Uncertainty Quantification by Convolutional Neural Network Gaussian Process Regression with Image and Numerical Data

Jianhua Yin[*] and Xiaoping Du[†]

Indiana University - Purdue University Indianapolis, IN, 46202, United States

Uncertainty Quantification (UQ) plays a critical role in engineering analysis and design. Regression is commonly employed to construct surrogate models to replace expensive simulation models for UQ. Classical regression methods suffer from the curse of dimensionality, especially when image data and numerical data coexist, which makes UQ computationally unaffordable. In this work, we propose a Convolutional Neural Network (CNN) based framework, which accommodates both image and numerical data. We first transform numerical data into images and then combine them with existing image data. The combined images are fed to CNN for regression. To obtain the model uncertainty, we integrate CNN with Gaussian Process (GP), which results in the mixed network CNN-GP. The simulation results show that CNN-GP can build accurate surrogate models for UQ with mixed data and that CNN-GP can also provide the uncertainty associated with the model prediction.

I. Nomenclature

A_{retri}	=	function for dimension reduction
G	=	Gaussian Process model
$g, \hat{g}$	=	generic functions
H	=	random field
h	=	the width of converted image
h_c	=	convection coefficient
I	=	moment of inertia
$\mathbf{Im}$	=	combined image
$\mathbf{im}$	=	image data
$\mathbf{im}_x$	=	image data converted from numerical data
k	=	thermal conductivity
M	=	external moment
T	=	temperature
T_a	=	ambient temperature
T_r	=	transfer function of numerical data to image data
t_z	=	plate thickness
$\boldsymbol{u}$	=	normalized numerical data
w	=	the width of image
$\boldsymbol{X}$	=	spatial coordinate
$\boldsymbol{x}$	=	numerical data
y	=	model response or label
$\mathbf{z}$	=	latent space output of convolutional neural network
β	=	width of a single bar
γ	=	width of a gap between bars

[*] Graduate Research Assistant, Department of Mechanical and Energy Engineering.
[†] Professor, Department of Mechanical and Energy Engineering.

ϵ	=	emissivity coefficient of plate surface
θ	=	angular displacement
θ_h	=	hyperparameter of convolutional neural network
λ	=	eigenvalue
μ	=	mean function of a random field
μ_G	=	mean prediction of Gaussian Process
ξ	=	standard normal random variable
ρ	=	correlation
σ	=	Stefan-Boltzmann constant
σ_G	=	prediction uncertainty of Gaussian Process
φ	=	eigen-function
ω	=	angular velocity

II. Introduction

COMPUTER simulation models, derived from physical laws, play a key role in engineering analysis and design. Their typical applications include prediction, sensitivity analysis, uncertainty quantification (UQ), what-if analysis, optimization, design exploration, and systems design, all of which need to run the simulation many times. The models represent sophisticated physical details across wide spatial and time domains and are usually computationally demanding.

Regression, also known as metamodeling, is increasingly used to replace computationally expensive simulation models with cheaper surrogate models or metamodels for UQ [1-5]. A simulation model is run a limited number of times, producing a set of labeled training points, based on which a surrogate model is built. Surrogate models can be built using traditional response surface modeling (RSM) and can also be built with machine learning (ML) methods, such as Gaussian process (GP), support vector machines (SVM), and neural network. However, the computational cost is high when the dimension of the problems is high [6, 7].

In many applications, such as medicine, computational mechanics, material design, additive manufacturing, both image and numerical data coexist. For example, in the severity assessment of stenosis [8], in addition to the patient image data, other numerical data, such as material properties and boundary conditions, are also inputted to computational fluid dynamics (CFD) simulation. Another example is the multidisciplinary optimization design of aircraft wings [9]. The structure or the geometry of the wings can be considered as image data; loading, material properties, and boundary conditions are numerical data. It is impossible or inconvenient to use typical surrogate modeling methods (RSM, GP, and SVM) to accommodate image data. Although the multi-input problem with both image and numerical data can be handled by a multi-input network [10], the implementation is complicated, and the efficiency may not be satisfactory.

Convolutional Neural Network (CNN) is a deep learning method that can deal with the high dimensionality in image input data. It is specifically designed to handle image data. CNN can recognize build-in features directly through different convolution layers without relying on manual feature selection. It has achieved extraordinary successes in computer vision, image recognition, speech recognition, and engineering applications. Although CNN is designed to deal with only image-like data, recent studies [11] have shed light on the new use of CNN with numerical data. These studies convert numerical data into image data and enable a new capability of treating pure numerical data. Inspired by the findings, we develop a new concept to explore possible ways of using CNN for both image and numerical data. We also quantify the uncertainty in the prediction made by CNN.

Our strategy is to convert numerical data into image data, merge the converted image with other image data, and train a surrogate model with CNN. The concept is illustrated by a dynamic MNIST dataset. Taking the advantage of the GP method, which can conveniently provide not only the prediction but also the uncertainty associated with the prediction, we integrate CNN and GP so that the method can deal with high dimensional problems with quantified model (epistemic) uncertainty in the prediction. The method can also potentially account for aleatory uncertainty associated with random model input variables, thereby quantifying the effect of both types of uncertainty.

The rest of the paper is organized as follows. Section III provides the procedure of converting numerical data into image data and merging converted images with existing images. Section IV shows how to integrate GP and CNN to quantify the prediction uncertainty, or epistemic uncertainty. In Section V, we present two examples followed by conclusions Section VI.

III. Convolutional Neural Network with Both Image and Numerical Data

III.A. Overview

A model with both numerical and image inputs is given by

$$y = g(\boldsymbol{x}, \mathbf{im}) \tag{1}$$

where $\boldsymbol{x}$ is a vector of numerical input variables or a 1D array, and $\boldsymbol{x} \in \mathbb{R}^{n_x \times 1}$ with n_x rows and 1 column; $\mathbf{im}$ is the input image, and $\mathbf{im} \in \mathbb{R}^{n_i \times m_i \times c_i}$ with n_i rows, m_i columns, and c_i channels. The model in Eq. (1) is usually computationally expensive, and we build its surrogate model by

$$y = \hat{g}(\boldsymbol{x}, \mathbf{im}) \tag{2}$$

The strategy is to convert the numerical data $\boldsymbol{x}$ into images. Mathematically, it is a task to transform a 1D array into a 3D array. Suppose after the transformation, numerical data in $\boldsymbol{x}$ becomes an image $\mathbf{im}_x$ in a 3D array. Denote the transformation by $T(\cdot)$; namely

$$\boldsymbol{x} = T(\mathbf{im}_x) \tag{3}$$

Then the new image $\mathbf{im}_x$ is merged with image $\mathbf{im}$. Denote the aggregated image by $\mathbf{Im} = (\mathbf{im}_x, \mathbf{im})$. The input is now $\mathbf{Im}$. The surrogate model in Eq. (2) is usually built by CNN, which is a specialized neural network designed for learning images features automatically. It is mostly used for classifications, but it can also be used for regression. A typical CNN consists of three types of layers, convolution, pooling, and fully connected layer.

After the conversion in Eq. (3), CNN can be used without any modifications, producing a surrogate or regression model

$$y = \hat{g}_{CNN}(\mathbf{Im}) \tag{4}$$

Once the surrogate model is built, a prediction can be made for a new input $(\boldsymbol{x}_{\text{new}}, \mathbf{im}_{\text{new}})$, called a test point, as follows

$$y = \hat{g}(\boldsymbol{x}_{new}, \mathbf{im}_{new}) = \hat{g}\left(T\left(\mathbf{im}_{x_{new}}\right), \mathbf{im}_{new}\right) = \hat{g}_{CNN}(\mathbf{Im}_{new}) \tag{5}$$

where $\mathbf{Im}_{\text{new}} = (\mathbf{im}_{x_{\text{new}}}, \mathbf{im}_{\text{new}})$ and $\mathbf{im}_{x_{\text{new}}} = T^{-1}(\boldsymbol{x}_{\text{new}})$.

III.B. Conversion of Numerical Data into Image Data

There are many ways to convert a 1D array of numerical data into a 3D array (an image). Herein we discuss two of them. One is the bar graph, and the other is the grayscale graph. Both ways can represent the measurable features of the numerical data. The dataset of $\boldsymbol{x}$ is first normalized to $[0, 1]$.

For the bar graph [11], there are many possibilities for a given dataset. An example is shown in Fig. 1 with a datapoint $\boldsymbol{u} = (0.2, 0.5, 0.3, 0.5, 0.8)$. The height of the image in pixels is $h = \beta n_x + \gamma(n_x + 1)$, where β is the width of a single bar, γ is the width of a gap between bars. Denote the maximum length of the bar, which is the width of transformed image, by w, the actual heights of the bars are $\boldsymbol{u}w$, where $\boldsymbol{u} = \left(u_1, u_2, \dots, u_{n_x}\right)$ are the normalized numerical data.

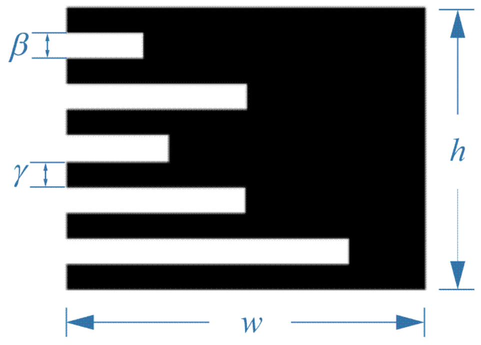

Fig. 1 A converted bar graph.

The width of the image w influences the resolution of numerical features since the number of pixels is an integer. A continuous variable u is discretized into w intervals. The larger is w, the higher is the resolution of numerical features, but the longer time is needed to process the image. Practically we set w to be the maximum value of (n_i, m_i).

$$w = \max(n_i, m_i) \tag{6}$$

If we have higher-dimensional numerical features, it is possible to merge the converged image using the height h. Then, we can set w to be a large value for a highest resolution.

The grayscale method is to transform the normalized data to grayscale images. The normalized value decides how dark or bright the pixels are. The pixels are black when the normalized value is 0 and white when the normalized value is 1. There are several ways to transform the normalized data to images. An example is shown in Fig. 2 using the same datapoint $\boldsymbol{u} = (0.2, 0.5, 0.3, 0.5, 0.8)$. The image height is $h = \beta n_x$ and the width still is w. We convert the numerical data as a band image, where β is band width.

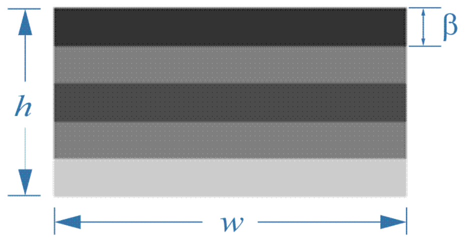

Fig. 2 A converted grayscale image.

The grayscale transformation is insensitive to the width of the image since the numerical feature are embedded in the gray levels but are not impacted by the dimension parameters. However, this transformation method is limited by the gray level. For the commonly used 8-bit color format of grayscale images, the color from black to white is discretized to 256 different shades of color whose range is 0-255. Black is 0 and white is 255. The numerical features are, therefore, discretized to 256 intervals, which means the resolution is fixed. Also, the numerical features are not influenced by the orientation of the image for both transformation methods.

The width of bars or bands for the two conversion methods used in Figs. 1 and 2 are large for a demonstration purpose. Based on our experience, the accuracy of CNN is insensitive to the value of the width, and a smaller value is preferred for a shorter computational cost.

III.C. Combination of Images

After numerical features $\boldsymbol{x}$ are converted into image $\mathbf{im}_x$, where $\mathbf{im}_x \in \mathbb{R}^{h \times w \times c_i}$, we merge it into the existing image $\mathbf{im}$, where $\mathbf{im} \in \mathbb{R}^{n_i \times m_i \times c_i}$. Assuming the $w = \max(n_i, m_i) = m_i$, we merge the two images vertically, which results in an aggregated image $\mathbf{Im} = (\mathbf{im}_x; \mathbf{im})$, where $\mathbf{Im} \in \mathbb{R}^{(n_i + w) \times m_i \times c_i}$. For example, we can merge the transformed images in Figs. 1 and 2 to an existing image as shown in Fig. 3. Fig. 3(a) indicates that the converted image $\mathbf{im}_x \in \mathbb{R}^{22 \times 28 \times 1}$ is merged to the existing image $\mathbf{im} \in \mathbb{R}^{28 \times 28 \times 1}$ vertically, which results in the aggregated image $\mathbf{Im} \in \mathbb{R}^{50 \times 28 \times 1}$. Similarly, as shown in Fig. 3(b), we combine $\mathbf{im}_x \in \mathbb{R}^{15 \times 28 \times 1}$ and $\mathbf{im} \in \mathbb{R}^{28 \times 28 \times 1}$, resulting in $\mathbf{Im} \in \mathbb{R}^{43 \times 28 \times 1}$.

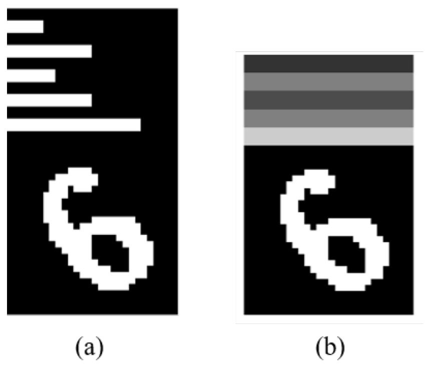

Fig. 3 An example of aggregated images.

Alternatively, the transformed images can also be merged to other sides of the existing image if they are rotated to suitable orientations. The orientations of the transformed images will not have a strong influence on the result of the CNN training.

After the transformed images are merged to the existing images, the aggregated images are loaded to CNN for regression.

IV. Uncertainty Quantification by CNN-GP with Both Image and Numerical Data

The CNN training process is straightforward. Once the CNN structure is determined, the CNN model ($\hat{g}_{CNN}$) is obtained using a group of training data $[(\mathbf{Im}_1, y_1); \ldots; (\mathbf{Im}_n, y_n)]$. Given a new untried image ($\mathbf{Im}^*$), we have the predicted response ($\hat{y}^*$). However, we do not know how accurate the prediction is unless we run the original computational model or a physical experiment, which is expensive. The discrepancy between the predicted response and real response can be estimated by epistemic uncertainty or model uncertainty.

When we run the original simulation model, we assume that the response from such simulation model is the ground truth without epistemic uncertainty. The model inputs, in most applications, however, are random, such as stochastic loading, material properties, and other random parameters. This kind of uncertainty is called aleatory uncertainty or data uncertainty. The purpose of this study to quantify epistemic uncertainty of surrogate models, and we do not consider epistemic uncertainty of simulation models whose surrogate models are to be built. It is possible to accommodate both types of uncertainty if the model uncertainty can be quantified.

To understand the accuracy of CNN, we need to quantify the epistemic uncertainty of the CNN prediction. The central idea is to combine CNN with GP regression. The GP regression is increasingly used in UQ for engineering applications [12-15] due to its ability to quantify epistemic uncertainty. A prediction of the GP model $G(\boldsymbol{x})$ consists of two parts, the prediction mean and the prediction variance, which are denoted by $\mu_G(\boldsymbol{x})$ and $\sigma_G^2(\boldsymbol{x})$, respectively The prediction of the GP model follows a normal distribution $G(\boldsymbol{x}) \sim N\left(\mu_G(\boldsymbol{x}), \sigma_G^2(\boldsymbol{x})\right)$. Therefore, we know the uncertainty of the prediction for a given input point using the GP model. A detailed review of GP can be found in [16].

However, GP regression suffers from the curse of dimensionality. It is very hard or impossible to train a GP model when images exit. Inspired by the existing model uncertainty quantification using GP [17, 18] and the mixed CNN [19], we propose to quantify the uncertainty of the CNN model by combining CNN with GP. As illustrated in Fig. 4, CNN is treated as a supervised dimension reduction process by several folds of convolution, pooling, and other layers. The high dimensional image is reduced to n_z hidden units, which are denoted by $\boldsymbol{z}$ from one of the fully connected layers (FC layer). Once the CNN model is built, we obtain the relationship between $\boldsymbol{z}$ and input variables. Then we use GP to obtain the relationship between response y and $\boldsymbol{z}$. Thus, the combination of CNN and GP yields the estimate of the prediction error by quantifying the epistemic uncertainty at an untried point.

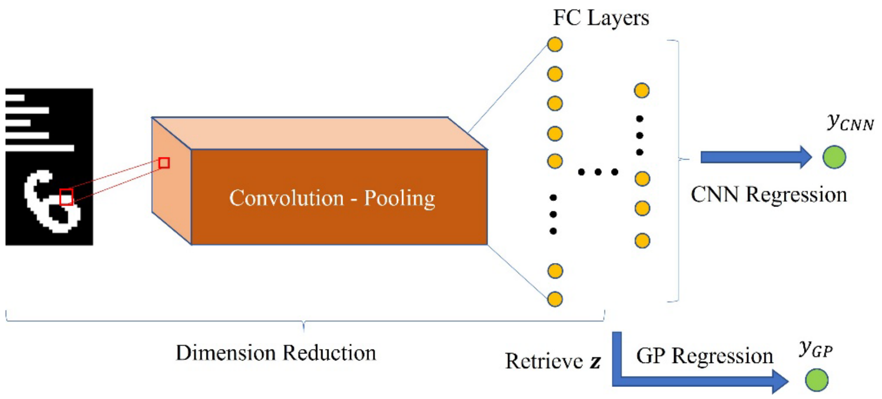

Fig. 4 Illustration of the uncertainty quantification method.

Recall that we use the training data $[(\mathbf{Im}_1, y_1); \ldots; (\mathbf{Im}_n, y_n)]$ to construct the CNN model ($\hat{g}_{CNN}$). Once the model is obtained, we can easily project the image from high dimensional space to a latent space ($\boldsymbol{z}$) by retrieving the hyperparameters of CNN. The projection function is given by

$$\boldsymbol{z} = A_{retri}(\boldsymbol{Im}, \theta_h) \tag{8}$$

where θ_h denotes the hyperparameters of the CNN model. Therefore, the original high dimensional images $[\mathbf{Im}_1; \ldots; \mathbf{Im}_n]$ is projected to the $\boldsymbol{z}$ latent space $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$.

Using the latent variables $(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n)$ and their corresponding labels $(y_1, \dots, y_n)$ as the training data, we construct a GP model, which is denoted by

$$y_{GP} = G(\boldsymbol{z}) = G\big(A_{retri}(\boldsymbol{Im}, \theta_h)\big) \tag{9}$$

where $y_{GP} \sim N\big(\mu_G(\boldsymbol{z}), \sigma_G^2(\boldsymbol{z})\big)$.

Given a new image $\mathbf{Im}^*$, we obtain $\boldsymbol{z}^*$ and y_{GP}^* at this untried point using Eqs. (8) and (9), respectively, where y_{GP}^* is a random variable following $y_{GP}^* \sim N\big(\mu_G(\boldsymbol{z}^*), \sigma_G^2(\boldsymbol{z}^*)\big)$. We then obtain the epistemic uncertainty at the test point.

V. Numerical Simulation

We now use two examples to show the detailed procedures and performance of the proposed method. In the first example, we use the well-known MNIST dataset [20] to test the concept. MNIST is a database containing grayscale images of handwritten digits, which are commonly used for training and testing image processing systems. We change the problem to a dynamics problem so that we relate it with potential engineering applications. The second example is a real engineering example, which involves nonlinear heat transfer. The regression accuracy is evaluated by the relative error. The accuracy of UQ is illustrated by comparing with the result of Monte Carlo Simulation (MCS).

V.A. Example 1: A dynamics problem using MNIST

The size of each image in MNIST is 28×28 pixels, and each digit has an associated label, which is the angle θ that the digit has rotated. We convert the original problem into a dynamics problem. Since the original MNIST dataset contains only image input, we modify it by adding numerical input variables. We assume that each digit is a rigid body and that the position in MNIST is the initial position of the rigid body. We use a digit 6 as an example for demonstration. As shown in Fig. 5, initially the rigid body rotates about its center of mass G at an initial angular velocity of ω_0 (rad/s), and the initial angular displacement of the body with respect to the vertical axis is θ (rad). To rotate the body back to the vertical axis, we apply a moment M (N·m) in the opposite direction of ω_0. The body reaches an angular velocity ω (rad/s) in its final position when $\theta = 0$.

The task is as follows: Given the geometry of the rigid body, its initial position θ, its initial angular velocity ω_0, and the external moment M, we find its final angular velocity ω. The input therefore includes $\mathbf{im} \in \mathbb{R}^{28\times28\times1}$ in MNIST and numerical variables $\boldsymbol{x} = (\omega_0, M)^T$, and the output is $y = \omega$. We therefore have both image and numerical data in the input. The data of ω_0 and M are generated randomly with a uniform distribution in the range of $[0, 1]$.

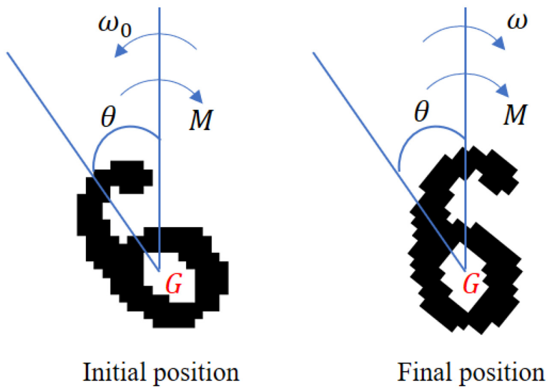

Fig. 5 A dynamics problem.

We now discuss how to generate labels for y. At first, we need to extract a rigid body from an image to calculate the inertia properties of the body. This is the task of segmentation. We define a threshold value for the segmentation. After testing with different threshold values, we found that a threshold of 0.2 is the best to keep the shapes of the extracted bodies smooth. One example of the extracted bodies is shown in Fig. 5.

Assume the rigid body is placed on a smooth horizontal surface. From the law of conservation of energy, we have

$$\tfrac{1}{2} I\omega_0^2 + M\theta = \tfrac{1}{2} I\omega^2 \tag{10}$$

where I is the moment of inertia about the center of mass, $\frac{1}{2}I\omega_0^2$ and $\frac{1}{2}I\omega^2$ are the initial and final kinetic energy, respectively, and $M\theta$ is the work done by the moment. The final angular velocity is obtained by solving Eq. (10).

We next follow the proposed strategy to transform the numerical vector $\boldsymbol{x} = (\omega_0, M)^T$ into image $\mathbf{im}_x$ and then merge $\mathbf{im}_x$ into the extracted rigid body $\mathbf{im}$. For the bar graph, we set $\beta = 1$ and $\gamma = 0$, and $w = 28$; thereby we have $\mathbf{im}_x \in \mathbb{R}^{2\times28\times1}$ and $\mathbf{Im} \in \mathbb{R}^{30\times28\times1}$.

We also use the grayscale method. The band width (β) is 1 and the image width is 28 in pixels. We therefore have $\mathbf{im}_x \in \mathbb{R}^{2\times28\times1}$ and $\mathbf{Im} \in \mathbb{R}^{30\times28\times1}$.

The merged image data ($\mathbf{Im}$) is fed to CNN to perform the regression task. There are 10,000 samples of which half of them are for training and the rest are for validation. We employ a four-fold convolution layers. The CNN structure and parameters are listed in Table 1.

Table 1 CNN model structure and parameters

Layer	Filter size	Filter number	Stride
Convolution layer 1	3 × 3	8	-
Average pooling	2 × 2	-	2
Convolution layer 2	3 × 3	16	-
Average pooling	2 × 2	-	2
Convolution layer 3	3 × 3	32	-
Convolution layer 4	3 × 3	32	-
Fully connected layer 1	8 neurons	-	-
Fully connected layer 2	1 neuron	-	-

The accuracy of the CNN regression with the bar graph transformation are shown in Fig. 6. The scatter plot shows that the predictions against the true values are distributed around the 45° line, which indicates that the proposed method can well handle the mixed data with both numerical and image data. The root-mean-square-error (RMSE) of the validation is 0.67 rad/s.

If a new point is given to the CNN, we do not know how confident we are to the prediction at this point unless we run the original expensive simulation model. In other words, we do not have epistemic uncertainty. It is, therefore, desirable to estimate the model or epistemic uncertainty using the proposed method CNN-GP, where we can obtain the standard deviation of the prediction.

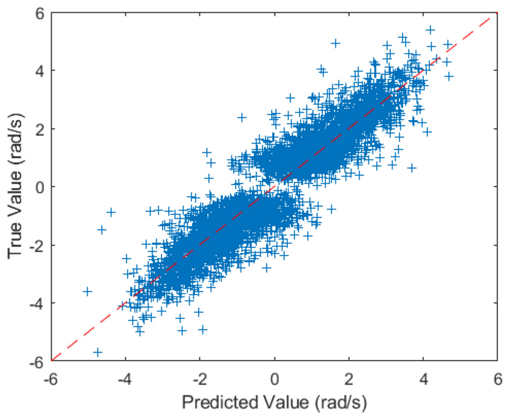

Fig. 6 CNN regression accuracy with the bar transformation.

Following the procedure in Section VI, we retrieve the variables in a latent space ($\boldsymbol{z}$), so the dimensionality of original high-dimension images is reduced. According to the CNN structure in Table 1, there are 8 neurons in the first fully connected layer. We reduce the dimensionality of the images (30 × 28) to 8 variables. Then, a GP model is constructed between $\boldsymbol{z}$ and model response y. Once we have the CNN-GP model, we can easily obtain the mean response and its standard deviation at a new point. Recall that we have 5,000 points for testing, among which we randomly chose 20 points. We use the CNN-GP model and obtain the mean responses and standard deviations of the 20 test points. The results are shown in Fig. 7. A circles represents a mean responses. The length of an error bar gives a 95% confidence interval (CI). Therefore, we know the confidence of the predictions with the epistemic uncertainty at the untried points. The true responses or labels from MNIST are also plotted in Fig. 7.

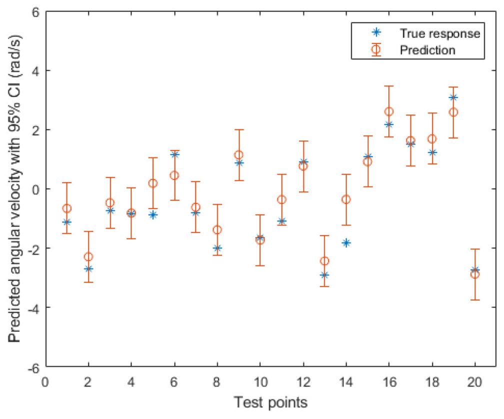

Fig. 7 CNN-GP regression with epistemic uncertainty.

As mentioned previously, we assume that the original computational model is exact without epistemic uncertainty. This implies that the labels from MNIST are exact. As what we have observed, the labels themselves may have some degree of uncertainty, and this uncertainty in labels is neglected in our UQ analysis since there is no way to quantify it. If the distribution of the uncertainty labels was known, we could include it in the UQ analysis.

V.B. Example 2: A nonlinear heat transfer problem

This example is a heat transfer analysis of a thin rectangle plate. The spatial domain is shown in Fig. 8. The temperature is fixed along the left boundary b_1 and is a random variable following a uniform distribution. The other three boundaries are thermal isolation without heat transfer along the boundaries. The temperature at point p_1 is the quantity of interest or the output.

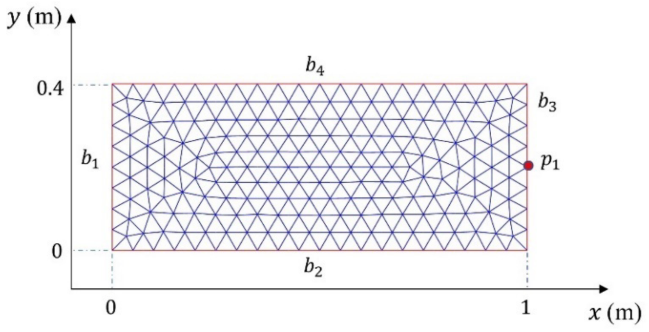

Fig. 8 The spatial domain of the heat transfer problem.

The nonlinear heat transfer is governed by a partial differential equation (PDE) given by

$$-kt_z\nabla^2T + 2h_cT + 2\epsilon\sigma T^4 = 2h_cT_a + 2\epsilon\sigma T_a^4 \tag{11}$$

where k is the thermal conductivity and is an uncertain parameter; t_z is the plate thickness; h_c is the convection coefficient; ϵ is the emissivity of the plate surface; σ is the Stefan-Boltzmann constant; T_a is the ambient temperature. Except for k, the other parameters are all constant. The randomness of k is characterized by a Gaussian random field, which is modeled by the truncated Karhunen-Loeve (K-L) expansion. The K-L expansion is given by

$$H(\boldsymbol{X}) = \mu(\boldsymbol{X}) + \sum_{i=1}^{m}\sqrt{\lambda_i}\varphi_i(\boldsymbol{X})\xi_i \tag{12}$$

where $\mu(\cdot)$ is the mean function of the random field; $\boldsymbol{X}$ is a 2D vector to represent the spatial location; λ_i and φ_i are the eigenvalues and eigenfunctions of the auto-correlation function; ξ_i is a set of independent standard normal random variables; m is the truncation number. In this work, we use the squared exponential kernel as the auto-correlation function. The correlation between two arbitrary points is given by

$$\rho_{i,j} = \exp\left\{-\left(\frac{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|}{\theta}\right)^2\right\} \tag{13}$$

where θ is the correlation length; i and j are two arbitrary points; and $\|\cdot\|$ represents the norm operation.

Once the mesh and random variables are known, we can generate realizations of the random field, which are represented by images. And the boundary condition is given as numerical data. We only include the left boundary into the image since the other boundaries are the same for all realizations of the random field.

We generate 10,000 samples in total, in which 2,000 samples are used for training and the rest are for testing. Four examples are shown in Fig. 9. The dark bars represent different boundary conditions with different gray scale and the other parts are different realizations of the random field. All the labels (the temperature at p_1) are solved by the Stochastic Finite Element Method (SFEM). The CNN structure is given in Table 2.

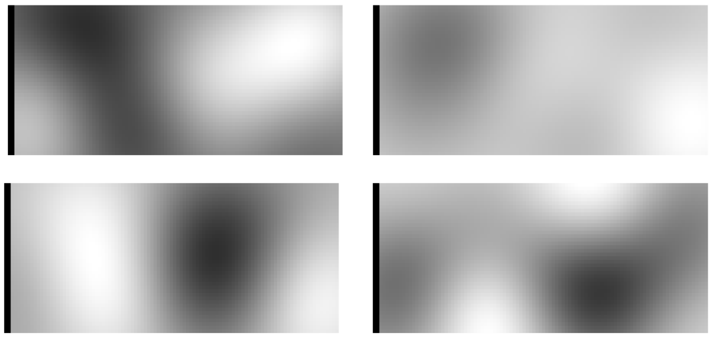

Fig. 9 Examples of merged image data.

Table 2 CNN model structure and parameters for example 2

Layer	Filter size	Filter number	Stride
Convolution layer 1	5×5	32	-
Average pooling	2×2	-	2
Convolution layer 2	3×3	28	-
Average pooling	2×2	-	2
Convolution layer 3	3×3	24	-
Convolution layer 4	3×3	16	-
Convolution layer 5	2×2	8	-
Fully connected layer 1	8 neurons	-	-
Fully connected layer 2	1 neuron	-	-

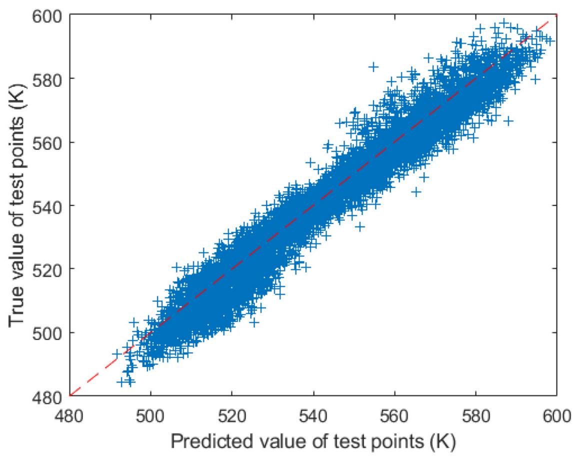

Fig. 10 True label versus the predicted label by CNN.

The regression result of CNN is shown in Fig. 10. The predictions and true labels scatter around the 45° line compactly for the test points. Recall that a mixed network CNN-GP is used to quantify the prediction uncertainty. The GP model is constructed with respect to the output of the first fully connected layer of the CNN. Since the layer has 8 neurons, the input of the GP model is eight dimensional. In other words, the CNN model can be assumed as a supervised dimension reduction process and the dimension of latent space is eight. After the CNN-GP model is obtained, we have the mean prediction with uncertainty information for training points and test points. Similarly, we randomly evaluate 20 points in the test dataset using the CNN-GP model. We obtain the epistemic uncertainty at the untried points as shown in Fig. 11.

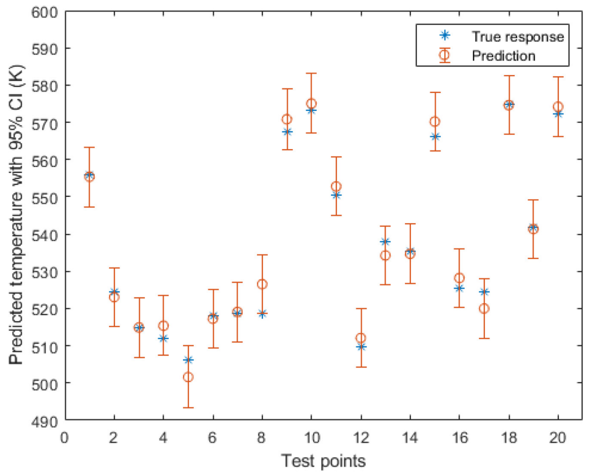

Fig. 11 Mean predictions and uncertainty of 20 testing points.

We compare the accuracy of different methods using relative errors between predictions and true responses, which are given in Table 3. In general, CNN-GP is more accurate than CNN. The average error and maximum error of CNN-GP are reduced compared with CNN. Besides, the standard deviations of the error decrease as well for both training and testing points. Although CNN has slightly worse accuracy than CNN-GP, its accuracy is acceptable.

Table 3 Regression accuracy of CNN and CNN-GP

Methods	Samples	Error (%)		
		Average	Max	Std
CNN	Training points	0.66	3.45	0.53
	Test points	0.70	4.67	0.54
GP- CNN	Training points	0.60	2.95	0.50
	Test points	0.62	4.40	0.51

We also compare the results of uncertainty propagation due to aleatoric uncertainty (the random filed k) for different methods using the 8,000 test points as shown in Table 4. MCS is the ground truth because we use SFEM to obtain true labels. It is shown that CNN-GP is slightly more accurate than CNN and has a subtle difference compared with MCS. CNN is also accurate enough with the errors of mean and standard deviation being 0.14% and 1.51%, respectively.

Table 4 Results of uncertainty propagation

Methods	Mean	Error (%)	Std	Error (%)
MCS	543.33	-	23.24	-
CNN	542.57	0.14	23.59	1.51
CNN-GP	543.38	0.01	22.90	1.46

VI. Conclusion

This study investigates a Convolutional Neural Network Gaussian Process method for mixed numerical and image. It also quantifies the uncertainty of the regression model generated by the method. The strategy is to first transform numerical data into image data and then merge the converted images to existing images. Thus, the model input becomes pure images that can then be fed into CNN for regression without any further modifications. The quantification of the model uncertainty is fulfilled by the integration of CNN and Gaussian Process regression. As a result, the output of the regression model contains the prediction and its standard deviation, which provides an estimate of the model error. With this distinctive feature, the method can be potentially applied to engineering problems where numerical and image data co-exist and can be employed for non-deterministic analysis and design where aleatory uncertainty exists in the model input. Potential applications include reliability-based design, robust design, and risk-based design. The proposed method, however, needs more computational time since it performs additional Gaussian Process regression after CNN. The additional computational time is not significant compared to that of CNN because the Gaussian regression is performed in a space with much fewer dimensions. The UQ results from the proposed method are different from those from the Gaussian process regression since the two methods have different input spaces. We will further investigate how to account for both epistemic and aleatory uncertainty.

Acknowledgments

The support from the National Science Foundation under Grant No. 1923799 is acknowledged.

References

[1] Wojtkiewicz, S., Eldred, M., Field, J., R, Urbina, A., and Red-Horse, J. "Uncertainty quantification in large computational engineering models," *19th AIAA Applied Aerodynamics Conference*. 2001, p. 1455.

[2] Roy, C., and Oberkampf, W. "A complete framework for verification, validation, and uncertainty quantification in scientific computing," *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. 2010, p. 124.

[3] Acar, P., and Sundararaghavan, V. "Uncertainty quantification of microstructural properties due to experimental variations," *AIAA Journal* Vol. 55, No. 8, 2017, pp. 2824-2832.

[4] Bae, H.-R., Grandhi, R. V., and Canfield, R. A. "Uncertainty quantification of structural response using evidence theory," *AIAA journal* Vol. 41, No. 10, 2003, pp. 2062-2068.

[5] Palar, P. S., Zakaria, K., Zuhal, L. R., Shimoyama, K., and Liem, R. P. "Gaussian Processes and Support Vector Regression for Uncertainty Quantification in Aerodynamics," *AIAA Scitech 2021 Forum*. 2021, p. 0181.

[6] Yin, J., and Du, X. "High-Dimensional Reliability Method Accounting for Important and Unimportant Input Variables," *Journal of Mechanical Design*, 2021, pp. 1-28.

[7] Yin, J., and Du, X. "Active learning with generalized sliced inverse regression for high-dimensional reliability analysis," *Structural Safety* Vol. 94, 2022, p. 102151.

[8] Yu, H. W., Khan, M., Wu, H., Du, X., and Sawchuk, A. P. "Uncertainty Quantification of Outflow Boundary Conditions on Non-Invasive Pressure Quantification in Aortorenal Artery System," *Summer Biomechanics, Bioengineering, and Biotransport Conference*. Seven Springs, PA, 2019.

[9] Lambe, A. B., and Martins, J. R. "Matrix-free aerostructural optimization of aircraft wings," *Structural and Multidisciplinary Optimization* Vol. 53, No. 3, 2016, pp. 589-603.

[10] Rosebrock, A. "Keras: Multiple Inputs and Mixed Data." https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/, last modified February 4, 2019, accessed February 16, 2021.

[11] Sharma, A., Vans, E., Shigemizu, D., Boroevich, K. A., and Tsunoda, T. "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific reports* Vol. 9, No. 1, 2019, pp. 1-7.

[12] Rajaram, D., Puranik, T. G., Renganathan, A., Sung, W. J., Pinon-Fischer, O. J., Mavris, D. N., and Ramamurthy, A. "Deep Gaussian process enabled surrogate models for aerodynamic flows," *AIAA Scitech 2020 Forum*. 2020, p. 1640.

[13] Satria Palar, P., Rizki Zuhal, L., and Shimoyama, K. "Gaussian Process Surrogate Model with Composite Kernel Learning for Engineering Design," *AIAA journal* Vol. 58, No. 4, 2020, pp. 1864-1880.

[14] Wei, X., and Du, X. "Robustness Metric for Robust Design Optimization Under Time- and Space-Dependent Uncertainty Through Metamodeling," *Journal of Mechanical Design* Vol. 142, No. 3, 2019.

[15] Jung, Y., Kang, K., Cho, H., and Lee, I. "Confidence-Based Design Optimization for a More Conservative Optimum Under Surrogate Model Uncertainty Caused by Gaussian Process," *Journal of Mechanical Design* Vol. 143, No. 9, 2021, p. 091701.

[16] Rasmussen, C. E. "Gaussian Processes in Machine Learning," *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 63-71.

[17] Arendt, P. D., Apley, D. W., and Chen, W. "Quantification of model uncertainty: Calibration, model discrepancy, and identifiability," *Journal of Mechanical Design* Vol. 134, No. 10, 2012.
[18] Bae, S., Park, C., and Kim, N. H. "Estimating Effect of Additional Sample on Uncertainty Reduction in Reliability Analysis Using Gaussian Process," *Journal of Mechanical Design* Vol. 142, No. 11, 2020.
[19] Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. "Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks," *arXiv preprint arXiv:1707.02476*, 2017.
[20] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* Vol. 86, No. 11, 1998, pp. 2278-2324.