K. M. Mosalam

Cyber-Physical Modeling and Machine-Learning Towards Smart Electrical Equipment Systems

PROJECT FOR COURSE #7

June 11, 2019

Preface

In this project, the students are expected to be familiar with two kinds of modeling techniques, namely continuous modeling using finite element method (FEM) and discrete modeling using beam-column elements. The dynamic properties of electrical equipment and structures will be studied and analyzed and the structural responses will be obtained through numerical simulation based on the above-mentioned modeling methods. For the purpose of building a smart system, machine-learning (ML) and deep learning (DL) are introduced and play important roles in decision making. Through the use of data collected from simulation or field, ML and DL can be performed for identifying current health condition of the system. The specifics are outlined below, which are divided into four parts.

Part 1: Finite Element Method

Part 2: Dynamics of Structures

Part 3: Data-driven Vibration-based Structural Health Monitoring

Part 4: Data-driven Vision-based Structural Health Monitoring

Part 1: Finite Element Method (FEM)

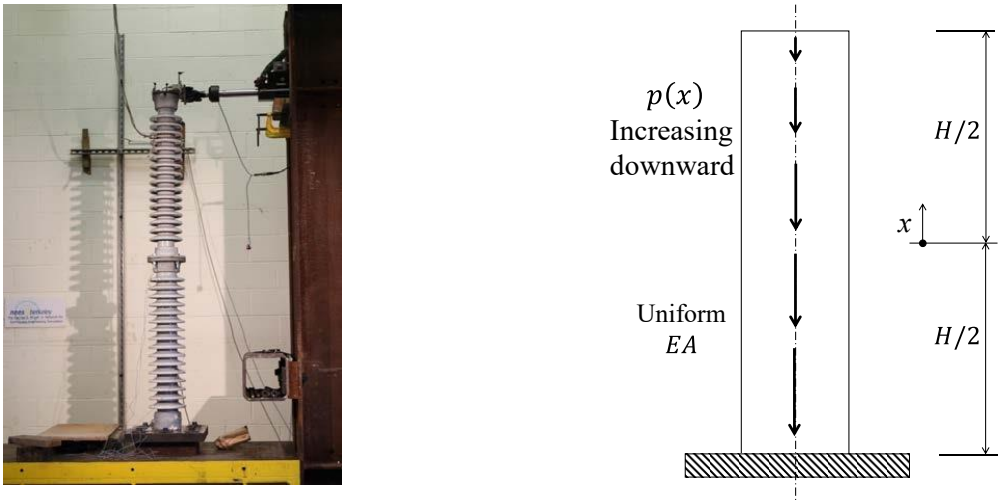
Finite Element Method (FEM) is a numerical approximate method for the analysis of continuous and discontinuous systems in engineering, applied science, and mathematics. Currently, FEM is the most widely employed method of analysis in civil, mechanical, aerospace engineering especially for stress analysis. It is also applied in fluid dynamics, thermal analysis, etc. The main advantage of FEM is that it can be applicable to either continuous (continuum) or discontinuous (frames, networks) systems or to combination of them, particularly when they are subjected to complex boundary conditions where closed form solutions are not available.

1.1 Rayleigh Procedure

Since FEM falls into the scope of approximate method, the principle of virtual displacement (PVD) and minimum potential energy (both are discussed in the lectures) provide the foundation for the approximate solution. Herein, we will go through the basic Rayleigh procedure which provides the foundation of the FEM foundation:

1. Assume an admissible form of the solution with one unknown parameter.
2. Apply PVD to obtain the best estimate of the exact unknown solution, e.g., displacement.
3. Approximate solution satisfies governing equation in an average (weak) sense.

Let us consider the real problem in the electrical system. The high voltage switch, with the porcelain insulator shown in Figure 1(a), is one of the components in this system, and it may experience several types of loads, e.g., gravity, wind, earthquake, or thermal. Now, suppose we can simplify it as a rod or a cantilever with total height H in Figure 1(b), and then convert the loading as a distributed load, $p(x)$, along its height, thus we can compute the displacement and axial force for each section along its height as a function of x. As mentioned in the lectures, for such simplified analytical model, we can solve it with exact resolution, but now it is time to use the Rayleigh procedure to approximate the solution and compare the results to the exact one.



(a) Real insulator (b) analytical model

Figure 1. Illustration of converting real insulator to a simplified analytical model considering gravity loading only

From preliminary analysis, we can obtain the **strong form** as follows:

$$\frac{dP(x)}{dx} + p(x) = 0 \tag{1.1}$$

Several compatibility and material properties can be listed as:

$$\varepsilon(x) = \frac{du}{dx} \tag{1.2}$$

$$\sigma(x) = E \cdot \varepsilon(x) \tag{1.3}$$

$$P(x) = EA\frac{du}{dx} \tag{1.4}$$

Now, given a uniform load case, $p(x) = -p_0$, according to the strong form, we obtain the exact solution for both displacement $u(x)$ and axial force $P(x)$ under this case.

$$u(x) = -\frac{p_0 H^2}{EA}\left(-\frac{1}{2}\left(\frac{x}{H}\right)^2 + \frac{1}{2}\left(\frac{x}{H}\right) + \frac{3}{8}\right) \tag{1.5}$$

$$P(x) = p_0 H\left(\left(\frac{x}{H}\right) - \frac{1}{2}\right) \tag{1.6}$$

Milestone 1:

1. Follow a similar procedure as in the lectures' slides, and **derive the weak form**, where we assume the trial function as $\bar{u}(x) = c\left(x + \frac{H}{2}\right)$, c is a constant to be determined. Be careful with the origin of x axis.
2. Plot displacement $u(x)$ and axial force $P(x)$ for both the approximate and exact solutions and discuss the comparison.
3. Repeat (1) and (2) by using Rayleigh-Ritz method including 2 terms of $(x + \frac{H}{2})$ and $(x + \frac{H}{2})^2$.

1.2 The Finite Element Approximation

Rayleigh procedure gives the sense of approximation, now in this section we will develop a better insight into FEM. In the FEM, the domain and boundary of a structure are subdivided into a number of elements connected by nodes, Figure 2. Subsequently, the body forces, traction BC's, and displacement BC's are specified for the structures. Finally, specified concentrated forces are applied at nodes.

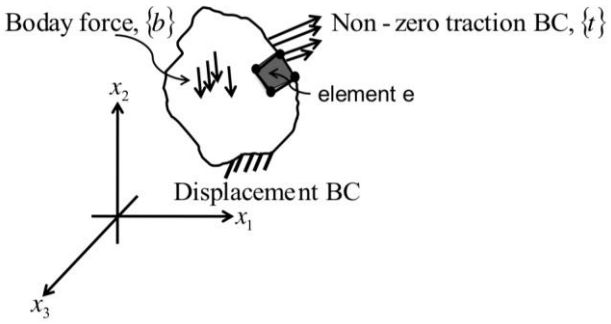
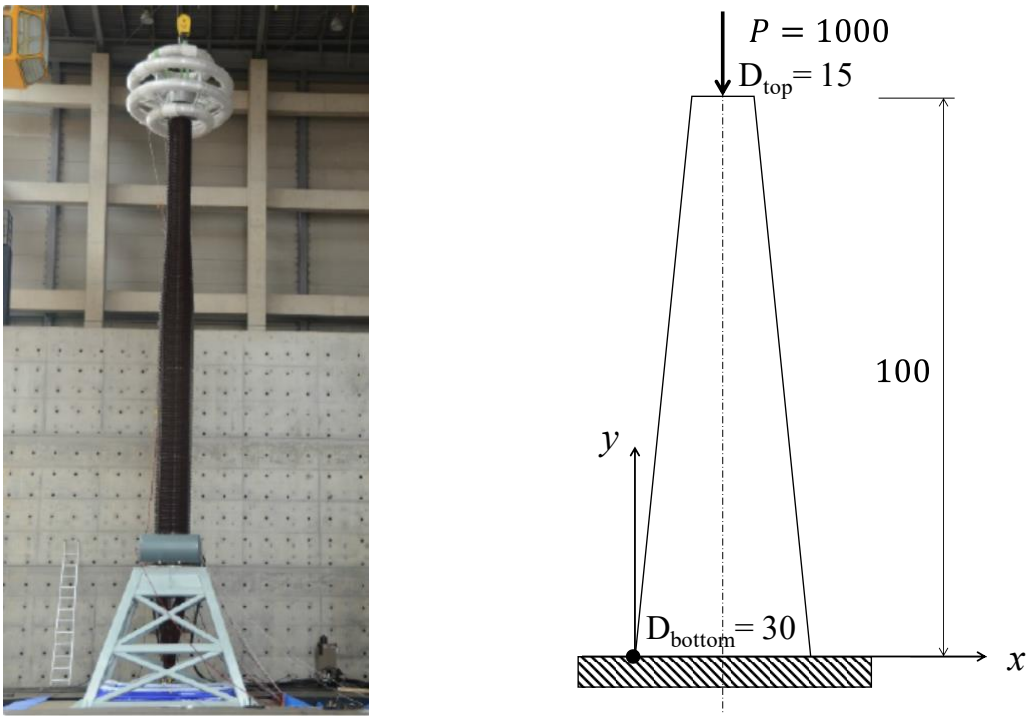


Figure 2. Subdividing the domain and prescribing the boundary conditions in the FEM



(a) Real insulator with varying cross-section (b) Analytical model

Figure 3. Illustration of insulator with varying cross-section and its analytical model

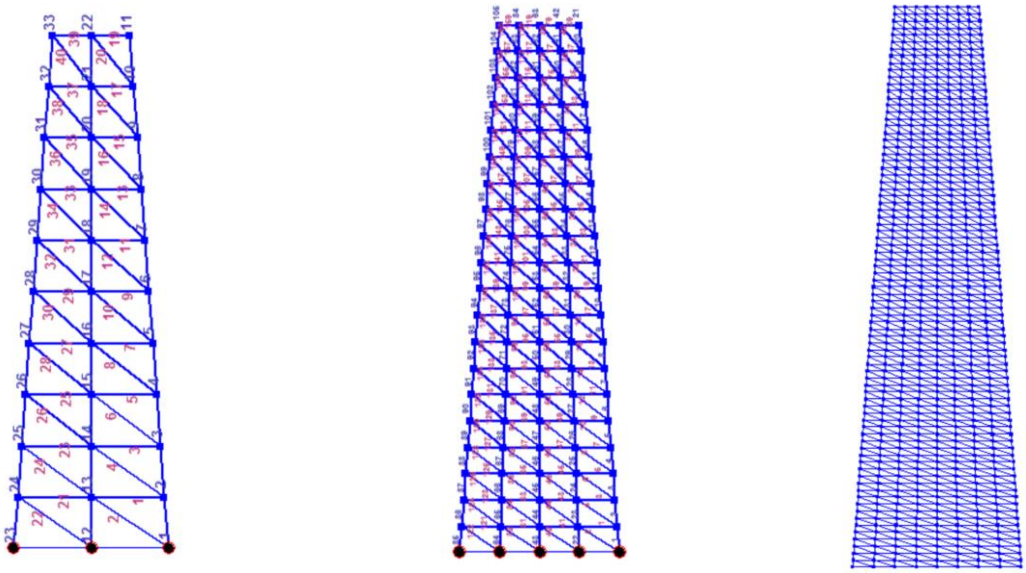
In the real application, there exist several variations of the insulator post due to design requirements, i.e., rendering the insulator with varying cross-section, as shown in Figure 3(a). Similar to milestone 1, the real structure can be converted into a simplified analytical model, Figure

3(b), where we also convert the fixture at the top of the insulator to a concentrated compression force. In the lectures, one regular cantilever beam is analyzed with FEM using structural analysis library FEDEASLab [1] as a custom toolbox in Matlab, which is developed at University of California, Berkeley. Herein, students are expected to follow similar steps to the lectures' problem and form the FE model for this varying cross-section insulator.

Milestone 2:

In this part of the project, we provide the starter code **Milestone2.m** in FEDEASLab, in the part1_FEM folder, students are expected to:

1. Implement the code for structural modeling, i.e., define coordinate for each node. Shown in Figure 3(b), the insulator's diameter of the idealized circular cross-sectional areas at the top and bottom as 15, and 30, respectively. The total height is 100. Note that all unites are consistent and be careful with the origin of x and y axes.
2. Create the mesh of the plane-stress problem with Constant Stress Triangular (CST) elements treating the circular section as an equivalent square for simplicity. Set up the mesh generation with the **Create_Block** function in **CantileverBeamwCST.m** in terms of m elements over height and n elements over width, where at least three combinations of (m, n) need to be checked: (a) $m = 10$ and $n = 2$; (b) $m = 20$ and $n = 4$; and (c) $m = 80$ and $n = 8$. The mesh configurations are shown in Figures 4(a), (b) and (c), respectively.
3. Compare the results for the above-mentioned mesh configurations. How does the displacement and stresses change and how sensitive are they to the mesh refinement?



(a) 10 × 2 mesh (b) 20 × 4 mesh (c) 80 × 8 mesh

Figure 4 Mesh configurations for three combinations

[1]To be provided as a zip file to the students.

Part 2: Dynamics of Structures

In the first part of this capstone project, we use FEM to investigate the mechanical properties of a single component in the electrical system of the high voltage switch. For the purpose of designing and integrating a smart system, in this part of the capstone, a support structure, one-story 3D braced steel frame, which holds the insulator post will be analyzed through its dynamic properties. Moreover, as an extension of this simple support structure, which can be reasonably modeled in the lateral direction as a single-degree of freedom (SDOF) system, the dynamic properties of a 3-story steel frame, which is thought as multi-degree of freedom (MDOF) system will be explored.

2.1 SDOF System

According to CIEE Electric Grid Research report (Mosalam et al., 2012), a shaking table test of a supporting structure with multiple insulators was performed, where the one-story support structure will be studied in this part of the capstone project, Figure 5.

(a) (b)

Figure 5 (a) Insulators are set on one-story steel frame (support structure). (b) Numerical model of the support structure.

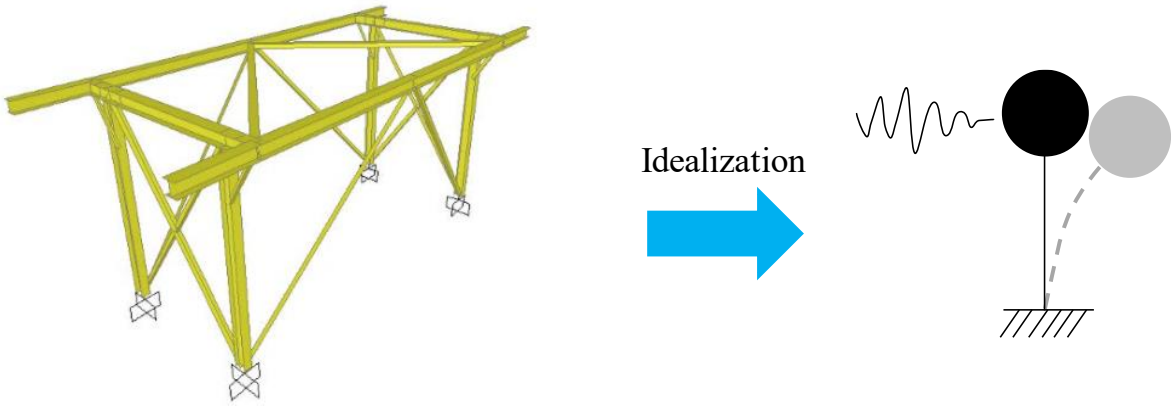


Figure 6 SDOF idealization

Since the support structure is one-story steel frame, it can be idealized as a SDOF system with equivalent stiffness and mass making use of some reasonable assumptions, Figure 6. According to the report (Mosalam et al., 2012), the support structure is fixed at the bottom, two conditions are considered with respect to including/excluding brace, and comparisons between different damping ratios are discussed. These factors influencing the basic dynamic properties of this SDOF system will be investigated in following milestone.

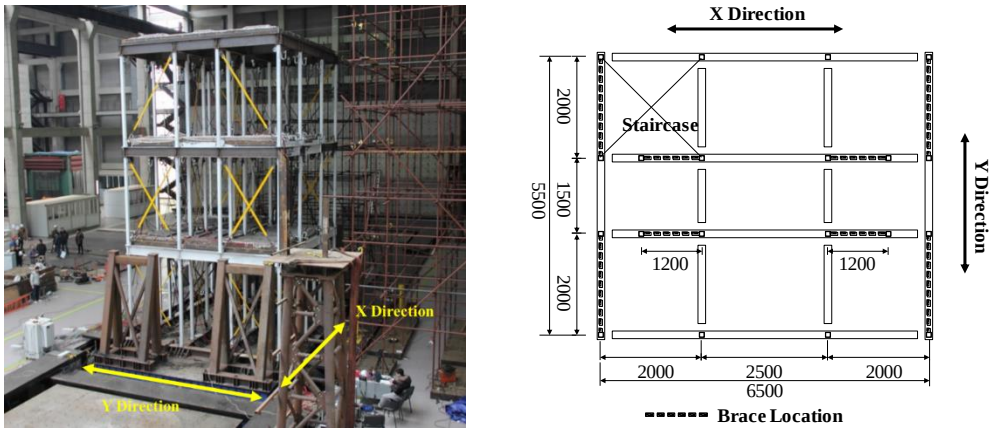
Milestone 3:

Since two conditions are considered, equivalent stiffness values for the support structure with and without braces are $36{,}400\ kN/m$ and $5{,}240\ kN/m$, respectively. Neglecting the weight of the braces, the mass of the support structure can be taken as 8 ton. In this part of the project, we provide the basic information of the SDOF system, students are expected to:

1. Compute the natural period and frequency for both structures with and without braces.
2. Considering damping ratios ξ of 1%, 3% and 5%, repeat step (1).
3. Compare and discuss results in (1) and (2) for the two systems.

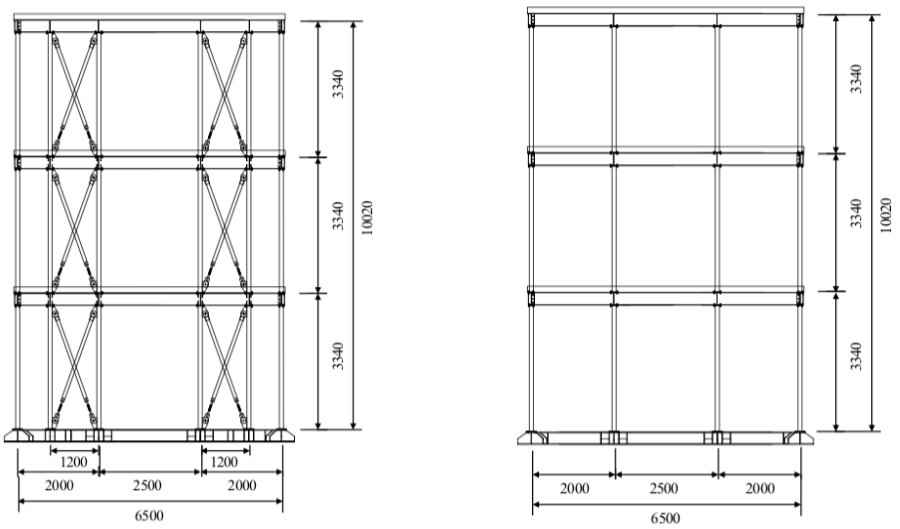
2.2 MDOF System

Now, let us consider the multi-degree of freedom (MDOF) system. The structure we will study is a 3-story, 3-bay, tension-only concentrically braced beam-through frame (TCBBTF) (Chen et al., 2018), Figure 7. The experiment of this steel frame was conducted in the State Key Laboratory of Disaster Reduction in Civil Engineering, Tongji University, Shanghai, China.



(a) General view (b) Plan layout (Dim. in mm)

Figure 7. 3-story tension-only concentrically braced beam-through frame (TCBBTF).



(a) Braced frame (Dim. in mm) (b) Unbraced frame (Dim. in mm)

Figure 8. Two types of frames in the TCBBTF system

For simplicity, in this capstone project, we will only explore specific single frames as part of the whole system, shown in Figure 8, where (a) is the braced frame in the Y direction, and (b) is the unbraced frame in the X direction. This idealization converts the problem from 3D to 2D, which can be solved by the material discussed in the lectures. Neglecting the small weight of the braces, the mass for each frame shown in Figure 8 are: 8.625 ton for the first floor, 8.625 ton for the second floor, and 5.875 ton for the third floor.

Milestone 4:

In this part of the project, we provide the starter code in FEDEASLab, students are expected to:

1. Build the numerical model for both braced and unbraced frames in Figure 8 via filling the corresponding geometric parameters in the starter code. Plot the structure with numbering, where you can simply obtain this via running the start code after implementation.
2. Assign the mass to each node of the frame. It is to be noted that the mass per node for one specific floor is approximately equal to the story mass divided by the number of nodes in this story.
3. Determine the natural (first) frequencies and periods for both braced and unbraced frames.
4. Select 3 ground motion records including earthquake record number 14 (14.txt) in the Motions folder, and then run dynamic time history analysis.
5. Plot the deformed shape of the structures output by the code.
6. Plot the acceleration history and displacement history of each story from the *.mat data file output by the code.
7. Comment on the differences in your results in items (3) to (6) about for the two cases of braced and unbraced frames.

Once completing the above steps, you have the necessary key information to simulate the time series data, i.e., acceleration and displacement for each story of the framed structure. If the numerical model is well-established, these structural responses will be close to the real situation under the same loading and boundary conditions. Thus, with such tool we can obtain as much data as needed, which can be used for data-driven structural health monitoring with machine learning.

Part 3: Data-driven Vibration-based Structural Health Monitoring

Structure health monitoring (SHM) has become an important topic in civil, mechanical and aerospace engineering in the last few decades, especially in structural damage detection. Several approaches and damage criteria were developed for SHM and referred to as vibration-based damage detection. They received considerable attention in the context of statistical pattern recognition, which includes the following steps: 1) operational evaluation, 2) structural data collection, 3) damage sensitive features extraction, and 4) statistical model development of classification. Based on the prevailing concept of performance-based engineering (PBE), qualifying uncertainties and developing empirical fragility functions according to this damage classification are expected to be natural future extensions of the above four steps. Since the turn of this century, time series (TS) modeling of vibration signals using family of autoregressive (AR) models was shown to be effective in damage detection and has been used to capture damage features in structures. Past studies using AR series models can be grouped into two major categories: i) coefficient-based, and ii) residual-based, where the former uses the coefficients of fitted model as damage feature, and the latter identifies damage through measurement of the residuals computed from difference between actual data and fitted data. With the increasing trend of machine-learning (ML) tools, automated structural damage recognition is becoming popular and attracting many researchers.

In this part of the capstone, students will learn the basic properties of TS, i.e., autocorrelation, stationarity, then fit the TS through a family of autoregressive models making use of these properties. Finally, we will introduce the algorithm of an end-to-end framework namely ARIMA-ML, to be discussed later, which combines TS modeling and ML classification together, briefly discussed in the lectures but explained in details in the following sections, to automatically select damage features and perform damage classification.

3.1 Basics of Time Series

In the time series analysis, several properties are important, e.g., stationarity, autocorrelation, partial autocorrelation. In this section, we will briefly go through them making use of the notation: x is one TS signal with duration T, and t and s are two particular time steps in this signal with values expressed as x_s and x_t.

3.1.1 Autocorrelation

Autocorrelation[2], also known as serial correlation, is the correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them. The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. It is often used in signal processing for analyzing functions or series of values, such as time domain signals.

Before defining the autocorrelation function (ACF), we introduce the autocovariance function, $\gamma(s,t)$, which is also known as second moment product for all s and t with mean values μ_s and μ_t at these respective times (note that E is the expectation operator):

$$\gamma(s,t) = cov(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)] \tag{3.1}$$

[2] https://en.wikipedia.org/wiki/Autocorrelation

The autocovariance function measures the linear dependence between two points on the same series observed at different times. Then, the ACF is defined as a normalized version of the autocovariance function:

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\,\gamma(s,s)}} \tag{3.2}$$

The ACF defines the relationship of two points with a measure of association between -1 and 1.

3.1.2 Partial autocorrelation

Partial autocorrelation function[3] (PACF) gives the partial correlation of a stationary time series with its own lagged values, regressing the values of the time series at all shorter lags. It contrasts with the ACF, which does not control for other lags. This function plays an important role in data analysis aimed at identifying the extent of the lag in an autoregressive model.

As for the TS x, PACF is defined as the correlation between x_t and x_{t+k} after removing the effect of the intervening variables $x_{t+1}, x_{t+2}, \ldots, x_{t+k-1}$. Consider predicting x_t based on a linear function of the intervening variables $x_{t+1}, x_{t+2}, \ldots, x_{t+k-1}$, say, $\hat{x}_t = \beta_1 x_{t+1} + \beta_2 x_{t+2} + \cdots + \beta_{k-1} x_{t+k-1}$, with β's chosen to minimize the mean square error of prediction. If we assume that β's have been so chosen and then think backward in time, it follows from stationarity that the best predictor of x_{t+k} based on the same $x_{t+1}, x_{t+2}, \ldots, x_{t+k-1}$, the predicted $\hat{x}_{t+k}$, can be formulated as $\hat{x}_{t+k} = \beta_1 x_{t+k-1} + \beta_2 x_{t+k-2} + \cdots + \beta_{k-1} x_{t+1}$.

The PACF at lag k is then defined to be the correlation between the prediction errors; that is:

$$\begin{aligned} \phi_{kk} &= corr(x_{t+k} - \hat{x}_{t+k}, x_t - \hat{x}_t) \\ &= corr(x_t - \beta_1 x_{t-1} - \beta_2 x_{t-2} - \cdots - \beta_{k-1} x_{t-k+1}, \\ &\qquad x_{t+k} - \beta_1 x_{t+k-1} - \beta_2 x_{t+k-2} - \cdots - \beta_{k-1} x_{t+1}) \end{aligned} \tag{3.3}$$

This coefficient is called partial autocorrelation at lag k ($k \geq 2$).

3.1.3 Stationary Time Series

A stationary time series is one whose statistical properties, e.g., mean, variance, and autocorrelation are constants over time. Most statistical forecasting methods are based on the assumption that the time series can be rendered approximately stationary (i.e., "stationarized") through the use of mathematical transformations. In statistics, the stationarity will be defined in two ways, strictly stationary and weakly stationary. Usually strictly stationary is too strong and hard to satisfy for most applications. Thus, we will only introduce and use weakly stationary and just refer to it as "stationary" (for simplicity) in the following text.

A weakly stationary TS, x_t, is a finite variance process such that:

1. the mean value function, μ_t, is constant and does not depend on time t.
2. the autocovariance function, $\gamma(s,t)$, depends on s and t only through their differences or lag $|s-t|$.

Many models and properties are required for a stationary TS, thus checking stationarity of the test TS is important. There are many ways to do so, but in this part of the capstone, we will simply introduce the Augmented Dickey-Fuller (ADF) test (Fuller, 2009). ADF test is widely used in statistics and econometrics, whose null hypothesis is the presence of a unit root in the TS signal

[3] https://en.wikipedia.org/wiki/Partial_autocorrelation_function

indicating the non-stationarity, and the alternative hypothesis is test-dependent that is usually the occurrence of stationarity or trend-stationarity. In this part of the capstone, we define the null hypothesis as a unit root of a univariate TS, and the alternative hypothesis is stationarity. If p-value (calculated probability) obtained by the ADF test on a TS signal is less than 0.05, the null-hypothesis is rejected and this indicates the stationary condition.

3.2 Autoregressive Model

In classical TS regression models, the relationship between different variables are established through regression analysis and then the links between different target observations are evaluated. autoregressive (AR) series model is a family of regression models used for general TS problems, which includes models for AR, moving average (MA), autoregressive moving average (ARMA), ARMA with exogenous (ARX), etc. However, there are some drawbacks limiting the use of AR series modeling in practice. The most notable is the requirement of stationary input. Such strong assumption is usually hard to achieve in real engineering applications, e.g., in SHM, where TS data (i.e., vibration signals) collected from sensors are usually non-stationary. Thus, elaborate data preprocessing and stationarity checks are inevitable before modeling, but these methods lack uniformity and systematical pipeline, and they may still not guarantee stationarity, i.e., preprocessing procedures like segmenting, de-trending, de-nosing, etc. may not work in some cases. However, there is one promising AR series model, namely the autoregressive integrated moving average (ARIMA) model. The ARIMA model is developed by performing several integrations (differencing) towards TS, which is shown to be quite effective to achieve stationarity compared to other traditional data processing methods in some cases (Shumway & Stoffer, 2011). Thus, ARIMA can be an alternative way to relieve such strong assumption compared to other AR series models. For more details about AR series models, refer to (Shumway & Stoffer, 2011).

In this capstone, details about ARIMA modeling are introduced and applied. ARIMA acts as a combination of both AR and MA models with several integrations, where AR aims to determine the dependent relationship of an observation with order p of lagged observations, namely AR(p). On the other hand, MA is used to determine the dependent relationship of the observation with order q of observed white noise error terms, namely MA(q) (Shumway & Stoffer, 2011). Moreover, using order I to represent the times of integration after fitting an ARMA(p, q) model, the full definition of the ARIMA model can be expressed as ARIMA(p, I, q). From the perspective of implementation, we do not perform integration, instead we perform differencing of raw TS multiple times first and then fitting an ARMA model. If a TS, expressed as $\{X_t\}$, is said to satisfy the stationarity condition after the dth differencing of a raw TS, expressed as $\{Y_t\}$, denoted as $\{X_t\} = \nabla^d\{Y_t\}$, and $\{X_t\}$ also follows an ARMA(p, q) model, then it can be said that $\{Y_t\}$ follows an ARIMA(p, d, q) model. Thus, we have,

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \cdots + \alpha_p X_{t-p} + e_t - \beta_1 e_{t-1} - \beta_2 e_{t-2} - \cdots - \beta_q e_{t-q} \qquad (3.4)$$

where α_i and β_i represent coefficients for the AR and MA parts, respectively, and e_t represents the error (noise) at time step t.

As a parametric model, the key factor in using ARIMA to fit-well a TS is to find the reasonable orders p, d and q. In the usual applications, taking $d = 1$ or 2 can achieve reasonable results (Shumway & Stoffer, 2011). It is not suggested to use high order differencing, since it

compromises the physical meaning and increases the risk of overfitting. As for orders p and q, they can be roughly estimated through the stationary TS properties, namely the autocorrelation function (ACF) and partial autocorrelation function (PACF). Shumway and Stoffer (2011) summarized the behavior of the ACF and PACF for specific AR series models as follows: 1) TS suitable for AR(p) models is observed as ACF tails off and PACF cuts off after lag p; 2) TS suitable for MA(q) models is observed as ACF cuts off and PACF tails off after lag p; and 3) TS suitable for ARMA(p, q) models is observed as both ACF and PACF tails off after order p and q, respectively. Thus, in the implementation of ARMA, it is necessary to obtain the ACF and PACF of the TS first, then find the cut-off and tail-off lags, and finally determinate the order empirically based on these lags. There are other approaches to determine the model specifications such as using Akaike's information criterion (AIC), or Bayesian information criterion (BIC), expressed in the two equations below. It is noted that most of the work of Nair et al. (2006) and Noh et al. (2007) were based on using AIC to determine the optimal order for the AR series model.

$$\mathrm{AIC} = 2k - 2\ln(\hat{L}) \tag{3.5}$$

$$\mathrm{BIC} = k\ln(n) - 2\ln(\hat{L}) \tag{3.6}$$

where n is the number of samples, k is the number of coefficients in the model, and $\hat{L}$ is the maximum value of the likelihood function for the model.

To avoid both over- and under-fitting for the AR series modeling, we can diagnose such undesired performances by residuals checks. This is true because the residuals (difference between the ground truth and the fitted data) should share similar properties to the white noise, such as independent and identical distribution (i.i.d), zero means, and common values of standard deviation (Cryer et al., 1991). Several statistical methods to examine the residuals exist including checking their standardized scale by standardization plot, their normality by quantile-quantile (Q-Q) plot, and their autocorrelation by ACF plot.

3.3 Smoothing-Segmentation-Normalization-Differencing (SSN-D)

In the procedure of signal processing, data cleaning or preprocessing is always the first essential step. Usually, moving average operation can be applied first to smooth the raw signals, which is helpful to get rid of some noisy data points. Considering the difficulties in maintaining the stationarity for a long-round TS signal and to have a better performance in achieving the stationarity condition, segmentation is adopted, where local stationarity is much easier to achieve for a short time period. Similarly, strong nonlinearity of the entire TS signal can also be relieved through fitting local linear model on each segment. On the other hand, due to different loading conditions, the scale of the TS collected by several sensors may vary from one sensor to another. Thus, normalization (standardization) may resolve this issue. Moreover, Nair et al. (2006) and Noh et al. (2007) indicate the effectiveness of performing segmentation and normalization.

Different from traditional segmentation, in this capstone, we set the principle for segment selection to maintain enough information for the structural stiffness under consideration. Thus,

each segment should contain at least one reciprocating cycle, such as the three sample segments in Figure 9. According to the sampling frequency (256 Hz) of the sensors, using 200 data points as the segmentation size should be sufficient in this capstone and we can discard the segments that violate this principle. Similar to the traditional way, the sliding window method is adopted, and in the numerical experiments, we consider the two patterns of non-overlapping and overlapping. Another advantage of segmentation is to greatly increase the amount of data especially when overlapping is adopted, which is beneficial for data-driven ML algorithms.

To minimize the variance of local time period, instead of normalizing the whole signal at one time, the signal is normalized within each segment, where Y represents a collection of TS signal records and Y_j^i is the jth segment in the ith record. The normalized signal $\tilde{Y}_j^i$ is obtained by subtracting the mean of Y_j^i and then dividing by the standard deviation (std) of Y_j^i as follows:

$$\tilde{Y}_j^i = \left(Y_j^i - mean\left(Y_j^i\right)\right)/std\left(Y_j^i\right) \tag{3.7}$$

After normalization, d times of differencing operations are conducted for each segment. The selection of d is based on demand and vary between cases.

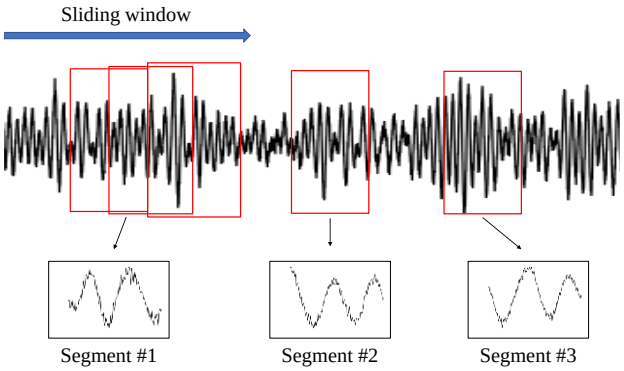


Figure 9. Segmentation operation using sliding window for one sample TS signal

Milestone 5:

In this part of the project, the students are expected to be familiar with and use functions of time series library in Matlab, i.e., arima(), adftest(), autocorr(), etc. to examine the properties of **one sample TS signal** we will provide, which is a real acceleration data collected from shaking table test discussed with relevance to milestone 4. Specific steps are as follows:

1. Compute the ACF, PACF for the sample TS signal (show plots).
2. Implement the code of SSN-D preprocessing, where sliding window method should have the options for both overlapping and non-overlapping patterns. Check stationarity for each segment and discard non-stationary ones.
3. Repeat (1) for first 10 segments generated by non-overlapping pattern and show plots.
4. Combine observed results in the first 10 segments with empirical principle for ARMA model selection and select 3 possible combinations of order (p, q).
5. Use arima() function in Matlab to fit the first 10 segments with proper order (p, q), and check the residuals based on the criteria introduced in the lectures.

3.4 Machine Learning in Classification

Machine-learning (ML), founded on pattern recognition as one mutual subject in Computer Science and Statistics, has already undergone substantial development over the past twenty years (Bishop, 2006). Nowadays, the implementation of ML in SHM pattern recognition can be thought of as a mature subject with plenty of well-developed and efficient algorithms (Farrar, 2012). ML has three major categories: unsupervised learning, supervised learning, and reinforcement learning based on data characteristics. In this capstone, only supervised learning will be applied. The supervised learning uses well-labeled data based on domain knowledge to analyze the input data and produce an inferred mapping function. Finally, the new unseen data sample can be assigned labels according to this mapping relationship. In the implementation of ML to SHM, a supervised learning algorithm is used to learn hidden relationships between some features extracted from the data and their corresponding damaged state of the structure.

An early exploration in this direction was performed by de Lautour & Omenzetter (2006). They numerically simulated a simple 3-story shear building and used coefficients fitted from floor acceleration by AR model as the input feature of the neural network. The results indicate the success in detecting and locating the damage of this simple structure. This preliminary work inspired the current capstone to develop an end-to-end detection framework/pipeline by ARIMA modeling in conjunction with ML classifier. In this section, three well-known ML classification algorithms are introduced, namely logistic regression (LR), neural network (NN), and support vector machine (SVM).

3.4.1 Logistic regression (LR)

The LR is a technique applied to problems with binary response variable, y, i.e., the number of available categories (cardinality) is two, with conditional probability $\mathrm{P(y = f|x)}$. In that case, a logit model is fitted between the input features x and the binary response (Truett et al., 1967). The formulation of LR is described as follows:

$$\ln\left(\frac{P(y=f|x)}{1-P(y=f|x)}\right)=\theta x \tag{3.8}$$

where θx is a linear combination of the features and f is the damage class. The optimization problem by which θ is determined can be expressed as follows:

$$\min_{\theta}\left\{-\frac{1}{m}\times\left[\sum_{i=1}^{m}\left\{y^i\ln\left[h\left(\theta x^i\right)\right]+\left(1-y^i\right)\ln\left[1-\left(h\left(\theta x^i\right)\right)\right]\right\}\right]\right\} \tag{3.9}$$

where $h\left(\theta x^i\right)=1/\left(1+\mathrm{e}^{-\theta x^i}\right)$ and m is the number of samples.

When the number of categories (cardinality) is more than two, LR can still be applied for classification using an approach called “one vs all.” For example, if we define four damage categories, namely “undamaged,” “minor,” “moderate,” and “major,” which are assigned categories 0, 1, 2, and 3, respectively, in this approach, one damage class is labeled as 1 and all the rest are labeled as 0 for training and the process is repeated for each class. Using Equation 3.8 and the training set, the coefficient vectors θ are evaluated. The LR algorithm can help find the linear boundary that separates each damage category from the other categories.

3.4.2 Neural Network (NN)

The NN is the most commonly used ML algorithm for its ability to handle nonlinear complex input-output relationships. A typical NN contains connected units or nodes known as artificial neurons. The network comprises of three main layers: the input layer, the hidden layer, and the output layer. The hidden layer can have multiple layers of its own. This layer finds the relationship between the input and the response variable. However, unlike LR, it can find the linear as well as the nonlinear boundary that separates each class from the remaining classes. For example, the feedforward network architecture is applied with the vector of damage features as the input layers and 2 hidden layers with the number of neurons equals n, and the output layer consists of the four damage states as shown in Figure 10(a).

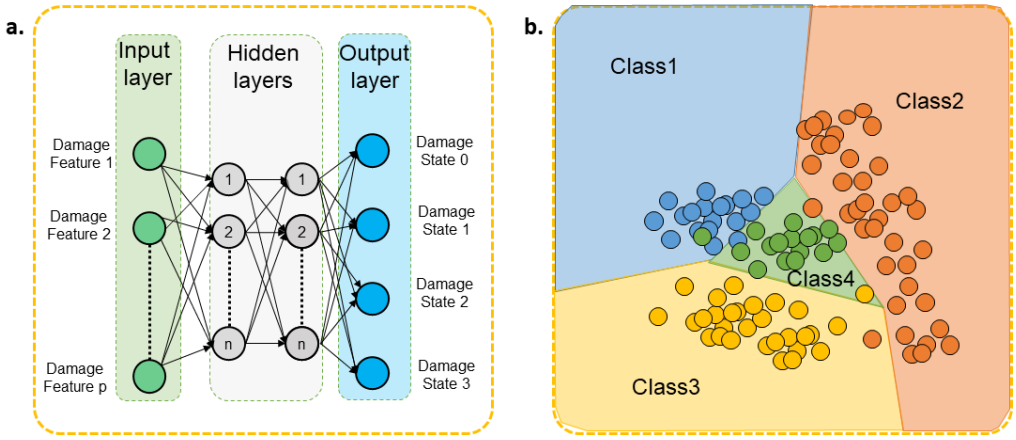


Figure 10. (a) Feedforward NN with 2 hidden layers and n neurons. (b) SVM classification of four classes with six distinct boundaries.

3.4.3 Support Vector Machine

The SVM is another common classical ML algorithm used for classification. SVM is based on the statistical learning theory, so it finds the optimum boundary (linear and nonlinear) separating the data of different classes (clusters). SVM maps the space of the original variables into an unknown high-dimensional feature space, where the data are linearly separable. In this space, the classes are separated through hyperplanes, to which correspond nonlinear boundaries in the original space, Figure 10(b).

3.5 ARIMA-ML Algorithm

To some degree, SHM for damage detection can be seen as a classification problem, which is within the scope of supervised learning. Therefore, in this capstone we aim to integrate both TS modeling and ML together for such purpose, namely ARIMA-ML algorithm (Gao et al., 2019). Figure 11 illustrates the basic framework of integrated ARIMA-ML by first passing raw TS data into SSN-D pre-processing module and stationarity check to generate the desired segments, then applying ARMA model to act as a damage feature extraction module, and finally feeding these features into the classification module, where multiple classification ML algorithms with a *voting mechanism* for decision making regarding the damage state identification or pattern recognition. The details of each module are described in (Gao et al., 2019).

In summary, with the input of one TS signal and a target TS model, the ARIMA-ML framework automatically processes and analyzes it with output of the corresponding label, i.e., damage state or damage pattern, and no human-interacted operations need to be performed during this procedure. Thus, such characteristic provides the convenience for practical deployment. It is noted that the desired input for this algorithm is the structural acceleration response collected before and after the possible damaging loading scenarios, e.g., due to a large earthquake, and such responses can be excited from small turbulences or white noise loadings of the structures. The intuition behind this definition is that the damage state or pattern occurring in the structure is determined except during the damaging loading stage, and small excitations, i.e., white noise with low amplitude, do not cause further damage to the structure, i.e., it remains in the linear range even though the structure may have a degraded stiffness compared to its undamaged state. Such condition is desired for AR series model, and the issues of strong non-stationarity and nonlinearity are significantly relieved compared to the damaging loading state.

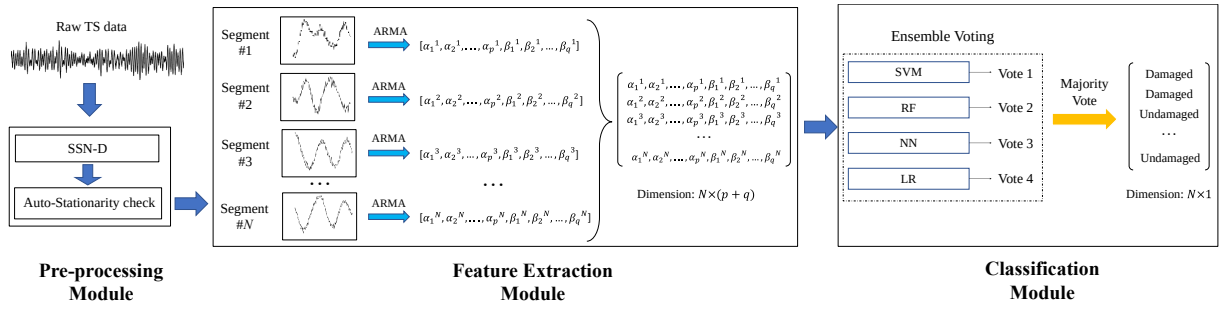


Figure 11. Framework of the ARIMA-ML algorithm

Milestone 6:

In this part of project, training data, a matrix form of damage feature and labels will be provided, which are extracted from real shaking table test of the 3-story steel frame mentioned in Section 2.2. According to (Gao et al., 2019), these features are extracted from ARIMA order (p, q) pairs, where we used $p = 7, q = 6$, for each segment from real experiments, and students already have the sense of such feature extraction in milestone 5. The classification task designed here is a binary case to identify whether the structure is damaged (D) or undamaged (UD). Moreover, in this part, the students are allowed to use ML library in Matlab instead of implementing ML algorithm by themselves. However, it is suggested to try to do so for an **Extra Credit (Bonus)** awarded for self-implementation.

The details of the assignment are listed below:

1. Apply LR to train the data and report on the test accuracy.
2. Apply SVM and tune the hyper-parameters to obtain the best classification accuracy.
3. Try NN with at least two different network designs (e.g., 2 and 3 hidden layers) and tune hyper-parameters (i.e., learning rate, etc.) to obtain the best classification accuracy.
4. Compute the confusion matrix (discussed in the lectures) for all cases (LR, SVM, and NN)
5. Compare results and submit a discussion of findings.

Part 4: Data-driven Vision-based Structural Health Monitoring

As mentioned in the lectures, other than vibration-based SHM presented in part 3, vision-based approach is another important and well-studied direction in nowadays SHM area. The objective of vision-based SHM is to detect vision patterns, e.g., cracking, spalling, and buckling to be able to extract information related to damage, i.e., damage level, damage type, etc., through images and videos. Usually this work is performed by human or some automated algorithm, but there exist many drawbacks for such approach where tedious and repetitive inspection work for human and inaccurate detection results for automated algorithms exist. In this data explosion epoch, artificial intelligence (AI) and machine learning (ML) technologies are developing rapidly, especially in applications of deep learning (DL) in computer vision, which made giant progress in recent years. In addition, the objective of the implementation of ML and DL is to make computers perform labor-intensive repetitive tasks and also learn from past experiences with a stable and highly accurate procedures. Considering the difficulties in vision-based SHM, which greatly relies on human visual inspection and detection accuracy, it is timely to implement the state-of-art DL technologies in vision-based SHM applications especially in detecting health conditions of electrical equipment and systems and evaluate their potential benefits.

Electrical equipment and systems require constant maintenance in order to supply constant and stable power, especially in urban areas. Electrical equipment systems in dense urban areas are complex and large scale, and often housed outdoors. Wind, precipitation, and ground motion put electrical equipment systems at risk of failure, especially where there are brittle electrical connections in the equipment. It is important to detect damage quickly, safely, and efficiently so that maintenance and repair can be conducted as soon as possible after an event. Slow response to damage can cause life safety issues, long down times, power outage, and financial losses. Transmission towers are an example of electrical equipment where damage must be detected and addressed quickly. An automatic failure detection of transmission towers can save communities the time and potential danger of traveling out to the field and investigation failure conditions. Thus, DL can be used to greatly improve the process of transmission tower maintenance and repair.

In this part of the capstone, students are expected to use image data we provide and adopt DL to train the convolutional neural network (CNN) for the purpose of transmission tower failure detection through image. An example of a CNN framework is shown in Figure 12.

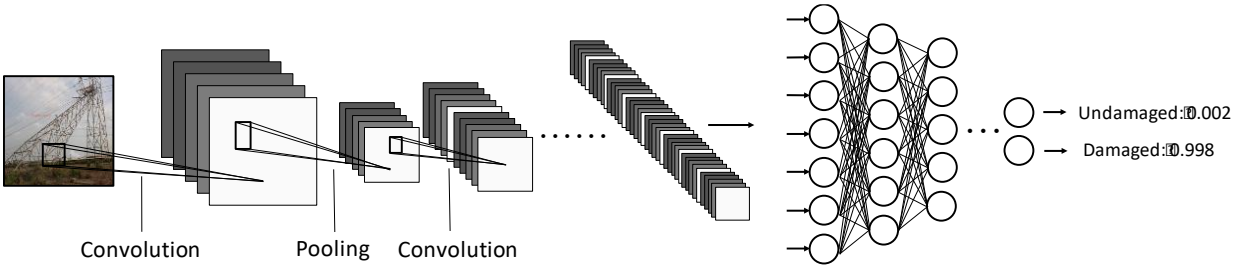


Figure 12. CNN framework for transmission tower failure detection

4.1 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) has been at the heart of spectacular recent advances in DL. Compared with traditional computer vision and ML approaches, CNN no longer needs hand-designed low-level features or so-called feature engineering where the millions of parameters inside a typical network are capable of learning amounts of mid-to-high level image representations with input data obtained from a pixel matrix (tensor). Another unique characteristic of deep CNN is its depth of architecture. Many well-designed CNN architectures, such as Visual Geometry Group (VGGNet) (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2015), and deep Residual Net (ResNet) (He et al., 2016) demonstrated the great performance improvement with substantially increasing the depth. A CNN primarily consists of two parts: image feature extraction using convolution layers and image classification with fully-connected (Fc) layers, whose input is the output of the last convolutional layer.

4.1.1 Convolutional layer

A convolutional layer extracts the features of an image using filters. A filter is a small sized matrix (e.g., $3 \times 3 \times$ number of image channels) that traverses the image matrix with a predetermined step size (called "stride") and performs the convolutional operation between the masked image submatrix and the filter's own kernel matrix, which is learned and tuned during the training process, shown in Figure 13. Each convolutional layer consists of multiple filters. These convolutional filters extract image features based on their relative locations, outputting "feature maps," which are the results of the convolutional operations. Detectable features include, but not limited to, straight or curved edges, color patterns, or geometric combinations. As more convolutional layers are added, higher image features are captured, such as complex geometric patterns.

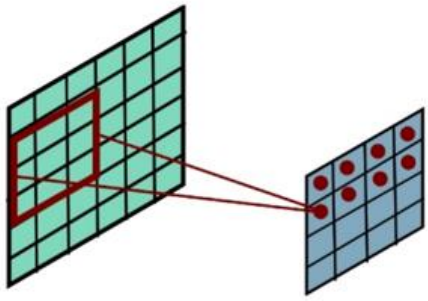


Figure 13. Convolutional operation

4.1.2 Fully-connected (Fc) layer

A fully-connected (Fc) neural network is another name of neural network (NN) mentioned in section 3.4.2 and Figure 8(a), which is added to the end of the convolutional layers in the setting of CNN. By this point, high level image features are extracted and summarized in high dimensional feature maps, which are fed into the first Fc layer. Each neuron in an Fc layer is connected to all the next layer neurons; each neuron calculates and outputs the dot product between the input vector and its feature weights (which is learned and tuned during the training process). The last Fc layer

calculates the percentage that a given input feature map belongs to a certain image class through the Softmax function (as discussed in the lectures).

4.1.3 Maxpooling layer

A Maxpooling layer down-samples each internal feature map by traversing its filter across the feature map (similar to convolutional filters) and extracting the maximum entry value of the feature map's submatrix masked by the filter. This is illustrated in Figure 14.

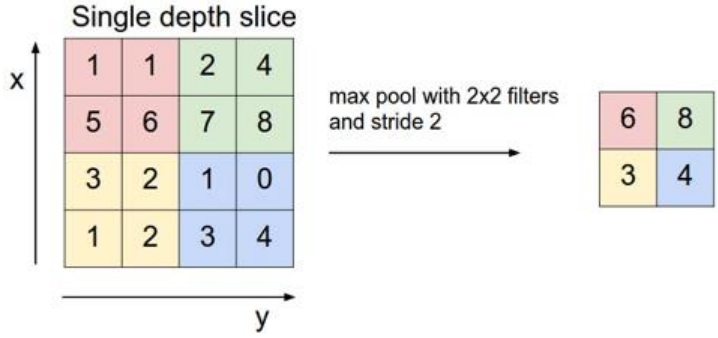
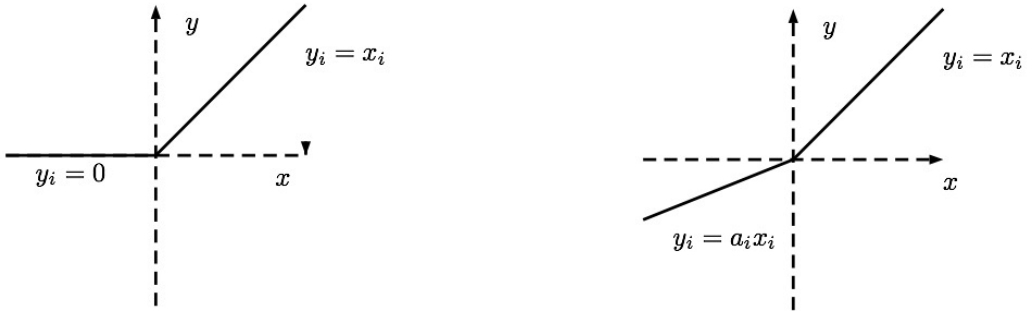


Figure 14. Maxpooling operation

4.1.4 Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is an important activation function at convolutional layers. Its operation is simple: $f(x) = max(0, x)$, shown in Figure 15(a). ReLU activation is crucial because it breaks the linear boundary between convolutional layers. Without ReLU, all convolutional operations are linear, which poses a problem because relative features are often nonlinear. ReLU greatly improves both classification accuracy and training speed. There are variations of the ReLU function as well, such as the LeakyReLU which does not ignore negative inputs, Figure 15(b).



(a) ReLU (b) LeakyReLU

Figure 15. Activation function

4.2 Image Dataset for Transmission Tower

If a CNN model and DL algorithm build the skeleton of the entire project, data and dataset are the blood and muscle to make everything runnable. Since using DL in image-based detection in SHM particularly for electrical equipment, there is no open-source well-labeled and well-organized image dataset related the topic of this capstone. Thus, we collected images from Internet, preprocessed them and finalized a small dataset for exploration purposes of this capstone project.

The dataset for the capstone contains about 400 images of transmission tower with both undamaged and damaged states, shown in Figures 16(a) and (b). The total dataset is split up to two sets, namely training set and validation set. The DL model you will develop will be trained on the training set and validated on the test set. The amounts of data in each set and category are listed in Table 4.1. For simplicity, we treat the ratio of undamaged state and damaged state as 1:1. Therefore, the model should have a better accuracy over 50%, which is a random guess of a binary classification problem on a balanced dataset.

(a) Undamaged state indicating no failure

(b) Damaged state indicating failure

Figure 16. Sample images in the dataset for both undamaged and damaged categories.

Table 4.1 Number of images in the training and validation sets

Set	Undamaged state	Damaged state
Training	181	181
Validation	23	23

Milestone 7:

In this part of the project, the students are expected to be familiar with and use functions of the Deep Learning (DL) library in Matlab. Students will implement the code to realize transmission tower failure detection via images with acceptable accuracy. An example of the training procedure is shown in Figure 17. Specific steps are as follows:

1. Apply the code to preprocess image data.
2. Run the code to load image data from the dataset we will provide.
3. Design 3 different CNN architectures based on choices of the following:
    a. Convolutional layer
    b. Fc layer
    c. Maxpooling layer
4. Modify one of the CNN you designed in part (2) above until it can achieve over 80% validation accuracy on the validation set.
5. **BONUS:** Implement data augmentation (as discussed in the lectures) and compare the modified model in part (3) above to test its performance with and without augmentation.

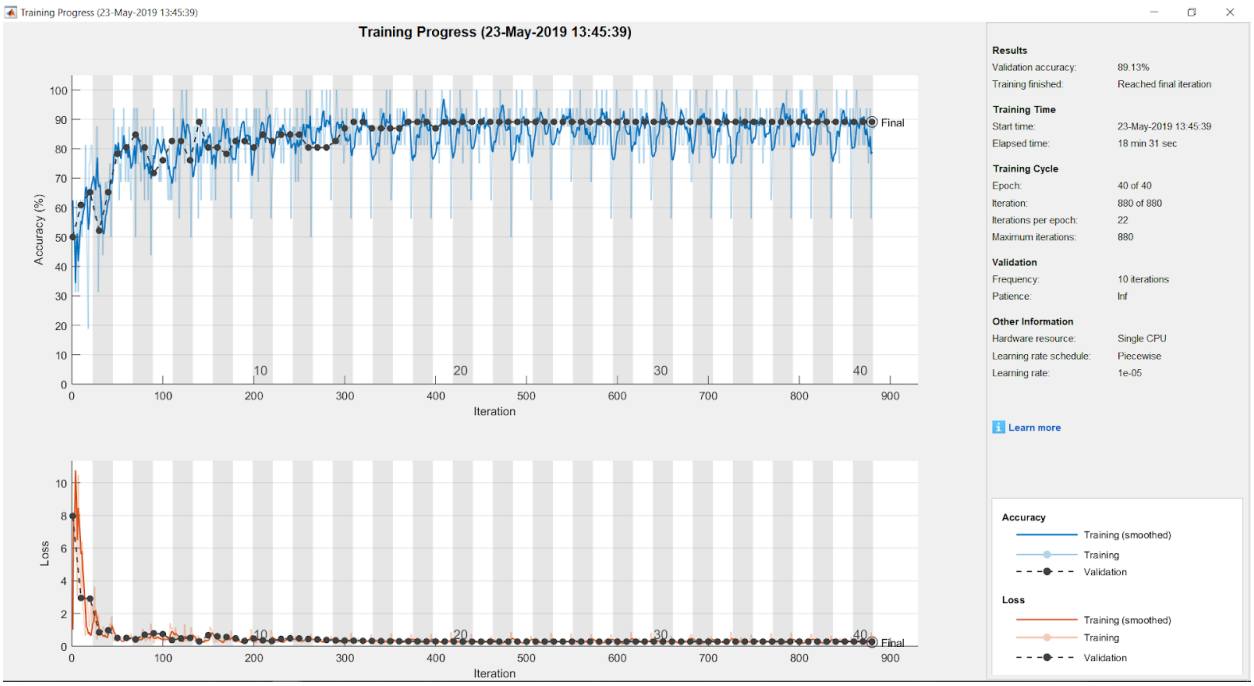


Figure 17. Training procedure of CNN model through the Matlab DL library

References

1. Mosalam, K. M., Moustafa, M. A., Günay, M. S., Triki, I., & Takhirov, S. (2012). Seismic performance of substation insulator posts for vertical-break disconnect switches. California Energy Commission Publication Number: CEC-500-2012. 2012
2. Nair, K. K., Kiremidjian, A. S., & Law, K. H. (2006). Time series-based damage detection and localization algorithm with application to the ASCE benchmark structure. Journal of Sound and Vibration, 291(1), 349-368.
3. Noh, H. Y., Nair, K. K., Kiremidjian, A. S., & Loh, C. H. (2009). Application of time series based damage detection algorithms to the benchmark experiment at the National Center for Research on Earthquake Engineering (NCREE) in Taipei, Taiwan. Smart Structures and Systems, 5(1), 95-117.
4. Farrar, C. R., & Worden, K. (2012). Structural health monitoring: a machine learning perspective. John Wiley & Sons.
5. Gao, Y., Chen Y., Mosalam, K. M., Chen Y. & Wang, W. Auto Regressive Integrated Moving Average–Machine Learning Approach for Damage Identification of Steel Structures. (Under preparation)
6. Muin, S., & Mosalam, K. M. (2017). Cumulative absolute velocity as a local damage indicator of instrumented structures. Earthquake Spectra, 33(2), 641-664.
7. Liang X., S. Muin & Mosalam K.M. (2018). Simulation-based Data-driven Damage Detection for Highway Bridge Systems," Eleventh U.S. National Conference on Earthquake Engineering, Integrating Science, Engineering & Policy, June 25-29, 2018, Los Angeles, CA, USA.
8. Mosalam, K. M., Alibrandi, U., Lee, H., & Armengou, J. (2018). Performance-based engineering and multi-criteria decision analysis for sustainable and resilient building design. Structural Safety, 74, 1-13.
9. Cerchiello, Vania, et al. (2016). Risk of building damage by modeling interferometric time series. 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, 2016.
10. Qian, Y., & Akira, M. (2008). Acceleration‐based damage indicators for building structures using neural network emulators." Structural Control and Health Monitoring: The Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures 15.6 (2008): 901-920.
11. Lei, Y., et al. (2003). Statistical damage detection using time series analysis on a structural health monitoring benchmark problem." Proceedings of the 9th international conference on applications of statistics and probability in civil engineering.
12. Chen, Y., Wang, W., & Chen, Y. (2018). Full-scale shake table tests of the tension-only concentrically braced steel beam-through frame. Journal of Constructional Steel Research, 148, 611-626.
13. Cryer, J. D., & Kellet, N. (1991). Time series analysis. Royal Victorian Institute for the Blind. Tertiary Resource Service.
14. De Lautour, O. R., & Omenzetter, P. (2006). Detection of seismic damage in buildings using time series analysis and pattern recognition. In Proceedings of the 4th World Conference on Structural Control and Monitoring (pp. 11-13).
15. Fuller, W. A. (2009). Introduction to statistical time series (Vol. 428). John Wiley & Sons.
16. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, in Proceedings of the IEEE International Conference Computer Vision & Pattern Recognition (CVPR), Boston, MA, 1–9.
17. Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.
18. He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in Proceedings of the IEEE International Conference on Computer Vision & Pat- tern Recognition (CVPR), Las Vegas, NV, 770–78.