

## RESEARCH ARTICLE

# A Generative Verification Framework on Statistical Stability for Data-Driven Controllers

**SUWON LEE** 

Department of Future Mobility, Kookmin University, Seoul 02707, Republic of Korea

e-mail: suwon.lee@kookmin.ac.kr

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant RS-2022-00165635.

**ABSTRACT** This study proposes a novel framework for evaluating the stability of data-driven controllers and the concept of statistical stability. The proposed framework can be used when it is challenging to show stability through conventional control theory. The novelty of this paper lies in that it provides a method for scientifically analyzing the stability of data-driven controllers, thereby improving the quality of data-driven controllers. The proposed framework consists of three parts: the generative model, controller optimizer, and verification model. A variational autoencoder is used to classify and randomly generate data, and the generated data are used to train the controller. A support vector machine is used to classify areas where the controller is statistically stable. The statistical stability of an optimal controller designed using a deep neural network structure is analyzed using the proposed framework.


**INDEX TERMS** Data-driven analysis, deep neural network, nonlinear controller, nonlinear system verification, optimal control, performance analysis, statistical stability, variational autoencoder.

## I. INTRODUCTION

A mathematical relationship between the control input and system output must be derived to design controllers for dynamic systems. Next, based on this relationship, a controller that can achieve the desired control performance and control goals is designed. Stability is an essential requirement for all control systems. In addition to stability, control systems must meet requirements including tracking given reference signals and suppressing disturbances and noise [1]. A suitable controller must ensure the stability of the control system and be able to follow a given reference signal with small error. For linear time-invariant systems, analysis of the stability can be conducted algebraically [2]. For nonlinear systems, stability analysis can be more challenging and differs for the system's dynamic characteristics [3]. Several methodologies exist for analyzing the stability of nonlinear systems, including locally linearizing the system and using bifurcation theory [4]. The methodologies require analyzing the system's differential equations and often conducting

input-output analysis. Because various types of nonlinearities exist, different mathematical tools are required for stability analysis [5].

Another objective of the control system is to achieve optimal performance. The optimal control of dynamic systems is often conducted using the calculus of variations and solving the two-point boundary value problem (TPBVP) to obtain the optimal trajectory [6]. This approach is also known as trajectory optimization. The benefit of trajectory optimization is that the optimal trajectories of the state and control input can be calculated, and the boundary conditions and path constraints can be considered [7]. However, in many cases, analytically solving the TPBVP is very challenging, and numerical methods are often utilized to solve the TPBVP [8]. Therefore, the process often requires high computational power and is time-consuming. Another limitation of the trajectory optimization approach is that the method uses open-loop control. For a linear system, the optimal controller design is based on solving system matrix equations assuming full knowledge of the system. The optimal controller is designed offline by solving Hamilton-Jacobi-Bellman (HJB) equations such as the Riccati equations [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Emanuele Crisostomi .

Therefore, although the controller is in the shape of the feedback control, it is an open-loop control. In practical applications, it is often important to be able to design controllers online without having complete knowledge of the plant dynamics. Modeling uncertainties may exist, such as parameter inaccuracy, unmodeled dynamics, and disturbances.

The feedback control approaches are used to compensate for the limitations of the open-loop control approach. In the case of linear systems, proportional-integral-derivative (PID) controllers can be designed for transfer function models. For nonlinear systems, various types of nonlinear controllers such as sliding mode control (SMC) [10], feedback linearization (FL) [11], and backstepping control [12] can be designed and applied depending on the characteristics of the control system [13]. However, controllers designed using conventional controller design methodologies have a limitation in that they must be designed to have a specific structure to ensure stability and optimality from a mathematical perspective. For example, LQRs can only consider the cost function of the quadratic form, and in the case of backstepping control, it can only be applied if the control system is in cascade form. To ensure stability using Lyapunov stability theory, it is necessary to find a suitable Lyapunov function, and a general methodology for finding the Lyapunov function does not exist. Thus, the stable region is often conservatively presented using quadratic functions [14]. A conservatively designed Lyapunov function may unnecessarily limit the performance of the nonlinear controller.

In this study, a new methodology that can overcome the limitations of the conventional controller design methodologies described above is proposed, and the statistical stability of the control system is analyzed. To this end, a generative verification model is proposed, and the stability and performance of data-driven control systems are statistically verified. Because deep neural networks are available to approximate and express arbitrary functions, feedback controllers consisting of deep neural networks can be designed by designing deep neural networks as a function for the state variable [15]. Furthermore, deep neural networks have flexibility in designing input-output structures such that it is possible to design arbitrary forms of information and state variables as inputs. Traditional model-based controller design methodologies derive the dynamic relationship between state variables and control commands, and then a design methodology for control is selected based on the relationship. Unlike conventional controller design methods, deep neural network controllers have no restrictions on the controller's mathematical form, which can improve the control system's performance.

The contribution of this study is to propose a concept of the statistical stability and binary stability conditions for nonautonomous systems with reference signals and to propose a framework to verify them. The binary Lyapunov stability condition is designed so that the definition of stability in conventional control theory can be applied to finite time

intervals. Although it is impossible to show asymptotic stability using the proposed stability analysis method, stability for time intervals long enough for practical use can be statistically guaranteed. The theorem on permutations of multisets is proven to satisfy the statistical stability of arbitrarily length time-series data.

This study is organized as follows. In Section II, the design of feedback controllers using deep neural networks is described. In Section III, the proposed framework is introduced, and the design methods for each element that makes up the proposed model and the role of each element are introduced. In Section IV, an example of implementing the proposed model and its results are shown. Finally, in Section V, the conclusions are described.

## II. DEEP NEURAL NETWORK-BASED CONTROLLERS

Let us consider the following control system.

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u})\end{aligned}\quad (1)$$

where  $\mathbf{x}$  is the state vector,  $\mathbf{u}$  is the control input, and  $\mathbf{y}$  is the system output. The trajectory of the state vector is represented as follows.

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t_f} f(\mathbf{x}, \mathbf{u}) dt \quad (2)$$

In general, feedback controllers  $\mathbf{u}(\mathbf{x})$  are designed in the form of functions for system state variables. Linear feedback controllers (including the PID controller and linear quadratic regulator) and nonlinear feedback controllers are examples of controllers in the form of functions for state variables. In particular, backstepping control [16] and extended state observer-based control [17] may use state variables created using a dynamic extension or similar methods.

Moreover, since the deep neural network can be used to approximate and express arbitrary functions, it is also possible to design a feedback controller consisting of only deep neural networks if the deep neural network is designed as a function for state variables. For example, if a linear activation function is used for a neural network of fully connected layer  $\mathcal{N}_D$  without a bias node and the state variable vector  $\mathbf{x}$  is used as the input of  $\mathcal{N}_D$ , then the output of  $\mathcal{N}_D$  can be expressed as a linear equation using one matrix  $\mathbf{W}$ , such as  $\mathcal{N}_D(\mathbf{x}) = \mathbf{W}\mathbf{x}$ . Therefore, if the output of  $\mathcal{N}_D$  is used as a control command,  $\mathcal{N}_D(\mathbf{x}) = \mathbf{u}$  has the same form as a linear controller.

Similar to  $\mathcal{N}_D$ , a deep neural network  $\mathcal{C}$  can represent a feedback controller that is a function of the state variable. Thus, the input to the deep neural network is a state variable  $\mathbf{x}$ , and the output can be expressed as  $\mathbf{u} = \mathcal{C}(\mathbf{x})$ . Nonlinear functions such as the rectified linear unit (ReLU) or sigmoid functions can be used as the activation function of  $\mathcal{C}$ . Moreover,  $\mathcal{C}$  may not be simply a fully connected layer but a deep neural network with arbitrary structures. Thus,  $\mathcal{C}(\mathbf{x})$  can be used as a nonlinear feedback controller with an arbitrary structure.

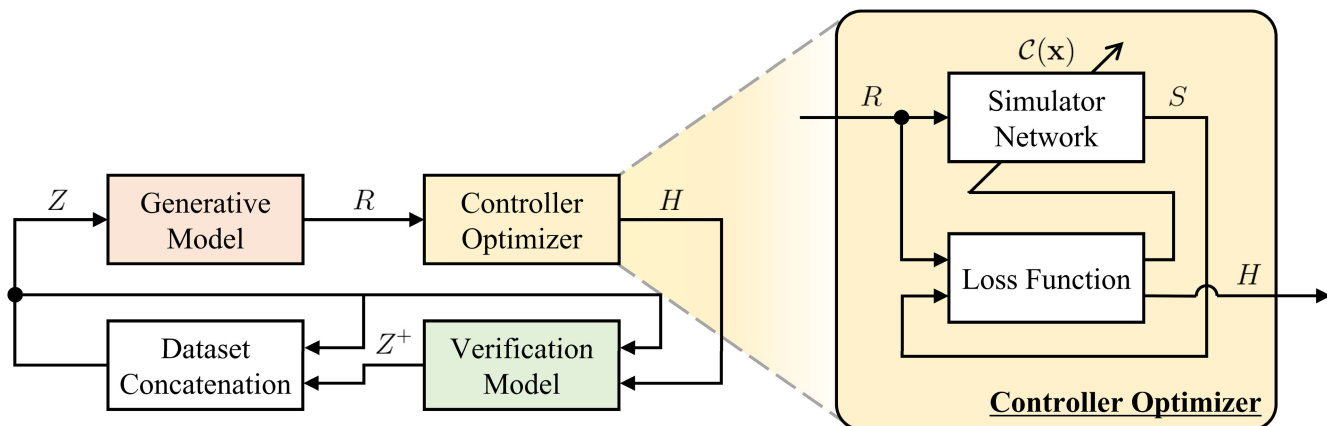


FIGURE 1. Generative verification model for DNN-based controller schematic.

Deep neural network structures are designed with various types of structures according to the goals to be learned. For example, deep neural networks that process images, which are two-dimensional data, use convolutional layers to extract feature points, and deep neural networks that process time-series data can use structures such as recurrent neural networks (RNNs) and long short-term memory (LSTM) [18], [19]. Likewise, deep neural network controllers for use as nonlinear feedback controllers should be designed as structures to improve control performance.

### III. GENERATIVE VERIFICATION MODEL

In this study, a generative verification model (GVM) is proposed as a methodology for verifying the performance and stability of nonlinear deep neural network controller models. Since nonlinear deep neural network controller models are nonlinear functions with arbitrary shapes, techniques such as Lyapunov stability analysis methodologies used for stability analysis in conventional nonlinear controller design methodologies cannot be applied. Since the GVM proposed in this study is applicable to all controllers given in the form of arbitrary nonlinear functions, it is versatile based on the fact that it can be utilized even if it is not a deep neural network controller.

The GVM structure for the controller stability analysis is shown in Figure 1. The controller stability analysis structure using GVM consists of three parts: controller optimizer (CO), generative model (GM), and verification model (VM). In Fig. 1, the input for CO is the set of time-series data  $R$ , which is used as the reference signal for the controller. The output of the CO is the simulation result and its evaluation, including some labels,  $H$ . The inputs for the VM are the set of feature vectors  $Z$  distributed in the latent space and the corresponding set of labels  $H$ . A new set of feature vectors  $Z^+$  is sampled in VM. GM generates a set of time-series data  $R$  for dataset  $Z$  during every iteration. More detailed explanations of each block represented in Fig. 1 are provided in the corresponding sections.

#### A. CONTROLLER OPTIMIZER

In the CO, the DNN-based controller is trained to have optimal performance for a given reference trajectory. Thus, the input of CO is a set of reference inputs. Since the reference input is a target to be followed by the control system's output, it is time-series data with various trajectories. For example, when designing a linear system controller such as a PID controller, a unit step function can be used as a reference trajectory to evaluate the time response. Additionally, since LQR aims to regulate the output to zero, it can be understood that the reference trajectory is a zero function.

In this study, it is assumed that any shape of time-series data is given as a reference input, such as a unit step function or a zero function, rather than a specific shape of reference trajectories. Thus, CO considers time-series data with arbitrary shapes of a particular length as a reference input. The right of Fig. 1 shows a block diagram of CO.  $R$  is a batch of reference inputs given in the form of time-series data. The CO is implemented as a DNN, where the simulator network contained in CO performs numerical simulations on  $R$ . The simulator network includes a deep neural network controller model  $\mathcal{C}(\mathbf{x})$ . The time-series dataset  $S$  obtained from numerical simulation is used in the loss function calculation with  $R$ . The deep neural network controller model  $\mathcal{C}(\mathbf{x})$  is trained to minimize the loss function. Then, the numerical simulation result  $s \in S$  performed with respect to all reference trajectories  $r$  included in the set  $R$ , i.e.,  $r \in R$ , are evaluated and classified. If the simulation result  $s$  is satisfactory, then  $s \in S_p \subset S$ . Otherwise,  $s \in S_f \subset S$ . The classification result is labeled to obtain  $\lambda \in [0, 1]$ .  $H$  is the set of labels  $\lambda$ .

Meanwhile, the controller model  $\mathcal{C}(\mathbf{x})$  is optimized by CO for a given  $R$ , and its optimal performance is determined by the loss function. For example, a loss function can be designed as Eq. (3). If the DNN-based controller model  $\mathcal{C}(\mathbf{x})$  is designed as a fully connected layer without bias and a linear activation function is used, then the CO-learned  $\mathcal{C}(\mathbf{x}) = \mathbf{W}\mathbf{x}$

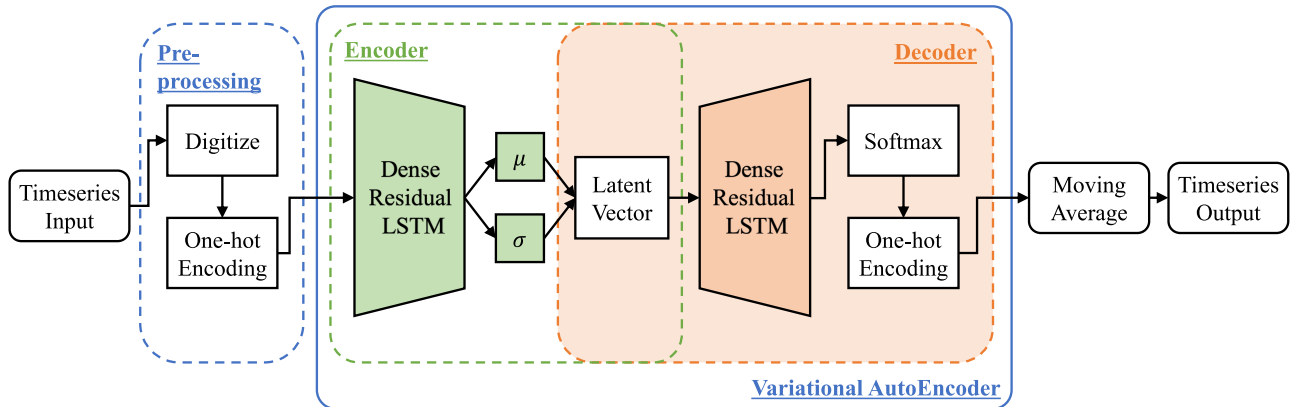


FIGURE 2. Variational autoencoder.

will be an LQR controller.

$$J = \mathbf{x}^T(t_f)\mathbf{F}\mathbf{x}(t_f) + \int_{t_0}^{t_f} (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathcal{C}(\mathbf{x})^T\mathbf{R}\mathcal{C}(\mathbf{x})) dt \quad (3)$$

However, the proposed CO in this work has versatility because it allows the consideration of an arbitrary nonlinear DNN-based controller model  $\mathcal{C}(\mathbf{x})$  and arbitrary loss functions depending on the control purpose.

A tracking controller is often designed by defining an error between the signal to be followed and the system output and designing a regulator for the error dynamics. This method can also be adopted when designing a deep neural network controller, but in this study, the state variables  $\mathbf{x}$  and the reference signals to be followed  $r$  are used as input to the controller to emphasize the flexibility of the deep neural network controller structure. That is,  $\mathbf{u} = \mathcal{C}(\mathbf{x}, r)$ . This implies that any beneficial input to a deep neural network controller can be used.

Figure 1 shows that the GVM has a structure that iterates along the direction of the arrow. An  $R$  output from the GM is given as an input to the CO, where a DNN-based controller model  $\mathcal{C}(\mathbf{x})$  is trained whenever a new  $R$  is given in each iteration. Meanwhile, the nonlinear DNN-based controller model trained by the CO results from learning about a given dataset  $R$  and is an arbitrary nonlinear function, so its stability is not guaranteed by CO alone. In other words, the tracking performance cannot be guaranteed if new time-series data  $\bar{r} \notin R$  not included in the dataset used for training are given as reference input. In this study, GVM, including CO, is proposed by designing GM and VM to overcome this.

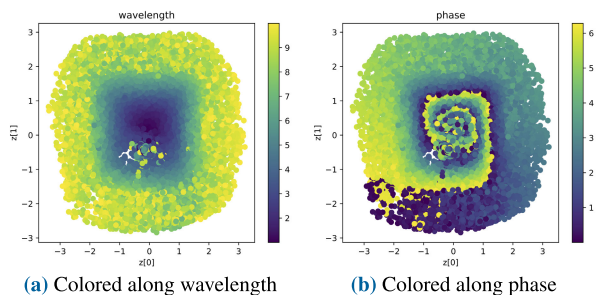
## B. GENERATIVE MODEL

GM generates a set  $R$  of time-series data. Since  $R$  is a training dataset that CO learns, it contains various forms of time-series data, making it valuable. GM is a kind of generative model. The generative model refers to a model that can generate new data that did not exist previously based on various input data. For example, training a generative model with time-series data as input data can generate new data similar to the training

data but not included in the training dataset. These generative models are usually trained using structures such as generative adversarial networks (GANs) [20], variational autoencoders (VAEs) [21], and adversarial autoencoders (AAEs) [22].

GM not only needs to generate new time-series data but also needs to have a feature space because the VM must have a vector space to learn the stability region. A machine learning technique that can create a feature space based on given data is called feature learning or representation learning [23]. Representation learning is widely used in fields such as speech recognition [24], object recognition [25], and natural language processing [26]. Since the VAE can obtain feature space by automatically performing representation learning while training the generative model, VAE is adopted in this study.

The GM is designed with a VAE structure to generate arbitrary shapes of time-series data. A VAE is a type of autoencoder, a structure in which an encoder and a decoder are connected. The encoder of the VAE has a structure in which the given input data are compressed and distributed on a multivariate vector space. Multivariate vector spaces are referred to as latent spaces or feature vector spaces. Each input data point is mapped to a point on the latent space, which is restored by the decoder in the same form as the input data. Figure 2 shows the structure of the VAE used in this study. The VAE is trained to minimize reproduction errors between input data and data restored by the decoder, such as a typical autoencoder [21]. The difference between a typical autoencoder and a VAE is that the latter adds a normalization term to the loss function so that the feature vectors obtained through the encoder follow a predetermined distribution in the latent space. Since the Gaussian distribution is considered in this study, the encoder structure, as shown in Figure 2, includes a node that outputs the average and standard deviation of the Gaussian distribution. The decoder of the VAE may be used as a generative model. Therefore, arbitrary time-series data can be generated by extracting any point in the latent space and passing it through the decoder. Meanwhile, output data obtained by decoding



**FIGURE 3. Feature vectors distributed in latent space ( $n_l = 2$ ) of VAE trained on sinusoidal time-series data.**

vectors located at similar positions in the latent space are similar.

In the VAE, the time-series data given as an input are compressed and converted into a feature vector on the latent space. The dimension of the latent space,  $n_l \in \mathbb{N}$ , is a hyperparameter that may be arbitrarily determined when designing the VAE. The larger the dimension of the latent space, the higher the classification performance of the VAE because the feature vector can contain more information as the dimension increases. Therefore, using a higher  $n_l$  is recommended.

In this study, we leverage these VAE features to generate a set  $R$  of reference trajectories. In other words, the role of GM is to generate various reference trajectories in an arbitrary form for the control system of CO to learn. The shape of time-series data that is suitable may vary depending on the type of control system. To obtain a good GM, a set of time-series data suitable for the reference trajectory must be used as training data. This study uses sinusoidal time-series data with various wavelengths, amplitudes, and initial phases.

In GVM, GM uses a decoder from a pretrained VAE. Unlike the process of CO learning a new controller in every iteration, GM is not retrained in every iteration. After the VAE is trained, only the decoder part of the VAE may be used as a GM. Thus, as shown in Figure 1, for GM, the input  $Z$  is a set of vectors distributed in the latent space. The set  $Z$  is concatenated with the output  $Z^+$  of the VM, and the number of elements gradually increases. GM generates a set of time-series data  $R$  for a new  $Z^+$ , including an existing  $Z$  every iteration.

**C. VERIFICATION MODEL**

The latent space of the VAE is a vector space from which the characteristics of time-series data are extracted, and the distribution of the data on the vector space reflects the characteristics of time-series data. Therefore, decoding two different feature vectors at similar locations in the latent space results in time-series data with similar forms of time response. Since the VAE automatically classifies the feature vectors obtained by encoding input data, the physical meaning of each dimension cannot be specified in advance. For example, if a VAE learns sine waves with various wavelengths and amplitudes and maps them to a two-dimensional feature vector space,

time-series data with similar wavelengths and amplitudes may be distributed at similar locations in the feature vector space. However, the corresponding two-dimensional vector coordinates do not precisely represent the wavelength and amplitude.

Figure 3 shows an example of classifying different sinusoidal time-series data with different wavelengths and amplitudes into a two-dimensional latent space. It can be seen that time-series data with similar characteristics are mapped to similar positions. Moreover, structured data such as sinusoidal time-series data can be classified into parameters with physical meaning, such as wavelength and amplitude. In contrast, time-series data with arbitrary shapes cannot be classified as a finite number of parameters with physical meaning. However, the VAE automatically extracts statistically similar features and classifies time-series data.

In this study, the performance region of the control system is derived by performing binary classification in the latent space  $\mathcal{F}$ , utilizing the VAE’s characteristics that it can classify time-series data with arbitrary forms. The latent space is a set of reference trajectories  $Z_p(\in \mathcal{F}_p \subset \mathcal{F})$  that the system output can follow using the deep neural network model controller  $\mathcal{C}(\mathbf{x})$  and a set of reference trajectories that cannot be followed.

Subspaces  $\mathcal{F}_p$  and  $\mathcal{F}_f$  of latent space  $\mathcal{F}$  represent areas where the system output can and cannot follow the reference trajectory using the controller  $\mathcal{C}(\mathbf{x})$ , respectively. If  $\mathcal{C}(\mathbf{x})$  varies for the same latent space, then subspaces  $\mathcal{F}_p, \mathcal{F}_f$  can change. Therefore, we express them as  $\mathcal{F}_p|_{\mathcal{C}}$  and  $\mathcal{F}_f|_{\mathcal{C}}$ .

The inputs of the VM in Figure 1 are  $Z$  and  $H$ . Here,  $Z$  is the set of vectors  $\mathbf{z}_i$  in the latent space, and  $H$  is the set of labels  $\lambda_i$  assigned for each vector  $\mathbf{z}_i$ . This label is determined by evaluating numerical simulation results using a CO-trained controller  $\mathcal{C}(\mathbf{x})$ . Given an ordered pair  $(\mathbf{z}_i, \lambda_i)$  of data and labels, a binary classification model  $\mathcal{D}(\mathbf{z})$  that bisects the latent space using a support vector machine (SVM) can be derived.  $\mathcal{D}(\mathbf{z})$  receives the feature vector  $\mathbf{z}_i$  and outputs a predicted value  $p_i(\in [0, 1])$ . Since the label  $\lambda_i$  corresponding to a given  $\mathbf{z}_i$  in the learning dataset  $(\mathbf{z}_i, \lambda_i)$  can vary with the controller  $\mathcal{C}(\mathbf{x})$ , the  $\mathcal{D}(\mathbf{z})$  that is learned also depends on the controller. Thus, it can be expressed as  $\mathcal{D}|_{\mathcal{C}}(\mathbf{z})$ .

Finally,  $\mathcal{D}|_{\mathcal{C}}(\mathbf{z})$  is used for the performance analysis of the controller. Since any reference trajectory can be converted to the feature vector  $\mathbf{z}$ ,  $\mathcal{D}|_{\mathcal{C}}(\mathbf{z})$  evaluates this feature vector to derive whether it is traceable. The latent space is bisected by  $\mathcal{D}|_{\mathcal{C}}(\mathbf{z})$  into  $\mathcal{F}_p|_{\mathcal{C}}$  and  $\mathcal{F}_f|_{\mathcal{C}}$ , where the followable region  $\mathcal{F}_p|_{\mathcal{C}}$  within the latent space  $\mathcal{F}$  is the performance region of controller  $\mathcal{C}(\mathbf{x})$ . It can be argued that the tracking performance and stability of the controller  $\mathcal{C}(\mathbf{x})$  are statistically guaranteed within  $\mathcal{F}_p|_{\mathcal{C}}$ . The statistical stability is addressed in the following section.

Meanwhile, the output of the VM shown in Figure 1 is  $Z^+$ , which is a newly extracted sample set from the latent space. This new set of samples is obtained by performing active sampling based on a classification model  $\mathcal{D}|_{\mathcal{C}}$  trained on the VM. Since a new set of samples  $Z^+$  is added to each

iteration of the GVM and accumulates on the entire dataset, the performance of the controller  $\mathcal{C}(\mathbf{x})$  trained by the CO and the classification model  $\mathcal{D}|_{\mathcal{C}}(\mathbf{z})$  trained by the VM can gradually increase using more iterations.

#### IV. STABILITY OF DNN-BASED CONTROLLERS

##### A. STABILITY FOR NONAUTONOMOUS SYSTEMS REVISITED

The conventional definition of stability for nonautonomous systems is revisited to set up the definition of statistical stability of DNN-based controllers. A system is described as stable if starting the system somewhere near its desired operating point implies that it will stay around the point ever after [3]. In this study, the stability for nonautonomous systems is considered because the reference tracking controller  $\mathbf{u} = \mathcal{C}(\mathbf{x}, \mathbf{r})$  includes a predefined reference signal  $\mathbf{r}(t)$ , which is a time-varying variable, i.e., an explicit function of time.

*Definition 1 (Stability for Nonautonomous Systems):* The equilibrium point  $\mathbf{0}$  is stable at  $t_0$  if for any  $R > 0$ , there exists a positive scalar  $r(R, t_0)$  such that

$$\|\mathbf{x}(t_0)\| < r \Rightarrow \|\mathbf{x}(t)\| < R \quad \forall t \geq t_0 \quad (4)$$

Otherwise, the equilibrium point  $\mathbf{0}$  is unstable.

The stability in Definition 1 is also known as the stability in the sense of Lyapunov, and defines the uniformly boundedness of the solution.

*Definition 2 (Uniformly Ultimately Boundedness for Nonautonomous Systems):* The solutions  $x(t)$  of nonautonomous system are uniformly ultimately bounded with ultimate bound  $b$  if there exist positive constants  $b$  and  $c$ , independent of  $t_0 \geq 0$ , and for every  $a \in (0, c)$ , there is  $T = T(a, b) \geq 0$ , independent of  $t_0$ , such that

$$\|x(t_0)\| \leq a \Rightarrow \|x(t)\| \leq b, \forall t \geq t_0 + T. \quad (5)$$

The solution is globally uniformly ultimately bounded if (5) holds for arbitrarily large  $a$ .

In general, the stability in the sense of Lyapunov is insufficient, and asymptotic stability is required to guarantee the performance of controllers.

*Definition 3 (Asymptotic Stability for Nonautonomous Systems):* The equilibrium point  $\mathbf{0}$  is asymptotically stable at time  $t_0$  if the system is stable and  $\exists r(t_0) > 0$  such that

$$\|\mathbf{x}(t_0)\| < r(t_0) \Rightarrow \|\mathbf{x}(t)\| \rightarrow \mathbf{0} \text{ as } t \rightarrow \infty \quad (6)$$

Note that to guarantee the asymptotic stability of the system, the time interval of infinity needs to be considered. However, the verification model can only cover time-series data of finite length. The concept of statistical stability and binary stability conditions are proposed in this study for data-driven controllers and verification models in the following sections.

##### B. STATISTICAL STABILITY

The definition of statistical stability is presented in Definition 4.

*Definition 4 (Statistical Stability):* Consider a domain  $\mathcal{F}$ , which is a set of finite-length trajectories  $\mathbf{r}(t)$ ,  $t \in [t_0, t_f]$ ,

and the initial state  $\mathbf{x}(t_0) \in \mathcal{U} \subset \mathcal{X}$ , where  $\mathcal{U}$  is a bounded set in the state space  $\mathcal{X}$  of an autonomous control system in Eq. (1). The control system with a DNN-based controller  $\mathbf{u}(t) = \mathcal{C}(\mathbf{x}(t), \mathbf{r}(t))$  is statistically stable in  $\mathcal{F}_p \subset \mathcal{F}$  with respect to a binary stability condition function  $\lambda(\mathbf{r}(t), h(\mathbf{x}(t), \mathbf{u}(t)))$  if  $\lambda(\mathbf{r}(t), \mathbf{y}(t)) = \text{true} \quad \forall \mathbf{r}(t) \in \mathcal{F}_p$  and  $\mathbf{x}(t) \in \mathcal{U} \quad \forall t \in [t_0, t_f]$ .

The binary stability condition function  $\lambda(\mathbf{r}, \mathbf{y}) : \mathcal{F} \mapsto [\text{true}, \text{false}]$  is a function that evaluates the binary stability between two different trajectories of the same time interval and returns a binary label. If  $\lambda(\mathbf{r}, \mathbf{y}) = \text{true}$ , then the two trajectories are binary stable. If  $\lambda(\mathbf{r}, \mathbf{y}) = \text{false}$ , then the two trajectories are not binary stable, i.e., binary unstable.

Binary stability conditions are proposed to consider the stability of nonautonomous systems based on time-series data of finite length. The binary stability conditions can be defined in various ways. One of the binary stability conditions can be defined from the stability in the sense of Lyapunov.

*Definition 5 (Binary Lyapunov Stability Condition):* The binary Lyapunov stability condition of two different trajectories  $a(t)$  and  $b(t)$  on the same finite time interval  $t \in [t_0, t_f]$  is considered to be satisfied when the error between two trajectories,  $e(t) = a(t) - b(t)$ , satisfies the following condition.

$$\forall t \in [t_0, t_f], \quad \|e(t)\| < \|e(0)\| \quad (7)$$

If the trajectories  $a$  and  $b$  are vectors, the binary stability condition is evaluated in an elementwise manner. Note that the binary Lyapunov stability condition is defined for a finite time interval. As stated in the previous section, the Lyapunov stability only guarantees the boundedness of the signal. The asymptotic stability condition is required for the performance of the tracking controller. However, asymptotic stability is only applicable for infinite time intervals, which is inapplicable for data-driven controllers that utilize time-series data of finite length. Another binary stability condition is defined to overcome this inapplicability. The definition of the binary integral stability condition is given in Definition 6.

*Definition 6 (Binary Integral Stability Condition):* The binary integral stability condition of two different trajectories  $a(t)$  and  $b(t)$  on the same finite time interval  $t \in [t_0, t_f]$  is considered to be satisfied when the mean error  $e(a, b)$  between the two trajectories is below a constant threshold  $\varepsilon$ .

$$e(a, b) = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} (a(t) - b(t))^2 dt < \varepsilon \quad (8)$$

The binary integral stability condition requires the integral error to be smaller than the given threshold,  $\varepsilon$ . The threshold is a design parameter that the designer can set to obtain a stability region of satisfactory tracking performance. That is, it is possible to obtain a stability region of the system that has a specific supremum value of the integral error. When the binary integral stability condition is invoked in defining the statistical stability, the mean error between the reference trajectory  $\mathbf{r}(t)$  and the output trajectory  $\mathbf{y}(t)$  is considered.

The term statistical stability implies that the stability is satisfied in a statistical manner. The binary integral stability condition in Definition 6 only guarantees the boundedness of the tracking error during a finite time interval. It seems insufficient, superficially. However, the statistical stability is sufficient to guarantee the boundedness during a much longer length of time for a data-driven controller with a split invariant loss function.

### C. SPLIT INVARIANCE OF A LOSS FUNCTION

Consider a trajectory  $\mathbf{r}(t)$ ,  $t \in [t_0, t_f]$ . Splitting  $\mathbf{r}(t)$  in  $N_s$  time intervals gives short trajectories  $\mathbf{r}_i(t)$  as follows.

$$\mathbf{r}(t) \Rightarrow \begin{cases} \mathbf{r}_0(t), & t \in [t_0, t_1] \\ \mathbf{r}_1(t), & t \in [t_1, t_2] \\ \vdots \\ \mathbf{r}_{N_s-1}(t), & t \in [t_{N_s-1}, t_{N_s} = t_f] \end{cases} \quad (9)$$

Then, the corresponding state trajectory  $\mathbf{x}(t)$ ,  $t \in [t_0, t_f]$  is also split as follows.

$$\mathbf{x}(t) \Rightarrow \begin{cases} \mathbf{x}_0(t), & t \in [t_0, t_1] \\ \mathbf{x}_1(t), & t \in [t_1, t_2] \\ \vdots \\ \mathbf{x}_{N_s-1}(t), & t \in [t_{N_s-1}, t_{N_s} = t_f] \end{cases} \quad (10)$$

The split invariant loss function is defined as follows.

*Definition 7 (Split Invariance of a Loss Function):* For a reference trajectory  $\mathbf{r}(t)$ ,  $t \in [t_0, t_f]$  and the corresponding state vector  $\mathbf{x}(t)$  of the control system in Eqs. (1,2) using a DNN-based controller  $\mathbf{u} = \mathcal{C}$  and the split set of time series in Eqs. (9,10), a loss function  $J(\mathbf{r}(t), \mathbf{x}(t))$  is split invariant if the following equation is satisfied.

$$\sum_{i=0}^{N_s-1} J(\mathbf{r}_i(t), \mathbf{x}_i(t))|_{t \in [t_i, t_{i+1}]} = J(\mathbf{r}(t), \mathbf{x}(t))|_{t \in [t_0, t_{N_s}]} \quad (11)$$

Definition 7 represents the fact that if a loss function is split invariant, then the summation of the loss value evaluated on every split time interval is equivalent to the loss value evaluated on the concatenated time interval. Therefore, if a DNN-based controller  $\mathcal{C}$  is trained for a split invariant loss function, training with a reference trajectory  $\mathbf{r}(t)$ ,  $t \in [t_0, t_f]$  and the initial state  $\mathbf{x}(t_0)$  is equivalent to training with a set of split time intervals  $\{\mathbf{r}_i(t)|i = 0, \dots, N_s - 1\}$  and the set of initial states  $\{\mathbf{x}(t_0), \dots, \mathbf{x}(t_{N_s-1})\}$ .

*Definition 8 (Permutations of Multiset):* For a domain  $\mathcal{F}$  that is a set of time-series data of finite length, the set of time-series data, which is a concatenation of a finite number of elements from  $\mathcal{F}$  allowing repetition, is defined as  $\mathbb{P}_{\mathcal{M}}(\mathcal{F})$ .

Note that  $\mathcal{F} \subset \mathbb{P}_{\mathcal{M}}(\mathcal{F})$  by definition. The statistical stability of a system on a given domain  $\mathcal{F}$  is expanded to  $\mathbb{P}_{\mathcal{M}}(\mathcal{F})$  with the following theorem.

*Theorem 1 (Statistical Stability for Permutations of Multiset):* If a control system in Eq. (1) with a DNN-based controller  $\mathbf{u} = \mathcal{C}$  is statistically stable in a domain of trajectories  $\mathcal{F}$  and a domain of state vector  $\mathcal{U} \subset \mathcal{X}$ , then the system is statistically stable in the domains  $\mathbb{P}_{\mathcal{M}}(\mathcal{F})$  and  $\mathcal{U}$ .

*Proof:* Consider a trajectory  $\mathbf{r}(t) \in \mathbb{P}_{\mathcal{M}}(\mathcal{F})$ ,  $t \in [t_0, t_f]$  constructed by concatenating  $N_s$  elements  $\mathbf{r}_i(t)$ ,  $i \in [0, N_s - 1]$  from the domain  $\mathcal{F}$ . If an arbitrary initial state  $\zeta_0 \in \mathcal{U}$  is chosen, then the resulting state trajectory of the system for the pair  $(\mathbf{r}(t), \zeta_0)$  is  $\mathbf{x}(t)$ ,  $t \in [t_0, t_f]$ , where  $\zeta_0 = \mathbf{x}(t_0)$ . The statistical stability for a pair of trajectory and initial state  $(\mathbf{r}(t), \zeta_0)$  is satisfied if and only if the statistical stability is satisfied for all of the pairs  $(\mathbf{r}_i(t), \zeta_i = \mathbf{x}(t_i))$ .

Assume that the system is statistically stable for a pair  $(\mathbf{r}_i(t), \zeta_i) \in (\mathcal{F}, \mathcal{U})$ . The resulting state trajectory is  $\mathbf{x}_i(t)$ ,  $t \in [t_i, t_{i+1}]$ . If we choose  $\zeta_{i+1} = \mathbf{x}_i(t_{i+1})$ , then  $\zeta_{i+1} \in \mathcal{U}$  because of the statistical stability. Then, the system is also statistically stable for the pair  $(\mathbf{r}_{i+1}(t), \zeta_{i+1})$ . Meanwhile, the system is statistically stable for the trajectory  $\mathbf{r}_0(t) \in \mathcal{F}$ ,  $t \in [t_0, t_1]$  and the initial state  $\zeta_0 = \mathbf{x}(t_0) \in \mathcal{U}$ . By mathematical induction, the system is statistically stable for every pair  $(\mathbf{r}_i(t), \zeta_i = \mathbf{x}(t_i)) \forall i \in [0, N_s - 1]$ . Therefore, the system is statistically stable for domains  $\mathbb{P}_{\mathcal{M}}(\mathcal{F})$  and  $\mathcal{U}$ .  $\square$

Theorem 1 suggests that a DNN-based controller can be effectively trained with a batch of short time-series data for the statistical stability of the control system, which is often operated over a longer timespan. Statistical stability and binary stability conditions should directly evaluate stability conditions for time-series data of a given length. However, the statistical stability of the control system can be secured for a time interval with an arbitrary length by Theorem 1. Theorem 1 extends the time interval length to make more practical use of the statistical stability and binary stability conditions defined in Definitions 4 to 6.

## V. RESULTS

In this section, the entire design framework proposed in Fig. 1 is realized one by one and interconnected. First, the DNN-based controller is designed and optimized for the system represented in Eqs. (1) and (2). Next is realization of the generative model. Then, the verification model is trained to obtain the stability region of the system in the latent space of the GM. SVM is used to train the decision boundary of the stability region. The training data for VM are labeled using the binary integral stability condition from Eq. (8). Finally, the process is iterated as described in Fig. 1 to increase the performance of the DNN-based controller and reliability of the stability region. After the process, the resulting optimal controller can be utilized for a much longer time interval than the length of the time-series data utilized in the controller optimization process, as described in Theorem 1. The longer time-series data are constructed as represented in Eqs. (9) and (10).

### A. DEEP NEURAL NETWORK-BASED CONTROLLER DESIGN AND NUMERICAL SIMULATION

In this study, the proposed deep neural network controller is trained by applying it to linear systems, and the tracking performance for reference inputs is analyzed.

1) PLANT

The standard second-order system is considered as follows.

$$\frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} \quad (12)$$

The state-space representation of the standard second-order system can be written as follows.

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} u \\ &= \mathbf{Ax} + \mathbf{Bu} \\ y &= [0 \ 1] \mathbf{x} \\ &= \mathbf{Cx} \end{aligned} \quad (13)$$

2) DNN-BASED CONTROLLER

The objective of the controller is to make the output  $y(t)$  track the given reference signal  $r(t)$ . Therefore, the state vector  $\mathbf{x}(t)$  and reference signal  $r(t)$  are given as inputs for the controller.

$$u(t) = \mathcal{C}(\mathbf{x}(t), r(t)) \quad (14)$$

The model structure of the DNN-based controller is shown in Fig. 4. The model includes fully connected layers with scaled exponential linear units (SELUs) and sigmoid activation functions.

In the conventional controller design methodology, it is necessary to mathematically define the input/output relationship of the system and then design a control command that can guarantee mathematical stability and performance. Therefore, conventional controller design methodologies can only be used to design controllers by clearly deriving mathematical relationships between specific variables and system outputs. In contrast, DNN-based controllers can select whatever information is available as input to deep neural networks, and it is not necessary to derive mathematical relational expressions directly for control commands.

Note that the information between the system output  $y$  and state vector  $\mathbf{x}$  is not given to the DNN-based controller. The controller is only trained to generate the control command  $u$  minimizing the loss value, a function of the reference signal  $r$  and state vector  $\mathbf{x}$ . The loss function considered in this study is given in Eq. (15).

$$J(r, y) = \frac{1}{T_f} \int_0^{T_f} (r(t) - y(t))^2 dt \quad (15)$$

Note that Eq. (15) is split invariant. Additionally, Eq. (15) can be directly used for the binary integral stability condition in Eq. (8). Because the objective is to train a tracking controller, the loss function in Eq. (15) minimizes the error between the reference signal  $r(t)$  and the output  $y(t)$ . Here, the system output is included in the loss function. However, the system output is not provided as an input for the DNN-based controller. That is, it can be stated that the DNN-based controller can achieve the control objective with limited information.

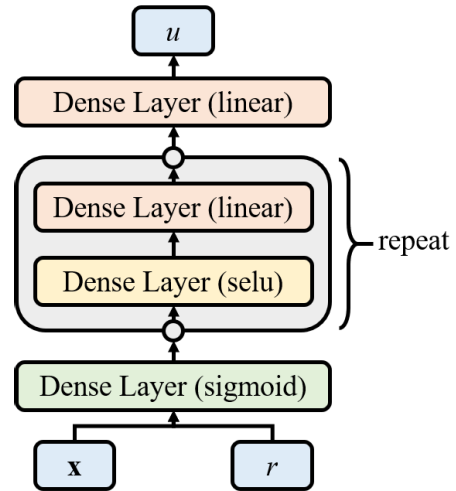


FIGURE 4. Neural-network structure for tracking controller.

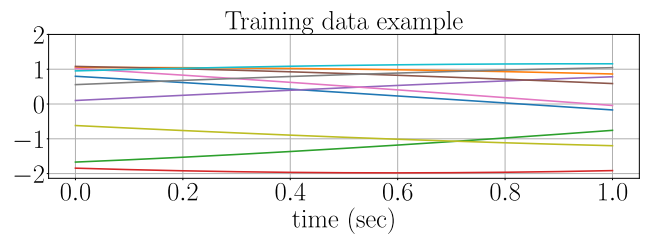


FIGURE 5. Example of training data.

3) TRAINING DATASET

In this study, the DNN-based controller of the structure described in Fig. 4 is trained. The training data are a set of fixed-length sinusoidal reference signals with random magnitudes and initial phases. The parameters for training are summarized in Table 1. Figure 5 shows some example signals from the training dataset. The different colors in the figure represent different training data. The number of data points in the training dataset is 1024.

4) TRAINING RESULT

Figure 6 shows the simulation results for performance verification of the trained controller  $\mathcal{C}(\mathbf{x})$ . Several results for randomly generated sinusoidal reference signals are displayed at once. The different colors in the figure represent different example cases, and the data with the same color represent the same case. The training data are time-series data with a time interval of 0.1 sec and a length of 1 sec. In contrast, the verification data are time-series data with a length of 20 sec and the same time interval. The validation reference signals are sinusoidal time-series data with random magnitudes, wavelengths, and initial phases. The training data have signals with a fixed wavelength of 10 sec only, but the validation data have signals with various wavelengths. It can be seen from the simulation result that the DNN-based controller can track the given reference signal in a small range of error.



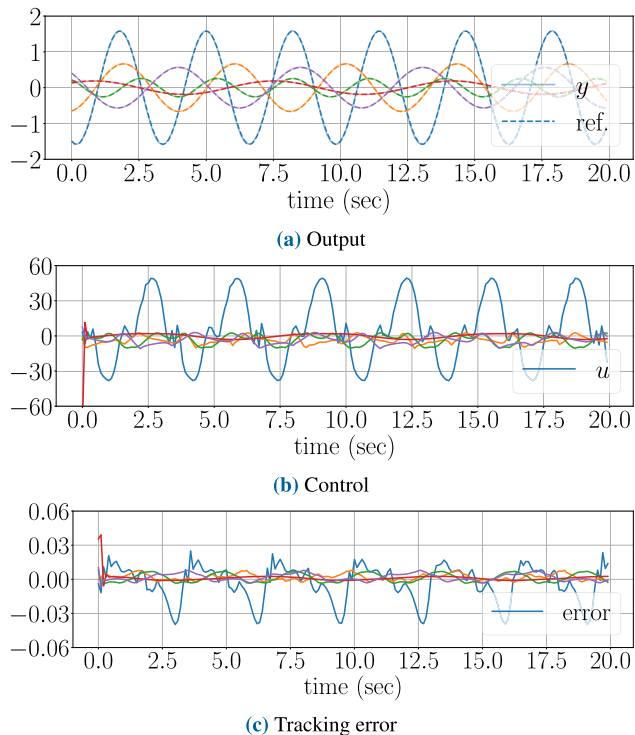


FIGURE 6. Tracking simulation examples for trained controller  $\mathcal{C}(\mathbf{x})$ .

TABLE 1. Parameters for system, controller, and training dataset.

Category	Parameter	(Range of) Value
System	$\zeta$	0.25
	$\omega_n$	1
Controller	Number of repeat blocks	5
Dataset	Magnitude of $r$	$\in [1, 2]$
	Wavelength of $r$ (sec)	10
	Time length (sec)	1

In this section, it has been shown that it is possible to train a tracking controller with good performance using the proposed DNN-based controller structure. However, the tracking performance cannot be guaranteed for reference signals not included in the training dataset. In this study, an approach of performance region analysis using GVM is proposed to overcome the limitations of DNN-based controllers.

**B. REALIZATION OF GENERATIVE MODEL**

In this study, the VAE is used for the representation learning of time-series data and to generate arbitrary time-series data. The latent space is set to be 3-dimensional. The latent space’s dimension  $n_l$  is set in three dimensions because the sinusoidal time-series data used for VAE’s training are generated according to three different characteristics, i.e., magnitudes, wavelengths, and phases. However, in general, the VAE may be trained using other types of time-series data, which may have more complicated features depending on the applications of the control system. Because it is difficult to visualize

the performance region classified using the SVM in the latent space where the dimension of latent space is 4D or higher ( $n_l \geq 4$ ),  $n_l = 3$  is considered in this study. If visualization is not considered, then it is generally desirable to set  $n_l$  to a larger value to enhance the performance of representation learning.

The latent space of the trained VAE is shown in Fig. 7, and each feature vector is colored according to the wavelength and amplitude of the time-series data used for training. In this study,  $2^{15}$  time-series data with random magnitudes, wavelengths, and initial phases are used to train GM. The magnitude is normalized to have values in the range of  $[-1, 1]$ . Note that the feature vectors are classified according to the wavelength and amplitude. Figure 8 shows a comparison between the time-series data (encoder input) used as input and the time-series data (decoder output) obtained by reconstruction to check the performance of the trained VAE. The trained VAE successfully reconstructs the input data. The decoder part of the trained VAE is used as the GM. The GM can generate random time-series data in the shape of a sinusoidal wave with various magnitudes, wavelengths, and initial phases. In Section V-A3, the training data for CO included sinusoidal data with a fixed wavelength of 10 sec. In contrast, in this section, the time-series data generated from the GM are used to construct the training data for the CO.

**C. PERFORMANCE REGION ANALYSIS USING VERIFICATION MODEL**

The performance region analysis using the GVM should be conducted to guarantee the stability and performance of the DNN-based controller. The DNN-based controller is repeatedly trained with increasing data, and its performance is improved using the GVM. The performance region in which the DNN-based controllers’ performance and stability are guaranteed is also derived.

**1) BINARY VERIFICATION USING SUPPORT VECTOR MACHINE**

The VM is designed using the SVM in the latent space of the VAE. In Fig. 1, the inputs for the VM are the set of feature vectors distributed in the latent space and the corresponding set of labels. The labels are obtained by evaluating the simulation results. In this study, the label is determined by thresholding the loss value from Eq. (15). When the threshold value is  $J_T$ , the label  $\lambda_i$  for feature vector  $\mathbf{z}_i$  is determined as follows.

$$\lambda_i = \begin{cases} 1 & \text{if } J(r_i, y_i|\mathcal{C}) < J_T \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

where  $r_i$  is the reference signal decoded from  $\mathbf{z}_i$ , and  $y_i|\mathcal{C}$  is the tracking simulation output on  $r_i$  using controller  $\mathcal{C}$ . The trained SVM  $\mathcal{L}$  is directly used to analyze the performance region of  $\mathcal{C}$ . Note that Eq. (16) with Eq. (15) is interpreted as a binary stability condition function from Definition 4 using the binary integral stability condition on Definition 6.

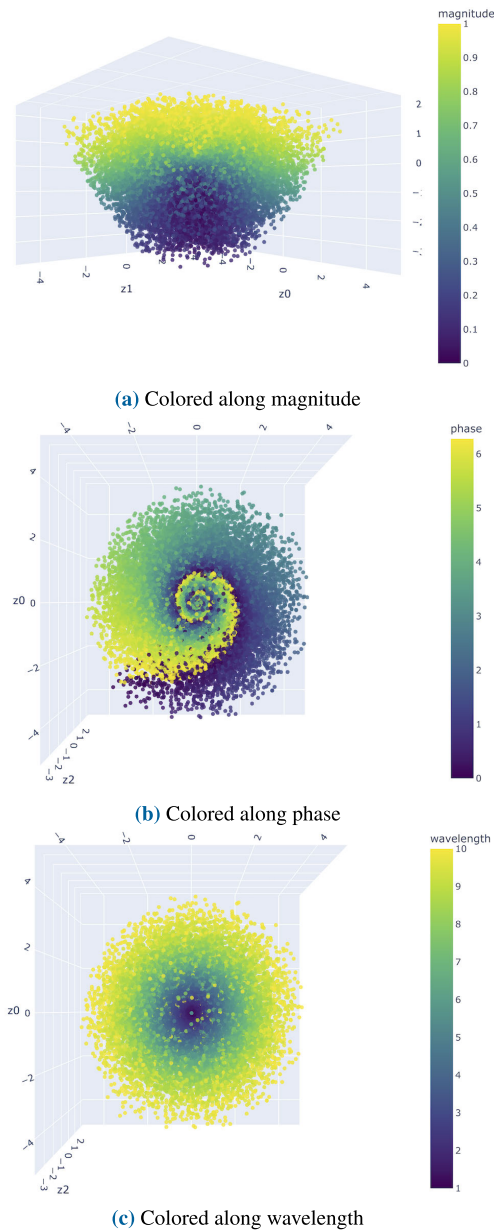


FIGURE 7. Feature vectors distributed in 3D latent space.

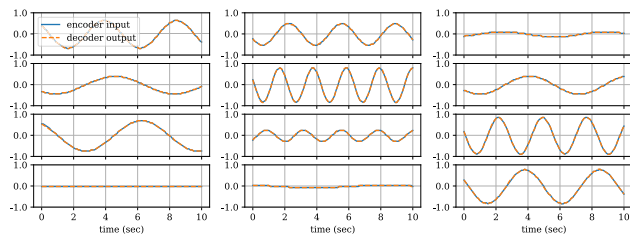


FIGURE 8. Reference signal examples generated using GM.

The VM is a machine learning model trained on the latent space. The true performance region  $\mathcal{F}_p^*$  that satisfies the statistical stability condition from Definition 4 in the latent space  $\mathcal{F}$  exists. However, the true performance region cannot

be attained analytically. Therefore, the VM is proposed in this study and is utilized to approximate the true performance region and attain an approximated performance region  $\mathcal{F}_p$ .

### 2) ACTIVE SAMPLING

The VM consists of the SVM and an active sampling process. After training the SVM  $\mathcal{L}$  using the given dataset  $(Z, H)$ , a new set of feature vectors  $Z^+$  is actively sampled. Fig. 1 shows that the output of the VM is  $Z^+$ . The newly sampled set  $Z^+$  is concatenated to the previous set  $Z$  to construct a new set with more elements. Then,  $H$  is updated by GM and CO, and the SVM is trained again. This process is iterated to retrain the controller  $\mathcal{C}$  and the SVM  $\mathcal{L}$ .

The active sampling algorithm is a technique to extract samples that can make the most significant model change to the current model  $\mathcal{L}$ . The sample points can be extracted more efficiently than the random sampling algorithm using the active sampling algorithm. Random sampling is conducted at the very first iteration of the GVM because the active sampling algorithm requires a trained model  $\mathcal{L}$ . Refer to [27], [28], [29], and [30] for detailed explanations of the active sampling algorithm and its applications on nonlinear system verification.

### 3) PERFORMANCE REGION

The performance region of the deep neural network controller is derived using the GVM in the latent space for reference signals. The parameters for GVM are summarized in Table 2. The generated time-series data are multiplied by the magnitude scaling coefficient to consider the reference signals of magnitudes larger than 1. The loss function from Eq. (15) is used. The time-series data used for training the controller in Section V-A are sinusoidal signals of features summarized in Table 1. In contrast, the time-series data used for training the controller in this section are randomly generated using the GM. Therefore, the reference signal may not be perfectly sinusoidal, and the range of wavelength and magnitude differ.

Figure 9 shows the convergence to the correct performance area of the controller according to the number of iterations. The sample points used to train  $\mathcal{L}$  are shown in the figure. The label for training the SVM is determined by Eq. (16). The decision boundary of the SVM is visualized in 2D planes that are orthogonal to each axis. In the figure, data points classified to belong to  $\mathcal{F}_p$  are indicated by red circles, and data points classified to belong to  $\mathcal{F}_f$  are indicated by blue circles. The decision boundary of the SVM that divides the stability region (performance region)  $\mathcal{F}_p$  and the instability region  $\mathcal{F}_f$  is indicated by a dotted line. The threshold  $J_T$ , which determines whether each sample is successful or unsuccessful, selects a value that ensures that only 10% of the total samples are classified as successful in the first iteration.

An increasing number of samples are used in each iteration. In each iteration, it can be seen that the newly added samples using active sampling are extracted near the performance boundary of  $\mathcal{L}$  in the previous iteration. Since the controller

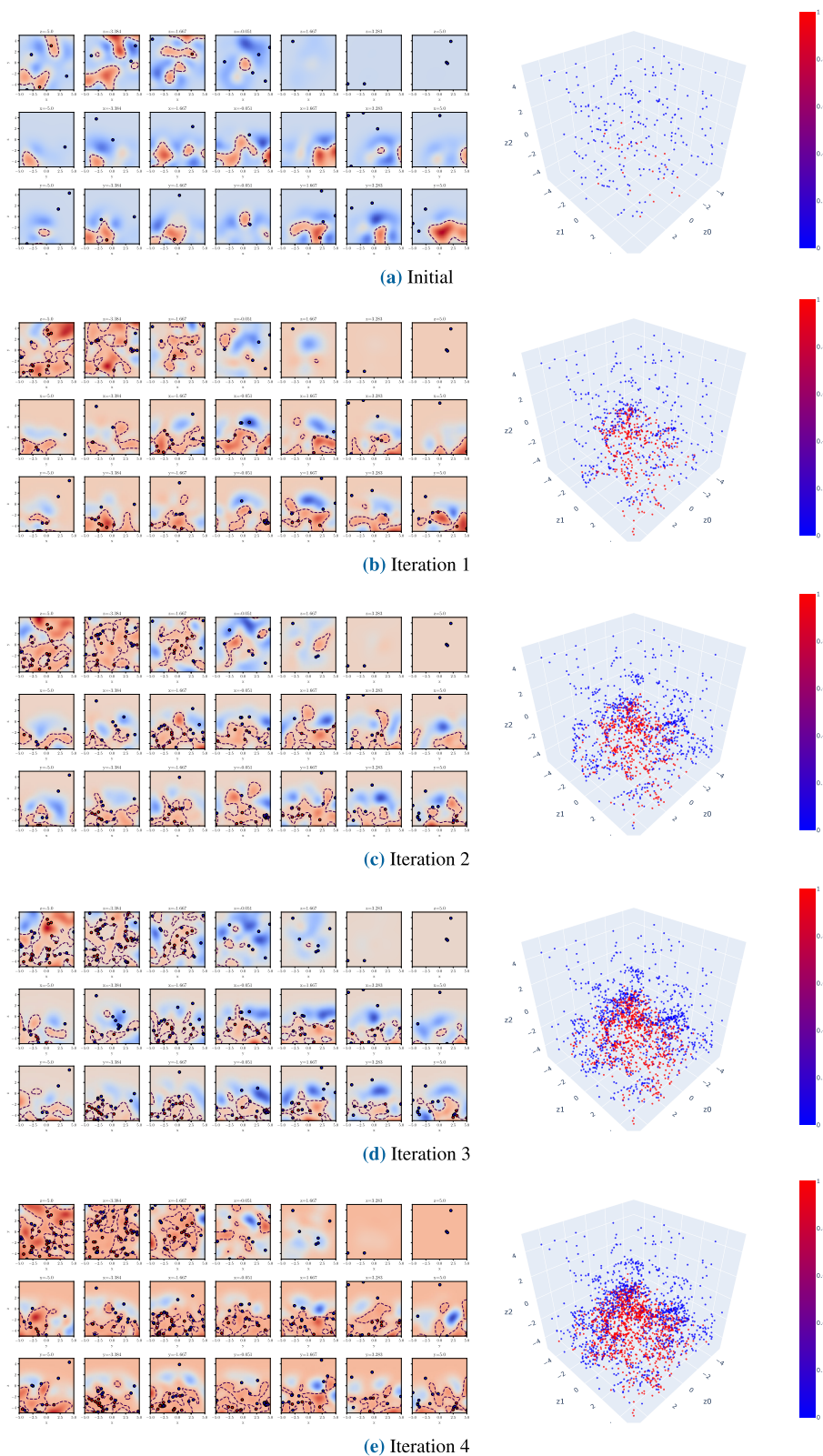


FIGURE 9. Time evolution of SVM decision boundary and labeled samples in latent space.

$\mathcal{C}$  is retrained for a given sample in each iteration, labels from Eq. (16) for a particular sample may vary from one iteration to another.

Figure 10 shows the feature vectors distributed in the latent space by coloring according to the log loss value of Eq. (15). It can be seen that the stability region of the controller is

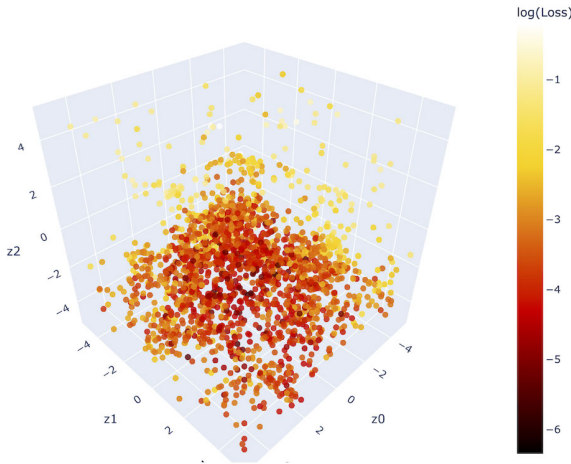


FIGURE 10. Log loss value of feature vector for trained controller.

TABLE 2. Parameters for training GVM.

Category	Parameter	(Range of) Value
GVM	$z$	$\in [-5, 5]$
	Time length (sec)	1.5
	Magnitude scaling coefficient	10
Plant	$\zeta$	0.25
	$\omega_n$	1
Controller	Number of repeat blocks	5
	saturation	$[-30, 30]$

determined according to the characteristics of the reference signal when comparing this result with the result in Fig. 7. In other words, the SVM is trained to classify the feature vectors of small magnitude and long wavelength belonging to  $\mathcal{F}_p$ , and the others belong to  $\mathcal{F}_f$ .

D. SIMULATION CASE STUDY

In this section, numerical simulations are conducted to investigate whether the control stability region of the optimal controller trained using the proposed framework has been properly derived. In general, it is difficult to analytically derive the control stability region with linear control theory due to nonlinearity, such as controller saturation. In this case, the stability region is often derived through Monte Carlo simulations. Since the control system considered in this study is a linear second-order system, it is possible to design a global optimal controller using a linear controller. However, it becomes difficult to obtain the boundary of the stability region of the global optimal controller because the entire vector space becomes the stability region. Therefore, controller saturation is considered in this study, and the boundary of the stability region can be obtained.

In this study, sinusoidal time-series data are used as a reference signal, so it is possible to check how the data are classified in the latent space, as shown in Fig. 7. In the figure, the magnitude of the signal shows a distribution proportional to the value of  $z_2$ . Figure 11 shows the simulation results

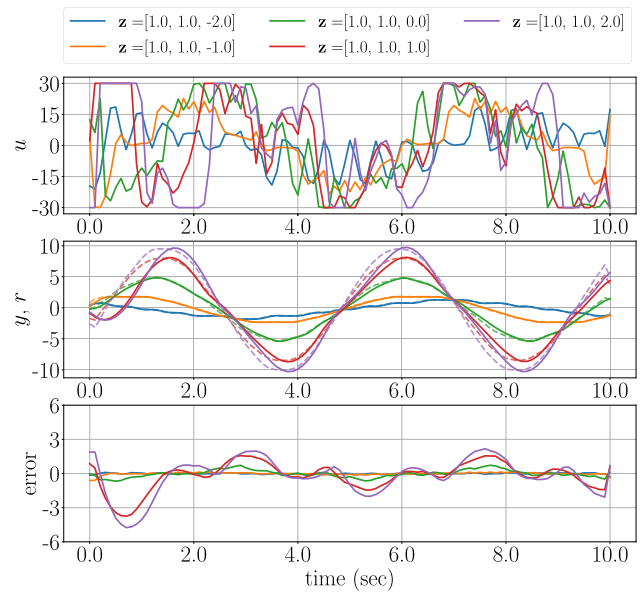


FIGURE 11. Simulation results for various magnitudes.

for various magnitudes. The tracking error is small for the reference signals of relatively small magnitudes, and the error increases as the magnitude increases.

In Fig. 7, the initial phase changes as the location of the sample point rotates along the circumference of a circle centered on the origin of the  $(z_0, z_1)$  plane. Figure 12 shows the simulation results for various initial phases. As expected, the magnitudes of the tracking error are similar for given reference signals of various initial phases.

In Fig. 7, the wavelength depends on the distance from the origin. Figure 13 shows the simulation result for various wavelengths. As the wavelength of the reference signal shortens, the tracking error increases because of the control input saturation.

When the distribution in the latent space according to the physical characteristics of the signal is known, it may be possible to predict or analyze the stability region based on the information. The stability region shown in Fig. 9 is distributed for areas with small magnitudes and long wavelengths and does not seem to be affected by the phase. However, in general, the shape of signals trained in GM may not be classified according to physical characteristics in advance. In contrast, in this case, by analyzing the stability region obtained through the GVM, the reference signal characteristics that make it easier for the controller to follow can be analyzed. Such characteristics may vary greatly depending on the control system.

From Fig. 7, it can be seen that the reference signals are classified in the feature vector space according to the characteristics of the time series. From Figs. 11-13, it is shown that the reference signal extracted from the latent space can be followed using a DNN-based controller, and its tracking performance depends on the physical characteristics of each reference signal. Figure 9 shows that the proposed

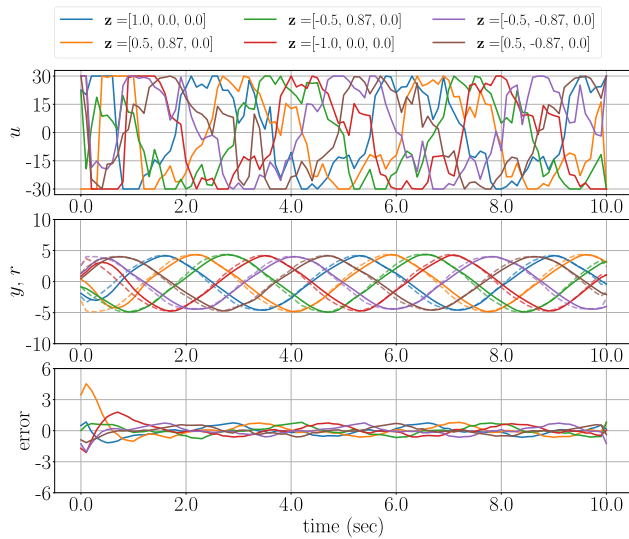


FIGURE 12. Simulation results for various phases.

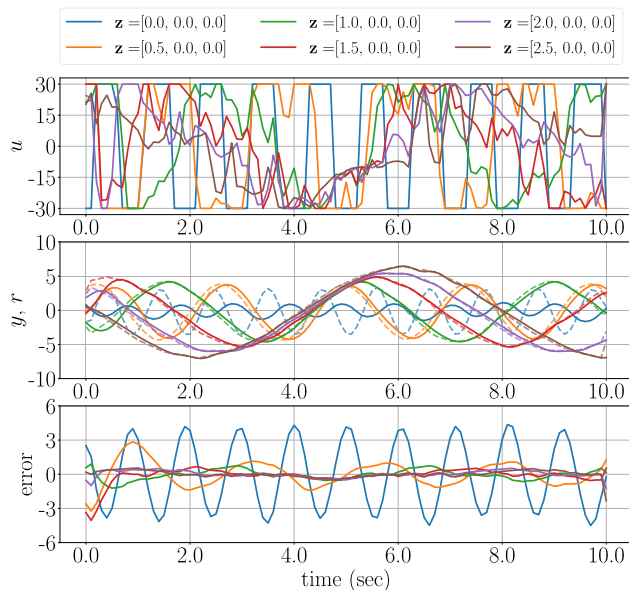


FIGURE 13. Simulation results for various wavelengths.

GVM framework further enhances the performance of the DNN-based controller while simultaneously deriving the performance domain of the controller. Eventually, the GVM framework can be used as an effective tool to analyze the performance of DNN-based controllers.

## VI. CONCLUSION

In this study, a novel design framework that can design a controller using a DNN structure was developed, and the stability and performance of the designed DNN-based controller were analyzed using statistical stability. The controller can be designed as a DNN of arbitrary structures. To derive a latent space for analyzing stability and performance, representation learning of time-series data was performed using a VAE. The controller can derive a stability region by training

the classification model in this latent space. The proposed GVM structure uses more data in every iteration to train the controller and the classification model, improving the controller's performance and deriving accurate performance regions. The proposed controller verification technique differs from the existing mathematical method and has high versatility, as there are no constraints on using a specific type of controller. Since this study focused on proposing a novel controller design framework, only state variables and reference values were used as input values for conciseness, as in the conventional controller design frameworks. As a further study, a DNN-based controller that uses more types of input data other than state and reference signals and the techniques proposed in this study can be implemented to evaluate the utility of the DNN-based controller and the proposed framework.

## REFERENCES

- [1] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York, NY, USA: OUP, 1998.
- [2] A. Bacciotti, *Stability and Control of Linear Systems*. Cham, Switzerland: Springer, 2019.
- [3] J.-J. E. Slotine, *Applied Nonlinear Control*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.
- [4] D. Kim, G. Oh, Y. Seo, and Y. Kim, "Reinforcement learning-based optimal flat spin recovery for unmanned aerial vehicle," *J. Guid., Control, Dyn.*, vol. 40, no. 4, pp. 1076–1084, Apr. 2017, doi: [10.2514/1.G001739](https://doi.org/10.2514/1.G001739).
- [5] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
- [6] D. E. Kirk, *Optimal Control Theory: An Introduction*. Englewood Cliffs, NJ, USA: Courier Corporation, 2004.
- [7] H.-S. Kim and Y. Kim, "Trajectory optimization for unmanned aerial vehicle formation reconfiguration," *Eng. Optim.*, vol. 46, no. 1, pp. 84–106, Jan. 2014, doi: [10.1080/0305215X.2012.748048](https://doi.org/10.1080/0305215X.2012.748048).
- [8] S. Lee and Y. Kim, "Optimal output trajectory shaping using Bézier curves," *J. Guid., Control, Dyn.*, vol. 44, no. 5, pp. 1027–1035, May 2021, doi: [10.2514/1.G005887](https://doi.org/10.2514/1.G005887).
- [9] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. Hoboken, NJ, USA: Wiley, 2012.
- [10] Y. Lee, Y. Kim, G. Moon, and B.-E. Jun, "Sliding-mode-based missile-integrated attitude control schemes considering velocity change," *J. Guid., Control, Dyn.*, vol. 39, no. 3, pp. 423–436, 2016, doi: [10.2514/1.G001416](https://doi.org/10.2514/1.G001416).
- [11] J. Lee and Y. Kim, "Neural network-based nonlinear dynamic inversion control of variable-span morphing aircraft," *Proc. Inst. Mech. Eng. G, J. Aerosp. Eng.*, vol. 234, no. 10, pp. 1624–1637, Aug. 2020, doi: [10.1177/0954410019846713](https://doi.org/10.1177/0954410019846713).
- [12] S. Lee, Y. Kim, Y. Lee, G. Moon, and B.-E. Jeon, "Robust-backstepping missile autopilot design considering time-varying parameters and uncertainty," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 6, pp. 4269–4287, Dec. 2020, doi: [10.1109/TAES.2020.2990819](https://doi.org/10.1109/TAES.2020.2990819).
- [13] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*, vol. 10. New York, NY, USA: Springer, 2013.
- [14] M. Kim, B. Jung, B. Han, S. Lee, and Y. Kim, "Lyapunov-based impact time control guidance laws against stationary targets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 2, pp. 1111–1122, Apr. 2015, doi: [10.1109/TAES.2014.130717](https://doi.org/10.1109/TAES.2014.130717).
- [15] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Netw.*, vol. 6, no. 6, pp. 801–806, Jan. 1993, doi: [10.1016/S0893-6080\(05\)80125-X](https://doi.org/10.1016/S0893-6080(05)80125-X).
- [16] Y. Yueneng and Y. Ye, "Backstepping sliding mode control for uncertain strict-feedback nonlinear systems using neural-network-based adaptive gain scheduling," *J. Syst. Eng. Electron.*, vol. 29, no. 3, pp. 580–586, 2018, doi: [10.21629/JSEE.2018.03.15](https://doi.org/10.21629/JSEE.2018.03.15).
- [17] S. Jung and Y. Kim, "Low-power peaking-free extended-observer-based pitch autopilot for morphing unmanned aerial vehicle," *J. Guid., Control, Dyn.*, vol. 45, no. 2, pp. 362–371, Feb. 2022, doi: [10.2514/1.G005998](https://doi.org/10.2514/1.G005998).

- [18] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, 2019. [Online]. Available: <https://direct.mit.edu/neco/article/31/7/1235-1270/8500>
- [19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020. [Online]. Available: <https://dl.acm.org/doi/10.1145/3422622>
- [21] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, Jan. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/9051780>
- [22] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2016, *arXiv:1511.05644*.
- [23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013. [Online]. Available: <http://ieeexplore.ieee.org/document/6472238/>
- [24] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-R. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Proc. Interspeech*, Chiba, Japan, Sep. 2010, pp. 1–4. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/binary-coding-of-speech-spectrograms-using-a-deep-auto-encoder/>
- [25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Stateline, NV, USA, 2012. [Online]. Available: <https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>
- [26] R. Socher, E. Huang, J. Pennin, C. D. Manning, and A. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, 2011. [Online]. Available: <https://papers.nips.cc/paper/2011/hash/3335881e06d4d23091389226225e17c7-Abstract.html>
- [27] S. Lee, Y. Lee, S. Lee, Y. Kim, Y. Han, and J. Park, "Data-driven capturability analysis for pure proportional navigation guidance considering target maneuver," *Int. J. Aeronaut. Space Sci.*, vol. 22, no. 5, pp. 1209–1221, Oct. 2021. [Online]. Available: <https://link.springer.com/article/10.1007/s42405-021-00387-7>
- [28] S. Lee, S. Lee, and Y. Kim, "Active sampling-based data-driven reachability verification for proportional navigation guidance law," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 1–6, 2019, doi: [10.1016/j.ifacol.2019.11.060](https://doi.org/10.1016/j.ifacol.2019.11.060).
- [29] J. F. Quindlen, U. Topcu, G. Chowdhary, and J. P. How, "Active sampling-based binary verification of dynamical systems," in *Proc. AIAA Guid., Navigat., Control Conf.*, Kissimmee, FL, USA, Jan. 2018, p. 1107, doi: [10.2514/6.2018-1107](https://doi.org/10.2514/6.2018-1107).
- [30] J. F. Quindlen, U. Topcu, G. Chowdhary, and J. P. How, "Active sampling for constrained simulation-based verification of uncertain nonlinear systems," 2017, *arXiv:1705.01471*.



**SUWON LEE** received the B.S. and Ph.D. degrees in mechanical and aerospace engineering from Seoul National University, Seoul, Republic of Korea, in 2015 and 2021, respectively.

He is currently an Assistant Professor with the Department of Future Mobility, Kookmin University. His research interests include planning, guidance, and control systems of missiles, UAVs, and UAMs.

• • •