
DE LA RECHERCHE À L'INDUSTRIE

# A machine learning technique for dynamic aperture computation

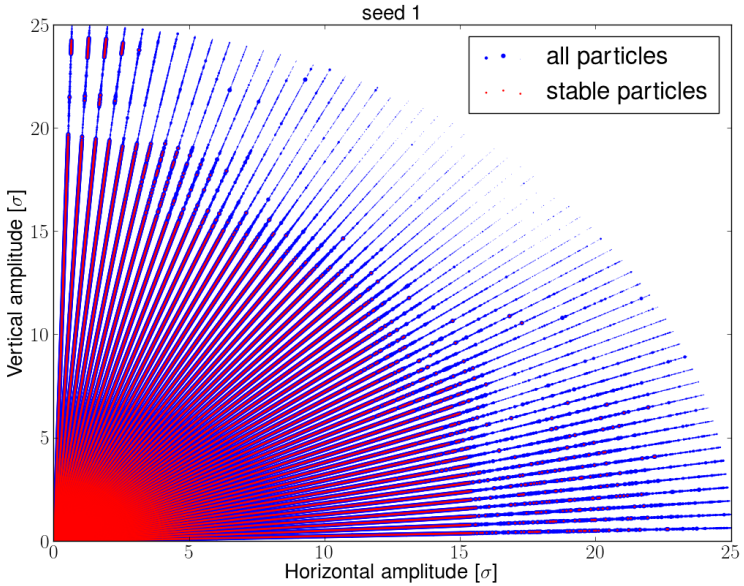
11 Mars 2021

Mehdi Ben Ghali, Barbara Dalena

# Outlook

- Introduction
- I – Minimal Echo-state Network
- II – Modular Echo-state Network
- III – LSTM Echo-state Network
- IV – ANN Readout
- Conclusion

# Motivation




**We are interested in using Machine Learning (ML) in order to model the Dynamic Aperture (DA) of future circular particles accelerators.**

**Dynamic Aperture represents the area of stable motion of a particle in an accelerator. It is currently determined using large HPC tracking simulations, which take from 1 day up to one week or more, depending on the test case and its complexity.**

**Our aim is to test the ability of Recurrent Neural Networks (such as Echo State Networks[1]) to replicate the evolution of non-stationary systems.**

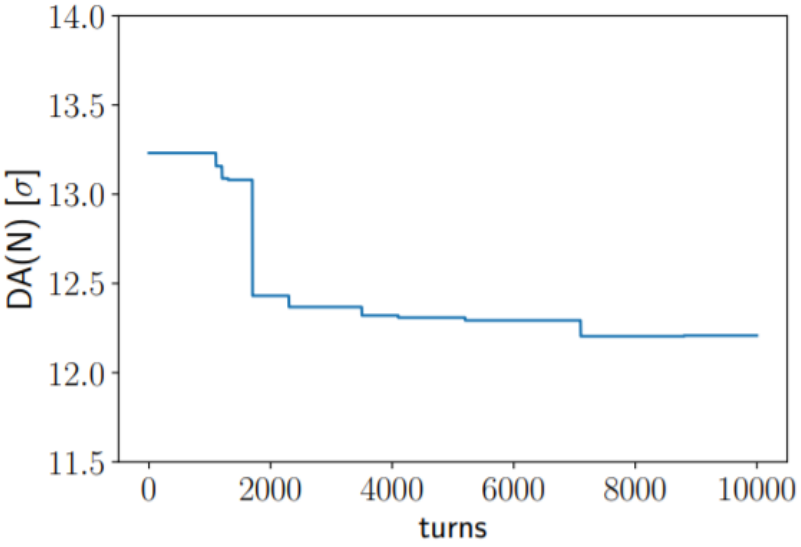
(1) Jaeger H. (2001a) The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science

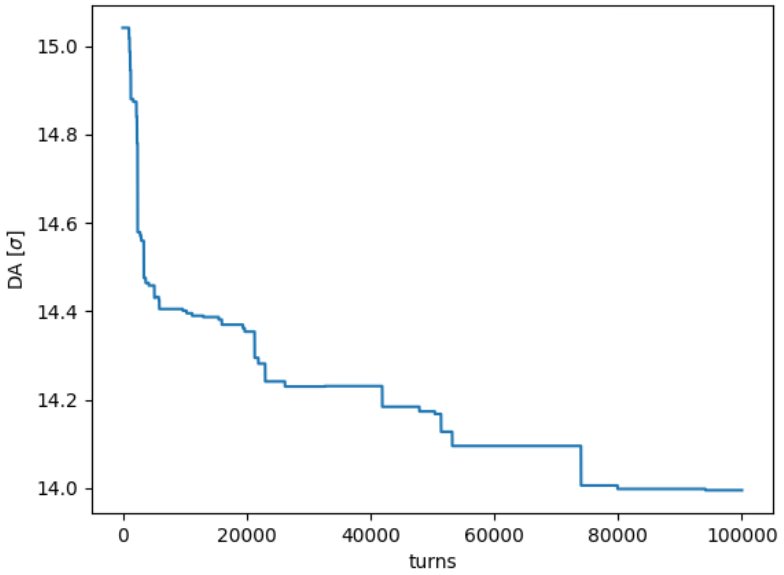
- Datasets **provided by tracking simulations**

- DA computed for different machines configurations (defined by randomly distributed magnetic field errors), called **seeds**

- 60 seeds of each type : with DA values for 10000 and 100000 turns respectively

- We work mainly with the **60 short 10000-turn seeds**, then extend model to long seeds



## Neural Network model

- Previous work and literature. show strong data interpolation performance but weak extrapolation power

- Very computationally-heavy training. Even more as structure is complex

## Echo-state Network

- Using a reservoir structure allows modelling of complex problems with a simple architecture.

- Linear Regression training ensures minimal computational strain.

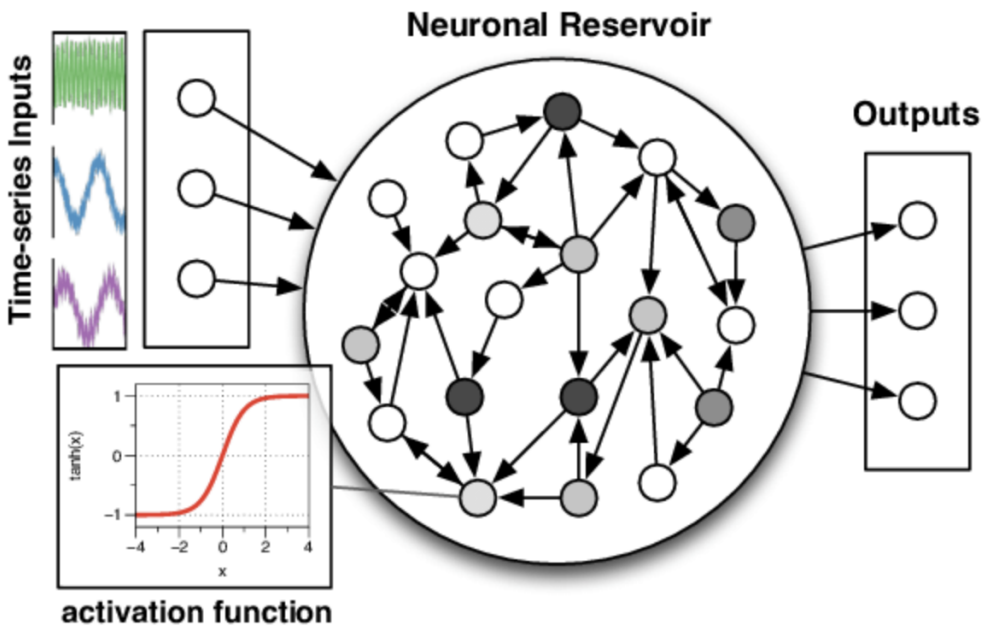
- Recurrent (RNN) structure they have been shown to achieve satisfactory results on extrapolation tasks.

# ANN vs ESN models

-An input is represented by a reservoir “state”, a random fixed representation into a higher space

-Successive reservoir states are then calculated recurrently using previous states and new inputs

-The reservoir output is then obtained using trainable output weights




**Update formula**

$$\mathbf{x} = (1-a) * \mathbf{x} + a * \mathbf{tanh}(W_{in} \cdot \mathbf{u} + W \cdot \mathbf{x})$$

**Output**

$$\mathbf{y} = W_{out} \cdot [1, \mathbf{u}, \mathbf{x}]$$

**Architecture :**

- **Leaky reservoir with a tanh activation function :**

$$\mathbf{x} = (1 - a) * \mathbf{x} + a * \mathbf{tanh}(W_{in} \cdot \ \mathbf{u} \ + W \cdot \mathbf{x})$$

- **Randomly generated weights and trainable output weights**

- **Ridge regression readout** $W_{out} = (\mathbf{Y} \cdot \mathbf{X}^t) \cdot ((\mathbf{X}\mathbf{X}^t)^{-1} + \lambda I))$

**Parameters :**

- **Leaking rate : "a". The coefficient used in the reservoir update. Higher leaking rates result in less previous-state impact and more input impact on reservoir state computation. (]0,1])**

- **Reservoir size : Number of reservoir units**

- **Ridge Coefficient : The Ridge regression coefficient**

**Mean Squared Error (MSE) is used as a prediction accuracy metric**

# I – Minimal Echo-State Network - Training

**Traditionally : Use the first points in a series for the training. The remaining points are predicted recurrently from the last reservoir state.**

**Unsuccessful in our case , so we use a different process :**

- **Weights are randomly initiated, all are fixed except output**

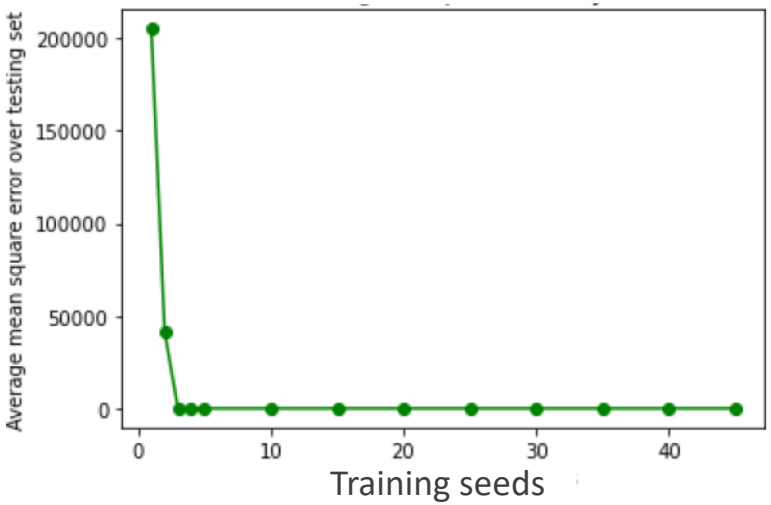
- **Training seeds are reserved**

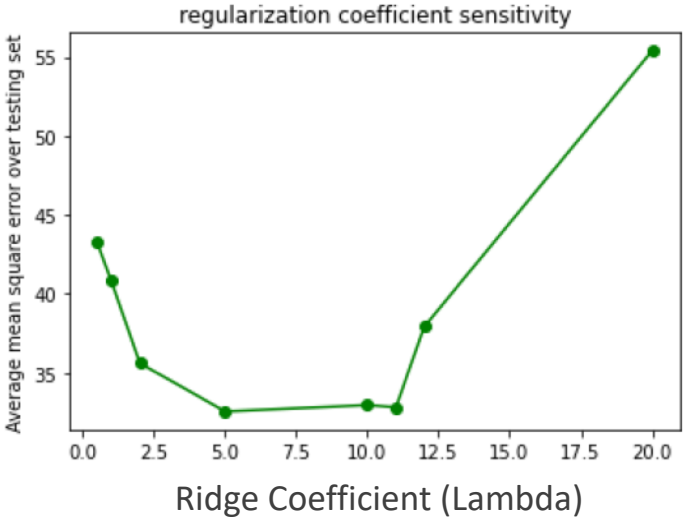
- **For each training seed :**
  - **> Start a new reservoir**
  - **> Updated for all data points**
  - **> Save states in global state matrix**

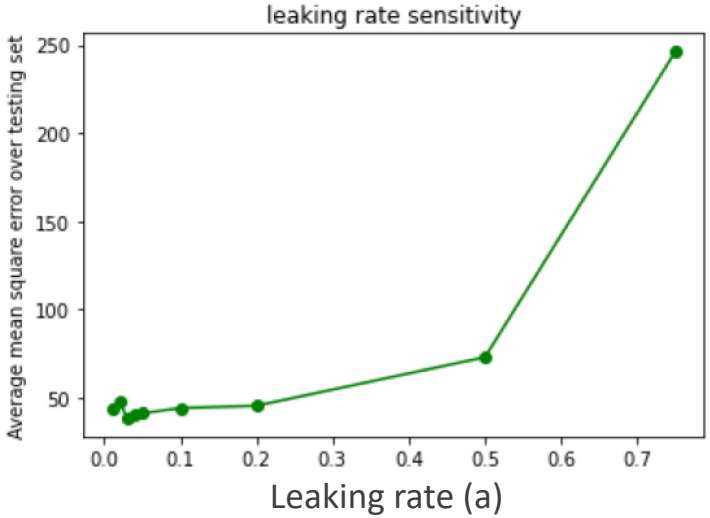
- **Training (once) : Compute output weights using Ridge regression with states as input and data as target**

- **Prediction (for each test seed) :**
  - **> Initiate a new reservoir**
  - **> Updating using the first data points of the seed**
  - **> Run it “generatively” to predict the remaining points**

## Optimized individually



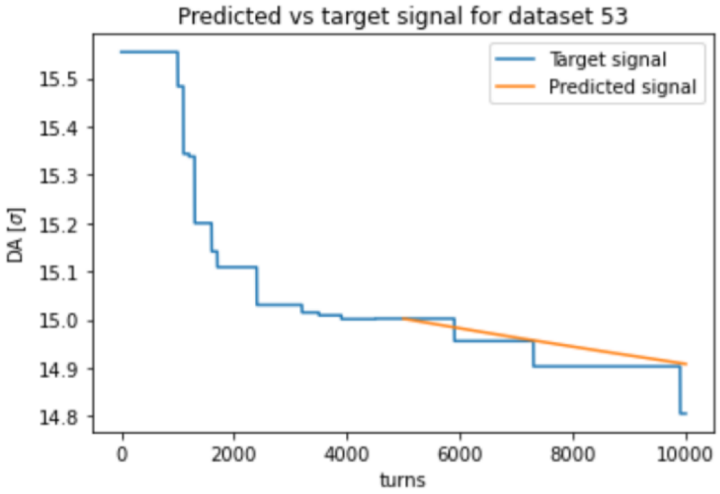



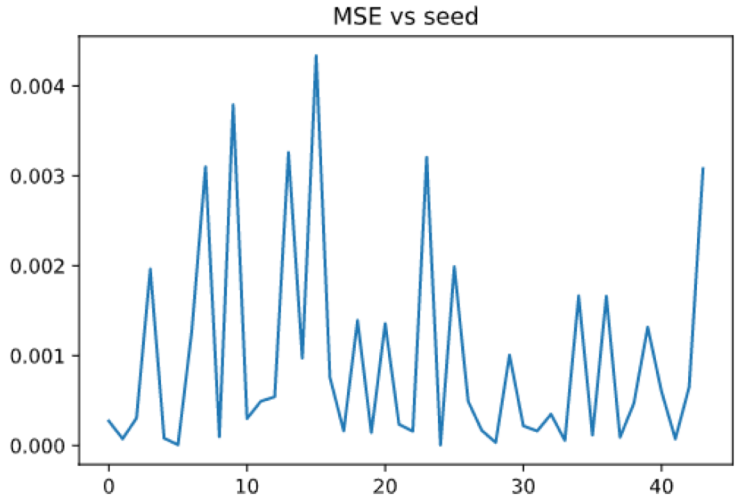




**Final values :**

- **5 training examples**
- **5.0 Ridge coefficient**
- **0.02 leaking rate**

**Using individually optimized hyperparameters,15 training datasets, and 100-unit reservoir :**




**Results are independent of random system seed, and only require a few training examples. However :**

**> Prediction is highly inaccurate on a few seeds**

**> Model does not seem to be capable of nonlinearity**

**> More training examples are needed in practice (best results for 15 vs 5)**

# II – Modular Echo-state network

**In order to more freely customize and improve ESN prediction, development shifted to a more customizable class-based version of the ESN.**

**Note that PyESN, an open-source Echo-State Network python library was tested and confirmed to yield identical results to our model.**

**We do however choose to carry on with the from scratch model as it allows for further customization down the line.**

**-> We then proceed to use this model to study the ways in which the model performance on our data can be improved**

# II – Modular Echo-state network – Hyperparameter optimization

- **Older hyperparameters as well as newer ones :**

- **Connectivity : Defines the sparsity of the reservoir matrix, i.e. the ratio of units to be set to zero (]0,1])**

- **Spectral Radius : The biggest authorized eigenvalue for the reservoir matrix. This must however be smaller than 1 for the reservoir to have echo-state property. (]0,1[)**

- **Regression coefficient : The new readout regression coefficient**

- **Leaking rate**

- **Reservoir size**

**Other parameters such as the random seed, the number of dropped states and reservoir noise level are considered but only show a slight effect on prediction**

## Grid Search/ Metric selection

Grid Search is used to **find out the best hyperparameter combinations** :

> Testing values are provided as a dictionary for each hyperparameter

> All possible combinations are automatically generated

> A different ESN instance is then created, trained and scored for each set of parameters

> The scores and parameter sets are stored and sorted, the best combination is then extracted

Grid search results are also used to determine the best error metric.

- Best penalizes the bad models (high error score)
- Rewards the best predictions (low error score)

The metric with the results best in accordance with this study's needs was the **median absolute error** (MAE)

## Hyperparameter correlation

We then attempt to study the influence of each hyperparameter on the model behaviour, as well as find possible correlations between certain parameters.

In order to do so, we use the grid search method with only two sets of parameters at a time, which makes the search faster and allows us to test entire intervals of values.

Tested parameter pairs are :

- Spectral radius and connectivity
- Leaking rate and tanh leak
- Elastic net regression parameters
- Reservoir size and connectivity

## The tanh leak parameter

**We introduce the tanh leak parameter *t*, which replaces *a* in the tanh term in the equation and will be optimized independently.**

**In literature, the reservoir update current input and previous state weights are correlated. The coefficient are the leaking rate *a* and (1-*a*), respectively.**

$$\mathbf{x} = (1 - a) * \mathbf{x} + a * \mathbf{tanh}(W_{in} \cdot \mathbf{u} + W \cdot \mathbf{x})$$

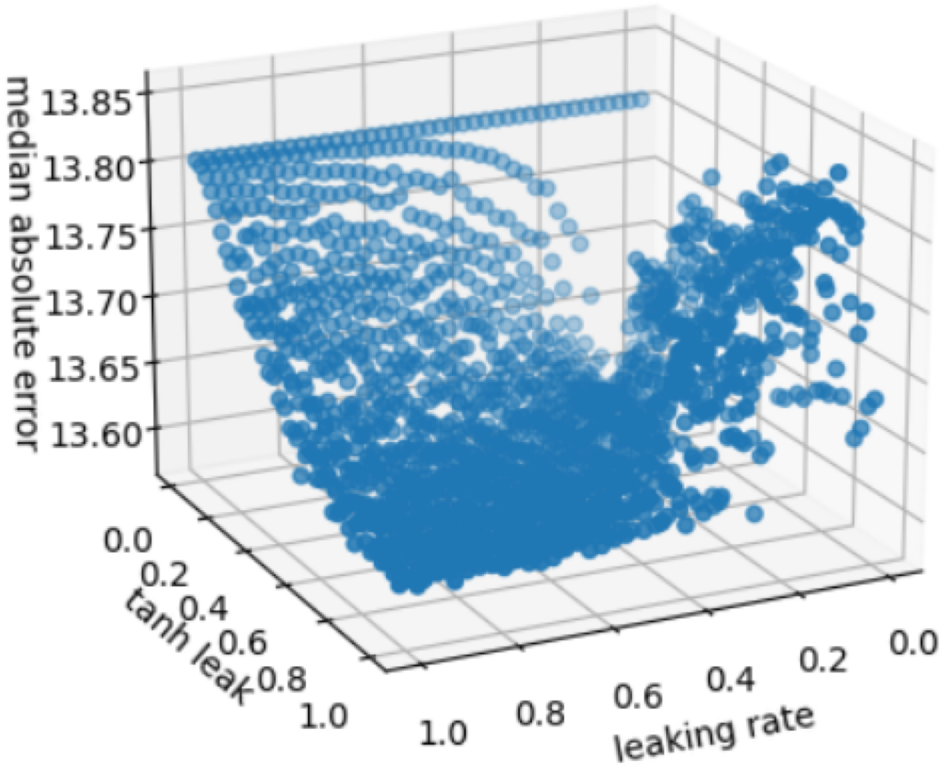
**In our case, empirical testing has shown that in some cases decorrelated terms can yield better results**

**The new reservoir update formula becomes :**

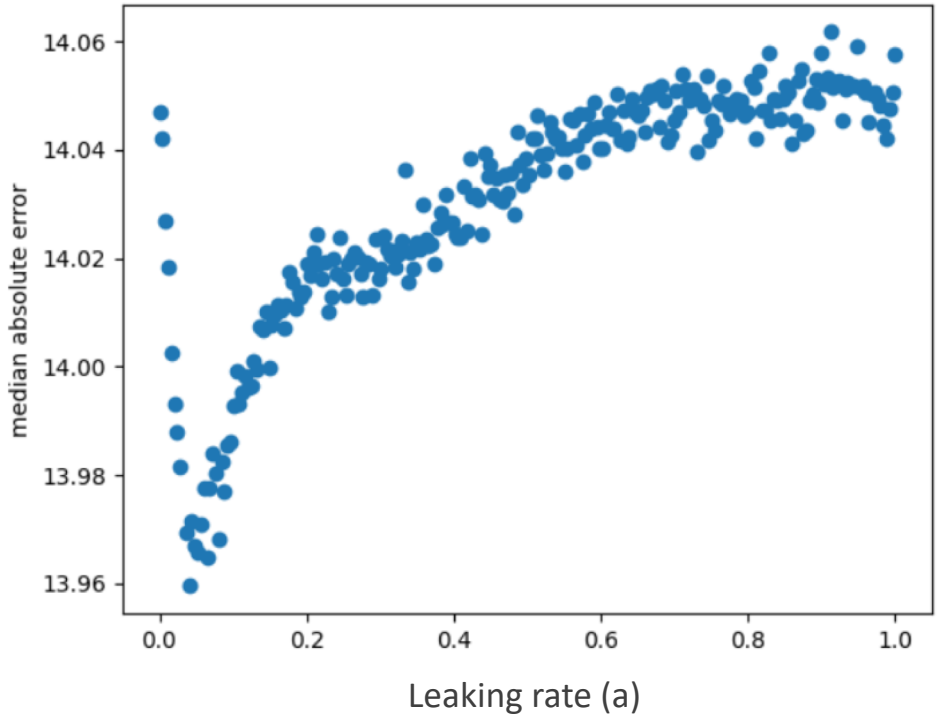
$$\mathbf{x} = (1 - a) * \mathbf{x} + t * \mathbf{tanh}(W_{in} \cdot \mathbf{u} + W \cdot \mathbf{x})$$

## Leaking rate/ tanh leak test

**Testing shows a 1:1 correlation between leak(a) and tanh leak(t), but with a large base**



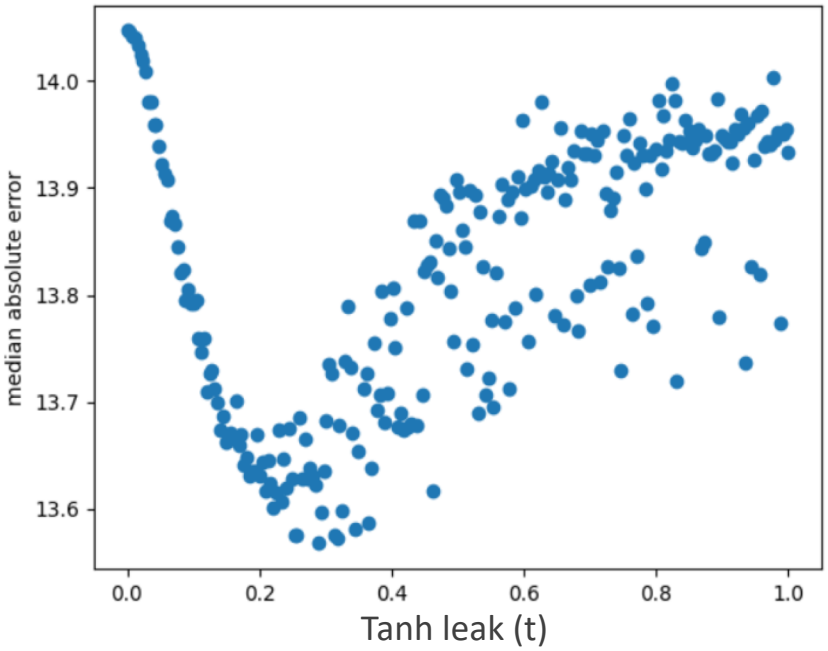
## Leaking rate test (t=0.25)




**We see that increasing the leaking rate very rapidly increases performance up to a point before increasing error again; This is in accordance with tests done on the minimal model.**

**a = 0.04 provides best results**

## Tanh leak test




**Fixed leaking rate (a=0.04)**

**Scores are better further from the t = a point. Error picks up as we go further away**

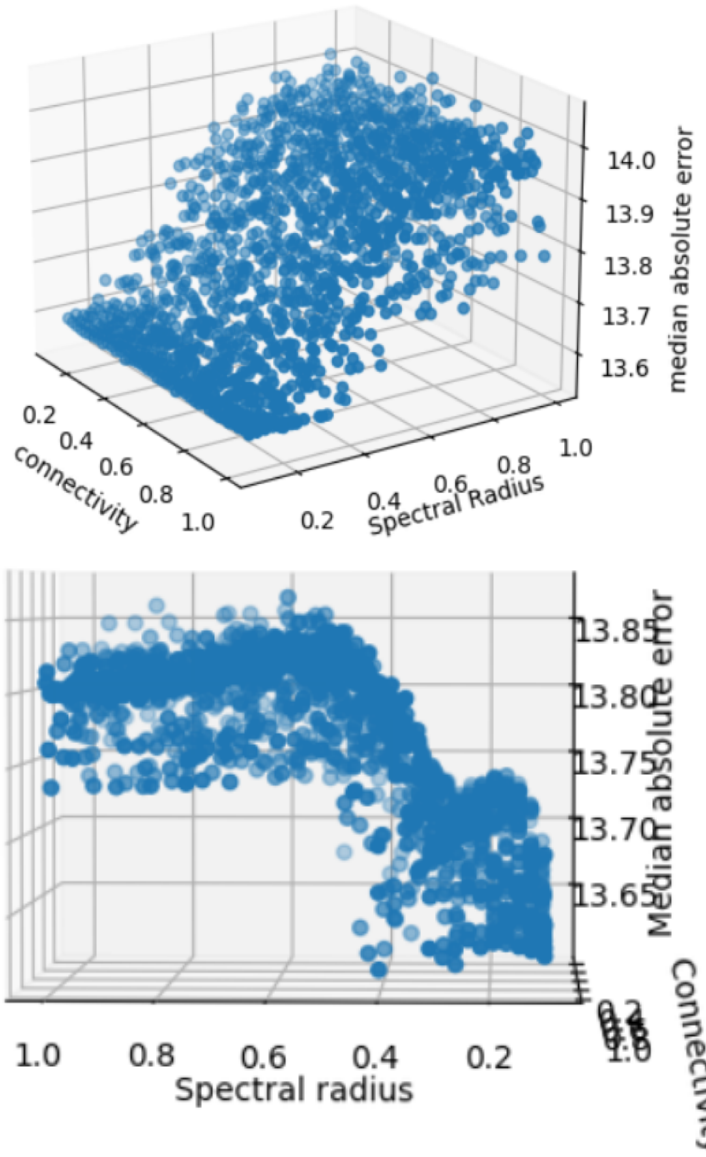
- **Smaller leaking values result in a slower-decaying reservoir which manages to hold information better.**
- **But while a 1 to 1 tanh leak should theoretically provide the lowest error, too small tanh coefficient values result in a stationary reservoir which does not update.**

**=> tanh leak != leaking rates provides better results only in the case of small leaking rates, as tanh leak needs to be big enough for reservoir to update.**

## Spectral radius/connectivity

**Both parameters define reservoir behaviour**

- **connectivity does not seem to have a significant impact on reservoir performance** (mostly spectral radius)
- **Lesser radius values result in better performance**. But decrease is steep and only occurs around 0.4~0.5, with a plateau for higher and lower values.
- **Spectral radius and connectivity do not seem to be correlated.**

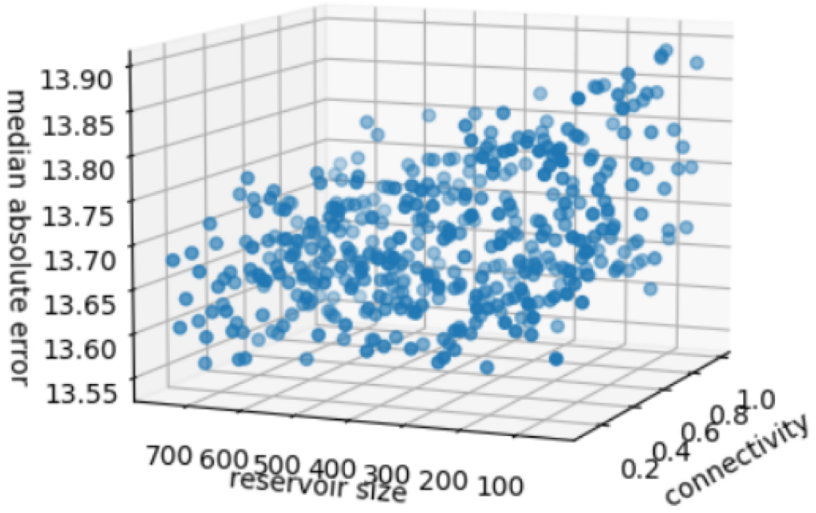


## Connectivity/Reservoir size

**Connectivity, represents the sparsity of the reservoir values.A bigger, more sparse reservoir could be equivalent to a smaller, more connected one.**

- **There does not seem to be a direct correlation between connectivity, reservoir size, and performance. The overall error seems to decrease as reservoir size increases**

- **Note however that reservoir size heavily increases computation time, so we will favor the smaller reservoir sizes with lowest error rather than the bigger ones**



## Elastic net regression

**Ridge regression is traditionally used in the readout layer to fit data. It is defined by the addition of an L2 norm penalty term. Opposed to this is the Lasso regression which uses an L1 norm penalty.**

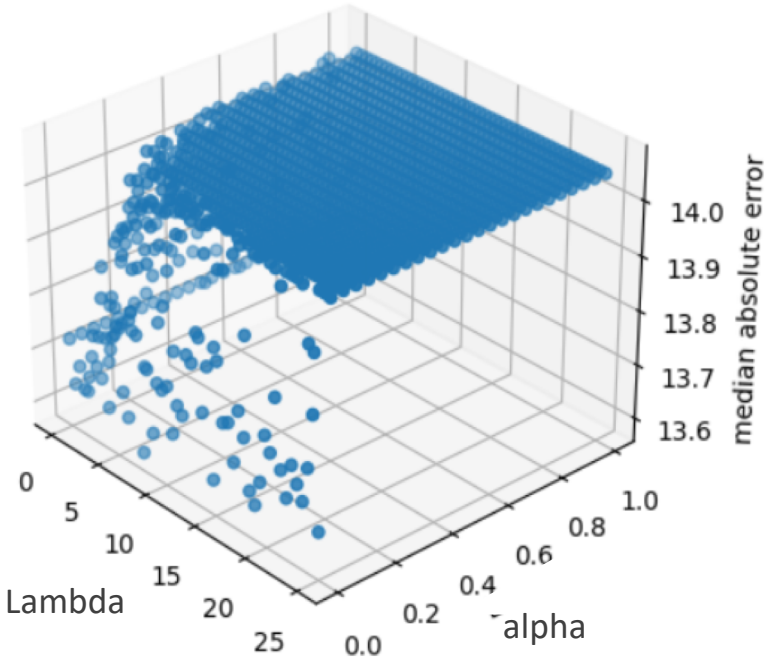
$$\min_{\beta_1,\cdots,\beta_p} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{p} \beta_j z_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

**In order to assess which type of regression is best used for our ESN model, we will be using an elastic net regression which combines both the penalty terms from the previous two. Using the alpha (L1 rate), we can give greater weight to one or the other. The lambda coefficient multiplied by these respective weights replaces both the ridge and lasso coefficient.**

$$\min_{\beta_1,\cdots,\beta_p} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{p} \beta_j z_{ij} \right)^2 + \lambda \underbrace{\left[ \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right]}$$

R(β) = Fonction de pénalité de la régression elasticnet. (α = 0) : Ridge ; (α = 1) Lasso.

## Elastic net regression (continued)




**Almost all of the points outside the alpha = 0 plane have the highest error value which confirms that a Ridge regression is the best fit for linear readout**

# Summary

| Grid search parameters | pairs | all |
|---|---|---|
| Spectral radius | 0.37 | 0.35 |
| Connectivity | 0.95 | 0.8 |
| Leaking rate | 0.04 | 0.25 |
| Tanh leak | 0.35 | 1 |
| Ridge coefficient | 5 | 5 |
| Score | 0.094 | 0.086 |

# II – Modular Echo-state network – Model customization

**New additions :**

- **Teacher-forcing**
- **Data sampling**
- **Fixing random generator seed**
- **Data Scaling**
- **Introducing a polynomial component to the regression**
- **Dropping the first reservoir states**

**The latter two had a minor effect on performance. Data scaling and teacher-forcing improve performance.**

## Data Sampling :

| nturn | tlossmin | nturnavg | mtime |
|---|---|---|---|
| 0.0 10 | 2.0 | 541.0 | 1568383434.45 |
| 1000.0 | 1111.0 | 1055.5 | 1568383434.45 |
| 1100.0 | 1225.0 | 1162.5 | 1568383434.45 |
| 1200.0 | 1355.0 | 1277.5 | 1568383434.45 |
| 1300.0 | 1467.0 | 1383.5 | 1568383434.45 |
| 1400.0 | 1712.0 | 1556.0 | 1568383434.45 |
| 1700.0 | 1899.0 | 1799.5 | 1568383434.45 |
| 1800.0 | 2368.0 | 2084.0 | 1568383434.45 |
| 2300.0 | 2429.0 | 2364.5 | 1568383434.45 |
| 2400.0 | 2512.0 | 2456.0 | 1568383434.45 |
| 2500.0 | 3622.0 | 3061.0 | 1568383434.45 |
| 3600.0 | 4722.0 | 4161.0 | 1568383434.45 |
| 4100.0 | 4747.0 | 4423.5 | 1568383434.45 |
| 4700.0 | 4850.0 | 4775.0 | 1568383434.45 |
| 4800.0 | 6042.0 | 5421.0 | 1568383434.45 |
| 6000.0 | 6914.0 | 6457.0 | 1568383434.45 |
| 6900.0 | 8277.0 | 7588.5 | 1568383434.45 |
| 8000.0 | 9566.0 | 8783.0 | 1568383434.45 |
| 9500.0 | 10000.0 | 9750.0 | 1568383434.45 |
| 9900.0 | 10000.0 | 9950.0 | 1568383434.45 |

- **Data values are only available for a few number of turns values**
- **We get values for the remaining turns through forward filling**
- **Turn values on which we haves samples are multiples of 100**

**> We sample down the data after filling, using a step of 100. Computation time improvement with no information loss**

*Future results are presented using the downsampled turn values. So we will be working with a 100-point long dataset when using the short seeds and a 1000-point long dataset when using long seeds

## Data scaling

**Two main test cases :**

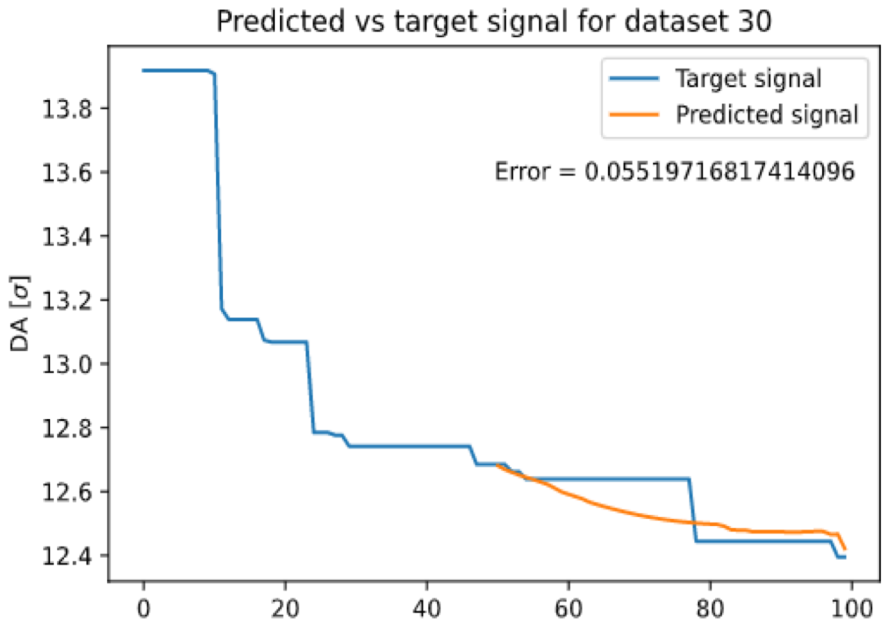
- **Standardization (normalization)** : mean removal and variance scaling, i.e. obtaining data with a mean of 0 and a variance of 1. This is done using the scikit-learn StandardScaler

- **Scaling** : scaling the data to a certain interval. More specifically, we try to bring close to the reservoir coefficients. We tried both the scikitlearn MinMaxScaler and MaxAbsScaler methods. Testing for the following intervals : [-1,1] (Win range), [0,1], [-0.5,0.5] (reservoir range)

**Which resulted in the following error averages after rescaling the output :**

- MinMax scaler
  - In [0,1] : 0.0626 MAE error
  - In [-1,1] : 0.0483 MAE => **best**
  - In [-0.5,0.5] : 0.0779 MAE
- MaxAbs scaler : 0.0583 MAE
- Standard Scaler : 0.065 MAE
- No scaling : 0.086 MAE

**The model has an improved performance compared to the minimal**




**With the following parameters :**

- a = 0.25, t =1
- Connectivity = 0.8
- Spectral radius = 0.35
- 250-unit reservoir
- [-1,1] min max scaling
- teacher forcing
- Ridge regression, 5.0 coefficient
- 20 training seeds, 30 test seeds

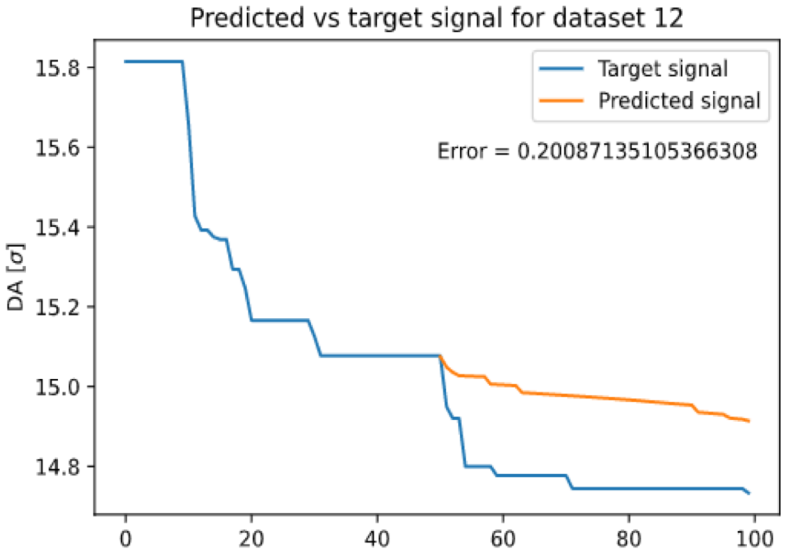
- **Capable of yielding non-linear solutions.**
- **Good average error scores (0.048) (within the error margin allowed for DA applications)**

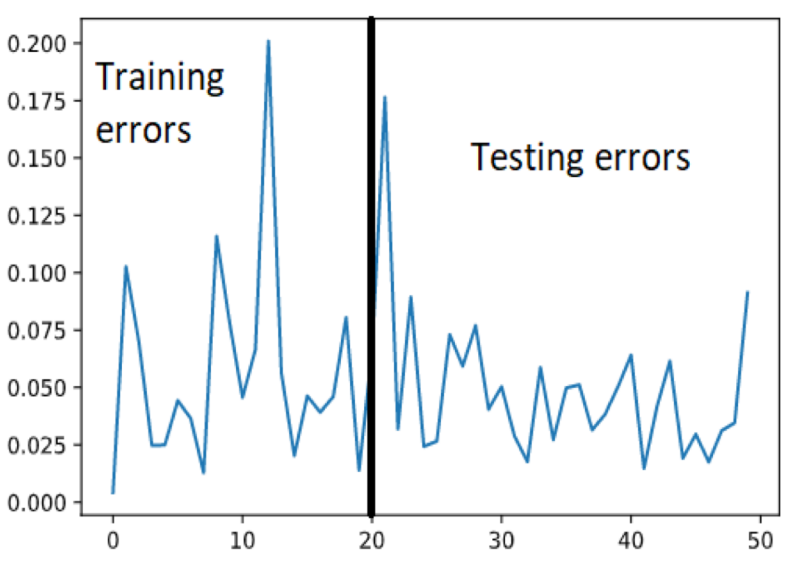
# II – Modular Echo-state network – Shortcomings

-The **best performance is obtained not using the parameters in accordance with manual analysis results, but the best grid search result.**

- **Some strong outliers have predictions that are out of target range and make model unreliable for real use cases.**

- **Comparable training and testing error**

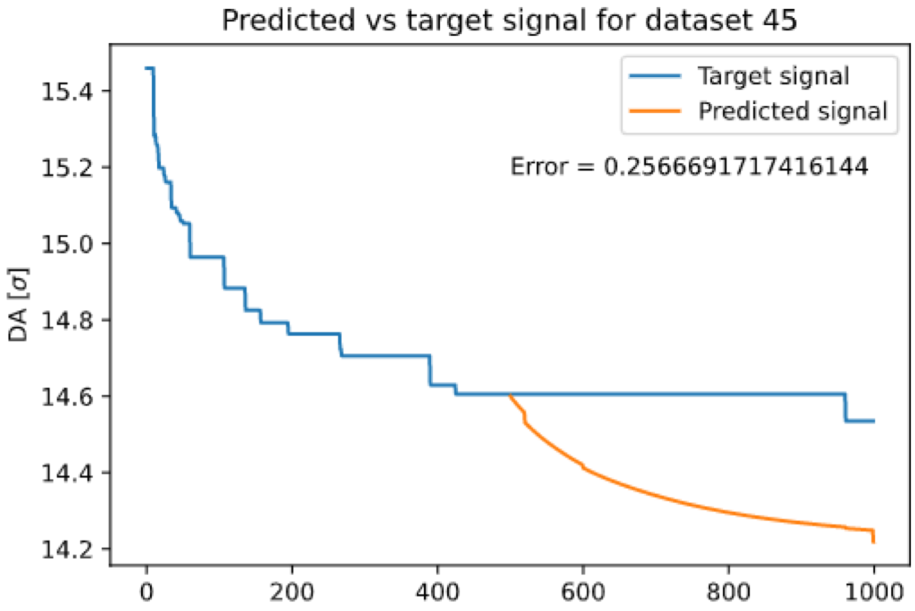


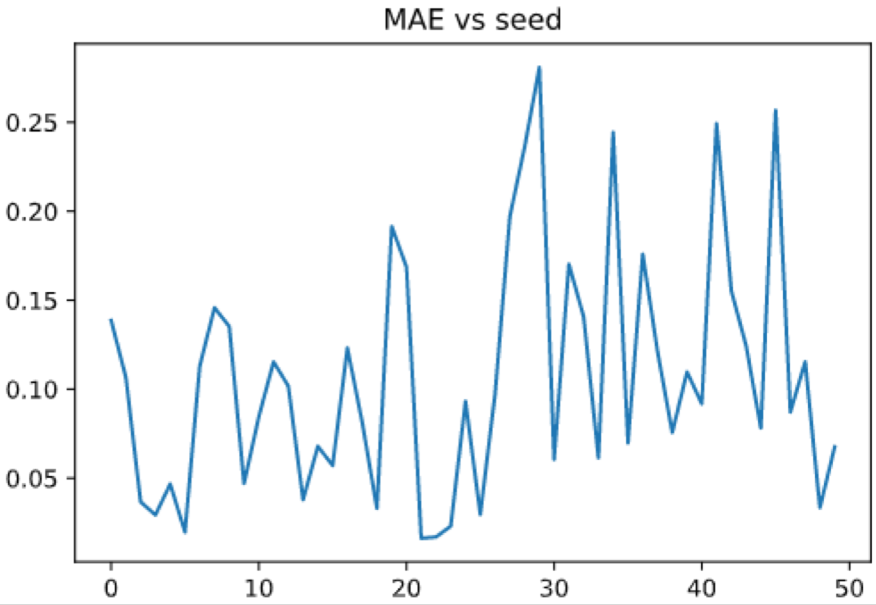



**Performance is worse when we run model with long seeds**

- Higher error values

- Many of the seeds diverge from real values

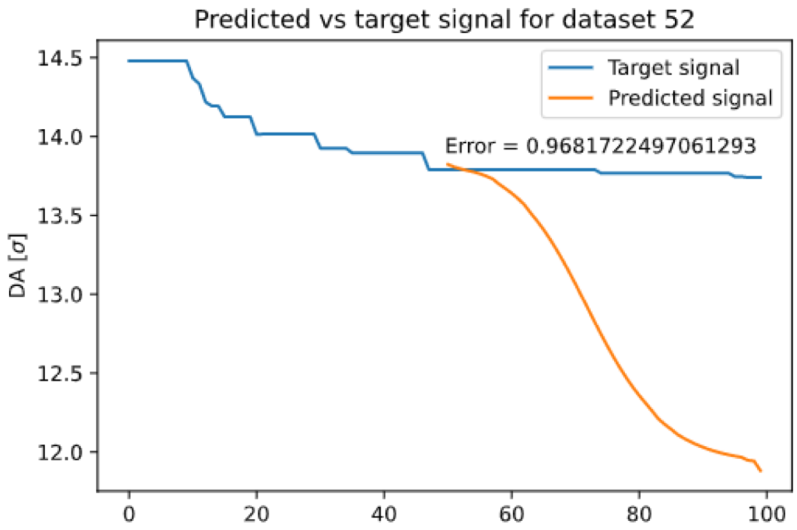


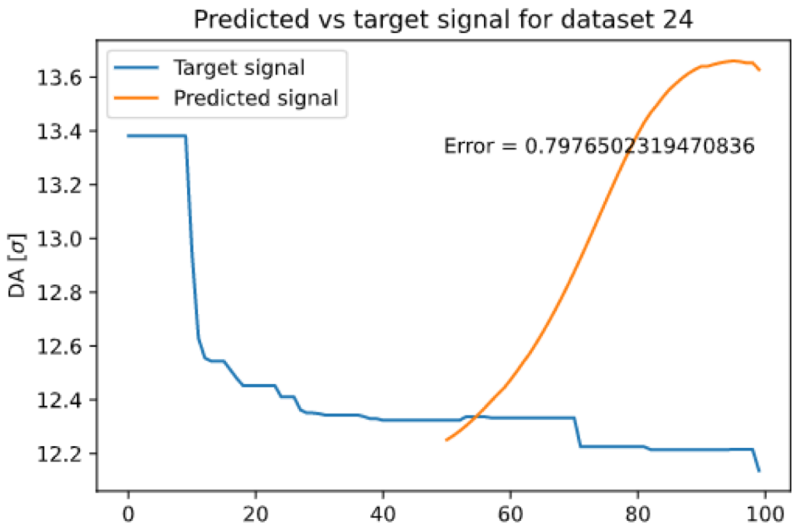



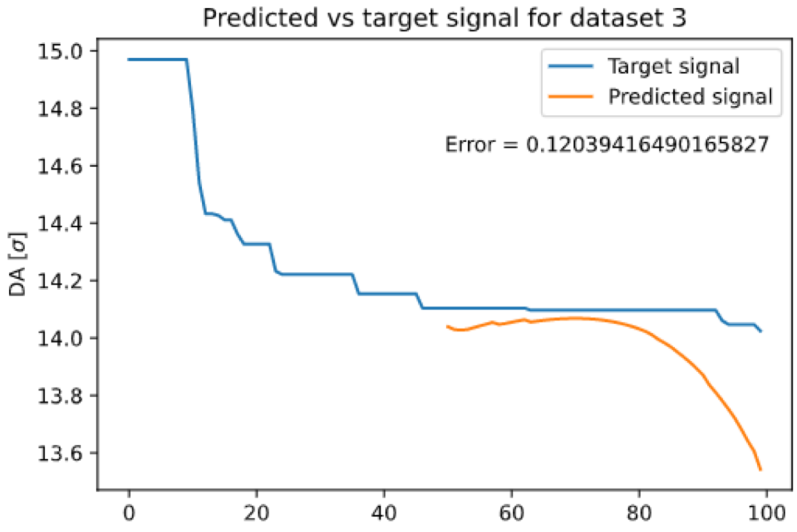

**With a 20/30 train/test split :**

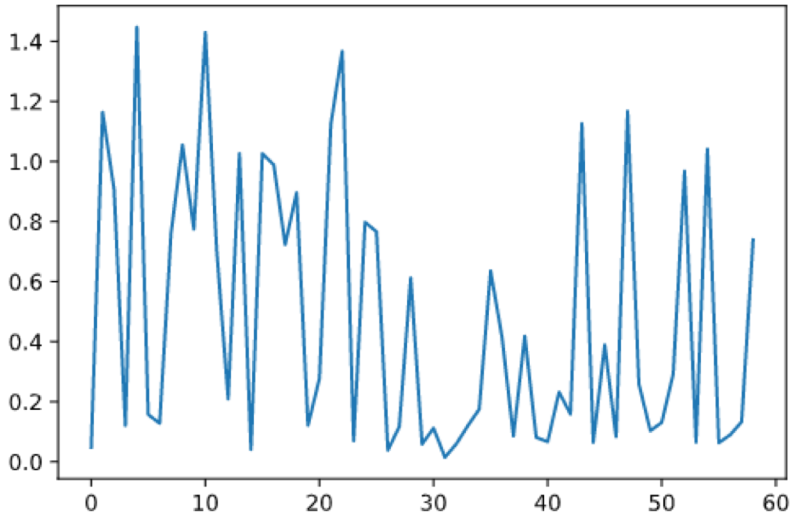
- **Less error on training data (overfitting)**

# II – Modular Echo-state network – No training seeds



# III – LSTM Echo-state network

- A type of recurrent network the that keeps track of prediction history using a "memory"

- In a recent paper[1], a similar structure using reservoirs to track the long-term and short-term history of the model shown great results in predicting time series data.

We reproduce the principle with the following architecture :

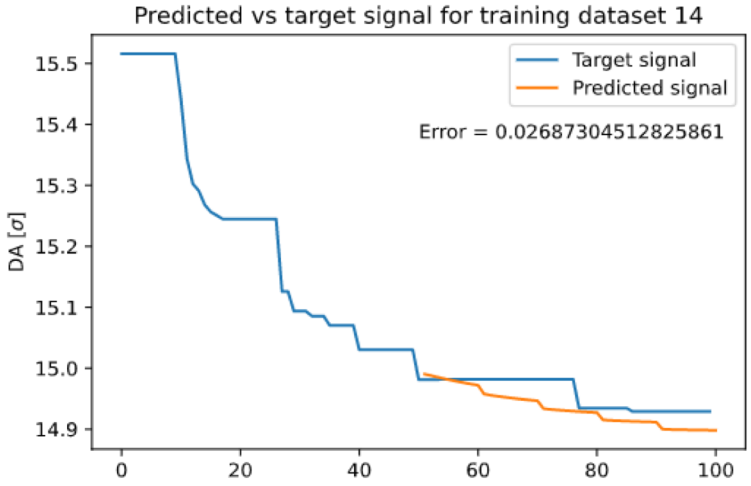
> A short-term reservoir : At one reservoir step, initiated using past n inputs and updated n times before being used to compute current state

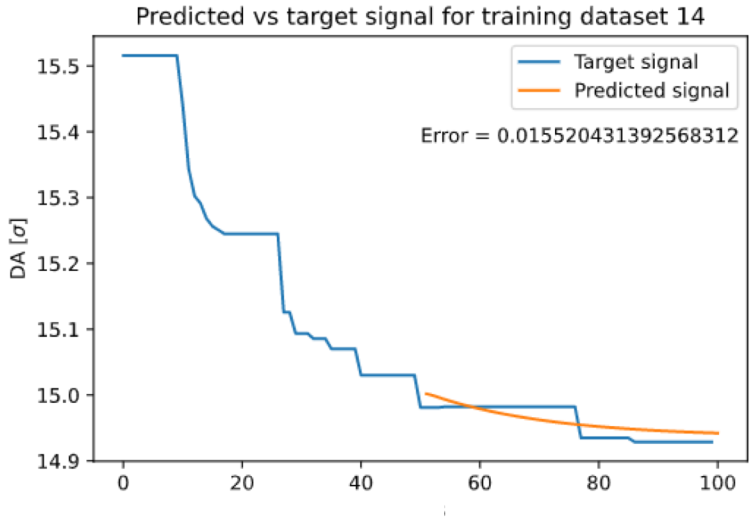
> A long-term reservoir : Only updated every few steps using the last long state and output

> A regular reservoir

States are then stacked, and used for training and prediction

(1) K. Zheng, B. Qian, S. Li, Y. Xiao, W. Zhuang and Q. Ma, "Long-Short Term Echo State Network for Time Series Prediction," in *IEEE Access*, vol. 8, pp. 91961-91974, 2020, doi: 10.1109/ACCESS.2020.2994773.




**With two identical setups, we can compare error :**

- **With the long reservoir**

```
Training error mean : 0.044909832446088975
Testing error mean : 0.060053022464241 25
```

- **Without the long reservoir**

```
Training error mean : 0.04905585877024869
Testing error mean : 0.03884261168064413
```

**We do not use a long-term reservoir in the following tests.**

**Given two identical configurations, unscaled data yields better results :**

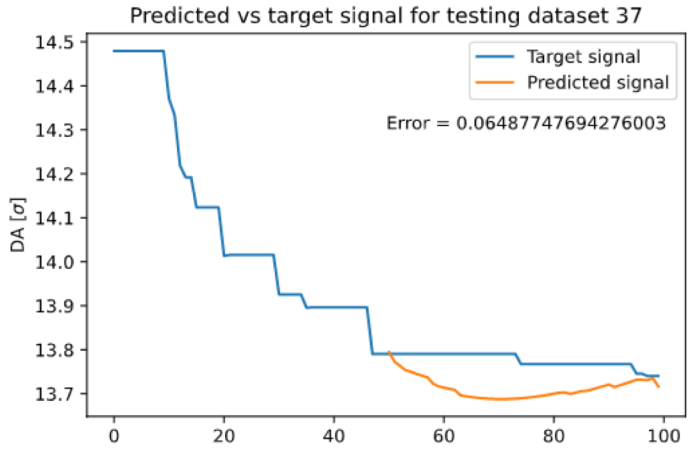
- **Error scores are comparable with unscaled data**
- **Scaled data predictions are “messier”, unscaled predictions are linear**

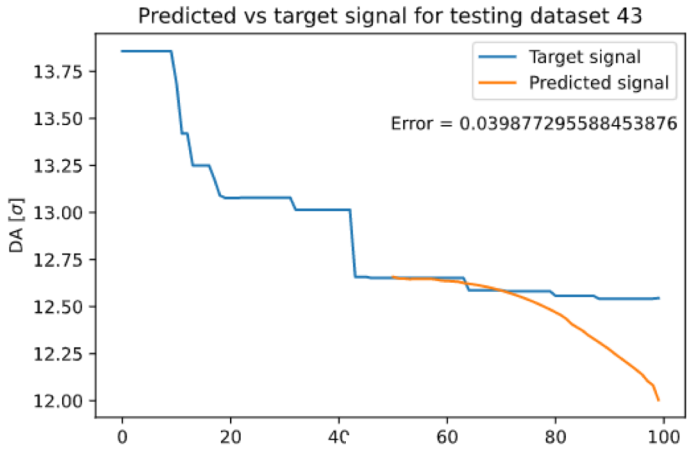
```
Training error mean : 0.06509996024506286
Testing error mean : 0.05385561176820641
```

Scaled data

```
Training error mean : 0.07720343009445661
Testing error mean : 0.05443118975288442
```

Unscaled data




**Tests made using the same scaling as the previous model, a different scaling could work better**

# III – LSTM Echo-state network – Hyperparameters

- **Newer hyperparameters considered :**

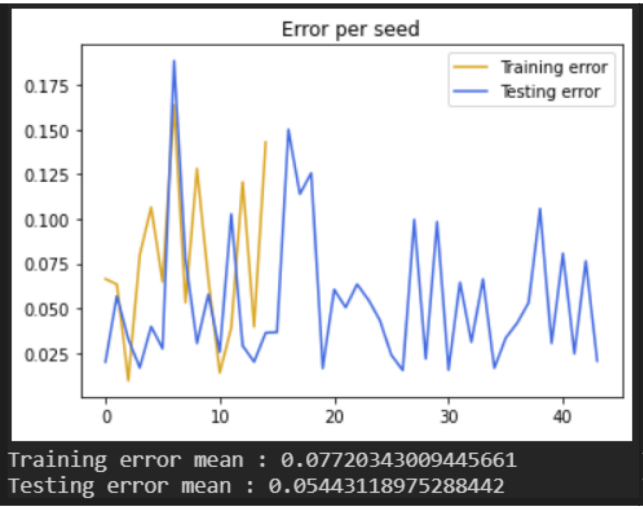
- **Short-term memory : The number of past steps to run through short-term reservoir. 15 works best.**

- **Reservoir weights : Predictions explode when weights are changed. This has not been explained. Weights set to 1 and long-term weight set to 0.**

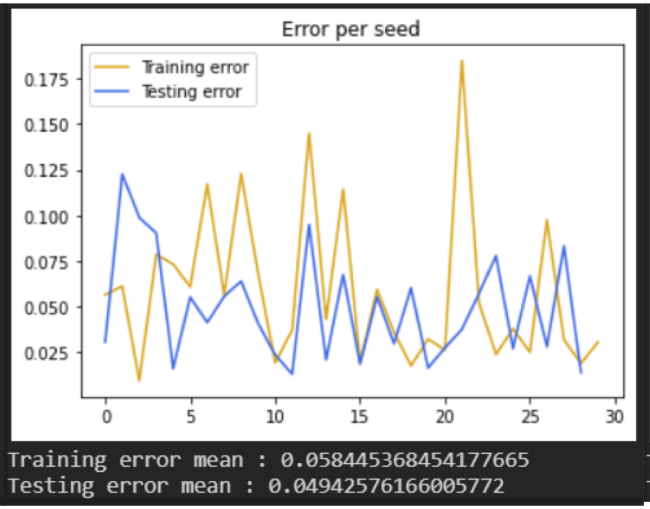
- **Reservoir sizes : Only same-size reservoirs have been tested. Using a slightly bigger reservoir (300-unit)**

**For previous hyperparameters, model works best with common reservoir hyperparameters for both reservoirs set to previous best values**

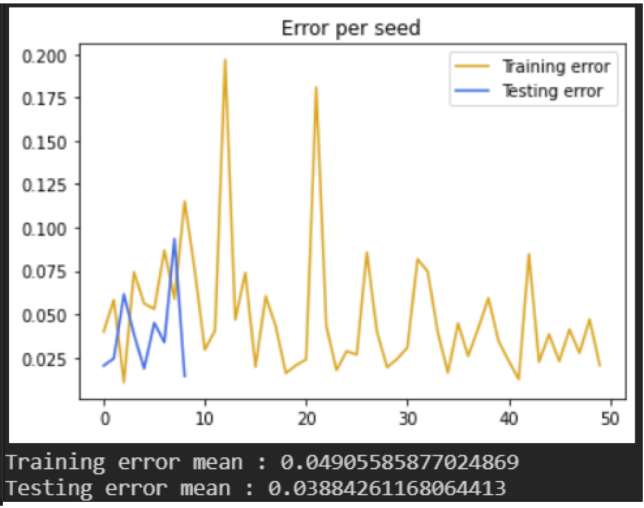
**We try feeding more training examples to the model**




(15:44) split




(30:29) split

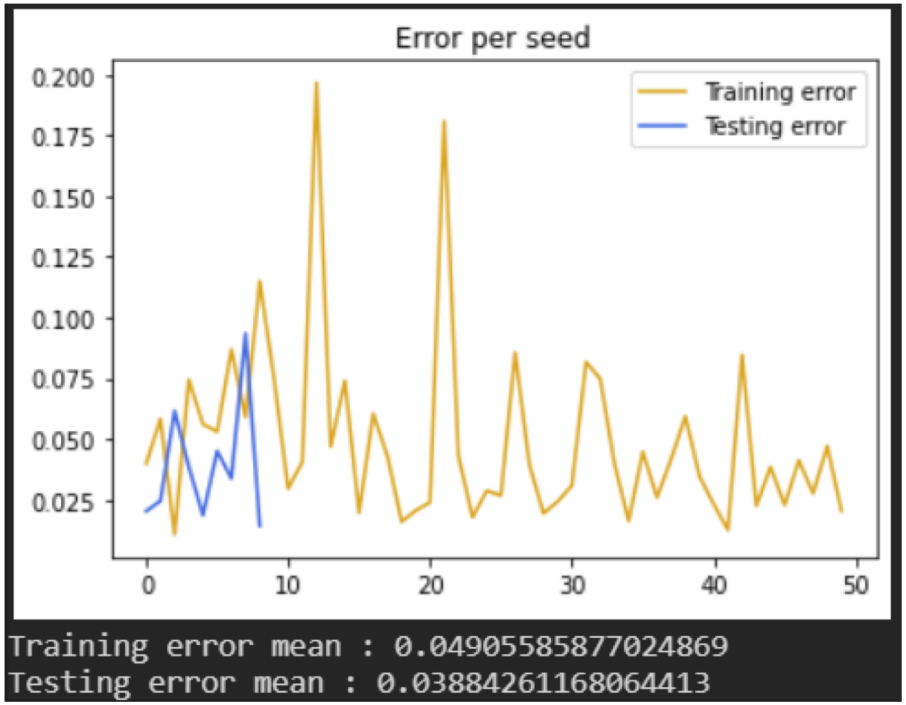



(50:9) split

- **Model responds very well to bigger training datasets**

- **But performance does not improve as much on outliers**

**-This is to be considered for future applications where we might have access to larger sets of data.**

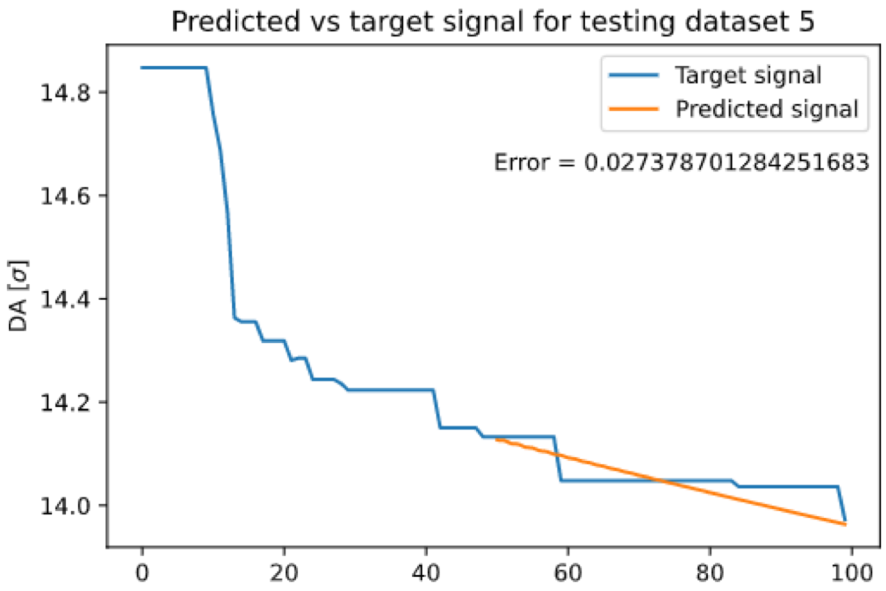



**Parameters :**

- a = 0.25, t =1
- Connectivity = 0.8
- Spectral radius = 0.35
- 300-unit reservoir
- No scaling
- teacher forcing
- Ridge regression, 5.0 coefficient
- Short-term memory = 15

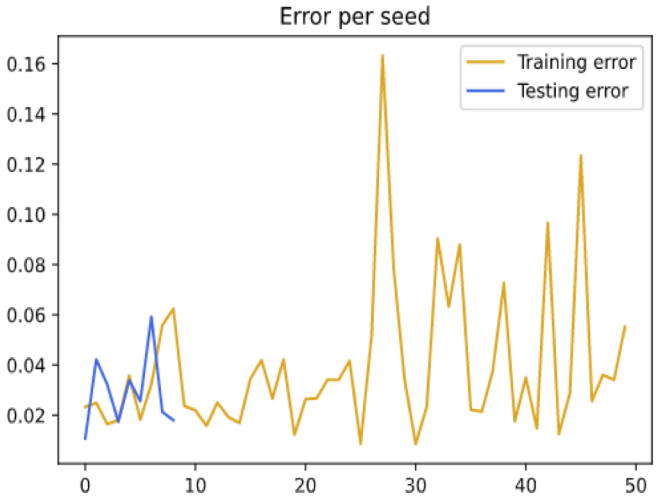
**Data split :** 50 training, 9 testing

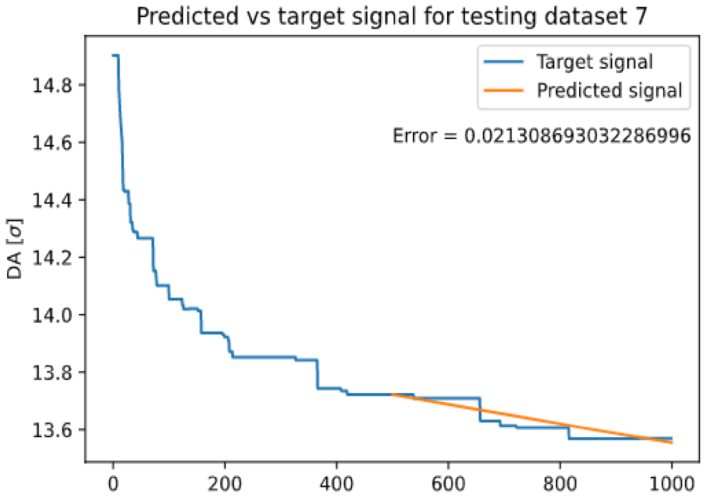
- **Great error scores (0.038 testing MAE)**
- **Most trainable model so far**
- **Model with the least divergent seeds (this is however to be double-checked with more test examples)**




- Model almost only results in linear predictions.
- More training than test error.
- To achieve better performance than the regular reservoir model, we need a lot of training examples (50 seeds). A 50-50 split results in comparable performance to previous model, only linear.
- This means we use a lot of data to make only a few predictions.

# III – LSTM Echo-state network – Prediction on long seeds



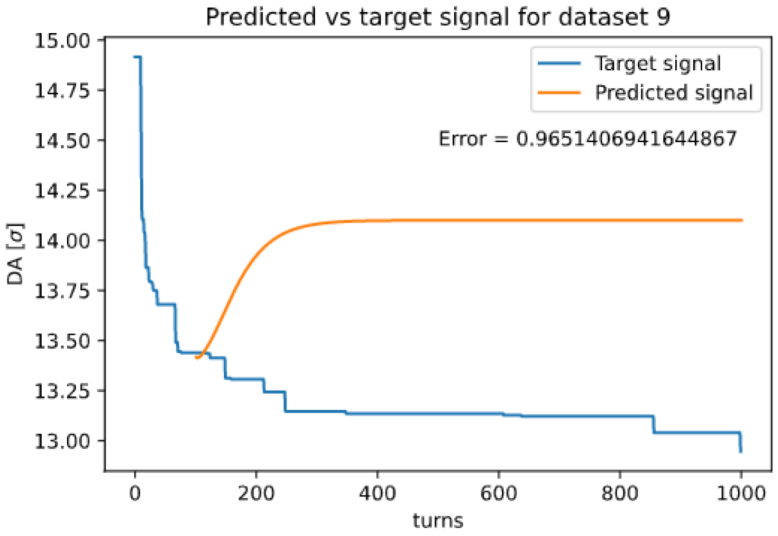

- The **best performance on long seeds yet**

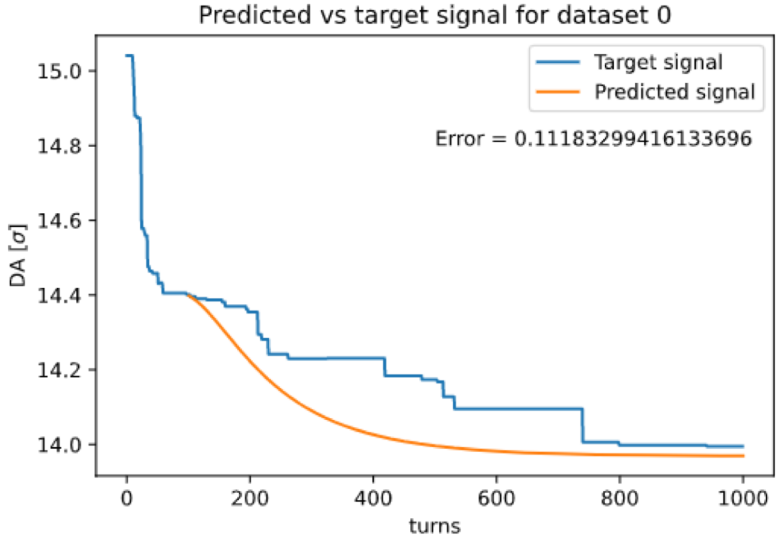
- Even **lower testing error than on short seeds** : this could be explained by the fact that the LSTM model responds well to more data

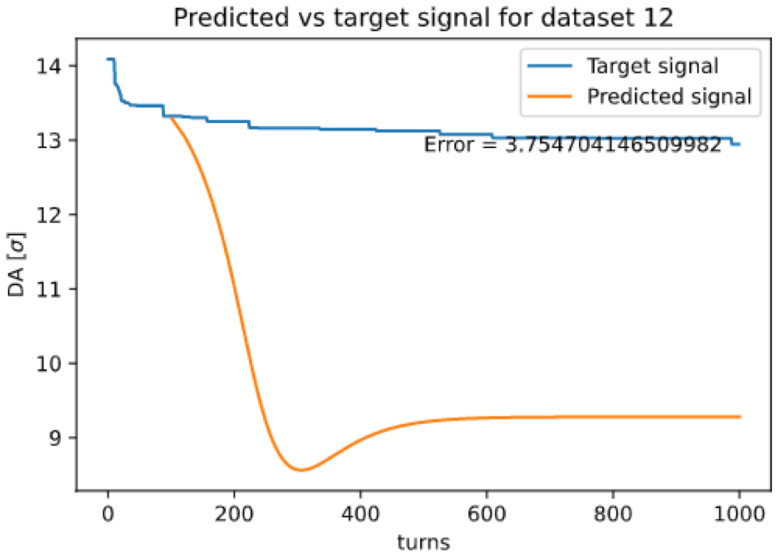
But :

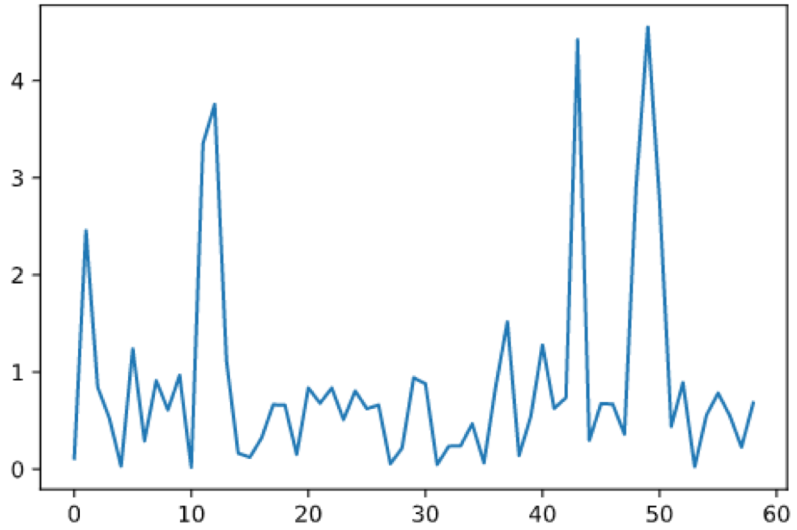
- **Higher training error than shorter seeds** (meaning specific seeds still influence prediction)

# III – LSTM Echo-state network – No training seeds



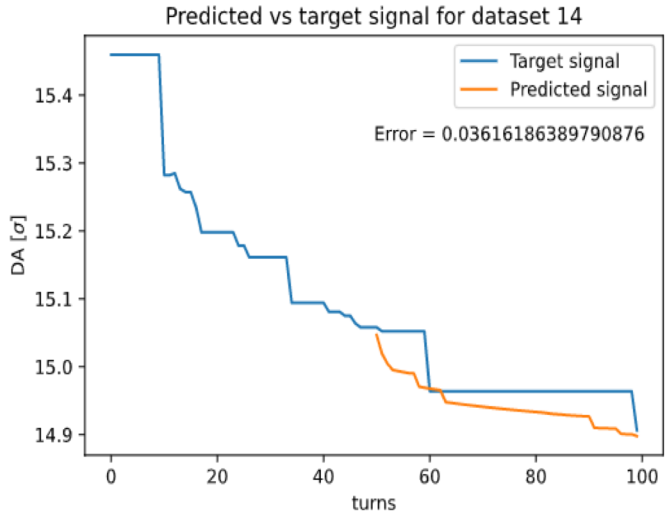



# IV – ANN readout layer

## We try replacing the linear readout with an ANN readout layer

- This has been shown to improve model performance in the case of a classification model[1].

- Preliminary testing has shown that for a worse scores but a stronger ability to reproduce values.




| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 253) | 64009 |
| dense_1 (Dense) | (None, 50) | 12700 |
| dense_2 (Dense) | (None, 10) | 510 |
| dense_3 (Dense) | (None, 1) | 11 |

(1) F. M. Bianchi, S. Scardapane, S. Løkse, R. Jenssen, "*Reservoir computing approaches for representation and classification of multivariate time series*", Transactions on Neural Networks and Learning Systems, IEEE, 2020

## Network structure : adapting to input size

- First tests with the ANN readout layer resulted in constant outputs
- Due to input having too many parameters : network is too small to keep up
- Future networks will scale down problems by using progressively smaller layers

## Training

- We minimize the median absolute error. It is unavailable in the default keras error metrics and is added as a custom loss function.
- Early stopping causes training to stop prematurely beyond a few epochs

## Validation split

- Small data size -> a small validation split results in few validations examples -> Outlier-only validation sets !
- Increasing validation solves this problem but only leaves a few training examples.

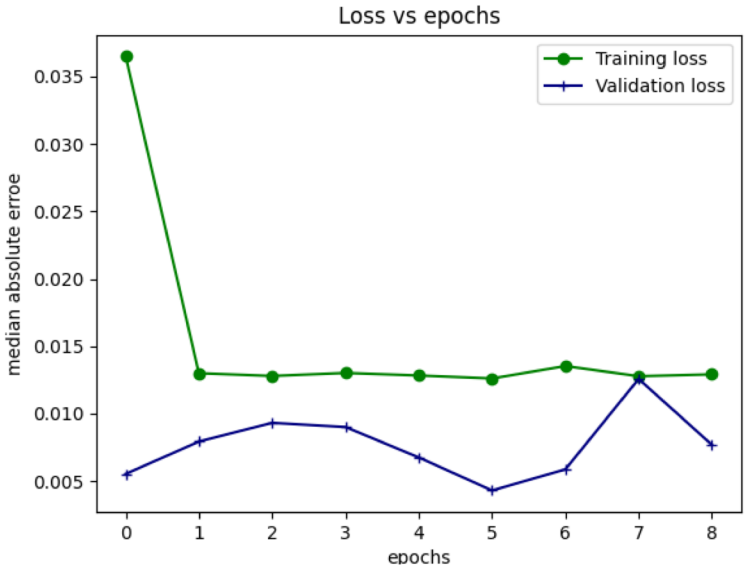
# IV – ANN readout layer – Training progress

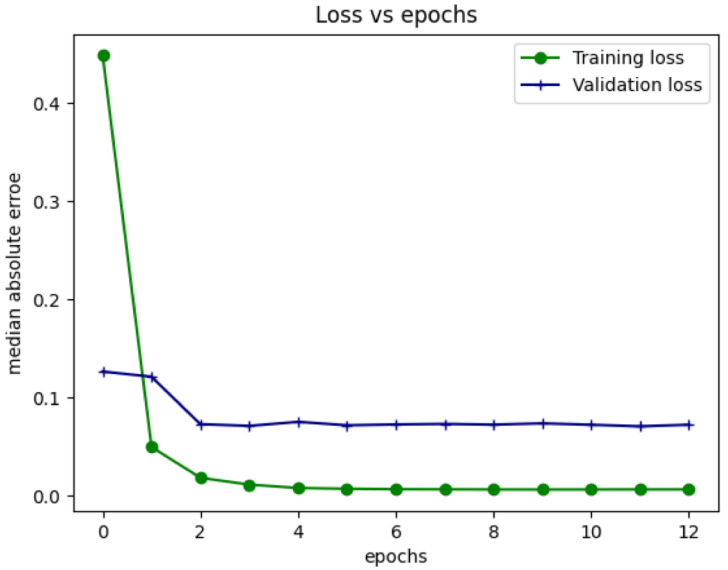
**Currently, the model is unable to progress beyond its initial performance.**

- **For small validation sets, the validation error is almost constantly lower than training error. This could be due to an outlier problem, or a small validation set being non representative.**

- **For bigger validation sets, the model plateaus past a certain early point and both the training and validation errors cease to improve. This is most likely due to not having enough training data**

- **Early stopping does not currently work, and should be addressed**



# Conclusions and perspective

- ESN shows strong potential for dynamic aperture computation.

- DA data scarcity limits technique effectiveness however. A model which does not need as much data as current examples must be achieved.

**In the future :**

- Benchmark model on reference time series data

- Improve ANN readout model

- Try time series decomposition solutions with current models (statsmodels tsa library)

# Merci de votre attention