

---

**NN for CHF**

**Dumont Jules**

**Sep 30, 2023**



**CONTENTS:**

<b>1</b>	<b>CHF_model_api</b>	<b>1</b>
1.1	CHF_model_api package . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>5</b>
	<b>Python Module Index</b>	<b>7</b>
	<b>Index</b>	<b>9</b>



## CHF\_MODEL\_API

### 1.1 CHF\_model\_api package

#### 1.1.1 Submodules

#### 1.1.2 CHF\_model\_api.config module

#### 1.1.3 CHF\_model\_api.myModel module

**class** CHF\_model\_api.myModel.**MyModel**(*hparams: dict | None = None, model\_name: str | None = None, auto\_save: bool = False, process\_number: int | None = None*)

Bases: object

**DATA** = {}

To complete

**displayPerf()**

**classmethod makeDataCopies**(*jobs, seed, input\_number*)

**makeRealPredictions**(*features\_data: list*) → list

**plotResult()** → None

**save**(*overwrite=False*) → None

save model in .h5 format, to make predictions save also the hparams of the model in json files

**train**(*logs=True, callbacks=True, train\_epochs=None*) → dict

train the model over train\_epochs epochs and save the results in the attribute hparams

**vizualize()** → None

save a png file containing vizual rpz of the network

CHF\_model\_api.myModel.**batch\_nrmse**(*y\_true, y\_pred*) → float

### 1.1.4 CHF\_model\_api.myTensorboard module

```
class CHF_model_api.myTensorboard.MyTensorboard(saved_models_name: List[str])
    Bases: object
    add_hparams()

    add_model(model) → None

    add_saved_model(model_name: str) → None

    copy_log(name) → None
        copy logs in ./logs/modelname/train et /validation to the same directories in ./hparams_tuning/sessionname

    copy_logs() → None
        get the logs of all the models to compare

    init_models() → List[MyModel]
        get the saved models by creating object my_models based on the name of the saved models

    load_tb() → None

    log_Hparams() → None

    log_Hparams_range() → None

    make_Hparams_range() → None

    set_up_directories() → None
        create validation and training directory and also a directory for each different model (with diff hp)
```

### 1.1.5 CHF\_model\_api.optimizer module

```
class CHF_model_api.optimizer.MyOptimizer(generic_hparams: dict, opti_architecture: bool = True,
                                           opti_dropout: bool = False, opti_learning_rate: bool = False,
                                           opti_lr_decrease: bool = False, opti_optimizer: bool = False,
                                           opti_activation_method: bool = False, type: int = 1, jobs: int
                                           = 2, trials: int = 5, db_name: str | None = None, metric: str
                                           = 'mpe', verbose: int = 0)

    Bases: object

    optimize_my_models()
```

### 1.1.6 CHF\_model\_api.tools module

```
CHF_model_api.tools.RemoveDirectoryContent(directory_path) → None
    just a function to clean when we want to reset

CHF_model_api.tools.eraseHparams(model_name: str) → None
    Erase stored hyperparameters

CHF_model_api.tools.extractFromPdf(path) → DataFrame
    create a csv file based on Groeneveld 2006 LUT pdf take 2 min to run
```

---

`CHF_model_api.tools.getHparamsSavedModel(model_name: str) → dict`  
 return a dict with all the hyperparameters saved in the directory hparams in saved\_models

`CHF_model_api.tools.loadData(data_seed: int = 1, input_number: int = 5) → dict`  
 take the data from a csv containing data SI units or create it from the Groeneveld 2006 LUT pdf return a dict containing the keys: 'validation\_targets', 'validation\_features', 'training\_features', 'training\_targets', 'mean', 'std' (mean and std of the training\_features before normalization we need to keep when predicting) and is meant to be add to DATA[seed] = {'validation':...}

`CHF_model_api.tools.myMsle(y_true, y_pred) → float`  
 Compute mean squared logarithmic error

`CHF_model_api.tools.nrmse(y_true, y_pred) → float`  
 Compute normalised root mean squared error

`CHF_model_api.tools.plotResults(predictions, y_val, save_fig=False)`

`CHF_model_api.tools.remove_backups(name) → None`

`CHF_model_api.tools.reset_directories() → None`

`CHF_model_api.tools.saveHparams(model_name: str, hparams: dict) → None`  
 save the dict containing results and hyperparameters in hparams idrectory

`CHF_model_api.tools.stdMP(y_val, predictions) → float`  
 compute the standart deviation of the ration Measured / predict from 1

`CHF_model_api.tools.utilsNnConfig(model)`

`CHF_model_api.tools.visualizeNn(model, name, description=False, figsize=(10, 8))`  
 Plot the structure of a keras neural network.

### 1.1.7 Module contents





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### C

- `CHF_model_api`, 3
- `CHF_model_api.config`, 1
- `CHF_model_api.myModel`, 1
- `CHF_model_api.myTensorboard`, 2
- `CHF_model_api.optimizer`, 2
- `CHF_model_api.tools`, 2



# INDEX

## A

`add_hparams()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`add_model()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`add_saved_model()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2

## B

`batch_nrmse()` (in module *CHF\_model\_api.myModel*), 1

## C

*CHF\_model\_api*  
 module, 3  
*CHF\_model\_api.config*  
 module, 1  
*CHF\_model\_api.myModel*  
 module, 1  
*CHF\_model\_api.myTensorboard*  
 module, 2  
*CHF\_model\_api.optimizer*  
 module, 2  
*CHF\_model\_api.tools*  
 module, 2  
`copy_log()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`copy_logs()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2

## D

*DATA* (*CHF\_model\_api.myModel.MyModel* attribute), 1  
`displayPerf()` (*CHF\_model\_api.myModel.MyModel* module), 1  
*method*, 1

## E

`eraseHparams()` (in module *CHF\_model\_api.tools*), 2  
`extractFromPdf()` (in module *CHF\_model\_api.tools*), 2

## G

`getHparamsSavedModel()` (in module *CHF\_model\_api.tools*), 2

## L

`load_tb()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`loadData()` (in module *CHF\_model\_api.tools*), 3  
`log_Hparams()` (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`log_Hparams_range()`  
 (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2

## M

`make_Hparams_range()`  
 (*CHF\_model\_api.myTensorboard.MyTensorboard* module), 2  
*method*, 2  
`makeDataCopies()` (*CHF\_model\_api.myModel.MyModel* class method), 1  
`makeRealPredictions()`  
 (*CHF\_model\_api.myModel.MyModel* module), 1  
*method*, 1  
*CHF\_model\_api*, 3  
*CHF\_model\_api.config*, 1  
*CHF\_model\_api.myModel*, 1  
*CHF\_model\_api.myTensorboard*, 2  
*CHF\_model\_api.optimizer*, 2  
*CHF\_model\_api.tools*, 2  
*MyModel* (class in *CHF\_model\_api.myModel*), 1  
`myMsle()` (in module *CHF\_model\_api.tools*), 3  
*MyOptimizer* (class in *CHF\_model\_api.optimizer*), 2  
*MyTensorboard* (class in *CHF\_model\_api.myTensorboard*), 2

## N

`nrmse()` (in module *CHF\_model\_api.tools*), 3

## O

`optimize_my_models()`  
*(CHF\_model\_api.optimizer.MyOptimizer  
method), 2*

## P

`plotResult()` *(CHF\_model\_api.myModel.MyModel  
method), 1*  
`plotResults()` *(in module CHF\_model\_api.tools), 3*

## R

`remove_backups()` *(in module CHF\_model\_api.tools),  
3*  
`RemoveDirectoryContent()` *(in module  
CHF\_model\_api.tools), 2*  
`reset_directories()` *(in module  
CHF\_model\_api.tools), 3*

## S

`save()` *(CHF\_model\_api.myModel.MyModel method), 1*  
`saveHparams()` *(in module CHF\_model\_api.tools), 3*  
`set_up_directories()`  
*(CHF\_model\_api.myTensorboard.MyTensorboard  
method), 2*  
`stdMP()` *(in module CHF\_model\_api.tools), 3*

## T

`train()` *(CHF\_model\_api.myModel.MyModel method),  
1*

## U

`utilsNnConfig()` *(in module CHF\_model\_api.tools), 3*

## V

`visualizeNn()` *(in module CHF\_model\_api.tools), 3*  
`vizualize()` *(CHF\_model\_api.myModel.MyModel  
method), 1*