F.A.Q. on LAB3 assignments (CNS 2017)

This document contains further clarifications on the output data required for the following assignments:

- "Neural Networks for temporal data processing" (Lab3-1)
- "Echo State Networks for temporal data processing" (Lab3-2)

In which format should I send the required data structures?

Data structures should be provided in a .mat file.

E.g. in the case of Lab3-1, Assignment1, apart from the source code (in .m format) and all the required figures (target vs output plot and learning curve in both .fig and .png format) it is required to provide: the net structure, the training record structure, training, validation and test MSE. Such data structures should be provided in a .mat file, e.g. using the command `save`:

```
save 'Lab3-1-TDNN-laserTask.mat' IDNN_net, IDNN_trainingRecord, IDNN_trMSE, IDNN_vlMSE, IDNN_tsMSE
```

```
save 'Lab3-1-RNN-laserTask.mat' RNN_net, RNN_trainingRecord, RNN_trMSE, RNN_vlMSE, RNN_tsMSE
```

What data should I provide?

In each assignment it is specified which data is required as the output of the assignment.

How many data should I provide?

For each assignment the provided output should refer only to the neural network selected through the model selection process (e.g. 1 IDNN and 1 RNN network for assignments in "Lab3-1", and 1 ESN for the assignments in "Lab3-2").

Very important note: your submission should be complete and as minimal as possible in terms of memory (e.g. NOT in the order of dozens of Mb)

Should I provide the source code of the assignments?

Yes.
As required in the assignments' text, source code in .m format for each exercise should be provided. **Note**: Besides the code (which is checked for errors and adoption of correct methodology), the data produced by you are useful for us to check your results (without assuming that we will run the code to produce them again). Thereby, all data (code, figures and data structures) must be provided, according to the specification of each assignment.

What is the value of N_U and N_Y in the Assignments for Lab3-2?

Both N_U and N_Y are equal to 1.

How should I set the values of the hyper-parameters for model selection?

You can either (manually) choose a set of values to use, either (systematically) use a grid.
In any case: the model for the final submission should have the hyper-parameterization selected on the validation set.

Hyper-parameters, such are, for instance, the size of the hidden layer, the length of the input delay for timedelaynet, the training function, the learning parameters, the number of training epochs, etc. should be *optimized*?

No, it is useful to sample them and to understand and decide a reasonable set of values to try in the grid. To find a suitable set of values for these models and tasks is part of your lab experience, a full optimization is out of the scope.

How many hyper-parameters should I try?

As many as you feel useful in order to perform a good model selection, i.e. a good trade-off between time resource and wide of the grid. The hyper-parameters that results more influent (check the sensibility of the results) will be the more interesting to be investigated.
Such choices are part of the quality of your work.

Should I report the set of hyper-parameter values used in my model selection (grid search)?

Yes, in the script .m file(s) you should report the values used in the grid. If you did not use a grid, but instead you have tried some values "by hand", at least you should report the list of values that you have tried in a comment line in the .m file.

Can I use GPU/multi-core etc to accelerate the training of the models in Matlab?

Yes.
Parallel Computing Toolbox allows Neural Network Toolbox to train networks faster. For example, to enable multi-core usage during training of a neural network you should use the option `'UseParallel'` and set it to `'yes'`, as in the following: `net = train(net,X,T,'UseParallel','yes');`
For further information (e.g. how to use GPUs to speed up the training), see the Matlab documentation (e.g. by typing `help train`).

How to plot both TR and VL/TS error in a unique plot (learning curve)?

To get a learning curve with both training and validation/test error plotted over the epochs, you should use `'divideind'` as divide-function in you net structure. For example:

```
tr_indices = 1:1:5000; %indices used for training
ts_indices = 5001:1:10092; %indices used for assessment
                           %can be validation or test
net.divideFcn = 'divideind';
net.divideParam.trainInd = tr_indices; %training indices specified
%indices for assessment should go in net.divideParam.testInd
net.divideParam.testInd = ts_indices;
%net.divideParam.valInd is used in Matlab for early stopping
% we don't need it -> set it empty!
net.divideParam.valInd = [];
```

Now, when the network undergoes the training process you can see the performance over epochs on both the training set and on your assessment set (validation or test).

How should I train the NARX network?

NARX networks should be trained in closeloop mode. In this modality, the actual output of the network is fed back into the input (otherwise it is the target output that is fed back through the output delay line, and this might result in an improper use of the network).
There are 2 ways of specifying that you want to use a NARX in closeloop form. In newer versions of Matlab, you need to specify the (output) feedback mode when you instantiate the network, e.g.

```
net = narxnet(inputDelays,feedbackDelays,hiddenSizes, feedbackMode,trainFcn)
```

where `feedbackMode` is a string that should be set to `'closed'`.
In older versions of Matlab you need to specify the `closeloop` feedback mode by using the command `closeloop`, e.g.

```
net = closeloop(net)
```

See the LaboratoryAssignment-Lab3-1 document for further information on this aspect.