# Usage of DSIPTS

This repo collect some examples related to the use of [dsipts] (https://gitlab.fbk.eu/dsip/dsip_dlresearch/timeseries). Using this repo you can train some DL models for timeseries forecasting using public datasets like Monarch or (six_dataset)[https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxIdTmyr0LXQfvaM9vIy] used for benchmarking timeseries models.

Here we report a complete approach using Hydra that allows to better manage multiple experiments.

This particular repo is structured for working with the six datasets (that are 9 because ethh has different resolution and periods): `['electricity','etth1','etth2','ettm1','ettm2','exchange_rate','illness','traffic','weather']`. The task is to predict `y` using past data and other past covariates for a variable number of steps depending on the paper used as reference.

# Installation

In a pre-generated environment install pytorch and pytorch-lightning (`pip install pytorch-lightning==1.9.4`) then go inside the lib folder and execute:

```
python setup_local.py install --force
```

Alternatively, you can install it from the package registry:

```
pip install --force dsipts --index-url https://dsipts:glpat-98SR11neR7hzxy__SueG@gitlab.fbk.eu/api/v4/projects/4571/packages/pypi
```

# Configuration

- copy the folder `all_six_datasets` inside a data folder (in what follows `/home/agobbi/Projects/ExpTS/data`).
- place yoursel in `bash_examples`
- train the models
- create the folders `csv` and `plots` in the `pathdir` in this case `/home/agobbi/Projects/ExpTS`

# Hydra

In you environment install hydra and the joblib launcher for the paralellization tasks:

```
pip install hydra-core
pip install hydra-joblib-launcher      ## if you have a big gpu
```

```
pip install hydra-submitit-launcher    ## if you are in a slurm envirionment
pip install hydra-optuna-sweeper       ## if you need optuna
```

The script used are: - **train.py** for trainint - **inference.py** for inference - **compare.py** for comparing different models

Hydra is used for composing configuration files. In our case most of the parameter can be reused among the different models and are collected under the general configuration file `config/config.yaml`. In what follows the `weather` dataset is used, and notice that this dataset has a frequency of **10 minutes**. The parameters here are the same described in the `dsitps` documentation but clearly some of them can not be modified since they depend on the selected time series. The configuration files related to this experiment can be found in `config_weather`; a generic config folder contains:

```
config_gpu.yaml              # containing the global configuration usually for gpu or local train, see below one example
config_slurm.yaml            # containing the global configuration for slurm training see below one example
compare.yaml                 # instructions for comparing different models
architecture/                # the folder containing the configurations specific for all the models to test
config_used/                 # this folder will be populated while training the models, and will be used in the comparison phase
```

The config file in the case of the weather dataset is reported and commented below.

```
dataset:
  dataset: 'weather'
  path: '/home/agobbi/Projects/ExpTS/data' ##path to data. In the folder data must be present the folder six_dataset

scheduler_config:
  gamma: 0.1
  step_size: 100

optim_config:
  lr: 0.0005
  weight_decay: 0.01

model_configs:
  past_steps: 16
  future_steps: 16
  quantiles: [0.1,0.5,0.9] ##if you want to use quantile loss, otherwise set it to []
  past_channels : null #dataset dependent  hydra expect you to set it anyway also if it depends on data
  future_channels : null #dataset dependent
  embs: null #dataset dependent
```

```
  out_channels: null #dataset dependent

split_params:
  perc_train: 0.7
  perc_valid: 0.1
  range_train: null
  range_validation: null
  range_test: null
  shift: 0
  starting_point: null
  skip_step: 1
  past_steps: model_configs@past_steps  ##this is a convinient what to reuse previous information!
  future_steps: model_configs@future_steps

train_config:
  dirpath: "/home/agobbi/Projects/ExpTS"
  num_workers: 0
  auto_lr_find: true
  devices: [0]

inference:
  output_path: "/home/agobbi/Projects/ExpTS"
  load_last: true
  batch_size: 200
  num_workers: 4
  set: "validation"
  rescaling: false (sometimes you want to get the errors on normalized datasets)

#since now standard things, these two sessions are the most crucial and useful

defaults:
  - _self_                             # take all this configuration
  - architecture: null                 # and let the use specify the architecture to use (be aware that the filed here is the same
  - override hydra/launcher: joblib  # use joblib for multiprocess allowing parallelization in case of multirun in a gpu/cpu envi
```

```
hydra:
  launcher:
    n_jobs: 2                  # parameters indicate the number of parallel jobs in case of multirun
    batch_size:1               #one worker per job
  output_subdir: null          # do not save any file
  sweeper:
    params:
      architecture: glob(*)    # this is a way to train all the models in the architecure folder
```

If you are using a SLURM cluster:

```
defaults:
  - _self_
  - architecture: null
  - override hydra/launcher: submitit_slurm  ## use slurm launcher

hydra:
  launcher:
    submitit_folder: ${hydra.sweep.dir}/.submitit/%j
    timeout_min: 600
    partition: gpu-V100  ##partition to use REQURED
    mem_gb: 6            ##gb requires      REQURED
    nodes: 1
    gres: gpu:1          ##number of GPU    REQURED
    name: ${hydra.job.name}
    _target_: hydra_plugins.hydra_submitit_launcher.submitit_launcher.SlurmLauncher
    setup:
      - conda activate tt  ##activate the conda environment first!
```

In the `config_weather/architecture` folder there are the selected models that have the following structure:

```
# @package _global_  ##care this must be present!


#the specified parameters below overwrite the default configuration having a more compact representation
model:
  type: 'linear'
```

```yaml
ts:
  name: 'weather'
  version: 1              # if you need to versioning a model
  enrich: ['hour']
  use_covariates: false   # if true all the columns of the dataset will be used as past features


## for more information about models please look at the documentation [here] (https://dsip.pages.fbk.eu/dsip_dlresearch/timeserie
model_configs:
  cat_emb_dim: 32         # dimension of categorical variables
  kernel_size: 5  # kernel size
  sum_emb: true           # if true each embdedding will be summed otherwise stacked
  hidden_size: 256        # hidden size of the fully connected block
  kind: 'linear'          # model type
  dropout_rate: 0.2       # dropout
  use_bn: false           # use or not bn layers in the first layers
  activation: 'selu'      # activation function



train_config:
  batch_size: 128
  max_epochs: 250
  gradient_clip_val: null          # pytorch lightening gradient clipping procedure
  gradient_clip_algorithm: 'norm'  # pytorch lightening gradient clipping procedure
```

Hydra allows us to train a specific model using if you are in a gpu environment

```
python train.py  architecture=linear --config-dir=config_weather --config-name=config_gpu
```

or, if you are in a slurm gpu cluster

```
python train.py  architecture=linear --config-dir=config_weather --config-name=config_slurm -m
```

or a list of models in paralle:

```
python train.py  -m architecture=linear, dlinear --config-dir=config_weather --config-name=config_slurm
```

or all the implemented models:

```
python train.py  -m  --config-dir=config_weather --config-name=config_slurm
```

In case of parallel experiment you should see at display something like:

```
(tt) agobbi@frontend:~/Projects/timeseries/bash_examples$ python train.py --config-dir=config_weather --config-name=config_slurm -m
[2023-05-03 16:46:59,679][HYDRA] Submitit 'slurm' sweep output dir : multirun/2023-05-03/16-46-58
[2023-05-03 16:46:59,680][HYDRA]        #0 : architecture=attention
[2023-05-03 16:46:59,690][HYDRA]        #1 : architecture=d3vae
[2023-05-03 16:46:59,696][HYDRA]        #2 : architecture=dlinear
[2023-05-03 16:46:59,700][HYDRA]        #3 : architecture=gru
[2023-05-03 16:46:59,710][HYDRA]        #4 : architecture=informer
[2023-05-03 16:46:59,715][HYDRA]        #5 : architecture=linear
[2023-05-03 16:46:59,720][HYDRA]        #6 : architecture=lstm
[2023-05-03 16:46:59,733][HYDRA]        #7 : architecture=mymodel
[2023-05-03 16:46:59,738][HYDRA]        #8 : architecture=mymodel_v2
[2023-05-03 16:46:59,742][HYDRA]        #9 : architecture=mymodel_v3
[2023-05-03 16:46:59,747][HYDRA]        #10 : architecture=nlinear
[2023-05-03 16:46:59,757][HYDRA]        #11 : architecture=persistent
```

Hydra will create a folder called `multirun` with all the experiments lauched nested as `date/time/x` where `x` indicates the id of the lauched joib. Inside `date/time/x` there will be a file called `train.log` containing all the information logged by the system and useful for debugging. Moreover, the error message and output messages generated by the `sbatch` slurm command can be found in the `date/time/.submitted/x` folder.

If the row `override hydra/launcher: joblib` is commented the train will be consecutive, otherwise in parallel. In the latter case the output in the terminal will be a mess, please check all is woking fine. In the future the logging will be more efficient.

Once the models are trained, the relative full configurations are saved in `config_used` and can be used for inference or comparison:

```
python compare.py --config-dir=config_weather --config-name=compare
```

where the compare file is:

```
models: 'config_weather'               ## path to the main config folder or list of configuration files
dirpath: "/home/agobbi/Projects/ExpTS" ## where are store the models and where to put the results
set: 'test'                            ## set to test
name: 'prova'
rescaling: false                       ## sometimes want to get the MSE on the scaled data
batch_size: 32                         ## batch size for the dataloader
```

or:

```
python compare.py --config-dir=config_weather --config-name=compare_slurm -m
```

if you are in a SLURM cluster where che `compare_slurm.yaml` file must contain also something like:

```
defaults:
  - _self_
  - override hydra/launcher: submitit_slurm

hydra:
  launcher:
    submitit_folder: ${hydra.sweep.dir}/.submitit/%j
    timeout_min: 600
    partition: gpu-V100
    mem_gb: 6
    nodes: 1
    gres: gpu:1
    name: ${hydra.job.name}
    _target_: hydra_plugins.hydra_submitit_launcher.submitit_launcher.SlurmLauncher
    setup:
      - conda activate tt
```

In the `dirpath` folder `/home/agobbi/Projects/ExpTS/` there are three folder now: `weights` containing the model and the weights, `plots` containing some plots coming from the `compare` script and the `csv` forder containing the files.

A typical example of plot is displayed below and shows the MSE at different lags in the test set for different models:

The loss plot is currenty broken on server, you can reproduce it form the notebook `4- results` (see the notebook section)

# Testing

You can use the `config_test` for testing your models. In this case you can use smaller model with fewer epochs:

```
python train.py -m --config-dir=config_test --config-name=config_gpu
```

# Same model differtent parameters

It is possible also to perform a fine tuning procedure on a specific model, in this case:

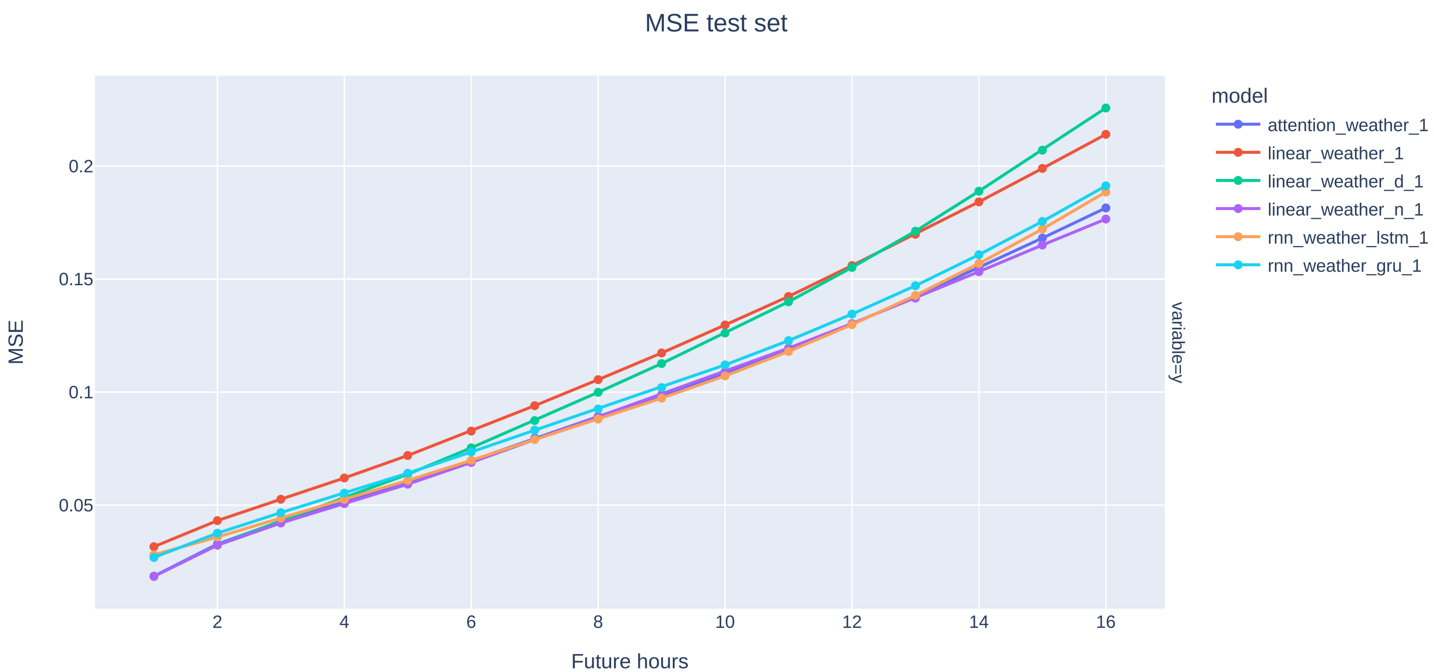


Figure 1: plot

```
python train.py --config-dir=config_weather --config-name=config_slurm -m architecture=mymodel model_configs.hidden_RNN=32,64,128
```

will spawn 3 paralle process trainin the same model with three different values of `hidden_RNN`. In case of multiple parameters to test hydra will generate all the couples of possibilities. This approach can explode very quickly, for this reason it is possible to use `optuna` for exploring the space of the configurations:

```
python train.py --config-dir=config_weather --config-name=config_slurm_optuna -m
```

In this case the configuration file should include the following part:

```
defaults:
  - _self_
  - architecture: null
  - override hydra/launcher: submitit_slurm #SLURM STUFF
  - override hydra/sweeper: optuna          ##OPTUNA SWEEPER

hydra:  ##SLURM STUFFS
  launcher:
    submitit_folder: ${hydra.sweep.dir}/.submitit/%j
    timeout_min: 600
    partition: gpu-V100
    mem_gb: 6
    nodes: 1
    gres: gpu:1
    name: ${hydra.job.name}
    _target_: hydra_plugins.hydra_submitit_launcher.submitit_launcher.SlurmLauncher
    setup:
      - conda activate tt


  output_subdir: null
  sweeper:
    sampler: ##OPTUNA STUFFS
      _target_: optuna.samplers.TPESampler ## see https://optuna.readthedocs.io/en/stable/reference/samplers/index.html for other
      seed: 123
      consider_prior: true
      prior_weight: 1.0
      consider_magic_clip: true
```

```yaml
  consider_endpoints: false
  n_startup_trials: 10
  n_ei_candidates: 24
  multivariate: false
  warn_independent_sampling: true
_target_: hydra_plugins.hydra_optuna_sweeper.optuna_sweeper.OptunaSweeper
direction: minimize
storage: null
study_name: tft
n_trials: 3 ## put the number of maximun trials
n_jobs: 3   ## parallel jobs


params:
  architecture: mymodel                                #architecture you want to finetune
  model_configs.num_layers_RNN: choice(1,2,3)          #parameters you want to explore, categorical
  model_configs.persistence_weight: range(0.1,0.9,0.1) #or continuous
```