***LiquidNeuralNetwork – LNN***

*The LNN Package*

* The LNN package contains a LNN module with two classes: LiquidNeuralNetwork and SequeceLearner.
  + *SequenceLearner:* Internal class used to define the steps the model will follow in the training process. Not intended for outside usage.
  + *LiquidNeuralNetwork:* Model constructed in an ‘sklearnic’ manner. It contains the *‘fit’* and *‘predict’* methods to train and use the model and it also contains a *‘save’* method to store the model parameters.

*How to use the LNN Package?*

* *Importing:*



* *Instantiation:*



* + Where *‘n\_neurons’* refer to the TOTAL number of neurons the model will have. The number of neurons has to satisfy:
  + *‘model\_type’* refers to the cell type used for the neurons where the available types are *‘LTC’* and *‘CfC’* (see *Documentación Técnica NCPs* for more info about the inners of these cells).
* *Data Structure:*
  + Both the features and the target/s must follow the same 3D shape (x, y, z):
    - The x dimension corresponds to the training batches
    - The y dimension corresponds to the observations for the model.
    - The z dimension corresponds to the features/targets

Internally these dimensions are referred as (0, -2, -1)

* *Fitting:*

A black background with white text

Description automatically generated

* + *‘data\_x\_train’* and *‘data\_y\_train’* are the 3D matrices following the previously explained rules for the observations and the targets.
  + *‘epochs’* corresponds to the number of times the model will see the observations
  + *‘clip\_value’* corresponds to the maximum value allowed for the gradient to achieve. If this value is surpassed the numbers are clipped.
  + *‘learning\_rate’* refers to the learning velocity the model has. It multiplies the error made in the prediction to avoid overfitting and non-convergent solutions.
  + *‘n\_jobs’* refers to the number of processes used by the model. If the number of processes surpasses the number of cores the CPU has, no significant performance enhancements will be yielded.
* *Predicting:*



* + *‘data\_x\_test’* is a 3D matrix following the previously explained rules for the observations to be predicted.
* *Saving:*



* + *‘path’* refers to the relative or absolute path where the model will be saved. It can be a string or a Path object from the *pathlib* library.

*Considerations*

* Training tips:
  + One of the main focuses to get accurate predictions is to use around 300ish epochs per target the model has. This does not ensure proper convergence (where one target seems to not be accurate) but it can help.
  + This model uses numerical iterative mathematical methods per neuron to train (instead of a simple linear computation) which extremely extends the training times, but it is compensated by using few neurons. The recommendation is to run the model on a GPU to mitigate this problem. Reducing the number of epochs to train can help but the model will not yield as accurate predictions.
* Making proper predictions:
  + Recall that this model uses *‘liquid time constants’* which means that the model adapts its parameters after each iteration to produce correct results. This translates in a different type of prediction pattern where the temporal point to be predicted needs the previous points to be accurate. Conventional deep learning models are trained and receive a new input never seen buy the model to make a prediction whereas LNNs require the complete time series for accurate results.