# REPORT

## ON SLURRY DEFORMATION PREDICTION PROGRAM

This program utilizes a hybrid approach combining a Backpropagation Neural Network (BPNN) with Genetic Algorithm (GA) optimization to predict slurry deformation. Here's a detailed breakdown:

Data Loading and Preprocessing:

* Data Loading: The program reads training data from 'training.csv' and validation data from 'validation.csv' using pandas. The training and validation dataset is divided manually from the original excel data file.
* Data Conversion: It converts the pandas DataFrames to numpy arrays for efficient processing.
* Missing Value Handling: Missing values in the training data are replaced with the mean of the corresponding column using np.nan\_to\_num().
* Feature Scaling: StandardScaler is used to standardize the features in the training data (zero mean and unit variance). This helps improve the performance of the neural network.
* Data Splitting: The training data is split into training and validation sets using train\_test\_split().

Data Augmentation:

* Noise Injection: To improve model robustness, the program adds noise to the training data using np.random.normal(). The noise factor (0.05) determines the level of noise added.
* Data Concatenation: The augmented data is concatenated with the original training data, effectively doubling the training examples.

Backpropagation Neural Network Model (BPNN):

* Model Creation: The function create\_bp\_model() defines the architecture of the BPNN using keras.Sequential().
* Layers: The model has an input layer, one or two hidden layers with ReLU activation, and an output layer with 94 units (presumably representing the output features).
* Regularization: L2 regularization is applied to the hidden layers using l2(l2\_reg) to prevent overfitting.
* Compile: The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function.

Training the BPNN:

* Function train\_bp\_model(): This function trains the BPNN on the training data.
* Early Stopping: Early stopping is used to prevent overfitting by monitoring the validation loss and stopping training when the loss plateaus for a certain number of epochs.
* Hyperparameters: The epochs and batch\_size control the training process.

Genetic Algorithm Optimization:

* DEAP Library: The DEAP library is used to implement the GA optimization.
* Objective Function: The objective function (evaluate\_individual()) calculates the MSE on the validation set for a given set of hyperparameters.
* Individuals: Each individual represents a set of hyperparameters for the BPNN (hidden units, learning rate, batch size, epochs, and L2 regularization).
* Evolutionary Process: The GA iteratively evolves a population of individuals by applying selection, crossover, and mutation operations to optimize the objective function.

Integrating GA with BPNN:

* optimize\_parameters(): This function runs the GA optimization and returns the best individual (set of optimized hyperparameters).
* integrate\_ga\_with\_bp(): This function uses the optimized hyperparameters from the GA to create and compile a new BPNN model.

Cross-Validation:

* KFold: K-fold cross-validation is performed using KFold() to assess the model's performance on different partitions of the data.
* Cross-Validation Procedure: The program iterates through the folds, optimizing the BPNN using the GA for each fold.
* Performance Metrics: The mean squared error (MSE), mean absolute error (MAE), and R-squared (R2) are calculated for each fold and averaged across all folds.

Evaluation and Adjustment:

* Final Model Evaluation: The program evaluates the final BPNN model using the test set and reports the MSE, MAE, and R2.
* Parameter Adjustment: If the validation performance is not satisfactory (MSE > 0.01), the program adjusts the learning rate of the BPNN based on the best individual found by the GA. This adjustment is aimed at improving the model's performance on the validation set.
* Re-Evaluation: The adjusted BPNN model is then re-evaluated on the test set.

Model Saving and Visualization:

* Model Saving: The trained and optimized BPNN model is saved to a file using bp\_model.save().
* Learning Curve Visualization: The program plots the training and validation loss over epochs to visualize the learning process.
* Predicted vs Actual Plot: The program also plots the predicted values against the actual values for the test set.

Overall, this program combines the strengths of both BPNN and GA, leading to a robust model for slurry deformation prediction. The GA optimization helps to find optimal hyperparameters for the BPNN, leading to improved accuracy and performance. The cross-validation process ensures that the model generalizes well to unseen data. Finally, the visualizations provide insights into the learning process and model performance.