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. Furthermore, the aim of the implementation is to offer a structure that can easily followed and adapted. 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 and Citation

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. As well as [4] when using the data balance option, [5] when using the WaveNet pytorch implementations and [6] when using the DSC implementation. For a complete list of all publicly available datasets please see the NILM survey paper.

<u>AMPds2 (CC-BY 4.0)</u>

Makonin, S. et al. Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. Sci. Data 3:160037 doi: 10.1038/sdata.2016.37 Titel anhand dieser DOI in Citavi-Projekt übernehmen (2016).

<u>REDD (MIT)</u>

Kolter, J. Zico. "REDD : A Public Data Set for Energy Disaggregation Research." (2011).

<u>REFIT (CC-BY 4.0)</u>

Firth, Steven; Kane, Tom; Dimitriou, Vanda; Hassan, Tarek; Fouchal, Farid; Coleman, Michael; et al. (2017): REFIT Smart Home dataset. Loughborough University. Dataset. https://doi.org/10.17028/rd.lboro.2070091.v1

<u>UK-DALE (CC BY 4.0)</u>

Kelly, J., Knottenbelt, W. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. Sci Data 2, 150007 (2015). https://doi.org/10.1038/sdata.2015.7

1.2. Dependencies

The requirements of the BaseNILM toolkit are summarized in the requirements.txt data file. In detail, 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.

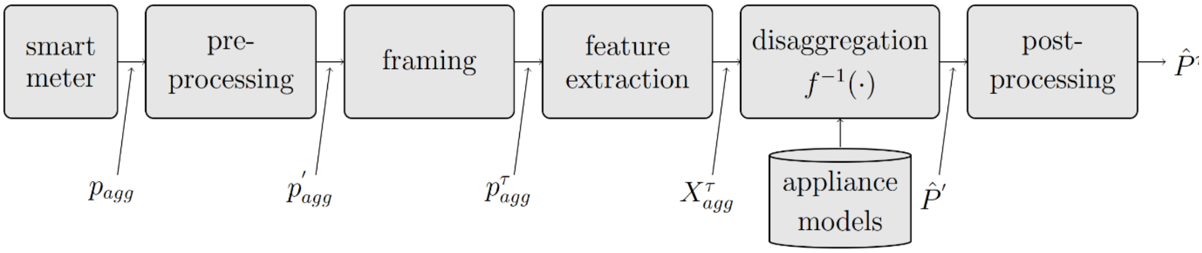

Fig. 1: Architecture implemented in the BaseNILM system.

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	train.py / test.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]	0	If 1 results will be plotted, if 2 time series will be plotted	[]
log	[0; 1]	0	If 1 logs are saved	[]
saveResults	[0; 1]	0	If 1 results will be saved to .\results	[]
Data Setup				
dataset	ampds, redd	redd	Name oft the dataset	[]
shape	[2; 3]	2	Number of dimensions of the dataset	[]
output	Integer	1	select output if Y is multidimensional (e.g. AMPds 0) P, 1) I, 2) Q, 3) S)	[]
granularity	Integer	3	Integer number of the sampling time	[sec]
downsample	Integer	1	Integer factor for down sampling the data	[]
limit	Integer	0	limit number of data points	[]
houseTrain	Integer	[2]	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.	[]
kfold	Integer	10	if 1) 'testRatio' is used for data splitting, otherwise k-fold cross validation	
selApp	Integer Array	[]	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: 5]	5	normalize data, if 0) none, 1) min-max (in this case meanX/meanY are interpreted as max values), 2) min/max one common value (meanX), 3) mean-std, 4) min/max using train-data 5) mean-std using train data	[]
normXY	[1; 2; 3]	3	1) X is normalized, if 2) Y is normalized, if 3) X and Y are normalized	
meanX	Integer	1	Mean value of the aggregated data for normalization	[W]
meanY	Integer Array	[1, 1, 1, 1, 1]	Mean values of the appliance data for normalization	[W]
stdX	Integer	0	Std value of the aggregated data for normalization	[W]
stdY	Integer Array	[0, 0, 0, 0, 0]	Std values of the appliance data for normalization	[W]
neg	[0; 1]	0	If 1 negative data will be removed during pre-processing	[]
Inactive	[0; 1]	0	if 0) off, if >0 inactive period will be removed from the training data (multiclass 0)	
Balance	[0; 1; 2]	0	if 0) data is not balanced >1) ratio of positive and negative batches is balanced (only when using seq2seq)	
Filt	String	None	If 'none' no filtering of data is applied, if 'median' median filtering is applied	[]
Filt_len	Integer (Odd)		length of the filter (must be an odd number)	[]
Parameter Setup				
Solver	String	SK	TF: Tensorflow, PT: PyTorch, SK: sklearn, PM: Pattern Matching and SS: Source Separation	
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	[]
opt	[0; 1]	0	if >0 models are optimized using keras tuner (note inputs and outputs of hypermodel must be set manually)	[]
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) the values is equal to the length of the output sequence	[]
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	[]
Patience	Integer	15	number of epochs patience when 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	[]
cDTW	float	0.1	pattern matching constraint on mdl size (%)	[]

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, e.g. active power, reactive power or current. In the current version of the BaseNILM tool the following datasets are provided (please note that the datasets have been reshaped and reformatted and thus are not exactly comparable to the original versions):

- Redd1 – Redd6 and ReddTrans1 – ReddTrans6
- Ampds2
- ukDaleTrans1 – ukDaleTrans5

5. Models

The BaseNILM model offers the possibility to utilize the three major modelling techniques, namely machine learning, pattern matching and source separation to perform energy disaggregation. In detail, tensorflow with keras backend (trainMdlTF.py), pytorch (trainMdlPT.py) and sklearn (trainMdlSK.py) are implemented for machine learning techniques to have a wide variety of models that can be utilized. Furthermore, for pattern matching and source separation custom modules are implemented, namely trainMdlPM.py and trainMdlSS.py. Moreover, the is a function trainMdlCU.py allowing completely custom implementation of NILM techniques, allowing new users to add their own frameworks and implementation approaches. The corresponding models are stored in models.py. The following models are currently available in the BaseNILM toolkit:

- Machine Learning: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Random Forest (RF), K-Nearest-Neighbour (KNN) and Support Vector Machine (SVM)
- Pattern Matching: Dynamic Time Warping (DTW), Global Alignment Kernel (GAK) and Minimum Variance Matching (MVM)
- Source Separation: Non-Negative Matrix Factorization (NMF) and Discriminative Sparse Coding (DSC)

6. 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. It must be noted that these results are not near the best performing approaches in the literature and only illustrate the operational principle of the BaseNILM toolkit. For a comparison with the literature achieving state-of-the art performance the reader is referred to the results in Table 6.

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	
	E_{ACC}	MAE	E_{ACC}	MAE	E_{ACC}	MAE	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 all input features (current is used as output feature, ghost power is not modelled and 10-fold cross-validation is applied).

Appliances	**CNN**		**WaveNILM [1]**		**HMM [2]**		**Frac. Calc. [3]**	
	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**	$\boldsymbol{E_{ACC}}$	**MAE**
DWE	72.37	0.12	-	-	-	-	-	-
FRE	95.60	0.13	-	-	-	-	-	-
HPE	97.80	0.06	-	-	-	-	-	-
WOE	95.79	0.02	-	-	-	-	-	-
CDE	98.04	0.02	-	-	-	-	-	-
AVG	95.55	0.07	93.9	-	94.0	-	94.7	-
AVG ALL	90.84	0.10	90.2	-	-	-	88.9	-

Moreover, the ampds2 dataset (using all appliances and the deferrable loads) was chosen to compare different models with each other. In detail, five-fold cross validation was applied (without modelling of ghost power) and current was used as an input and output feature. The results for all loads can be found in Table 7 and for the deferrable loads in Table 8.

Table 7: Benchmark results for the ampds2 dataset using all appliances and different modelling techniques. (*MVM is only applicable to one-dimensional data, thus the results are significantly worse)

Results	**Machine Learning**			**Pattern Matching**			**Source Separation**	
	RF	**LSTM**	**CNN**	**DTW**	**GAK**	**MVM***	**NMF**	**DSC**
ACC	86.78	93.26	93.29	91.79	91.63	90.14	48.82	58.54
F1	86.90	93.30	93.37	92.33	92.19	90.78	61.20	62.02
$\boldsymbol{E_{ACC}}$	81.62	90.62	90.94	86.65	86.47	79.29	36.62	45.47
RMSE	2.08	1.74	1.65	2.83	2.52	2.89	2.42	3.64
MAE	0.21	0.11	0.10	0.15	0.15	0.23	0.74	0.64
SAE	0.002	0.042	0.041	0.001	0.000	0.002	0.000	0.205

Table 8: Benchmark results for the ampds2 dataset using deferrable appliances and different modelling techniques. (*MVM is only applicable to one-dimensional data, thus the results are significantly worse)

Results	**Machine Learning**			**Pattern Matching**			**Source Separation**	
	RF	**LSTM**	**CNN**	**DTW**	**GAK**	**MVM***	**NMF**	**DSC**
ACC	95.13	99.04	99.29	99.13	99.04	98.57	51.03	75.15
F1	96.66	98.83	99.00	99.08	98.99	98.53	55.09	82.95
$\boldsymbol{E_{ACC}}$	87.26	95.03	95.71	93.09	93.10	85.07	38.12	59.43
RMSE	2.37	1.27	1.19	2.17	1.96	3.54	0.37	3.76
MAE	0.21	0.08	0.07	0.11	0.11	0.25	2.23	1.47
SAE	0.006	0.038	0.039	0.002	0.001	0.012	0.001	0.232

7. Quick Start

For a first test run use start.py to train, test and plot a 10-fold cross validation using the AMPds2 dataset with five loads (deferrable loads). If you don't want to train simply set 'setup_Exp['train']=0' as the models for the example test run are already stored in BaseNILM \mdl. For changing parameters and adapting the parameters please refer to chapter 3. The average results for 10-fold cross validation can be found in Table 6 as well as below.

	FINITE STATES		POWER ESTIMATION			PERCENT OF TOTAL	
item ID	ACCURACY	F-SCORE	E-ACCURACY	RMSE	MAE	EST	TRUTH
DWE	97.22%	95.86%	72.37%	0.87%	0.12%	2.51%	5.54%
FRE	99.98%	99.97%	95.60%	0.24%	0.13%	35.58%	36.53%
HPE	99.99%	99.99%	97.80%	0.58%	0.06%	37.60%	37.55%
WOE	99.39%	99.31%	95.79%	0.57%	0.02%	6.51%	6.77%
CDE	99.93%	99.93%	98.04%	0.69%	0.02%	13.81%	13.61%
AVG	99.30%	99.01%	95.55%	1.26%	0.07%	96.01%	100.00%

8. 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.

9. 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.

[4] Pan, Yungang, et al. "Sequence-to-subsequence learning with conditional gan for power disaggregation." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

[5] Jiang, Jie, et al. "Deep Learning-Based Energy Disaggregation and On/Off Detection of Household Appliances." ACM Transactions on Knowledge Discovery from Data (TKDD) 15.3 (2021): 1-21.

[6] Batra, Nipun, et al. "Towards reproducible state-of-the-art energy disaggregation." Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation. 2019.