Calibration using AI

JUSTINE Jean   
 *Högskolan Väst*  
*EIGSI*Trollhättan Sweden  
jean.justine@aol.com

*Abstract*—The work we had to do was to find the inverse function to calibrate sensors. To do so, we will train neural network, to find the best models, that can predict efficiently the true value using the sensors output as training set, and the true value of the environment as target, then when the models are precise enough we will have our calibration algorithm. All of the code is available on GitHub here: https://github.com/Justine-IA/Deep\_learning-/tree/main/Project

# Data Analysis

Firstly, the project will be a Deep Learning project that consist of finding an algorithm that predict true value in function of the value that a sensor finds, for that we will use a dataset that measure pollution in a city the dataset is available here: <https://archive.ics.uci.edu/dataset/360/air+quality>.

## Requirement

The requirement will be to find the inverse function to find the real value, through the value detected by the sensors, because sensor data (y) is a function of true measurement (x).

We have to find the inverse function, which is: *f^{-1}(y) = x.*

This is an important task because having the real measurement can be crucial to take actions later on.

## Data

The dataset is composed of time feature, value that the sensor finds, like hourly average sensor response of CO, and also the real value (Ground Truth) like of CO, not the one the sensor detected. There is no missing value according to the website but there is some not correct value like negative number of pollution or -200 for sensors temperature so we removed them as well as for the GT data, there is a lot of missing value so we just replace them with the one closer in time to them, the time and date will be used for a Recurrent Neural Network later on because the sensor is detecting pollution in a city so it will be useful to train a model with the times feature, because of cars or anything else that influence pollution. We also have humidity and temperature in addition to all the other feature of pollution and else.

## Visualisation

Here we have some visualization for the data such as:

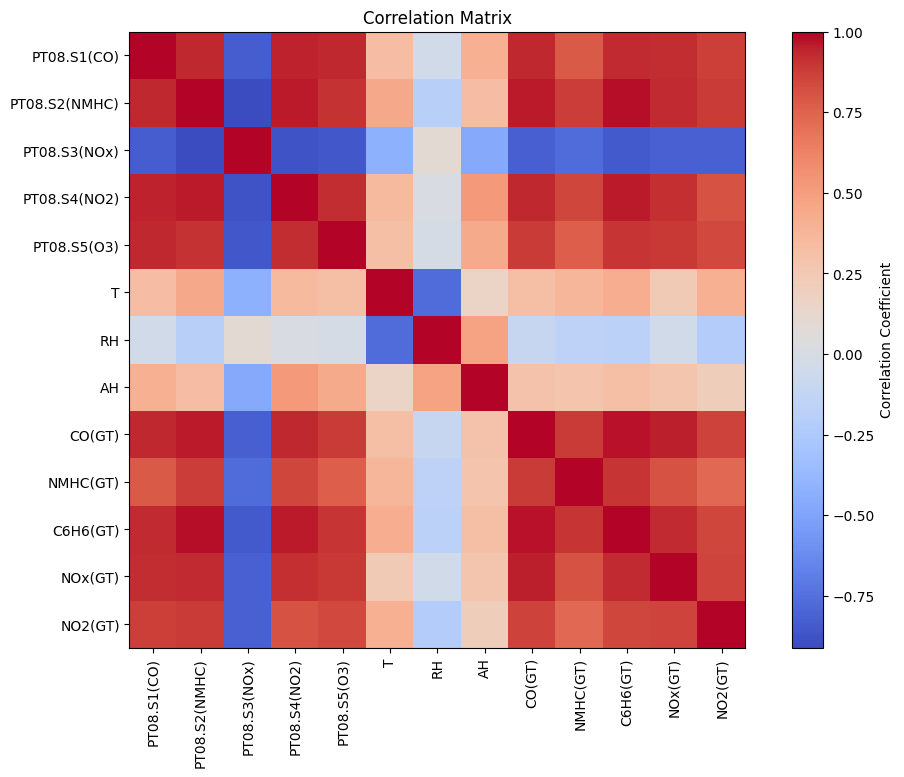


Fig. 1. Correlation Matrix

As shown in fig1, we can see the correlation between the polluting elements measured and its ground truth are very related to each other. We can also see that NOX and RH are not very related to anything but themselves.

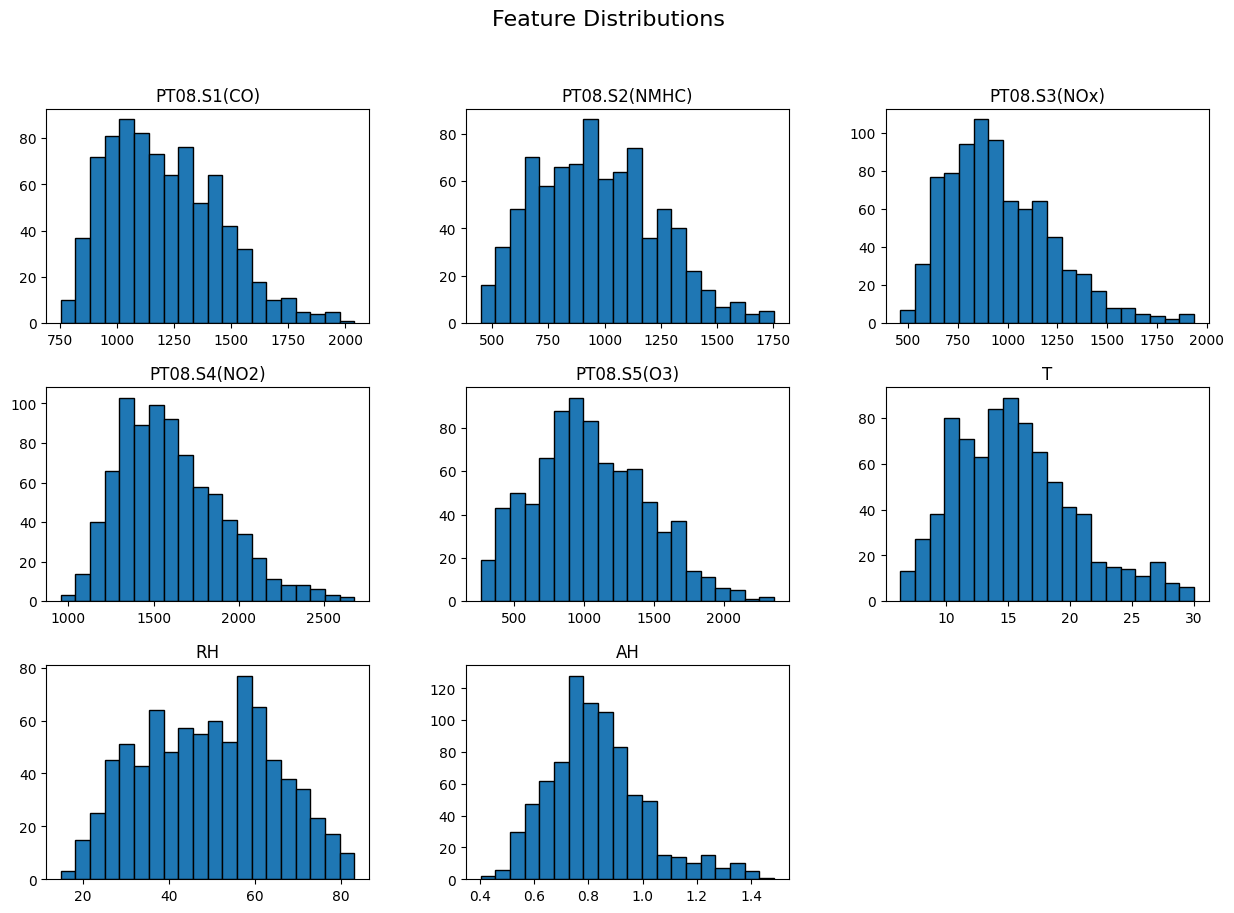


Fig. 2. Feature Distribution

We can see that in Fig 2, most of the data is correct and doesn’t need further data cleaning except for a scaling later on.

## Requirement

Some requirement that are in existing research. For temperature and light sensors some Neural Network (NN) were develop for [1].

Also ML based calibration can handle over time drift, and environment variability, improving long time accuracy [2].

List of requirement:

* Acquire Data
* Choosing Model
* Model Training and validation
* Figure of merits
* Documentation

# Algorithm Design

## Data Modeling

Firstly we fetch the data, then, as said in the first part, we then separate the feature form the target, y, the target, will take all of the GT value, x will take the rest, we will build a Recurrent NN so we keep time in a variable call X\_RNN, whereas X will not have time or date.

After that we need to scale the data, firstly we will need to transform the data into numerical value, like 2004 for year, 10 for month, 23 for day, 18 for hour instead of the usual 23/10/2004 and 18:00, we also have day of the week, after that we will add cyclical encoding for the time and day of the week, using cos and sin to have a cycle.

After all of that we need to do the real scaling we just import standards scaler and scale X and X\_RNN as well as y, we need to scale the data for having a better consistency, better metrics calculation, and it prevents certain data to be too dominant compare to others.

After we split the data between X train X test y train y test as usual but we also have X train RNN and y test RNN which contains the date and time.

Also, we had to reshape X train and test RNN, into a 3D tensor shape with samples, time step and features, because the original data was a 2D tensor with only samples and features, and for RNN it only takes 3D tensor because it needs the time so we added time step.

## Algorithm Design

### I choose as algorithm**:** **Recurrent Neural Network** and Deep Neural Network. I choose RNN because it is design and efficient to handle sequential data, and help capturing temporal dependencies in our dataset. In our case the feature is a shape of 5 time steps. I started the model with a layer of Simple RNN using relu as activation, then 0.2 dropout, then Dense layer of 32 neuron with Relu asd activation, another dropout of 0.3 and finally the output layer with the shape of our target [3].

The other algorithm is **Deep Neural Network** (DNN), I choose this one because it is efficient at finding complex pattern in high dimensional data, and nonlinear relationship. The model has an input layer of 128 neurons, using Relu as activation, a dropout layer of 0.3, then another dense layer of 64 neurons still using Relu, once again a dropout layer of 0.2 the last hidden layer is Dense with 32 neurons still using Relu and finally the output layer is like the RNN but with linear activation [4].

## Figure of merits

For figure of merits I choose, **Mean Square Error** (MSE) for the loss, and **Mean Absolute Error** (MAE) for the metrics.

## First Results

I used optimizer Adam for both algorithm and I got those results :

### **RNN Performance**

* **Test Loss (MSE): 0.1773**
* **Test MAE: 0.2411**

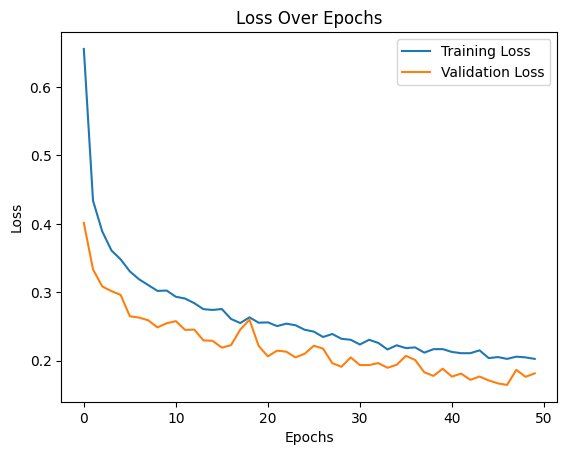


Fig. 3. Loss over epochs for RNN

### **DNN Performance**

* **Test Loss (MSE): 0.2387**
* **Test MAE: 0.2565**

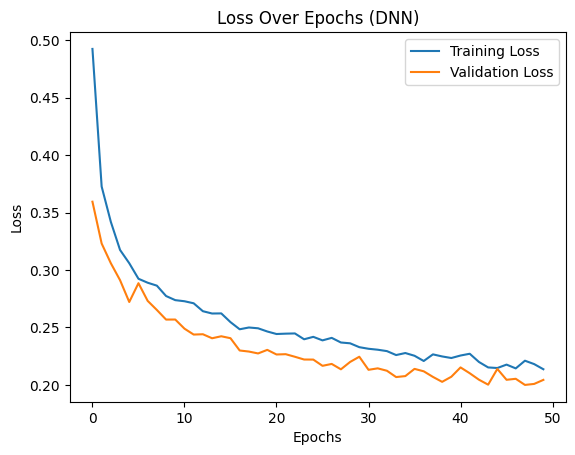
****

Fig. 4. Loss over epochs for DNN

We can see that the RNN outperform the DNN both in MAE and MSE, the DNN is slightly below but is still performing well even on sequential data. Also as seen in Fig 1. And Fig 2 the graph shows us that both model learn pretty well, they decrease the loss over the epochs the validation loss show that the model generalizes well on unseen data. But there is still some improvement that can be done.

## Improvement

To improve the two models, I choose multiple way:

* **Increasing complexity:**

For the DNN I increased complexity by adding more neurons in the dense layers, to 256, 128, 64.

For the RNN I did the same thing but with less neurons, I increased to 128 64 and 32.

The reasons for that is: adding more neurons, improve the models ability to understand more complex and nonlinear pattern in the data, also it’s better to understand high dimensional data.

* **Change SimpleRNN to LSTM in the RNN model:**

I changed it and increased the number of neurons to 128, the reason for that is that LSTM is better at understanding temporal patterns, as well as long term dependencies.

* **Increase Dropout Rate to 0.5 in total instead of 0.4:**

Dropout help to prevent over fitting, we did not have it with the first models but with more complexity and more epochs it could have happen so we decided to increase the dropout rate from 0.4 to 0.5.

* **Increasing the number of Epochs and batch size:**

We increased the number of epochs from 50 to 100, and batch size from 32 to 64 in the goal of allowing the model to fully converge because the loss did not attaign a plateau of no amelioration. We increased Batch size to stabilize more efficiently the gradient.

## Results of Improvement

We will discuss the results of the improvement after incorporated all of the above

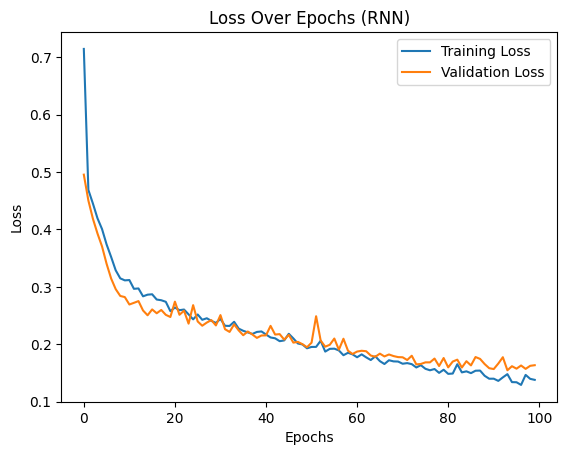


Fig. 5. Loss over epochs for more complex RNN

As seen in Fig 5. The validation and training loss stabilize around 70 epochs, the two lines are very close indicating that there is no overfitting and we had those results:

* Training MSE : 0.1379
* Vvalidation MSE : 0.1635
* New Test MSE : 0.1583
* New Test MAE : 0.225

Compare to what we had first there is a slight improvement, our results were already good but become even better. Showing that the LSTM layer is more efficient than the previous Simple RNN layer, as well as more epochs and more neurons provided better results, also there is no overfitting, thanks to the increase in the dropout rate.

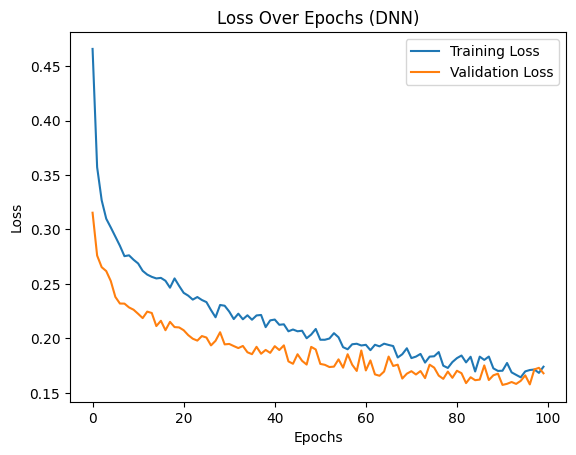


Fig. 6. Loss over epochs for more complex DNN

In Fig 6. We witness the growth of the new and more complex DNN, over 100 epochs. In this graph we can see the model is generalizing well, around 80 epochs the validation stabilizes showing good performance on unseen data. As for the training, it continues to decrease but is lower than the validation set showing a slight overfitting.

We had those results:

* Training MSE : 0.174
* Validation MSE : 0.168
* Test MSE: 0.162
* Test MAE: 0.227

We had slighlty worse results than the RNN but still very efficient. The improvement we did had an impact and improved the model, more neurons helped as well as more epochs.

# Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

## Abbreviations and Acronyms

Define

# Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

**The template is designed for, but not limited to, six authors.** A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than six authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

##### References

1. Y. Zhang, L. O. H. Wijeratne, S. Talebi, and D. J. Lary, "Machine Learning for Light Sensor Calibration," Sensors, vol. 21, Sep. 2021. Available: <https://www.mdpi.com/1424-8220/21/18/6259>.
2. Z. Wang, "Evaluating the Efficacy of Machine Learning in Calibrating Low-Cost Sensors," Applied and Computational Engineering, vol. 71, 2024. Available: <https://www.ewadirect.com/proceedings/ace/article/view/15136>.
3. S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," International Journal of Forecasting, vol. 36, no. 1, Jan. 2020.
4. F. Kastner, B. Janßen, F. Kautz, and M. Hübner, "Exploring Deep Neural Networks for Regression Analysis," in Proceedings of the International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO), 2018