Supplementary Material of "RGCNU: Recurrent Graph Convolutional Network With Uncertainty Estimation for Remaining Useful LifePrediction"

Qiwu Zhu, Qingyu Xiong, Zhengyi Yang, and Yang Yu

Two-layer GCN: The graph information of nodes with two-layer GCN can be formulated as

$$f(x_t, A) = \mathrm{ReLU}\left(\hat{A}\left(\hat{A}x_t W_0\right)W_1\right) \tag{10}$$

where $x_t \in \mathbb{R}^{N\times 1}$ is the features of the equipment at time slice $t \in \{1,\ldots,F\}$, $X = \{x_1, x_2, \ldots, x_F\}$; $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \in \mathbb{R}^{N\times N}$ is the re-normalization trick, $\tilde{A} = A + I \in \mathbb{R}^{N\times N}$ means the adjacency matrix A plus a self-loop, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \in \mathbb{R}^{N\times N}$ is the corresponding degree matrix of $\tilde{A}$. Also, $W_0 \in \mathbb{R}^{F\times J}$ and $W_1 \in \mathbb{R}^{J\times K}$ represent the parameter matrix from the input feature dimension F to the output feature dimension J and K, respectively.

TDL computation process: The computation process can be summarized as follows:

$$\begin{aligned}
I_t &= \sigma(W_{zi} * Z_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1}) \\
F_t &= \sigma\left(W_{zf} * Z_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1}\right) \\
O_t &= \sigma(W_{zo} * Z_t + W_{ho} * H_{t-1} + W_{co} \odot C_t) \\
C_t &= F_t \odot C_{t-1} + I_t \odot \tanh(W_{zc} * Z_t + W_{hc} * H_{t-1}) \\
H_t &= O_t \odot \tanh(C_t)
\end{aligned} \tag{11}$$

where $Z_t \in \mathbb{R}^{N\times K}$ denotes the output of SCL, $t \in \{1,\ldots,F\}$, I_t, F_t, H_{t-1}, O_t and C_t denote the input gate, the forget gate, the previous hidden state, the output gate and the final memory cell respectively, h_t is the final hidden state, $\odot$ is the Hadamard product, $*$ is the convolution operation, $W_{\alpha\beta}(\alpha \in z, h, c,\ \beta \in i, f, o, c)$ denotes the learnable parameter, and σ is the activation function.

Loss function: Different from the traditional loss function focusing on the regression value, we assume that the predicted RUL shows a Gaussian distribution with mean $\hat{y}_i$ and variance $\hat{\sigma}_i$ after considering the data uncertainty, then the loss function is a likelihood function, given by

$$p_\theta(y \mid \mathbf{x}) = \prod_{i=1}^{M} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(\hat{y}_i - y_i)^2}{2\hat{\sigma}_i^2}}. \tag{12}$$

The negative log likelihood is used as follows:

$$\mathcal{L}(\theta) = -\log p_\theta(y \mid \mathbf{x}) = \sum_{i=1}^{M} \frac{(\hat{y}_i - y_i)^2}{2\hat{\sigma}_i^2} + \frac{1}{2}\log\left(\sigma_i^2\right). \tag{13}$$

Considering the numerical stability for regressing the variance $\hat{\sigma}_i^2$ and avoiding dividing the loss function by zero, we train the network to predict the log variance, $s_i := \log\hat{\sigma}_i^2$. The exponential mapping also allows the network to regress unconstrained scalar values, which means $\exp(-s_i)$ is resolved to the positive domain giving valid values for variance.

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{i=1}^{M} \frac{1}{2}\exp(-s_i)\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + \frac{1}{2}s_i. \tag{14}$$

From (14), the RUL prediction model estimates large s_i to temper the error term, $\exp(-s_i)\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$, instead of overfitting on these noisy samples. The prediction model is discouraged from predicting large s_i for all samples, which may lead to underfitting of $\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$ and, in turn, larger s_i term punishes the model. This makes the model adapt the s_i, and allows the model to learn to attenuate the effect from noisy samples.

Furthermore, considering that the unconstrained s_i may explode and result in large degradation uncertainties, an L2 regularization term of s_i is added to the loss function. The final loss function can be obtained as (7).

Experimental details:

Datasets: C-MAPSS contains four datasets, each of which includes training and test data. Table 3 describes the size of the data sets. After obtaining the model through cross-validation on training data, the trained model is then evaluated on test data.

Table 3. Detail of the Experimental Data

	FD001	FD002	FD003	FD004
Training units	100	260	100	249
Test units	100	259	100	248

Implementation details: We train RGCNU by Adam Optimizer with learning rate $1\mathrm{E}{-3}$, λ is 0.15, 0.2, 0.25 and 0.13, respectively for the four data sets. The windows size is 30, 20, 30 and 18, respectively. L is 10. The hidden units in TDL are 64. The other hyper-parameters are shown in Table 4.

Table 4. Default Hyper-Parameters of RGCNU

Parameter	Epoch	Dropout	Weight decay	α	J/C	Φ_1	FC_1/FC_2
Value	150	0.8	$5\mathrm{E}{-4}$	3	64	64	14/1

Evaluation metrics: The main difference between the two metrics is that the score function penalizes later predictions more heavily (i.e., the predicted RUL is larger than the actual RUL), whereas RMSE places equal weight on early and later predictions.

$$\begin{aligned}
\mathrm{RMSE} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\Delta_i)^2} \\
\mathrm{Score} &= \sum_{i=1}^{n} s_i, \quad s_i = \begin{cases} e^{-\frac{\Delta_i}{13}} - 1, & \text{for } \Delta_i < 0 \\ e^{\frac{\Delta_i}{10}} - 1, & \text{for } \Delta_i \geq 0 \end{cases}
\end{aligned} \tag{15}$$

where n is the number of test samples, $\Delta_i = \hat{y}_i - y_i$, $\hat{y}_i$ is the predicted value, and y_i is the true value.