Universitat Rovira i Virgili

Neural and Evolutionary Computation

Activity 1: Prediction with Back-Propagation and Linear Regression

Queralt Benito Martín

December 8, 2024

Contents

1 Introduction

This project aims to design, train, and evaluate a neural network-based model to predict CO2 emissions using advanced machine learning techniques.

The core objective is to develop a robust predictive model that minimizes errors and generalizes well to unseen data. In doing so, the project explores the implementation of various neural network architectures and optimization techniques, including hyperparameter tuning, regularization methods, and validation strategies like k-fold cross-validation. These enhancements aim to improve the model's accuracy and prevent overfitting, ensuring that the predictions remain reliable and consistent across different data subsets.

To ensure a comprehensive evaluation, this project compares the performance of the neural network model (BP) against alternative approaches, including a multi-linear regression model (MLR) and another neural network variant (BP-F). These comparisons highlight the advantages and limitations of each method, emphasizing the benefits of using advanced deep learning techniques for predictive tasks.

Additionally, the project incorporates regularization techniques and cross-validation to refine the neural network's ability to handle complex patterns in the data while avoiding overfitting. These steps are important in achieving a balance between model complexity and predictive performance.

Through experimentation, detailed analysis, and comparison of results, this project provides valuable insights into the application of neural networks for predictive modeling. It demonstrates the importance of selecting the right model architecture, tuning hyperparameters, and employing robust validation techniques to build a reliable machine learning solution for CO2 emissions prediction.

The Github repository that contains the dataset used and the notebook files with the python files and the scripts can be found on the following link: https://github.com/queraltbenito/NEC.git.

2 Selecting and analyzing the dataset

The dataset used in this project focuses on vehicle specifications, fuel consumption, and CO2 emissions. The primary objective is to predict CO2 emissions and analyze the impact of vehicle attributes on these emissions. To achieve reliable and accurate predictions, it was important to first preprocess and normalize the data, to ensure consistency, quality, and suitability for machine learning models.

The dataset includes both categorical and numerical features. Key features include Brand, Vehicle Type, Transmission, and Fuel Type (categorical), as well as Engine Size (L), Cylinders, Fuel Consumption (City, Hwy, Combined), and CO2 Emissions (g/km) (numerical). The target variable, CO2 Emissions (g/km), is a continuous variable that will be predicted using regression models. The dataset contains 12 columns corresponding to the key features mentioned, and a total of 7385 entries.

The initial step in preprocessing involved exploratory data analysis (EDA) to identify missing values, inconsistent data types, and outliers. Since this dataset did not contain any missing,

it was not necessary to remove them.
Furthermore, we looked for outliers in the data by examining the statistical properties of the numerical features in the dataset. The key statistics analyzed were the mean, median, minimum, maximum, and interquartile range (IQR). Since any extreme values were found in the min and max columns compared to the mean and 75th percentile (Q3), we considered that the data did not have any outliers.

Categorical features, such as Brand, Vehicle Type, Transmission, and Fuel Type, were originally an 'object' data type. They were converted to the category data type so that these variables could be easily transformed into dummy variables later, allowing their use in regression models. Dummy variable encoding allows categorical data to be represented numerically, facilitating its compatibility with machine learning algorithms.

Moreover, to standardize the numerical features and improve model performance, Min-Max Scaling was applied to variables such as Engine Size (L), Cylinders, Fuel Consumption (City, Hwy, Combined), and CO2 Emissions (g/km). This transformation scaled all numerical features to a range of 0 to 1, ensuring that no single feature dominated the learning process due to differences in magnitude.

The last step of the data preprocessing step was to split the dataset into two subsets: 80% for training and 20% for testing. This split ensured that the models could be trained on a large portion of the data while reserving a separate set for evaluating model performance. Random shuffling was applied to eliminate any inherent ordering bias in the dataset. The preprocessed dataset was saved into two separate files: train-data.csv for training and test-data.csv for testing. This step ensures reproducibility and allows models to be retrained or re-evaluated without reprocessing the data.

The functions used for the newtork are the following:

- fit(X, y): Trains the model using forward and backward propagation.
- predict(X): Generates predictions for new data.
- loss-epochs(): Returns training and validation errors over epochs for visualization.

3 Implementation of BP

The neural network is implemented from scratch using Python. It is structured as a fully connected feedforward neural network, with support for multiple layers and various activation functions. The forward propagation computes activations layer by layer, and the activation function is applied to compute the final activations. Supported activation functions include sigmoid, relu, linear, and tanh.

The backpropagation algorithm minimizes the Mean Squared Error (MSE) by:

1. Calculating the error at the output layer.
2. Propagating the error backward through the network using weight gradients.
3. Updating weights and thresholds using momentum-based gradient descent.

For training the model the training data is split again 80% training and 20% validation sets. Moreover, in the training process the network iteratively updates weights and thresholds over multiple epochs, and training and validation errors are tracked for analysis.

To optimize the neural network, multiple configurations of hyperparameters were tested, including:

- Layer Structure: Number of units in hidden layers ([9, 5], [9, 7]).
- Learning Rate: Tested values like 0.001, 0.01, and 0.05.
- Momentum: Evaluated values such as 0.5, 0.9, and 0.95.
- Activation Functions: Compared sigmoid, relu, and tanh.
- Epochs: Tested durations like 15, 30, and 50 epochs.

The metrics used for evaluating the performance of the model are the Mean Squared Error (MSE), which measures average squared difference between predicted and actual values, the Mean Absolute Error (MAE), which measures the average absolute difference between predicted and actual values, and the Mean Absolute Percentage Error (MAPE), that measures the average percentage difference relative to actual values.

These are the results for the hyperparameters evaluation that has been conducted:

Number of layers	Layer Structure	Num epochs	Learning Rate	Momentum	Activation function	MAPE	MAE	MSE
4	[2016, 9, 5, 1]	30	0.01	0.9	sigmoid	4.5579	0.01064	0.0002
4	[2016, 9, 5, 1]	30	0.001	0.9	sigmoid	42.5829	0.1034	0.0170
4	[2016, 9, 5, 1]	30	0.05	0.5	sigmoid	4.9432	0.0113	0.0002
4	[2016, 9, 5, 1]	30	0.05	0.95	sigmoid	3.0850	0.0068	0.0001
4	[2016, 9, 7, 1]	30	0.05	0.95	sigmoid	55.1860	0.1666	0.0700
4	[2016, 9, 5, 1]	30	0.05	0.95	tanh	349.4608	1.015	1.1784
4	[2016, 9, 5, 1]	30	0.05	0.95	relu	100.0	0.3595	0.1482
4	[2016, 9, 5, 1]	15	0.05	0.95	sigmoid	2.8601	0.0073	0.0001
4	[2016, 9, 5, 1]	50	0.05	0.95	sigmoid	3.2013	0.0068	0.0001
4	[2016, 9, 5, 1]	15	0.001	0.9	sigmoid	42.1959	0.1031	0.0169

First, we analyzed the impact of different learning rates in the error metrics. As we can see, a learning rate of 0.05 generally produces better results (lower MAPE, MAE, and MSE) compared to smaller learning rates like 0.001 or 0.01. This is because a smaller learning rate (0.001) causes the model to converge too slowly, leading to worse results due to insufficient updates during training. Also, when the learning rate is too high in a neural network, the training process becomes unstable, and the model may fail to converge to an optimal solution.

Then we also saw the effects in the performance of the network of the changing momentum. Higher momentum values (0.95) seem to stabilize and speed up convergence, yielding better performance compared to lower momentum values (0.5).Lower momentum might hinder the optimization process, making it harder for the model to escape local minima.

Moreover, it was also studied which was the optimal number of epochs. We can observe that increasing the number of epochs from 15 to 30 can improve the model's performance by allowing it to better learn the patterns in the data. However, excessively increasing the epochs to 50

shows diminishing returns, with marginal improvements in error metrics. Nonetheless, since the results obtained for 15 and 30 epochs are nearly the same, we have considered that training the model with 15 epochs is beneficial since less computer resources need to be used, making this model more environmental-friendly.

Moreover, different configurations of the hidden layer structure have been tested. We have found that a structure of [2016, 9, 5, 1] generally performs better than [2016, 9, 7, 1], as adding more neurons in the second hidden layer introduces unnecessary complexity, which can lead to overfitting or difficulty in optimization.
Also, different activation functions have been used, and we have found that the sigmoid activation function consistently outperforms tanh and ReLu in this dataset. This can be due to the fact that the Sigmoid yields the best overall metrics, likely due to its suitability for problems with bounded outputs. ReLU performs poorly in this case, potentially due to the vanishing gradient issue or improper weight initialization, especially with a small learning rate. Also, Tanh produces even worse results, likely because it exaggerates the vanishing gradient issue and struggles with weight updates in deeper networks.

After this analysis, we have found that the best hyperparameter combination for this model, based on the error metrics, has the following configuration:

- Learning Rate: 0.05
- Momentum: 0.95
- Activation Function: Sigmoid
- Number of Epochs: 15
- Layer Structure: [2016, 9, 5, 1]

This configuration achieves the lowest MAPE (around 2.86%), MAE, and MSE, indicating excellent predictive performance on the dataset.
For this combination of hyperparameters, we have displayed a scatter plot of the predicted values vs the real values and a plot of the evolution of the training and validation error as a function of the number of epochs.

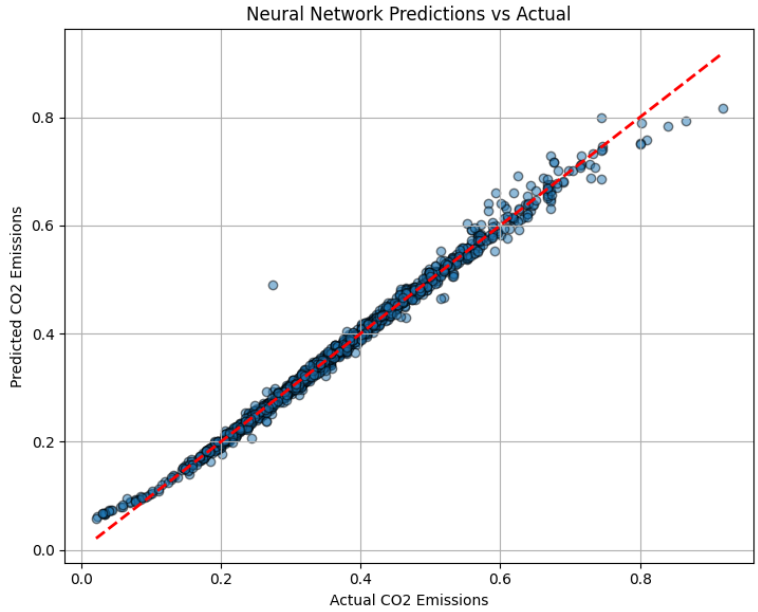

Figure 1: Scatter plot of the neural network predicitons vs actual of the best configuration

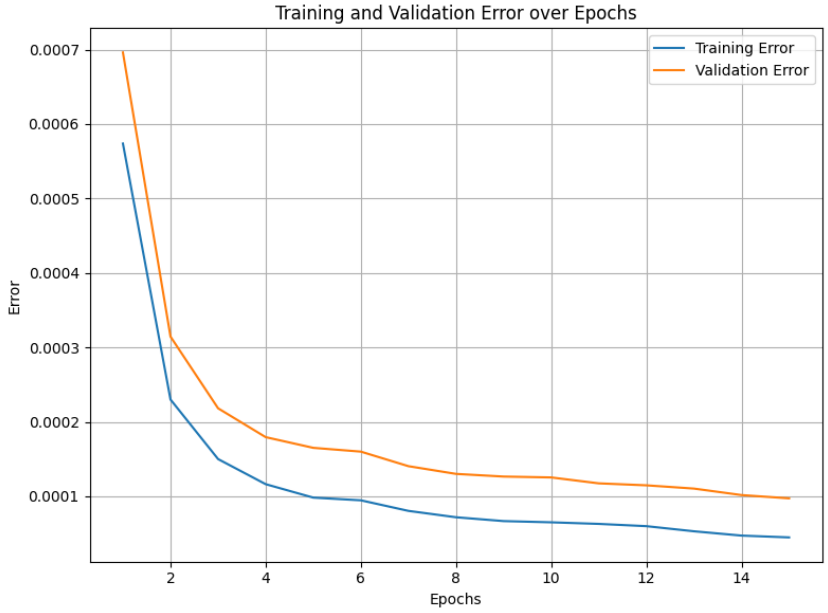

Figure 2: Plot of the training and validation errors over epochs of the best configuration

The scatter plot shows a strong alignment of predicted CO2 emissions with the actual emissions along the diagonal reference line. Most of the points are concentrated close to the diagonal, indicating high prediction accuracy. A few minor deviations suggest that the model might slightly underperform in certain cases, but these outliers are minimal. In conclusion, the neural network demonstrates excellent predictive performance for the given dataset and hyperparameters. The alignment suggests the network generalizes well to unseen test data.

The training error decreases consistently across the 15 epochs, showing effective learning of the model.The validation error follows a similar trend, converging without significant overfitting. Both errors stabilize towards the end of the training process, with validation error remaining slightly higher than training error, which is expected. The slight gap between training and validation errors is a sign of good generalization, as the model balances learning from the training data while performing reliably on the validation set.

We can compare this plots with the ones obtained for other hyperparameter combinations. We will use the model that has the following configuration:

- Learning Rate: 0.01
- Momentum: 0.9
- Activation Function: Sigmoid
- Number of Epochs: 30
- Layer Structure: [2016, 9, 5, 1]

For this combination, we obtain the following plots:

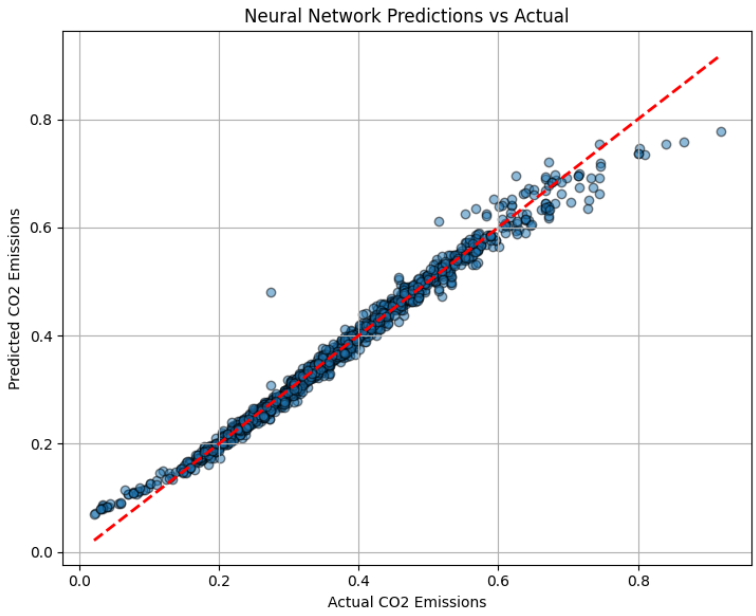

Figure 3: Scatter plot of the neural network predicitons vs actual

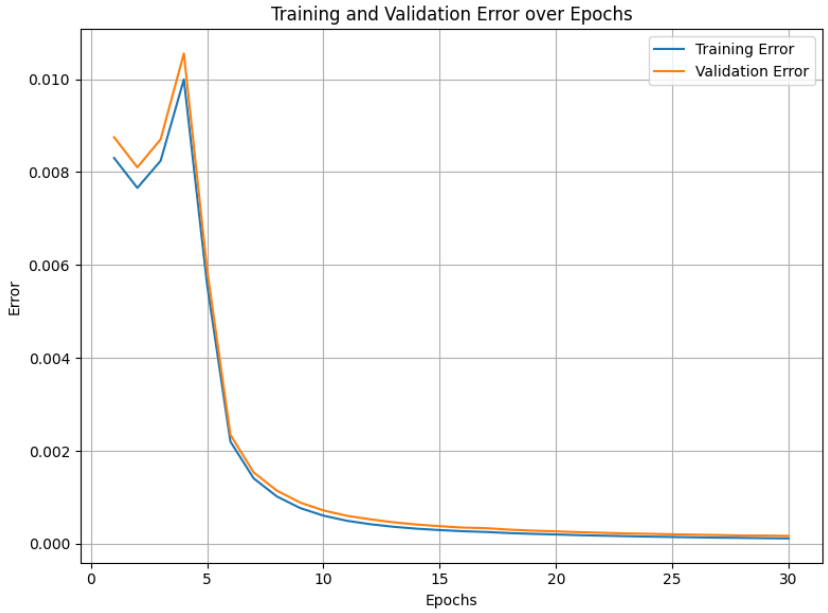

Figure 4: Plot of the training and validation errors over epochs

In this case, the points in the scatter plot deviate more significantly from the diagonal reference line compared to the configuration with optimal hyperparameters. This suggests that the model is less accurate in capturing the relationship between input features and CO2 emissions.Some clusters of predictions appear systematically biased, either consistently overpredicting or underpredicting CO2 emissions. The increased spread around the diagonal line indicates higher residual errors, resulting in larger MSE and MAE metrics. Also, the outlier behavior is more pronounced, highlighting specific instances where the model struggles to generalize.
For the training and validation plot, we can see that both training and validation errors start at higher values compared to the optimal configuration. This indicates that the model struggles to learn effectively due to a potentially poor initialization or suboptimal hyperparameter settings. Moreover, the early oscillation in validation error suggests instability during the initial epochs, possibly caused by a learning rate that is not well-tuned or a mismatch between the complexity of the network and the dataset. Despite these oscillations, the training error eventually converges, but the validation error stabilizes at a much higher value, indicating poor generalization.

Lastly, we will also visualize the plots obtained for the following combination of hyperparameters:

- Learning Rate: 0.001
- Momentum: 0.9
- Activation Function: Sigmoid
- Number of Epochs: 30
- Layer Structure: [2016, 9, 5, 1]

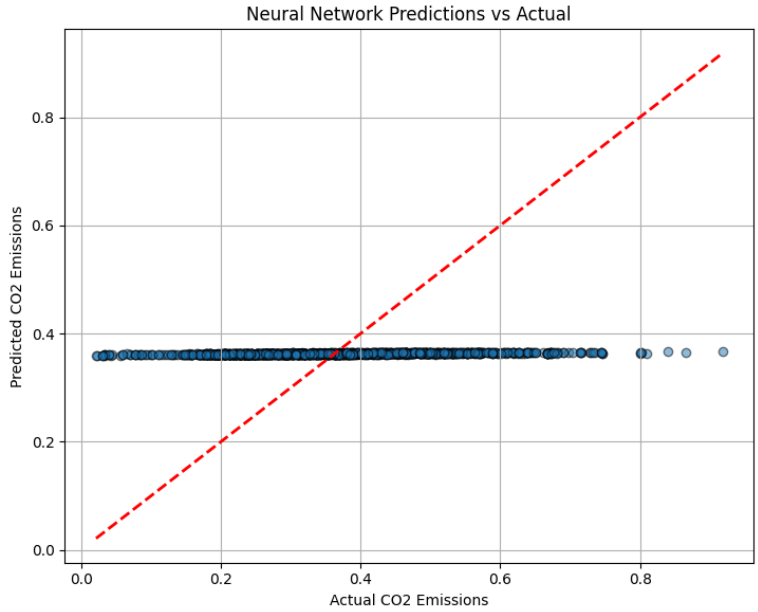

Figure 5: Scatter plot of the neural network predicitons vs actual

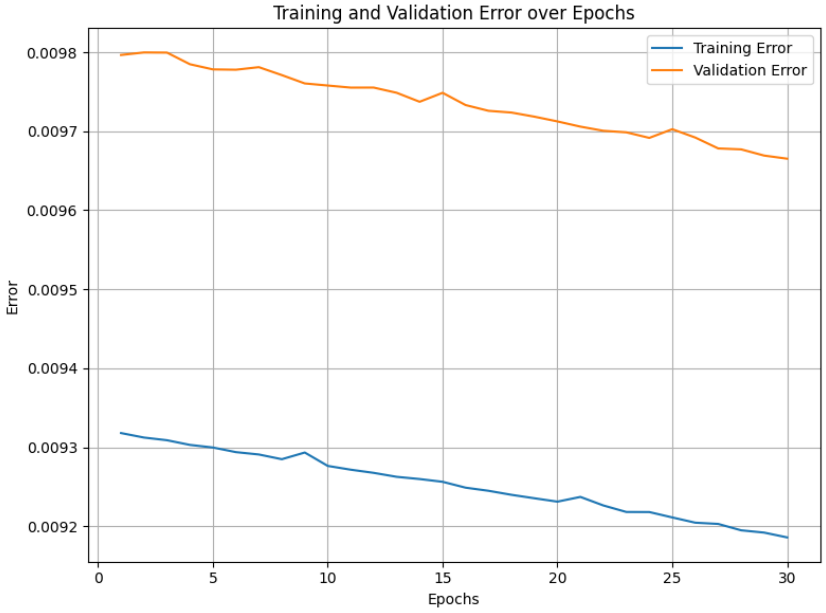

Figure 6: Plot of the training and validation errors over epochs

The predicted CO2 emissions are nearly constant, resulting in a horizontal line regardless of the actual values. This indicates that the model fails to capture the relationship between the input features and the target variable. Such behavior suggests that the model is underfitting, as it is unable to learn meaningful patterns from the data. The complete lack of variance in the predictions highlights a high bias problem. The model's complexity or hyperparameter configuration does not allow it to adapt to the data's distribution.

Both training and validation errors start at relatively high values and exhibit minimal improvement over epochs. This suggests that the model configuration, including hyperparameters like the learning rate or activation function, is unsuitable for the dataset. A noticeable gap between the training and validation errors remains throughout the training process. This gap suggests that the model is failing to generalize, which could also indicate poor feature representation or insufficient capacity.

4 Obtaining and comparing predictions using the three models (BP, BP-F,MLR-F)

The MLR-F model, implemented using Scikit-Learn's LinearRegression class, is a linear regression model. Its goal is to establish a linear relationship between the input features and the target variable, CO2 emissions. This model assumes that the relationship between the dependent variable and independent variables is linear. All preprocessed features, including both numerical and categorical variables. Categorical variables were transformed into dummy variables to enable their use in the regression model.
MLR-F uses ordinary least squares (OLS) to estimate the coefficients of the linear equation. No explicit hyperparameters like learning rate or epochs are needed, as the model finds the best-fit line analytically.

The BP-F model is a feedforward neural network built using TensorFlow's Sequential API. It features two hidden layers with ReLU activation and an output layer with linear activation for regression tasks. The parameters used in this model are as follows:

- Input Layer: Accepts all numerical and encoded categorical features as input.
- First Hidden Layer: 9 neurons with ReLU activation.
- Second Hidden Layer: 5 neurons with ReLU activation.
- Output Layer: A single neuron with a linear activation function to predict CO2 emissions.
- Optimizer: Adam optimizer, which combines the benefits of momentum and adaptive learning rates to improve convergence speed and robustness.
- Loss Function: Mean Squared Error (MSE), chosen to minimize the squared differences between predicted and actual CO2 emissions.
- Metrics: Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are used alongside MSE to evaluate the model's performance.
- Epochs: 30 iterations over the dataset to optimize weights.
- Batch Size: 32 samples per batch for gradient updates.
- Validation Split: 20% of the training data is used for validation during training to monitor generalization performance.

Most of the parameters are chosen to be similar as the implementation we made of the neural network. The architecture is intentionally simple yet sufficient for this dataset. Using two hidden layers avoids underfitting while remaining computationally efficient. ReLU activation, Adam optimizer, and batch size choices ensure stable and efficient training, reducing the likelihood of gradient instability or vanishing gradients.

The primary distinction between the two models lies in their complexity and the way they use input features. MLR-F directly learns coefficients for linear relationships, whereas BP-F uses layered transformations and non-linear activations to model more complex relationships. BP-F requires hyperparameter tuning and is computationally intensive compared to MLR-F. However, this complexity enables BP-F to better capture non-linear patterns in the data.

The performance of both of this models, in comparison with the original neural network model, is the following:

Model	MAPE	MAE	MSE
MLR-F	144428313532.6380	397632169.5809	4.4979e+18
BP-F	3.50672	0.01068	0.0002
BP	2.8601	0.0073	0.0001

We have also obtained plots for evaluating both of these models. For the MLR-F model, we obtained the scatter plot of the predicted values vs the real values:

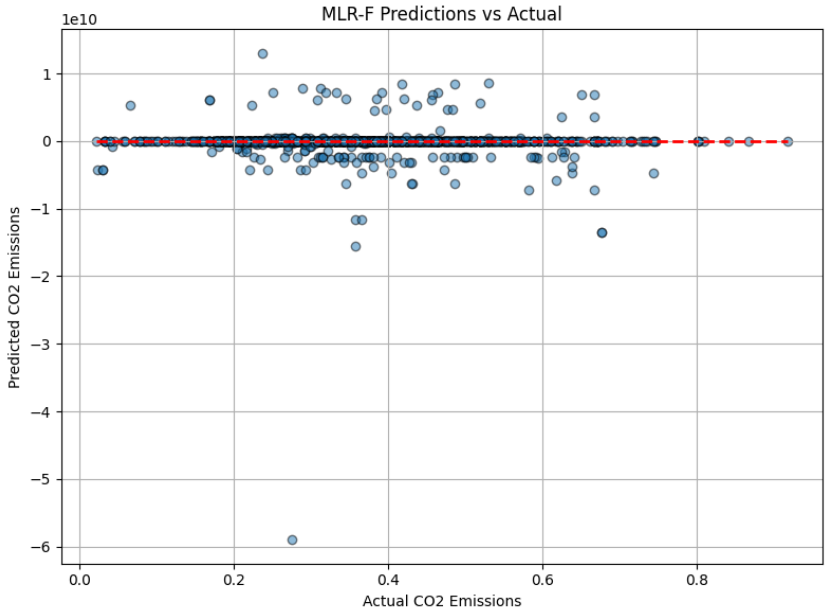

Figure 7: Scatter plot of the neural network predicitons vs actual of the MLR-F model

The scatter plot demonstrates a clear inadequacy in the predictions made by the Multi-Linear Regression (MLR-F) model. Most predicted values are concentrated around a horizontal line, failing to capture the variability in the actual CO2 emissions. This suggests that the model struggles to generalize or represent the underlying relationships between the input features and the target variable. The horizontal alignment of predictions around a constant value might indicate that the linear assumptions of the MLR-F model do not suit the complex relationships in the dataset. This could result from the dataset's non-linear interactions, which a linear regression model cannot effectively capture. Also, there are a few extreme outliers with both very high and very low predicted values, far removed from the actual values. These outliers suggest that the model lacks robustness and struggles with extreme cases in the data.

For the BP-F model we have obtained the following plots:

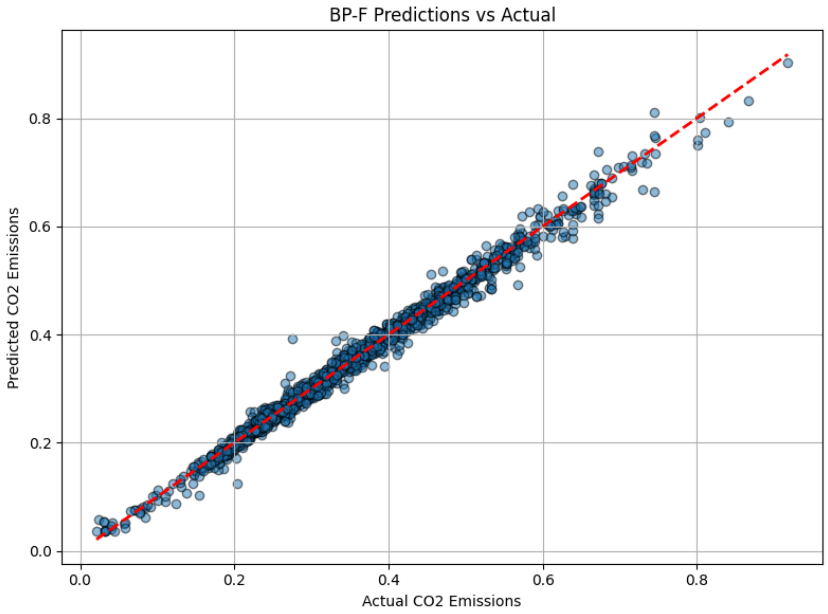

Figure 8: Scatter plot of the neural network predicitons vs actual of the BP-F model

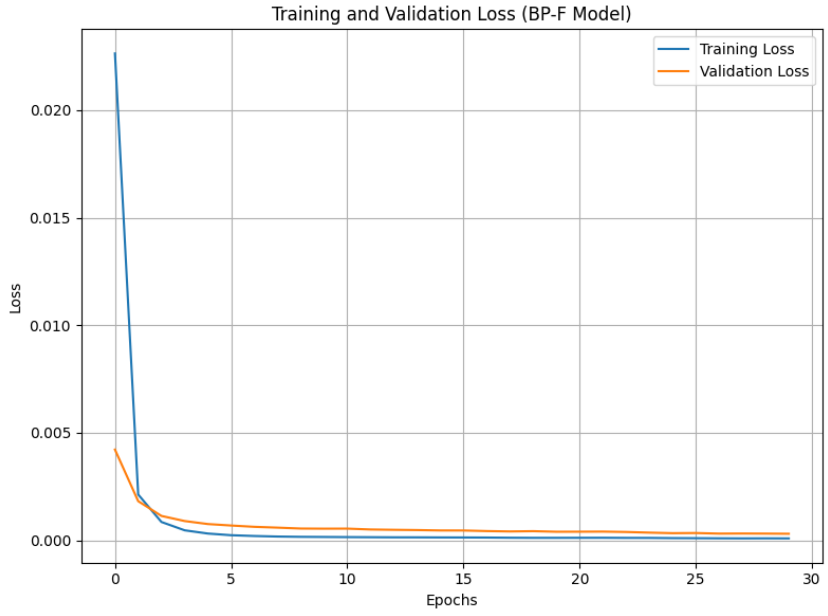

Figure 9: Plot of the training and validation errors over epochs of the BP-F model

The scatter plot shows a strong alignment of predicted CO2 emissions with the actual values along the red diagonal line. This indicates that the BP-F model effectively captures the underlying relationships in the data. The points are tightly clustered around the diagonal line, with minimal scatter, suggesting low variance in the predictions. This reflects the BP-F model's ability to generalize well on the test data. While most predictions are accurate, a few outliers

deviate significantly from the diagonal line. These outliers may arise from rare or extreme cases in the dataset that the model finds difficult to predict accurately. However, their presence is minimal, indicating robustness in the model. The strong alignment suggests that the BP-F model successfully captures non-linear relationships between the input features and CO2 emissions, outperforming linear regression approaches like MLR-F.

The training and validation loss decrease sharply within the first few epochs, indicating that the model learns efficiently during the early stages of training. This rapid convergence is a hallmark of well-tuned hyperparameters and effective optimization using the Adam optimizer. Both training and validation loss reach very low values by the end of the training process. This reflects the model's high accuracy and ability to minimize errors. The validation loss closely follows the training loss throughout the training process, with no significant divergence. This indicates that the model generalizes well to unseen data and does not overfit the training set. The loss curves stabilize after a few epochs, showing minimal fluctuations. This stability highlights that the chosen learning rate, momentum, and architecture are well-suited for the dataset.

We have seen that the MLR-F model has the worst performance, and that for this task is not suitable. The MLR-F model assumes a linear relationship between input features and the target variable. As evident in the scatter plot, the predictions fail to capture the non-linear relationships present in the data, leading to poor alignment with actual values. In the case of BP-F, we have observed that it effectively captures non-linear patterns, as evidenced by the strong alignment of predictions with actual values in the scatter plot. The Adam optimizer facilitates rapid convergence and low loss values for both training and validation data, as shown in the loss plot. Also, the lack of overfitting, demonstrated by similar training and validation loss curves, indicates that the model generalizes well to unseen data. This model achieves a MAPE of 3.51%, indicating strong alignment between predicted and actual values. It also has an MAE of 0.0107 and an MSE of 0.0002, showcasing its precision in minimizing both absolute and squared errors. The BP-F model demonstrates exceptional performance due to its advanced optimization techniques (Adam optimizer) and non-linear modeling capabilities. It outperforms MLR-F by a significant margin. However, the BP model we have created achieves a MAPE of 2.86%, the lowest among all models, highlighting its higher accuracy. With an MAE of 0.0073 and an MSE of 0.0001, the BP model slightly outperforms BP-F, showing that careful manual tuning of hyperparameters and architecture can achieve state-of-the-art results.
In conclusion, despite being implemented from scratch, the BP model delivers the best overall performance. This demonstrates the effectiveness of its architecture (Learning Rate: 0.05, Momentum: 0.95, Activation Function: Sigmoid) and hyperparameter tuning.

5 Effect of the different regularization techniques in the Neural Network

In this section the aim is to introduce two regularization techniques to the neural network to try to improve the prediction results. These tecnhiques are L2 regularization and dropout.

L2 regularization is a technique used in machine learning to prevent overfitting. It works by adding a penalty term to the model's loss function, which discourages the model from assigning large weights to its parameters. This penalty is proportional to the square of the weights, which means the model is incentivized to keep the weights small and balanced. By doing this, the model becomes simpler and less sensitive to small fluctuations in the data, improving its ability

to generalize to new, unseen data.
We used values like 0.01 and 0.1 for L2 regularization. These values were chosen because they introduce a small-to-moderate penalty on large weights. A value of 0.01 ensures that the model remains flexible and does not over-penalize the weights, which could lead to underfitting. However, a larger value, like 0.1, adds a stronger penalty, which can help in cases where the model is overfitting severely by forcing it to simplify.

Dropout is another regularization technique designed to prevent overfitting, but it works differently. During training, dropout randomly disables (or "drops out") a percentage of the neurons in the network on each pass. This forces the network to learn redundant representations of the data, as no single neuron can depend too heavily on another. This makes the model more robust and improves its generalization.
Dropout rates of 0.2 and 0.5 were selected because they represent common thresholds in practice. A dropout rate of 0.2 removes 20% of the neurons during training, which is a light regularization. It's useful when the model is already performing well but might slightly benefit from additional robustness.A dropout rate of 0.5 is more aggressive, dropping 50% of the neurons. This is helpful when the model has a higher risk of overfitting, as it forces the network to learn more generalized patterns.

Combining L2 regularization and dropout allows the model to benefit from both approaches: L2 reduces overly complex weights, and dropout prevents reliance on specific neurons.
After applying both of these techniques to the BP-F model, we have obtained the following results:

L2 Regularization	Dropout Rate	MSE	MAE	MAPE
0.01	0.2	0.0053	0.03549	17.6518
0.01	0.5	0.01147	0.07554	33.1617
0.1	0.2	0.0189	0.1092	45.2360
0.1	0.5	0.0189	0.1091	45.0479

The table results demonstrate the impact of L2 regularization and dropout rates on the neural network's performance, showing how these techniques influence the model's ability to generalize while avoiding overfitting. The combination of L2 regularization (0.01) and dropout rate (0.2) achieved the best overall performance, with an MSE of 0.0053, MAE of 0.03549, and MAPE of 17.6518. This indicates that a small penalty for large weights (L2 = 0.01) is effective at reducing overfitting, while a moderate dropout rate (0.2) prevents co-adaptation of neurons without excessively impairing the model's learning capacity.

As the table shows, increasing either the L2 regularization parameter (to 0.1) or the dropout rate (to 0.5) leads to significantly worse performance. Higher L2 values penalize the weights more heavily, restricting the model's flexibility and resulting in underfitting. Similarly, a higher dropout rate (0.5) removes a substantial portion of neurons during training, leading to less stable learning and reduced predictive accuracy, as evidenced by the elevated MSE, MAE, and MAPE values.

In conclusion, the selected values of L2 regularization = 0.01 and dropout rate = 0.2 strike the right balance between reducing overfitting and maintaining sufficient model complexity for accurate predictions. These values were chosen through experimentation, as they minimize error metrics while preserving generalization capabilities, highlighting their suitability for the

dataset and task at hand.

6 Introduce Cross Validation in the model selection and validation

Cross-validation is a robust method to evaluate the performance of a model by splitting the dataset into multiple subsets (folds). It ensures that the model is trained and validated on different parts of the dataset, reducing the risk of overfitting or underfitting. For this task, k-fold cross-validation has been implemented to assess the generalization performance of the neural network.

First, we used 5-fold cross-validation. The dataset was split into 5 folds, a common choice that balances computational efficiency and robust error estimation. In 5-fold cross-validation, the dataset is divided into 5 subsets of approximately equal size. The model is trained on 4 folds and validated on the remaining fold. This process is repeated 5 times, with each fold used as the validation set once. The results are averaged across the folds to provide a comprehensive evaluation of the model. For each fold, 80% of the data is used for training, and 20% is used for validation. This ensures that the model is trained on a majority of the data while reserving sufficient data for unbiased validation. The training and validation data change in each iteration of the folds, ensuring that every data point is used for validation exactly once.

These are the results obtained for the BP model:

Folds	MSE	MAE	MAPE
1	0.000155	0.007184	inf
2	0.000122	0.006935	2.278262
3	0.186800	0.398080	112.765027
4	0.000186	0.007939	2.474771
5	0.000167	0.008274	2.944541

The 5-fold cross-validation results show variability in the performance metrics across folds. The majority of the folds exhibit low MSE (ranging from 0.000122 to 0.000186), and MAE values are similarly small (around 0.0069 to 0.0083). This indicates that the model performs well in most of the splits, maintaining a high level of predictive accuracy. In Fold 3, both MSE (0.1868) and MAE (0.398) are significantly higher compared to the other folds. This suggests that the model struggled with the data distribution in this particular split, possibly due to uneven distribution of outliers or a poor representation of the data in this fold. For most folds, the MAPE is reasonable and below 3%, indicating low relative error. Fold 1 has an infinite MAPE, likely caused by zero or near-zero true values in the test set, which leads to a division by zero in the MAPE calculation. Fold 3 also exhibits a high MAPE (112.76%), reinforcing the observation that the model had difficulty generalizing for this fold.
Overall, the model shows consistent performance in most folds but is affected by data distribution issues in specific splits.

We have also implemented the 5-fold validation to the BP-F model. These are the results obtained:

Folds	MSE	MAE	MAPE
1	0.000236	0.010869	inf
2	0.000222	0.010604	3.439467
3	0.000331	0.012218	3.876651
4	0.000387	0.013148	3.979098
5	0.000275	0.011485	3.833861

As we can see, the MSE values across the folds remain relatively small, indicating that the model performs well in minimizing the squared differences between predicted and actual values. The MAE values are similarly consistent, with small variations across the folds. The lowest MAE is 0.010604 in Fold 2, while the highest is 0.013148 in Fold 4. These values demonstrate that the model is effective at minimizing the absolute prediction errors, maintaining reasonable performance even with varying data splits. The MAPE values are generally low, ranging between approximately 3.4% and 3.9% across Folds 2 to 5. However, Fold 1 has an infinite MAPE, likely due to division by zero in the percentage error calculation for certain data points. This suggests that a small number of target values might be zero or extremely close to zero in Fold 1, which skews the MAPE calculation.
Overall, the model exhibits good generalization ability, as the evaluation metrics remain consistent across most folds. The slightly higher MSE and MAE in Fold 4 might indicate variability in the training and validation splits, potentially due to a slightly harder subset of the data in this fold.

In conclusion, when comparing the BP and BP-F models using 5-fold cross-validation, we find some differences. The BP model achieves lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) in certain folds, indicating strong predictive accuracy on some subsets of the data. However, its performance is inconsistent, with significant variability across folds, including outliers such as the high MSE and MAPE in fold 3, which suggest sensitivity to specific data distributions. In contrast, the BP-F model demonstrates more consistent results across all folds, with slightly higher average errors but no extreme outliers. This stability suggests that the BP-F model is less prone to overfitting and better suited for generalization to unseen data. While the BP model may excel in specific cases, its variability makes it less reliable, whereas the BP-F model's consistent performance across folds highlights its robustness.