Pipe Roughness Estimation in Water Distribution Networks Using EPANET Net3

CIE 500 Research Project

Linyue Luo

05/05/2025

Why Pipe Roughness Prediction Matters?

Pipe roughness affects water pressure, flow efficiency, and energy use.

As pipes age, corrosion and scale buildup increase roughness, leading to:

- Higher pumping costs
- Greater risk of pipe failures
- Poor service quality for end users

Hazen-Williams Equations:

$$H_{n_j} - H_{n_i} = h_L = 10.667 C^{-1.852} d^{-4.871} L q^{1.852}$$

Net 3

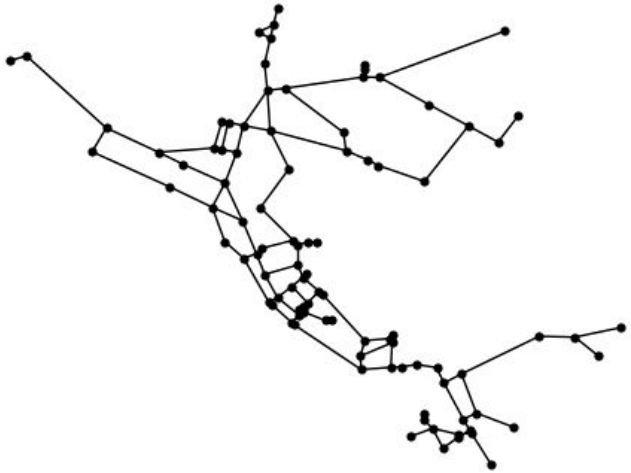
A benchmark dataset for **water distribution network analysis**.

Provides realistic **pipe attributes, flow, and pressure data** for testing methods.

Used in research for **hydraulic modeling and leak detection**.

Nodes: 97

Pipes: 117



What Problem Am I Solving?

Accurately **predicting pipe roughness** across a water distribution network.

Many roughness values in real-world systems are:

- **Assumed**, not measured
- **Outdated** due to aging, corrosion, or repairs

Manual inspection is expensive, slow, and often impossible in buried infrastructure.

Methodology

- **Graph Construction:** Represented the water network as a graph: Nodes and Edges
- **Feature Engineering:**

1) Pipe features: length, diameter, flowrate, headloss; Node features: elevation, demand.
2) Experimented with **“flipping”** node and edge features: Treated pipes (edges) as nodes in the GNN to enable edge-level prediction; Used node context as edge features and vice versa.
3) Applied normalization to all features and labels.

- **Model Architecture**: GCNConv layer
- **Training Strategy:**

1) Simulated real-world conditions by splitting pipes into “known” (training) and “unknown” (testing) sets.
2) Used mean squared error loss and Adam optimizer

- **Evaluation:** R2 Score, RMSE and Plots.

Experiments

Data Split

- Randomly selected a subset of pipes as “known” (training set) - 30%
- Remaining pipes treated as “unknown” (test set) for evaluating generalization - 70%

Normalization

- Assessed impact of normalization on model performance

```
# Modify the pipe selection to be more structured
num_pipes = len(wn.link_name_list)
num_known_pipes = int(0.3 * num_pipes)  # Use 30% of pipes as known
known_pipe_indices = np.random.choice(num_pipes, num_known_pipes, replace=False)
known_pipe_mask = np.zeros(num_pipes, dtype=bool)
known_pipe_mask[known_pipe_indices] = True
```

Node Features

- Elevation
- Base Demand

```python
# Add elevation,demand as node features
elevations = []
demands = []
for node in nodes:
    node_obj = wn.get_node(node)
    if node_obj.node_type == "Junction":
        ele = node_obj.elevation
        if len(node_obj.demand_timeseries_list) > 0:
            # Use the base value of the first demand pattern
            dem = node_obj.demand_timeseries_list[0].base_value
        else:
            # If no demand pattern exists, default to 0
            dem = 0.0
    elif node_obj.node_type == "Tank":
        ele = node_obj.elevation
    elif node_obj.node_type == "Reservoir":
        # a Reservoir node typically does not store its head as a direct numeric attribute (node_obj.head)
        ele = node_obj.head_timeseries.base_value
    else:
        ele = 0  # or some default
        dem = 0.0

    # print(node, node_obj.node_type, ele)  # Debug print if there is any "None" value
    elevations.append(ele)
    demands.append(dem)
```

Pipe Attributes

- Length
- Diameter
- Flow Rate
- Head Loss

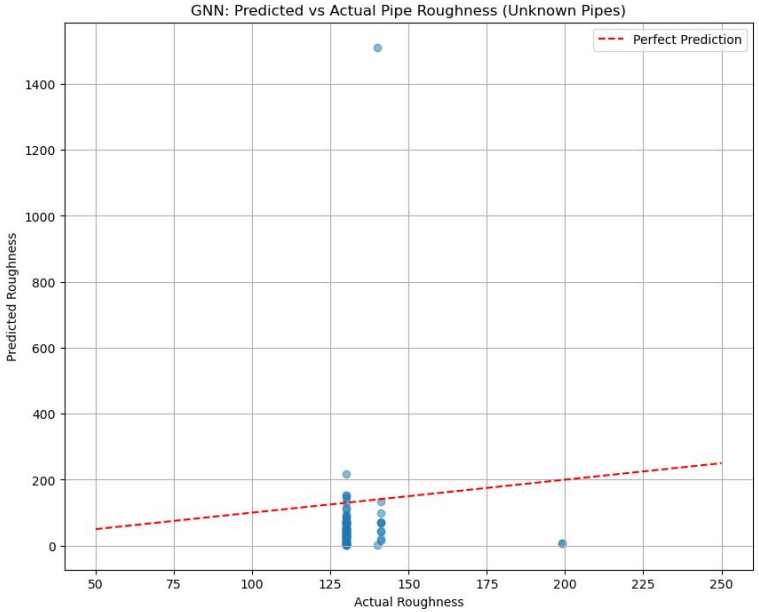
```python
edges = []
edge_attrs = []
roughness_label = []
known_mask = []  # Track which pipes are known

# Loop through each link in the model
for idx, (link_name, link) in enumerate(wn.links()):
    if link.link_type == 'Pipe':
        src_idx = node_mapping[link.start_node_name]
        dst_idx = node_mapping[link.end_node_name]
        length = link.length
        diameter = link.diameter
        roughness = link.roughness
        flowrate = abs(Q_flowrate[link_name].mean())

        if flowrate > 0 and diameter > 0 and roughness > 0:
            # Hazen-Williams for head loss calculation
            headloss = (10.67 * length * flowrate**1.852) / (roughness**1.852 * diameter**4.87)
        else:
            headloss = 0.0
        # Append edge
        edges.append((src_idx, dst_idx))
        # If an undirected graph, append (dst_idx, src_idx)
        # Append edge attributes
        edge_attrs.append([length, diameter, flowrate, headloss])
        roughness_label.append(roughness)
        known_mask.append(known_pipe_mask[idx])
```

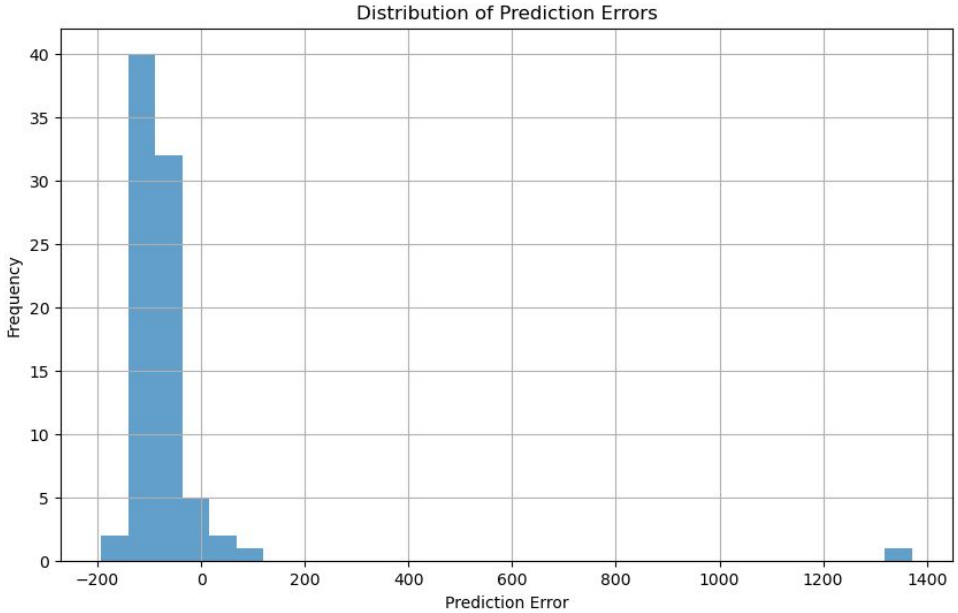
Model

```python
class RoughnessGNN(nn.Module):
    def __init__(self, in_channels, hidden1_channels, hidden2_channels):
        super(RoughnessGNN, self).__init__()
        self.conv1 = GCNConv(in_channels, hidden1_channels)
        self.conv2 = GCNConv(hidden1_channels, hidden2_channels)
        self.dropout = nn.Dropout(0.2)
        self.fc = nn.Linear(hidden2_channels, 1)

    def forward(self, x, edge_index, edge_attr):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.dropout(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = self.dropout(x)
        x = self.fc(x)
        return x

# Initialize model and training
model = RoughnessGNN(in_channels=edge_attr.shape[1], hidden1_channels=32, hidden2_channels=64)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)
criterion = nn.MSELoss()
```

Results-Before Normalization

Evaluation Metrics for Unknown Pipes:
R^2 Score: -259.4197
RMSE: 177.2671





Normalization Process

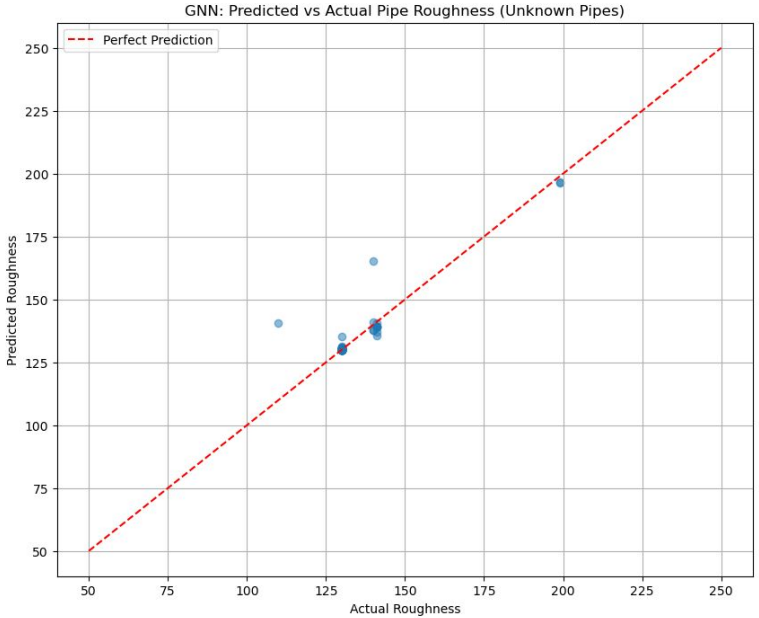
```
StandardScaler()
.fit_transform(X, y=None, **fit_params)
.inverse_transform(X, copy=None)
```

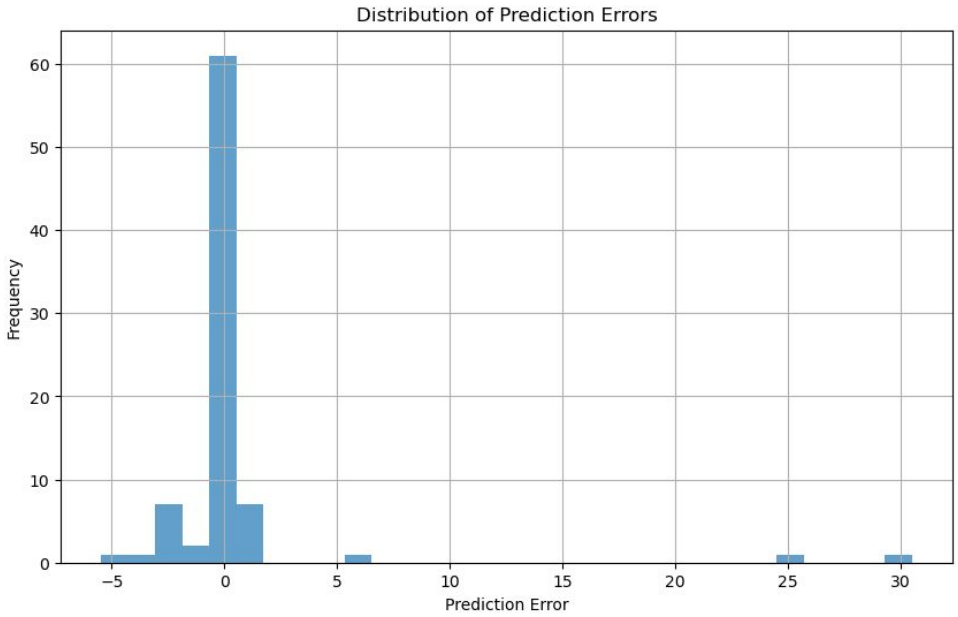
The standard score of a sample x is calculated as:

z = (x - u) / s

Results-After Normalization

Evaluation Metrics for Unknown Pipes:
R^2 Score: 0.8399
RMSE: 4.5393





Next Steps:

- Modify structure of model
- Add more input data
- Enlarge dataset
- Other methods for the prediction: conv.NNConv (supports edge features)
- Flow Rate and Demand are changing over time, how to utilize them?
- Deal with Edge List when flipping node features and edge attributes.