Deep Learning: Fully Connected Neural Networks. Laboratory No° 2

Ing. Andrés Manuel Prieto Álvarez <andres.prieto@utp.edu.co>
Maestría en ingeniería de sistemas y computación
Universidad Tecnológica de Pereira

Abstract - This laboratory study focuses on the Fully Connected Neural Network architecture (FCNN), exploring its applications in classification tasks. A comprehensive investigation was conducted on the design and implementation of the network, demonstrating its efficacy in discerning patterns within diverse datasets. The classic XOR problem was addressed, showcasing the network's capability to capture non-linear relationships. Furthermore, regression tasks were carried out using a mathematical approach, taking advantage of matplotlib, numpy and python. The analysis of various datasets provides insights into the nuanced performance of neural networks, emphasizing their versatility in both classification and regression domains.

Keywords - Fully Connected, Neural Network, Regression, Classification, Pattern Recognition

I. MATERIALS AND METHODS

A. Google Colab

Google Colab, short for Colaboratory, is a free, cloud-based platform provided by Google that allows its users to write and execute Python code in a web-based environment. It's a part of the Google Drive suite of applications. This environment was used to handle the entire development process of the study. Core features used:

- Online Python Environment
- Free GPU Access
- Interactive Visualizations

The following figures were generated using Google Colab, also some code snippets were taken from source code created on Colab too.

B. Datasets

To develop this laboratory 3 different datasets were used, 2 of them were used for classification problems and the other one was used for regression problems.

1) XOR gate - XOR gate, is a digital logic gate that outputs true (or 1) only when the number of true inputs is odd. In other words, it produces a true output if the number of true inputs is one. If the number of true inputs is even, the XOR gate outputs false (or 0). We can see a simple example at table 1.

Input	Output
[0, 0]	0
[0, 1]	1
[1, 0]	1
[1, 1]	0

Table 1. XOR

2) “DataLeastSquareClasificationtTwoClass” - This dataset represents a conventional classification problem, this datasets own 2 separable tags or classes (1, & -1). This dataset was used in the classification FCNN practice. The dataset was divided in 2 different sets, a training set represented for dot or circles and a testing set represented for triangles; a representation of this dataset

merged for representations purposes can be found at figure 1.

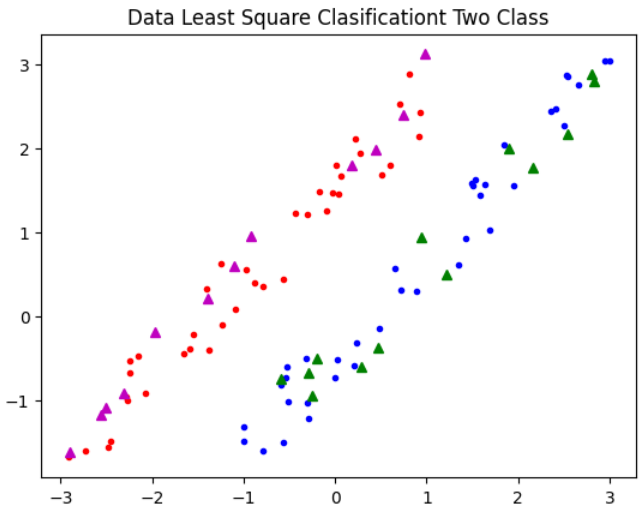

Figure 1. Data Least Square Classification Two Class

3) “DataXsquareNeuronalNetwork” - This dataset is pretty different from previous two, it was used for a regression analysis, so the nature of the data changed in context and meaning, when a FCNN is doing a regression analysis the expected output is a prediction for some specific data a prediction in short terms, due to this characteristics, the representation of this dataset diverge from previous two, in figure 2 there is a simple representation of 2 section of the data set green triangles were used for test while magenta dots for training.

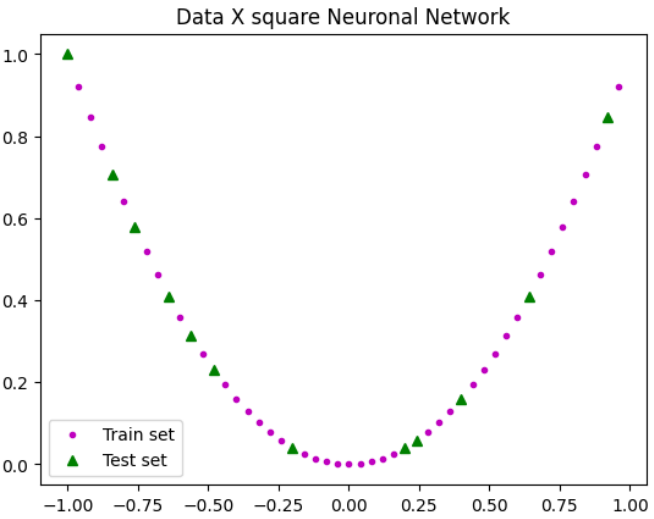

Figure 2. Data X square Neuronal Network

4) “DataHeavisideNeuronalNetwork” - This is the weirdest dataset used for the study, this dataset contains a step-like function representation of the data spore associated with each dataset, also, when you see the tags used in this dataset for each pair X, Y; it could be misinterpreted as a two-class classification problem, something similar to figure 1 or table 1; since almost all the data in this dataset is characterized for being some input in X having a 1.0 or 0.0 in its pair Y. Due to this particularity the initial analysis of the data can be harder, actually, contrasting the time dedicated to analyze each dataset this particular one took significantly a longer period of time than the other 4. The elongated timeframe is attributed to the dataset's unique structural aspects, demanding meticulous scrutiny to unravel its complexities fully.

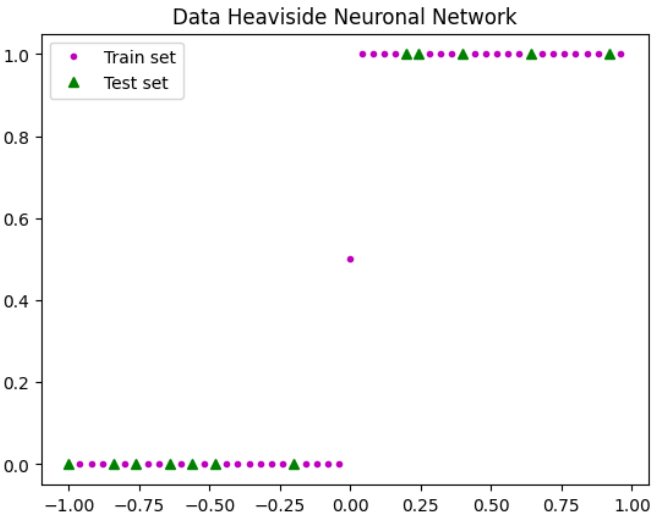

Figure 3. Data Heaviside Neuronal Network

5) “DataAbsoluteNeuronalNetwork” - This dataset is closer to figure 2 than figure 3, just like previous figure 2, triangles belong to testing set, while circles belong to training. This dataset reminds a lot to an absolute function in fact, the data this dataset proposes could have been taken from an absolute function, which is a good case of study for the regression case of the FCNN, since there is an equivalent between an input (X) also an expected output (Y), so this case can be solved using the FCNN’s

regression case. It is also curious to note that the representation of this dataset is a bit similar to the letter “V”.

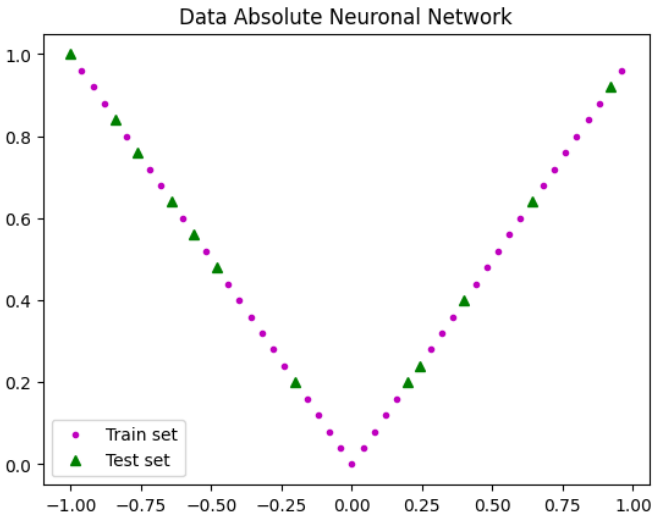

Figure 4. Data Absolute Neuronal Network

C. Neural Network Architecture

A Fully Connected Neural Network, also known as a dense neural network or multi-layer perceptron (MLP), is a fundamental type of artificial neural network architecture. It consists of multiple layers of nodes, where each node in a layer is connected to every node in the subsequent layer. The layers are typically organized into an input layer, one or more hidden layers, and an output layer. This architecture is explained in the following section of pseudo code:

```
Parameters
Initialize weights and biases
Initialize weights and biases for hidden layers
Initialize weights and biases for output layer
Training
for epoch in range(epochs):
    for i in range(len(training_data)):
        Forward pass Calculus
        Backward pass (Backpropagation)
                Error calculus

        Update weights and biases
// Repeat until convergence
```

This pseudocode provides a basic sequential structure for a fully connected neural network, including initialization, forward pass, and training using a simple gradient descent approach. The actual implementation details, such as the choice of activation functions, loss functions, and optimization algorithms, would depend on the specific requirements of the corresponding task.

D. XOR Problem

The XOR problem is not linearly separable, and a single-layer perceptron lacks the capacity to capture the non-linear relationships required to correctly classify XOR inputs. To solve the XOR problem, more complex neural network architectures such as FCNN.

A Fully Connected Neural Network (FCNN) can solve the XOR problem because it has the capacity to learn and represent non-linear relationships between input features, enabling it to capture the complex pattern inherent in the XOR function, which a single-layer perceptron, for example, cannot achieve due to its limitation to linear decision boundaries.

II. RESULTS

A. XOR

Unlike simpler models, the FCNN's multi-layered structure with interconnected nodes allows it to learn and navigate non-linear relationships, making it adept at capturing the XOR pattern; this behavior can be quickly observed by checking figure 5 in which loss is compared with epochs. To achieve this behavior a FCNN was implemented using an XOR as a dataset, also following previous descriptive pseudocode.

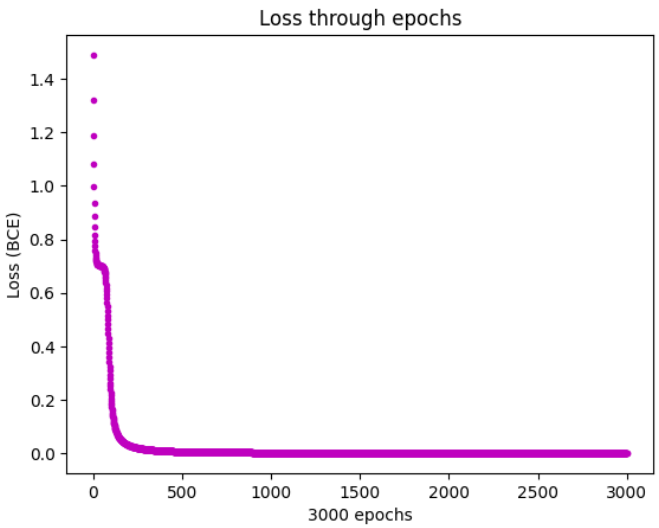

Figure 5. XOR Loss

This FCNN used 2 inputs as X1 & X2 as an array pair [X1, X2] introducing data as a the XOR table previously mentioned; 2 hidden layers and 1 output layer, using hyperbolic tangent as activation function in each layer but the last using a sigmoidal function for the output layer, this implementation used a BCE as loss function, since this is a classification FCNN. The hyperparameters of this implementation were setted to 0.3 as learning rate, 3000 for iterations and 3 layers. It must be noted that 3000 iterations are a bit exaggerated as it can be seen in figure 5; the convergence is achieved around iteration 1000, there is not a big difference or improvement after that epoch. This overdone behavior was done on purpose to see a better "tail" in the loss graph during the training process.

The following table shows a summary of the training process, showing the predictions.

Input	Expected	Prediction
[0, 0]	0	0.00057248
[0, 1]	1	0.99931117
[1, 0]	1	0.99932748
[1, 1]	0	0.00057248

Table 2. XOR Predictions Verification

As seen in table 2 the XOR table matches the dataset, the prediction also matches the expected behavior. This prediction means that the XOR problem is being solved. Anyway, to analyze the metrics in a better way, table 3 breaks down the metrics this FCNN got.

Metric	Value
Precision	1.0
Sensibility	1.0
Specificity	1.0
Accuracy	1.0
F1 score	1.0

Table 3. Metrics XOR Neural Network

The metrics obtained from the XOR problem showcase a model with exceptional performance. With precision, sensibility, specificity, accuracy, and F1 score all reaching a perfect score of 1.0, the model demonstrates impeccable ability in correctly identifying both positive and negative instances. Precision reflects the accuracy of positive predictions, sensibility highlights the model's capability to capture all positive instances, specificity underscores its accuracy in identifying negative instances, accuracy provides an overall measure of correct predictions, and the F1 score harmonizes precision and sensibility. The uniformly high scores across these metrics signify a robust and reliable performance of the Fully Connected Neural Network in successfully tackling the XOR problem.

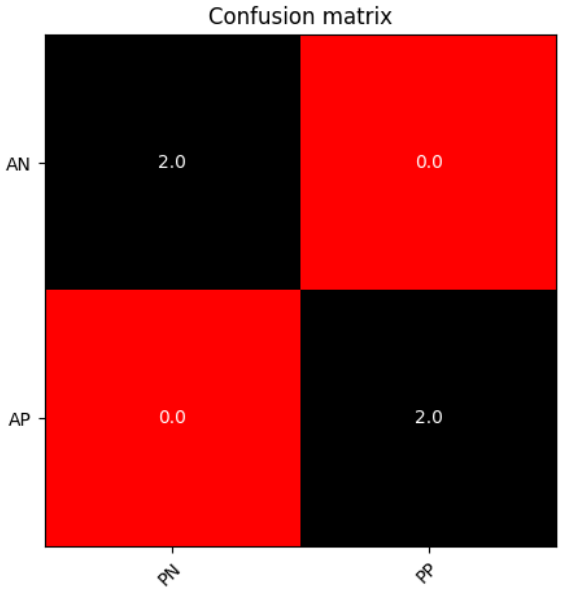

Figure 6. XOR Confusion matrix

The confusion matrix for the XOR problem reveals a model with discriminatory capabilities. In the context of binary classification, the matrix depicts four key metrics: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). In this case, all metrics exhibit optimal values. The model accurately identifies all positive instances (TP), correctly recognizes all negative instances (TN), avoids any false positive identifications (FP), and eliminates false negatives (FN). This symmetrical matrix, with all values aligning to a perfect classification scenario, underscores the reliable performance of the FCNN in successfully solving the XOR problem.

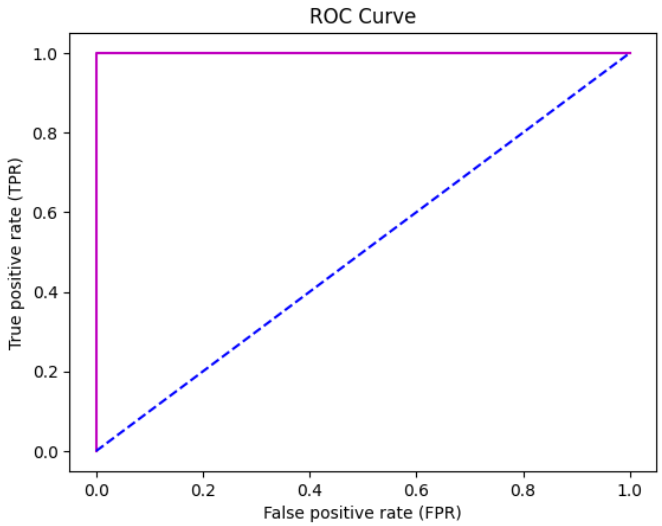

Figure 7. XOR ROC Curve

As a side complementary metric, the Receiver Operating Characteristic (ROC) curve for the XOR problem graphically illustrates the discriminatory ability of the FCNN. The curve exhibits a characteristic upward leftward trend, showcasing the model's ability to achieve high sensitivity without sacrificing specificity. The area under the ROC curve (AUC) further quantifies this performance, and in this case, it attains a score of 1.0 (Unfortunately, the AUC was not able to be added in the figure, due to a matplolib issue with colab). This implies that the model effectively distinguishes between positive and negative instances without any trade-off, making the ROC analysis a testament to the outstanding classification prowess of the Fully Connected Neural Network in addressing the complexities of the XOR problem.

B. Classification Fully Connected Neural Network

This section delves into the capabilities of the Classification FCNN. Using a similar FCNN to previous XOR FCNN, the focus here is on the context of classification throughout the exploration of its layered architecture. Having in mind that the data here is different from previous XOR “dataset”, which was closer to a true false table than a dataset itself, figure 8 shows the reliability of the FCNN, as this figure shows, XOR which is a way simpler than the dataset used here, loss seems to be pretty similar

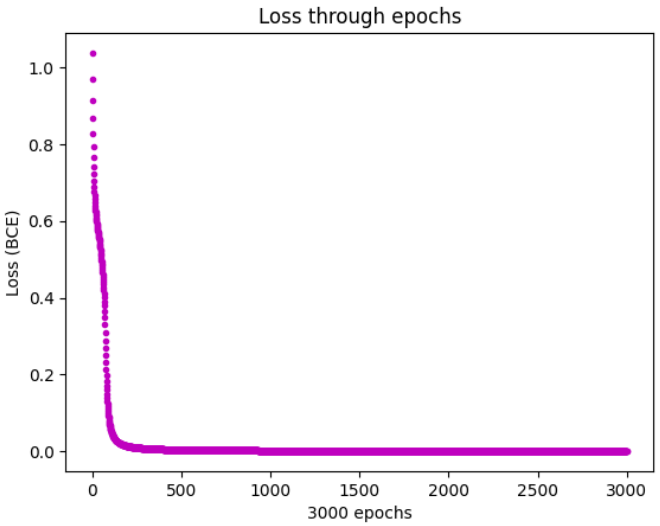

Figure 8. Classification Loss

It’s important to note that figure 6 shows loss for a BCE or “Binary Cross Entropy” as a classification-like dataset this loss function is the only loss function seen in class that could match the dataset “Data Least Square Classification Two Class”, this dataset was splitted in 2 sub datasets using a 75 x 3 matrix (75% of the dataset) for training, to check the FCNN training process, figure 9 was plotted. In this figure can be seen the decision plane in the linear case of feed-forward model.

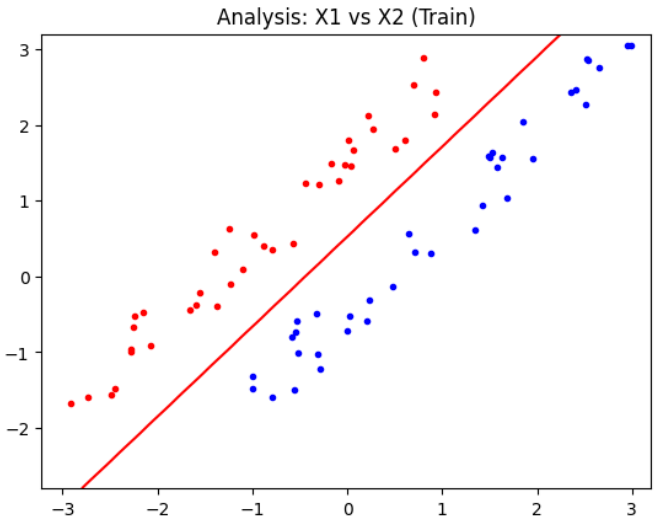

Figure 9. Classification Train Stage

To make a comparison of the testing process the 25 x 3 matrix (25% of the dataset) was used for testing the training process using its respective decision plane. See figure 10.

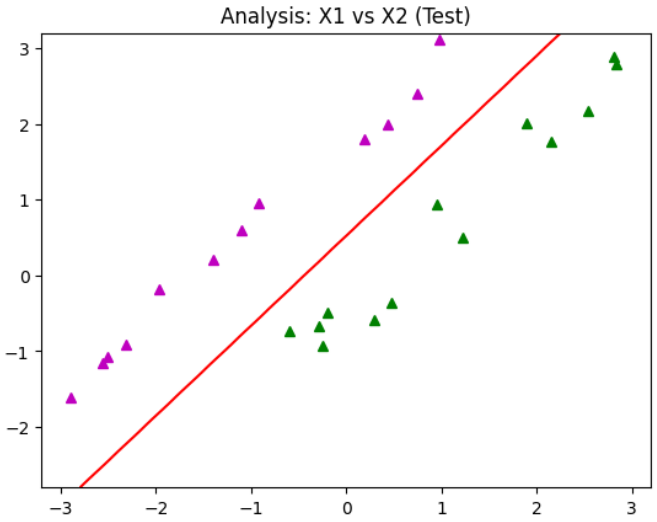

Figure 10. Classification Test Stage

Previous seen results for figures 10 and 11 were achieved using a learning rate of 0.3, 3000 iterations and a Classification Fully Connected Neural Network, using an architecture of 2 inputs X1, X2; 2 neurons in the input layer, 2 neurons in the hidden layer and 1 output layer; the architecture used 2 hyperbolic tangent for each layer and a sigmoidal activation function in the output layer. It is important to note that gradient for bias and weights were initialized as 0, while bias and weight were initialized using a numpy random rand function using that took FCNN parameters as input for the function

Metric	Value
Precision	1.0
Sensibility	1.0
Specificity	1.0
Accuracy	1.0
F1 score	1.0

Table 4. Metrics Classification Neural Network

The specific architectural configuration descriptive previously for the Classification FCNN delves into the evaluation of metrics that encapsulate its performance. Notably, precision, sensibility, specificity, accuracy, and the F1 score all exhibit a score of 1.0. Precision reflects the accuracy of positive predictions, sensibility highlights the model's capability to capture all positive instances, specificity underscores its accuracy in identifying negative instances, accuracy provides an overall measure of correct predictions, and the F1 score harmonizes precision and sensibility. This remarkable alignment of metrics at the pinnacle signifies the FCNN's proficiency in classification tasks, reaffirming its robustness in achieving accurate and balanced predictions across various data scenarios (At least the tested ones).

The confusion matrix, as a tool for assessing the Classification FCNN's performance, succinctly encapsulates the model's discriminative capacity. In this binary classification context, the matrix showcases values for True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Showing the model's balance. See figure 11.

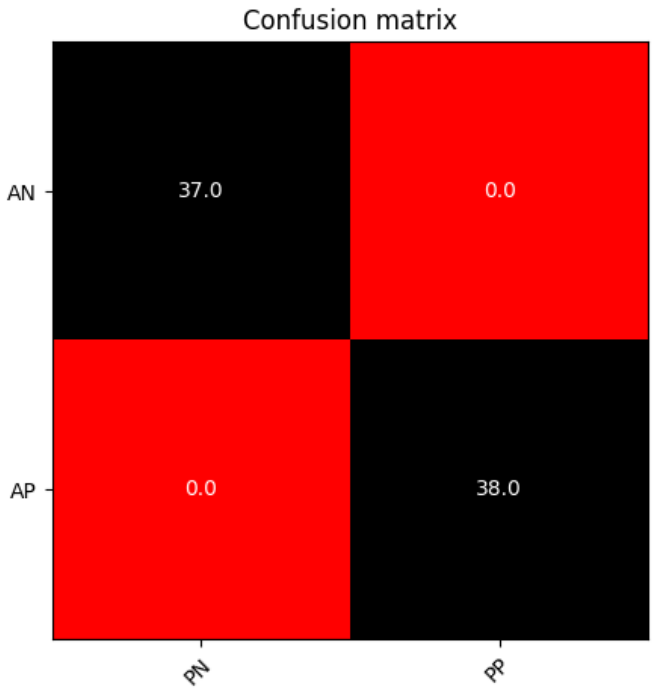

Figure 11. Classification Confusion Matrix

Figure 11, presents the confusion matrix, succinctly captures the classification outcomes of the model. In a binary classification scenario, the matrix indicates that there are 37 instances correctly predicted as positive (True Positives) and 38 instances correctly predicted as negative (True Negatives). Notably, there are no instances of false positives (incorrectly predicted positive) or false negatives (incorrectly predicted negative), resulting in a matrix with perfect values. This signifies the Classification FCNN's ability to make precise and accurate predictions for both positive and negative classes, highlighting a robust and reliable performance.

The ROC curve provides a graphical representation of the Classification FCNN's discriminatory ability. In this binary classification context, the curve illustrates the trade-off between sensitivity and specificity, showcasing the model's ability to achieve high sensitivity without compromising specificity. See figure 12

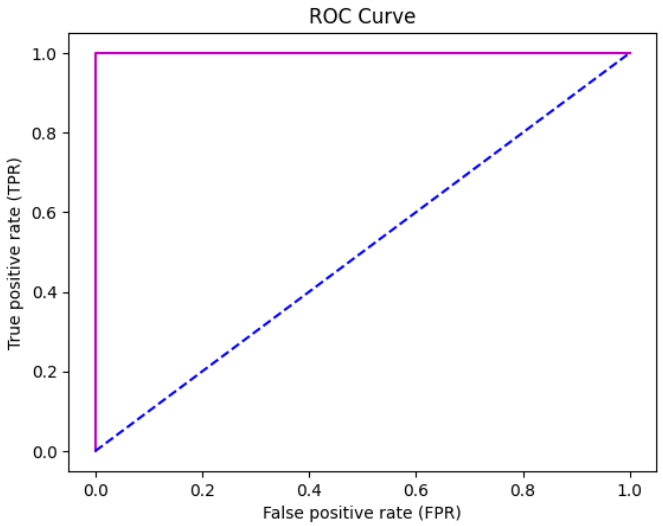

Figure 12. Classification ROC Curve

The curve's ascent towards the top-left corner emphasizes the FCNN's excellence in distinguishing between positive and negative instances. With a area under the ROC curve “AUC” of 1.0 (Unfortunately the AUC was not added to the plot due to matplotlib issues), this analysis reinforces the model's exceptional classification performance, affirming its reliability in navigating nuanced classification scenarios

C. Regression Fully Connected Neural Network

1) General description

In the examination of the Regression Fully Connected Neural Network (FCNN), attention is directed toward its versatile application in addressing regression tasks. Employing a multi-layered architecture with interconnected nodes, the FCNN excels in predicting continuous output values. This section systematically explores the inner workings of the Regression FCNN, shedding light on its capacity to navigate diverse datasets and model continuous variables. The objective is to present a comprehensive understanding of the Regression FCNN's role and effectiveness in the domain of regression, highlighting its adaptability and precision in making predictions.

As a disclaimer, all but 3 second implementations use an architecture in which 38 x

2 matrices (75% of each dataset) were used for training purposes while matrices of 12 x 2 (25% of the corresponding dataset) were used for testing each model.

The hyperparameters governing each neural network's training and optimization processes are systematically configured to strike a balance between efficiency and accuracy. With a single input, 38 data points, and one output, the network is poised to effectively handle the intricacies of the given datasets. The learning rate, set at 0.3, delineates the step size during the iterative optimization process, influencing the model's ability to converge effectively. A total of 3000 iterations for each model but the third one, due to a small issue that will be explained later, provides the network with ample opportunities to learn and adjust its parameters to minimize loss. The architectural design, comprising three layers, guides the flow of information through the input, hidden, and output layers. This thoughtful selection of hyperparameters, encompassing input dimensions, data size, output specifications, learning rate, iteration count, and layer configuration, collectively orchestrates a comprehensive framework for the FCNN's training and subsequent predictive tasks.

The neural network architecture employed in this study is systematically structured to accommodate the intricacies of the given tasks. Beginning with the input layer comprising two neurons, the subsequent layers unfold with precision. The input layer seamlessly connects to the first hidden layer, characterized by two neurons. This layer, in turn, feeds into the second hidden layer, maintaining consistency with two neurons. The architecture culminates in the output layer, featuring a single neuron. This structured configuration, denoted as [2, 2, 2, 1], delineates the sequential organization of neurons across the input, hidden, and output layers, respectively. The purposeful arrangement of neurons in this architecture aligns with the model's capacity to learn and adapt to the complexities embedded in the datasets under scrutiny.

The activation functions assigned to each layer within the neural network further enhance its adaptability and performance. Adhering to a systematic approach, the first hidden layer employs the hyperbolic tangent (tanh) activation function, fostering non-linearity and enabling the model to capture intricate patterns within the data. The subsequent hidden layer seamlessly continues with the tanh activation function, maintaining consistency in the approach to non-linear transformations. The choice of the linear activation function in the output layer signifies the regression nature of the network, facilitating the generation of continuous predictions. This meticulous alignment of activation functions, denoted as [tanh, tanh, linear], contributes to the nuanced capacity of the FCNN to navigate diverse datasets and optimize for both classification and regression tasks.

The concept of loss holds centrality in the evaluation of neural network performance. In the context of this study, loss represents the quantitative measure of the model's deviation from actual values during training. It serves as an indicator, guiding the optimization process by quantifying the disparity between predicted and true values. The reduction of loss is the overarching goal during the training phase, achieved through iterative adjustments to the model's parameters. This analysis incorporates loss as a fundamental metric, providing insights into the convergence and efficacy of the FCNN across different datasets.

2) Specific analysis of the models

a) Data square Neuronal Network

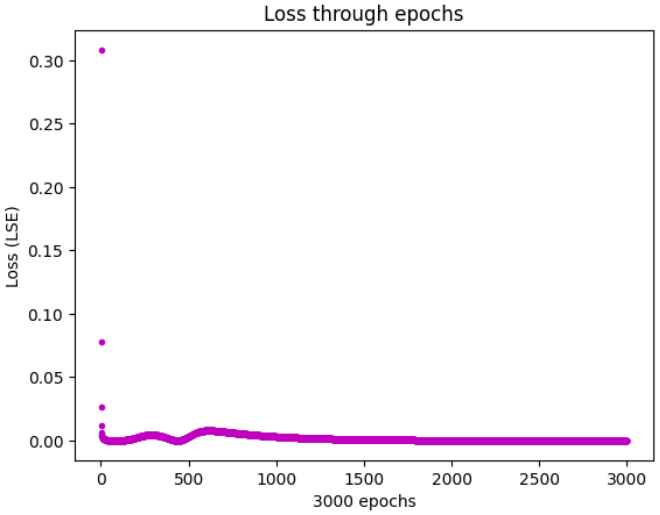

Figure 13. Square Loss

The Root Mean Square Error (RMSE) value of 0.42957316550596225 serves as a quantitative measure of the model's predictive performance in the context of regression. RMSE represents the square root of the average squared differences between the predicted values and the actual values in the dataset. A RMSE of 0.42957316550596225 implies a relatively accurate and precise regression model, with small prediction errors on average across the dataset.

Figure 14 and Figure 15 shows the training process for the "DataXsquareNeuronalNetwork" dataset providing valuable insights into the evolution of the FCNN during optimization. The graph, generated using Matplotlib in Google Colab, visually illustrates the dynamic nature of the model and its prediction performance while training.

.

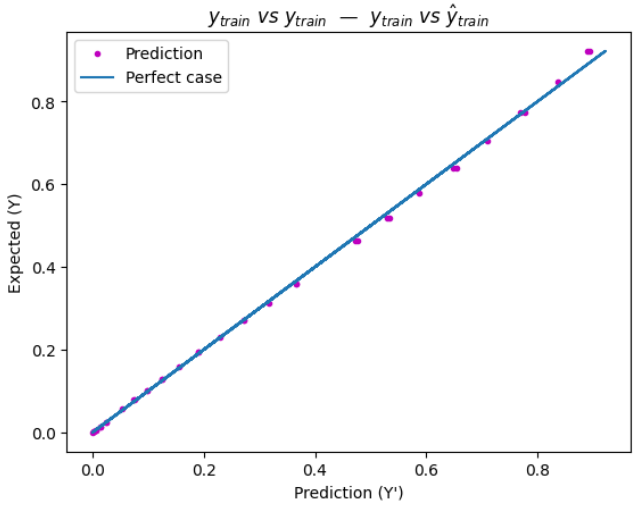

Figure 14. Square - Y vs Y & Y vs YP (Train)

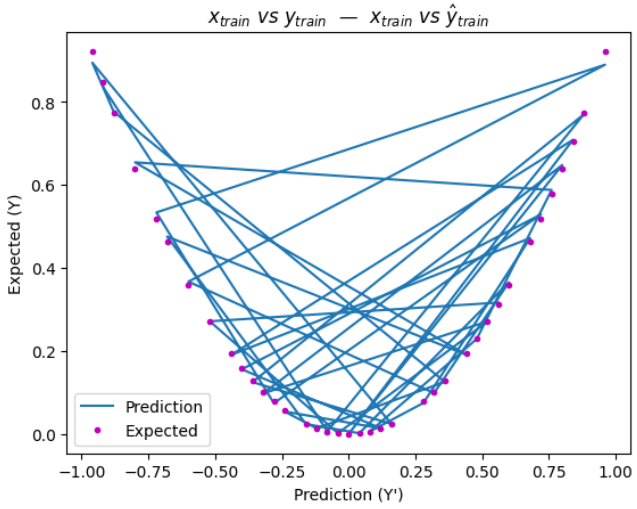

Figure 15. Square - X vs Y & X vs YP (Train)

Figure 16 and Figure 17 shows the testing process for the "DataXsquareNeuronalNetwork" dataset encapsulates the FCNN's performance in testing stage. Constructed using Matplotlib in Google Colab, the graph visually portrays the model's behavior on unseen data. As the FCNN generalizes its learned patterns to the test set, the graph captures a snapshot of the model's adaptability.

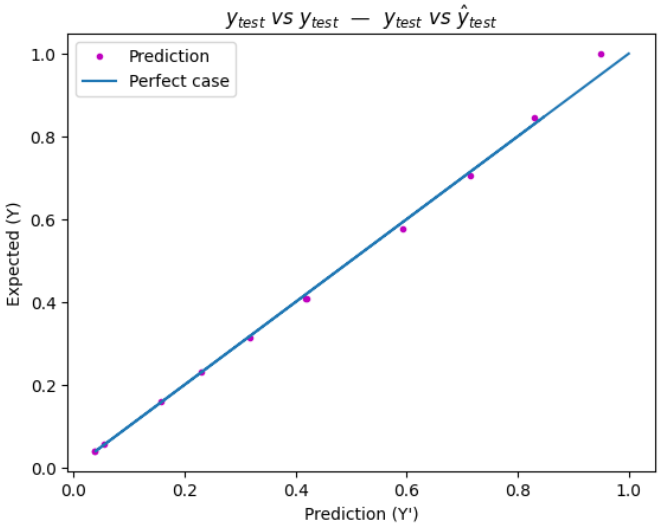

Figure 16. Square - Y vs Y & Y vs YP (Test)

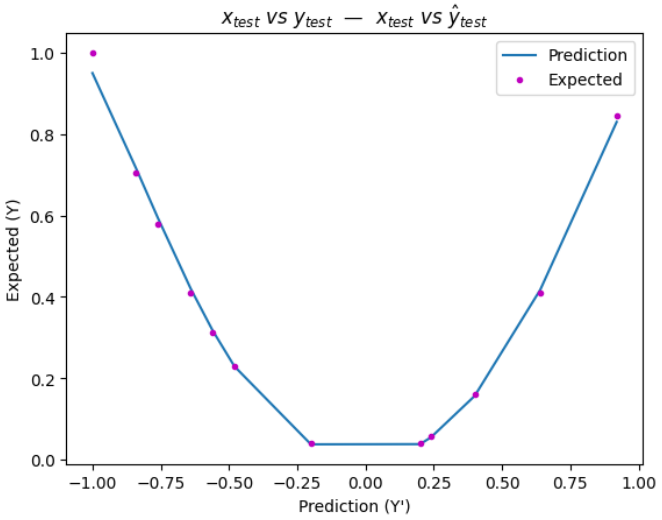

Figure 17. X vs Y & X vs YP (Test)

b) Data Heaviside Neuronal Network

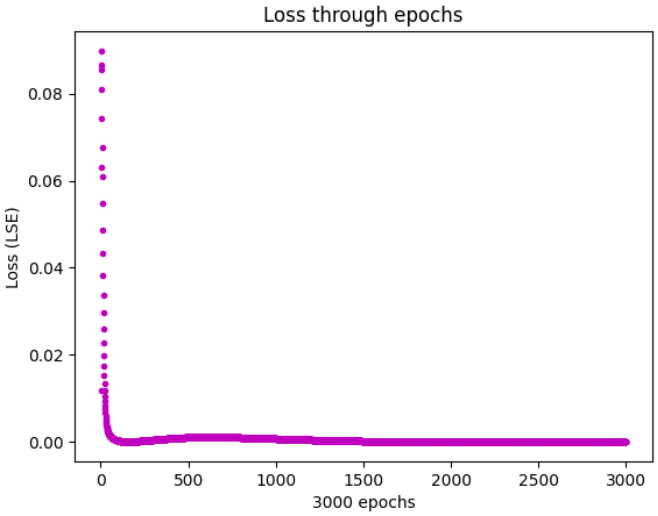

Figure 18. Square Loss

The Root Mean Square Error (RMSE) value of 0.28304695870388324 serves as a quantitative measure of the model's accuracy in predicting continuous values within a regression context. Specifically, this RMSE value indicates that, on average, the model's predictions deviate by approximately 0.28 units from the actual values in the dataset. A RMSE of 0.28304695870388324 implies a high level of accuracy and precision in the regression model, with relatively small prediction errors on average across the dataset

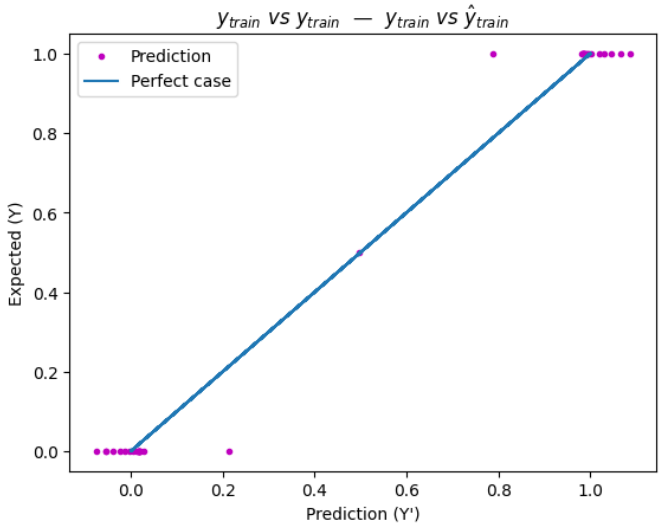

Figure 19. Square - Y vs Y & Y vs YP (Train)

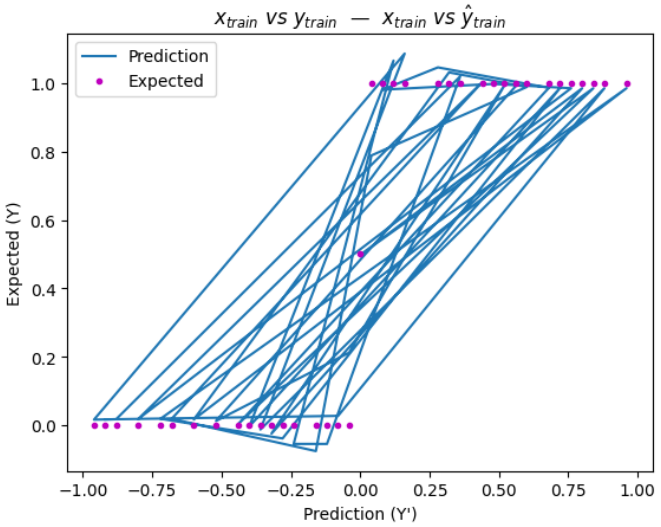

Figure 20. X vs Y & X vs YP (Train)

The "DataHeavisideNeuronalNetwork" dataset introduces a unique challenge, characterized by a step-like function representation and an unconventional distribution of data points. The graphically complex nature of this dataset, which predominantly features binary outcomes, requires meticulous scrutiny to unravel its structural intricacies. The FCNN process of training under these conditions can be seen in figures 19 and figure 20. While figure 21 and figure 22 show the testing process.

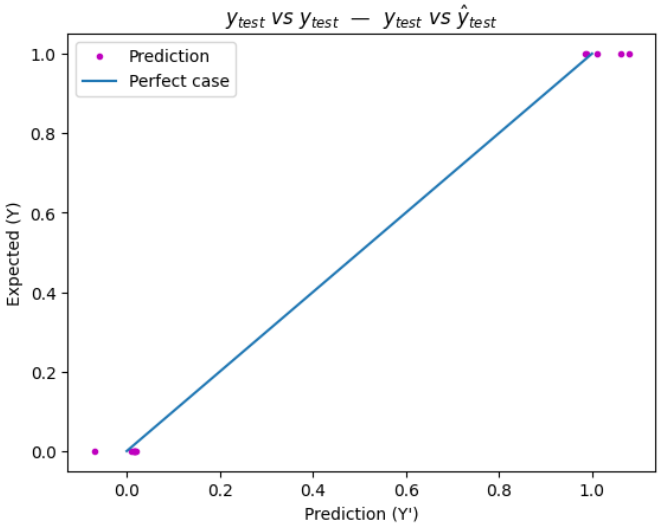

Figure 21. Square - Y vs Y & Y vs YP (Test)

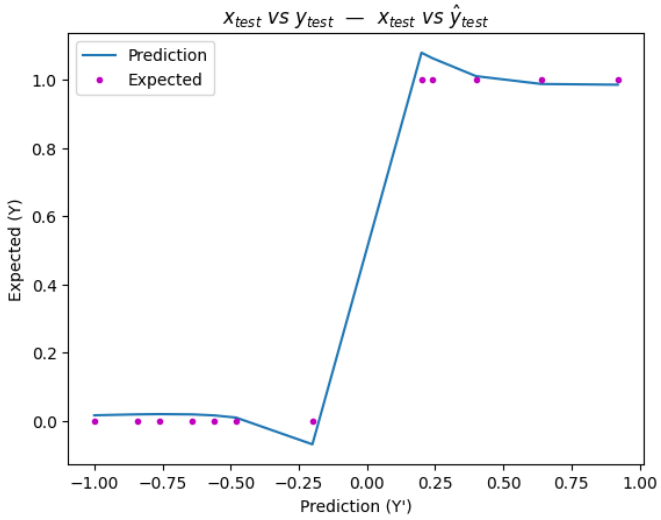

Figure 22. X vs Y & X vs YP (Test)

c) Data Absolute Neuronal Network

The "DataAbsoluteNeuronalNetwork" dataset presents a distinctive scenario, reminiscent of an absolute function. With a distribution resembling the shape of the letter "V," this dataset is strategically chosen for regression analysis within the FCNN framework. The dataset's inherent characteristics, mirroring the behavior of an absolute function, pose a nuanced challenge for the neural network. The FCNN is tasked with learning the underlying patterns and relationships between input (X) and expected output (Y), making it a case study for regression tasks.

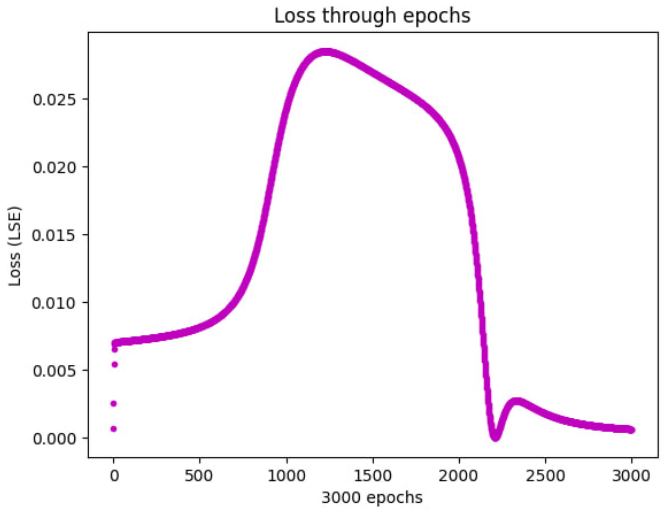

Figure 23. Square Loss

Pointing one specific case, while using 3000 iterationsFigure 23 was generated, this weird behavior in the lost graph was to tackle moving iterations from 3000 to 5000 as can be seen in figure 24.

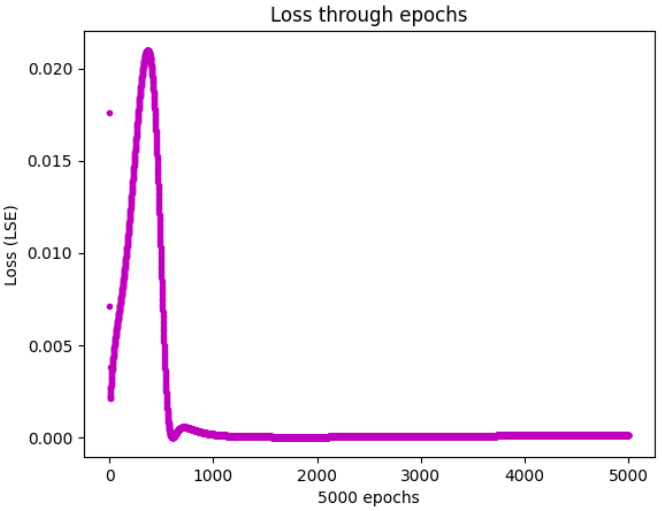

Figure 24. Square Loss

The Root Mean Square Error (RMSE) value of 0.28304695870388324 serves as a quantitative indicator of the accuracy of a regression model. This particular RMSE value signifies that, on average, the model's predictions deviate by approximately 0.28 units from the actual values in the dataset. In the context of regression evaluation, a lower RMSE value reflects better predictive performance, indicating that the model's predictions closely align with the true values. Therefore, a RMSE of 0.28304695870388324 suggests a high level of precision and accuracy in the regression model, with relatively small prediction errors on average across the dataset.

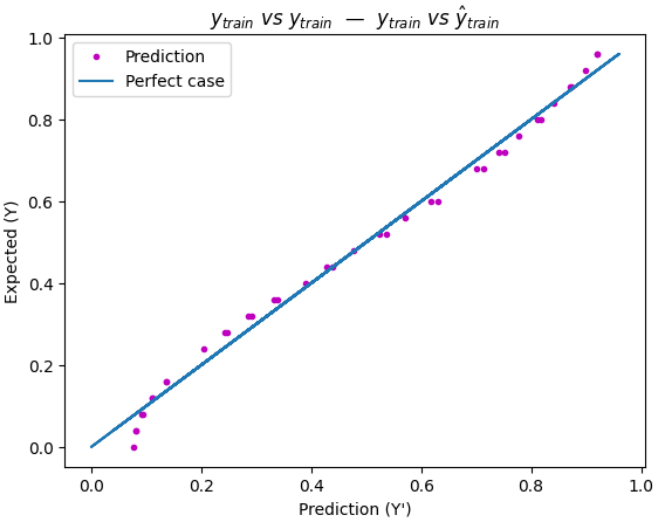

Figure 25. Square - Y vs Y & Y vs YP (Test)

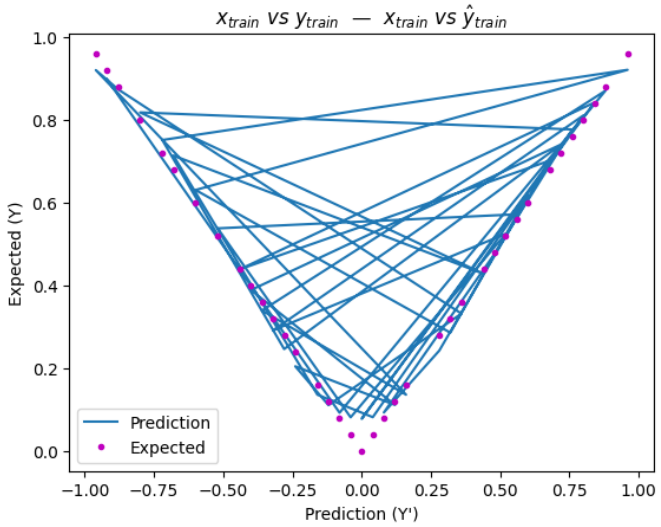

Figure 26. X vs Y & X vs YP (Train)

Figure 25 and Figure 26 show the training and processes for the "DataAbsoluteNeuronalNetwork" dataset offer a visual narrative of the FCNN's adaptability in handling regression tasks. The training graph, generated using Matplotlib in Google Colab, dynamically captures the evolution of the loss function as the FCNN refines its parameters over successive epochs. This visual insight provides a glimpse into the model's learning trajectory, showcasing its ability to minimize errors and optimize for accurate predictions. The corresponding testing graph extends this narrative, illustrating the FCNN's performance on previously unseen data at figure 27 and figure 28 showing testing stage

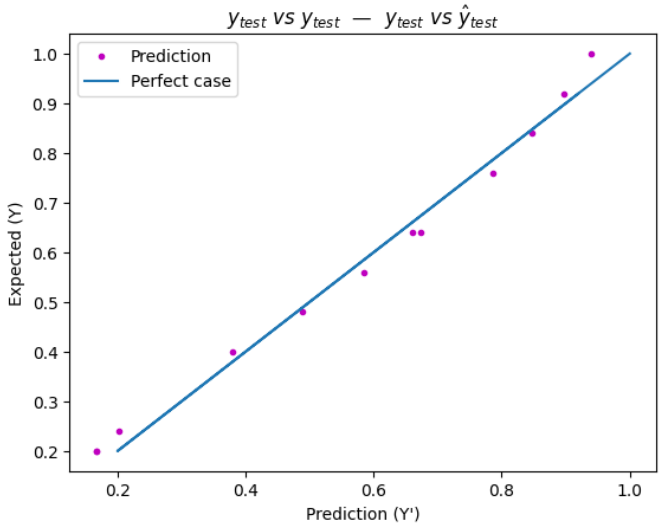

Figure 27. Square - Y vs Y & Y vs YP (Train)

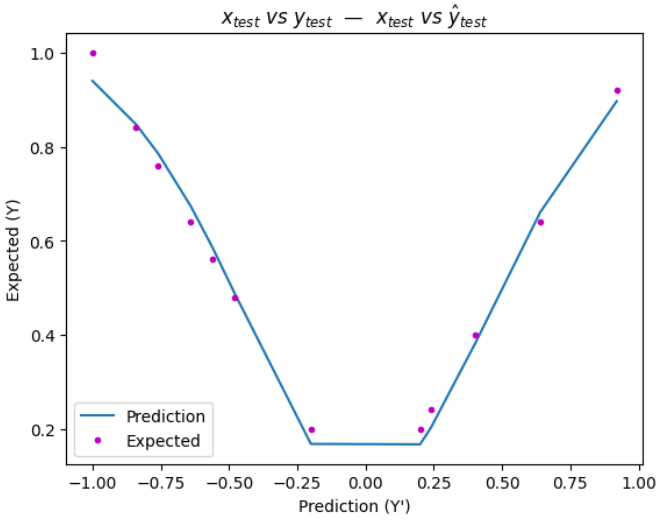

Figure 28. X vs Y & X vs YP (Test)

III. CONCLUSIONS

The comprehensive exploration of Fully Connected Neural Networks (FCNNs) in this laboratory study reveals their versatility and efficacy across classification and regression tasks. The meticulously designed FCNN architecture, activation functions, and hyperparameters contribute to its adaptability in capturing intricate patterns within diverse datasets. The model's successful resolution of the XOR problem highlights its capacity to handle non-linear relationships, which eludes simpler models like single-layer perceptrons. The FCNN's exceptional performance in classification tasks, as evidenced by impeccable metrics and confusion matrices, underscores its reliability in making accurate and balanced predictions.

IV. REFERENCES

[1] AndresMpa “Deep Learning”. GitHub. October 13, 2023. Rerence: https://github.com/AndresMpa/deep-learning/tree/main/labs/task/laboratory_2

[2] ChenaoB “ChenaoB/Database”. GitHub. November 7, 2023. Reference: https://github.com/ChenaoB/Database