



How to maximize NeurEco?

A quick guide

NeurEco is a Neural Networks (NN) factory relying on parsimony. The creation process of each NN is automatic based on the parameters entered by the user.

Here are some helpful tips to maximize your experience with using NeurEco.

1. Validation data: The foundation

The validation set plays a crucial role during the learning process as it stops the enrichment when overfitting is detected. If such a set is not chosen carefully it can lead to a building process stopping prematurely or on the contrary to an overfitting issue with an oversized network.

The automated selection made by NeurEco is usually what suits best the learning process. But on some occasions the end-result may be sensitive to the “*validation data percentage*” setting. If the resulting model is not satisfactory it might be helpful to explore this setting.

If you prefer to provide the validation dataset yourself, it is crucial that it represents well how the model would be used with new data.

Example:

If dealing with data deriving from time-series (a sensor measuring a quantity for example) and getting access to several trajectories to form the learning and validation sets, a pertinent way to separate learning from validation is to pick entire trajectories and add them to the validation set.

Another way to do this would be to select individual points in every trajectory and assign them to validation. This would likely lead to overfitting as each of these points would have very similar neighbors among the learning set.

Another important aspect of this selection is the notion of convex hull. If possible, provide a learning set that contains the points forming the convex hull of both the validation and test data.

2. Normalization is important: Choose wisely

Use GPU: ☐ False
GPU:

Advanced settings ▼

Disconnect inputs if possible: ☐ True

Final learning: ☐ True

Validation data percentage:

Regularization coefficient:

Start build from checkpoint:

Input normalization ▼

Shift type:

Scale type:

normalize per feature: ☐ True

Output normalization ▼

Shift type:

Scale type:

normalize per feature: ☐ False

Output normalization is particularly sensitive as it can change the cost function. Set “*normalize per feature*” to **true** if trying to fit targets of different natures (temperature and pressure for example) and want to give them equivalent importance.

Set “*normalize per feature*” to **false** if trying to fit quantities of the same kind (a set of temperatures for example) or a field.

If neither of these options suits you, do not hesitate to normalize your data your own way prior to feeding them to NeurEco (and deactivate output normalization).

The input normalization is by default tuned to suit NeurEco. Once again, if dealing with inputs of the same nature (a field for example) do not hesitate to switch “*normalize per feature*” to **false**.

3. Explore Checkpoints

NeurEco saves checkpoints throughout the learning process.

These can be useful to you for the following reasons:

- Some building processes might take a long time. It may happen that the model is already sufficiently efficient, but the enrichment process lasts longer to reach slightly better accuracy. In that case you can start exploring the capabilities of the model using existing checkpoints.
- It may happen that the end-result of the model is slightly overfitting the data. In that case, don't hesitate to check if a checkpoint does not offer better prediction capabilities.

Model from every checkpoint can be exported in all supported formats.

Please contact our NeurEco team if you have any questions about the software.
We will be happy to help you having the best user experience.

support@adagos.com