Assignment #2: Predicting Ethernet network configuration correctness using feedforward Neural Networks

Version: 11th Nov 2021.

Instructors: Nicolas Navet (nicolas.navet@uni.lu), Patrick Keller (patrick.keller@uni.lu), Long Mai (long.mai@uni.lu).

Technological context

Ethernet networks are starting to be used for in-vehicle communications. The main reason is the important bandwidth offered by Ethernet, which is needed to implement modern automotive functions such as automated/assisted driving and infotainment services.

In the early stage of the design of a car, the network architect defines the architecture of the network (e.g., topology and link speeds). The question that we want to answer is to predict whether a given set of flows (making up the traffic exchanged between the Electronic Control Units) will meet their performance constraints on the candidate architecture. Performance constraints can be for instance deadlines ("the engine speed value must be received within 5ms after its production by the suspension") or throughput ("at least 7Mbps must be available for software update"). If all performance constraints of the flows are met, the network configuration (i.e., the architecture plus the flows) is said *feasible*. In the following, we will consider the network topology shown in Figure 1.

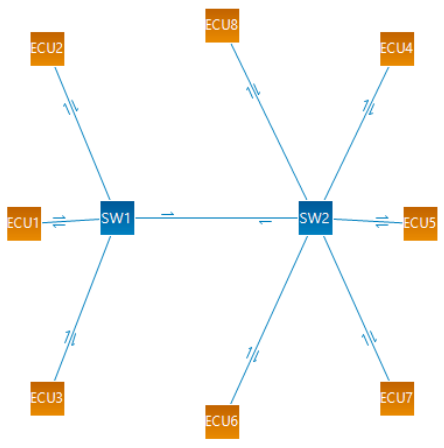

Figure 1: Topology of the network. There are 9 links, all operating at 100Mbps except the inter-switch link whose data rate is 1Gpbs.

There are algorithms to know whether a given set of flows will be feasible on a given topology. Those algorithms are however very compute intensive and cannot be used in design-space exploration when many design options (i.e., candidate solutions) have to be considered. Using classification, we would like to avoid running these algorithms and predict whether a network configuration will be feasible or not based on a training set (8000 configurations, and 2000 for the testing set).

Data set and template

The content of the dataset, in file “data_X.csv” (X in {testing, training}) available on Moodle, is as follows:

- Column 1: number of critical streams (e.g. dynamic of the vehicle, braking, suspension, etc)
- Column 2: number of audio streams
- Column 3: number of video streams
- Column 3: number of best-effort streams (code upload, etc.)
- Column 5: Maximum network load on any of the network links
- Column 6: Gini index

For this practical, you must not normalize the data. Whether a better accuracy can be achieved with normalization, and which normalization technique would work best, is outside the scope of the practical.

A code template is provided to you in file “Assignment2_template”. You are not allowed to change any of the methods but *train_model()* and possibly adding some callbacks (as seen in the exercises).

The labels of the samples are provided in file “labels_X.csv” (X in {testing, training}), please refer to function *load_dataset()* in the template to understand how training and testing sets are loaded.

Objectives and evaluation

The objectives of the practical are twofold:

- Provide you with some practical experience in using neural networks for prediction tasks, the process of designing a neural network and tuning its hyperparameters to get the maximal performance.
- Give you some experience in using standard ML tools: Keras, TensorFlow and Anaconda.

The practical is very open on purpose. Your final objective it to obtain the maximal accuracy on the testing set, while minimizing over/under-fitting. You have a complete freedom in terms of the shape, size, activation functions, weight initializers, etc, of the feedforward neural network.

Half of the grade will be automatically allocated based on the accuracy of the prediction and the amount of overfitting as calculated by the `grade_model()` function. You can thus estimate in advance the grade you will receive for that criteria. The quality and content of the report will determine the rest of the grade. **Hint: In some cases you might have to adapt the grade method to your particular choice of the loss function, if so, explain your changes and the reason why it was necessary in the report!**

Some guidelines and hints:

- First understand the data, its format and make sure you are able to **load it correctly**!
- Start with simple solutions, like the ones experimented in the classes, before considering more complex ones,
- Follow a systematic approach when tuning the network: do not set values at random, do not try to optimize all parameters at once (you may change two at once to save some time),

- Overfitting will be detrimental to your model (and your grades), hence you should pay extra attention to this phenomenon and try to mitigate it (e.g., with dropout),
- As a large portion of the grade will be determined by an algorithm based on one execution on our machine, you should make sure that the accuracy is stable. One step in that direction is to set the **seeds of the random generator** to a fixed value as done in the template provided.
- If your computer is too slow, consider using https://colab.research.google.com/
- **Plagiarism will be punished, this is not a group-work.**

The model that will be graded is the one that will be returned by the function *train_model()*.

What is expected from you in the report:

- Explain your design **decisions** for:
  1. the shape of the neural networks and its activation functions,
  2. the value of all hyperparameters. (E.g. If you leave everything at default, explain why!)
- Explain the (important) successive experiments having led to the final solution.
- For the final solution (and for some intermediate solutions if helpful for your argumentation) provide:
  1. a plot of the loss score **per epochs** on the **training** set,
  2. the prediction accuracy **per epoch** on the **training** set and **testing** set.
- Explain how you assessed your model (performance, overfitting, etc.) based on those plots.

Restrictions:

- You must use Keras with TensorFlow as the backend, as done before in the course.
- The **maximum number of parameters** (*model.summary()*) in the model is 2,000,000 (two millions). The **maximum number of training epochs** is 50. Any model that use more training epochs will be graded with only 50 epochs. TensorFlow v2.1 is used for grading, as mentioned in the instruction on how to install TensorFlow.
- **Do not** change the grading function.

Additional information:

You must submit on Moodle an archive file containing both your report in pdf format and the Python script, either .py or .ipynb, for the final solution. The deadline is December 1st, 2021 23:59:59 on Moodle. This is an individual project and solutions must not be shared among you. **Solutions which we consider too similar will have their grade divided by two.** You can of course use all other external sources you need**.** Do not hesitate to ask us for guidance during the class or by email, our role is to help you achieve the objectives**.**

The grade will consider the performance of your model on the testing set, the quality of the code and the report (writing, presentation, correctness). An individual oral presentation of the results may be organized.