UNIVERSITY OF TORONTO | Engineering

Hybrid approach for RUL estimation using Deep Learning

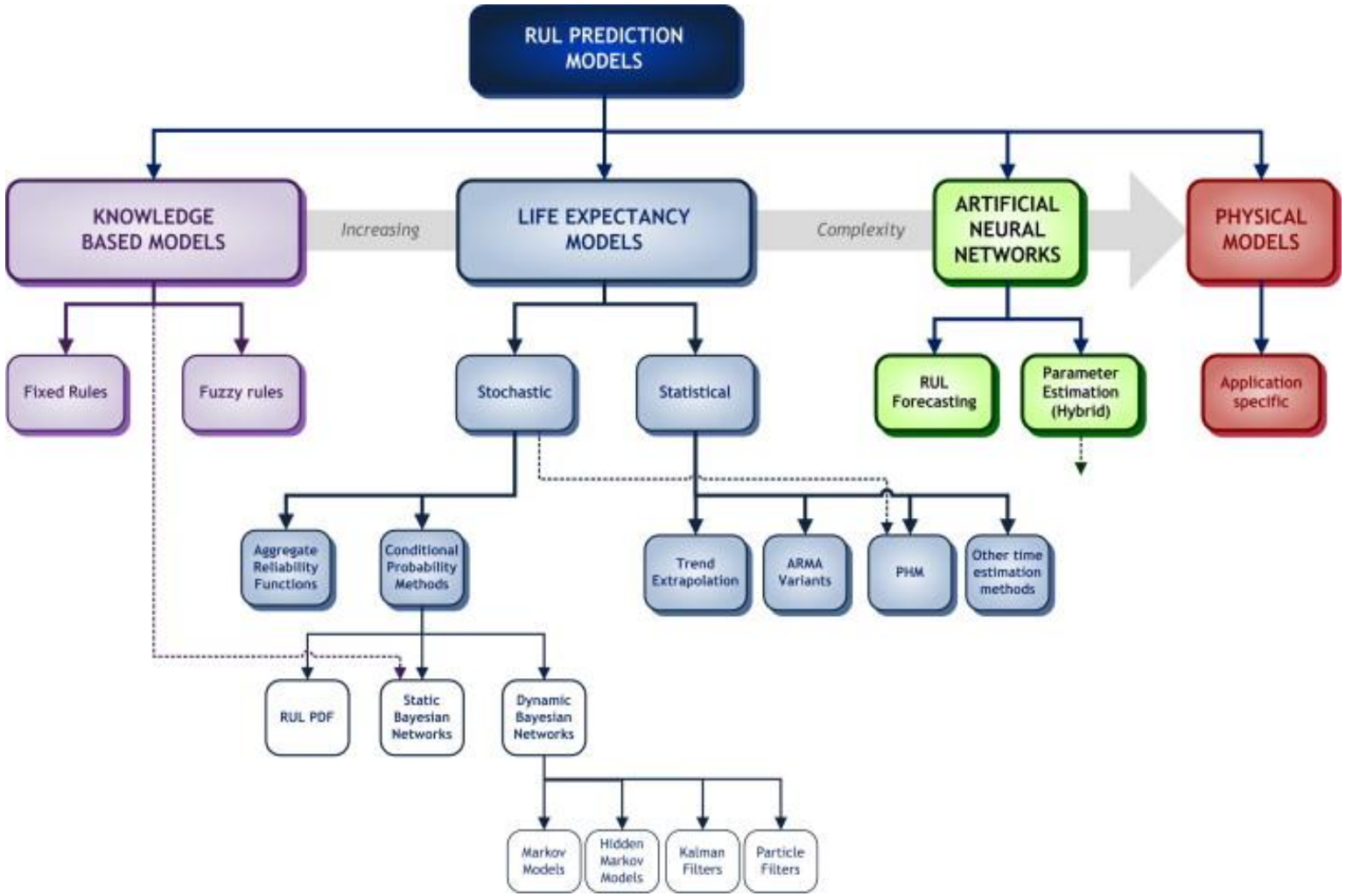
Remaining Useful Life

- The Remaining Useful Life (RUL) is a subjective estimate of the number of remaining years that an entity is estimated to be able to function in accordance with its intended purpose before failure

- Primary incentives for pursuing estimation of RUL include

1. Ease in sale/purchase of equipment
2. Creating robust Predictive Maintenance Policies
3. RUL is valuable in certain wear-based failures, such as heat exchanger fouling, ethane cracker furnace tubes coking and reactor and catalyst deactivation, in terms of equipment usage, and safety

RUL Estimation



Developing Hybrid Approach

- The main aim of the project is to develop a hybrid prognostic framework in which deep learning-based trajectory prediction enables the degradation model (based on stochastic process to address unit-to-unit variability) to overcome the limitations of pre-set trajectories of the traditional degradation models

- The DNN models discussed are:

1. Convolutional Neural Network (CNN)
2. CNN-LSTM
3. Bayesian Neural Network- Bayes by Backprop (BBP)
4. Bayes by Backprop (BBP) + MC dropout
5. BBP + Stochastic Gradient Langevin Dynamics

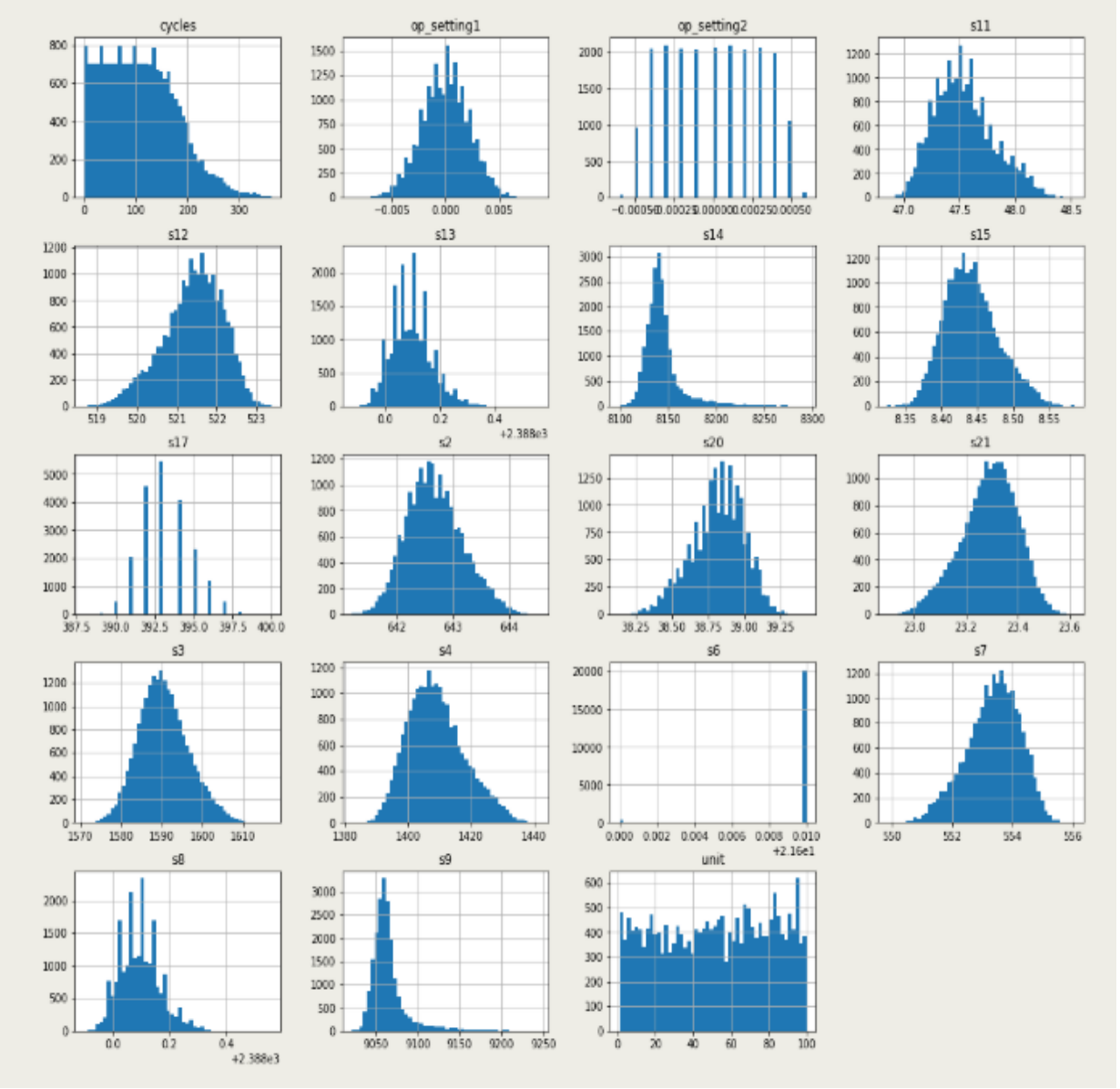
About the Dataset

NASA's Turbofan Engine Degradation Simulation Data Set containing engine degradation simulation using C-MAPSS is used.

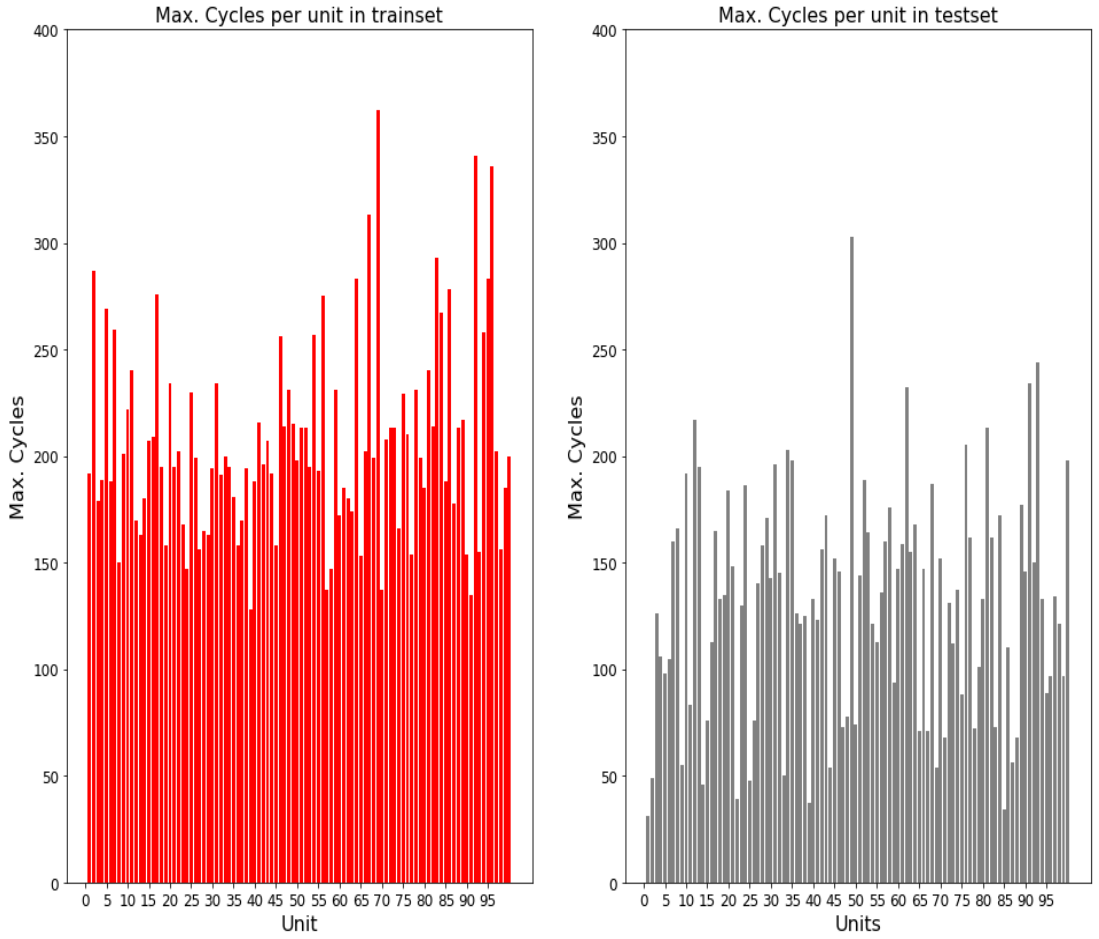
Here, the dataset FD001 is used where 100 simulation of same make units are in the sea level condition with failure due to HPC degradation fault.

The features include:

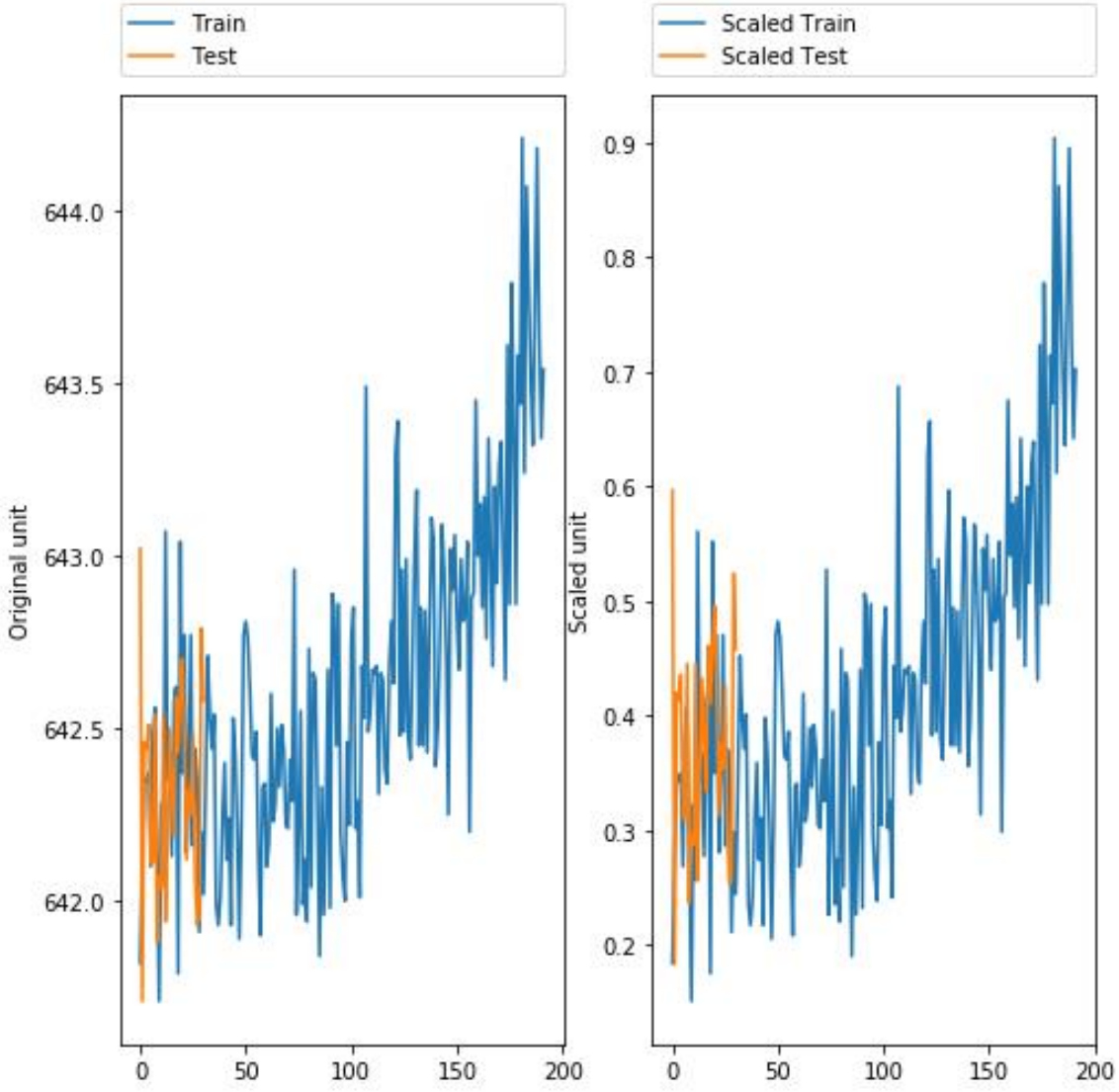
1) unit number
2) time, in cycles
3) 3 ideal operational settings
4) 26 sensor measurements



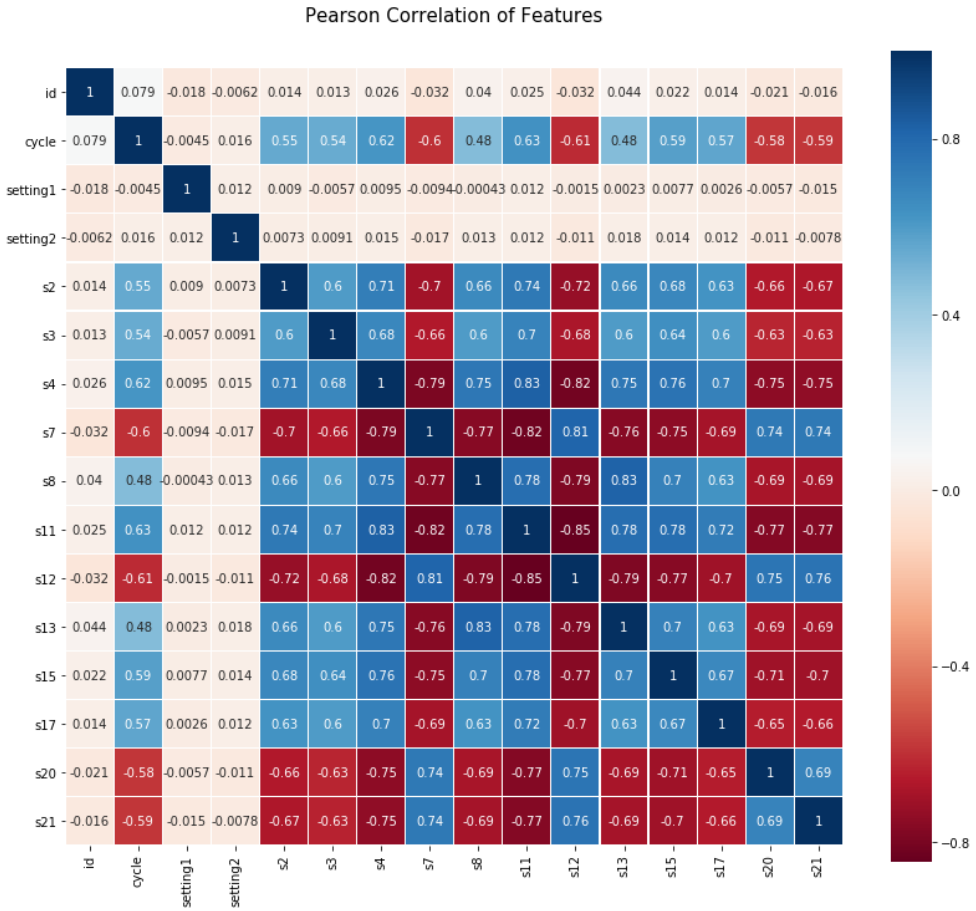
Data Processing

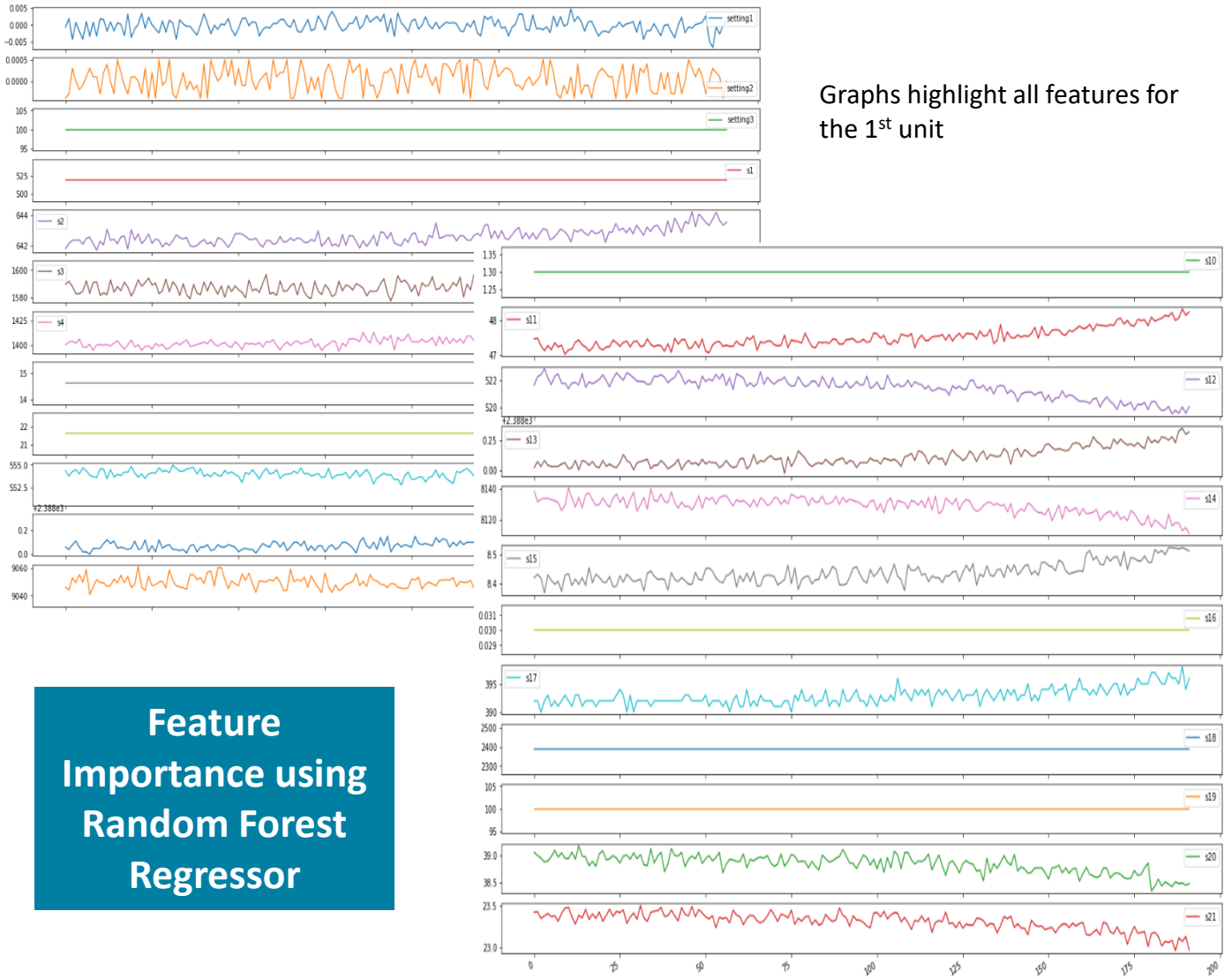


Comparison of Scaled and Original Feature Unit



Feature Selection





Graphs highlight all features for the 1st unit

Feature Importance using Random Forest Regressor

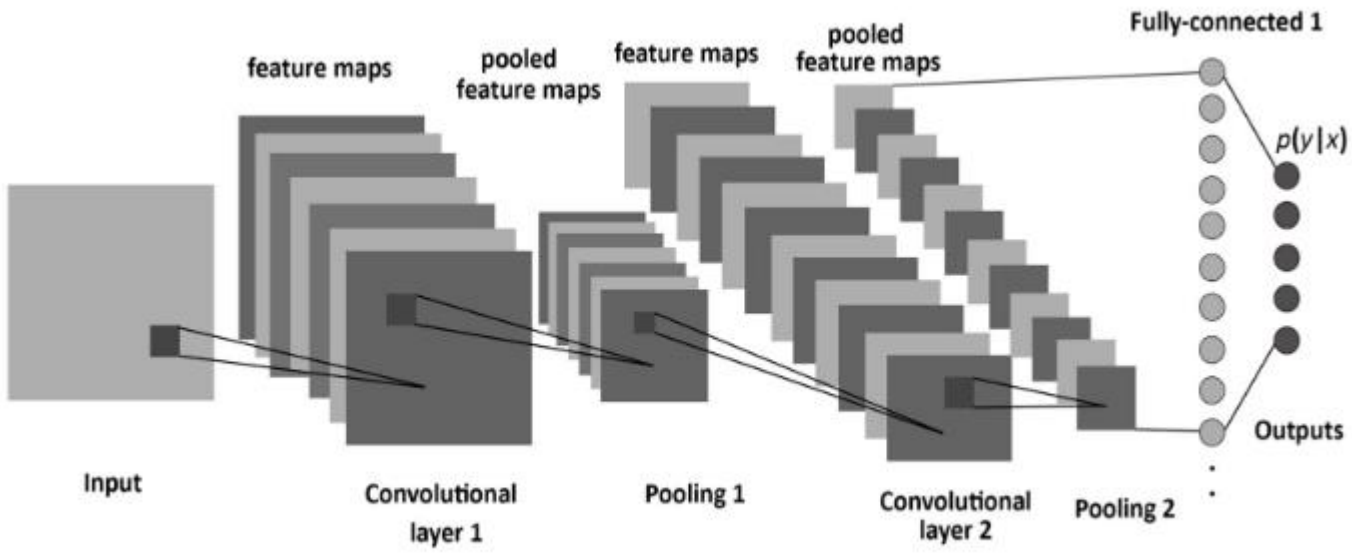
Convolutional Neural Network

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input data, assign importance (learnable weights and biases) to various aspects/objects in the data and be able to perform regression or classification on it.

Steps:

1. Provide input data into convolution layer
2. Choose parameters, apply filters with strides, padding if requires. Perform convolution on the data matrix and apply ReLU activation to the matrix.
3. Perform pooling to reduce dimensionality size
4. Add as many convolutional layers until satisfied
5. Flatten the output and feed into a fully connected layer (FC Layer)
6. Output the class using an activation function

RMSE: 30.57



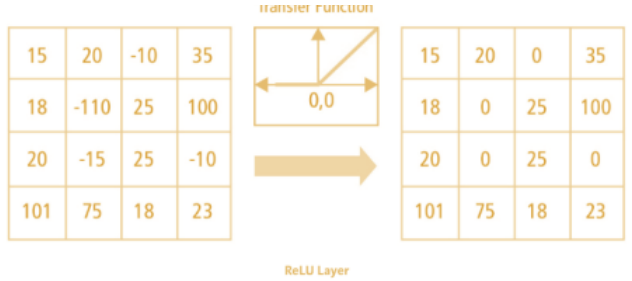
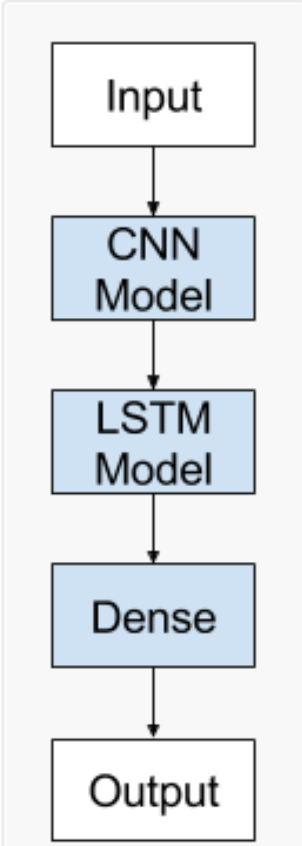


Figure 7 : ReLU operation

CNN-LSTM

- The CNN LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs (Long Short-Term Memory) to support sequence prediction.

- RMSE: 29.12



Convolutional Neural Network Long Short-Term Memory Network Architecture

Bayesian Neural Network- Bayes by Backprop (BBP)

Neural Networks as a conditional model

$$p_\theta(y|x)$$

Bayes' theorem:

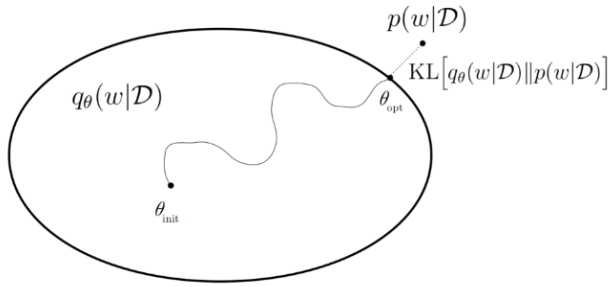
$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\ p(\theta)}{p(\mathcal{D})}$$

Bayesian inference

$$p(y^\star|x^\star, \mathcal{D}) = \int p(y^\star|x^\star, \theta)\ p(\theta|\mathcal{D}) d\theta.$$

Bayes by Backprop

$$q_\theta(w|\mathcal{D}) \approx p(w|\mathcal{D})$$



This is realized by minimizing the Kullback-Leibler (KL) divergence between p and q, what can be seen as an optimization problem:

$$\theta_{opt} = \arg\min_\theta \mathrm{KL}\Big[q_\theta(w|\mathcal{D})\|p(w|\mathcal{D})\Big]$$

KL-divergence

$$\mathrm{KL}\big[q_\theta(w|\mathcal{D})\|p(w|\mathcal{D})\big] = \int q_\theta(w|\mathcal{D}) \log \frac{q_\theta(w|\mathcal{D})}{p(w|\mathcal{D})} dw$$

Hence, we arrive at a tractable objective function:

$$\theta_{opt} = \arg\min_\theta \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

$$\text{with sample } w^{(i)} \text{ from } q_\theta(w|\mathcal{D})$$

Local Reparamerisation Trick

$$\theta = (\mu, \sigma^2)$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

$$f(\epsilon) = w = \mu + \sigma \cdot \epsilon$$

We have these two parameters of interest incorporated in every weight value and can both calculate the derivative of it and re-write it into a probability distribution. Our parameters are then updated, i.e. learnt, according to:

$$\Delta\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu}$$

$$\Delta\sigma = \frac{\partial f}{\partial w}\ \frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

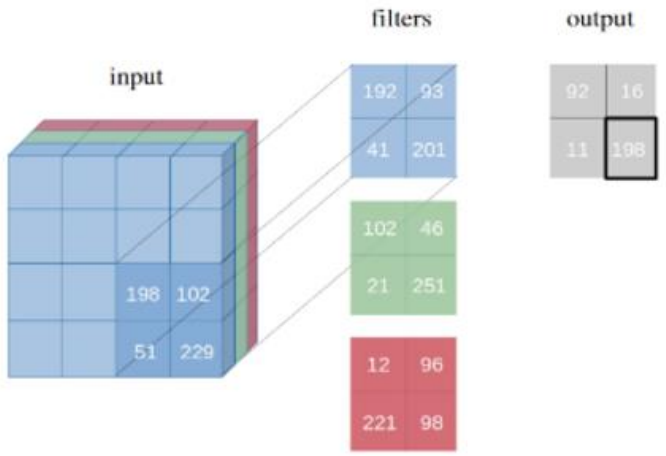
$$\mu \leftarrow \mu - \alpha\Delta\mu$$

$$\sigma \leftarrow \sigma - \alpha\Delta\sigma$$

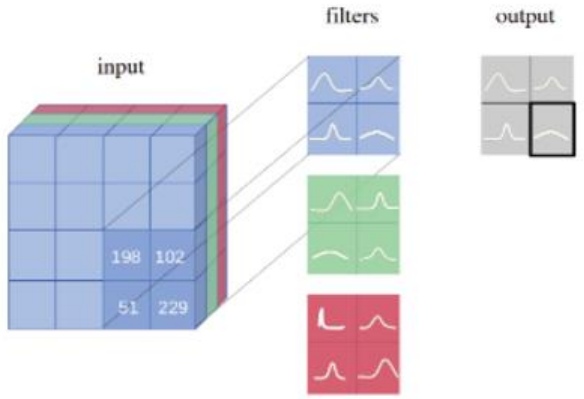
$$\theta^{opt} = (\mu^{opt}, \sigma^{opt})$$

Bayesian Neural Network

CNN:



BNN:



Local reparameterisation trick for convolutional layers

$$q_\theta(w_{ijhw}|\mathcal{D}) = \mathcal{N}(\mu_{ijhw}, \alpha_{ijhw}\mu_{ijhw}^2) \quad [1]$$

$$b_j = A_i * \mu_i + \epsilon_j \odot \sqrt{A_i^2 * (\alpha_i \odot \mu_i^2)} \quad [2]$$

PyTorch

PYRO

RMSE: 23.14

Bayes by Backprop (BBP) + MC dropout

- In this framework, the model casts dropout training in deep neural networks (NNs) as approximate Bayesian inference in deep Gaussian processes
- This mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy
- RMSE: 24.63

BBP + Stochastic Gradient Langevin Dynamics

- In this variant of BBP, by adding the right amount of noise to a standard stochastic gradient optimization algorithm, the iterates converge to samples from the true posterior distribution.
- This seamless transition between optimization and Bayesian posterior sampling provides an inbuilt protection against overfitting
- RMSE: 28.83

Next Steps…

- Evaluate and optimize the DNN models
- Develop Model-based approach using Wiener Process for comparative analysis
- Development of Hybrid Model

Thank you