#### Modified Deep Learning Neural Network

The proposed system processes the pre-processed data using a Modified Deep Learning Neural Network (MDLNN). The previous method for water quality classification [44] employs an ANN using only one Hidden Layer(HL), which requires more time for training and provides less accuracy.

In the previous methods [44], large weights in an ANN imply an extra complicated network that quickly overfits the training data. If the model is tested on unseen or new data, then it takes longer to train, performs poorly, and results in less accuracy. Furthermore, overfitting increases the generalization error.

The proposed MDLNN reduces training time and improves classification accuracy by utilizing hyperparameter tuning procedures. Proper selection of hyperparameters for training the model enables Neural Networks to learn better and faster, leading to enhanced tuning process performance. The proposed method uses hyperparameters in two ways: first for model designing and then for model training. The model design uses Neural Network (NN) architecture hyperparameters like the number of HL and neurons for each layer. For model training, hyperparameters like activation function, dropout, optimizer, normalization, learning rate, batch size, and a number of epochs are utilized. Figure 3.7 depicts a general representation of MDLNN having input layers, hidden layers, and output layers.

Diagram, schematic

Description automatically generatedFigure 3.7: Structure of MDLNN

The proposed system makes extensive use of training data, necessitating the use of several HL. The proposed system makes use of a dataset with a dimension of ten as an input. The proposed system creates an optimized model using Adam optimiser that includes four HL and one output layer. Adam(Adaptive Moment Estimation) is seen as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum.

Adam is also an adaptive gradient descent algorithm and it optimize a learning rate, weights and biases per-parameter function f(θ). But Adam does not converge to an optimal solution in some areas like object recognition or machine translation. Adam also suffers from a weight decay problem.

The exponential moving average of past squared gradients as a reason for the poor generalization behaviour of adaptive learning rate methods. To resolve this issue and to get optimum weight, the proposed algorithm keep running deflection and average on momentum with respect t and velocity with respect to time. If the deflection is greater than the threshold then the average value is used to smoothing of weight. This method is used to optimize an objective function f(θ), with parameters θ (weights and biases) using modified Adam algorithm and therefore, algorithm is called as Modified Deep Learning Neural Network.

Modified Adam Algorithm

(*Compute the deflection of current and previous momentum)*

( Compute average of current and previous momentum)

(Compute bias – corrected the first moment estimate)

Here, is weight at time t, ()

If the accuracy does not improve after 500 epochs, the early stopping mode is preferred to end a learning process. Except for the last layer, model freezing is done that is based on maximum accuracy with the least validation loss for a specific epoch. As a result of using the checkpoint, the best model is obtained. The first hyperparameter to fine-tune is the number of neurons in every hidden layer. In this scenario, the total number of neurons in every layer is kept constant. It could be custom-made. The number of neurons utilized must be directly proportionate to the complexity of the solution. More neurons are required to achieve improved prediction. The number of neurons is chosen between 10 and 750 in a 16-step size. The number of HL is increased based on cross-validation to the accuracy of the testing dataset. Linear activation functions are frequently used in the neural network output layer to solve regression problems. Cross-entropy activation functions are suitable for binary classification since they generate values of 0 and 1. The proposed method is used to solve the multiclass classification problem, which requires class membership on more than two labels, and therefore, the MDLNN uses a non-linear activation function. In the input layer, relu is used as an activation function, leaky relu is used in the HL, and SoftMax is applied for the output layer. The sigmoid and Tanh activation functions have dying relu and vanishing gradient problems, and therefore, the proposed method selects a leaky relu function. The Leaky relu activation function addresses these problems, is more effective, and requires less computing time. SoftMax is used as the activation function in the output layer since it generates a single value output for every node in the output layer. The output values of the softmax function are obtained as probabilities with a probability sum of 1.0.

The MDLNN employs batch normalization to make sure that neither activation is going too low or too high. As a result, a higher learning rate is used with the batch normalization technique. It introduces noise into the activation of the selected layer to add a regularization effect. Covariate shift is also kept to a minimum. Batch normalization helps to reduce sensitivity to initial starting weights. Dropout regularization on larger networks is used to minimize the overfitting issue. During the training phase, the dropout method does not consider randomly chosen neurons, and as a result, these randomly chosen neurons are dropped out at random. This means that on the forward pass, their contribution to the activation of downstream neurons is removed temporally, and any weight updates are not applied to the neuron on the backward pass. The dropout method enables the network to become less sensitive to the specific weights of the neurons. As a result, the proposed system enhances a network by significantly reducing training time, leading to better generalization, improving accuracy, and minimizing the chances of overfitting the training data.