Fortgeschrittenenpraktikum
Data Science

Prof. Dr. T. de Wolff
Dr. C. Brauer
Dr. M. Neumann-Brosig

Industry Project

Neural Networks for Leakage Detection in Vacuum Bagging

With regard to the application at hand, your goal is to learn a parameterized function f_θ that predicts leakage coordinates $y \in [-1,1] \times [-1,1]$ from normalized flow rates

$$x \in \{x \in \mathbb{R}^4 \mid \sum_j x_j = 1,\ x \geq 0\} \ .$$

For that purpose, you have training data $(x^i, y^i)_{i=1}^m$ available. Moreover, you are equipped with prior knowledge in the form of so-called equivariance conditions that can be incorporated into the machine learning pipeline. To see what that means, let us start with a fixed vector of flow rates $x = (\mu_1, \ldots, \mu_4)$ and associated leakage coordinates y as displayed in Figure 1 (top row, left), and suppose that the position of the leakage is rotated clockwise by 90° (top row, second from left). In this situation, if you neglect potential measurement noise, it is natural to claim that rotated coordinates y_{90} should be associated with accordingly shifted flow rates $x_{90} = (\mu_4, \mu_1, \mu_2, \mu_3)$. By analogous arguments, you can also perform a flip along the vertical axis (bottom row, left) and claim that the resulting coordinates y_{flipped} come along with flow rates $x_{\text{flipped}} = (\mu_2, \mu_1, \mu_4, \mu_3)$. Subsequent rotations and flips result in a total of seven additional virtual configurations per sample.

It is possible to derive a dedicated network architecture that incorporates the above-mentioned equivariance conditions explicitly, *i.e.*, for each fixed example with $f_\theta(x) = y$ your network will satisfy all of the following conditions

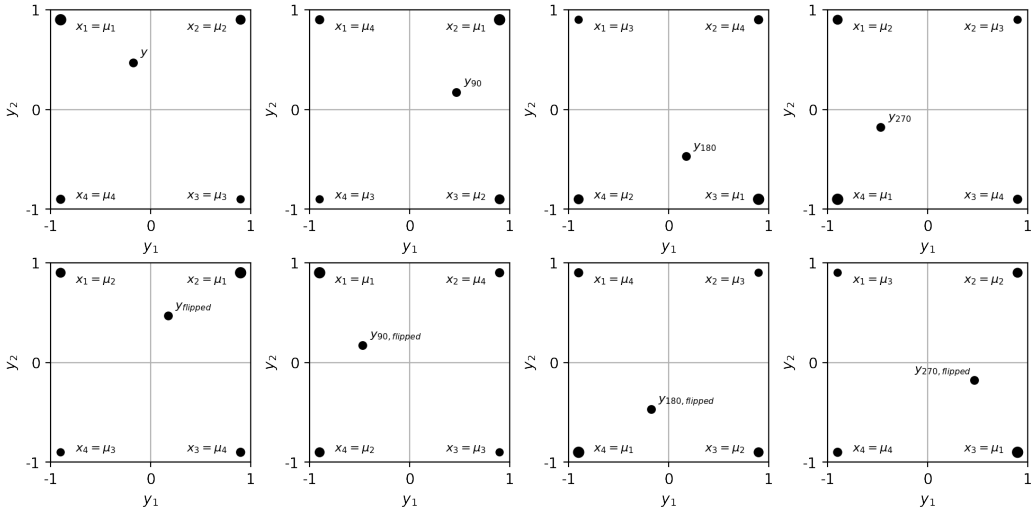

Figure 1: Motivation of equivariance conditions. Markersizes (in corners) indicate flow rates.

by construction:

$$
\begin{aligned}
f_\theta(x_{90}) &= y_{90} \\
f_\theta(x_{180}) &= y_{180} \\
f_\theta(x_{270}) &= y_{270} \\
f_\theta(x_{flipped}) &= y_{flipped} \\
f_\theta(x_{90,flipped}) &= y_{90,flipped} \\
f_\theta(x_{180,flipped}) &= y_{180,flipped} \\
f_\theta(x_{270,flipped}) &= y_{270,flipped}
\end{aligned}
$$

This is not true if you use standard architectures. However, it is possible to create also virtual training examples according to Figure 1 and use these to generate an augmented dataset. By using the augmented dataset for training, you can provide additional information also to standard networks.

In this project, your overall tasks are to implement, train and compare equivariant and standard network architectures for the task of leakage detection. To that end, you are going to use datasets of different sizes, also in combination with data augmentation. This will enable you to assess the additional value of prior knowledge that is incorporated directly through a dedicated network architecture and / or through virtual training examples.

For each training set size $m \in \{100, 1000, 10000\}$, proceed as follows.

1. Load the datasets
   `leakage_synth_dataset_train_[m].csv`
   `leakage_synth_dataset_validation_1000.csv`
   `leakage_synth_dataset_test_1000.csv`
   and store the data as NumPy-Arrays
   `X_train`, `Y_train`
   `X_validation`, `Y_validation`
   `X_test`, `Y_test`.

2. Train a standard fully connected neural network using the MSE loss (and the training data of course). Hyperparameters like number of epochs, batch size, optimizer, learning rate, depth and width of the network, and activation functions – all up to you. You can use a framework like KerasTuner to optimize hyperparameters, or simple `for`-loops if you prefer that. Anyway, you should use the validation data in order to evaluate and compare candidate hyperparameter configurations in terms of the yielded MSE. As soon as you have chosen a winner model, evaluate the performance of this one on the test data.

3. Augment the dataset. Each training example (x, y) can be complemented by seven additional virtual examples as illustrated in Figure 1. Clockwise rotations and flips on input data x are obtained as described above. A clockwise rotation on output data y is obtained through $y_{90} = (y_2, -y_1)$ and a flip along the vertical axis on the same data is represented by $y_{flipped} = (-y_1, y_2)$. All other operations (rotation angles and associated flips) can be computed by subsequent application of these two operations.

4. Repeat 2. on the augmented dataset.

5. Train an equivariant neural network on the original dataset. To that end, you need to implement a custom model using the Keras Subclassing API. All layers except the output layer must have a weight matrix that looks as follows

$$W^\ell \coloneqq \begin{bmatrix} a & b & c & b \\ b & a & b & c \\ c & b & a & b \\ b & c & b & a \end{bmatrix}$$

where a, b, c are the parameters that shall be learned. Moreover, equivariant layer have no bias vectors, *i.e.*, you have $b^\ell = 0$. The output layer of an equivariant network has a weight matrix

$$W^L \coloneqq \begin{bmatrix} d & -d & -d & d \\ -d & -d & d & d \end{bmatrix} \tag{1}$$

with only one parameter d, and no bias unit as well. Apart from these specifically structured weight matrices and the absence of bias vectors, equivariant networks look all like standard networks. However, they do not come as pre-implemented Keras layers, so you have to use the subclassing API. Once you have this, you can do anything else just as in case of the standard network.

6. Train an equivariant network on the augmented dataset.

If you repeat these steps on all three datasets, you finally have 12 winner models that you can compare in terms of their performance on the test data, so that you should be able to answer the follwing questions:

- Which model worked best depending on the size of the dataset?
- In which cases did data augmentation enhance the performace?

Finally, compare your models in terms of a visual inspection. This can, *e.g.*, be accomplished by visualizing model predictions on the following sets:

- $\{x \in \mathbb{R}^4 \mid x_1 + x_3 = 0.5,\, x_2 = x_4 = 0.25\}$
- $\{x \in \mathbb{R}^4 \mid x_2 + x_4 = 0.5,\, x_1 = x_3 = 0.25\}$

If your implementation of equivariant networks is correct, all associated predictions should lie on a diagonal of the coordinate plane.

Last but not least, Christoph Brauer will be available in the Question Time room in Big Blue Button as follows:

- Thursday, January 27th, 13:15–14:45
- Wednesday, February 2nd, 11:30–13:00
- Thursday, February 10th, 13:15–14:45
- Wednesday, February, 16th, 11:30–13:00
- Wednesday, February, 23rd, 11:30–13:00