A simple example to understand BatchNormalization layer as input layer

Ana Gorodisch[1], Ivan Anduaga Marchetti[2], Delfina Villeri[3], Elmer A. Fernández[4]

1 Data Scientist at Embryoxite
2 CSO at Embryoxite
3 CEO at Embryoxite
4 DataLab, Fundación para el progreso de la Medicina, CONICET

Corresponding Author: Elmer A. Fernández elmerfernandez at fpmlab.org.ar

Abstract

The introduction of the BatchNormalization (BN) layer in Deep Learning Neural Networks (DLNN) equalizes the distribution of input variables for the subsequent inner layer, enabling the use of higher learning rates and reducing the sensitivity to initialization. Typically, BN layers are used within DLNNs that are fed with pre-normalized training sets. However, their potential as input normalization layers to create end-to-end models, thereby minimizing the need for manual data preprocessing, remains underexplored. This study presents a simple example to demonstrate the impact of BN layers used at the input stage on model interpretability and optimization in deep learning.

Introduction

Data Normalization plays a vital role in overall data science applications, from statistics, to machine learning and Artificial Intelligence (Huang et al., 2023). The main and claimed advantage is that it allows fair comparison between input variables by means of centring and/or scaling, avoiding that one variable rules over the rest when variables have vastly different scales, since the direct comparison of raw values can be misleading. Normalization techniques, like z-scores or min-max scaling, put variables on a common ground, with the rationale of providing a fair and more meaningful comparison across different data points or groups. In this way data normalization in statistics provides similar weights to variables and helps ensure the data is prepared for meaningful comparisons, reduces the impact of outliers, allows for the use of specific statistical tests, provides equal variable weighting and ultimately facilitates clearer communication and interpretation of statistical results. On the other hand, in machine learning applications data normalization allows for

efficient distance calculations, improves convergence and stability of error correction learning based models and reduces the risk of overfitting. Likewise in deep learning models, data normalization also contributes to faster learning rates, avoids the large error gradients problem as well as it avoids the output saturation by large input values on the network layers.
In this way, centring and scaling is a common practice performed over the input data to such models. These centering and/or scaling procedures require the estimation of some population based statistics or metrics (for instance mean and standard deviation or min-max values) that may affect the results (González-Montoro et al., 2017) by mean and variance shifts from the population values.
In the field of Deep Learning Neural Networks (DLNN) it has been recognized that normalization strategies of the inputs of the inner network layers during learning, produce faster learning rates and better generalization capabilities by means of i) diminishing the vanishing effect of the activation functions and ii) by introducing the concept of covariance shift, referring the change in the distributions of internal nodes in the network (Ioffe et al., 2015). These effects are achieved by the introduction of the BathNormalization layer that tends to equalize the input variables distribution of the next inner layer. The BatchNormalization (BN) layer has been mainly used in between inner layers and it has been well studied (Bjorck et al., 2018). Even though it can also be used as an input layer, thus introducing the data normalization step into the neural network pipeline and, more importantly, into the learning process, their effect has not been fully addressed.
Here, by using a simple DL model to predict equilibrated urea for hemodialysis monitoring (Fernández et al., 2001), we show that using BN layer as input not only provides the expected normalization (i.e. making input variables comparable by the moving average based z-score normalization), but also, by the use of the shifting and scaling learned hyperparameters, it weighs the input variables accordingly to their importance on prediction, thus providing an important feedback for better understanding of the problem at hand and also for explanatory tasks crucial for feature engineering.
Source code:
https://github.com/elmerfer/DataLabLearningSeries/tree/main/BatchNormalization%20layer%20as%20Input%20Layer

Material and Methods

Patients

One hundred and thirty-three stable patients were selected from two units in Buenos Aires, Argentina (Unit A: RTC Monte Grande and Unit B: RTC Adrogue). All had undergone chronic HD treatment for at least 3 months.
The selection criteria were: (1) patients without infections or internations in the last 30 days; (2) all patients with A-V fistula with blood flow rate (QB) 6300 ml/min, and (3) written prior consent to the study.

All patients received HD three times a week with Baxter® machines (model 1550) with variable bicarbonate and sodium. Hollow-fiber polysulfone (Fresenius F6 and F8) and cellulose diacetate (FB170 and 210, Nissho Corp.) dialyzers were used.
For the purpose of this study, all patients were dialyzed over 240 min and blood (QB) and dialysate (QD) flows were fixed at 300 and 500 ml/min respectively. The mean age, mean predialysis body weight (preBW) and mean ultrafiltration (UF, difference between pre- and postdialysis body weights) of the population studied are shown in table 1.

Blood Sampling

All blood samples were obtained at the mid-week HD session. Four blood samples were obtained from the arterial line at different times for urea determinations: (1) urea predialysis (Upre), at the beginning of the procedure; (2) intradialysis urea (U120) at the middle of the HD session (120 min from the beginning); (3) urea postdialysis (Upost) at the end of the HD session, and (4) equilibrated urea (eqU) 60 min after the end of HD.
The Urea samples were determined as explained in Fernández et al., 2001

Urea Measurement

Three U determinations were made on each blood sample by autoanalyzers (Hitachi 704 in Unit A and Technicon RA 1000 Bayer in Unit B) and their average recorded. The coefficient of variation (CV) for the method was 1.08 and 1.11% respectively for the autoanalyzers.

Final data set

The used data set holds for 5 hemodialysis process variables $\left(U_0, U_{120}, U_{pos}, PP, UF\right)$ $\left(U_{eq}\right)$, with very different distribution characteristics (See Figure XXX), measured over 121 patients, This yield to the input matrix $X^{121x6} = \left[U_{i0}, U_{i120}, U_{ipos}, PP_i, Uf_i\right] \forall\, i = 1{:}\,121$ and the output vector $Y^{121x1} = \left[U_{ieq}\right]$.

BatchNormalization layer (BN)

It was introduced by Ioffe et al. in 2015 and is based on the normalization via mini-batch statistics. Assuming a training set of N samples, a mini-batch consists of M << N random samples of the training set. When the DLNN is stimulated by this min-batch ($\boldsymbol{B}$), a BN layer with d-dimensional input $x = \left(x^{(1)},\ldots, x^{(d)}\right)$ will normalize each k-th (k=1..d) input dimension according to $\hat{x}^{(k)} = \frac{x^{(k)} - \mu_\beta^{(k)}}{\sqrt{\sigma_\beta^{(k)^2} + \epsilon}}$, where $\mu_\beta^{(k)} = \frac{1}{M}\sum_1^M x_i^{(k)}$ and $\sigma_\beta^{(k)^2} = \frac{1}{M}\sum_1^M \left(x^{(k)} - \mu_\beta^{(k)}\right)^2$. This yield $\hat{x}^{(k)}$ to have zero mean and standard

deviation equal one. Then, each k-th BN output is calculated as $y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$ where γ and β are learnable parameters. At the end of the training process, the BN layer stores 4 parameters for each input, the moving mean $(\hat{\mu})$ and moving variance $(\hat{\sigma^2})$ and the γ and β parameters. The moving mean/variance are calculated as follows $\theta_{t+1} = \alpha\theta_t + (1 - \alpha)\mu_t$(where $\theta = \hat{\mu}$ or $\hat{\sigma^2}$) and α being the momentum learning rate parameter

When the BN is used as an input layer, its input dimensions equals the training set dimension, thus, during inference, it performs its BN transformation over the raw input data, but using the moving average/variance instead.

Experiments

DLNN optimal architecture

The proposed approach follows the one originally suggested by Fernandez et al.,which implies a sequential fully connected architecture with one linear output node. By following the seminal approach, here the normalized (z-score) input data was used as well as an end-to-end approach by using the raw data as input to the neural network model. This yields the evaluation of two DNN architectures: one comprising only fully connected layers and another with batch normalization layers as its first layer and following each hidden layer. To define the optimal architecture, a grid search process (see Table 1) was employed to optimize the number of hidden layers, the number of nodes in each layer, and the activation function for the hidden layers. In all cases, the output layer consisted of a single node with a linear activation function. For each combination of hyperparameters, an end-to-end model with and without batch normalization layers was evaluated, as well as a model with normalized input with and without batch normalization layers. A 5-fold cross-validation within the training set was utilized, selecting the best architecture based on the configuration that yielded the lowest Mean Squared Error (MSE). The best architecture for both the end-to-end model and the model with normalized input were identified and used for further explorations.

Hyperparameter	**Values**
Number of hidden layers	1, 2, 3, 4
Number of nodes in the hidden layers	2, 3, 4, 5, 6
Activation function of the hidden layers	tanh, relu, sigmoid
Activation function of the output layer	linear, relu

Table 1: Hyperparameters and its values used in order to find the optimum model

Sensitivity Analysis

In order to explore the effect of the Batch Normalization layer, several sensitivity analyses were carried out. With the aim of obtaining more robust results, the end-to-end model, described in the previous section, was trained on raw data using 10-fold cross-validation. The objective was to systematically "turn off" each input of the Batch Normalization layer to analyze the significance of its parameters by examining the output. Specifically, the relationship between the gamma parameter of BN and the mean squared prediction error (MSPE) of the network was explored by sequentially setting each variable to zero. This was done for both the first BN layer, where model inputs were individually "turned off" and set to zero, and for the BN layer following the hidden layers, where each of the 6 nodes was individually "turned off".

Results

BatchNormalization Layer allows a lesser number of learnable parameters

By building different DLNN architectures for equilibrated Urea regression for both z-score normalized inputs and raw inputs through a BatchNormalization input layer, the number and size of hidden layers was optimized by a grid search strategy (see table 1). The optimal DLLN architecture was the one achieving the lowest Mean Squared Error for both input types.

The final DLNN architecture for z-score normalized inputs (DLNNnorm) resulted in 2 hidden layers with 6 nodes each of them and a sigmoid activation function (i.e FC_6+FC_6+Out-1, see figure 1A) yielding 85 trainable parameters. Meanwhile the DLNN´s architecture fed with the raw input data (DLNNe2e) resulted in 1 hidden layer with 6 nodes with a sigmoid activation function (i.e BN+FC_6+BN+Oup_1, see figure 1B) yielding 87 parameters of which 65 were trainable ones, and 22 non-trainable. In both models the output layer consists of a single node with a linear activation function.

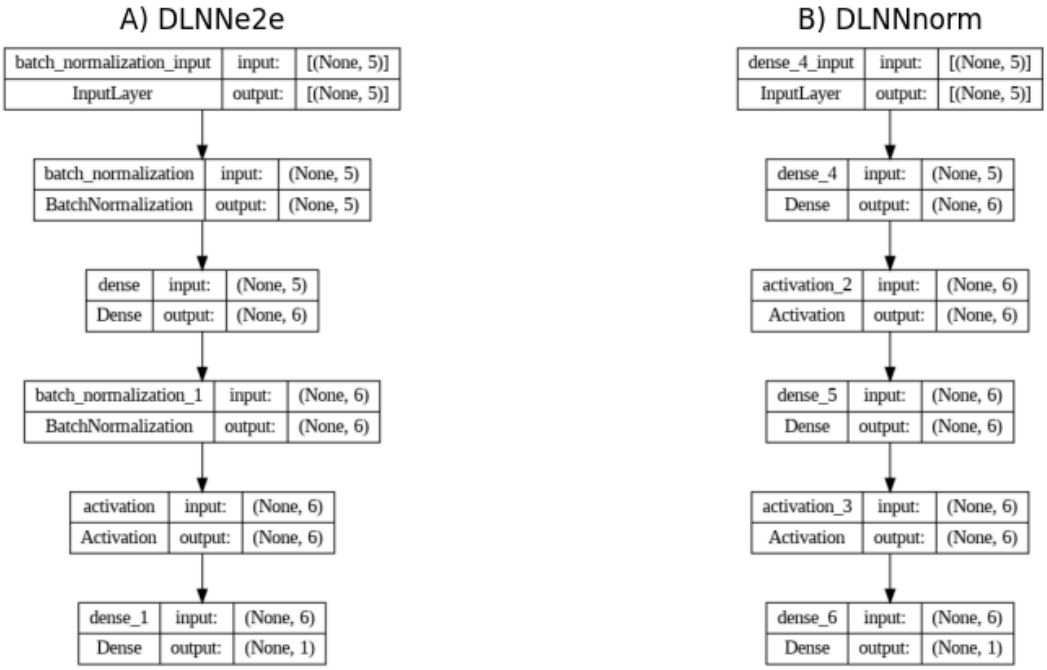

Figure 1: Final architecture of both DLNNe2e model (A) and DLNNnorm model (B)

Sensitivity analysis reveal that the BN scale parameter (gamma) is related to the input variable´s importance on prediction

Figure 2 shows the distribution of the test set Input variables, the z-score normalized ones, the Batch normalization output prior to scaling (γ) and offset (β), and the final BN input layer output (Figure 2A, Figure 2B, Figure 2C and Figure 2D respectively). It can be observed that, as expected, the moving average and variance normalization parameters were very similar to the z-score normalization ones (See table 2). However, the gamma scaling parameter weights the inputs by changing its ranges, thus proving different significance of the input for the next layer. In table 2 the normalization parameters from the z-score as well as the BN moving estimates, the scale (γ) and shifting/offset (β) parameter values are shown. It is possible to observe that the BN scale parameter (γ) is greater for Upos input variable.

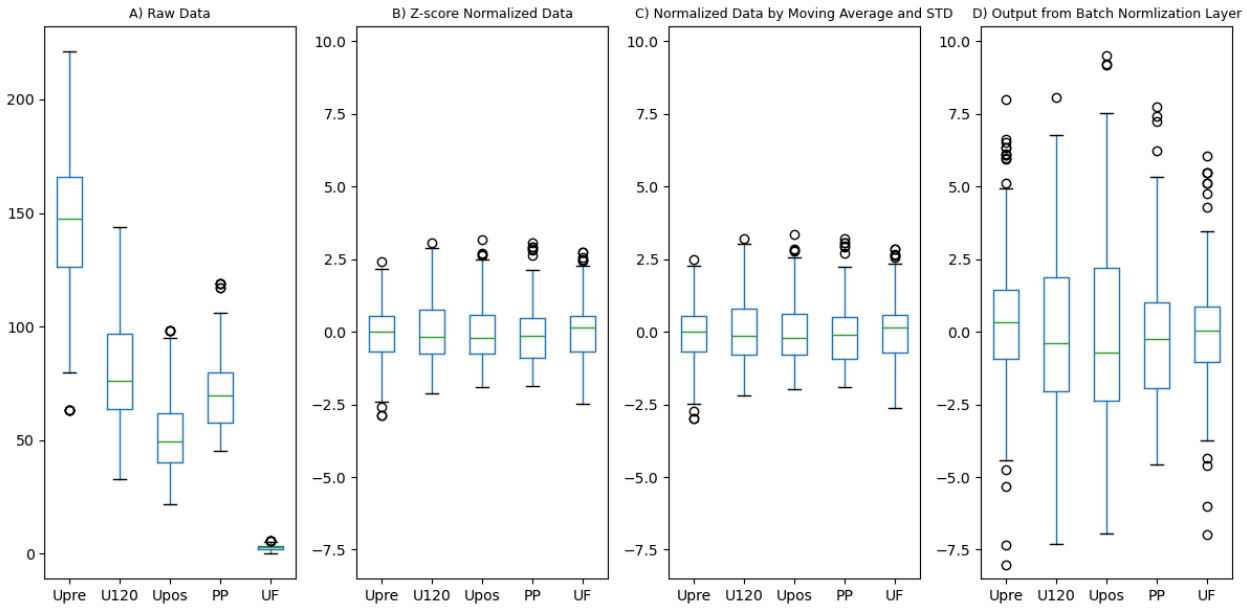

Figure 2: Distribution of the test set input variables for all 10 folds. A) Raw data, B) Z-score normalized data, C, Normalized data by the moving average and standard deviation from de Batch Normalization layer, and D) output from the Batch Normalization layer

	μ	std	beta	gamma	mean_scaler	std_scaler
Upre	148.89 ± 3.27	32.27 ± 1.15	1.60e-07 ± 2.66e-07	1.58 ± 1.75	149.09 ± 2.99	33.56 ± 1.19
U120	79.36 ± 2.16	22.03 ± 0.70	4.47e-08 ± 5.01e-07	1.61 ± 1.91	79.52 ± 1.88	22.89 ± 0.61
Upos	53.00 ± 1.55	15.94 ± 0.66	-3.61e-08 ± 9.67e-07	3.27 ± 0.57	53.19 ± 1.48	16.62 ± 0.61
PP	72.18 ± 1.21	16.13 ± 0.72	5.78e-08 ± 4.22e-07	2.06 ± 0.59	72.53 ± 1.22	16.80 ± 0.80
UF	2.65 ± 0.07	1.10 ± 0.05	-7.52e-08 ± 2.81e-07	1.39 ± 1.05	2.66 ± 0.08	1.15 ± 0.05

Table 2: Z-score normalization parameters as well as the BN parameters for each variable. The table depicts, for each value, its average and standard deviation across the 10 folds.

In order to evaluate the effect of each variable on prediction, we sequentially dropout variable inputs (by setting them as zero value), feed the model and evaluate the MSPE. In figure 4, it is possible to observe that the scale parameter (γ) of the input BN layer correlates with the MSPE. This suggests that γ provides information regarding the importance of the variables in prediction. Therefore, the γ BN values, modulate the variable inputs by weighting them, in this case, according to their importance on prediction, thus providing clues about the role of each variable.

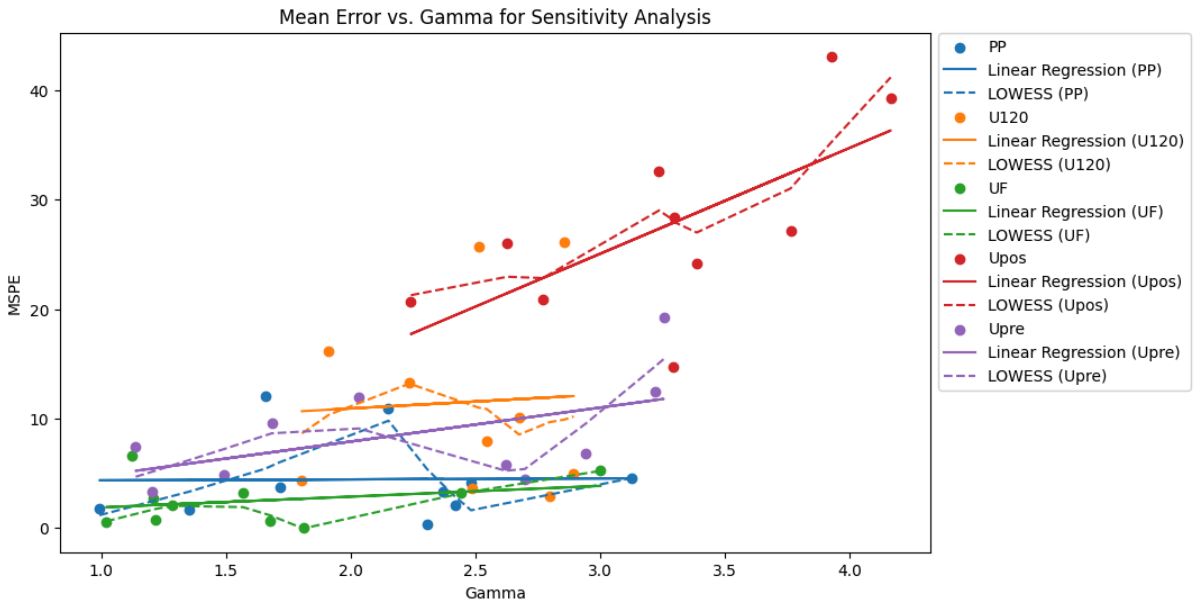

Figure 3: MSPE upon turning off each variable versus its respective gamma, done across all 10 folds.

The observed correlation between γ and the MSPE, is lost when the same model is trained with already normalized inputs as depicted in Figure 4.

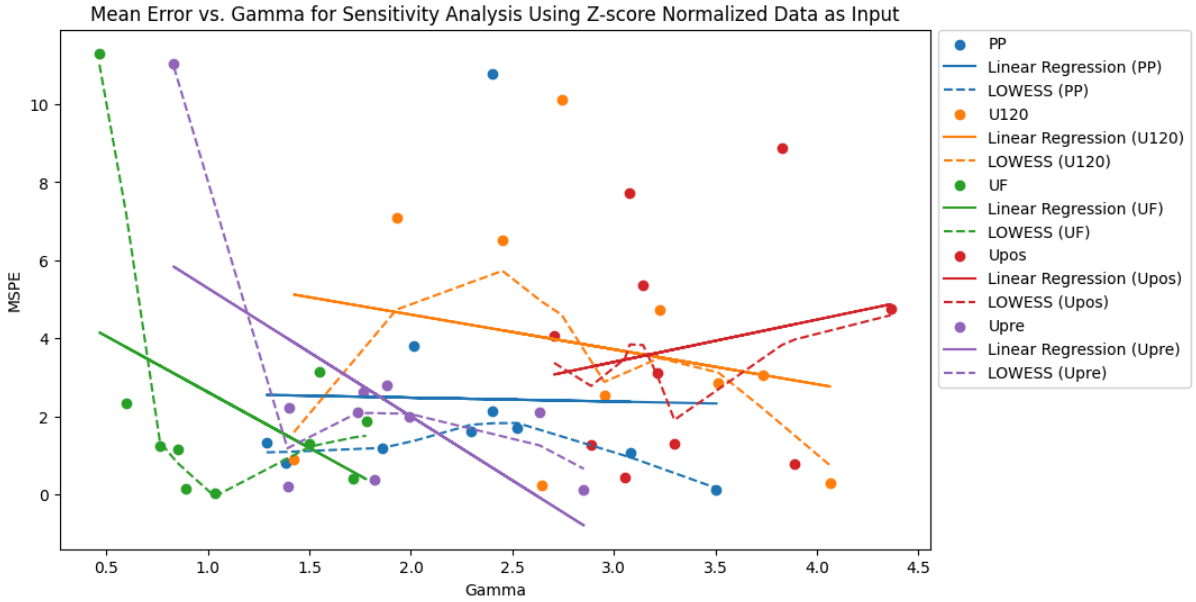

Figure 4: Using Z-score normalized data, MSPE upon turning off each variable versus its respective gamma, done across all 10 folds

Sensitivity on hidden layers

To further explore the effect of BN on its input variables, the end-to-end model incorporates two Batch Normalization layers (as depicted in Figure 1), i.e. one as its initial layer and a second one following its hidden layer. A sensitivity analysis similar to that conducted for the first Batch Normalization layer was performed for the second one. In this instance, the outputs of the hidden layer, which serve as input for

the Batch Normalization layer, were systematically dropped out (set to zero) to examine the relationship between their γvalues and the resulting MSPE. These findings are illustrated in Figure 6, revealing no apparent correlation between MSPE and the gamma value.

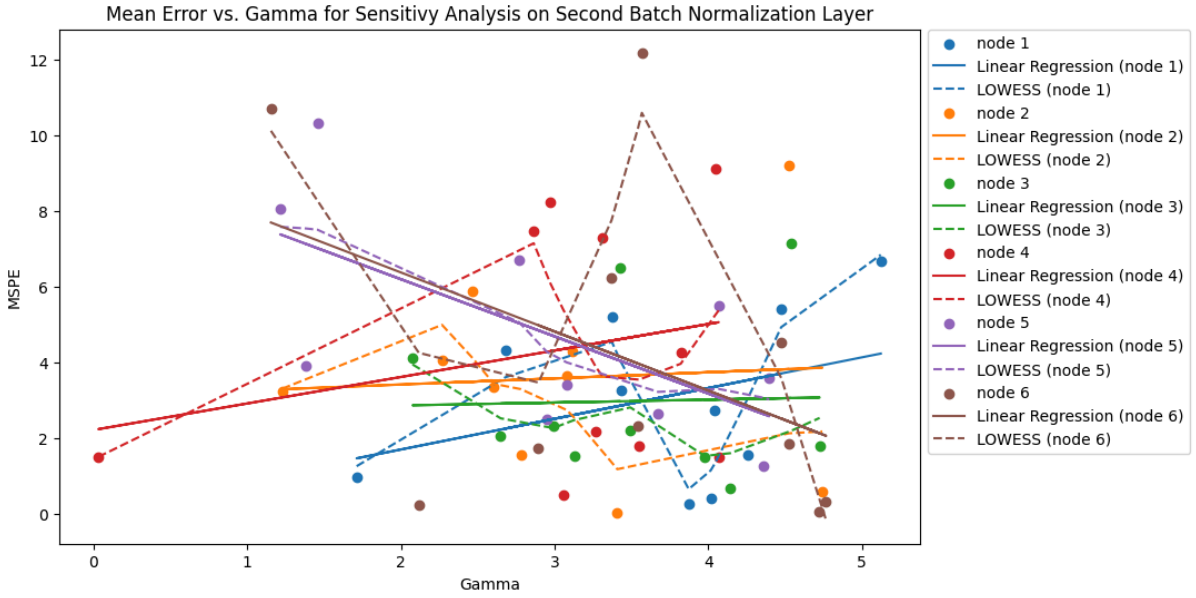

Figure 5: MSPE upon turning off each node versus its respective gamma, derived from the second Batch Normalization layer, done across all 10 folds

Discussion

Centering and scaling based on mean and standard deviation values are foundational preprocessing steps across various statistical, machine learning, and deep learning models. These techniques serve multiple purposes, such as preventing a single variable from dominating others and enhancing learning rate optimizations. As illustrated in Figure 2B and 2C, the normalization performed by the Batch Normalization layer closely resembles that of the Z-score normalization applied to the data prior to its input into the network. These findings are corroborated by Figure 3, which depicts that there is essentially no difference between the moving average used for data normalization with the BN and the mean used for normalizing the data with a Z-score normalization.

While it may seem sufficient for Batch Normalization (BN) layers to center and scale variables using batch estimates of moving averages and standard deviations in each learning cycle, the additional scaling and offsetting learning parameters introduced by BN significantly influence model performance [Santurkar et al]. Through sensitivity analysis, it is shown that BN goes beyond normalization, offering, in our case, insights into variable importance for predictions via scaling gamma parameters. When the DLNN is fed with raw data, Figure 4 shows a clear positive correlation between the gamma parameter of the BN layer for each variable and the MSPE when each variable is set to zero. This suggests that the higher the gamma, the

higher the importance of the variable for the prediction task, rendering UPos as the most influential variable for the determination of equilibrated urea. In this manner, we observe that centering and scaling need not be exactly zero and one, as in traditional z-score normalization; instead, utilizing gamma and beta, BN provides variable importance and offset shifts, potentially impacting the net output of subsequent processing layers.
However, this behavior is not observed when the data is normalized prior to its input into the network nor for the second BN layer when each of its input nodes are set to zero. This implies that the scaling of the BN layer is highly dependent on the actual values of its inputs. When its inputs are normalized and there is not much disparity between their range of values, the gamma parameter is unable to indicate which variables are most influential in fulfilling the task at hand.

Conclusión

The incorporation of a Batch Normalization (BN) layer as an input layer introduces additional parameters that play a crucial role in enhancing model performance. Through sensitivity analysis, where each input variable was systematically removed at a time, it was observed that the learning parameter gamma in BN correlates with inputs that have a more significant impact on prediction errors. This underscores the importance of BN not just as a normalization layer but as a tool for identifying influential input variables. Moreover, BN's flexibility in centering and scaling, represented by gamma and beta, allows for nuanced adjustments that can impact the overall output of subsequent processing layers. These findings highlight BN's role in enhancing interpretability and optimization in deep learning models right from the input stages.

References

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). Understanding Batch Normalization. https://doi.org/https://doi.org/10.48550/arXiv.1806.02375

Fernández, E. A., Valtuille, R., Willshaw, P., & Perazzo, C. A. (2001). Using artificial intelligence to predict the equilibrated postdialysis blood urea concentration. Blood Purification, 19(3), 271–285. https://doi.org/10.1159/000046955

González-Montoro, A., Prato, L., Casares, F., Balzarini, M., & Fernandez, E. (2017). Appropriate sample size for standardization parameters estimation reduces misdiagnoses of molecular-based risk predictors in breast cancer. Medical Science and Technology, 58, 111–118. https://doi.org/10.12659/mst.905935

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2023). Normalization techniques in training dnns: Methodology, analysis and Application. IEEE

Transactions on Pattern Analysis and Machine Intelligence, 45(8), 10173–10196. https://doi.org/10.1109/tpami.2023.3250241

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift. https://doi.org/https://doi.org/10.48550/arXiv.1502.03167