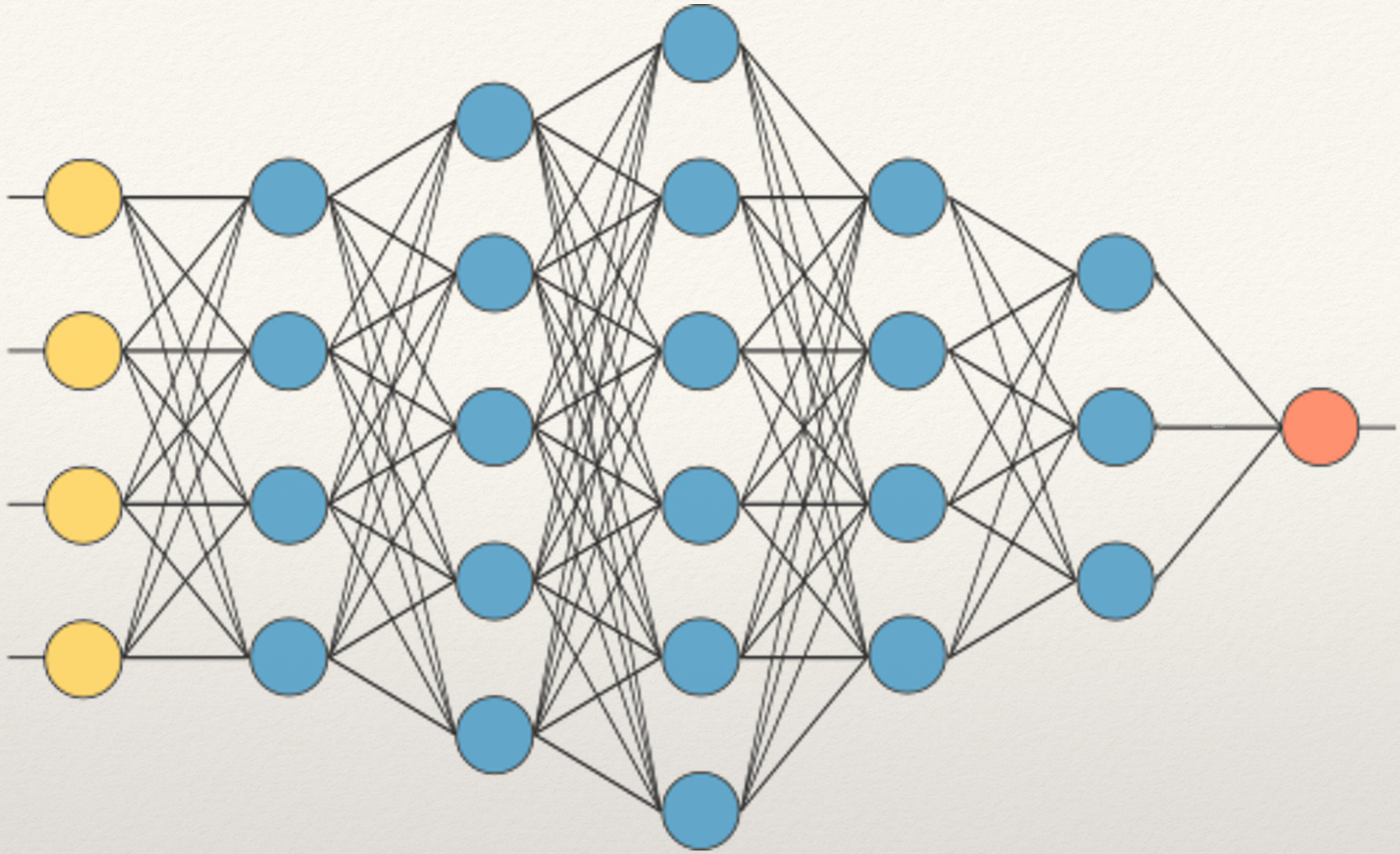
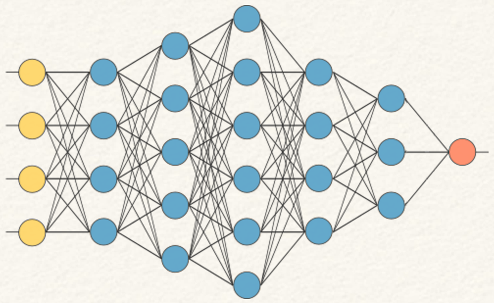


*Optimization Project - Giovanni Pelosi*

# Heating and Electricity Consumption Prediction

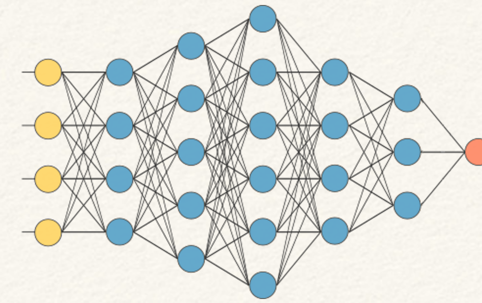
Artificial Neural Network Approach

# Heating and Electricity Consumption Prediction

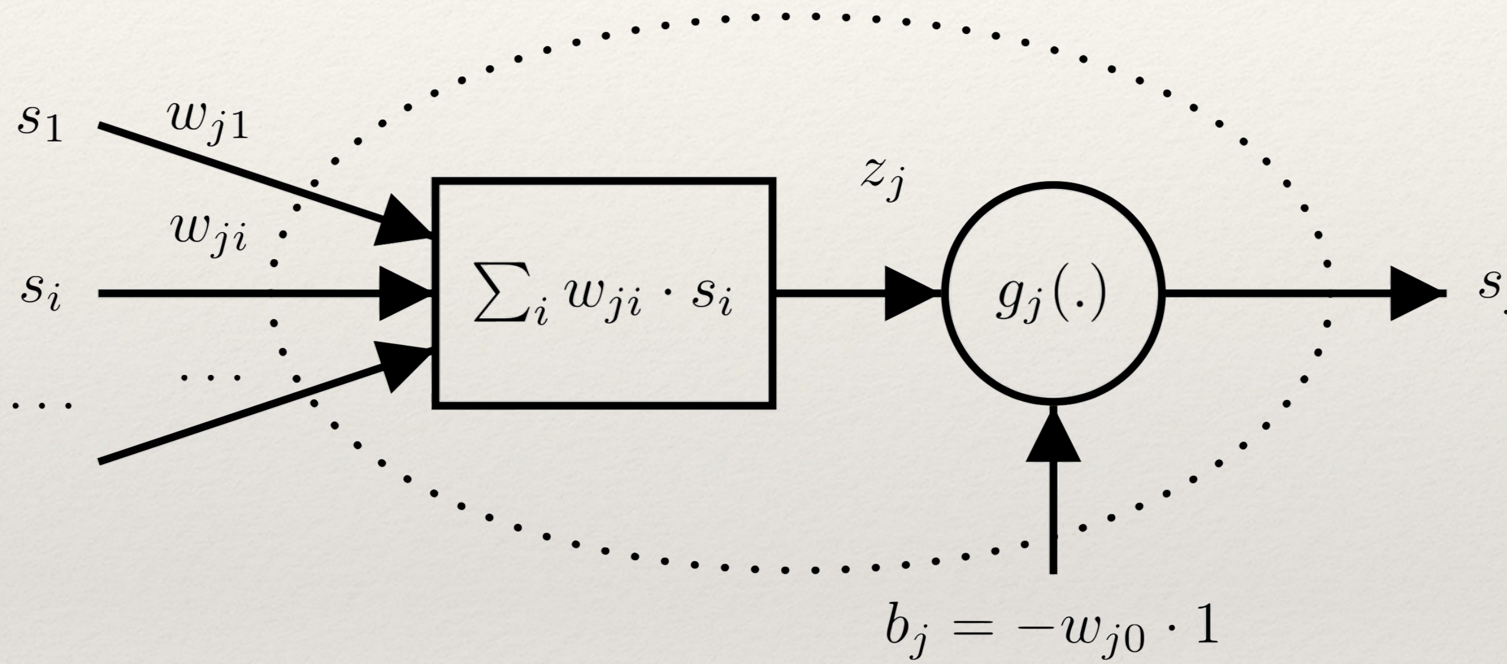


- Artificial Neural Network
  - Introduction
  - Training
  - Stochastic Gradient Descent
  - ADAM
  - Deep Neural Networks (RNN and LSTM)
- Project
  - Data structure
  - Theoretical goal
  - Experiments
  - Results
  - Considerations and Conclusions
- References

# Artificial Neural Network



An ANN is a net made up by **Artificial Neurones**

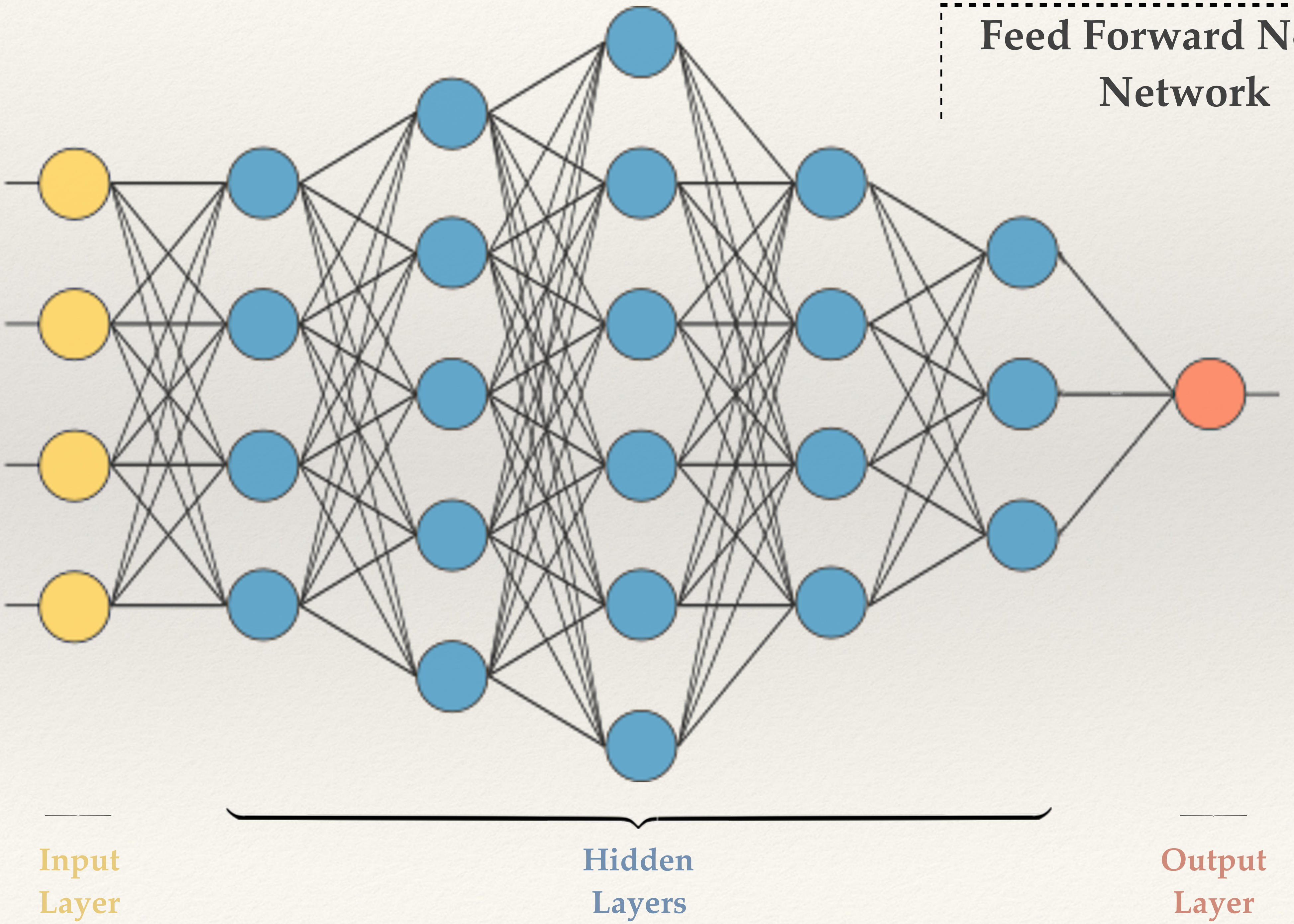



- Weights $w_{ji}$
- Activation value $z_j$
- Activation threshold (bias) $b_j$
- Activation function $g_j(\cdot)$ [Sign, Linear, Sigmoid, Hyperbolic tangent,ReLU, …]

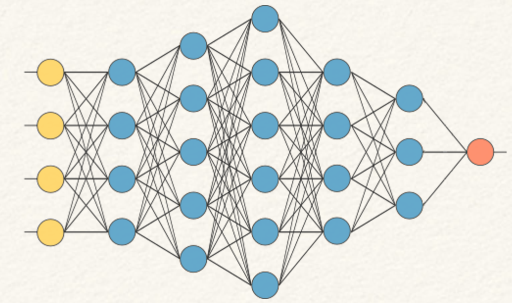
# Artificial Neural Network

An ANN is a net made up by **Artificial Neurones**

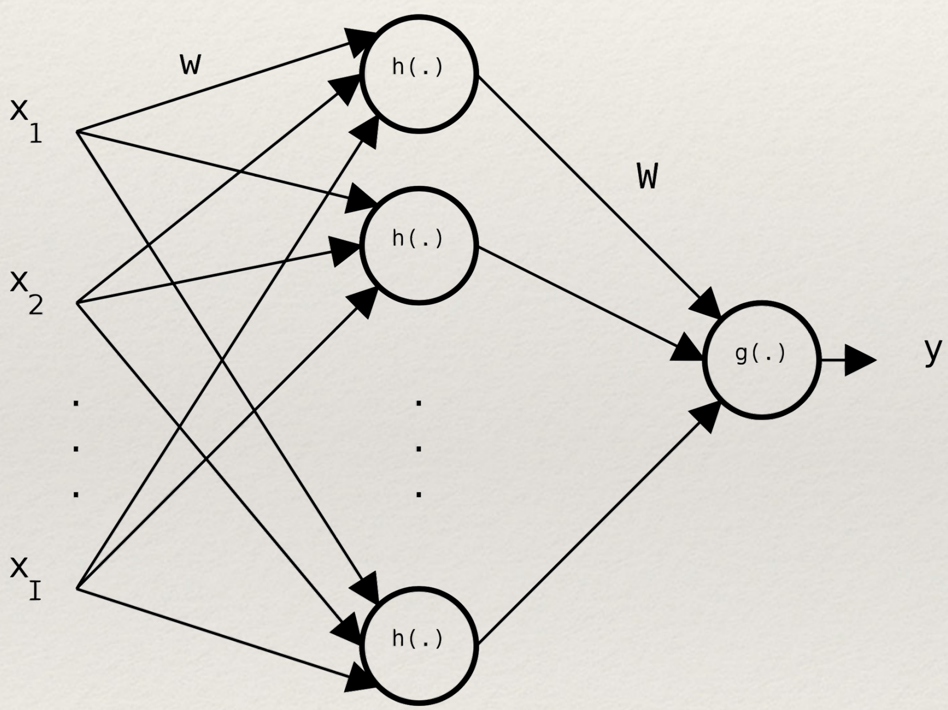
**Feed Forward Neural Network**



# Training an ANN



- Back-propagation (Gradient Descent) $w^{new} := w^{old} - \eta \frac{\partial E}{\partial w}$
- 2 phase algorithm (1 propagation, 2 weight update)




$$y = g\left(\sum_{j}^{J} W_j \cdot h\left(\sum_{i}^{I} w_{ji} \cdot x_i\right)\right)$$

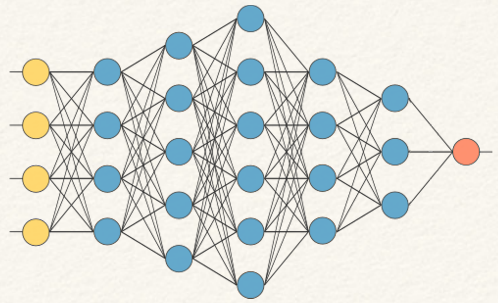
$$E = \sum_{n}^{N} (t_n - y_n)^2$$

$$a_j = \sum_{i}^{I} w_{ji} \cdot x_i \qquad b_j = h(a_j) \qquad A = \sum_{j}^{J} W_j b_j$$

$$\begin{aligned}
\frac{\partial E}{\partial W_j} &= \sum_{n}^{N} 2(t - g(A)) \cdot \frac{\partial}{\partial W_j}(t - g(A)) \\
&= \sum_{n}^{N} 2(t - g(A)) \cdot (-g'(A)) \cdot \frac{\partial}{\partial W_j} A \\
&= \sum_{n}^{N} 2(t - g(A)) \cdot (-g'(A)) \cdot b_j
\end{aligned}$$

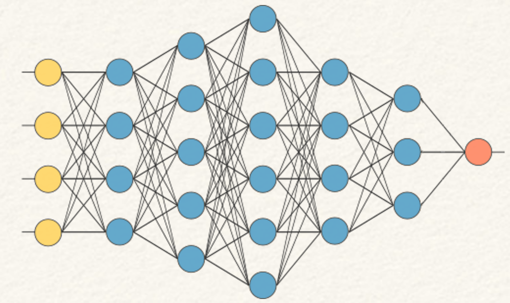
$$W_j^{k+1} = W_j^k + 2\eta \sum_{n}^{N} (t - g(A)) \cdot g'(A) \cdot b_j$$

# Issues in Training ANN



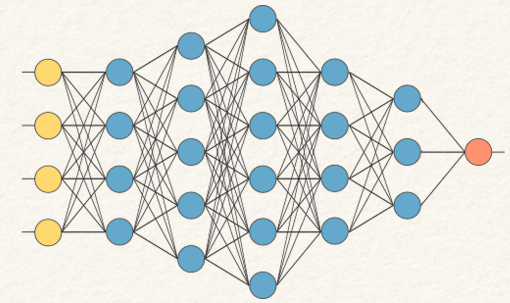
- Convergence
    - Gradient Descent with Momentum
    - Quasi Newton Methods
    - Conjugate Gradient
- Local Optima
    - Multiple Restart
    - Randomised Algorithms
- Generalisation and Overfitting
    - Early Stopping
    - Weight Decay

# SGD Algorithm



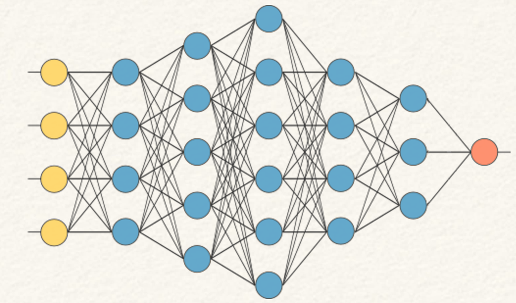
- Online Algorithm: Each input generates a weight adjustment
- Mini-Batches in practice
- If the learning rate decreases with an appropriate rate, SGD converges to a global minimum when the objective function is convex (to a local minimum otherwise)

# Improving SGD



- **Momentum**: remembers the update $\Delta w$ at each iteration and determines the next update as a linear combination of the gradient and the previous update
- **AdaGrad** *Adaptive Gradient Algorithm*: SGD with per-parameter learning rate [ increases the learning rate for more sparse parameters and decreases the learning rate for less sparse ones ]
- **RMSProp** *Root Mean Square Propagation*: learning rate is adapted for each of the parameters. It divides the learning rate of a weight by a running average of the magnitudes of recent gradients for that weight.

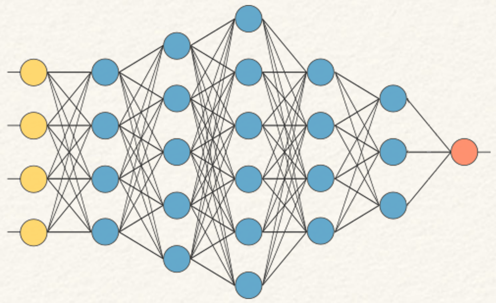
# ADAM algorithm



**Adaptive Moment Estimation**

- Combination of AdaGrad and RMSProp
- Invariant to diagonal rescaling of the gradients
- Suited for problems that are large in terms of data and/or parameters
- Adapts the parameter learning rates based on the average first moment (the mean) as in RMSProp, and on the average of the second moments of the gradients (the uncentered variance)

# Deep Neural Networks



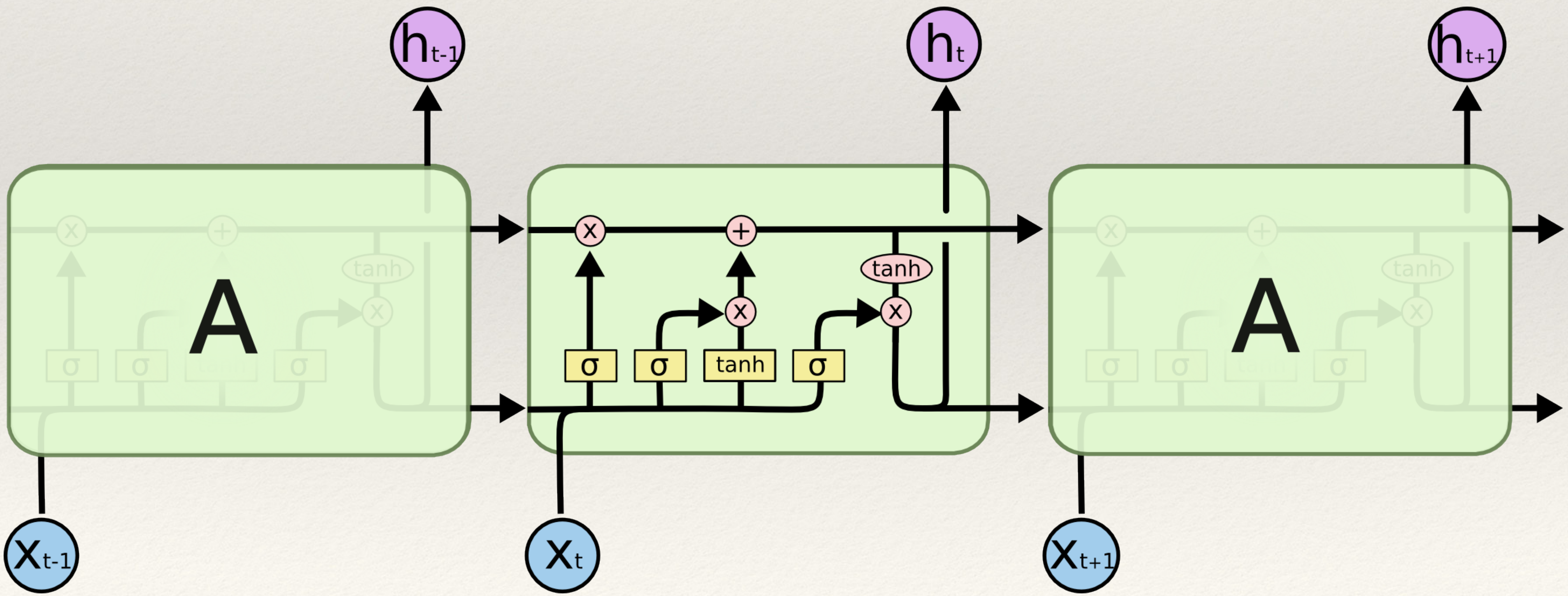
- FeedForward Neural Networks FFNNs
- Recurrent Neural Networks RNNs
- Long Short Term Memory Networks LSTM
- Convolutional Neural Networks CNNs

# Recurrent Neural Networks

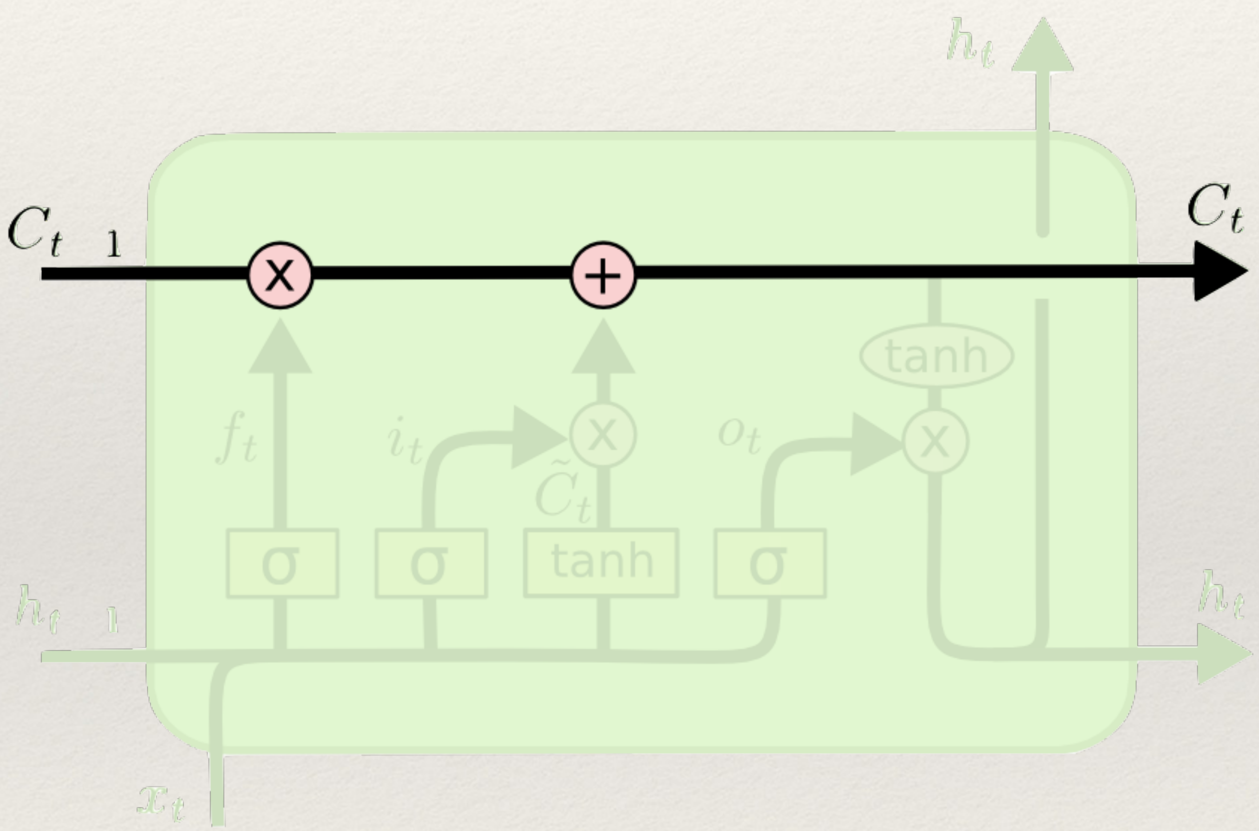
- Add a new unit *b* to the hidden layer and a new input unit c(t) to represent the value of *b* at time (t – 1). *b* thus can summarise information from earlier values of x arbitrarily distant in time

# Long Short Term Memory

- Special kind of RNN, capable of learning long-term dependencies

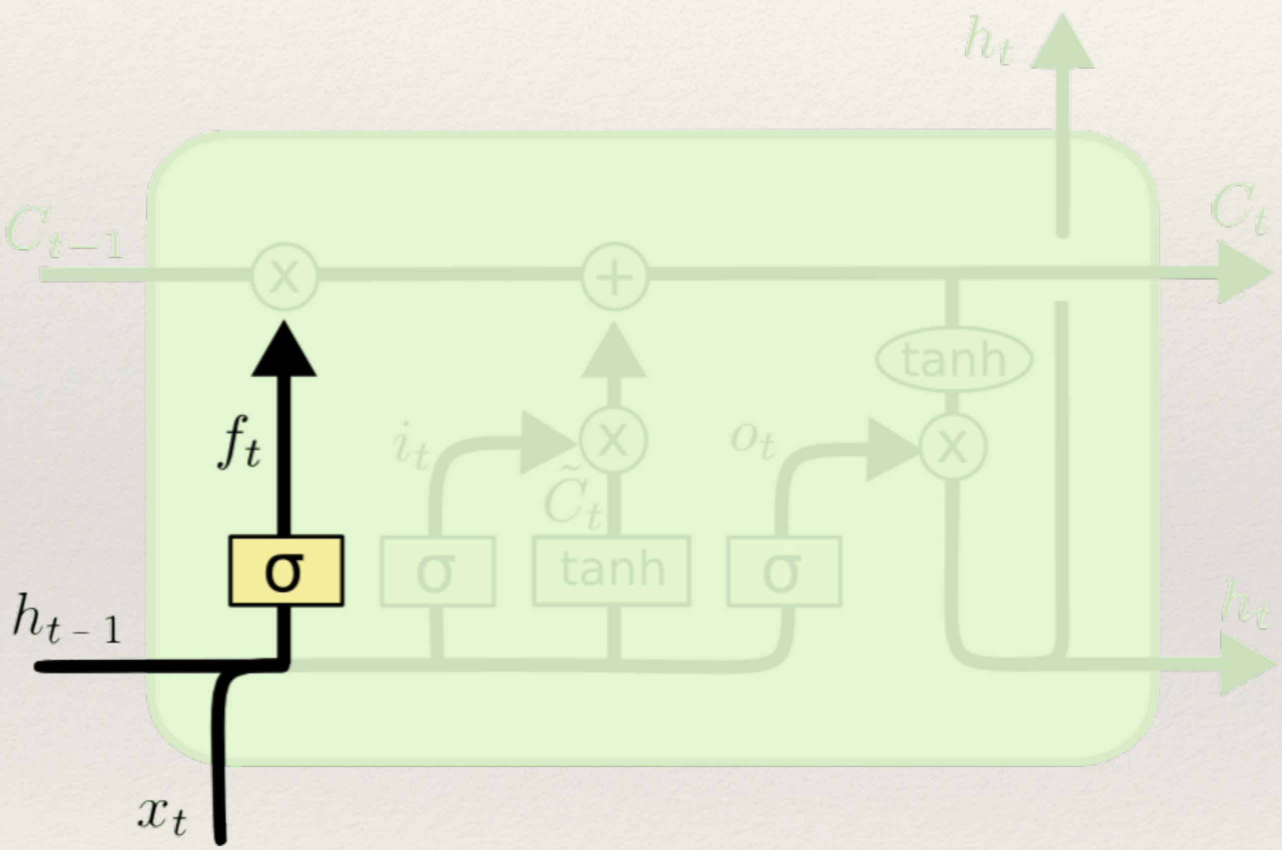


# Long Short Term Memory




Cell State

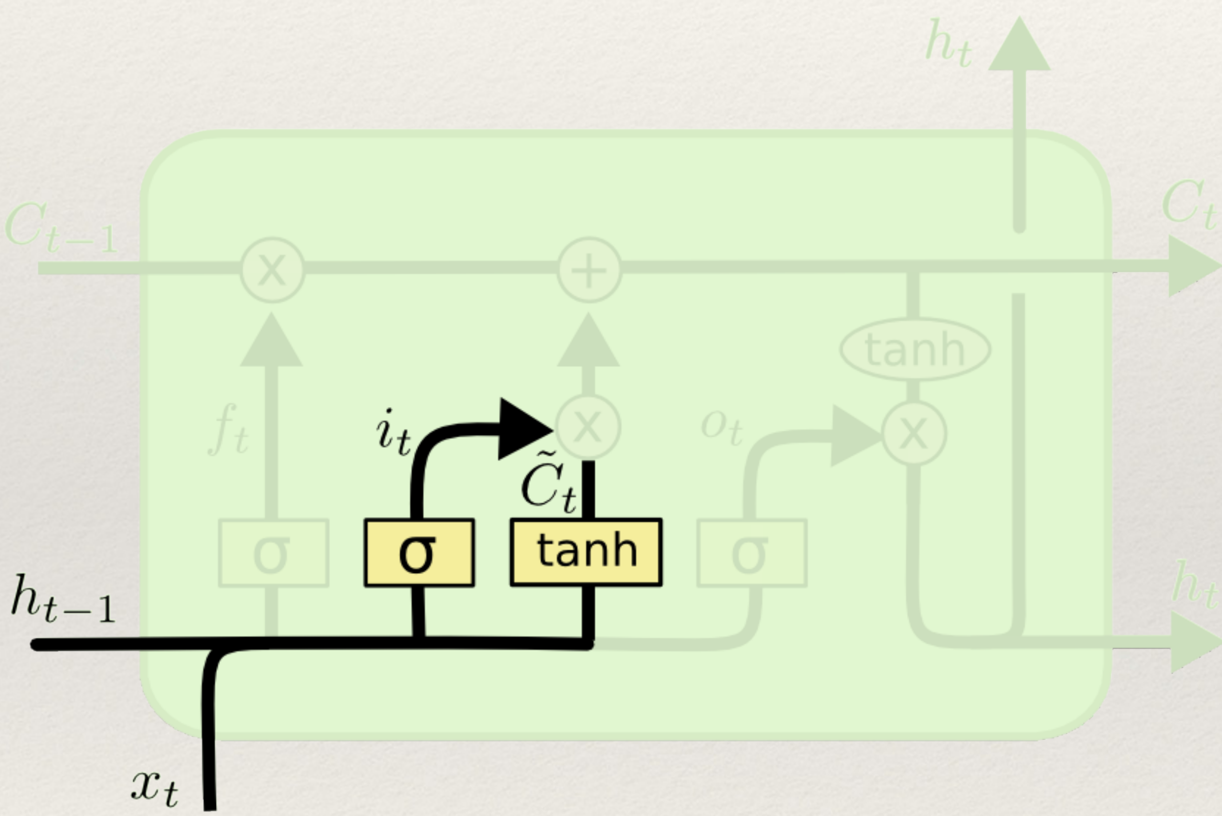
# Long Short Term Memory




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Forget Gate

# Long Short Term Memory

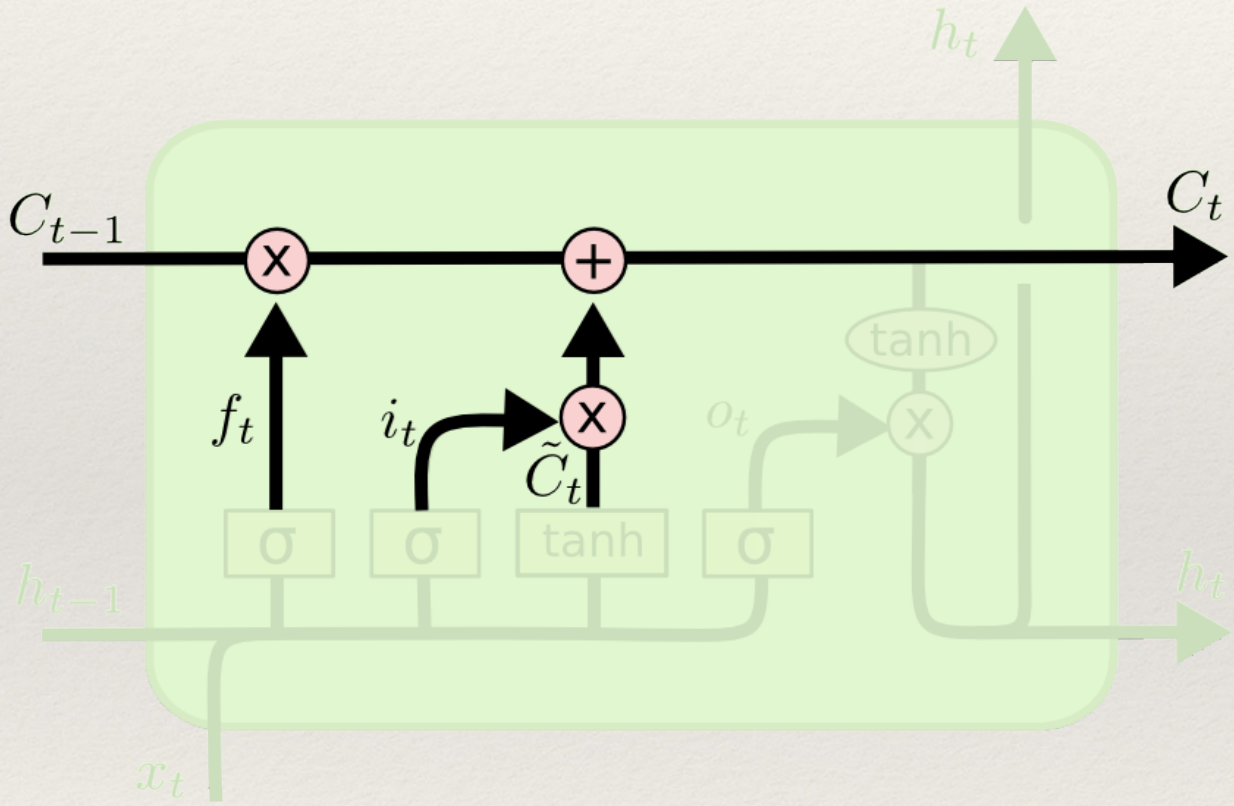



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Input Gate

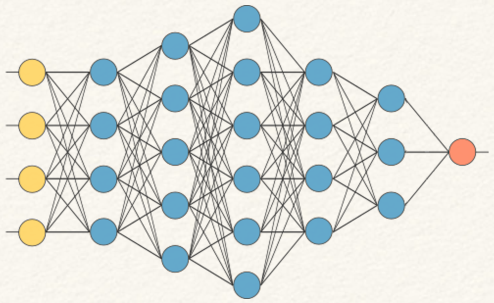
# Long Short Term Memory

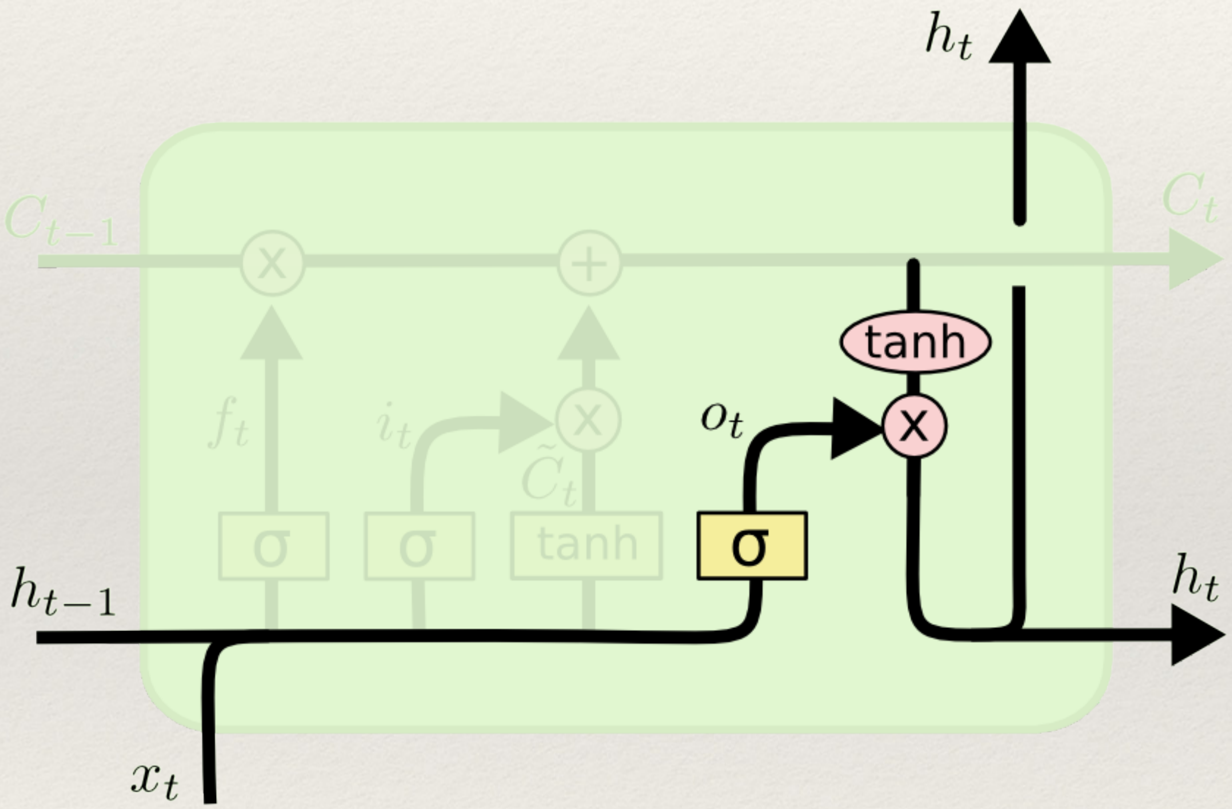



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Memory Gate

# Long Short Term Memory



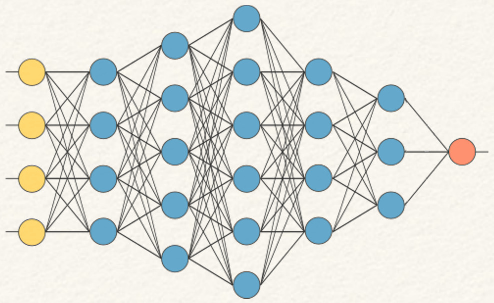



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Output Gate

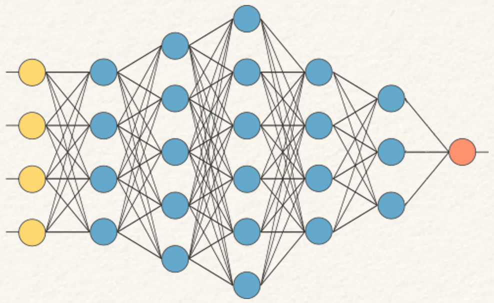
# PROs and CONs of DNNs



| Pros | Cons |
| --- | --- |
| Important features are learned automatically from data | A lot of data is needed |
| No domain knowledge required | Hard to analyse - Black box |
| | Initialization |

# Project Goal

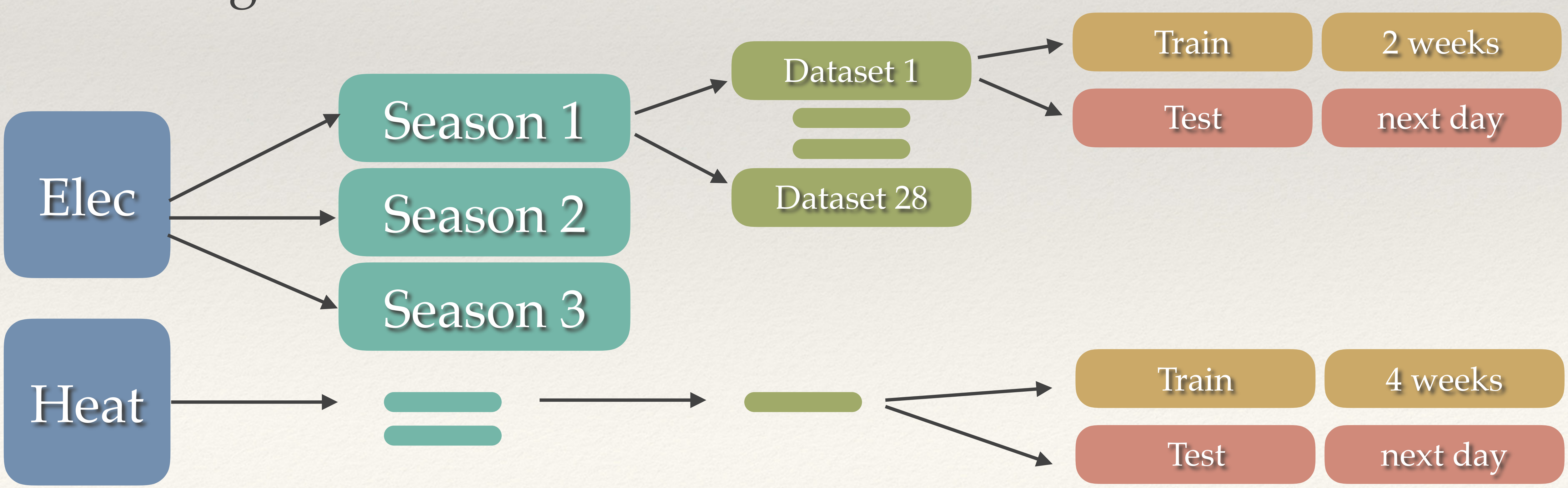
## Heating and Electricity Consumption Prediction



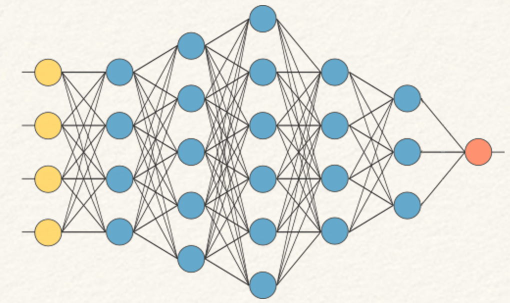
- Predict the next 24 hours of Heating and Electricity consumption of an hospital.

# Data (1)

- 2 datasets : one for the heating data and one for the electricity data
- Each dataset is divided in 3 sub-datasets (3 seasons)
- Each season is composed by 28 small dataset used for training and testing



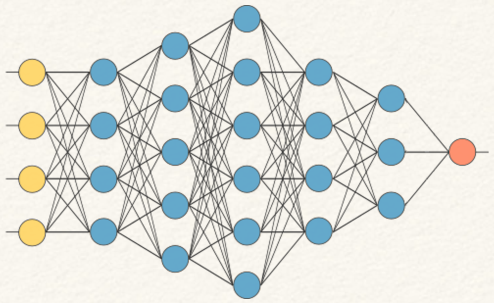
# Data (2)



- 11 features + 15 features

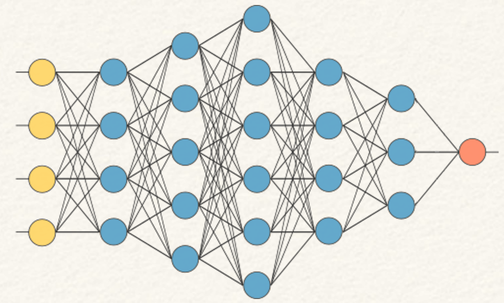
| Original | Additional |
| --- | --- |
| 4 weather forecast infos for the specific hour (sample) | 12 weather forecast infos of the 3 previous (hours) samples |
| 6 binary variables that identify the week day | 3 varaibles indicating the electric (or heating) consumption of the same hour of the same week day of the previous three weeks |
| 1 binary variable that states if the day is holiday or not | |

# Theoretical Goal



- To discover if a DNN trained on the "11 features" dataset is able to predict with a accuracy similar to that of a single hidden layer NN trained on handcrafted features (11 + 15). Is the DNN able to learn the important features that characterise the given problem?
- Compare the results with a LSTM NN trained on the 11 features. Is a LSTM NN able to exploit the past to better predict the future?

# Experiments



1. First experiment with simple FFNN
2. Preprocess the dataset by removing seasonal components
3. Modify the structure and the training options of the nets
4. ~~*Data multiplication*~~
5. Enlarge the dataset (adding 2 weeks in the training [slide_horizon and incremental]

# Best Results

| ELECTRICITY Top 10 - 2 weeks dataset | | | | | | |
|---|---|---|---|---|---|---|
| | Season 1 | | Season 2 | | Season 3 | |
| | kind of net : | rmse | kind of net : | rmse | kind of net : | rmse |
| 1 | "all_deseason_40" | "0.059021" | "small_deseason_70" | "0.11623" | "all_deseason_56" | "0.055899" |
| 2 | "all_deseason_72" | "0.059078" | "small_deseason_58" | "0.11646" | "all_deseason_80" | "0.055916" |
| 3 | "all_deseason_32" | "0.059283" | "small_deseason_82" | "0.11652" | "all_deseason_44" | "0.056047" |
| 4 | "all_deseason_44" | "0.059527" | "small_deseason_62" | "0.11675" | "all_deseason_40" | "0.056232" |
| 5 | "all_deseason_76" | "0.060081" | "small_deseason_66" | "0.11722" | "all_deseason_36" | "0.056552" |
| 6 | "all_deseason_68" | "0.060191" | "small_deseason_74" | "0.11753" | "all_deseason_76" | "0.056567" |
| 7 | "all_deseason_16" | "0.060398" | "small_deseason_78" | "0.11814" | "all_deseason_68" | "0.056705" |
| 8 | "all_deseason_80" | "0.060421" | "small_57" | "0.1183" | "all_deseason_60" | "0.057028" |
| 9 | "all_deseason_8" | "0.061021" | "small_61" | "0.12021" | "all_deseason_64" | "0.057055" |
| 10 | "all_deseason_36" | "0.061359" | "small_81" | "0.12337" | "all_deseason_72" | "0.057151" |

| HEATING Top 10 - 4 weeks dataset | | | | | | |
|---|---|---|---|---|---|---|
| | Season 1 | | Season 2 | | Season 3 | |
| | kind of net | rmse | kind of net : | rmse | kind of net : | rmse |
| 1 | "all_55" | "0.1058" | "all_75" | "0.071589" | "all_deseason_84" | "0.059433" |
| 2 | "all_35" | "0.10647" | "all_43" | "0.07177" | "all_deseason_64" | "0.059747" |
| 3 | "all_71" | "0.10706" | "all_55" | "0.071785" | "all_deseason_60" | "0.060165" |
| 4 | "all_31" | "0.10754" | "all_3" | "0.072347" | "all_deseason_28" | "0.06032" |
| 5 | "all_59" | "0.10768" | "all_47" | "0.072429" | "all_deseason_72" | "0.06074" |
| 6 | "all_43" | "0.10825" | "all_35" | "0.072557" | "all_deseason_68" | "0.060958" |
| 7 | "all_63" | "0.10903" | "all_31" | "0.072701" | "all_deseason_80" | "0.06116" |
| 8 | "all_67" | "0.11012" | "all_67" | "0.073447" | "all_deseason_56" | "0.061252" |
| 9 | "all_39" | "0.1102" | "all_79" | "0.073576" | "all_deseason_36" | "0.061338" |
| 10 | "all_83" | "0.11024" | "all_51" | "0.073652" | "all_deseason_76" | "0.061376" |

From 1 to 56 -> SGD Algorithm ; From 57 to 84 -> ADAM Algorithm

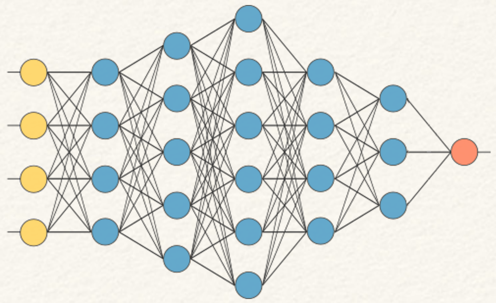
# Best Results

| ELECTRICITY - 4 weeks Increased dataset | | | | | | |
|---|---|---|---|---|---|---|
| | Season 1 | | Season 2 | | Season 3 | |
| | Kind of net | rmse | Kind of net | rmse | Kind of net | rmse |
| *2 weeks Best* | *"all_deseason_40"* | *"0.059021"* | *"small_deseason_70"* | *"0.11623"* | *"all_deseason_56"* | *"0.055899"* |
| 1 | FFNN_SH_3 | "0.07161" | FFNN_I_8 | "0.17826" | FFNN_SH_3 | "0.054087" |
| 2 | FFNN_SH_4 | "0.072802" | FFNN_SH_3 | "0.18829" | FFNN_SH_2 | "0.054209" |
| 3 | FFNN_SH_2 | "0.073216" | FFNN_SH_2 | "0.19272" | FFNN_SH_5 | "0.056093" |
| 4 | FFNN_SH_5 | "0.074144" | FFNN_SH_4 | "0.19675" | FFNN_SH_4 | "0.059436" |
| 5 | FFNN_I_8 | "0.075395" | FFNN_SH_1 | "0.19687" | FFNN_I_8 | "0.061119" |
| 6 | FFNN_SH_1 | "0.076003" | FFNN_SH_5 | "0.20191" | FFNN_SH_1 | "0.070659" |
| 7 | LSTM_I_6 | "0.089365" | LSTM_I_6 | "0.22048" | LSTM_I_6 | "0.085532" |
| 8 | LSTM_I_7 | "0.090354" | LSTM_I_7 | "0.22324" | LSTM_I_7 | "0.10065" |

| HEATING - 6 weeks Increased dataset | | | | | | |
|---|---|---|---|---|---|---|
| | Season 1 | | Season 2 | | Season 3 | |
| | Kind of net | rmse | Kind of net | rmse | Kind of net | rmse |
| *4 weeks Best* | *"all_55"* | *"0.1058"* | *"all_75"* | *"0.071589"* | *"all_deseason_84"* | *"0.059433"* |
| 1 | FFNN_SH_1 | "0.10814" | FFNN_SH_1 | "0.087646" | FFNN_SH_1 | "0.069542" |
| 2 | FFNN_I_8 | "0.11132" | FFNN_SH_2 | "0.088027" | FFNN_SH_4 | "0.071043" |
| 3 | FFNN_SH_4 | "0.11171" | FFNN_SH_3 | "0.091076" | FFNN_SH_5 | "0.073204" |
| 4 | FFNN_SH_5 | "0.11209" | FFNN_SH_4 | "0.091672" | FFNN_I_8 | "0.075838" |
| 5 | FFNN_SH_3 | "0.11351" | FFNN_SH_5 | "0.091917" | FFNN_SH_2 | "0.077845" |
| 6 | FFNN_SH_2 | "0.11388" | FFNN_I_8 | "0.099661" | FFNN_SH_3 | "0.078878" |
| 7 | LSTM_I_7 | "0.34203" | LSTM_I_6 | "0.13229" | LSTM_I_7 | "0.20192" |
| 8 | LSTM_I_6 | "0.36638" | LSTM_I_7 | "0.14767" | LSTM_I_6 | "0.27224" |

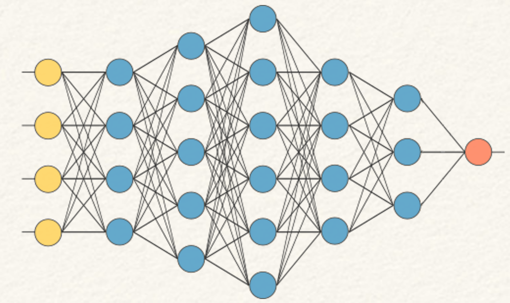
- SH : Sliding Horizon dataset
- I : incremental dataset
- RMSE : average of the daily error per season

# Considerations (1)

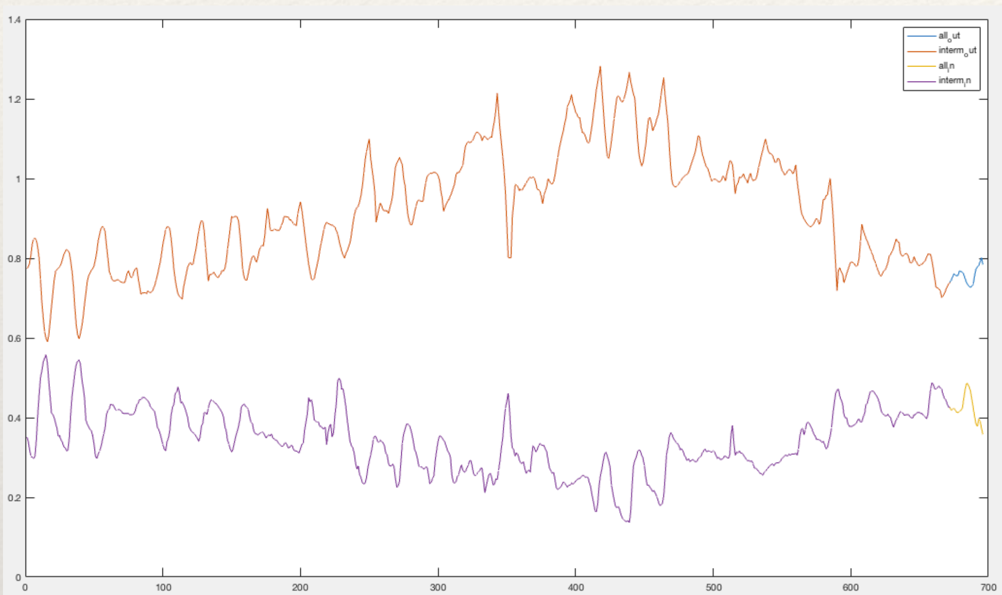


- 26 features vs 11 features / de-trend vs trend / preprocessing

# Considerations (2)

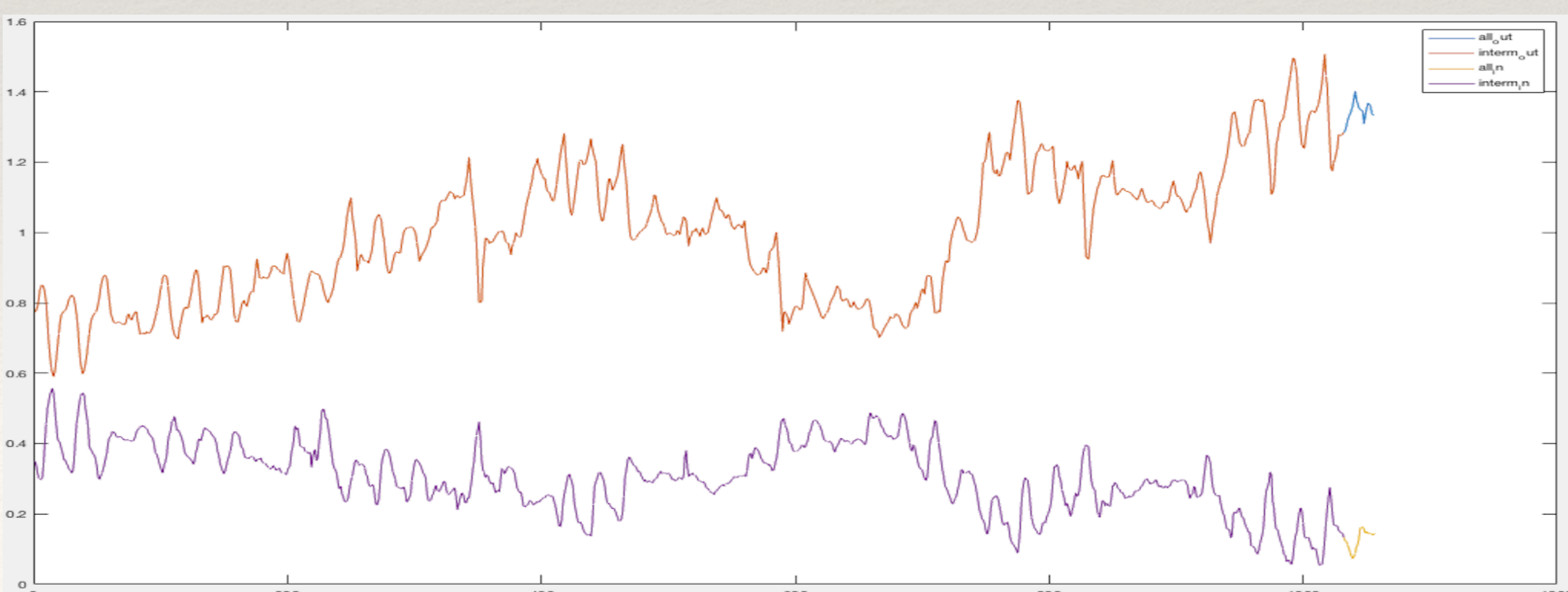


- Enlarged dataset vs original dataset




Heating 4 weeks sample

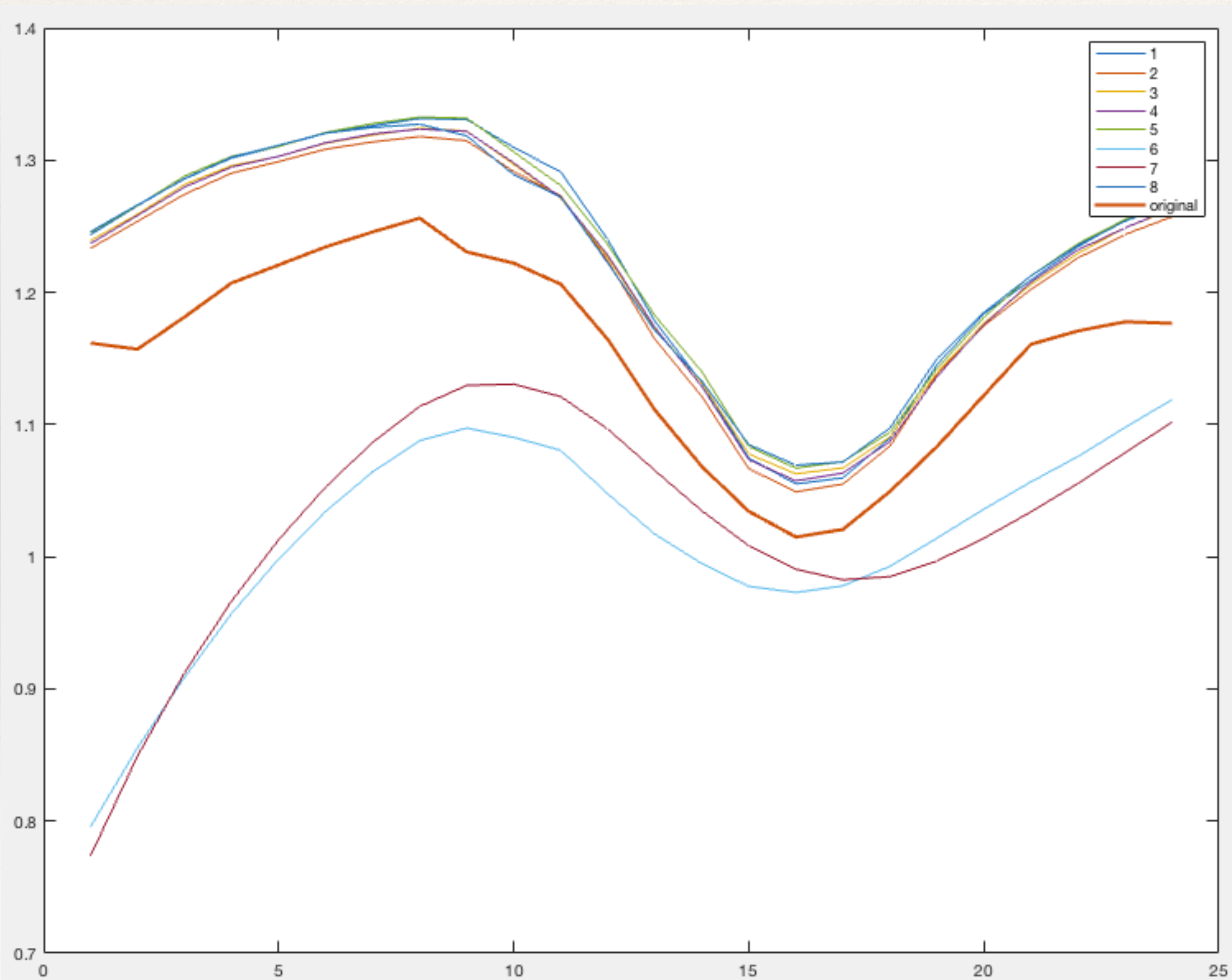
Increased presence of annual trends




Heating 6 weeks sample

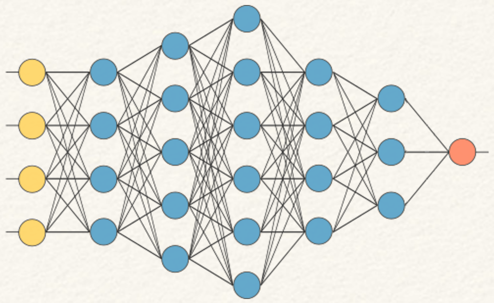
# Considerations (3)

- LSTM initial problems (slow start) → slow learning



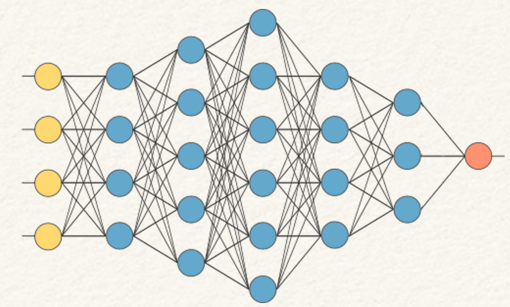

| |
|---|
| FFNN_SH_1 |
| FFNN_SH_2 |
| FFNN_SH_3 |
| FFNN_SH_4 |
| FFNN_SH_5 |
| LSTM_I_6 |
| LSTM_I_7 |
| FFNN_I_8 |

# Conclusions



- DNN not always good → need a large amount of "good" training data
- DNN might not be able to extract the useful features
- Training execution time vs accuracy
- Black box model → not easy to find the best structure or training options

# References




- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://it.mathworks.com/help/nnet/examples/time-series-forecasting-using-deep-learning.html
- https://it.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html
- https://it.mathworks.com/help/nnet/ug/ist-of-deep-learning-layers.html
- https://machinelearningmastery.com/improve-deep-learning-performance/
- https://it.mathworks.com/videos/developing-forecast-models-from-time-series-data-in-matlab-part-1-93067.html
- https://it.mathworks.com/videos/developing-forecast-models-from-time-series-data-in-matlab-part-2-93066.html
- Diederik P. Kingma, Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION
- Grippo, Manno, Sciandrone, Decomposition Techniques for Multilayer Perceptron Training
- LeCun, Bengio, Hinton, Deep learning
- Glorot, Bengio, Understanding the difficulty of training deep feedforward neural networks
- Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent