**🧾 First Model Summary for Supervised Regression**

**🎯 Task:**

Predict **2 continuous output variables** based on **3 input features**, using a deep neural network.

**🔢 Input Features (3):**

* size\_formation
* comm\_delay
* shape\_error (user can choose among avg\_30s, avg\_1min, avg\_2min or avg\_5min)

**🎯 Output Targets (2):**

* comm\_range
* comm\_failure

**⚙️ Hyperparameters Description**

**🔧 Model Architecture**

* **Hidden layer sizes**: [256, 128, 64, 32, 16]  
  Defines the number of neurons in each hidden layer. This structure allows the network to progressively reduce the dimensionality and extract higher-level features.
* **Number of layers**: 5  
  Total number of hidden layers, corresponding to the five values above.
* **Activation function**: ReLU  
  Applied between each hidden layer to introduce non-linearity.
* **Dropout rate**: 0.2  
  20% of neurons are randomly dropped during training to prevent overfitting.

**🏋️ Training Parameters**

* **Batch size**: 64  
  Number of samples processed before the model updates its weights.
* **Number of epochs**: 200  
  Maximum number of times the model sees the entire training dataset.
* **Early stopping patience**: 20  
  Stops training if the validation loss does not improve after 20 epochs, preventing overfitting and saving time.

**🧠 Optimization Settings**

* **Learning rate**: 0.01  
  Controls the step size in the direction of the gradient; a key factor in how fast the model learns.
* **Weight decay**: 0.01  
  L2 regularization term that penalizes large weights, helping to reduce overfitting.
* **Optimizer**: Adam  
  An adaptive optimizer that adjusts the learning rate for each parameter, often the best default choice.

**📉 Loss Function**

* **Loss**: Mean Squared Error (MSELoss)  
  Measures the average squared difference between the predicted and true values — standard for regression tasks.

**🧪 Preprocessing**

* **Input normalization**:  
  All input features (size\_formation, comm\_delay, shape\_error) are already scaled between 0 and 1, which is ideal for neural networks.