocessing.py	predicted results	post-processed results

3. Experimental Setups

The BaseNILM tool offers a set of pre-implemented option to configure the NILM disaggregation architecture. The complete list, including description and possible values, is tabulated in Table 3.

Table 3: Complete options for the configuration of the BaseNILM tool and experimental setup.

Experimental Setup				
Name	**Values**	**Default**	**Notes**	**Units**
experiment_name	String	test	Name of the experimental setup that will be also used as label for results and models.	[]
author	String	Pascal	Name of the author running the experiment.	[]
configuration_name	String	baseNILM	Name of the configuration.	[]
train	[0; 1]	1	If 1 a new model will be trained	[]
test	[0; 1]	1	If 1 testing will be performed	[]
plotting	[0; 1]	1	If 1 results will be plotted	[]
saveResults	[0; 1]	0	If 1 results will be saved to .\results	[]
Data Setup				
dataset	ampds, redd,	redd	Name oft he dataset	[]
shape	[2; 3]	2	Number of dimensions of the dataset	[]
granularity	Integer	3	Integer number of the sampling time	[sec]
downsample	Integer	1	Integer factor for down sampling the data	[]
houseTrain	Integer	[1, 3]	Integer number for the house used for training	[]
houseTest	Integer	2	Integer number for the house used for testing	[]
houseVal	Integer	5	Integer number for the house used for validation	[]
testRatio	[0, 1]	0.1	Ration between testing and training data.	[]
selApp	Integer Array	[0, 1, 2, 3, 4]	Integer indices of appliances	[]
ghost	[0; 1; 2]	0	If 0 ghost data will be ignored, if 1 ghost data will be treated as own appliance, if 2 ideal data (without ghost data) will be used	[]
normData	[0; 1; 2; 3; 4]	1	If 0 no normalization will be used, if 1 min/max, if 2 min/max (one common value), if 3 mean/std, if 4) min/max using training data	[]
meanX	Integer	522	Mean value of the aggregated data for normalization	[W]
meanY	Integer Array	[500, 200, 700, 400]	Mean values of the appliance data for normalization	[W]

stdX	Integer	814	Std value of the aggregated data for normalization	[W]
stdY	Integer Array	[800, 400, 1000, 700]	Std values of the appliance data for normalization	[W]
neg	[0; 1]	0	If 1 negative data will be removed during pre-processing	[]
Parameter Setup				
algorithm	[0; 1]	1	If 0 classification is used if 1 regression	[]
classifier	String	RF	possible classifier: 1) ML: RF, CNN, LSTM \ 2) PM: DTW \ 3) SS:	[]
trans	[0; 1]	0	If 0 'houseTest' is split into test and training set as specified in 'testRatio', if 1 transfer learning is applied, e.g. 'houseTrain' are used for training and 'houseTest' and 'houseVal' for testing and validation respectively	[]
framelength	Integer	10	Number of samples per frame	[]
overlap	Integer	9	Number of samples overlaping between two successive frames	[]
p_Threshold	Integer	50	Binary threshold deciding if a device is on or off	[W]
multiClass	[0; 1]	0	If 0 one model per appliance is used, if 1 one model for all appliances is used	[]
seq2seq	[0; 1]	0	If 0 seq2point is used, if 1 seq2seq is used (only if multiClass=0 and overlap=0)	[]
feat	[0; 1; 2]	0	if 0 raw values are used, if 1 1D features are calculated, if 2 2D feature are calculated (only for shape 3 data)	[]
Model Setup				
batch_size	Integer	1000	Batch size for DNN based approaches	[]
epochs	Integer	100	Number of epochs for training	[]
valsteps	Integer	50	Number of validation steps	[]
shuffle	Boolean	False	Applying shuffeling during training	[]
verbose	[0; 1; 2]	0	Level of displaying training progress	[]
n_neighbors	Integer	5	Number of neighbors for KNN	[]
max_depth	Integer	10	Maximum depth for random forest	[]
random_state	Integer	0	Number of random states for random forest	[]
n_estimators	Integer	32	number of estimators for RF	[]
kernel	String	Rbf	Kernel function for SVM	[]
C	Integer	100	regularization parameter SVM	[]
epsilon	Double	0.1	epsilon parameter SVM	[]
gamma	Double	0.1	scale parameter SVM	[]

4. Datasets

Currently the base NILM tool allows only to read data from '.mat' files. The files must be named according to the name of the dataset and must be arrays of size $(M+2) \times T \times F$, where the first two columns are used for the timestamp and the aggregated signals, while the other M columns are used for the appliance signals. For multi-dimensional datasets, a third dimension for F features can be included. In the current version of the BaseNILM tool the following datasets are provided:

- Redd1 – Redd6
- ReddTrans1 – ReddTrans6
- Ampds2

5. Results

In the following chapter a set of reference results is provided using first all appliances available in a respective dataset and then the so-called deferable loads in a second scenario. Ten-fold cross validation has been used to evaluate the performance, while only raw samples have been used and a frame length of ten was used using sequence to point learning. The full experimental setups can be found in the respective setup files. The results are presented in Table 4 and Table 5.

Table 4: Benchmark results for the REDD dataset using all appliances in each dataset (including the modelling of ghost power). One model for all appliances has been used.

Dataset	**KNN**		**RF**		**LSTM**		**CNN**	
	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**
REDD-1	80.92	7.96	79.38	8.55	76.26	9.55	81.73	7.44
REDD-2	86.28	6.45	85.85	6.62	83.26	7.81	88.14	5.58
REDD-3	72.03	13.25	72.98	12.81	74.14	11.91	77.22	10.72
REDD-4	71.19	11.26	71.82	11.14	71.64	10.53	76.82	8.87
REDD-5	64.61	22.00	67.74	18.80	64.09	20.70	67.43	20.29

REDD-6	60.26	24.71	59.32	25.08	61.28	23.16	65.71	20.64
AVG	72.55	14.27	72.85	12.83	71.78	13.94	**76.18**	**12.26**
$\boldsymbol{AVG_{1-4,6}}$	74.14	12.73	73.87	12.84	73.32	12.59	**77.92**	**10.65**

Table 5: Benchmark results for the REDD dataset using the deferrable appliances in each dataset (including the modelling of ghost power). One model for all appliances has been used.

Dataset	KNN		RF		LSTM		CNN	
	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE
REDD-1	86.63	20.92	85.17	23.07	84.44	23.32	87.18	19.65
REDD-2	90.89	8.52	90.48	8.88	88.98	10.22	92.14	7.36
REDD-3	87.03	23.33	87.54	22.41	87.77	21.57	89.15	19.43
REDD-4	85.85	19.36	85.98	19.37	86.59	18.10	88.32	15.88
REDD-5	87.48	34.75	87.47	35.09	86.67	32.79	90.98	21.65
REDD-6	90.19	18.39	89.60	19.10	85.33	25.80	89.89	18.47
AVG	88.01	20.88	87.71	21.32	86.63	21.97	**89.61**	**17.07**
$\boldsymbol{AVG_{1-4,6}}$	88.12	18.10	87.75	18.57	86.62	19.80	**89.34**	**16.16**

Furthermore, some enhanced results are calculated using the Ampds2 dataset and multi-dimensional feature vectors. The results, as well as comparisons with the literature, are presented in Table 6.

Table 6: Benchmark results for the Ampds2 dataset using the deferrable appliances and two-dimensional PQ signatures (current is used as output feature as in, ghost power is not modelled and 10-fold cross-validation is applied).

Appliances	PQ		WaveNILM [1]		HMM [2]		Frac. Calc. [3]	
	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE	$\boldsymbol{E_{ACC}}$	MAE
DWE	61.65	0.10	-	-	-	-	-	-
FRE	94.92	0.14	-	-	-	-	-	-
HPE	97.23	0.08	-	-	-	-	-	-
WOE	91.27	0.03	-	-	-	-	-	-
CDE	97.62	0.02	-	-	-	-	-	-
AVG	94.91	0.07	93.9	-	94.0	-	94.7	-
AVG ALL	88.12	0.19	87.5	-	-	-	88.9	-

6. Conclusion

A python implementation for NILM has been presented. While, several features have been included already, the toolkit is far away from being complete. New models, datasets, features and functionalities will be successively added in the future. We hope the toolkit is useful to new researcher entering the area of NILM.

7. References

[1] Harell, A., Makonin, S., & Bajić, I. V. (2019, May). Wavenilm: A causal neural network for power disaggregation from the complex power signal. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8335-8339). IEEE.

[2] Makonin, S., Popowich, F., Bajić, I. V., Gill, B., & Bartram, L. (2015). Exploiting HMM sparsity to perform online real-time nonintrusive load monitoring. IEEE Transactions on smart grid, 7(6), 2575-2585.

[3] Schirmer, P. A., & Mporas, I. (2020, May). Energy disaggregation using fractional calculus. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3257-3261). IEEE.