Applications of Remote & Real-Time Sensing, Artificial Intelligence, and Edge Computing in Structural/Wind Engineering

Thomas Kang

Seoul National University

Contents

Concrete compressive strength prediction using ML

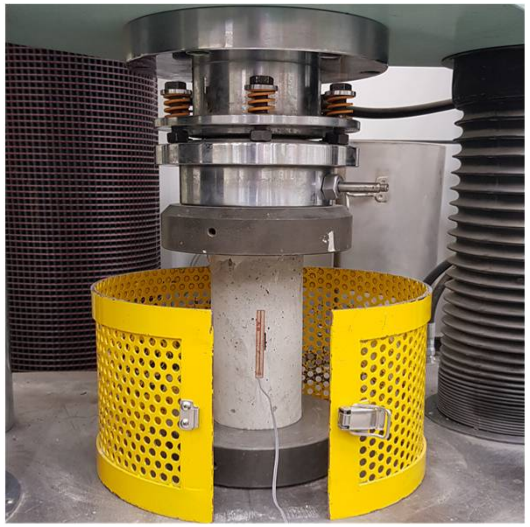
Concrete Compressive Strength Prediction Using ML

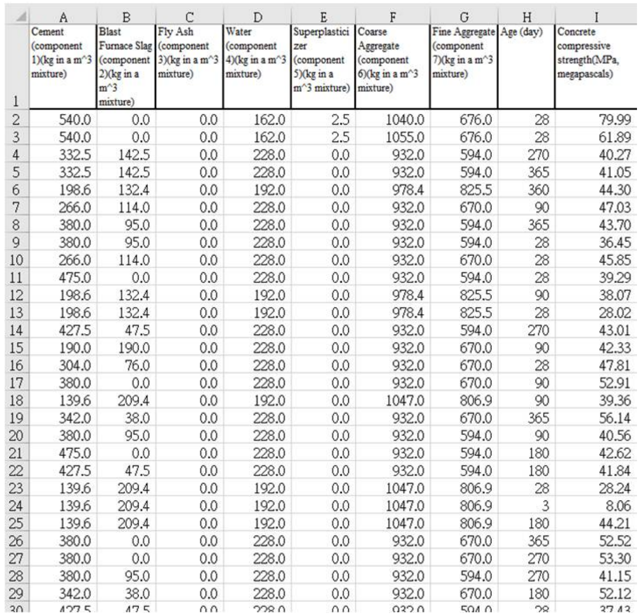
Condition	OPC w/o admixture	Concrete w/o AE	Concrete w/ AE	OPC	High early strength Portland Cement	Moderate heat Portland Cement
Estimation Equation	$\frac{w}{c} = \frac{215}{f_{28} + 210}$	$\frac{w}{c} = \frac{23}{f_{28} + 13.9}$	$\frac{w}{c} = \frac{16.2}{f_{28} + 7.4}$	$\frac{w}{c} = \frac{51}{\frac{f_{28}}{k} + 0.31}$	$\frac{w}{c} = \frac{41}{\frac{f_{28}}{k} + 0.17}$	$\frac{w}{c} = \frac{66}{\frac{f_{28}}{k} + 0.64}$
Reference	Korea Concrete Standard Specification (1999)	US ACI 211.1 (1993)		Japan Construction Standard Specification (2003) $^*k$ denotes the strength of cement		

Concrete Compressive Strength Prediction Using ML

- **Dataset was created by combining 38 data from experiments and 1030 data from open-source.**

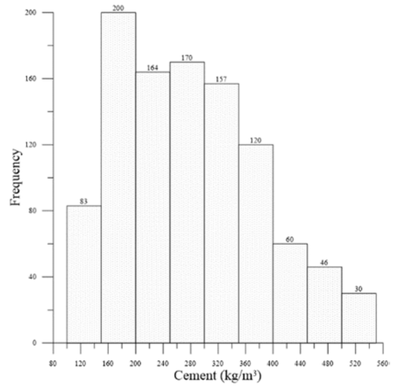
- **Total dataset (1068) was divided into training set (748; 70%) and testing set (320; 30%).**



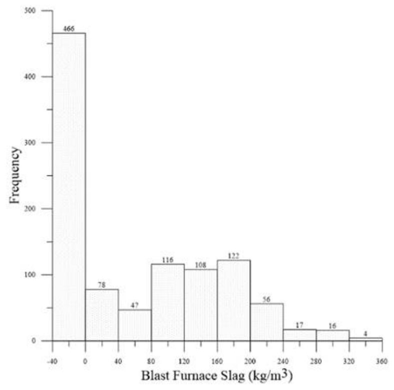


	A	B	C	D	E	F	G	H	I
1	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5)(kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Fine Aggregate (component 7)(kg in a m^3 mixture)	Age (day)	Concrete compressive strength(MPa, megapascals)
2	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
3	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
4	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
5	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
6	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
7	266.0	114.0	0.0	228.0	0.0	932.0	670.0	90	47.03
8	380.0	95.0	0.0	228.0	0.0	932.0	594.0	365	43.70
9	380.0	95.0	0.0	228.0	0.0	932.0	594.0	28	36.45
10	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85
11	475.0	0.0	0.0	228.0	0.0	932.0	594.0	28	39.29
12	198.6	132.4	0.0	192.0	0.0	978.4	825.5	90	38.07
13	198.6	132.4	0.0	192.0	0.0	978.4	825.5	28	28.02
14	427.5	47.5	0.0	228.0	0.0	932.0	594.0	270	43.01
15	190.0	190.0	0.0	228.0	0.0	932.0	670.0	90	42.33
16	304.0	76.0	0.0	228.0	0.0	932.0	670.0	28	47.81
17	380.0	0.0	0.0	228.0	0.0	932.0	670.0	90	52.91
18	139.6	209.4	0.0	192.0	0.0	1047.0	806.9	90	39.36
19	342.0	38.0	0.0	228.0	0.0	932.0	670.0	365	56.14
20	380.0	95.0	0.0	228.0	0.0	932.0	594.0	90	40.56
21	475.0	0.0	0.0	228.0	0.0	932.0	594.0	180	42.62
22	427.5	47.5	0.0	228.0	0.0	932.0	594.0	180	41.84
23	139.6	209.4	0.0	192.0	0.0	1047.0	806.9	28	28.24
24	139.6	209.4	0.0	192.0	0.0	1047.0	806.9	3	8.06
25	139.6	209.4	0.0	192.0	0.0	1047.0	806.9	180	44.21
26	380.0	0.0	0.0	228.0	0.0	932.0	670.0	365	52.52
27	380.0	0.0	0.0	228.0	0.0	932.0	670.0	270	53.30
28	380.0	95.0	0.0	228.0	0.0	932.0	594.0	270	41.15
29	342.0	38.0	0.0	228.0	0.0	932.0	670.0	180	52.12
30	427.5	47.5	0.0	228.0	0.0	932.0	594.0	28	37.43

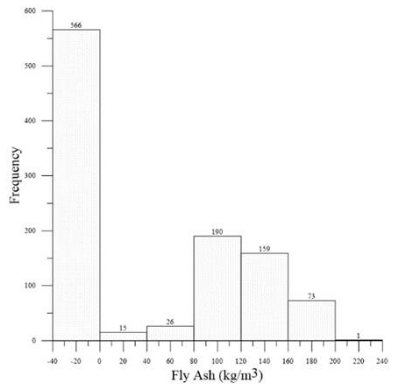
Concrete Compressive Strength Prediction Using ML



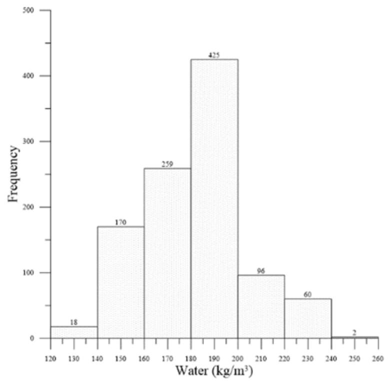
Cement



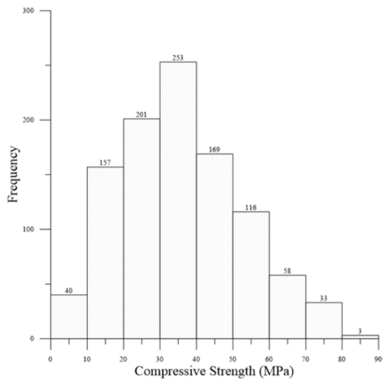
Blast furnace slag



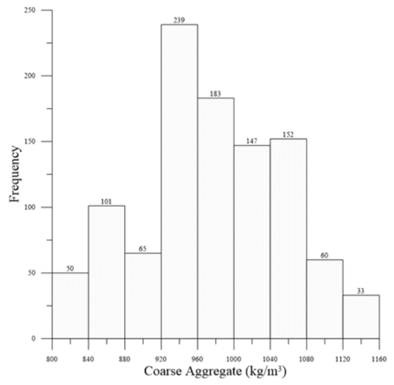
Fly ash



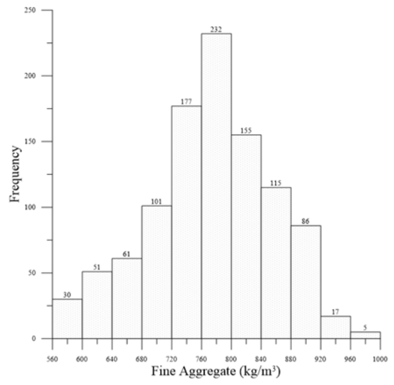
Water



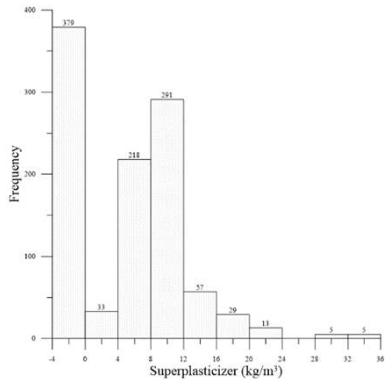
Compressive strength



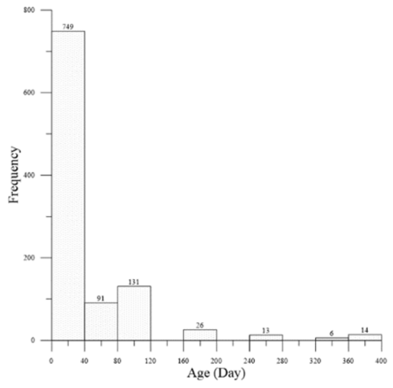
Coarse aggregate



Fine aggregate



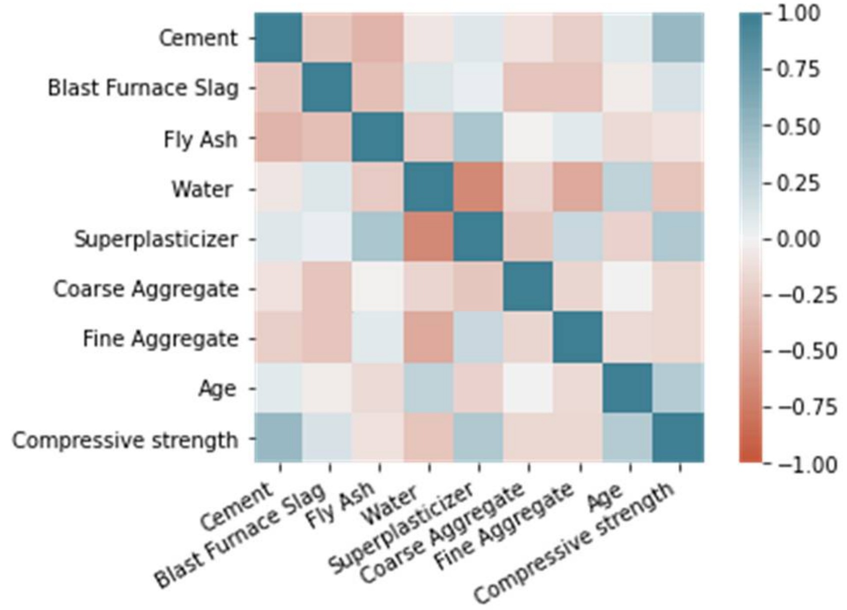
Superplasticizer



Age

Concrete Compressive Strength Prediction Using ML

- **Heat map visualization**



Compressive strength (target)

1) Positive effect

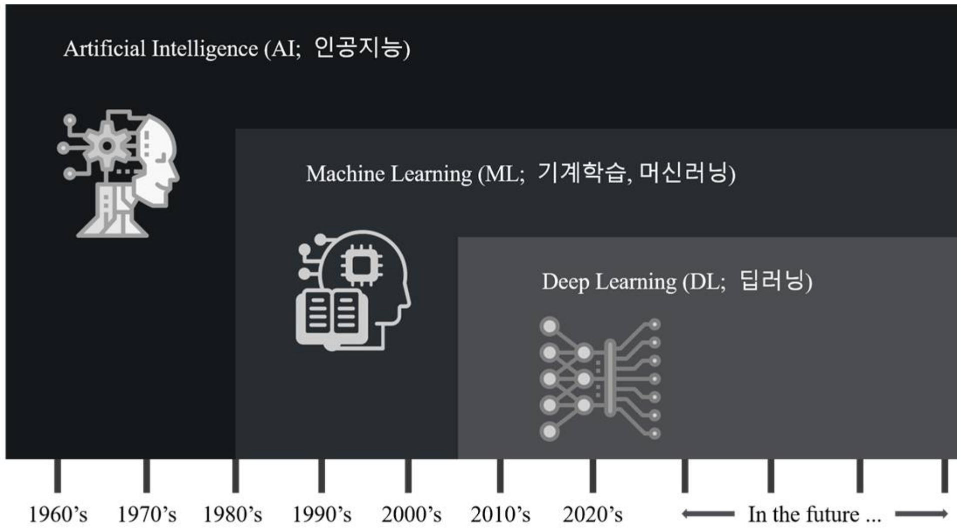
: Cement, blast furnace slag, superplasticizer, and age

2) Negative effect

: Fly ash, water, coarse aggregate, and fine aggregate

Concrete Compressive Strength Prediction Using ML

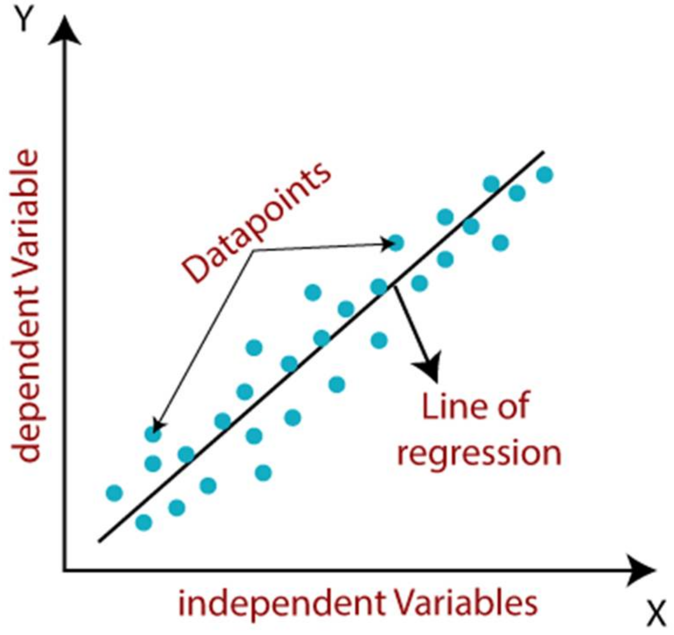
- **In this study, several machine learning models were introduced to compare regression performance.**


Artificial Intelligence (AI; 인공지능)
Machine Learning (ML; 기계학습, 머신러닝)
Deep Learning (DL; 딥러닝)
1960's
1970's
1980's
1990's
2000's
2010's
2020's
In the future ...

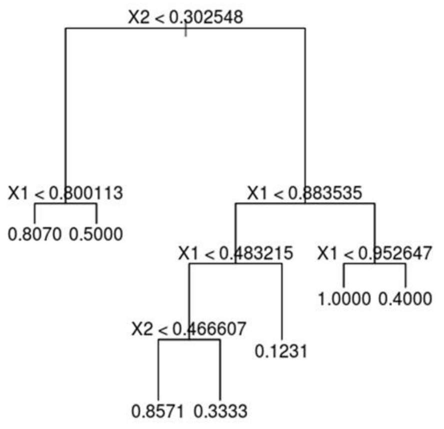
- **Linear Regression**
- **Decision Tree**
- **Ensemble Tree**
- **Support Vector Machine**
- **Gaussian Process Regression**
- **Neural Network (Deep Learning)**

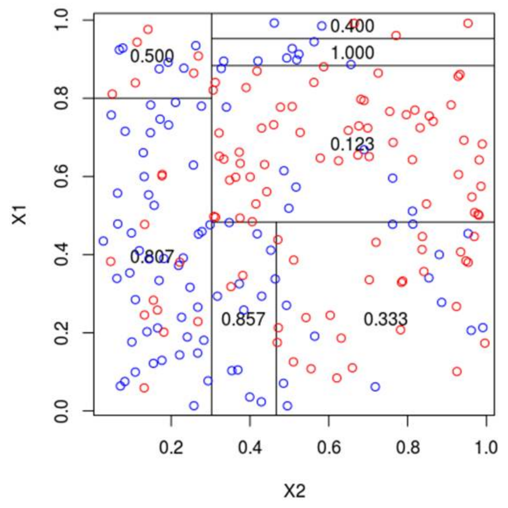
Concrete Compressive Strength Prediction Using ML

- **Linear regression**



- **Decision tree**

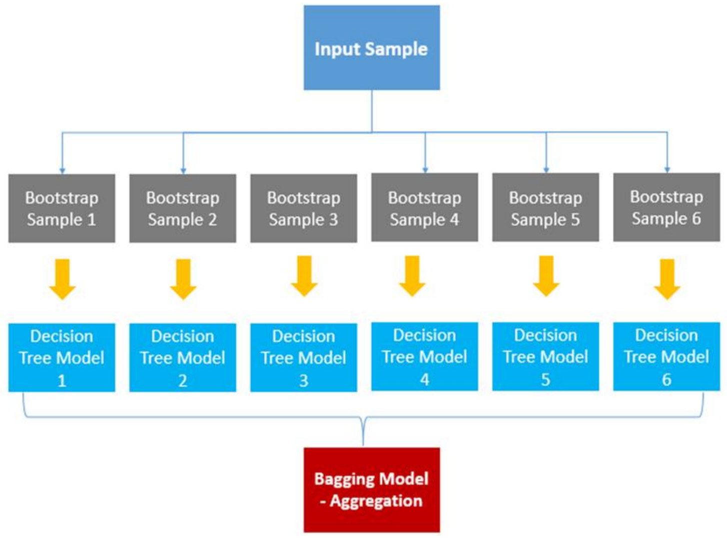


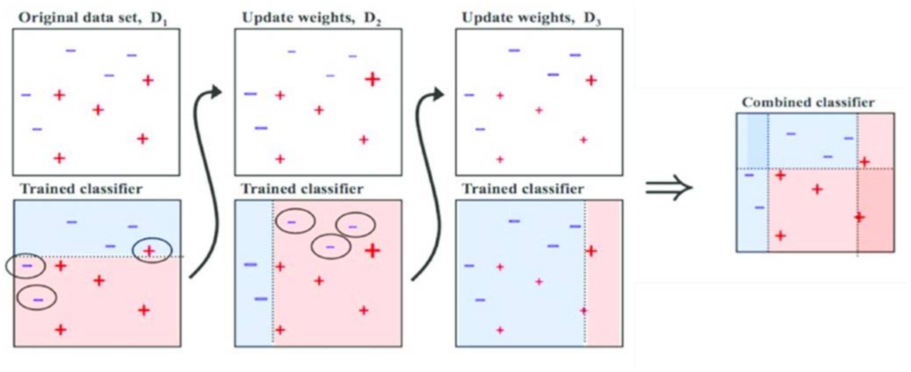


https://www.ingeniovirtual.com/descubrezapatillas/tags/4268?p=6.28.4121027.1.25.43.supervised+learning+linear+regression
https://iprathore71.medium.com/complete-guide-to-decision-tree-cee0238128d

Concrete Compressive Strength Prediction Using ML

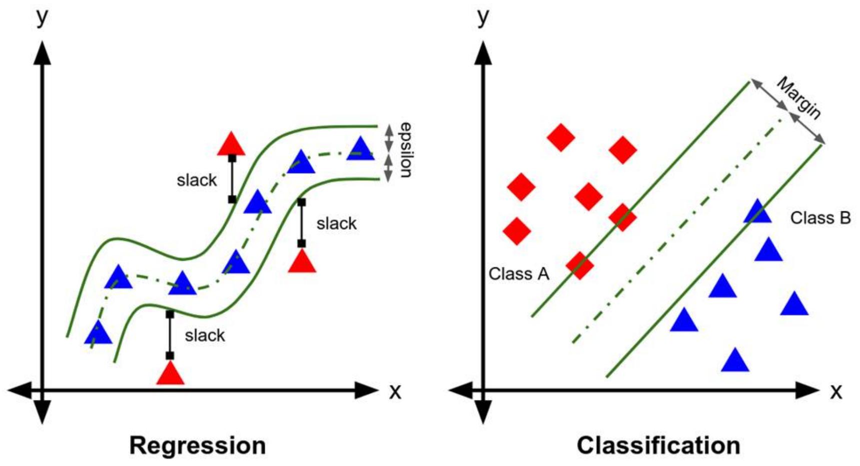
- **Ensemble (Bagged trees)**
- **Ensemble (Boosted trees)**



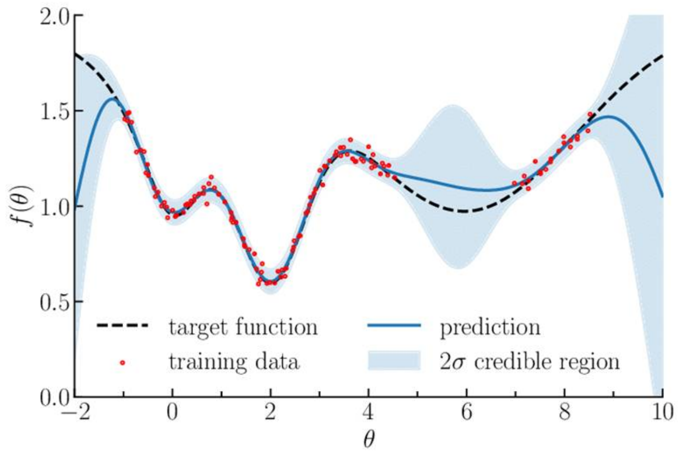


Concrete Compressive Strength Prediction Using ML

- **Support Vector Machine**

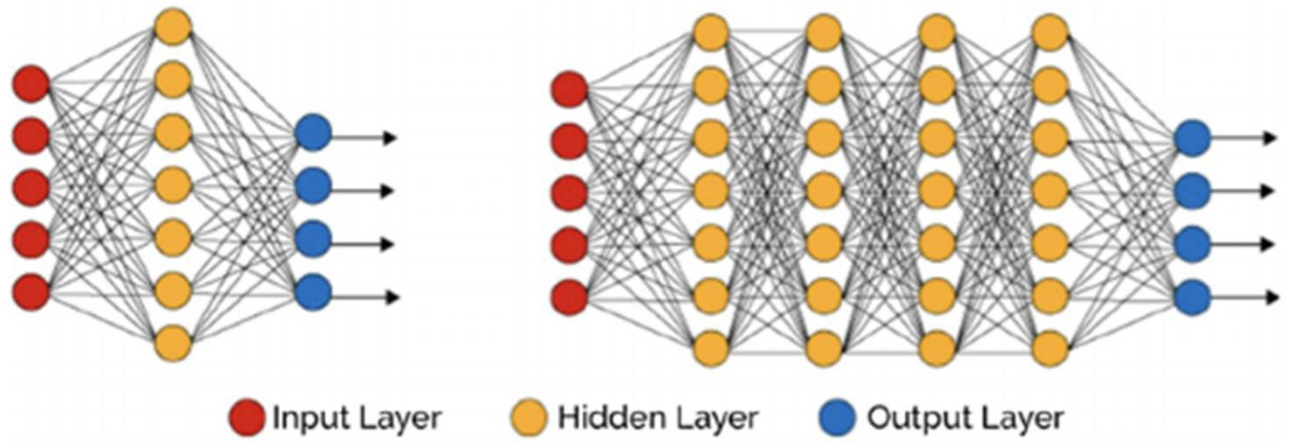


- **Gaussian process regression**



Concrete Compressive Strength Prediction Using ML

- **Neural network**



Concrete Compressive Strength Prediction Using ML

- **Data preprocessing (normalization)**

: Min-max scaler

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **How to evaluate model performance?**

1) RMSE

2) R-square

3) MAE

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

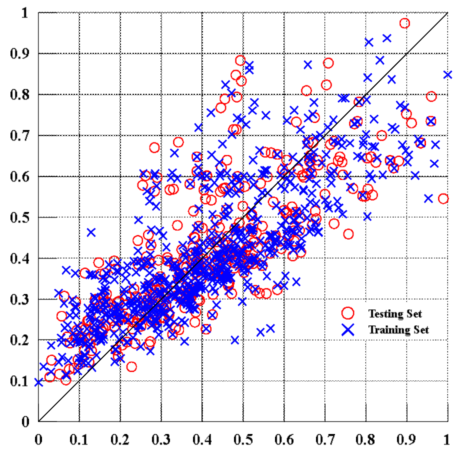
Where,

$\hat{y}$ $-$ *predicted value of y*

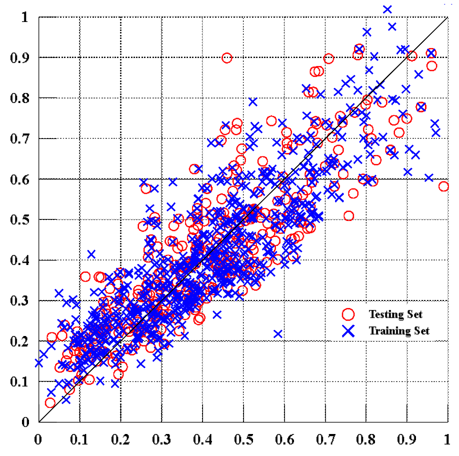
$\bar{y}$ $-$ *mean value of y*

Concrete Compressive Strength Prediction Using ML

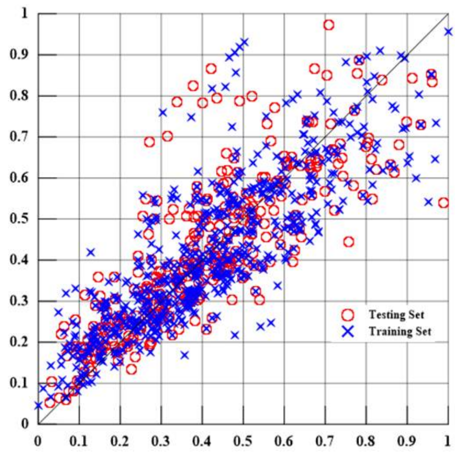
- **Linear Regression (LR)**



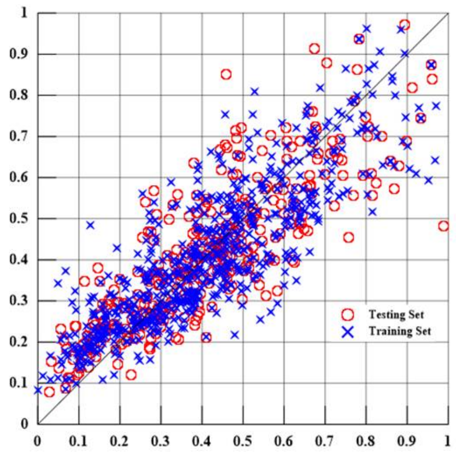
Simple LR



Interactions LR



Robust LR



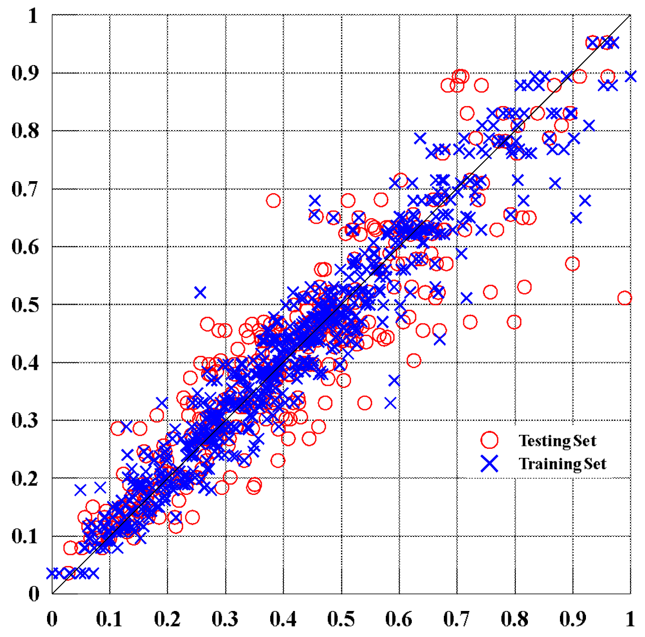
Stepwise LR

Concrete Compressive Strength Prediction Using ML

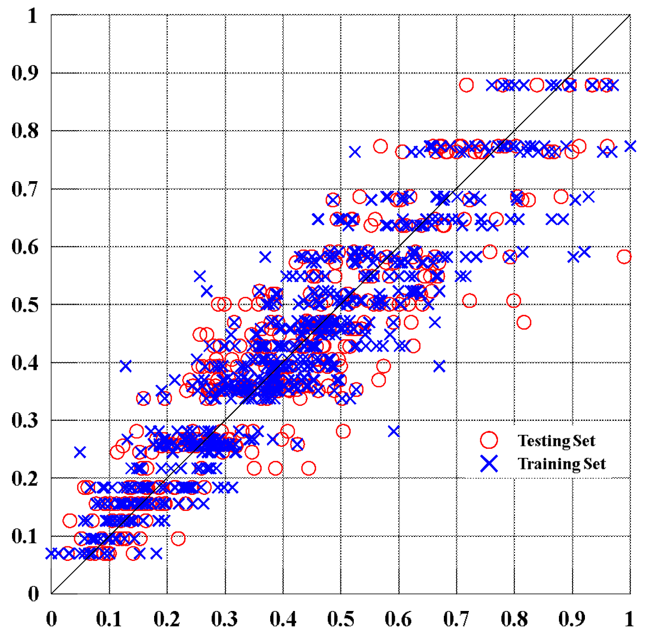
Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Linear Regression (LR)	Simple LR	0.1286	0.1325	0.6236	0.5926	0.1006	0.1035
	Interactions LR	0.1051	0.1085	0.7554	0.7269	0.0821	0.0858
	Robust LR	0.1109	0.2119	0.8326	-0.0415	0.1060	0.1190
	Stepwise LR	0.1095	0.1116	0.7296	0.6850	0.0848	0.0894

Concrete Compressive Strength Prediction Using ML

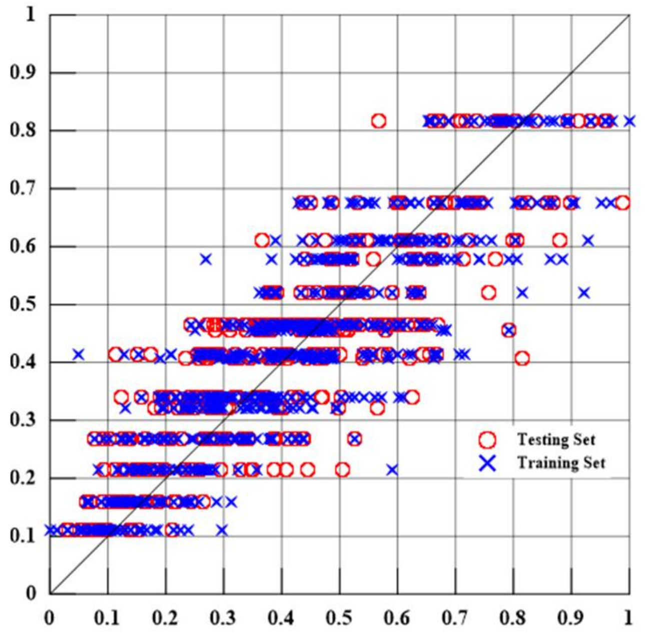
- **Decision Tree (DT)**



Fine DT (4)



Medium DT (12)



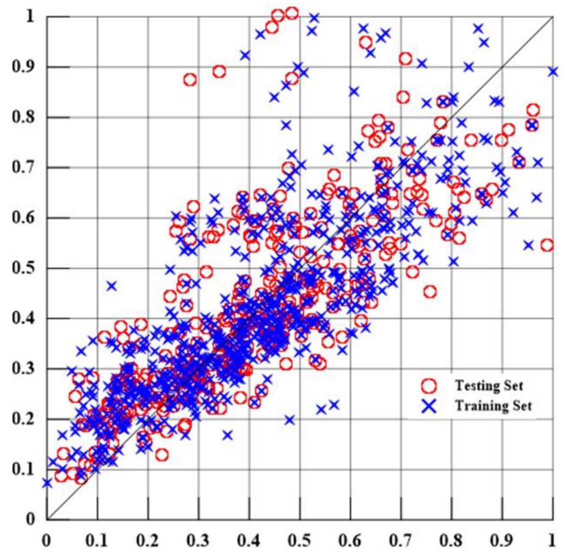
Coarse DT (36)

Concrete Compressive Strength Prediction Using ML

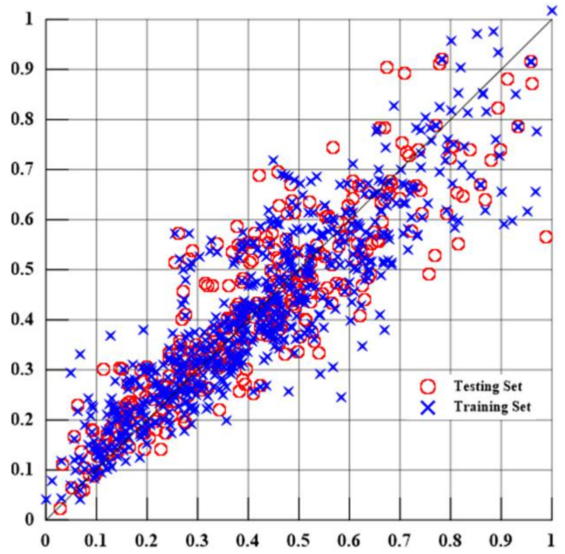
Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Decision Tree (DT)	Fine DT	0.0539	0.0894	0.9329	0.8147	0.0381	0.0640
	Medium DT	0.0802	0.0933	0.8515	0.7983	0.0598	0.0729
	Coarse DT	0.1072	0.1160	0.7348	0.6980	0.0828	0.0925

Concrete Compressive Strength Prediction Using ML

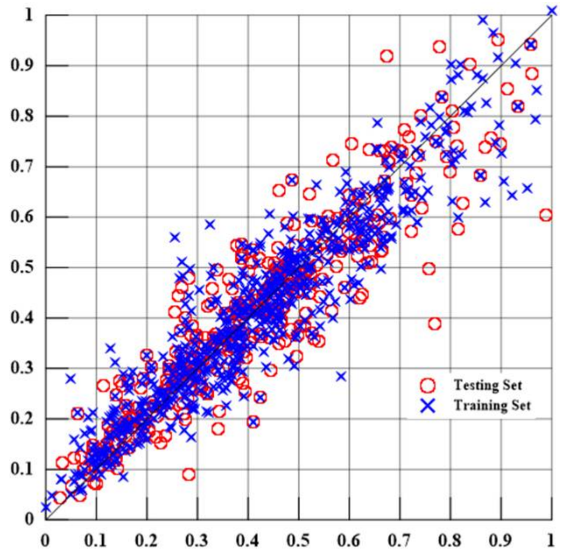
- **Support Vector Machine (SVM)**



Linear SVM



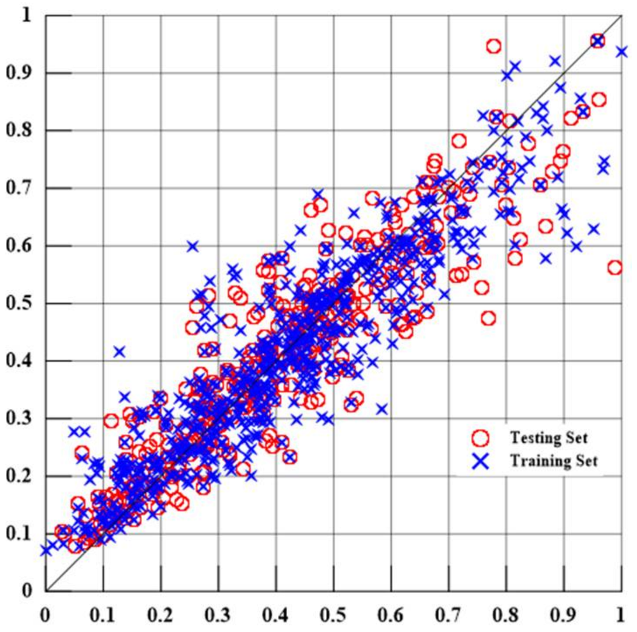
Quadratic SVM



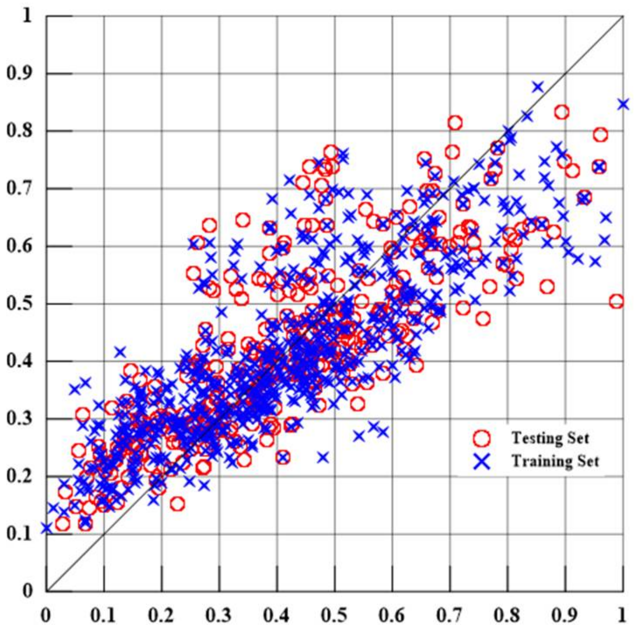
Cubic SVM

Concrete Compressive Strength Prediction Using ML

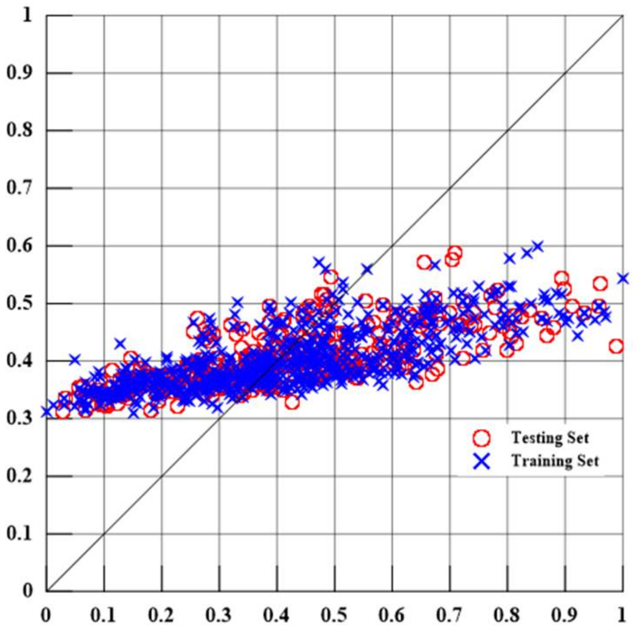
- **Support Vector Machine (SVM)**



Fine Gaussian SVM (0.71)



Medium Gaussian SVM (2.8)



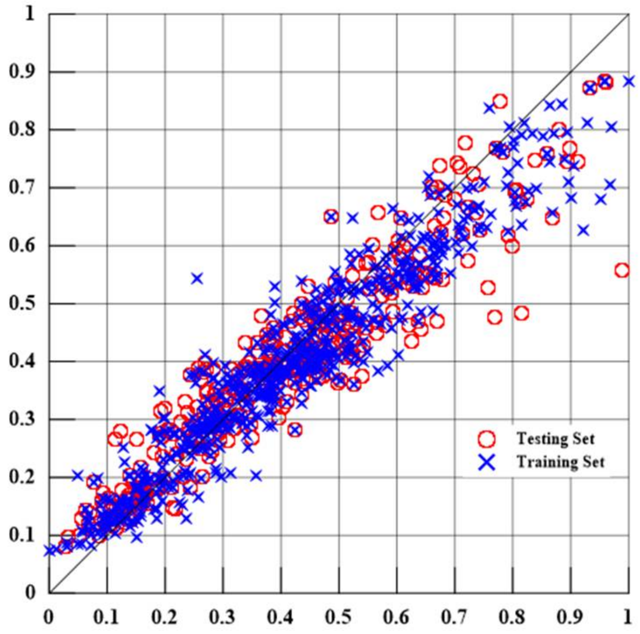
Coarse Gaussian SVM (11)

Concrete Compressive Strength Prediction Using ML

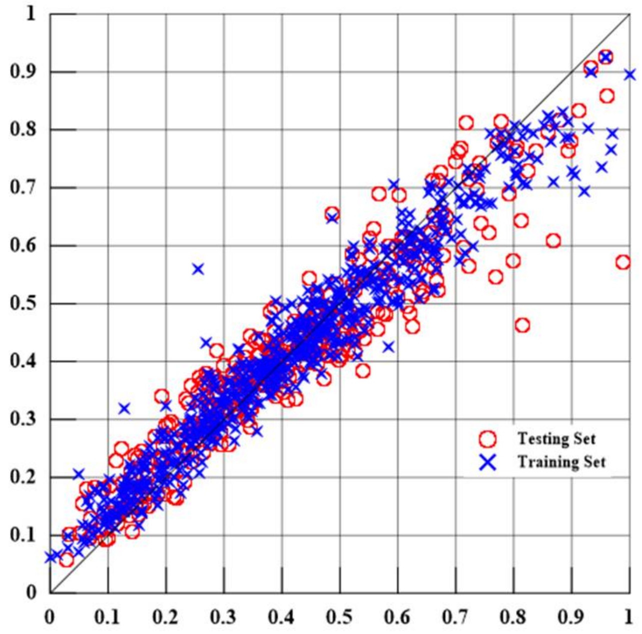
Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Support Vector Machine (SVM)	Linear SVM	0.1341	0.1474	0.5850	0.4964	0.0985	0.1048
	Quadratic SVM	0.0948	0.0963	0.7926	0.7847	0.0707	0.0717
	Cubic SVM	0.0738	0.0851	0.8745	0.8319	0.0534	0.0622
	Fine Gaussian SVM	0.0825	0.0882	0.8430	0.8197	0.0595	0.0652
	Medium Gaussian SVM	0.1207	0.1253	0.6637	0.6362	0.0947	0.0987
	Coarse Gaussian SVM	0.1731	0.1740	0.3085	0.2976	0.1391	0.1396

Concrete Compressive Strength Prediction Using ML

- **Ensemble (EN)**



Boosted Trees EN



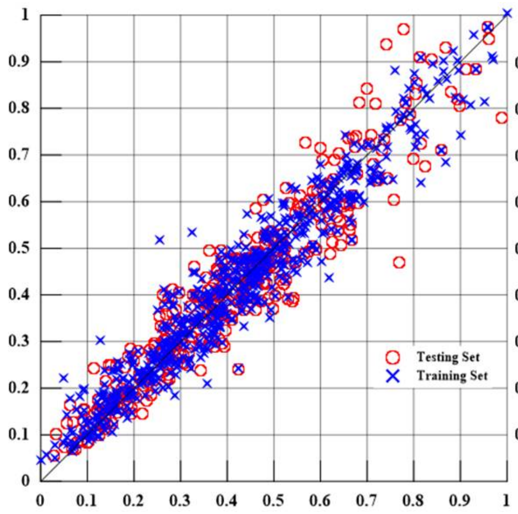
Bagged Trees EN

Concrete Compressive Strength Prediction Using ML

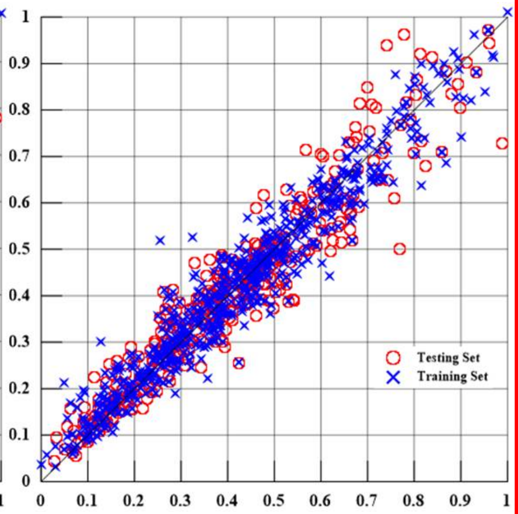
Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Ensemble (EN)	Boosted Trees EN	0.0671	0.0786	0.8963	0.8569	0.0508	0.0584
	Bagged Trees EN	0.0553	0.0717	0.9293	0.8807	0.0410	0.0536

Concrete Compressive Strength Prediction Using ML

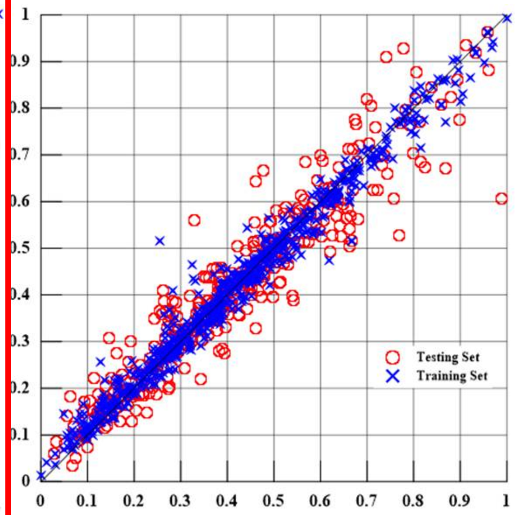
- **Gaussian Process Regression (GPR)**



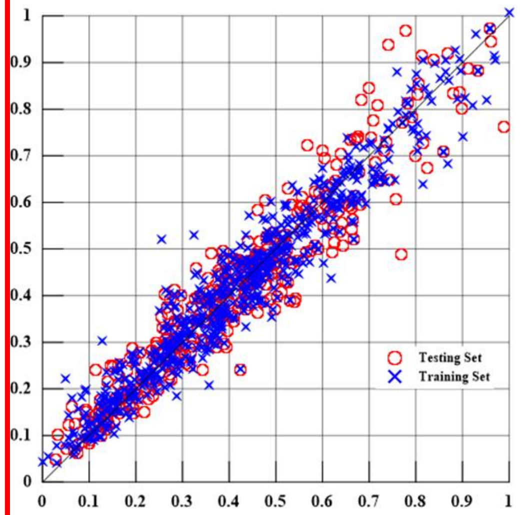
Squared Exponential GPR



Matern 5/2 GPR



Exponential GPR



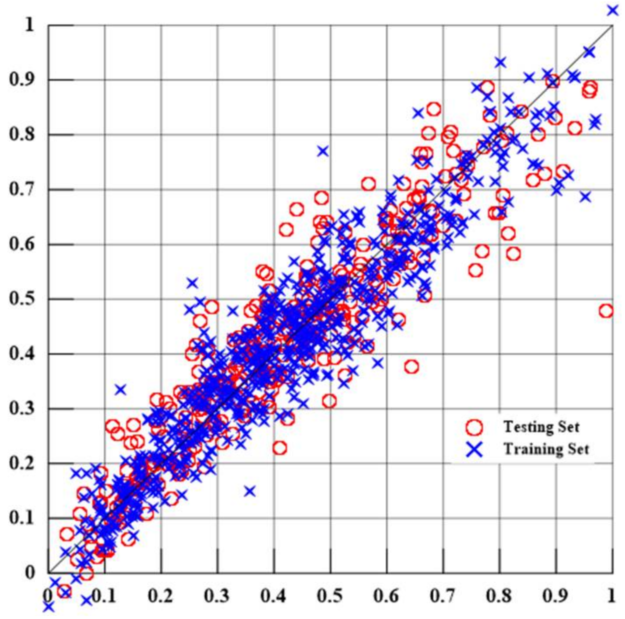
Rational Quadratic GPR

Concrete Compressive Strength Prediction Using ML

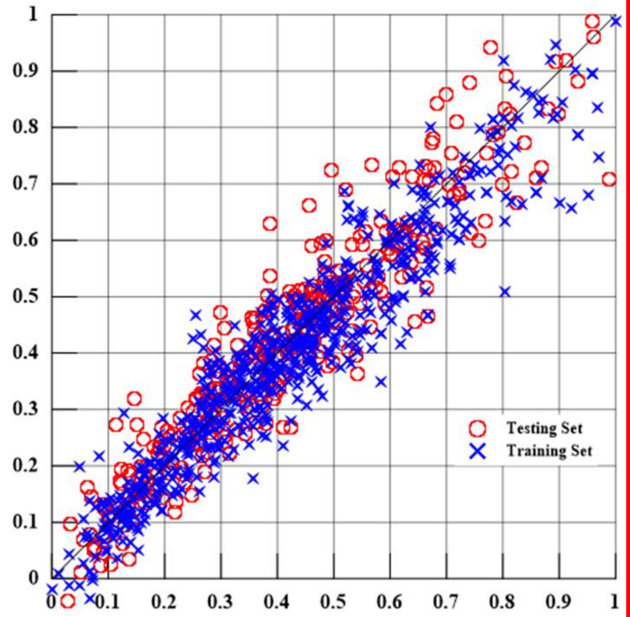
Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Gaussian Process Regression (GPR)	Squared Exponential GPR	0.0544	0.0665	0.9318	0.8976	0.0406	0.0509
	Matern 5/2 GPR	0.0517	0.0646	0.9383	0.9033	0.0382	0.0488
	Exponential GPR	0.0315	0.0661	0.9771	0.8988	0.0211	0.0473
	Rational Quadratic GPR	0.0540	0.0658	0.9326	0.8997	0.0403	0.0505

Concrete Compressive Strength Prediction Using ML

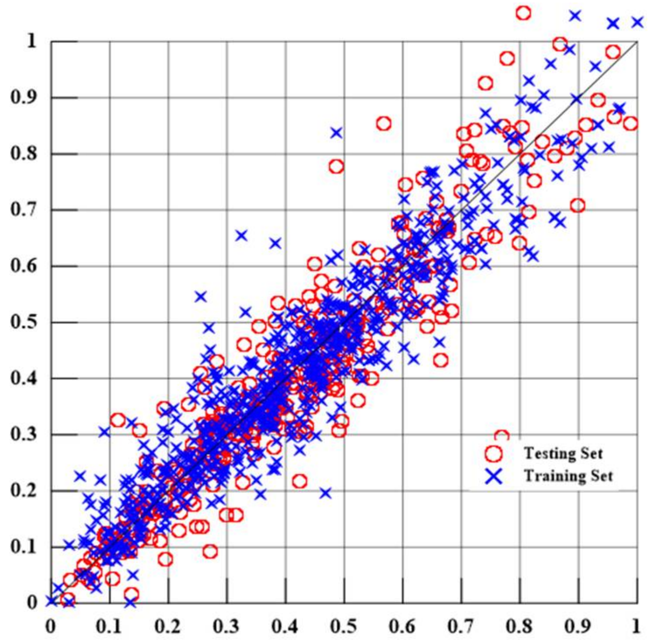
- **Neural Network (NN)**



Narrow NN (10, ReLu)



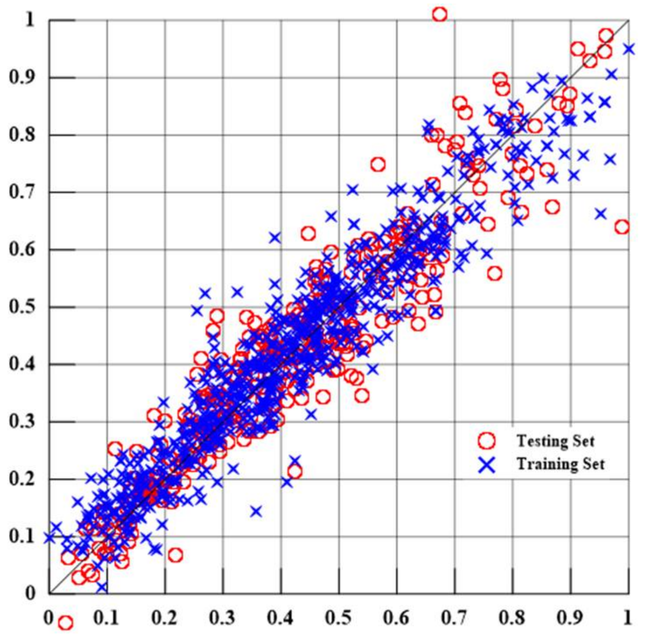
Medium NN (25, ReLu)



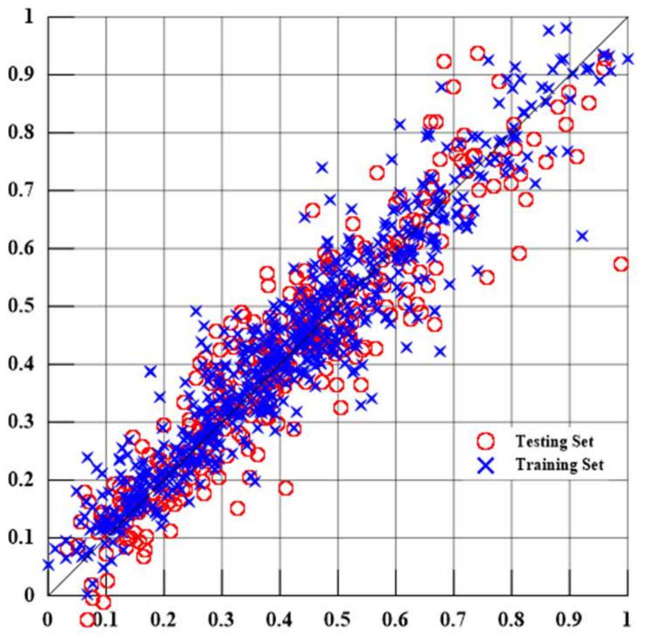
Wide NN (100, ReLu)

Concrete Compressive Strength Prediction Using ML

- **Neural Network (NN)**



Bi-layered NN ([10, 10], ReLu)



Tri-layered NN ([10, 10, 10], ReLu)

Concrete Compressive Strength Prediction Using ML

Types of Machine Learning Model		RMSE		Coefficient of Determination		MAE	
		Training	Testing	Training	Testing	Training	Testing
Neural Network (NN)	Narrow NN	0.0652	0.0874	0.9020	0.8229	0.0484	0.0663
	Medium NN	0.0610	0.0699	0.9140	0.8867	0.0435	0.0526
	Wide NN	0.0721	0.0863	0.8802	0.8274	0.0528	0.0598
	Bi-layered NN	0.0747	0.0702	0.8714	0.8858	0.0577	0.0543
	Tri-layered NN	0.0711	0.0783	0.8833	0.8579	0.0537	0.0604

Summary

- For concrete compressive strength prediction, cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age were used as input variables.

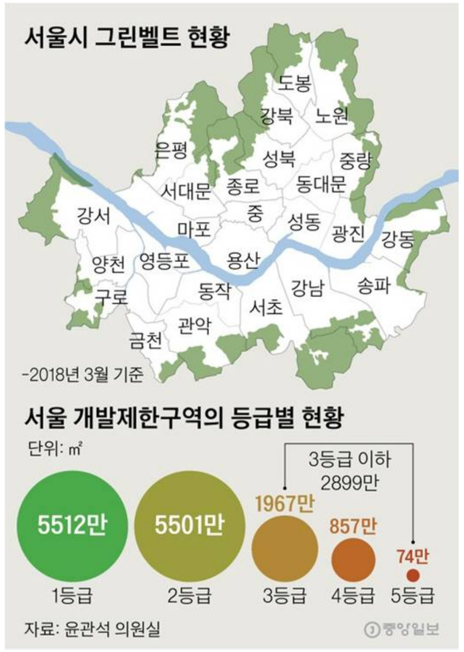
- As shown in the results, GPR (Gaussian Process Regression) and ANN (Artificial Neural Network) models outperformed other machine learning models.

- By considering more variables such as curing temperature, humidity, and detailed aggregate size information, the prediction accuracy of machine learning models can be improved in the future.

Wind Pressure Coefficients Prediction Using LSTM RNN

Wind Pressure Coefficients Prediction Using LSTM RNN

Research Background



Lack of Lands to Develop

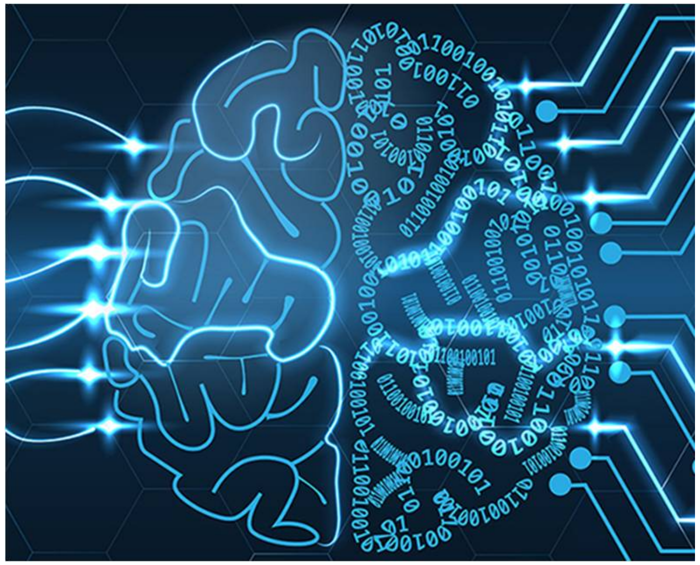
Increasing City Density

Emerging High-rise Building

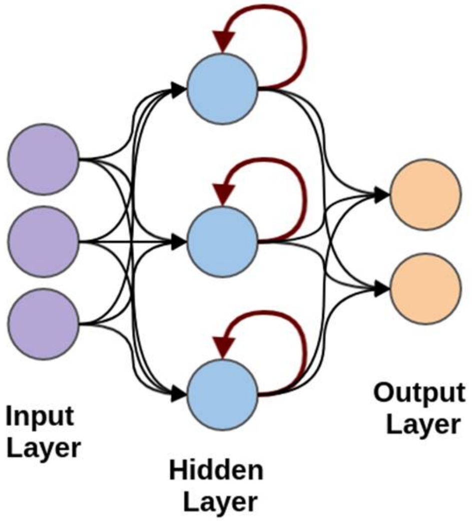
Wind Pressure Coefficients Prediction Using LSTM RNN

Problem Statement & Research Objective

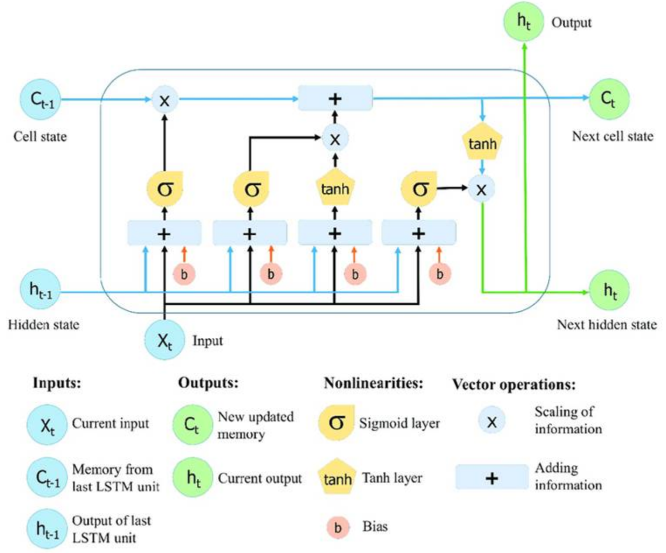
"How to Evaluate Wind Load More Smartly?"



Deep Learning



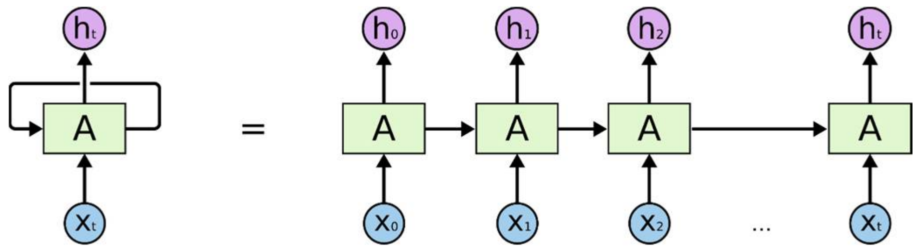
Recurrent Neural Network



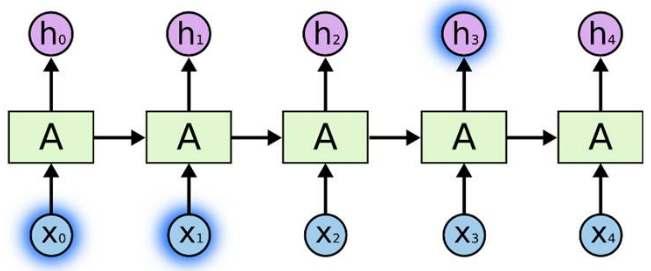
Long Short Term Memory Cell Model

Wind Pressure Coefficients Prediction Using LSTM RNN

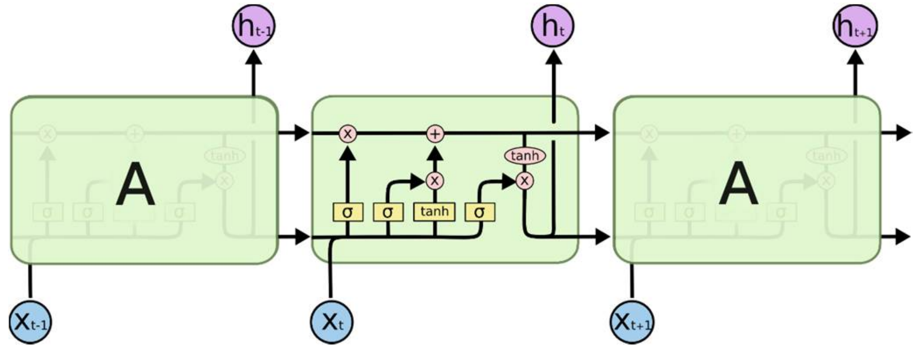
Theoretical Background



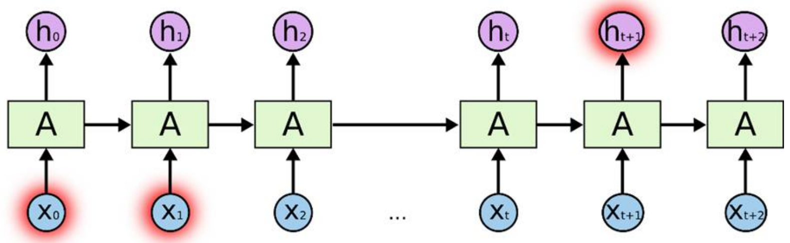
Recurrent Neural Network



Predict Output Using Previous and Current Data



LSTM Network

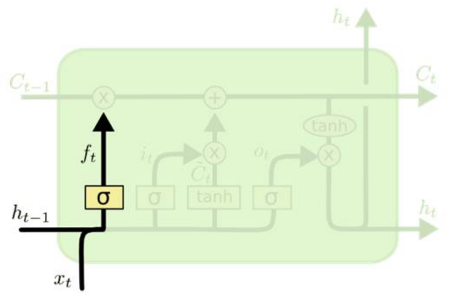


h_0 h_1 h_2 h_t h_{t+1} h_{t+2}
A A A A A A
X_0 X_1 X_2 ... X_t X_{t+1} X_{t+2}

Problem of Long Term Dependency

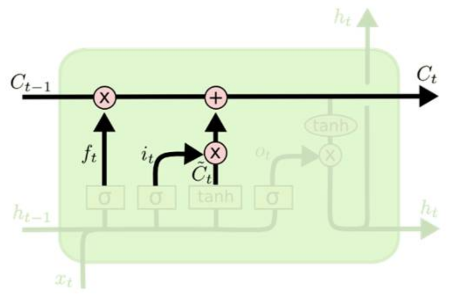
Wind Pressure Coefficients Prediction Using LSTM RNN

Theoretical Background



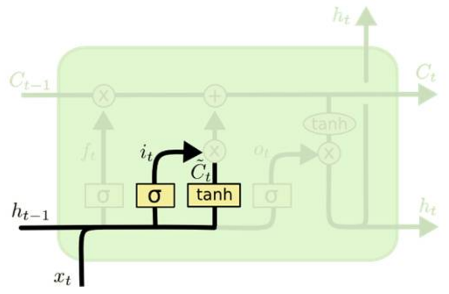
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget Gate



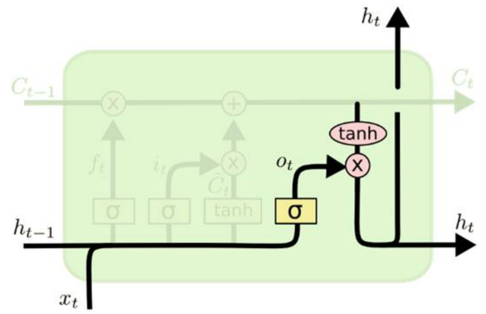
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Update Gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Input Gate

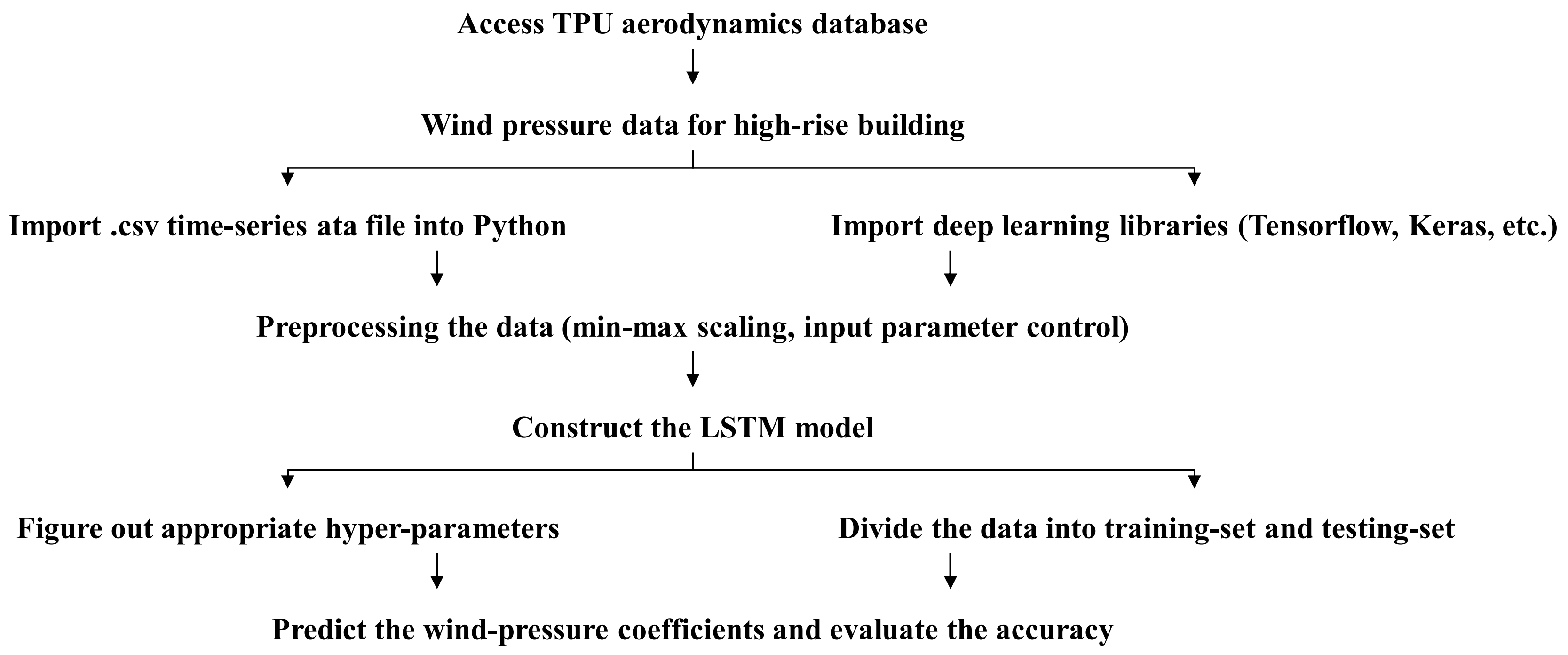


$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Output Gate

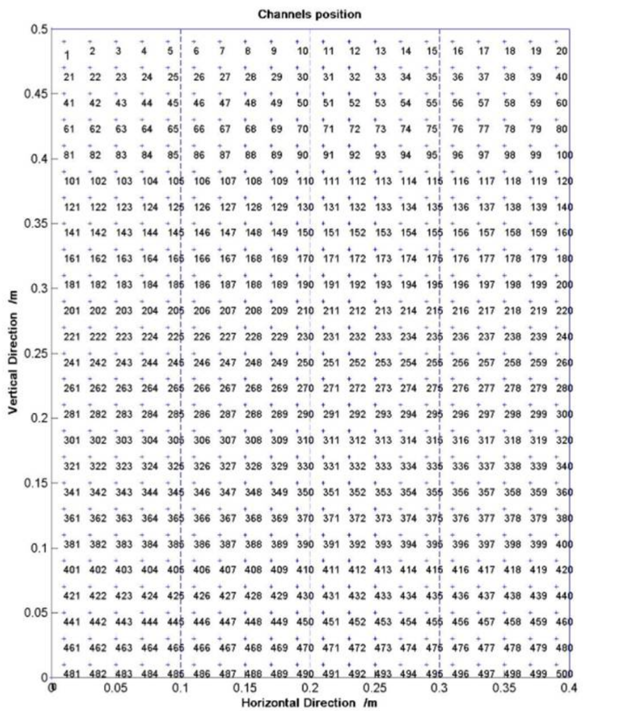
Wind Pressure Coefficients Prediction Using LSTM RNN

Research Framework


Access TPU aerodynamics database
Wind pressure data for high-rise building
Import .csv time-series ata file into Python
Import deep learning libraries (Tensorflow, Keras, etc.)
Preprocessing the data (min-max scaling, input parameter control)
Construct the LSTM model
Figure out appropriate hyper-parameters
Divide the data into training-set and testing-set
Predict the wind-pressure coefficients and evaluate the accuracy

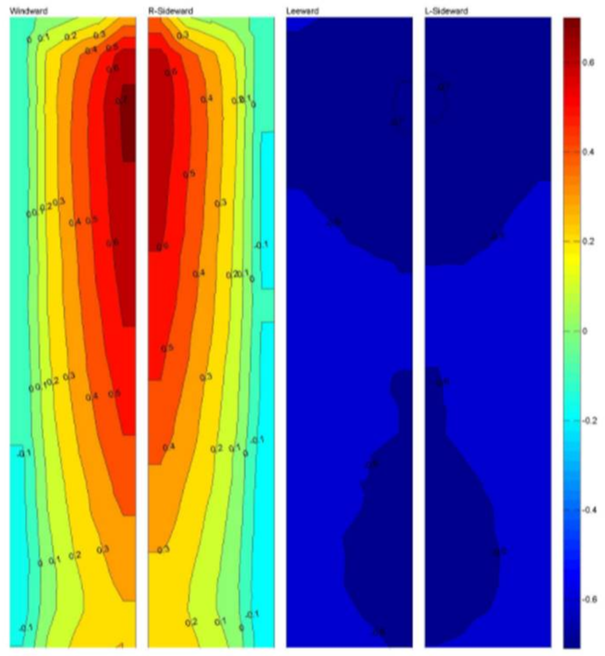
Wind Pressure Coefficients Prediction Using LSTM RNN

Data Exploration



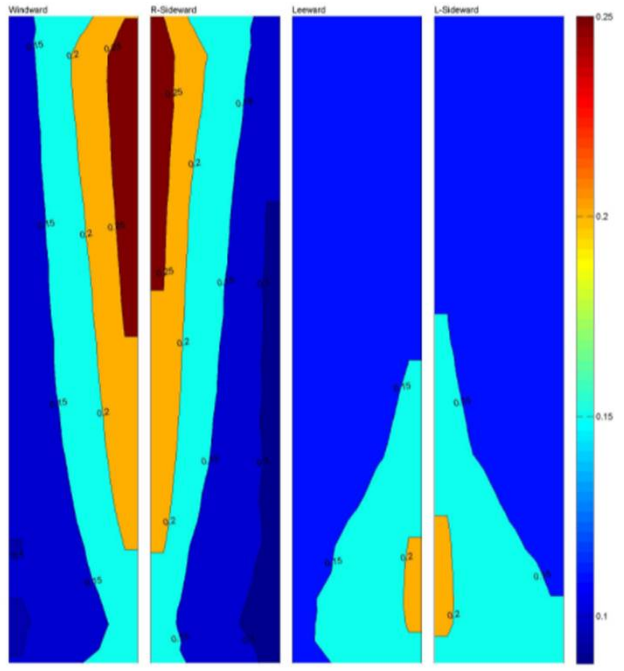
Model geometrical parameters of a high-rise building: H=0.5m, B=0.1m, D=0.1m,Model scale=1/400.

Pressure Tap Locations



Mean wind pressure coefficients on a high-rise building.
Model geometrical paramters: H=0.5m, B=0.1m, D=0.1m.
Wind field paramters: C_L = 1/400, α=0.3 ,I_H=0.49, θ=45°.

45 degree – Mean value



RMS wind pressure coefficients on a high-rise building.
Model geometrical paramters: H=0.5m, B=0.1m, D=0.1m.
Wind field paramters: C_L = 1/400, α=0.3 ,I_H=0.49, θ=45°.

45 degree – RMS value

Wind Pressure Coefficients Prediction Using LSTM RNN

Methodology

Time-Series Data

Wind Attack Angles : 11 angles

0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 degrees

Each data has 32768 points during 32.768 sec.

Training Set : 26214 points (80%)

Testing Set : 6554 points (20%)

500 pressure taps : 500 time-series x 32768 points

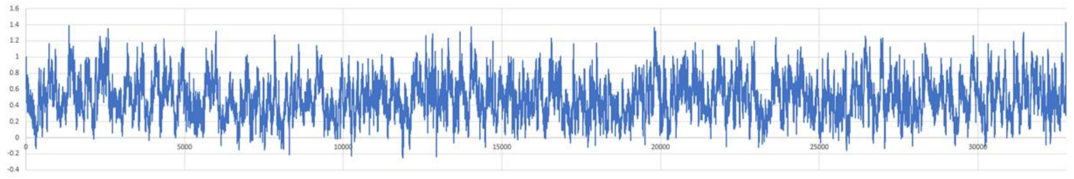
※ All data are normalized with min-max scaler

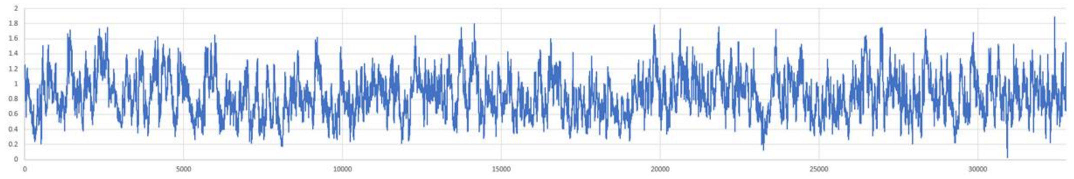
Hyper-parameters

Epoch : 100

Drop-out Ratio : 0.05

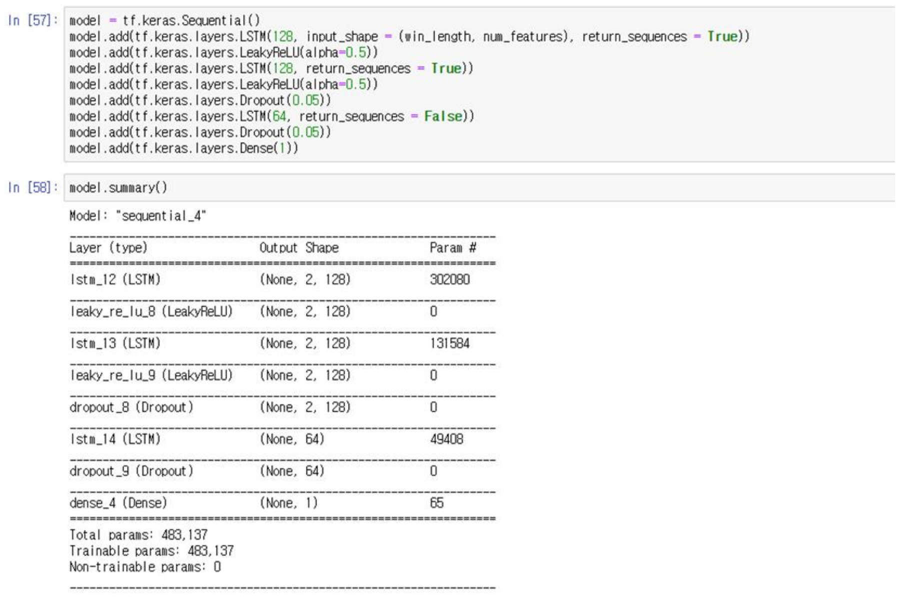
Batch Size : 10





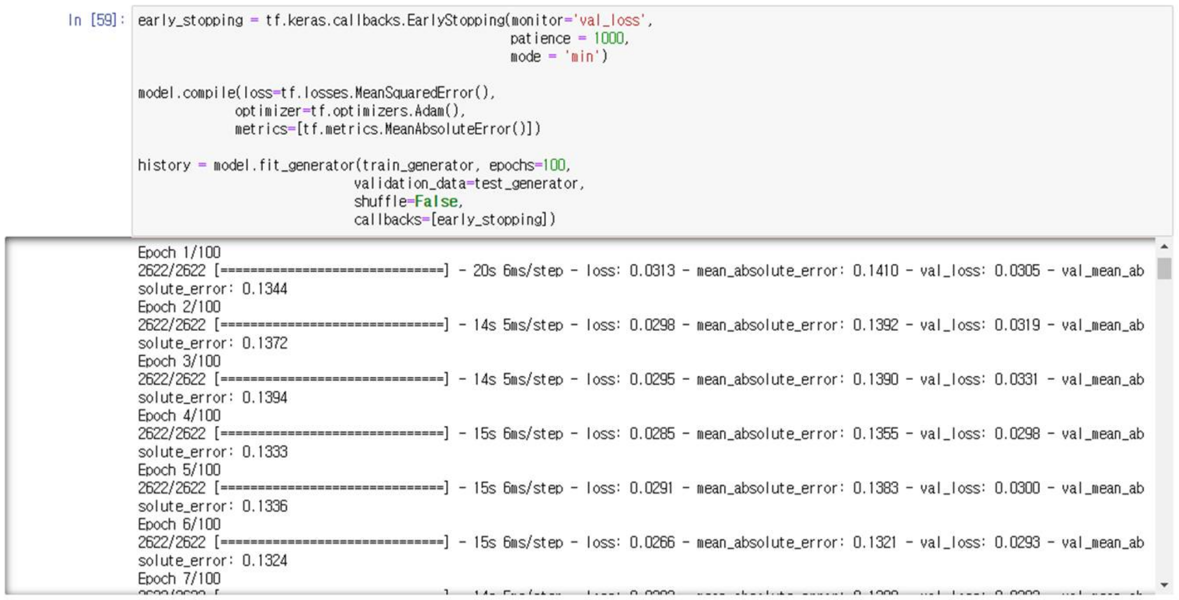
Wind Pressure Coefficients Prediction Using LSTM RNN

Model Development



```
In [57]: model = tf.keras.Sequential()
         model.add(tf.keras.layers.LSTM(128, input_shape = (win_length, num_features), return_sequences = True))
         model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
         model.add(tf.keras.layers.LSTM(128, return_sequences = True))
         model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
         model.add(tf.keras.layers.Dropout(0.05))
         model.add(tf.keras.layers.LSTM(64, return_sequences = False))
         model.add(tf.keras.layers.Dropout(0.05))
         model.add(tf.keras.layers.Dense(1))

In [58]: model.summary()

         Model: "sequential_4"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         lstm_12 (LSTM)               (None, 2, 128)            302080
         _________________________________________________________________
         leaky_re_lu_8 (LeakyReLU)    (None, 2, 128)            0
         _________________________________________________________________
         lstm_13 (LSTM)               (None, 2, 128)            131584
         _________________________________________________________________
         leaky_re_lu_9 (LeakyReLU)    (None, 2, 128)            0
         _________________________________________________________________
         dropout_8 (Dropout)          (None, 2, 128)            0
         _________________________________________________________________
         lstm_14 (LSTM)               (None, 64)                49408
         _________________________________________________________________
         dropout_9 (Dropout)          (None, 64)                0
         _________________________________________________________________
         dense_4 (Dense)              (None, 1)                 65
         =================================================================
         Total params: 483,137
         Trainable params: 483,137
         Non-trainable params: 0
         _________________________________________________________________
```

Building LSTM Model



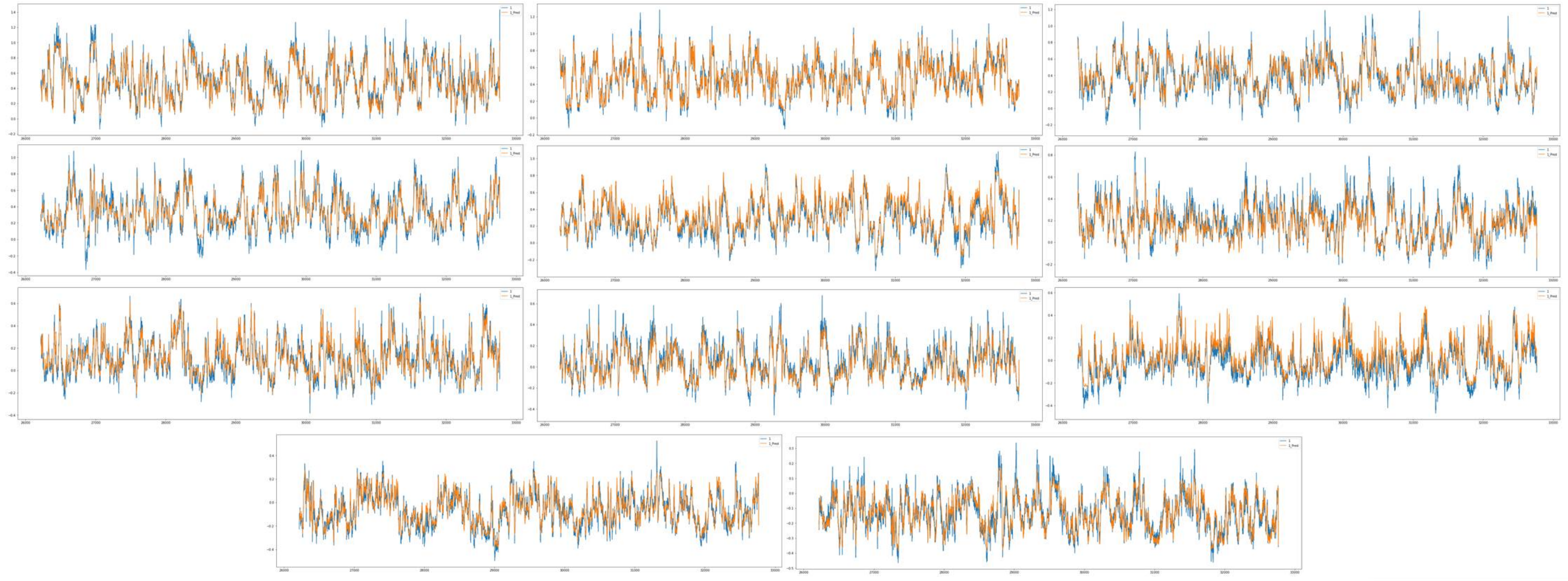
```
In [59]: early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                            patience = 1000,
                                                            mode = 'min')

         model.compile(loss=tf.losses.MeanSquaredError(),
                       optimizer=tf.optimizers.Adam(),
                       metrics=[tf.metrics.MeanAbsoluteError()])

         history = model.fit_generator(train_generator, epochs=100,
                                       validation_data=test_generator,
                                       shuffle=False,
                                       callbacks=[early_stopping])

         Epoch 1/100
         2622/2622 [==============================] - 20s 6ms/step - loss: 0.0313 - mean_absolute_error: 0.1410 - val_loss: 0.0305 - val_mean_ab
         solute_error: 0.1344
         Epoch 2/100
         2622/2622 [==============================] - 14s 5ms/step - loss: 0.0298 - mean_absolute_error: 0.1392 - val_loss: 0.0319 - val_mean_ab
         solute_error: 0.1372
         Epoch 3/100
         2622/2622 [==============================] - 14s 5ms/step - loss: 0.0295 - mean_absolute_error: 0.1390 - val_loss: 0.0331 - val_mean_ab
         solute_error: 0.1394
         Epoch 4/100
         2622/2622 [==============================] - 15s 6ms/step - loss: 0.0285 - mean_absolute_error: 0.1355 - val_loss: 0.0298 - val_mean_ab
         solute_error: 0.1333
         Epoch 5/100
         2622/2622 [==============================] - 15s 6ms/step - loss: 0.0291 - mean_absolute_error: 0.1383 - val_loss: 0.0300 - val_mean_ab
         solute_error: 0.1336
         Epoch 6/100
         2622/2622 [==============================] - 15s 6ms/step - loss: 0.0266 - mean_absolute_error: 0.1321 - val_loss: 0.0293 - val_mean_ab
         solute_error: 0.1324
         Epoch 7/100
```

Training Network

Wind Pressure Coefficients Prediction Using LSTM RNN

Results

Attack Angle	0 deg	5 deg	10 deg	15 deg	20 deg	25 deg	30 deg	35 deg	40 deg	45 deg	50 deg
RMSE	0.04993	0.03380	0.04038	0.04504	0.04224	0.04564	0.03601	0.03911	0.05655	0.03961	0.04659



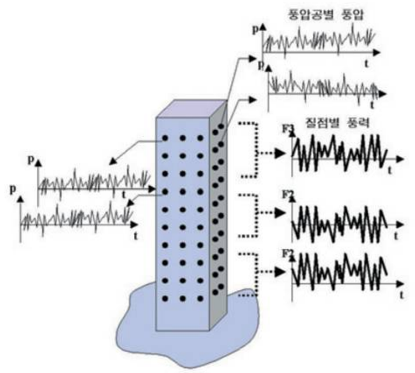
Wind Pressure Coefficients Prediction Using LSTM RNN

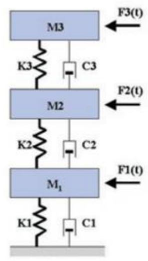
Discussion

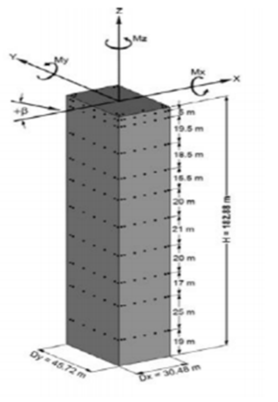
Attack Angle	0 deg	5 deg	10 deg	15 deg	20 deg	25 deg	30 deg	35 deg	40 deg	45 deg	50 deg
RMSE	0.04993	0.03380	0.04038	0.04504	0.04224	0.04564	0.03601	0.03911	0.05655	0.03961	0.04659

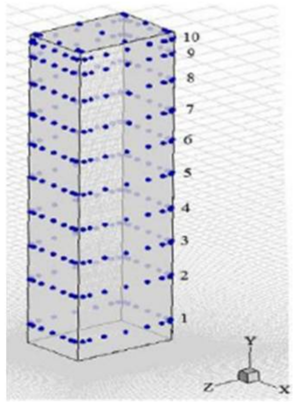
Quite accurate wind pressure coefficients prediction with all attack angles

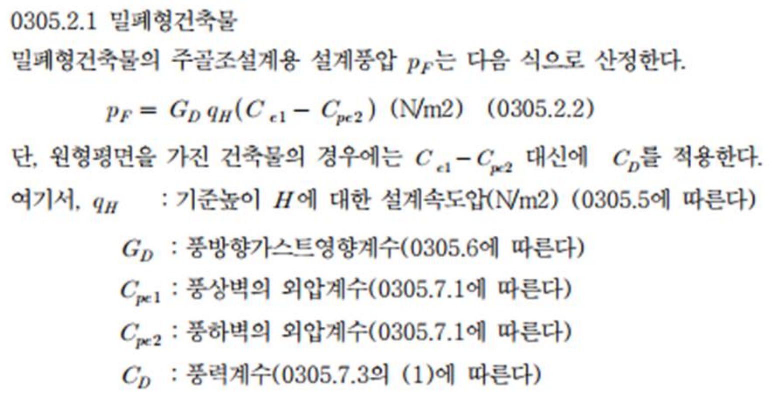
Needs more accuracy for peak estimation to use in the wind load calculation – adjustment factor may need to be analyzed separately.










0305.2.1 밀폐형건축물

밀폐형건축물의 주골조설계용 설계풍압 p_F는 다음 식으로 산정한다.

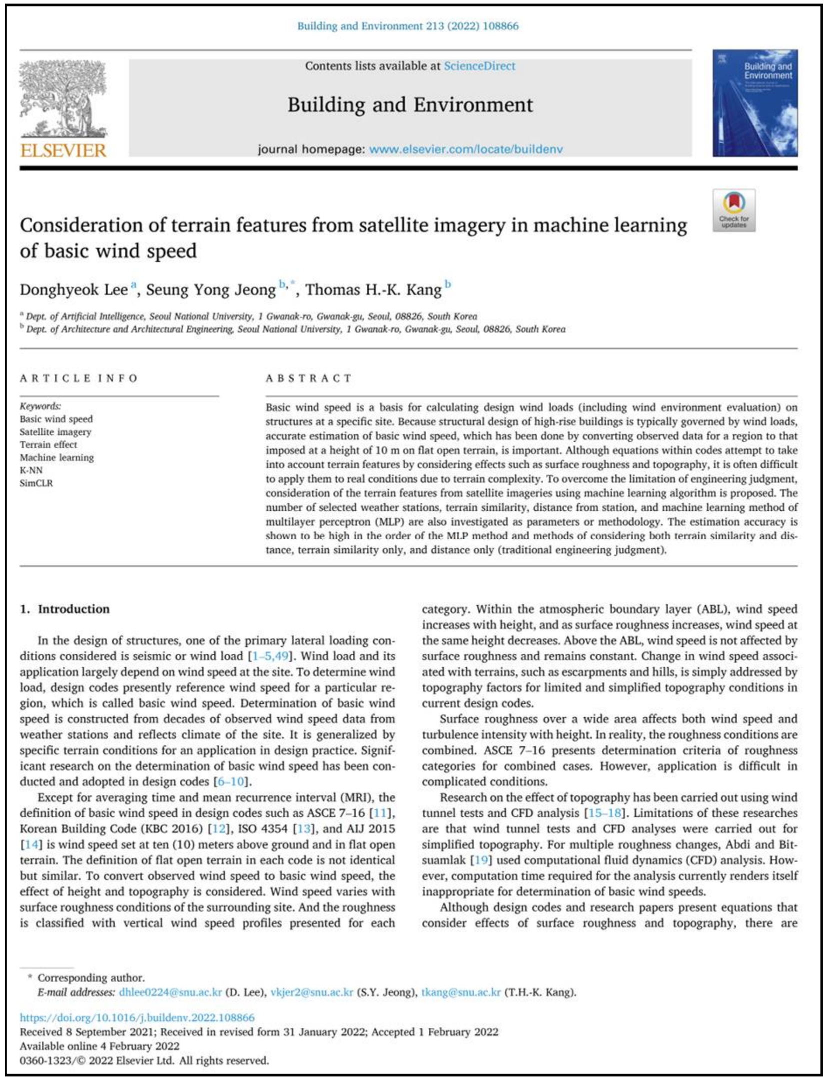
$$p_F = G_D\, q_H (C_{e1} - C_{pe2}) \text{ (N/m2)} \quad (0305.2.2)$$

단, 원형평면을 가진 건축물의 경우에는 $C_{e1} - C_{pe2}$ 대신에 C_D를 적용한다.

여기서, q_H : 기준높이 H에 대한 설계속도압(N/m2) (0305.5에 따른다)

G_D	: 풍방향가스트영향계수(0305.6에 따른다)
C_{pe1}	: 풍상벽의 외압계수(0305.7.1에 따른다)
C_{pe2}	: 풍하벽의 외압계수(0305.7.1에 따른다)
C_D	: 풍력계수(0305.7.3의 (1)에 따른다)

Determination of Basic Wind Speed Using Machine Learning Method



Building and Environment 213 (2022) 108866

Contents lists available at ScienceDirect

Building and Environment

journal homepage: www.elsevier.com/locate/buildenv

Consideration of terrain features from satellite imagery in machine learning of basic wind speed

Donghyeok Lee[a], Seung Yong Jeong[b,*], Thomas H.-K. Kang[b]

[a] Dept. of Artificial Intelligence, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 08826, South Korea
[b] Dept. of Architecture and Architectural Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 08826, South Korea

A R T I C L E I N F O

Keywords:
Basic wind speed
Satellite imagery
Terrain effect
Machine learning
K-NN
SimCLR

A B S T R A C T

Basic wind speed is a basis for calculating design wind loads (including wind environment evaluation) on structures at a specific site. Because structural design of high-rise buildings is typically governed by wind loads, accurate estimation of basic wind speed, which has been done by converting observed data for a region to that imposed at a height of 10 m on flat open terrain, is important. Although equations within codes attempt to take into account terrain features by considering effects such as surface roughness and topography, it is often difficult to apply them to real conditions due to terrain complexity. To overcome the limitation of engineering judgment, consideration of the terrain features from satellite imageries using machine learning algorithm is proposed. The number of selected weather stations, terrain similarity, distance from station, and machine learning method of multilayer perceptron (MLP) are also investigated as parameters or methodology. The estimation accuracy is shown to be high in the order of the MLP method and methods of considering both terrain similarity and distance, terrain similarity only, and distance only (traditional engineering judgment).

1. Introduction

In the design of structures, one of the primary lateral loading conditions considered is seismic or wind load [1–5,49]. Wind load and its application largely depend on wind speed at the site. To determine wind load, design codes presently reference wind speed for a particular region, which is called basic wind speed. Determination of basic wind speed is constructed from decades of observed wind speed data from weather stations and reflects climate of the site. It is generalized by specific terrain conditions for an application in design practice. Significant research on the determination of basic wind speed has been conducted and adopted in design codes [6–10].

Except for averaging time and mean recurrence interval (MRI), the definition of basic wind speed in design codes such as ASCE 7–16 [11], Korean Building Code (KBC 2016) [12], ISO 4354 [13], and AIJ 2015 [14] is wind speed set at ten (10) meters above ground and in flat open terrain. The definition of flat open terrain in each code is not identical but similar. To convert observed wind speed to basic wind speed, the effect of height and topography is considered. Wind speed varies with surface roughness conditions of the surrounding site. And the roughness is classified with vertical wind speed profiles presented for each category. Within the atmospheric boundary layer (ABL), wind speed increases with height, and as surface roughness increases, wind speed at the same height decreases. Above the ABL, wind speed is not affected by surface roughness and remains constant. Change in wind speed associated with terrains, such as escarpments and hills, is simply addressed by topography factors for limited and simplified topography conditions in current design codes.

Surface roughness over a wide area affects both wind speed and turbulence intensity with height. In reality, the roughness conditions are combined. ASCE 7–16 presents determination criteria of roughness categories for combined cases. However, application is difficult in complicated conditions.

Research on the effect of topography has been carried out using wind tunnel tests and CFD analysis [15–18]. Limitations of these researches are that wind tunnel tests and CFD analyses were carried out for simplified topography. For multiple roughness changes, Abdi and Bitsuamlak [19] used computational fluid dynamics (CFD) analysis. However, computation time required for the analysis currently renders itself inappropriate for determination of basic wind speeds.

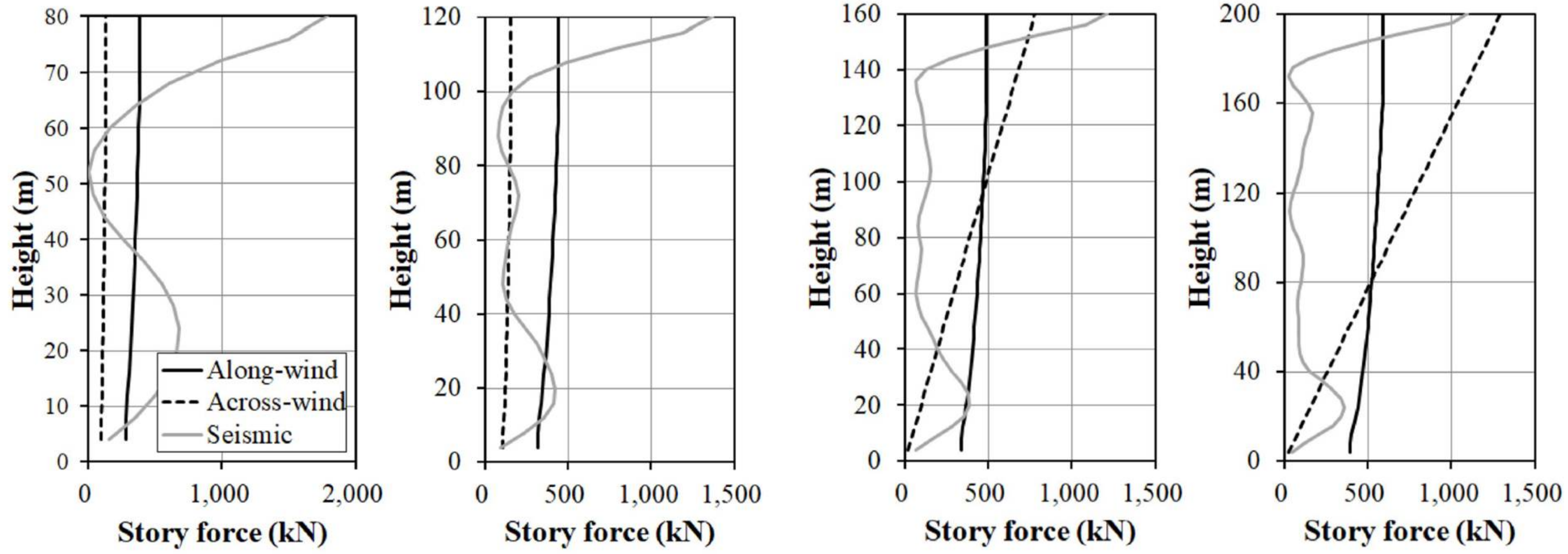
Although design codes and research papers present equations that consider effects of surface roughness and topography, there are

* Corresponding author.
E-mail addresses: dhlee0224@snu.ac.kr (D. Lee), vkjer2@snu.ac.kr (S.Y. Jeong), tkang@snu.ac.kr (T.H.-K. Kang).

https://doi.org/10.1016/j.buildenv.2022.108866
Received 8 September 2021; Received in revised form 31 January 2022; Accepted 1 February 2022
Available online 4 February 2022

Determination of Basic Wind Speed Using Machine Learning Method

- According to the existing study, in Korea, the magnitude of wind load is higher than that of seismic load in case of RC buildings with more than 30 stories.
- Wind load is determined by wind speed at construction site, and wind speed is generally provided by design codes or standards.



(a) 20-story building (b) 30-story building (c) 40-story building (d) 50-story building

Determination of Basic Wind Speed Using Machine Learning Method

$$q_H = \frac{1}{2}\rho V_H^2 \quad [\mathrm{N/m^2}]$$

$$V_H = V_0 K_{zr} K_{zt} I_w$$

ρ = Air density (1.22kg/m^2)

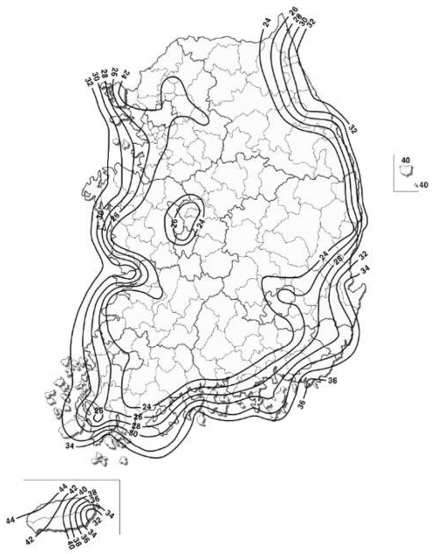
V_H = Design wind speed at height H [m/s]

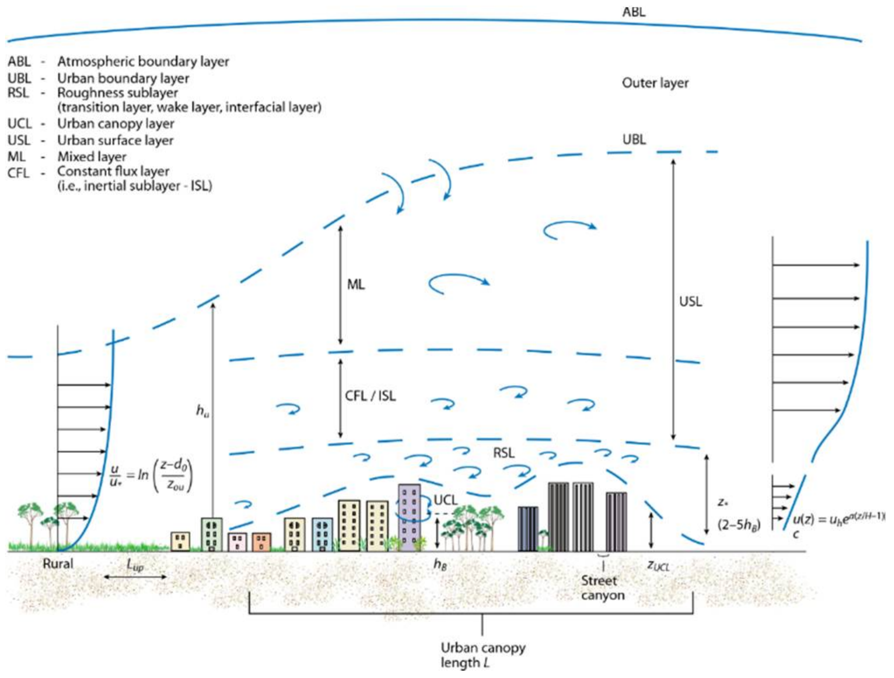
V_0 = Basic wind speed [m/s]

K_{zr} = Mean wind speed profile factor at height H

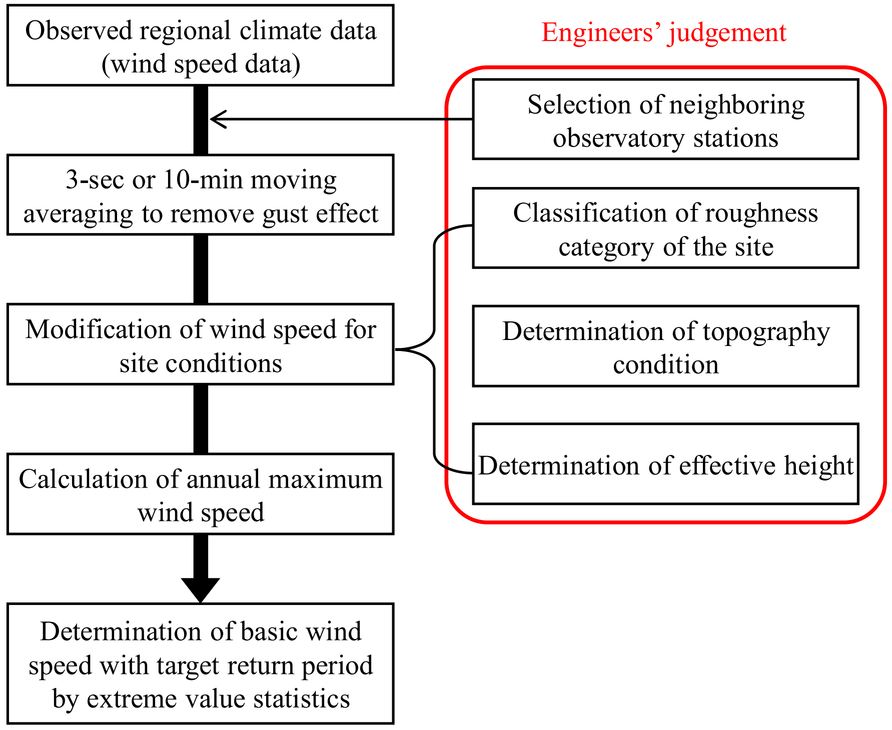
K_{zt} = Topography factor

I_w = Importance factor




ABL
ABL - Atmospheric boundary layer
UBL - Urban boundary layer
RSL - Roughness sublayer (transition layer, wake layer, interfacial layer)
UCL - Urban canopy layer
USL - Urban surface layer
ML - Mixed layer
CFL - Constant flux layer (i.e., inertial sublayer - ISL)
Outer layer
UBL
ML
USL
CFL / ISL
RSL
UCL
h_u
$\frac{u}{u_*} = ln\left(\frac{z-d_0}{z_{0u}}\right)$
z_*
$(2-5h_B)$
$u(z) = u_h e^{a(z/H-1)}$
Rural
L_{up}
h_B
Street canyon
z_{UCL}
Urban canopy length L

Determination of Basic Wind Speed Using Machine Learning Method



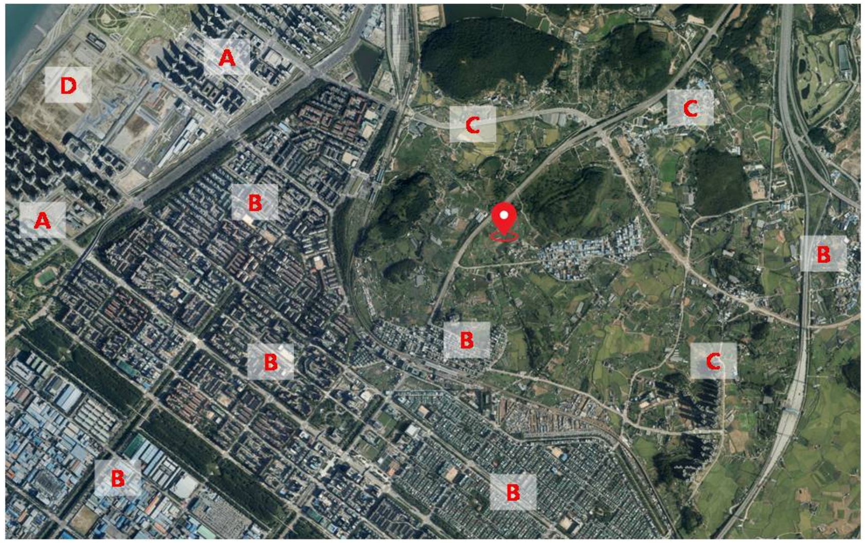
Observed regional climate data (wind speed data)
Engineers' judgement
Selection of neighboring observatory stations
3-sec or 10-min moving averaging to remove gust effect
Classification of roughness category of the site
Modification of wind speed for site conditions
Determination of topography condition
Calculation of annual maximum wind speed
Determination of effective height
Determination of basic wind speed with target return period by extreme value statistics

- KDS 41 10 15 and ASCE 7-22 present that the basic wind speed can be estimated based on observed wind speed data.
- However, the estimation of basic wind speed based on observed data requires several engineer's judgement.

  - ✓ Selection of observatory stations
  - ✓ Classification of roughness category
  - ✓ Determination of topographic condition
  - ✓ Determination of effective height

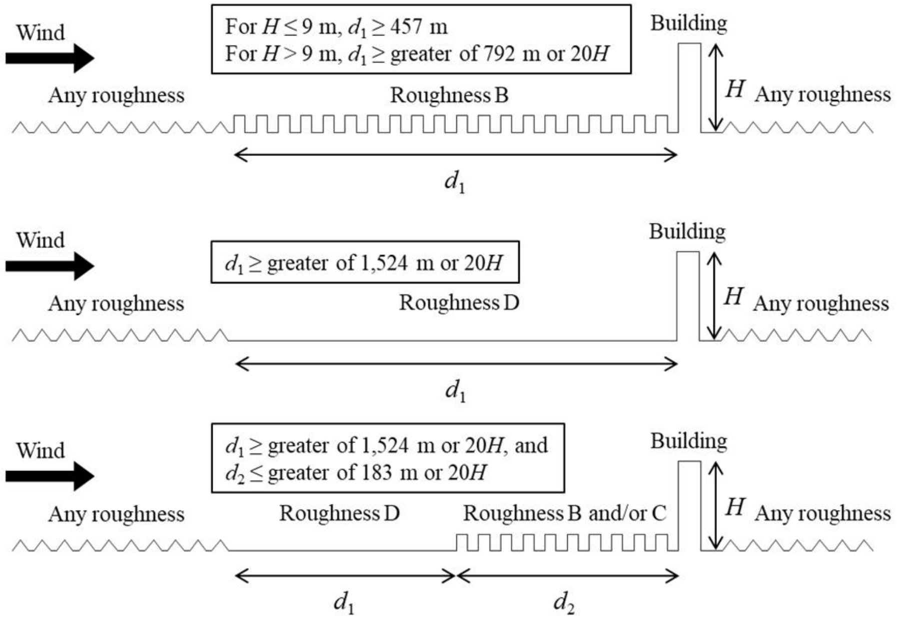
- In case of determination of ground roughness category, it may be inaccurate since it relies on the engineer's judgment based on satellite image or field observation

Determination of Basic Wind Speed Using Machine Learning Method



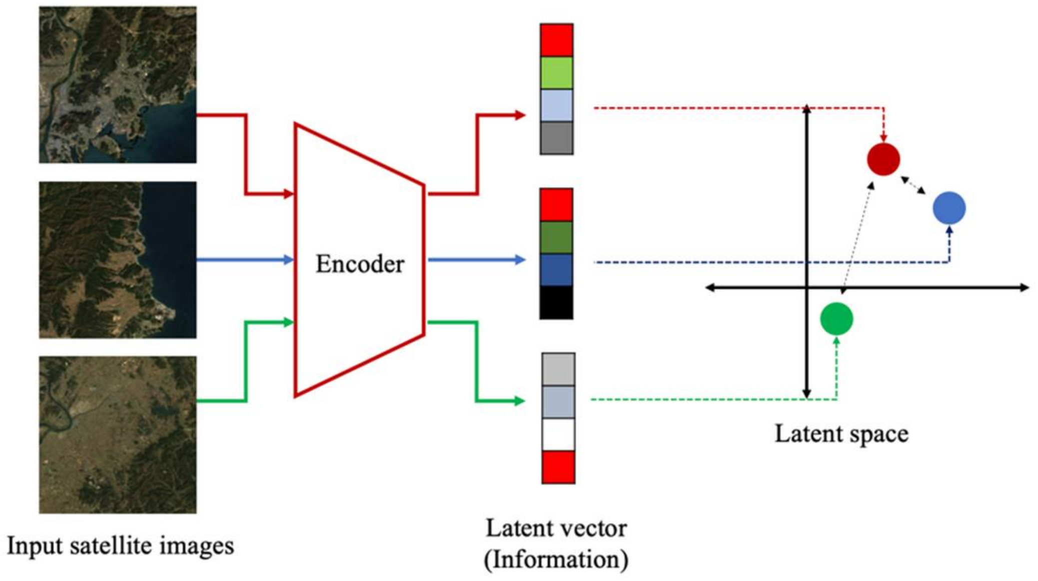
- Depending on the wind direction, surface roughness category can be mixed and it is hard to make an unarguable decision.
- Since quantitative standard for surface roughness category does not exist, it is difficult to judge accurately.

Surface Roughness	Description
A	Large city center with closely spaced tall buildings higher than 10-story
B	City with closely spaced residential houses with heights of 3.5 m or so or sc attered medium-rise buildings
C	Open terrain with scattered obstructions with heights of 1.5～10 m or so or scattered low-rise buildings
D	Exposed open terrain with few obstructions or scattered obstructions less th an 1.5 m in height or grassland, beach, airport etc.



Wind
For $H \leq 9$ m, $d_1 \geq 457$ m
For $H > 9$ m, $d_1 \geq$ greater of 792 m or $20H$
Building
Any roughness
Roughness B
H
Any roughness
d_1
Wind
$d_1 \geq$ greater of 1,524 m or $20H$
Building
Any roughness
Roughness D
H
Any roughness
d_1
Wind
$d_1 \geq$ greater of 1,524 m or $20H$, and
$d_2 \leq$ greater of 183 m or $20H$
Building
Any roughness
Roughness D
Roughness B and/or C
H
Any roughness
d_1
d_2

Determination of Basic Wind Speed Using Machine Learning Method

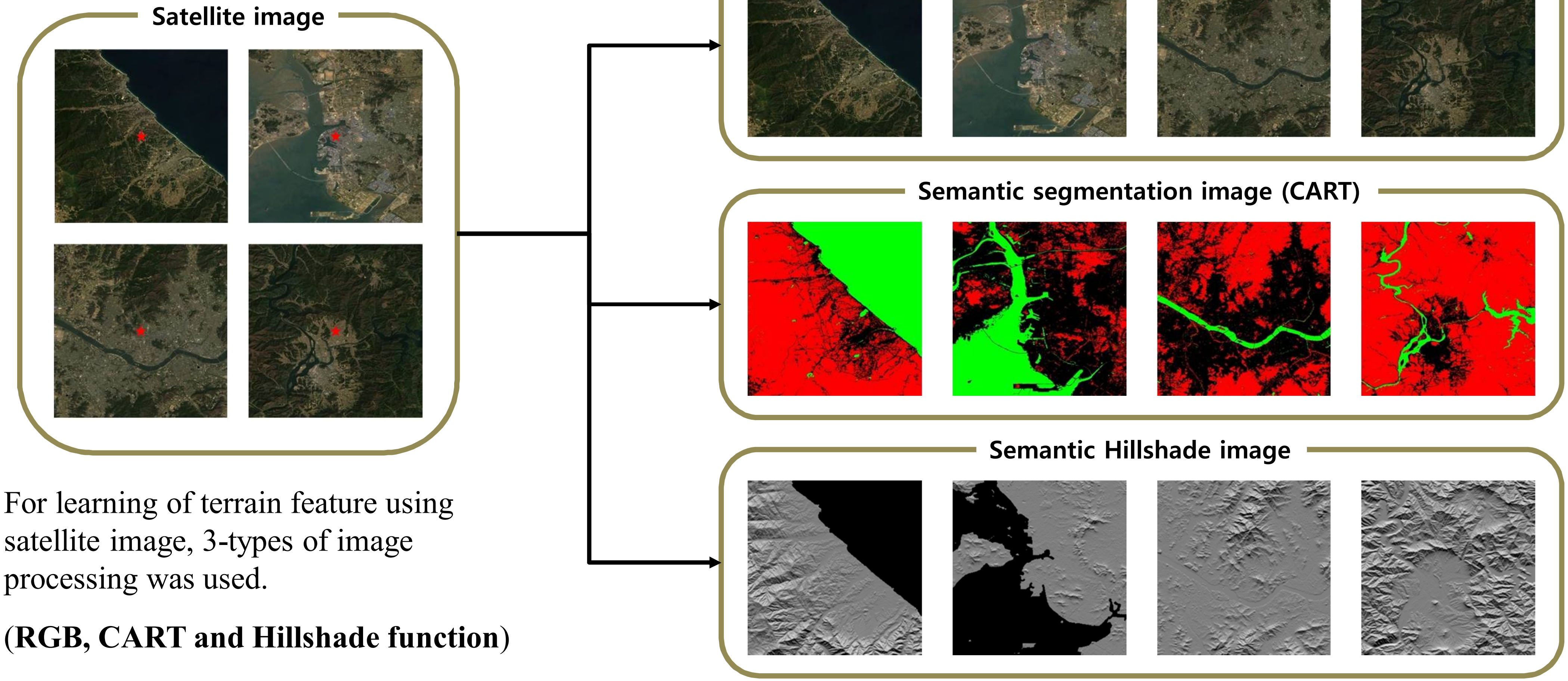


Quantitative representation of satellite images in latent space

- Machine learning has the advantage of being able to quantitatively determine high-dimensional data by decomposing high-dimensional data into low-dimensions.
- Deep learning as a method of machine learning, has advantages that it can handle high-dimension data, such as image.
- Since the topographical features expressed in satellite images are high-dimensional and abstract, they can be easily classified and quantitatively discriminated through machine learning.

Determination of Basic Wind Speed Using Machine Learning Method

Terrain feature learning - Satellite image processing



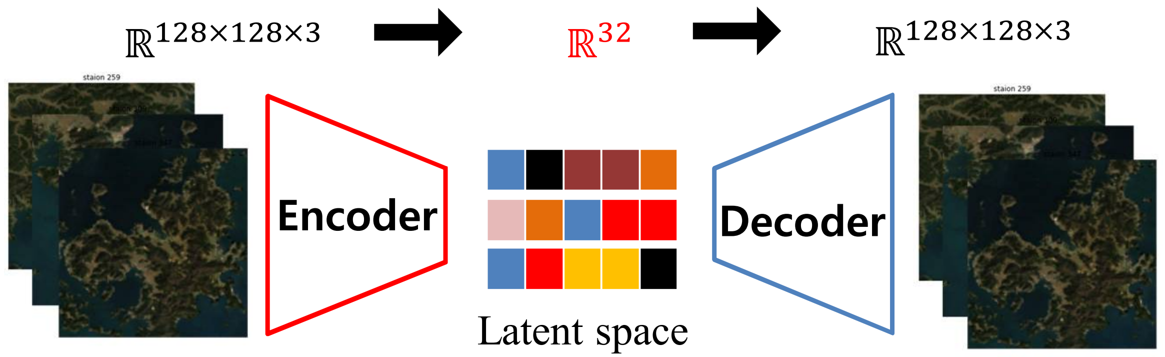
- For learning of terrain feature using satellite image, 3-types of image processing was used.

  (RGB, CART and Hillshade function)

Determination of Basic Wind Speed Using Machine Learning Method

Terrain feature learning – Learning algorithm

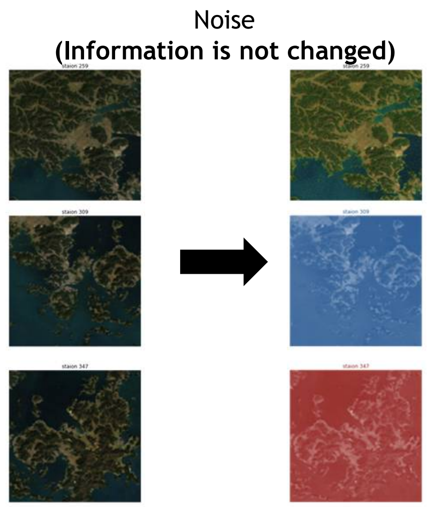
- **VAE**

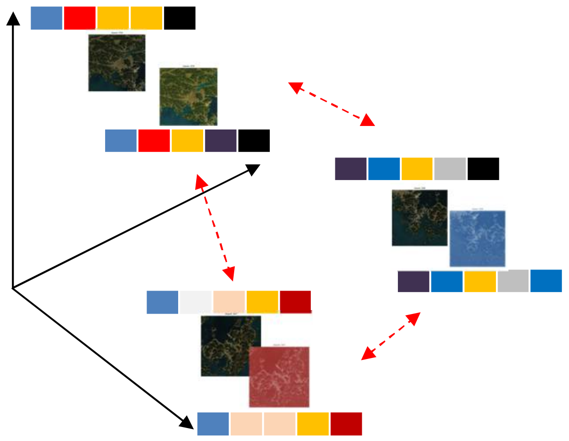


$\mathbb{R}^{128\times128\times3}$
$\mathbb{R}^{32}$
$\mathbb{R}^{128\times128\times3}$
Encoder
Latent space
Decoder

- The encoder decomposes the satellite image into latent vector, and the decoder reconstructs the vector into original satellite image.
- Through this, the encoder is trained to store representative information of satellite images in a latent space.

- **SimCLR**

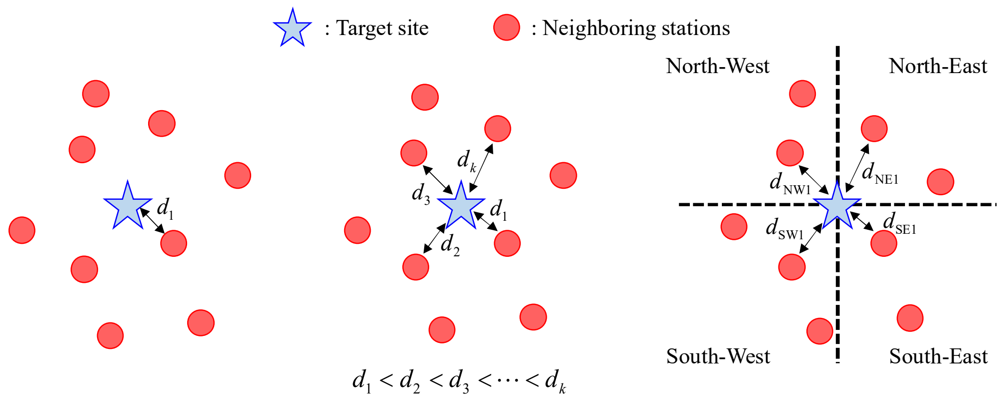




- Adding artificial noise to input satellite image and training the learning model
- The model is trained to store representative terrain information of image in latent space <u>even if it has noise.</u>

Determination of Basic Wind Speed Using Machine Learning Method

Approach 1 - Prediction of basic wind speed by machine learning



(a) Select-One method (b) K-NN method (c) Each-One method

✓ **Select-one method :**

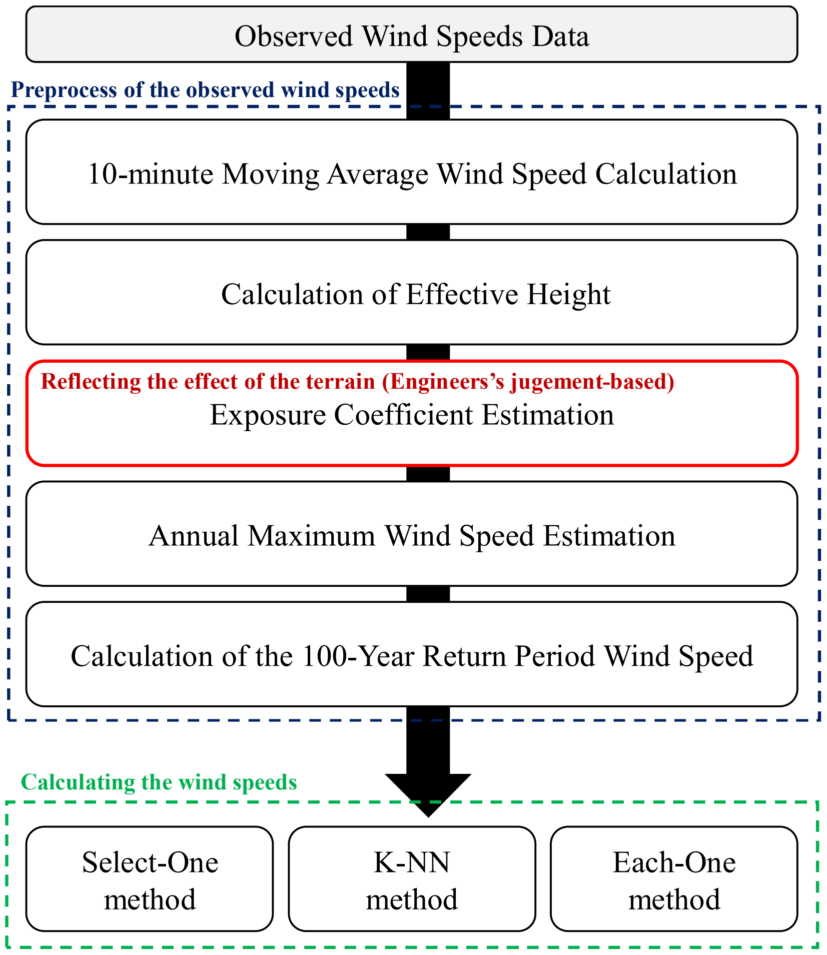
Select one nearest observatory station

✓ **K-NN method :**

Select K number of nearest observatory station

✓ **Each-one method:**

Select one observatory station at each quarter


Observed Wind Speeds Data
Preprocess of the observed wind speeds
10-minute Moving Average Wind Speed Calculation
Calculation of Effective Height
Reflecting the effect of the terrain (Engineers's jugement-based)
Exposure Coefficient Estimation
Annual Maximum Wind Speed Estimation
Calculation of the 100-Year Return Period Wind Speed
Calculating the wind speeds
Select-One method
K-NN method
Each-One method

Conventional method to estimate basic wind speed

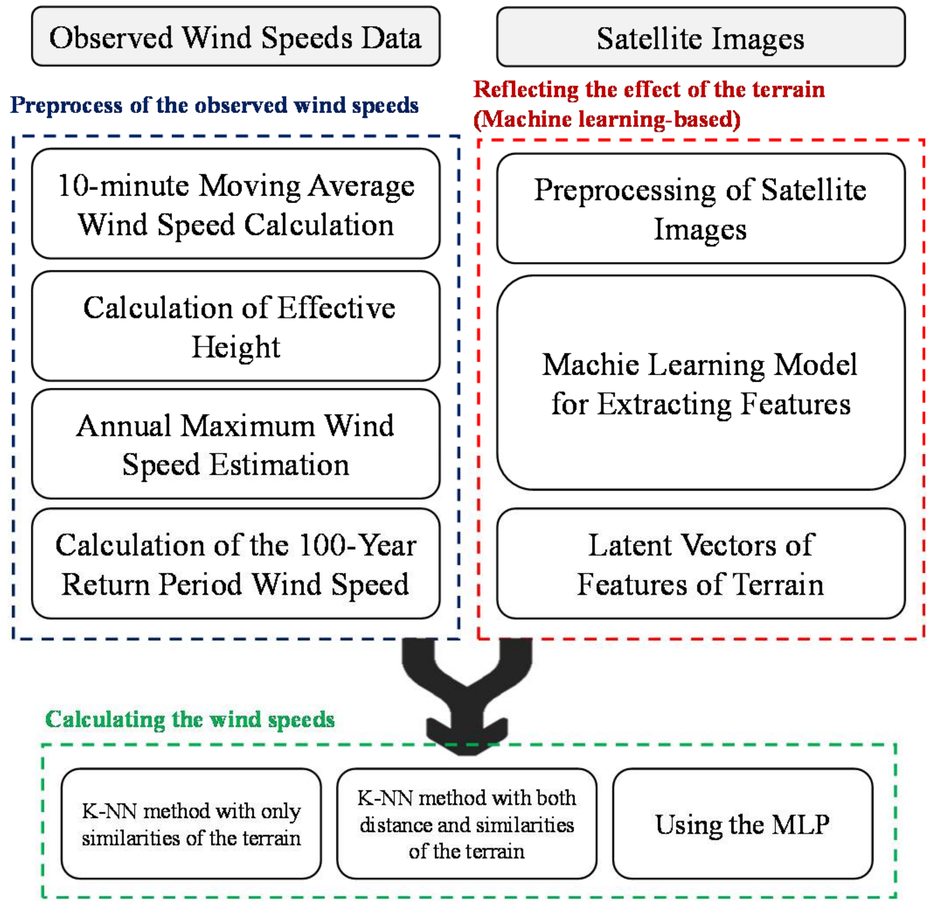
Determination of Basic Wind Speed Using Machine Learning Method

Approach 1 - Prediction of basic wind speed by machine learning

Method	All data (318 stations)		Except top 10 errors	Except top 20 errors
	Mean of errors (μ_{error}), m/s	Standard deviation of error (σ_{error}), m/s	Mean of errors (μ_{error}), m/s	Mean of errors (μ_{error}), m/s
Select-Randomly	5.364	4.350	-	-
Select-One	4.252	3.405	3.928	3.669
K-NN (k = 5)	3.466	2.712	3.187	2.987
K-NN (k = 10)	3.368	2.660	3.087	2.900
K-NN (k = 15)	3.258	2.610	2.984	2.803
Each-One	3.556	2.773	3.271	3.072

Determination of Basic Wind Speed Using Machine Learning Method

Approach 1 - Prediction of basic wind speed by machine learning



Observed Wind Speeds Data
Satellite Images
Preprocess of the observed wind speeds
Reflecting the effect of the terrain (Machine learning-based)
10-minute Moving Average Wind Speed Calculation
Calculation of Effective Height
Annual Maximum Wind Speed Estimation
Calculation of the 100-Year Return Period Wind Speed
Preprocessing of Satellite Images
Machie Learning Model for Extracting Features
Latent Vectors of Features of Terrain
Calculating the wind speeds
K-NN method with only similarities of the terrain
K-NN method with both distance and similarities of the terrain
Using the MLP

Machine learning-based method to estimate basic wind speed

- ✓ **K-NN method :**
  Select K number of nearest observatory stations.
  ⇒ Best result in baseline experiment
- ✓ **K-NN method with similarity of terrain :**
  Select K number of observatory station having only **similar terrain** feature with target site.
- ✓ **K-NN method with both distance & similarity :**
  Select K number of **nearest** observatory station having **similar terrain** feature with target site.
- ✓ **MLP (Multi-Layer Perceptron) :**
  Flexible artificial neural network
  ⇒ Trained model judges all.

Determination of Basic Wind Speed Using Machine Learning Method

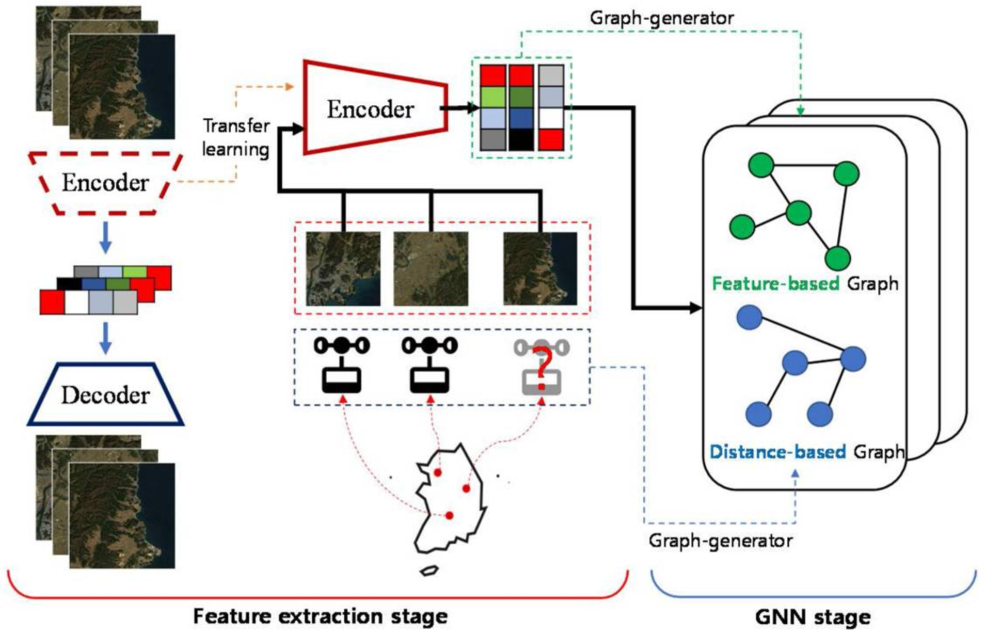
Approach 1 - Prediction of basic wind speed by machine learning

Method	All data (318 stations)		Except top 10 errors	Except top 20 errors
	Mean of errors (μ_{error}), m/s	Standard deviation of error (σ_{error}), m/s	Mean of errors (μ_{error}), m/s	Mean of errors (μ_{error}), m/s
K-NN (k = 15) (Based on distance only)	3.258	2.610	2.984	2.803
K-NN (k = 15) with machine learning (based on terrain similarity)	3.194	2.657	2.923	2.734
K-NN (k = 15) with machine learning (based on both distance and terrain similarity)	3.182	2.640	2.908	2.725
MLP (Multi-Layer Perceptron)	2.917	2.300	2.697	2.539
Each-One	3.556	2.773	3.271	3.072

- Result using artificial neural network showed the best accuracy

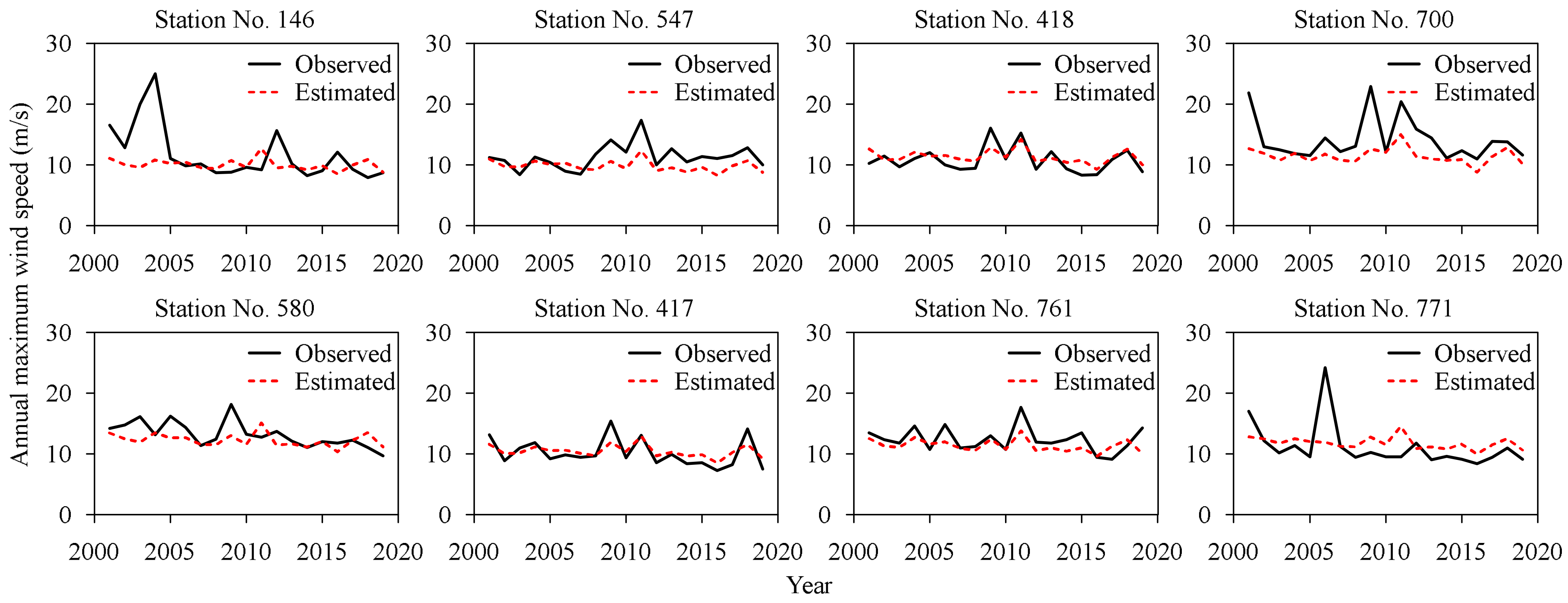
Determination of Basic Wind Speed Using Machine Learning Method

Approach 2 - Prediction of annual maximum wind speed using GNN method



- In Approach 1, prediction performance using neural networks (MLP) is higher than the performance of other methods.
- However, Approach 1 could only be deriving a single value of the basic wind speed.
- Approach 2 uses Graph Neural Network (GNN) to predict the annual maximum wind speed for multi-years based on terrain information from satellite images.
- Results were compared with actual observed annual maximum wind speed data.

Approach 2 - Prediction of annual maximum wind speed using GNN method



- Tendency of prediction and observed data was quite similar, but the accuracy of the peak value was insufficient.
- Adjustment factor that addresses the difference between the measured and estimated peak values can be developed.

Determination of Basic Wind Speed Using Machine Learning Method

Summary

- Terrain similarity can be considered through the terrain feature learned through machine learning.
- MLP model (deep learning), which predicts wind speed directly from the location information and terrain feature of the target site, showed the best accuracy.
- As a result of predicting the annual maximum wind speed using GNN (**Approach 2**), the tendency of prediction and observed data was quite similar, but the accuracy of the peak value was insufficient yet.
- The pattern of the predicted annual maximum wind speed was similar among the stations, but it was judged to be a characteristic of the wind load itself, not an overfitting problem.
- Additional research is needed on predictive models that can reflect wind direction and seasonal influences as well as terrain features.

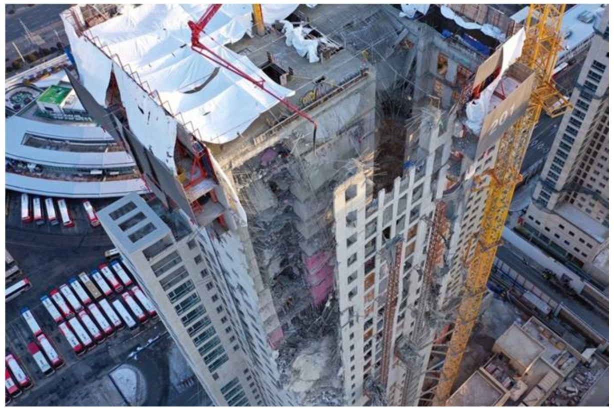
Smart NDT Using DL & Edge Computing

Collapse Disaster

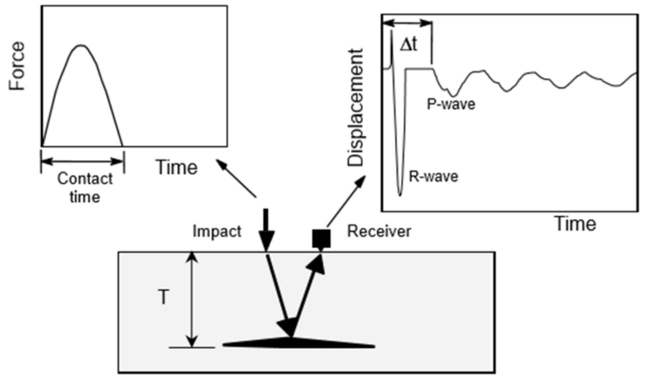
Seongsu Bridge Collapse (1994)

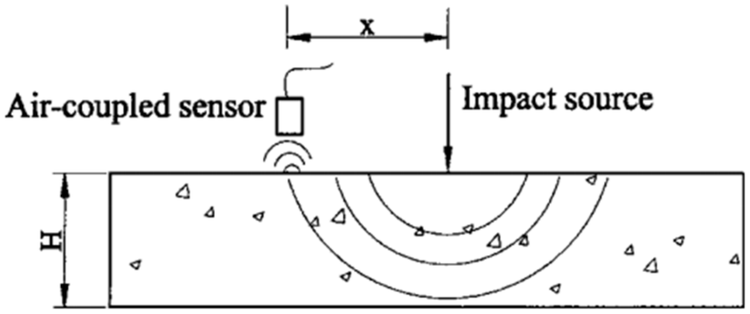
Sampoong Department Store Collapse (1995)

Gyeongju Mauna Resort Collapse (2014)

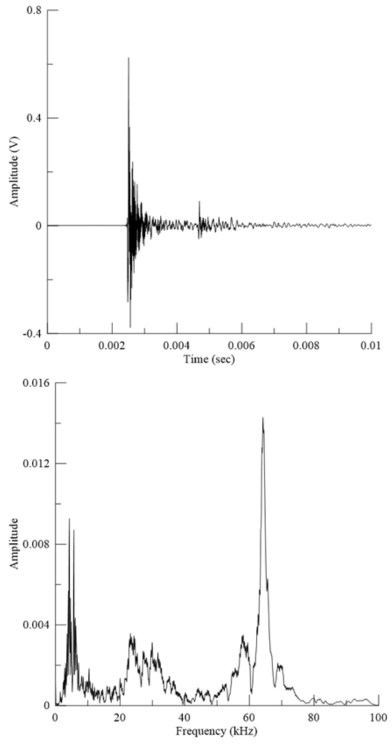

Gwang-ju Apartment Collapse (2022)

Impact Echo System

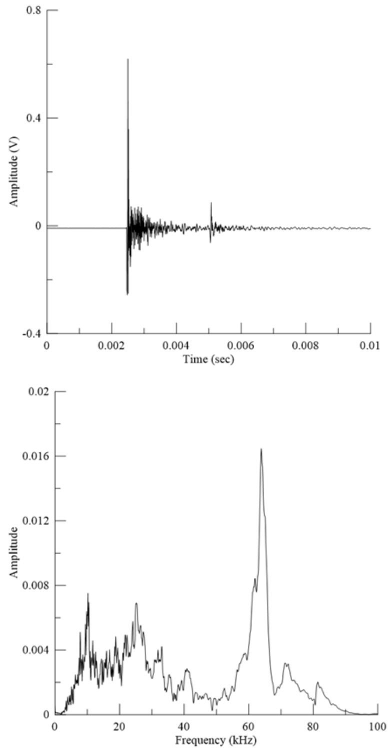




Impact Echo System



Internal Defect O

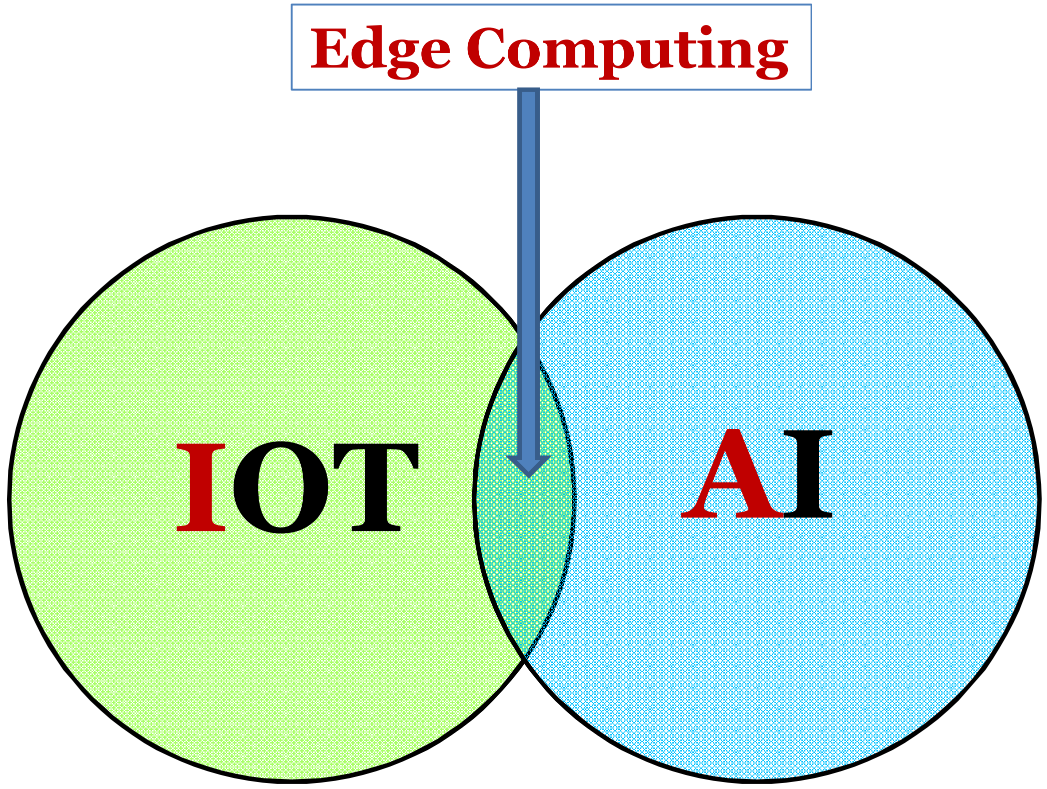


Internal Defect X

Data Processing

Why Edge Computing? Cloud Becomes Crowded.

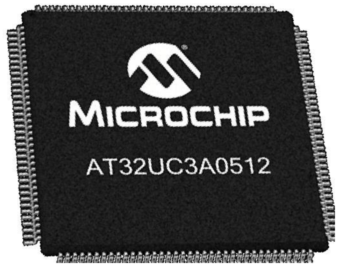
Edge Coputing



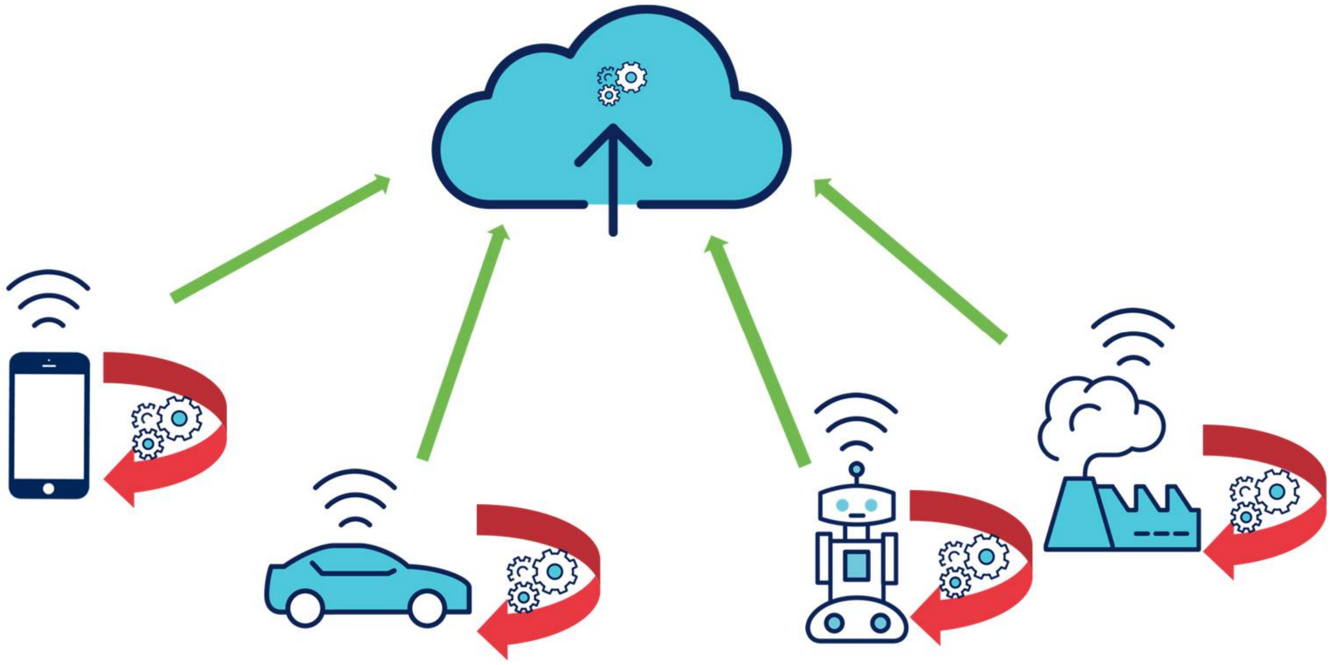
MCU

Edge Computing

CPU

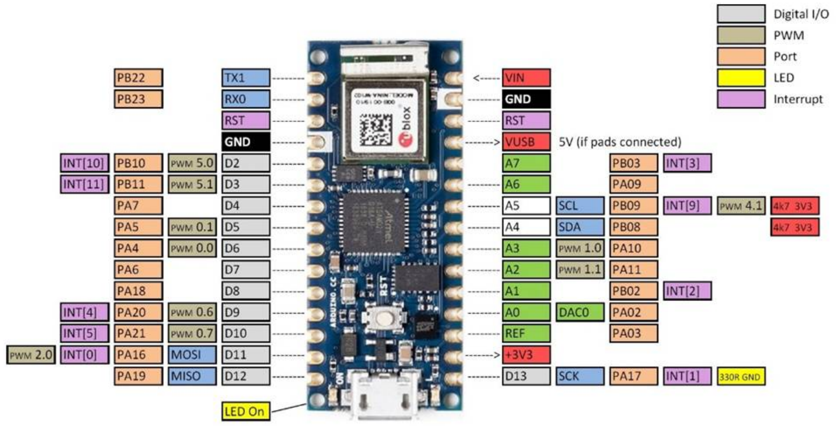


MCU

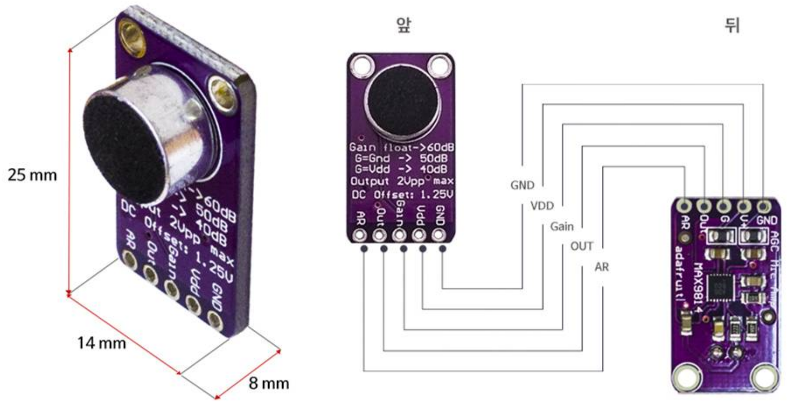


Catching right moment

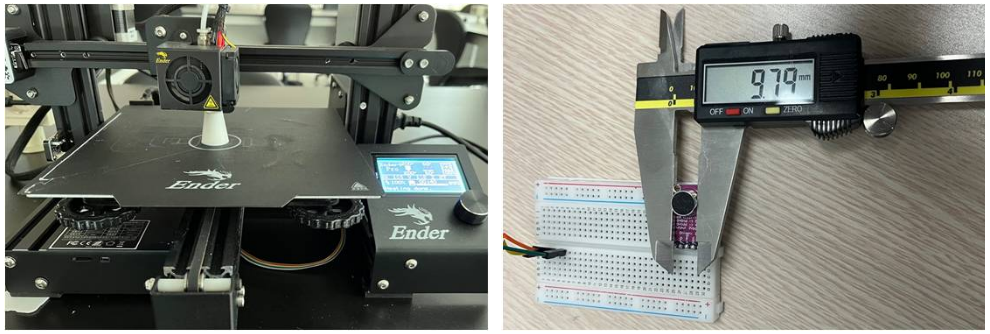
Smart NDT Using DL & Edge Computing



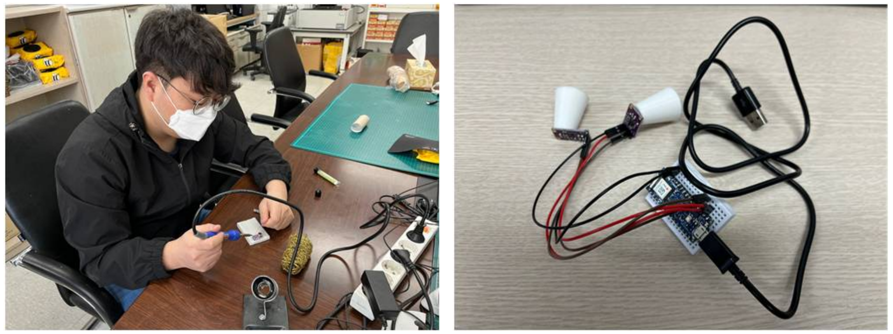
Arduino Nano 33 IoT (MCU)



MAX 9814 Microphone with Amplifier



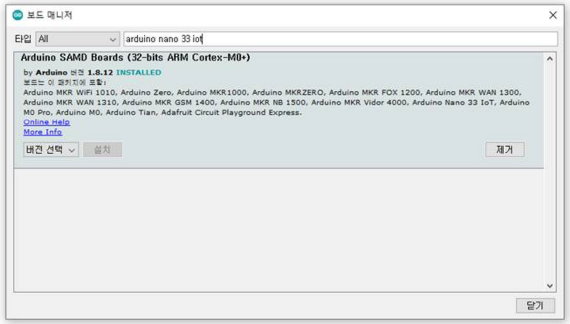
3D Printed Sound Cone



Circuit & Manufacturing (Two Sensors)

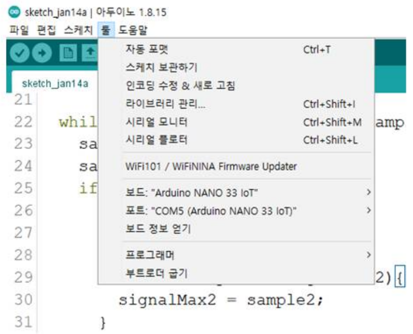
Smart NDT Using DL & Edge Computing

```
const int sampleWindow = 50; //Sample window width in microsec (50micros = 20kHz)
unsigned int sample;

void setup(){
  Serial.begin(2000000);
}

void loop(){
  unsigned long startMicros = micros(); //Start of sample window
  unsigned int peakToPeak = 0; //Peak to peak level
  unsigned int peakToPeak2 = 0;

  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;

  unsigned int signalMax2 = 0;
  unsigned int signalMin2 = 1024;

  int sample;
  int sample2;

  while(micros() - startMicros < sampleWindow){
    sample = analogRead(A0);
    sample2 = analogRead(A1);
    if (sample < 1024){
      if (sample > signalMax){
        signalMax = sample;
      }
      else if (sample2 > signalMax2){
        signalMax2 = sample2;
      }
      else if (sample < signalMin){
        signalMin = sample;
      }
      else if (sample2 < signalMin2){
        signalMin2 = sample2;
      }
    }
  }

  peakToPeak = signalMax - signalMin;
  peakToPeak2 = signalMax2 - signalMin2;

  double volts = (peakToPeak * 5.0) / 1024;
  double volts2 = (peakToPeak2 * 5.0) / 1024;

  double result = volts-volts2;

  Serial.println(result);
}
```



타입: All
arduino nano 33 iot

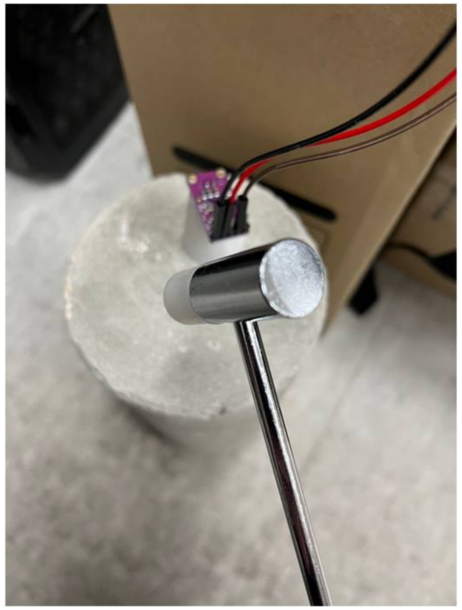
Arduino SAMD Boards (32-bits ARM Cortex-M0+)
by **Arduino** 버전 **1.8.12** INSTALLED
보드는 이 패키지에 포함:
Arduino MKR WiFi 1010, Arduino Zero, Arduino MKR1000, Arduino MKRZERO, Arduino MKR FOX 1200, Arduino MKR WAN 1300, Arduino MKR WAN 1310, Arduino MKR GSM 1400, Arduino MKR NB 1500, Arduino MKR Vidor 4000, Arduino Nano 33 IoT, Arduino M0 Pro, Arduino M0, Arduino Tian, Adafruit Circuit Playground Express.
Online Help
More Info

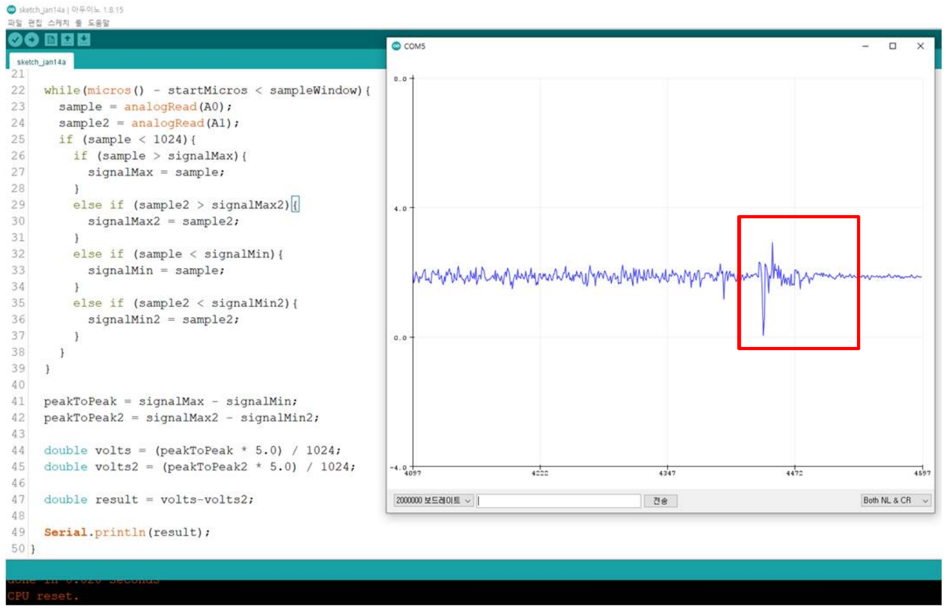

sketch_jan14a | 아두이노 1.8.15

파일 편집 스케치 툴 도움말
sketch_jan14a
자동 포맷 Ctrl+T
스케치 보관하기
인코딩 수정 & 새로 고침
라이브러리 관리... Ctrl+Shift+I
시리얼 모니터 Ctrl+Shift+M
시리얼 플로터 Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater

보드: "Arduino NANO 33 IoT"
포트: "COM5 (Arduino NANO 33 IoT)"

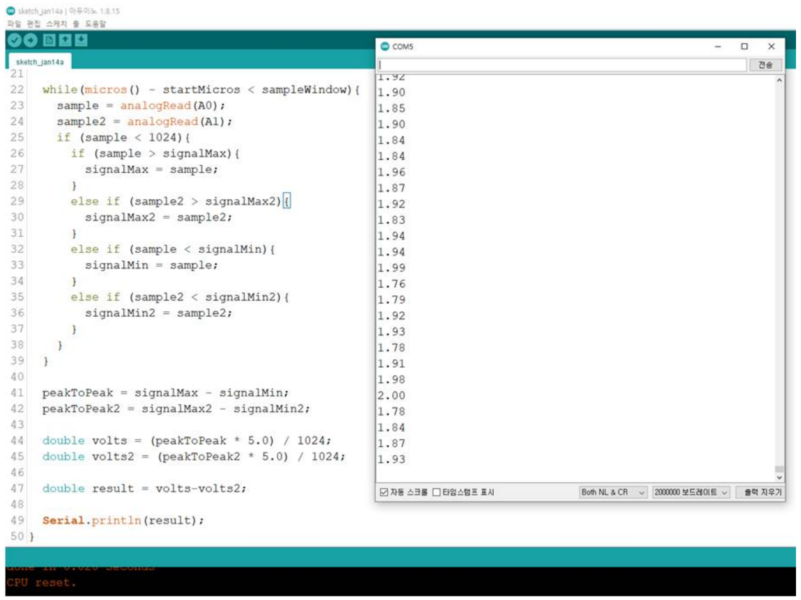
Smart NDT Using DL & Edge Computing





```
  while(micros() - startMicros < sampleWindow){
    sample = analogRead(A0);
    sample2 = analogRead(A1);
    if (sample < 1024){
      if (sample > signalMax){
        signalMax = sample;
      }
      else if (sample2 > signalMax2){
        signalMax2 = sample2;
      }
      else if (sample < signalMin){
        signalMin = sample;
      }
      else if (sample2 < signalMin2){
        signalMin2 = sample2;
      }
    }
  }

  peakToPeak = signalMax - signalMin;
  peakToPeak2 = signalMax2 - signalMin2;

  double volts = (peakToPeak * 5.0) / 1024;
  double volts2 = (peakToPeak2 * 5.0) / 1024;

  double result = volts-volts2;

  Serial.println(result);
}
```

```
CPU reset.
```



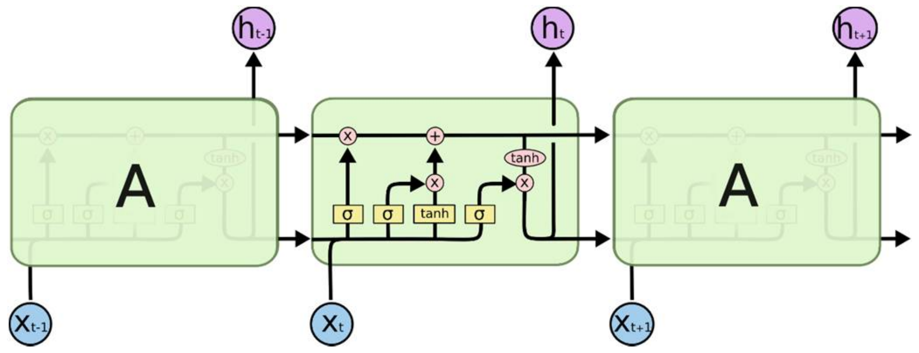
```
  while(micros() - startMicros < sampleWindow){
    sample = analogRead(A0);
    sample2 = analogRead(A1);
    if (sample < 1024){
      if (sample > signalMax){
        signalMax = sample;
      }
      else if (sample2 > signalMax2){
        signalMax2 = sample2;
      }
      else if (sample < signalMin){
        signalMin = sample;
      }
      else if (sample2 < signalMin2){
        signalMin2 = sample2;
      }
    }
  }

  peakToPeak = signalMax - signalMin;
  peakToPeak2 = signalMax2 - signalMin2;

  double volts = (peakToPeak * 5.0) / 1024;
  double volts2 = (peakToPeak2 * 5.0) / 1024;

  double result = volts-volts2;

  Serial.println(result);
}
```

```
1.90
1.85
1.90
1.84
1.84
1.96
1.87
1.92
1.83
1.94
1.94
1.99
1.76
1.79
1.92
1.93
1.78
1.91
1.98
2.00
1.78
1.84
1.87
1.93
```

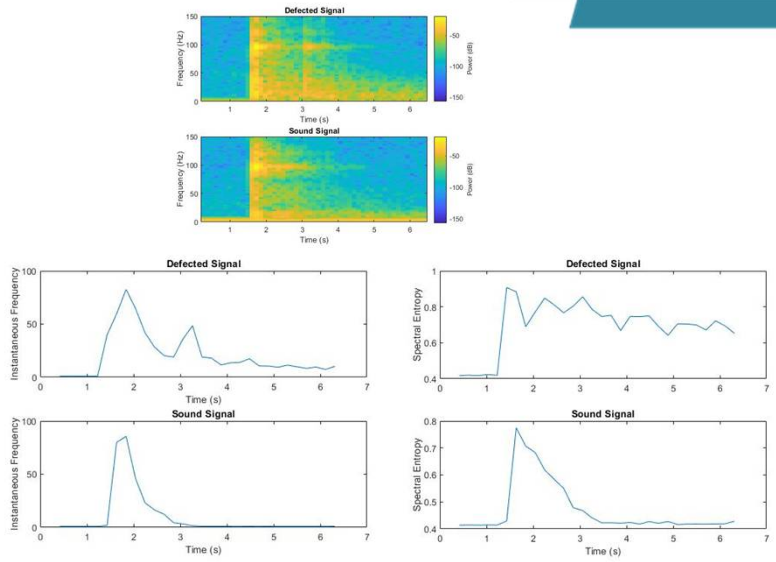
```
CPU reset.
```

Data Reading / Recording with IDE

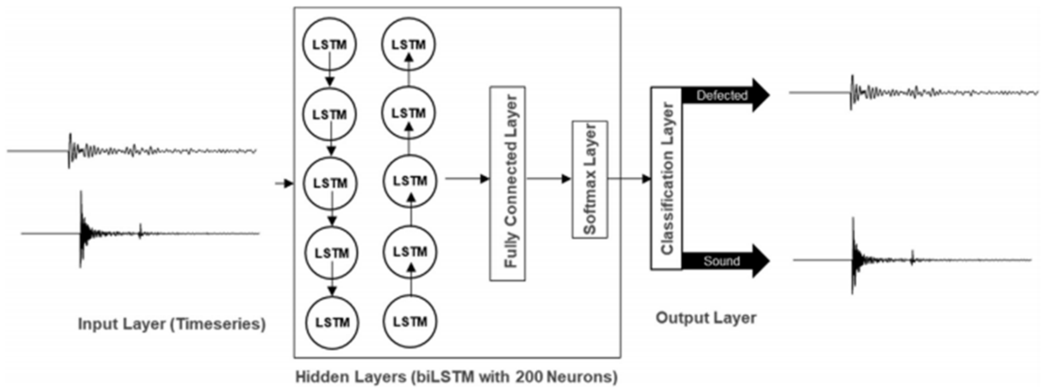
Smart NDT Using DL & Edge Computing



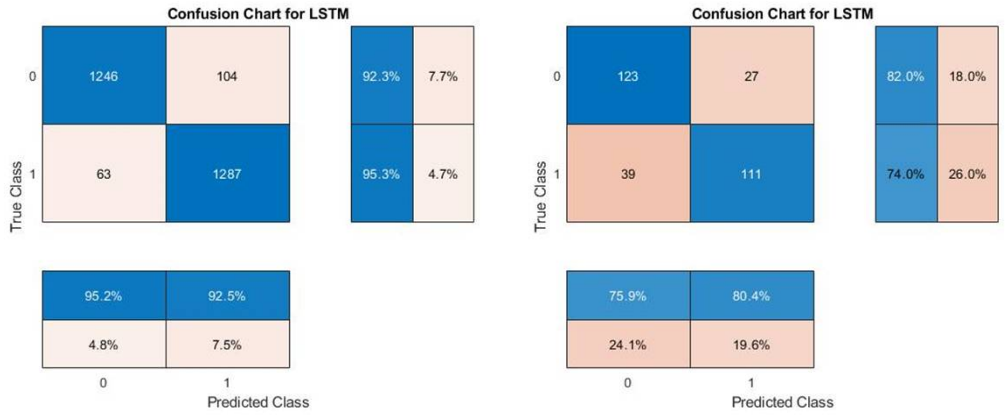
LSTM RNN



Feature Extraction



BiLSTM Networks



Confusion Matrix Results

Smart NDT Using DL & Edge Computing

Summary

- Non-destructive testing is a critical maintenance method for ensuring the safety of structures, but it takes a highly competent individual with a great deal of knowledge as well as a significant investment of money and time.
- Internet of Things (IoT) device that enables impact-echo measurement was developed.
- For the classification of impact-echo time series data, bi-directional long-short term memory neural network was used.
- A pre-trained AI model was embedded in the MCU to implement real-time classification-capable edge computing.