FutureEnergySystems&Technology:Course#7

Resilience-Based Engineering: Cyber-Physical Modeling of Smart Structural Systems for Electrical Equipment and Substations

Comprehensive Project [Total score (without the 4 extra credits) = **80 points**]

Last update: July 6, 2024

Introduction

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 lumped mass models. The dynamic properties of electrical equipment and structures will be studied and analyzed. The corresponding structural response will be obtained through numerical simulations based on these two modeling methods. For developing smart systems, machine learning (ML) and deep learning (DL) techniques are introduced as they play important roles in decision-making. Using data collected from simulation or field, ML and DL can be performed to identify current health conditions of the system. The specifics are divided into three parts.

Part 1: Finite Element Method

Part 2: Dynamics of Structures

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

Remarks for the submission of the solution:

1. When submitting the report, make sure to select the relevant pages for each problem on `Gradescope`.
2. Sometimes explanations and discussions are as important as the numerical results. Please, pay attention to these requirements.
3. Neatness will be rewarded.

Part 1: Finite Element Method (FEM)

The 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, and 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 a combination of them, particularly when they are subjected to complex boundary conditions (BCs) where closed-form solutions are not available.

Let us consider a real problem in an electrical system. The 230 kV high voltage switch, with the porcelain insulator shown in Figure 1 (right), is one of the components in this system, which

Figure 1: 230 kV switch and its experimental research activities.

may experience several types of loads, e.g., gravity (concentrated Corona load at the top, P_v, and complex distributed load due to its *tapered shape*, $p_v(y)$), lateral load (wind or earthquake), P_h, or thermal load (due to electric arc), as shown in Figure 2.

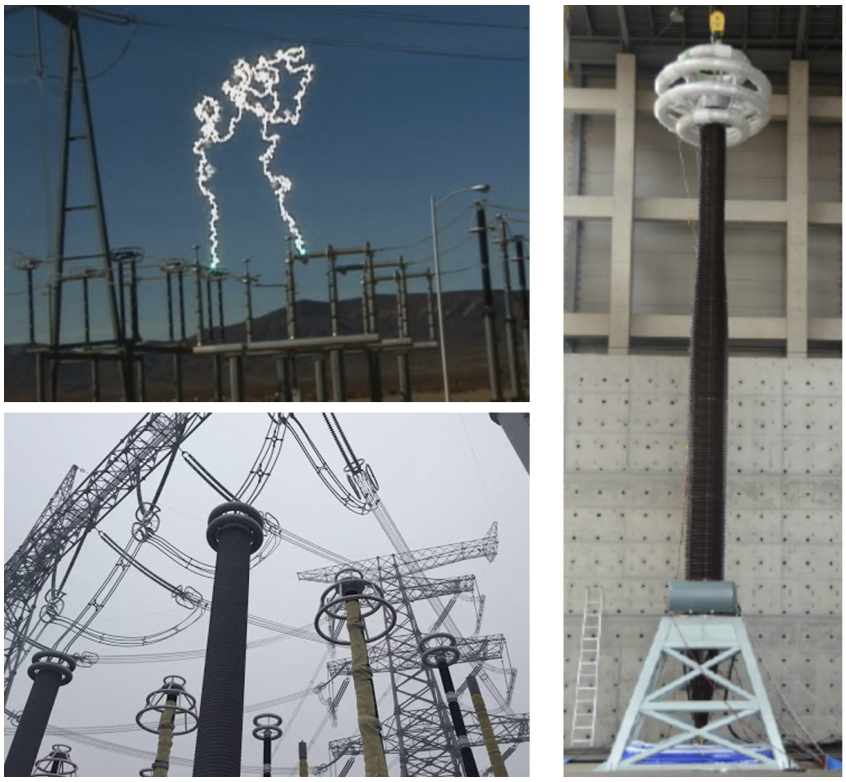

Figure 2: Different types of loading on high voltage electric equipment in substations.

Suppose we can simplify the tapered insulator into a (geometrically) uniform rod or a cantilever with total height, H, and section stiffness, EA, in Figure 3. Subsequently, convert the gravity load (the only considered loading in this part of the project) into two parts, a concentrated load, P_v, due to the Corona fixture, and a distributed load, $p_v(y)$, due to the post self-weight along its height to

account for the varying cross section. The wind load, P_h, will not be considered herein. Thus, we can compute the displacement and axial force for each section of the insulator post along its height as a function of the vertical coordinate, y, due to the effect of the concentrated load P_v together with the distributed load $p_v(y)$.

In terms of the FEM model, the real structure can be converted into a 1-D (linear) simplified model using FEM. The structure is discretized into m elements and $n = m+1$ nodes with numbering shown in Figure 4.

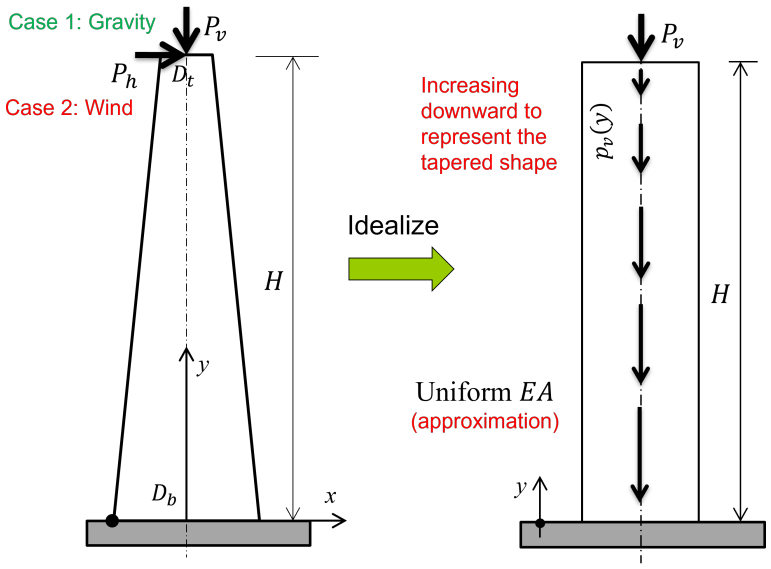

Figure 3: Idealization of a post insulator for subsequent modeling using FEM.

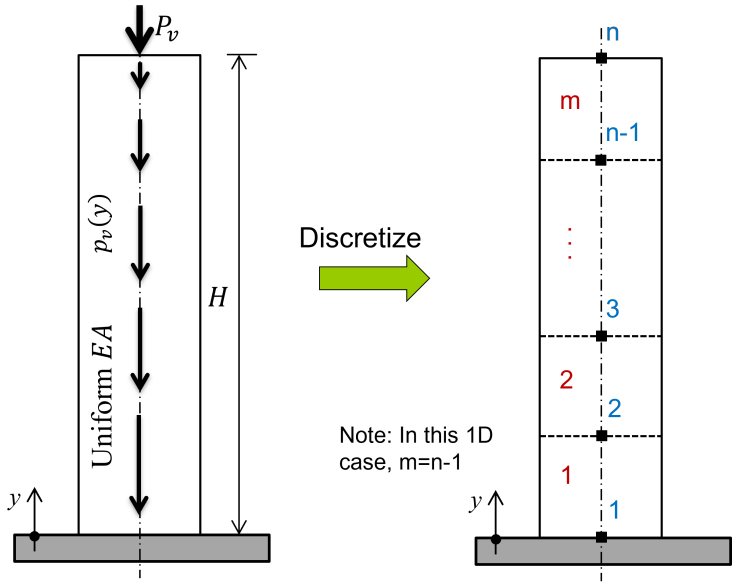

Figure 4: Discretization & numbering of insulator FEM model (Red #: elements; Blue #: nodes).

In Part 1 of your project, assuming the following properties for the insulator:

- Height, $H = 4,000$ mm,

- Uniform cross-sectional area for geometric modeling, $A_{av} \approx 17,671$ mm^2,
- Porcelain material modulus of elasticity, $E = 70,000$ MPa (Note: 1 MPa = 1 N/mm^2),
- The loading function, $p_v(y)$, shown in Figure 4, is only assumed based on the *linearly* varying cross-sectional area, $A(y)$, (not the linearly varying diameter, $D(y)$, for simplicity) between the top section with area A_t corresponding to a diameter D_t, and the bottom section with area A_b corresponding to a diameter D_b, as shown in Figure 3,
- Top diameter, $D_t = 100$ mm,
- Bottom diameter, $D_b = 200$ mm,
- Porcelain equivalent material (accounting for attached heavier material, e.g., metallic flanges) unit weight, $\gamma = 3.5 \times 10^{-5}$ N/mm^3. It is noted that the formulated $p_v(h)$ should produce a negative value where the negative sign indicates that the direction of gravity load is opposite to the coordinate system, i.e., positive y direction points upward while the gravity load points downward,
- Additional Corona fixture as a concentrated load at the top, $P_v = -4,000$ N (again, notice the negative sign), and
- Consider at least five 1-D elements that are aligned along the height, i.e., $m \geq 5$.

To reiterate, when considering the geometric shape (e.g., formulate $[k_e]$ in Milestone 1), assume a constant cross-sectional area A_{av} over the height. However, when considering the distributed gravity load $p_v(y)$ (e.g., formulate $\{p_{0_e}\}$ in Milestone 1), assume a (linearly) varying value over the height caused by the linear variation of the cross-sectional area (You may notice that in Figure 3, diameter $D(y)$ is a linearly varying value over the height, but to simplify the problem here, we are assuming a linear cross-sectional area $A(y)$. You will need to calculate the top and bottom cross-sectional areas A_t and A_b, and conduct linear interpolation in between).

Milestone 1 [25 pts]

In this part of the project, the students are expected to analyze the mechanical deformation of the insulator structure using FEM. For your convenience, the units for the given values above are consistent. Therefore, in the calculation, you do not need to be concerned with the conversion of units. It is required to conduct the following:

1. (5 pts) Formulate the $[k_e]$ and $\{p_{0_e}\}$ for each element. Hint: the distributed gravity load $p_v(y)$ affects every element and its corresponding end nodes, while the concentrated load P_v only affects the top node n. A brief discussion on how to handle concentrated loads is included in slide 34 of the lectures where the easiest way is to add the concentrated load directly to the corresponding nodal degree-of-freedom (DOF).
2. (6 pts) Formulate the $[K]$ and $\{P_0\}$ for the structure. Hint: the dimensions of $[K]$ and $\{P_0\}$ should be $n \times n$ and $n \times 1$, respectively, where n is the number of nodes. In your project submission, you are free to use hand calculations, or design a program with n self-defined or fixed to a number such that the number of elements $m \geq 5$ as specified above. In other words, you can choose m as you wish as long as it is ≥ 5.

3. (8 pts) Obtain the nodal displacements (for nodes 1 to n). Notice that node 1 is fixed to the ground. In FEM, such constraints (together with the concentrated load, P_v at node n) constitute what is called the **boundary conditions** (BCs). In this problem, a very simple trick (as discussed in the lectures) to handle these BCs is to remove the row and column corresponding to node 1 from the $[K]$ matrix and $\{U\}$ & $\{P_0\}$ vectors, i.e., obtain $[K]_{m\times m}$, $\{U\}_{m\times 1}$ & $\{P_0\}_{m\times 1}$ and directly apply P_v at node n with the proper sign convention. Moreover, notice that for this problem, you need to invert the $[K]$ matrix, i.e., calculate $[K]^{-1}$. You are free to use any tool to do the calculation. However, a template is given in the Jupyter Notebook `Course7.ipynb`, which uses a common Python library `numpy` to perfrom the matrix inverse operation and multiplication. The template is meant to help you with matrix calculations, and you are not obliged to use it.

4. (3 pts) Comment on the obtained maximum vertical displacement (expected to be at node n, i.e., Δ_n) from a deformation safety point of view of the attached conductor cables to the top of the insulator post, assuming the tolerance for these conductor cables should not move vertically (under gravity loads only) more than 0.015 mm.

5. (3 pts) If the safety condition above is not satisfied, without doing any further calculations, what would you suggest to modify the post insulator design and satisfy the safety requirement.

6. Extra Credit #1: Vary the number m with at least 3 values, e.g., $m = 2, 4, 6$ and comment on the conversion of the numerical solution with the increase of the number of elements in terms of Δ_n.

7. Extra Credit #2: Recall that when considering the geometric shape (e.g., formulate $[k_e]$ in requirement #1), we assumed a constant cross-sectional area A_{av} over the height. Now, consider the linear variation of the post insulator cross-sectional area, $A(y)$, to be approximated on the element level, not only for the self-weight calculations (may need to be revised from requirements #1 & #2 above) but also for the element discretization and the final solution. Report the obtained nodal displacements and comment on the comparisons with the findings in requirement #3.

Once completing Milestone 1, you have the necessary key information to simulate the static response, i.e., displacement of the structure. If the numerical model is well-established, these structural responses are close to the real situation under the same loading and boundary conditions. Thus, with such a tool we can obtain as much data *spatially* as needed, which can be used for data-driven structural health monitoring (SHM) with machine learning, Part 3.

Part 2: Dynamics of Structures

In the first part of this project, we have used 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 project, a supporting structure with multiple insulators (Figure 5) will be analyzed for its dynamic properties. The structure can be reasonably idealized in the lateral direction as a single degree-of-freedom (SDOF) system. This type of idealization is applied to many energy systems such as the photovoltaic (PV) system in Figure 6.

According to the CIEE (California Institute for Energy and Environment, `https://uc-ciee.org/`) 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

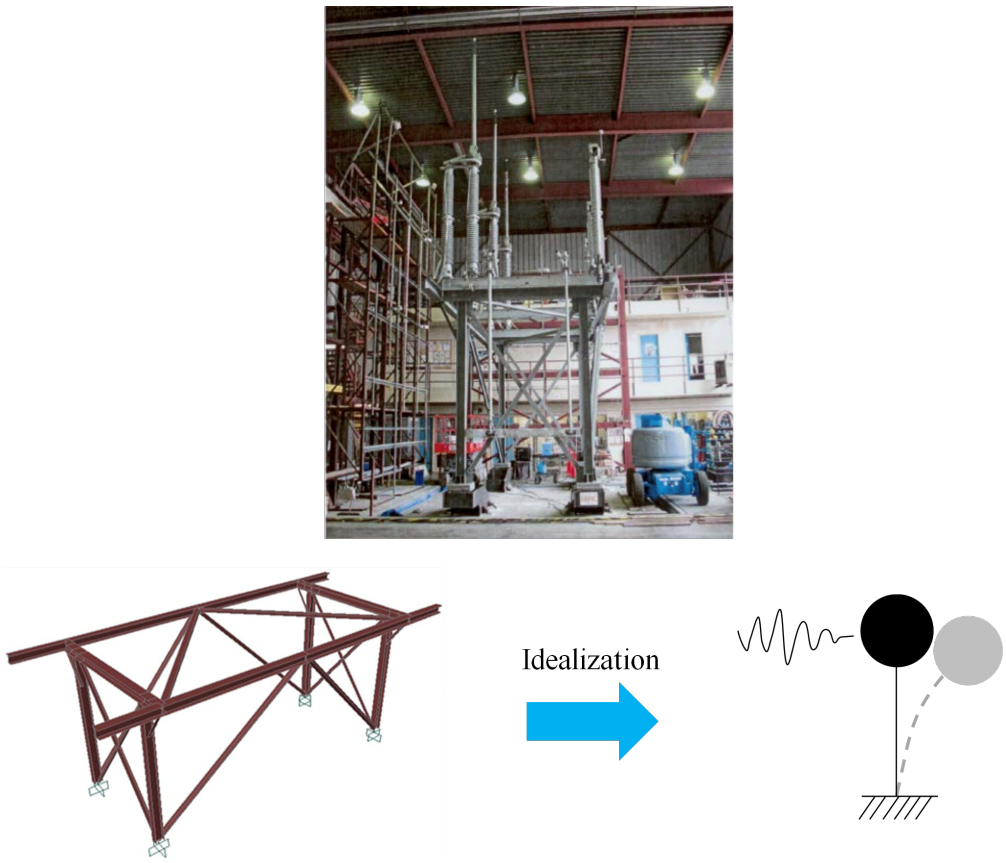

Figure 5: Idealizing the support structure of a 230 kV switch as a SDOF.

studied in this part of the capstone project, Figure 5. In theory, the design of such structures requires detailed modeling of the structural systems using commercial software. However, this process would be time-consuming and requires large computational power. In structural engineering practice, a **lumped-mass model** is frequently used to conduct a preliminary analysis. A lumped-mass model simplifies the distributed mass over the height of the building or other structures into a discrete concentrated mass, and the structural supporting system is simplified as a spring (and dashpot) that connects the lumped mass to the supporting ground. Other simplifying assumptions of the analytical model includes:

- The lumped mass is rigid, i.e., only the spring (and dashpot) can deform. In other words, the beams of the structure are assumed infinitely stiff in flexure (bending) and axially inextensible.
- Only the horizontal displacement of the lumped mass is considered. Moreover, the spring is deforming elastically (i.e., Hooke's law is applicable) in the horizontal direction, i.e., the lateral resisting force in the spring is proportional to the relative horizontal displacement between the lumped mass and the support. The columns (represented by the spring) are axially inextensible, so the mass does not translate vertically.
- Only 1D analysis is conducted where the horizontal translation in the out-of-plane direction, the vertical translation, and all rotational DOFs are not considered.

Since the support structure is a one-story steel frame, and only the horizontal translation is considered, it can be idealized as a SDOF lumped mass system with equivalent stiffness and mass

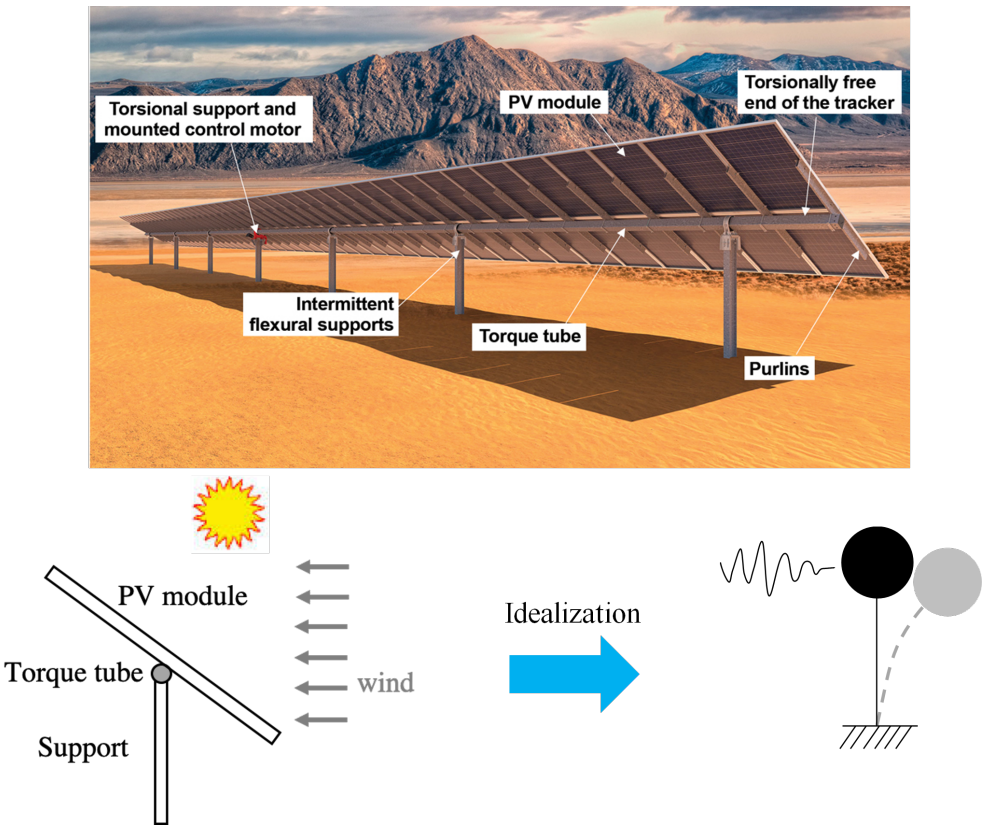

Figure 6: Idealizing a PV panels system as a SDOF.

making use of the above assumptions, as shown in Figure 5. According to Mosalam et al. (2012), the support structure is fixed at the base, and comparisons between different parameters, e.g., loading magnitude and damping ratios, are to be discussed. These factors influencing the basic dynamic properties of the SDOF system will be investigated herein.

From the lectures, we compute the natural frequency with damping for a SDOF system, ω_d, as

$$\omega_d = \omega_n \sqrt{1-\zeta^2} = \sqrt{(k/m)}\sqrt{1-\zeta^2}, \tag{1}$$

where, ω_n is the natural frequency without damping, k is the stiffness of the structure, m is the mass of the structure, and ζ is the damping ratio. Moreover, the relationship between the vibration period T [sec] and frequency ω [rad/sec] is

$$T = 2\pi/\omega. \tag{2}$$

For our structure, the equivalent stiffness values for the support structure with braces are 36.4×10^6 N/m. Neglecting the weight of the braces, the mass of the support structure can be taken as 8,000 kg. The damping ratio ζ is assumed to be 5% for now, which will be varied later in Milestone 2. Same as Milestone 1, the units of the parameters provided here are consistent for the purpose of your calculations in Milestone 2.

Milestone 2 [15 pts]

In this part of the project, the students are expected to analyze the dynamic properties of the structure, in particular, the displacement responses under dynamic loading. Such basic dynamic

properties of structures could be analyzed using hand calculations. However, it is difficult to handle complex load cases, e.g., a random seismic excitation. Therefore, in most cases, software and packages based on numerical methods for time integration are used to simulate the response of structures. In this part, we will be using the Python code provided to implement time integration. In particular, a small integration time step of 0.01 sec (`dt=0.01`) will be used, in order to obtain accurate results. It is required to conduct the following:

1. (2 pt) Calculate the frequency ω [rad/sec] and T [sec].

2. (2 pts) If you open the notebook for the first time, the default dynamic loading pattern is a sine wave as a function of time t as $p(t) = 44,500\sin(20\pi t)$, where $0 \leq t \leq 0.6$ sec, see the formula marked with `loading`. Report the maximum absolute displacement for $0 \leq t \leq 5$ sec.

3. (2 pts) Plot the relationship between time and displacement. You can plot it in the notebook and save it to your submission.

4. (2 pts) Scale the loading amplitude by a factor of 2, i.e., $p(t) = 89,000\sin(20\pi t)$. Report the maximum absolute displacement. What do you observe? Why does this occur?

5. (2 pts) Scale the loading frequency by a factor of 1/2 and 2, i.e., $p(t) = 44,500\sin(10\pi t)$ and $44,500\sin(40\pi t)$, respectively. Do not forget to change the loading amplitude back to 44,500. Report the maximum absolute displacement. What do you observe? Why does this occur?

6. (2 pts) Now change the damping ratio ζ to 0%. Again, do not forget to change other parameters back to the settings in requirement #2. Report the maximum absolute displacement. What do you observe? Why does this occur? Hint: In theory, ω and T need to be changed as they depend on ζ. However, you might find that their changes are very small. Therefore, you definitely need to change ζ, but you will find changing T is not that important.

7. (3 pts) Only repeat requirements #2 & #3 above but for these cases separately (Again, do not forget to change other parameters back to the settings in requirement #2):

   (a) $\Delta t = 0.02$ sec,

   (b) $u_0 = 0.02$ m,

   (c) $\dot{u}_0 = 0.10$ m/sec, and

   (d) $\beta = 1/4$ for average acceleration instead of the used linear acceleration.

   Briefly, compare all cases and comment on the results.

8. Extra Credit #3: The provided codes are based on Newmark's method. Write another Python program to implement a different time integrator (e.g., central difference method) and report your results for requirement #2.

Once completing Milestone 2, you have the necessary key information to simulate the time series data (dynamic response), i.e., displacement of the structure. If the numerical model is well-established, these structural responses are close to the real situation under the same loading and boundary conditions. Thus, with such a tool we can obtain as much data *temporally* as needed, which can be used for data-driven SHM with machine learning, Part 3.

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

As mentioned in the lectures, the vision-based approach is an important and well-studied direction in nowadays structural health monitoring (SHM). The objective of vision-based SHM is to detect vision patterns, e.g., cracking, loss of material (i.e., spalling of concrete cover in reinforced concrete structures), and buckling, to be able to extract information related to damage, i.e., damage level, damage type, etc., through images and/or videos. Usually, this work is performed by humans or some automated algorithm, but there exist many drawbacks for such approaches, e.g., tedious and repetitive inspection work for humans and inaccurate detection results for automated algorithms. In this data explosion epoch, artificial intelligence (AI) and machine learning (ML) technologies are developing rapidly, especially in the 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 able to perform labor-intensive repetitive tasks and also learn from past experiences of domain experts 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-the-art DL technologies in vision-based SHM applications and evaluate their potential benefits, especially in detecting health conditions of critical infrastructure such as electrical equipment and substation systems.

Electrical equipment and substation systems require continuous 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 such 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 extreme event. Slow responses to damage can cause life safety issues, long down times, power outage, and financial losses, i.e., slow recovery and a less resilient system in the face of extreme events.

Transmission towers are an essential part of the electrical grid where damage must be detected and addressed quickly. An automatic failure detection of transmission towers can save communities time and potential danger of traveling out to the field to investigate failure conditions. Thus, DL can be used to greatly improve the process of transmission tower maintenance and repair.

In this part of the project, the students are expected to use the provided image data and adopt DL to train their convolutional neural network (CNN) for the purpose of transmission tower failure detection through images. An example of a CNN framework is shown in Figure 7.

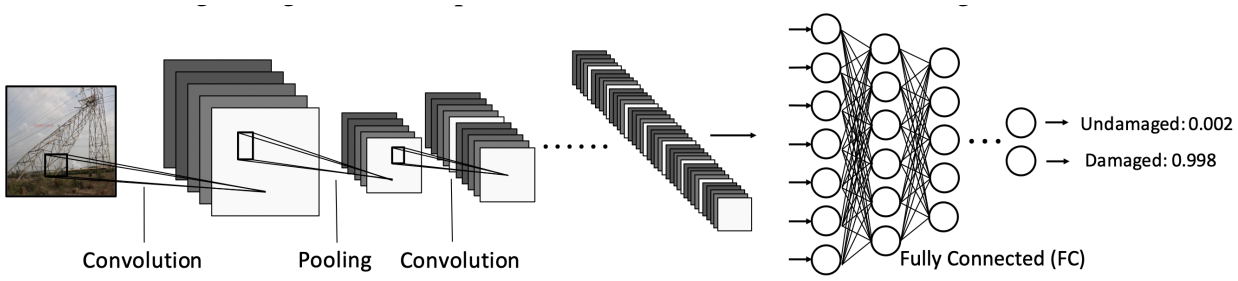

Figure 7: CNN framework used for failure detection of transmission towers.

3.1 Convolutional Neural Network (CNN)

The convolutional neural network (CNN) has been at the heart of the recent spectacular advances in DL. Unlike traditional computer vision and ML approaches, CNN no longer needs hand-designed low-level features, or the so-called feature engineering, where millions of parameters inside a typical network are capable of learning large amounts of medium-to-high level image representations with input data obtained from a pixel matrix (tensor). Another unique characteristic of the 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 Network (ResNet) (He et al., 2016) demonstrated the great performance improvement with substantially increasing the depth. A CNN primarily consists of two parts: (a) image feature extraction using convolution layers and (b) image classification with fully connected (FC) layers, whose input is the output of the last convolutional layer.

3.1.1 Convolutional layer

A convolutional layer extracts the features of an image using filters. A filter is a small-sized matrix (e.g., $3 \times 3 \times$(# 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, as shown in Figure 8(a). These filters (kernels) are learned and tuned during the CNN training process. In practice, zero padding is used before the convolutional operation to adjust the size of images, where p layers of zeros are added on the periphery of the original images, as shown in Figure 8(b). Each convolutional layer consists of multiple filters. These 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-level image features are captured, such as complex geometric patterns.

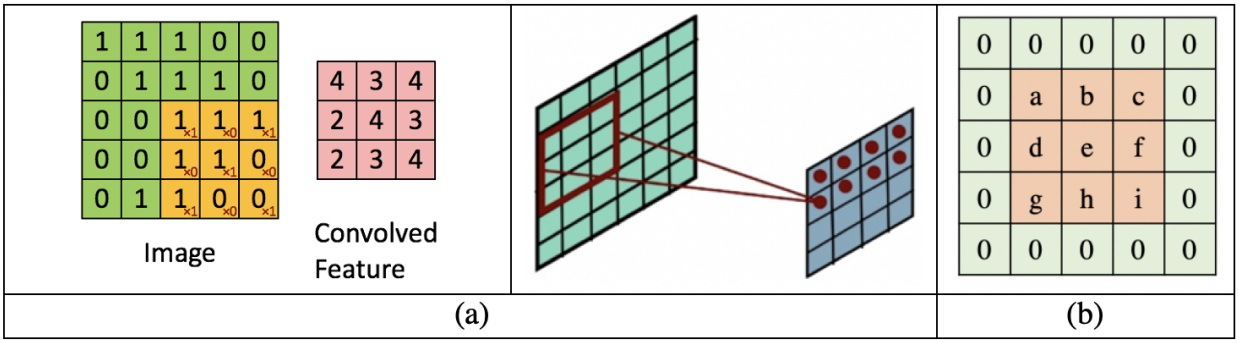

Figure 8: (a) Convolutional operation; (b) Zero padding in the convolutional operation ($p = 1$).

3.1.2 Max Pooling layer

A Max Pooling layer down-samples each internal feature map by traversing its filter across the feature map (similar to the convolutional filters) and extracting the maximum entry value of the feature map's submatrix masked by the filter. This is illustrated in Figure 9. A similar process called "Average Pooling" can be performed to extract the average value of the entries of the feature map's submatrix masked by the filter, instead of the maximum entry value.

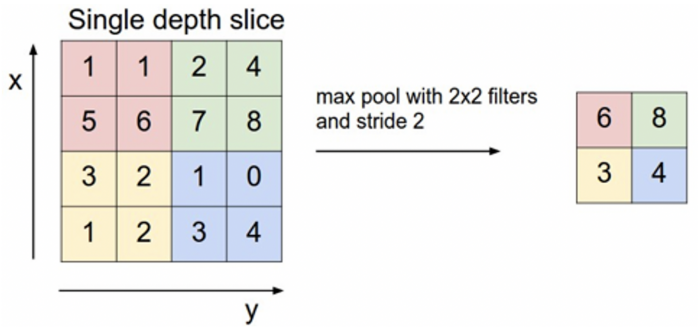

Figure 9: Max Pooling operation.

3.1.3 Fully connected (FC) layer

A fully connected (FC) neural network is another name of neural network (NN), which is added to the end of the convolutional layers in the setting of the CNN, as shown in Figure 7. 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 the 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 are 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).

3.1.4 Activation function: Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is an important activation function at the convolutional layers. Its operation is simple: $f(x) = \max(0, x)$, as shown in Figure 10(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 LeakyReLU which does not ignore negative inputs, Figure 10(b).

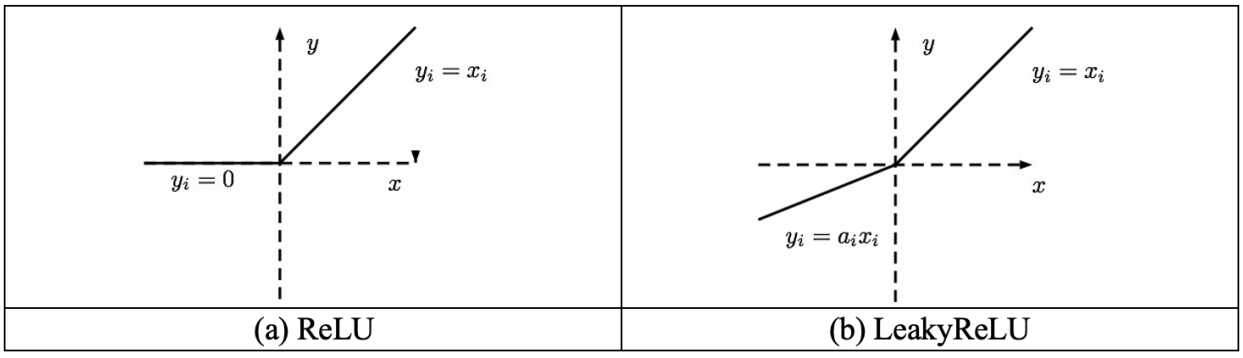

Figure 10: Activation functions: (a) ReLU; (b) LeakyReLU (a_i is a small constant, e.g., 0.01).

Milestone 3 [15 pts]

In this milestone, the students are expected to understand the mechanisms of the CNN filters. During the lectures, the students have learned the mechanism of edge detectors using CNN filters. For images, the higher-level features come from the composition of the lower-level features. Pixel-level features are combined into *edges* using edge detectors. Then, edges are combined into *motifs*, which are combined into *parts*, and finally parts are combined into *objects*, refer to Figure 11.

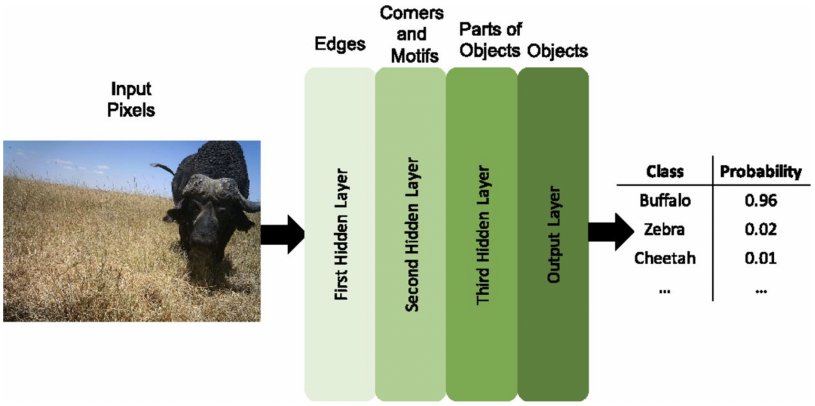

Figure 11: Raw pixels (input layer) processed to detect edges (1st hidden layer), then corners & textures (2nd hidden layer), then object parts (3rd hidden layer), and so on if there are more layers, until the final predictions (output layer).

The students are expected to answer the following questions:

1. (3 pts) Design a filter with size 3×3 such that horizontally oriented edges in a grey scale image are detected.

2. (3 pts) Design a filter with size 3×3 such that the diagonally oriented edges (specifically, edges with inclined angle of $45°$, from the **upper left** to the **lower right**) are detected.

3. (5 pts) Suppose the original grey scale image has dimension $m \times n$. A filter with size $f \times f$ is applied on the image. A padding size of p and a stride of s are adopted in both dimensions of the image. Derive the expression for the dimension of new image $m' \times n'$, in terms of m, n, f, p & s, after the convolutional operation. You might need the floor operation (i.e., round DOWN to the nearest integer[1]) in your final result. The reason is that in most practices, the filters are fully contained within the images, and they cannot go beyond the images (see Figure 12 for a demonstration).

4. (4 pts) In some layers, we want to keep the dimension consistent (i.e., equal) between the "old" image and "new" image after the convolutional operation. Suppose the filter size $f = 5$, and the stride $s = 1$, what should be the padding size p to keep the dimension consistent?

In practice, the filters are not designed manually. Instead, the values of the filters are trained using *back-propagation*. In the next section, a simple CNN model will be trained to classify between the *damaged* state and the *undamaged* state of transmission towers using the package `Keras` (`https://keras.io/about/`).

[1]The *floor function*, $\lfloor x \rfloor$, takes a real number x, and gives the greatest integer $\leq x$, e.g., $\lfloor 2.7 \rfloor = 2$.

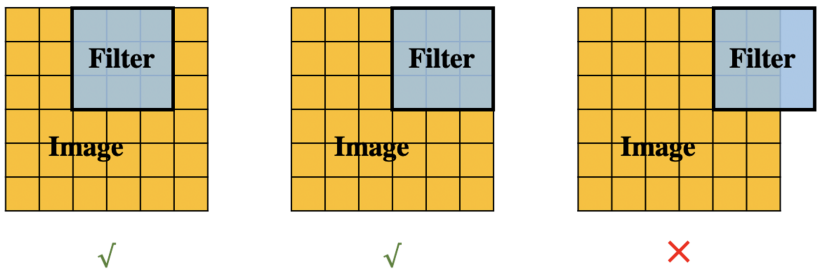

Figure 12: The filters should be fully contained within images (left & middle) rather than go beyond the image boundary (right).

3.2 Image Dataset for Transmission Tower

If a CNN model and DL algorithm build the skeleton of the entire project, the data and datasets are the blood and muscle to make everything runnable. Unfortunately, there is no open-source well-labeled and well-organized image dataset related to the topic of this part of the project, i.e., using DL in image-based detection in SHM for electrical equipment. Thus, we collected images from the Internet, preprocessed them and finalized a small dataset for exploration purposes of this part of the project. This dataset contains about 400 images of transmission towers with both undamaged and damaged states, Figures 13(a) and (b). The total dataset is split into two sets: training set & validation (test) set. The DL model you will develop is to be trained on the training set and validated on the test set. The amount of data in each set and category are listed in Table 1. For simplicity, we treat the ratio of undamaged & damaged states as $1:1$. Therefore, the model should have an accuracy over 50%, which is the random guess of this binary classification problem on a balanced dataset.

Table 1: Number of images in the training and validation (test) sets.

Set	Undamaged state	Damaged state
Training	181	181
Validation (Test)	23	23

In this part of the project, we provide the starter code in the Jupyter Notebook `Course7.ipynb`. The students are expected to be familiar with the basics in deep learning (DL), i.e., convolutional filter, activation function, mini batch, etc. The students will implement the code to realize the transmission tower failure detection via images with an acceptable accuracy. To speed up the training, we re-scaled the images to 28×28 pixels and 224×224 pixels and simplified them to grey scale (one channel instead of 3 RGB colored channels). Please, refer to the folders `train_28`, `test_28`, `train_224` & `test_224`. In this part of the project, you should plan your strategy as follows:

1. Install the required package `Keras` by installing the parent package `Tensorflow`. For this purpose, open the Anaconda Prompt on the Windows, or the Terminal on the MacOS system. Next, run the following code to install the package `Tensorflow`:

```
install tensorflow   or
pip install tensorflow==2.0.0   or
```

(a) Undamaged state indicating no failure (no damage)

(b) Damaged state indicating failure (damage)

Figure 13: Sample images in the datasets for both undamaged and damaged categories.

```
pip install tensorflow==2.3.1
```

where the `2.0.0` or `2.3.1` is the package version number. Try to use one of the three commands that does not report error on your machine. This installation would take a while (several minutes on average). You could check that you successfully installed the packages by importing the package in the Jupyter Notebook as follows:

```
import tensorflow
```

2. Design a one-layer CNN architecture based on the following choices and record the architecture in the report:
   - Convolutional layer, i.e., different numbers and sizes of filters;
   - FC layer, i.e., different numbers of neurons and numbers of layers;
   - Pooling layer, i.e., Average Pooling or Max Pooling;
   - Activation function, i.e., Sigmoid, Tanh, ReLU, etc.
3. Tune the CNN you designed above with different settings, i.e., different training epochs, batch sizes, and learning rates.

Milestone 4 [25 pts]

For this milestone, the following is required:

1. (10 pts) Train your CNN network using (`train_28`, `test_28`) following the above strategy. The training process will take at least several minutes. You will need to adjust and tune your model. Your grade will be based on the "test set" accuracy you achieved, and in order to get full credit, the test set accuracy should be equal to or higher than 80%:

$$\text{grade} = \begin{cases} 0, & \text{if accuracy} \leq 50\% \\ \text{linear interpolation}, & 50\% < \text{accuracy} < 80\% \\ 10\,\text{pt}, & \text{accuracy} \geq 80\% \end{cases} \tag{3}$$

   For this part, you will need to submit the Jupyter Notebook, which should include the printed results of accuracy (your printed results will be taken into consideration for grading). Hint: the training process is stochastic, which means even though you use identical model parameters and train twice, the results might be different each time. In order to achieve a higher accuracy, you may choose to adjust the model parameters, or you can retrain the model (but it is generally more effective to tune the model parameters rather than retrain the model).

2. (15 pts) Repeat the above work for the large size dataset (`train_224`, `test_224`) and using a larger CNN (at least 2 layers), and report what you have observed (e.g., comparison between the smaller size dataset and the larger size dataset).

3. Extra Credit #4: Collect similar datasets to those shown in 1 but for another type of power generation/distribution (or water distribution) components or electrical equipment other than transmission towers and repeat items 1 & 2 above using your own datasets. Make sure to indicate the source from which you obtained your datasets.

References

1. 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.

2. Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

3. 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.

4. 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, `https://escholarship.org/uc/item/9hz5h2sd`.