Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-97625

A DEEP LEARNING BASED APPROACH TO PREDICT SEQUENTIAL DESIGN DECISIONS

Molla Hafizur Rahman
Department of Mechanical Engineering
University of Arkansas
Fayetteville, AR

Charles Xie
Concord Consortium
Concord, MA

Zhenghui Sha[1]
Department of Mechanical Engineering
University of Arkansas
Fayetteville, AR

ABSTRACT

During a design process, designers iteratively go back and forth between different design stages to explore the design space and search for the best design solution that satisfies all design constraints. For complex design problems, human has shown surprising capability in effectively reducing the dimensionality of design space and quickly converging it to a reasonable range for algorithms to step in and continue the search process. Therefore, modeling how human designers make decisions in such a sequential design process can help discover beneficial design patterns, strategies, and heuristics, which are important to the development of new algorithms embedded with human intelligence to augment computational design. In this paper, we develop a deep learning based approach to model and predict designers' sequential decisions in a system design context. The core of this approach is an integration of the function-behavior-structure model for design process characterization and the long short term memory unit model for deep leaning. This approach is demonstrated in a solar energy system design case study, and its prediction accuracy is evaluated benchmarked on several commonly used models for sequential design decisions, such as Markov Chain model, Hidden Markov Chain model, and random sequence generation model. The results indicate that the proposed approach outperforms the other traditional models. This implies that during a system design task, designers are very likely to reply on both short-term and long-term memory of past design decisions in guiding their decision making in future design process. Our approach is general to be applied in many other design contexts as long as the sequential design action data is available.

Keywords: Sequential decision making, Deep learning, Artificial neural network, Engineering design process, CAD.

NOMENCLATURE

ANN	Artificial Neural network
FFNN	Feed Forward neural network
HMM	Hidden Markov model
LSTM	Long short-term memory
MM	Markov model
REP	Repetitive model
RNN	Recurrent neural network

1. INTRODUCTION

Design involves an iteratively searching process of design space for desired solutions. In such a search process, designers make sequential decisions that involve the selection of design actions and/or determination of design parameters so that they can get the best output. Since uncertainties are always accompanied with the search process, strategies of tradeoff and decisions on when and where to explore and exploit design space are important to the quality of design outcomes and the resources needed to achieve the outcomes. For example, in our previous sequential design behaviors study [1], we integrated Markov chain and unsupervised clustering methods to explore designers' sequential behaviors and observe that designers follow certain design patterns, such as a high frequent design synthesis-related operations, in engineering systems design that has many coupling design variables and exhibits large design uncertainties.

These sequential decision-making strategies are often the key features that differentiate expert designers and novice. These strategies are also the essences of human intelligence in design and the reason why sometimes human designers are effective in reducing the dimensionality of design space and quickly converging it to a reasonable range for search. For example, human shows very different strategy or patterns as compared to computational algorithms [2]. Also, it is found that human designers are more effective than algorithms doing search for

[1] Corresponding author: zsha@uark.edu

promising design candidates in certain situations, such as early-stage systems design with large and high-dimensional or discrete solution space [3].

Therefore, modeling and machine learning of human sequential design decisions is of great interest and has potential impact to engineering design process and design automation. On the one hand, successful modeling of sequential decision-making can help discover quantifiable design patterns and strategies which are useful to existing computational design framework by embedding human intelligence. Early computational design mainly solves parametric and configuration design tasks through solver framework or linear mathematical programming. On the other hand, the discovery of beneficial design patterns from expert designers can be used to train novice designers. Moreover, computational modeling of sequential decision-making can help build artificial design agent which can be used in CAD systems and work collaboratively with human designers to improve design outcomes by reducing unnecessary design iterations and reinforce useful iterations.

However, modeling and predicting human sequential design decisions is challenging. First, human decisions are a result of mental process that is hidden, implicit and sometimes tacit [4] because they are difficult to be transferred in an explicit way like writing or verbalizing. Second, design decisions are intricate, particularly in systems design scenarios where design process often spans over a long period and designers' decisions involve with multiple interdependent variables; thus their decisions are highly correlated in the time sequence. Moreover, in complex design, patterns are weak and strategies vary across different designers. So, it is very challenging for traditional sequential learning models such as Markov model, autoregressive integrated moving average (ARIMA), etc., to discover prominent design patterns for the purpose of prediction. To address these challenges, the **objective of this study** is to computationally model and predict human designers' sequential decisions in the context of engineering systems design. The **research question** we aim to answer is:

What is the effect of past design decisions on predicting designers' future sequential decision making?

To achieve the objective, we develop a deep learning based approach with recurrent neural network (RNN). Artificial neural network (ANN) which mimics the human brain has been proved to be capable in machine learning sequential behavioral patterns in various fields recently such as natural language processing [5], healthcare [6], image recognition [7], finance [8], etc. Despite its success in these fields, the capability of ANN and deep learning in design research on sequential design decisions has not been explored. This is one of the motivations of this study.

Particularly in this paper, we focus on the systems design problem that involves parametric and configuration design decisions. In real design scenario, especially in industry from product design to product manufacturing pipeline, configuration and parametric design plays a vital role where designers make decisions on which design components to use and what design parameter values to choose in order to construct the desired design artefact that satisfies a set of given constraints. These decisions can be recorded continuously in time scale as a sequence of actions taken. In the proposed approach, these sequential design actions are transcribed by the function-behavior-structure (FBS) design process model in order to obtain the sequential data that depicts the design process stages of each designer. Then we use ANN on those processed data to train a deep learning model that can be used, in turn, to predict a designers' design sequence. To gauge the performance of the developed model, we adopt accuracy metrics including parentage of prediction correctness and the area under the receiver operating characteristics (ROC) curve. These metrics are used to compare our model with the existing models of sequential design decisions in the engineering design field, such as Markov Chain model and Hidden Markov Chain model.

The **major contributions** of the work are twofold: 1) An approach that integrates FBS-based design process model and RNN to model, learn and predict human sequential design decisions in the context of systems design. 2) An exploration and examination of the capability of deep learning models in predicting human sequential design decisions as compared to commonly-used models in engineering design field.

The remaining paper is organized as follows. In Section 2, we discuss the state-of-the-art research on sequential design decision making and deep learning of sequential data. In Section 3, we present our research approach and briefly introduce the technical backgrounds on different types of ANN models. In Section 4, a case study is presented on predicting designers' sequential decisions in a solar energy system design project with the proposed approach. The details of the design experiment and data collection are provided in this section too. In Section 5, comparative study is performed, the results are presented and discussed. Finally, we conclude the paper with closing insights and discuss our future work in Section 6.

2. LITERATURE REVIEW

In this section, we first discuss the relevant studies on sequential decision making in the engineering design field. Next, we review the current work that leverages deep learning models in training and predicting sequential data in different areas.

2.1 Studies on sequential design decisions

Several studies have been done in engineering design field to explore the sequential patterns, optimize the sequence of design task and finding heuristics from the sequence learning. Particularly, a large number of works have been conducted based on Markov chain model for sequence learning. For example, in order to compare designers' design behaviors, function-behavior-structure ontology and first-order Markov chain [9] was used. Second-order Markov chain is also used to explore the effect of previous experience and design knowledge on design sequence [10]. Moreover, in order to explore the designers' sequential learning strategy, McComb et al. [11] use Markov model in a truss design problem. Their results indicate that the first-order Markov chain better represents designers' action sequences. In a later study, they use hidden Markov model

(HMM) to study the patterns of sequential design state in the same design problem. They found four hidden states in the configuration design and observed that designers used the first two states to topology operation, third state to spatial, and the fourth state to parameter operation. The trained HMM model is utilized to compare the design performance (in term of strength to weight ration of the truss design) among designers [12].

To study human sequential design behaviors in different design scenarios, there have been studies conducted based on Bayesian Optimization (BO) framework. For example, in order to mimic the human searching strategy, Sexton and Ren [3] develop a searching process using BO algorithm which can replace human solvers from a design process. Sha et al. [13] also integrated Weiner-process BO with game theory to develop a model for estimating designers' sequential decisions while two designers compete with each other for monetary reward.

Prior studies on sequential design processes have been also focused on project task level in support of product development and project management. For example, design structure matrix (DSM) [12-14] has been used to study task sequencing for identifying the sequence that minimizes expected project completion time. Some other work has been grounded in theoretical processes. For example, Miller et al. [14] use multi-objective formulations to study the design process sequentially advancing through to smaller sets of alternatives using models of increasing fidelity. In addition, optimization approaches, such as the expected value of perfect information [15], genetic algorithm [16], and optimal learning [17], have been utilized in studying optimal design sequences. However, these studies are fundamentally different from this study in that they formulate a design problem and cast it into a sequential decision process to be optimized with normative models. In this study, however, we focus on sequential decision-making of *human* designers. It is about the actual actions that designers sequentially take. By modeling and analyzing such a design sequence at a fine-grained resolution, it is expected that insights and new knowledge regarding the design process can be obtained.

2.2 Deep learning to model sequential data

In recent years, deep learning techniques have shown their promise in predicting complex sequential data. For example, customer behavior is essentially a sequential data where they interact with market continuously over time. Santolaya et al. [18] leverage historical customer interaction data to predict the items that a customer may buy in future. This study shows that recurrent neural network (RNN) can successfully model the sequential customer data.

A large number of studies have been performed in order to recognize speech and text patterns using ANN due to the sequential nature of those events. Many early works are focused on integrating feed forward neural network (FFNN) with HMM [19] to discover the patterns of speech. Also, RNN-based approach has been used to in phone probability estimation which aims to recognize phone in speech in terms of sound or gesture of a specific language [20]. Recently, long short term memory (LSTM), a special variant of RNN is introduced [21] and has shown better performance in both handwritten text recognition [22] and speech recognition [23] than previously used sequential learning models such as HMM.

In the medical field, deep learning-based algorithms, e.g., FFNN and RNN/LSTM have been utilized to treat sequential data in different application scenarios. For example, both FFNN [24] and RNN [25] have been used to explore the complex patterns of gene expression. In biomedical sequential signal processing problems, such as electrocardiogram (ECG) signal and brain decoding [26], the success of many deep learning models is also mention-worthy.

Deep learning is also used to study human daily sequential activities such as walking, standing, eating, etc. For example, Baccouche et al. [27] develop a fully automated deep learning model by integrating convolution neural network (CNN) and RNN to classify human daily routine activities without any prior knowledge. Apart from this areas, deep learning is successfully applied in online fraud detection [28], identifying discrimination [29], and predicting financial time series [30] through the use of users' online sequential log data.

Although researches have been conducted on modeling and predicting sequential data using deep learning models in many disciplines, to the best of our knowledge, no studies have been conducted yet to handle the sequential design data using deep learning in engineering design field. Design sequential data follows some unique patterns which are quite different from the sequential data in other fields. For example, designers perform conceptual design at the beginning of a design project and involve design actions and strategies that would be totally different from the ones in the embodiment design phase. Also, some of the design tasks may be iterated within one design phase but not in another. These patterns are not the same as the ones typically seen in natural language or human daily routine activities where deep learning approaches normally success. So, even though deep learning shows optimistic results in many

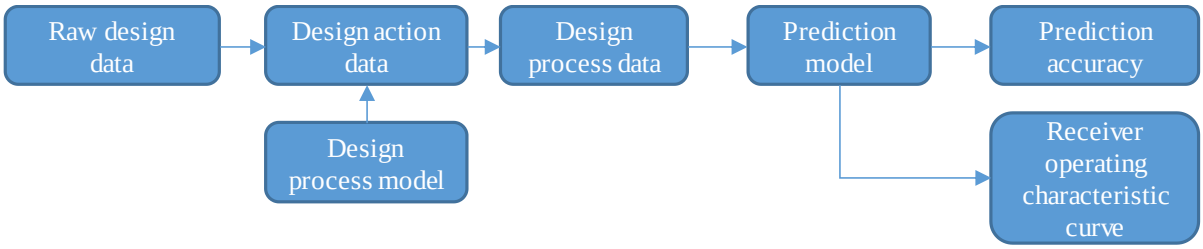

FIGURE 1: The overall research approach

fields, its predictive power is still unexplored in studying sequential design behaviors.

3. Research Approach and Technical Background

In this section, first, we introduce the research approach of this study. Next, we introduce the technical background regarding the deep learning models adopted in our approach.

3.1 The research approach

The approach starts with the raw data collection of designers' sequential design decisions from different sources such as the action logger of computer-aided design (CAD) software, interviews of designers, design documents, etc. The raw data contains the details of human design behaviors (i.e., design actions) as well as design artefact's information, such as values of design parameters, simulation results, etc. In this study, we only extract the design actions which are only design-related, for example, in a CAD environment, adding a new component or editing that component. But changing the camera view is not considered as a design action in this study. Designers act based on given requirements and constraints, thus these actions essentially reflect their design thinking and strategies during a design task. Next, we apply design process model to convert the design actions into design process data. Design process model consists of a series of design stages that characterize a design process. This treatment transforms the action space into a design process space, which can significantly reduce the dimensionality of the sequential data. This leads to better interpretation and understanding of designers' design process and sequential design thinking (see Section 4.2 for details). Then, we use the sequential design process data to train deep learning models and predict the next immediate design action category (i.e., the design stage defined by a design process model) based on the trained models. In this study, we use ANN models including FFNN and RNN to implement the deep learning approach. Finally, we evaluate the predictive performance of these models and compared them with several commonly used models with different metrics, such as training accuracy, testing accuracy, and the receiver operating characteristics (ROC) curve, and area under curve, at both aggregated level and design process stage level (see Section 5 for details). Figure 1 depicts a schematic diagram of the overall approach used in this study.

3.2 Artificial neural network

An artificial neural network (ANN) is a biological inspired mathematical model, introduced by Rosenblatt [31] that mimics human brain to learn from the input dataset and produce the predictive outcomes based on that dataset. An ANN consists of artificial neurons which are known as nodes. The process starts with passing weighted inputs to the artificial neurons. This means each input is multiplied by individual weight. Then all the weighted input is summed with an adjustable unit bias that can help the artificial neural in a learning process. Finally, the sum of the weighted inputs and the bias are passed through the activation function to produce the final output.

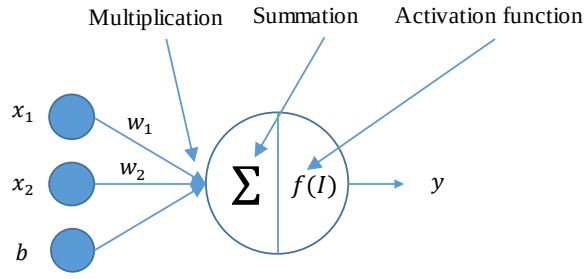

FIGURE 2: An artificial neuron and the basic building blocks of an artificial neural network

Depending on the problem studied, the activation function can be varied. Commonly used activation functions are step function, linear function, non-linear sigmoid function, non-linear hyperbolic tangent function [32]. Combined with these activation functions, the output of an artificial neural is mathematically represented as follows:

$$y = f(\sum_{i=1}^{m} w_i \cdot x_i + b), \qquad (1)$$

where y is the output, x_i is the input value, w_i is the weight value, b is the bias and f is the activation function. Figure 2 shows an artificial neuron with its operations including multiplication, summation and activation. In reality, a single node is not efficient to solve a complex problem. For this reason, ANN commonly consists of large numbers of nodes that interact with each other through their weighted interconnections which eventually builds the architecture of a particular ANN. The power of the ANN mainly depends on these architectures, such as the number of the nodes and how each of the nodes operates [33]. In the following sections, we briefly introduce the two of the most important ANN architectures that adopted in this study.

3.2.1 Feedforward neural network

ANN that follows the feedforward architecture means that the information follows one direction from input to output with no back loops. In addition to the input layer and output layer, FFNN may have single or multiple hidden layers, as shown in Figure 3. When FFNN has only one layer between input and

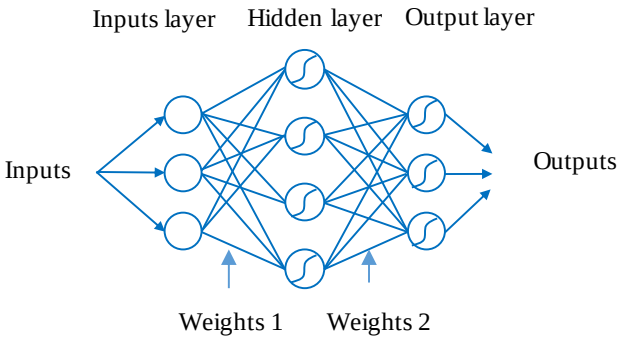

FIGURE 3: The architecture of a feed forward neural network. Nodes are neurons and links represent weights. Information flows from left (input) to right (output).

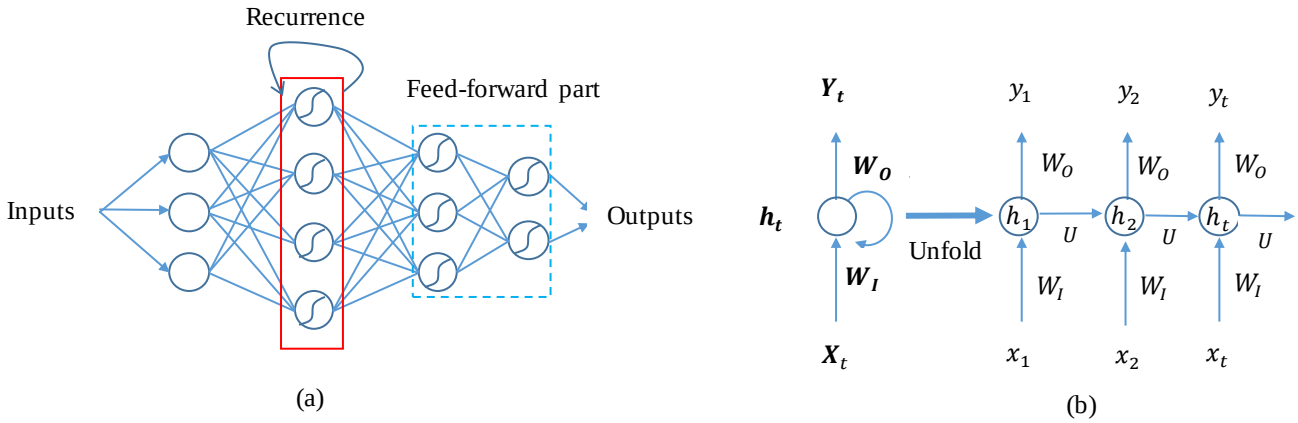

FIGURE 4: (a) Standard recurrent neural network architecture with feedback loop; (b) Unfolded recurrent neural network.

output, it is known as the single-layer perceptron. An FFNN with more than one hidden layer including the output layer is called multilayer perceptron. An FNN with a single hidden layer between the input layer and the output layer (as the one shown in Figure 3) is often sufficient to be universal function approximation [34]. However, deep neural network with additional hidden layers outperforms this shallow model. A standard depth of deep neural network (i.e., the number of hidden layers) may vary from two or three to even one thousand [7].

3.2.2 Recurrent neural network

The structure of an RNN architecture is similar to that of an FFNN. The only distinction is that there is no restriction on back loops. So, the information not only passes in one direction forward but also does it flow backward, called *recurrence* (see Figure 4). This feature allows RNN to create a hidden state which carries information from the previous time steps. Figure 4(a) shows the architecture of an RNN. The feedback arrow indicates the recurrence of the hidden layer. The two layers in the dot-line box show the fully connected feed-forward information flow. An RNN can be unfolded, and Figure 4(b) shows an unfolded structure, where each input node of the input layer takes input as $\boldsymbol{X}_t = [x_1, x_1, \ldots . x_t]$. These input units are connected to the hidden nodes with the weight matrix $\boldsymbol{W_I}$. The hidden layer associated with the hidden nodes can be represented as $\boldsymbol{h}_t = [h_1, h_2, \ldots \ldots, h_t]$. The hidden nodes are also connected with a weight value, $\boldsymbol{U}$. Finally, the hidden layer is connected to the output layer $\boldsymbol{Y}_t = [y_1, y_2, \ldots ., y_t]$ via the weight matrix $\boldsymbol{W}_O$. The output of the RNN can be presented with the following equation,

$$\boldsymbol{Y}_t = f(\boldsymbol{W}_0 \boldsymbol{h}_t), \tag{2}$$

where f indicates the activation function (logistic sigmoid or hyperbolic tangent functions are typically used). There are several types of recurrent unit or node. The most common and simplest RNN unit is called *simple recurrent unit* with the hidden layer defined as follows,

$$\boldsymbol{h_t} = \sigma(\boldsymbol{W}_I \boldsymbol{X}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{b}_H), \tag{3}$$

where, $\boldsymbol{W_I}$ and $\boldsymbol{U}$ are the weights as mentioned earlier, σ is the sigmoid activation function and $\boldsymbol{b}_H$ is the bias of the hidden layer. In order to compute the value of the current hidden node $\boldsymbol{h}_t$, the output from the previous time step is combined with the input of the network $\boldsymbol{X}_t$.

Although RNN can be used for capturing long-term dependencies, *simple recurrent units* are not effective in this task due to vanishing gradient problem [35]. To solve this problem, Hocreiter et al. [21] proposed the long short term memory units (LSTM), a special type of mechanism where information flow is controlled by three different *gates* namely input gate, forget gate and output gates. The corresponding equations described by Graves et al. [23], are as follows,

$$\boldsymbol{i_t} = \sigma(W_{xi}\boldsymbol{x}_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \tag{4}$$
$$\boldsymbol{f_t} = \sigma(W_{xf}\boldsymbol{x}_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f, \tag{5}$$
$$\boldsymbol{c_t} = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \tag{6}$$
$$\boldsymbol{o_t} = \sigma(W_{xo}\boldsymbol{x}_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \tag{7}$$
$$\boldsymbol{h_t} = o_t tanh(c_t), \tag{8}$$

where i_t, $\boldsymbol{f_t}$, and $\boldsymbol{o_t}$ are input gate, forget gate and output gate, respectively. Their associated weight matrices are $\boldsymbol{W}_{xi}$, $\boldsymbol{W}_{xf}$ and $\boldsymbol{W}_{xo}$, respectively. $\boldsymbol{c_t}$ is the memory cell which stores memory from the previous time step. W_{hi} is the hidden-input gate weight matrix. The bias terms (i.e., b_i, b_f, b_c and b_o) are added to each of the corresponding gates as well.

LSTM is more widely used than simple RNN in many domains for its capability of modeling long-term dependencies. To study to what extent the past decisions of designers' can influence their future decision-making, we use LSTM as a representative of RNN in our study. In the next section, we overview a case study for data collection in order to implement these models.

4. Predicting sequential design processes in solar energy systems design: A case study

In this section, we present a case study on solar energy systems design and implement the proposed approach to predicting designers' sequential design decisions. First, we

FIGURE 5: An example of the energy-plus home design

introduce the design experiment conducted for data collection. Next, we present the collected data and introduce the methods for processing it.

4.1 The design context

In order to collect sequential design behavioral data, we conducted a series of design challenges on real-world engineering design problems. The challenges were held at the University of Arkansas in 2018. Both undergraduate and graduate students from engineering disciplines participated in these challenges. In this study, we mainly adopt the data collected from one challenge where the students were asked to design a solarized energy-plus home in Texas with the budget of $200,000 (see Figure 5).

The design objective is to maximize the annual net energy (ANE) given a budget. The design requirements and constraints are provided to the participants so they start the project with more focus on configuration design and parametric design. To keep the design complexity at a manageable level, the design variables are mainly related to the nine components that have direct impact on the design objective (see Table 1). More details on the design experiment and design problem are discussed in our previous article [1].

Students' designs were conducted within a computer-aided design (CAD environment, called Energy3D. Energy3D is a full-fledged CAD software specially built for solar systems design [36]. Energy3D has several unique features such as interactive visualization, high-fidelity simulation, and built-in financial evaluation. These features can help designers effectively explore and exploit the design space. Moreover, Energy3D has a non-intrusive data action logger. That means designers are not aware of the data collection process, and this help reduces participants' cognitive burden that are normally introduced in experimental settings. As a result, the data that reflects designers thinking and decision-making can be less biased. Energy3D sort and log the sequential design action data at a fine-grained level. For example, it logs every performed action and intermediate artefacts in every 20 seconds [37]. This high-resolution data provides us with a large amount of data that are essential to implementing deep learning models.

4.2 Data collection and preprocessing

Energy3D collects the continuous flowof design action data in JSON format which includes time-steps, design actions, design parameter values and simulation results. In this study, a total of 38 engineering students participated in this design challenge. Among them, 20 students are undergraduate students and 18 are graduate students. 29 students are from Mechanical Engineering. On average, the design action log records 1500 lines and 220 intermediate files per student. Example of a line of the design action log is presented below:

{"Timestamp": "2017-11-16 11:01:23", "File": "EnergyPlusHome.ng3", "Add SolarPanel": {"Type": "SolarPanel", "Building": 2, "ID": 58, "Coordinates": [{"x": 30.785, "y": 1.185, "z": 37.009}]}}

We ignored the actions (i.e., "*camera*", "*add human*") that does not have effects on the design outcomes (i.e. ANE). After removing those irrelevant actions, there are 300 actions per participant on average and 115 actions are unique. Analysis of such a high dimension action space would yield results hard to interpret. To better understand the design process in this case study and designers sequential decision-making strategies, the function-behavior-structure (FBS) based design process model [38] is adopted. According to the FBS model, a coding scheme can be established to transcribe different types of design actions to seven design process stages including *Formulation* (F), *Analysis* (A),*Evaluation* (E), *Synthesis* (S), *Reformulation 1* (R1), *Reformulation 2* (R2), and *Reformulation 3* (R3) (see details in [1] about how each type of design actions corresponds to the design process stages in FBS model). This dimension reduction also help us to reduce the effect of "curse of dimensionality" [39]. Curse of dimensionality occurs when there is large number of features but the total dataset is limited.

TABLE 1: The design requirements and constraints of the solarized energy-plus home

Components	Requirements
Story	1
Number of windows	> 4
Size of windows	>1.44 m^2
Number of doors	≥1
Size of doors (Width × Height)	>1.2 m × 2m
Height of wall	>2.5m
Distance between ridge to panel	>0

Design process stages	F	A	E	S	R1	R2	R3
Formulation	1	0	0	0	0	0	0
Analysis	0	1	0	0	0	0	0
Reformulation 1	0	0	0	0	1	0	0
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
Analysis	0	1	0	0	0	0	0

Figure 6: One hot vector representation of a sequence.

Given a set of actions data, we need to convert or encode the sequences in a way such that it can be implemented as an input in neural network. The most popular encoding technique is known as "one hot encoding" [40]. One hot encoding transforms a single variable of n observations with m distinct variables into m binary variable with n observations. Each observation indicates the presence (1) for the corresponding position of that variable and absence (0) in all other dimension. Figure 6 shows the one hot vector presentation of a sequence.

5. RESULTS AND DISCUSSION

In this section, we present the results of the LSTM and FFNN models and compare them with the models that are commonly used in existing literature, such as the Markov model (MM) and hidden Markov model (HMM). Additionally, we develop a repetitive model (REP model) for comparison because we found from our previous study [1] that designers quite frequently repeat the previous design action in the CAD environment. For example, we found that on overage 53.6% of design actions were simply repeating the action just one step before. So, in the REP model, we simply use the average percentage of occurrence of each design stage as the model to predict the next design stage with the highest percentage value. Finally, a random model is developed as the benchmark for all the models investigated. The purpose is to examine whether the design sequences indeed follow certain patterns or just shows randomness. For our study, we have seven design process stages. Then the prediction of the next stage will be a random selection of one process stage from seven following a uniform distribution. On average, every process stage has a probability of 1/7 to be selected.

In the following two sections, we first evaluate the performance of different models in terms of training accuracy and prediction accuracy regardless of the category of design actions (i.e., the design process stages defined by the FBS model). Next, we perform in-depth analysis on how accurately each design process stage in the next step can be predicted and compare the performance of different models using the metrics of receiver operating curve (ROC) and the area under ROC curve (see Section 5.2 for details).

5.1 Evaluation of model performance at the level of entire sequence

To validate the models, we adopt k-fold cross-validation [41] technique where we divide our data into 5 folds. First, we use any 4 folds to train the models and leave the remaining fold for validation purpose. Next, we train the models on a new combination of 4 folds including the previously withheld fold and validate the model again with the remaining one. In this way, we iterate through all over the 5 rounds. An illustration of the 5-fold cross-validation method is shown in Figure 7.

Keras deep learning library [42] is used to run the HMM, FFNN and LSTM model, and we programmed by ourselves for the MM. While going through each of the rounds, training data set performs forward pass and backward pass (a.k.a. backpropagation) [43] in order to update the models' parameters

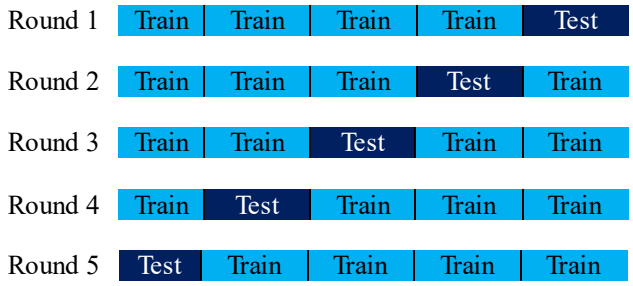

FIGURE 7 Training and testing data split according to 5-fold cross validation technique

(including both weight values and bias values). When forward pass and backward pass complete passing the whole data set, it's called one *epoch*. During testing, we predict the next action (a_{t+1}) by passing the previous actions from time 0 to t as the input into the trained model. So, if a design sequence has n actions, then $n-1$ predictions will be made. Then, by comparing with the real observation of a design sequence, we count the total number of correctly predicted actions (n_{cp}) and divide it by the total number of predictions, i.e. $n-1$. In this way, we can get the prediction accuracy of that model in every epoch. In this study, only the prediction accuracy of the last epoch (when the model is fully trained) in each round is taken and the average from five rounds is used as the metric for evaluating a model's predictive power. The mathematical expression of this metric is as follows:

$$prediction\ accuracy = \frac{1}{R}\sum_{i=1}^{R}\left(\frac{n_{cp}}{n_i^{max}-1}\right), \qquad (9)$$

where, R denotes the number of rounds for cross validation. $R = 5$ in this study. n_i^{max} is the maximal number of actions of the design sequence (i.e., the length of the longest design sequence) in round i. In our experiment, we found all the models converge after 20 epochs. The models were trained by stochastic gradient descent [44] algorithm with a learning rate of 0.1. This learning rate is determined by trial and error for producing the best accuracy. Section 5.3 presents a sensitivity analysis on this hyperparameter.

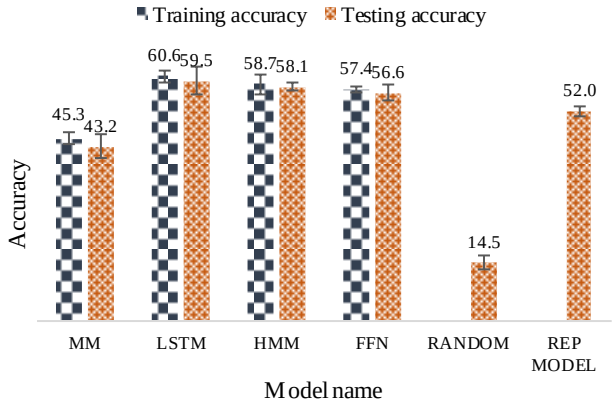

FIGURE 8: Training and testing accuracy of 4 models.

MM provides the prediction of the design process state in the next step based on the state in the current time step. With the design sequence as input, training of a MM will produce a 7×7 transition probability matrix because the FBS design process model contains seven design process stages which represent seven states in MM. Each of the entry of the matrix defines the probability of one design stage transitioning to the next design stage. We follow the same structure in Figure 7 to train and test MM. The trained model (i.e., the transition probability matrix) is an aggregation by averaging the matrices obtained from every designer in the training dataset. When testing a MM model, for every given FBS design process stage in a design sequence, the MM will predict seven probabilities of design stages following that given stage, and the one with the highest probability is picked for comparing with the real data, and then calculating the prediction accuracy. Markov model is not associated with any external parameters. So, we calculate the prediction accuracy just based on the transition probability matrix.

Figure 8 shows a comparison of the prediction accuracy of both training data and testing data among all the models. The baseline model, i.e., the random model, shows the least accuracy of 14.5 % with the standard deviation of 1.66%. The accuracy of the REP model is much higher than the random model which is about 52%. This is because REP model is developed based on the simple repetition process in design and it is observed that when designers were working on this design problem using CAD software, many of them repeated their previous action very frequently and the REP model captures such a pattern. We present only the prediction accuracy for the random model and the REP model because these models are not built with data and the training process is not needed.

It is shown from Figure 7 that both MM and HMM yields better performance than the random model with the prediction accuracy of 43.2% and 58.1%, respectively. This indicates that designers' actions indeed follow certain patterns and are not random. In MM, each action is dependent only on the action of the previous step. As a consequence, MM does not encode "memory" of past events in the prediction. On the other hand, the inclusion of hidden-state architecture in HMM allows it to "remember" a few past states. As in design, designers do have to refer to past information in guiding their future design decisions, the successful modeling of past information into the hidden state may be the reason why HMM has significant higher prediction accuracy (the average is 58.1%) than those of MM and REP model. This observation echoes many of the existing studies on the comparison between MM and HMM, and the conclusion that HMM outperforms MM [45] is very likely due to the reason that HMM can better model the interdependencies between past states and current state.

We also observe that HMM outperforms the FFNN model. FFNN gives the average prediction accuracy of 56.6% (with the standard deviation of 1.79%) that is 1.5% lower than that of the HMM. The ability of passing information from the previous state to the current hidden state makes HMM a better predictor than FFNN in this study. FFNN does not essentially have hidden state as it does not consider feedback loop in hidden layers and there not connection between hidden units. (see Section 3.2.1 for the architecture of FFNN).

Among all of the models, LSTM produces the highest prediction accuracy with an average value of 59.5% and the standard deviation is 3.49%. This implies that during the systems design process, there is a strong correlation between designers' previous actions and their next immediate actions. For example, we observe that one designer, most of the time, analyzes "*Building cost*" after adding several new components, such as "*Add window*" → "*Edit window*" → "*Building cost*", and again, "*Add Rack*" → "*Edit Rack*" → "*Add SolarPanel*" → "*Building cost*". This infers that after adding new components, this designer started configuring the related components to try improving the design performance and check if the total cost is still within the budget. Since LSTM leverages longer "memory" of past events and their interconnections in predicting future states, it yields the best performance in this study. LSTM's highest prediction accuracy also implies that designer doesn't not only recall short-term memory (like what MM does), but also use long-term memory information in their design process.

The results of prediction accuracy presented in Figure 8 demonstrate the performance of the models that can "recall" information from more than one-time step (such as the LSTM and the HMM) and the models that recall no memory (such as the MM, the REP model, and the random model). Generally, the models that encode longer "memory" in their architecture generally performs better in predicting designers' future actions. The possible reason is that in systems design, there are many design variables that are interdependent, and designers may not be able to immediately understand such a complex relationship. One method that can help understand the interdependencies among design variables and their effects on design objective is to continue changing the variables and then perform simulations to see how it would affect the objective value. Since there are multiple variables in this design case study (see Table 1), designers often perform a series of configurations (such as "*Add wall*"→"*Edit wall*"→"*Edit Foundation*"→"*Add Roof*"→"*Add Rack*"), and then perform ANE analysis cost evaluation. This process may have to be repeated several times in order to find the best configuration/combination of design variables for the desired objective. Such a pattern can reflect designers' exploration-exploitation strategies for design tradeoff (i.e., the sequential decision-making strategy) and their design heuristics. From the results, the hidden states of LSTM and HMM work as a memory unit seem to well capture those patterns.

5.2 Evaluation of model performance at each category of design actions

In order to understand the models' performance at a finer resolution, we check how well each model can predict each category of design actions, i.e., the design process stage defined by the FBS design process model. To achieve this, we adopt receiver operating characterizes curve (ROC) [46] as the method which evaluates model's two operating characteristics (true positive rate and false positive rate) on each design process stage under different threshold values from 0 to 1. After obtaining the

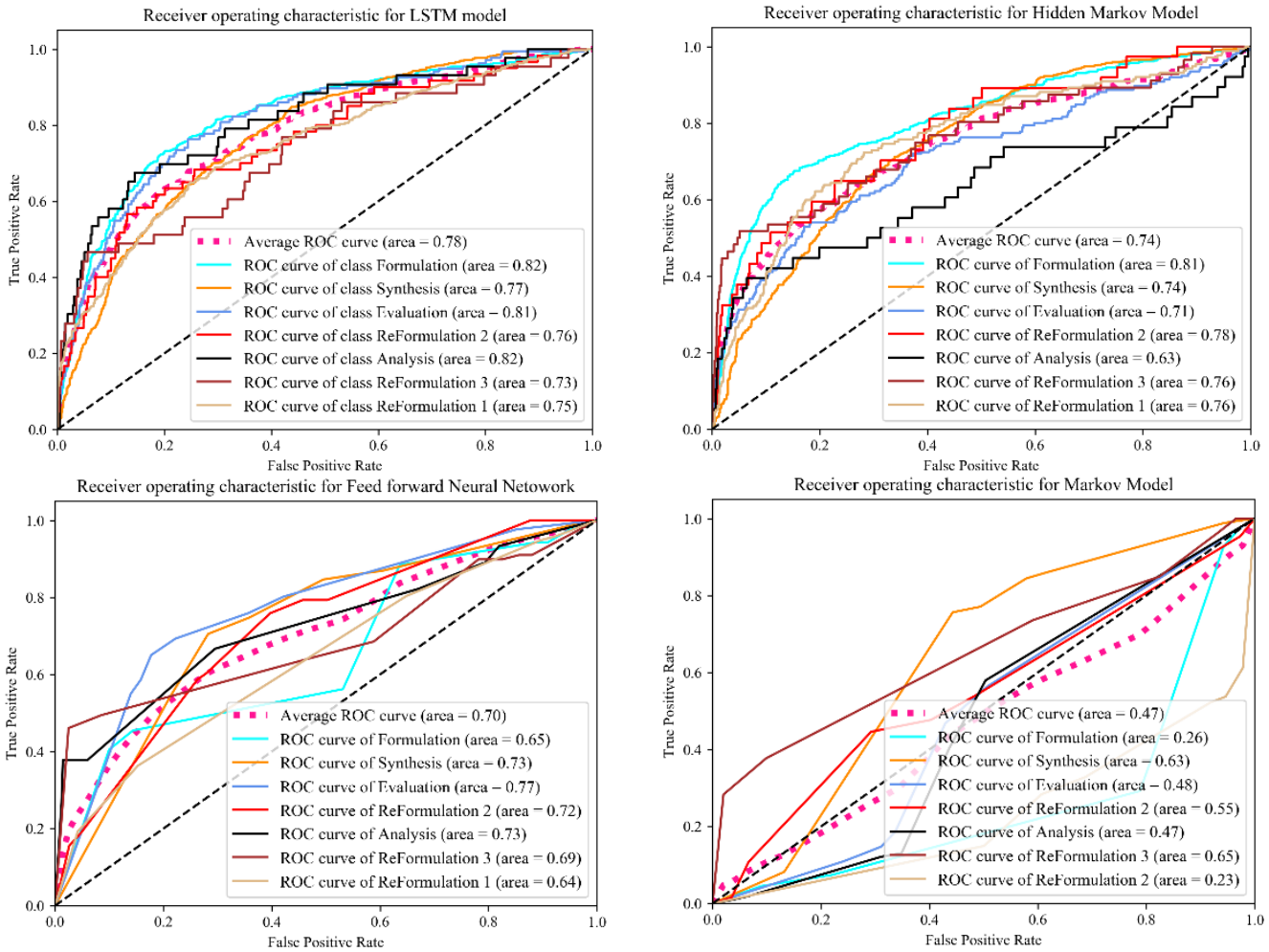

FIGURE 9: Receiver operating characteristics (ROC) curves for LSTM, HMM, FFNN and MM, respectively. Each of the figure contains the average area under the curve (AUCROC) score of the model as well as the AUCROC score of each design process stages. The closer the score is to 1, the higher predictive performance does a model have.

ROC curves for each design process stage, the area under the ROC curve (AUCROC) is used to provide one single metric which aggregates the predictive performance cross the thresholds so that we can compare on which design process stage does the model perform better. Larger AUCROC indicates better predictive performance.

For example, for the LSTM model as shown in Figure 9(a), the AUCROC of *Formulation* and *Analysis* both reaches the maximum of 0.82 among the seven design process categories. LSTM model also produces good AUCROC score (0.81) for *Evaluation*. These results imply that designers tend to enter into these design stages after completing a certain series of design tasks. For example, the designers must first construct the house which involves many design actions related to *Formulation* and then evaluate the performance by simulating annual net energy (*Analysis*) and analyzing the building cost (*Evaluation*).

However, LSTM has relatively lower AUCROC values for *Reformulation 1, 2 and 3*. This is because *Reformulation* involves the design actions of removing components, such as remove a solar panel or remove a window. These removal actions are often paired with another *Reformulation* and/or *Formulation* actions, such as add a wall or add a window. These action pairs reflect designers' fine-tuning behaviors on particular design components immediately based on the observations from the CAD interface and no necessary to run a simulation for feedback to support their design decisions. Therefore, referring to the action in the last step should be sufficient for prediction

Table 2: Area under the receiver operating characteristics curve (AUCROC) score for different models.

	LSTM	HMM	FFNN	MM
Formulation	0.82	0.81	0.65	0.26
Synthesis	0.77	0.74	0.73	0.63
Evaluation	0.81	0.71	0.77	0.48
Analysis	0.82	0.63	0.73	0.47
Reformulation 1	0.75	0.76	0.64	0.23
Reformulation 2	0.76	0.78	0.72	0.55
Reformulation 3	0.73	0.76	0.69	0.65
Average	0.79	0.74	0.70	0.47

Table 3: Different hyperparameter settings for LSTM model.

No.	LSTM size	Dense Number	Dense Size	Dropout	Learning rate	Training accuracy	Testing accuracy
1	256	1	7	0.3	0.1	60.61%	59.50%
2	128	1	7	0.3	0.001	60.12%	58.48%
3	256	1	7	0.2	0.01	60.41%	58.97%
4	128	1	7	0.3	0.01	59.60%	59.16%
5	128	1	7	0.3	0.1	62.63%	59.52%
6	256	2	128 and 7	0.2	0.1	56.20%	54.95%

and it does require to use long-term memory in predicting these design stages. This may also be the reason why MM can produce higher AUCROC scores for *Reformulation 2 and 3* (0.55 and 0.63, respectively). For example, *Reformulation 2* contains the design actions related to the removal of solar panels. As solar panels directly affect the system performance, most designers spent a significant amount of time fine-tuning (e.g., add, remove and then add back again) this component, and therefore, there exist a large number of action pairs of *Add Solar Panel* → *Remove Solar Panel* in the designs sequence. Since MM predicts the state only one-time step ahead based on the current state, it captures this design pattern very well. But, on the other hand, it does not effectively capture the patterns that involve much and long historical information, such as the *Evaluation and Analysis* stages, as compared to the other models. Please note that MM has the least AUCROC for *Formulation*. This is because MM is derived from frequency of the event. In the design, the repetition of *Formulation* (corresponds to adding components) does not occur frequently. For example, once a designer finishes adding all the necessary components, e.g., "Add Wall" and "Add Window", she/he would never take those actions again because the house has been already established. Instead, she/he tends to start fine-tuning the associated parameters through the actions of "Edit Wall" and "Edit Window" (i.e., the auctions related to *Synthesis*).

If we take average for the AUCROC scores from every design process stages, that average value can be used to compare the performance between different models, as shown in the last row of Table 2. We observe that on average the LSTM model outperforms the other models with the AUCROC of 0.79. The HMM model (0.74) also performs better than FFNN (0.70) and MM (0.47). Both FFNN and HMM perform relatively in average across all the design process stages. Their scores are also in between the ones from LSTM and MM, respectively, both for each design process stage as well as for the final average value. This indicates that even if HMM and FFNN take historical design information into their prediction, they are not effectively processing those information during the model training. But LSTM's *gate mechanism* (e.g., input gate, forget gate and output gates) seem well to capture and process the dependent relations between different design stages during a design process, therefore, it yields the best performance. Figure 9 visually shows the ROC curves of the models including MM, HMM, FFNN and LSTM. These results cross-validate the conclusion we reached from the prediction accuracy results shown in Figure 8.

5.3 Sensitivity analysis

When training an LSTM model, there are several pre-determined hyperparameters, such as the number of LSTM layer, LSTM size, the number of dense layer, the size of dense layer, learning rate and dropout value. LSTM size refers to LSTM nodes in each LSTM layer. Dense number indicates the number of layers of the feedforward part in Figure 4. The size of dense layer indicates the number of nodes in each dense layer. Dropout is the value of dropout regularization. Learning rate is the converge rate used in the stochastic gradient descent algorithm used in backpropagation. In order to prevent the model overfitting, we use dropout regularization [47] with two different values.

To investigate how the prediction accuracy would be affected by these hyperparameters, we perform a sensitivity analysis by changing the values of these parameters and values the corresponding prediction accuracies. In the experiment, we use one layer of LSTM for all the settings but try various number of LSTM nodes. Table 3 shows the training accuracy and test accuracy of the LSTM models with different hyperparameter settings. From all the settings, it is observed that the model with one dense layer performs better (i.e. above 58%) that the models with two dense layers (i.e., 56.20% for training and 54.95% for testing, respectively). Given the same number of dense layers and the same dense size, it is observed that a learning rate of 0.001 produces relatively lower performance (58.48%) than those of other settings. But the dropout rate (changing from 0.3 to 0.2) and the LSTM size (changing from 256 to 128 nodes) do not influence the model significantly. Among all the settings, it is found that the model with LSTM unit 128, dropout value with 0.3 and learning rate with 0.1 provides the best accuracy.

6. CONCLUSION

In this study, a deep learning approach is developed to analyze and predict the sequential design decisions in a system design context. We use Energy3D as the research platform to conduct design challenges and collect designers' sequential design behavioral data. Then, the FBS-based design process model is adopted to transform the sequential design action data into the sequential design process data. Based on the design process data, we established two deep learning models, i.e., the FFNN and the LSTM, to predict designers' next immediate design process stage. These deep learning models are evaluated with different performance metrics including training accuracy, testing accuracy, and area under ROC curve. Their predictive

performances are compared with other four models, including a MM, an HMM, a repetitive model, and a random model. The predictive power is assessed at the level of entire design sequence as well as at the level of each design process stage.

We found that on average LSTM outperforms all the other models while FFNN shows lower performance than traditionally used HMM. From the ROC curve analysis, we found that LSTM yields better performance on the design actions that belong to *Formulation*, *Evaluation*, and *Analysis*, while HMM better predicts the actions related three *Reformulation* design processes. With these findings, we revisit the research question that we aim to answer: *What is the effect of past decisions on predicting designers' future sequential decision making?* Since LSTM incorporates historical information of both short-term and long-term and captures the interconnections between the current design stage and past, its higher predictive perform indicates that designers effectively leverage both short-term and long-term memories in guiding their sequential decision making in engineering systems design. But increasing the LSTM size does not improve the prediction significantly, so it is inferred that designers would not maintain a long working memory in systems design activities. Validating the effect of working memory and investigating how exactly short-term memory play a role in sequential decision making is our ongoing research.

This work shows that deep learning can be a stepping stone for modeling and predicting sequential decision making in engineering design and facilitating design automation. The approach introduced in this paper is general and can be implemented in many other design areas, especially complex configuration design problems, to extract the design decision-making strategies and design heuristics. However, there are some limitations in our approach. For example, in this study, we only consider time-dependent data to explore designer's sequential decision. Time-independent data such as demographics or experiential knowledge pertaining to designers may also affect their design decisions and strategies. Additionally, to train a high-fidelity deep learning model, a large number of subjects is required which may not be available readily in college setting.

In future work, on the one hand, we will continue doing the experiment to collect more data. On the other hand, we plan to develop a more robust deep learning architecture, which combines both dynamic information (time-series data) and static information (designer-specific data), for better understanding the roles that different factors play in sequential design decision. We envision that by adding the time-independent feature, the model performance of LSTM can be further improved.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support from the U.S. National Science Foundation (NSF) CMMI through grant CMMI-1842588, DUE-1348530 and DRL-1503196. We also greatly acknowledge the advices provided by Dr. Xintao Wu and Dr. Shuhan Yuan on the topic of recurrent neural network modeling.

REFERENCES

1. Rahman, M.H., et al. *Automatic Clustering of Sequential Design Behaviors*. in *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2018.
2. Panchal, J.H., Z. Sha, and K.N. Kannan, *Understanding design decisions under competition using games with information acquisition and a behavioral experiment.* Journal of Mechanical Design, 2017. **139**(9): p. 091402.
3. Sexton, T. and M.Y. Ren, *Learning an Optimization Algorithm Through Human Design Iterations.* Journal of Mechanical Design, 2017. **139**(10): p. 101404.
4. Brockmann, E.N. and W.P. Anthony, *The influence of tacit knowledge and collective mind on strategic planning.* Journal of Managerial Issues, 1998: p. 204-222.
5. Collobert, R. and J. Weston. *A unified architecture for natural language processing: Deep neural networks with multitask learning*. in *Proceedings of the 25th international conference on Machine learning*. 2008.
6. Miotto, R., et al., *Deep learning for healthcare: review, opportunities and challenges.* Briefings in bioinformatics, 2017. **19**(6): p. 1236-1246.
7. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
8. Heaton, J.B., N.G. Polson, and J.H. Witte, *Deep learning in finance.* arXiv preprint arXiv:1602.06561, 2016.
9. Kan, J.W.T. and J.S. Gero. *Using the FBS ontology to capture semantic design information in design protocol studies.* in *About: Designing. Analysing Design Meetings*. 2009.
10. Yu, R.O.N.G.R.O.N.G., et al. *An empirical foundation for design patterns in parametric design*. in *20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Daegu, South Korea, May*. 2015.
11. McComb, C., J. Cagan, and K. Kotovsky, *Capturing human sequence-learning abilities in configuration design tasks through markov chains.* Journal of Mechanical Design, 2017. **139**(9): p. 091101.
12. McComb, C., J. Cagan, and K. Kotovsky, *Mining process heuristics from designer action data via hidden markov models.* Journal of Mechanical Design, 2017. **139**(11): p. 111412.
13. Sha, Z., K.N. Kannan, and J.H. Panchal, *Behavioral Experimentation and Game Theory in Engineering Systems Design.* Journal of Mechanical Design, 2015. **137**(5): p. 051405-051405-10.
14. Smith, R.P. and S.D. Eppinger, *A predictive model of sequential iteration in engineering design.* Management Science, 1997. **43**(8): p. 1104-1120.
15. Griffin, S., N.J. Welton, and K. Claxton, *Exploring the research decision space: the expected value of*

information for sequential research designs. Medical Decision Making, 2010. **30**(2): p. 155-162.
16. Meier, C., A.A. Yassine, and T.R. Browning, *Design process sequencing with competent genetic algorithms.* Journal of Mechanical Design, 2007. **129**(6): p. 566-585.
17. Duff, M.O.G. and A. Barto, *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. 2002, University of Massachusetts at Amherst.
18. Santolaya, D.S.a., nchez, *Using recurrent neural networks to predict customer behavior from interaction data*. 2017.
19. Mohamed, A.-r., G.E. Dahl, and G. Hinton, *Acoustic modeling using deep belief networks.* IEEE Transactions on Audio, Speech, and Language Processing, 2012. **20**(1): p. 14-22.
20. Robinson, T., *An application of recurrent nets to phone probability estimation.* IEEE transactions on Neural Networks, 1994. **5**(2).
21. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
22. Graves, A., et al. *Unconstrained on-line handwriting recognition with recurrent neural networks*. in *Advances in neural information processing systems*. 2008.
23. Graves, A., A.-r. Mohamed, and G. Hinton. *Speech recognition with deep recurrent neural networks*. in *2013 IEEE international conference on acoustics, speech and signal processing*. 2013.
24. Chen, Y., et al., *Gene expression inference with deep learning.* Bioinformatics, 2016. **32**(12): p. 1832-1839.
25. Lee, B., et al., *DNA-level splice junction prediction using deep recurrent neural networks.* arXiv preprint arXiv:1512.05135, 2015.
26. Soleymani, M., et al. *Continuous emotion detection using EEG signals and facial expressions*. in *2014 IEEE International Conference on Multimedia and Expo (ICME)*. 2014.
27. Baccouche, M., et al. *Sequential deep learning for human action recognition*. in *International workshop on human behavior understanding*. 2011.
28. Zhang, R., F. Zheng, and W. Min, *Sequential Behavioral Data Processing Using Deep Learning and the Markov Transition Field in Online Fraud Detection.* arXiv preprint arXiv:1808.05329, 2018.
29. Yuan, S., X. Wu, and Y. Xiang. *A Two Phase Deep Learning Model for Identifying Discrimination from Tweets*. in *EDBT*. 2016.
30. Korczak, J. and M. Hemes. *Deep learning for financial time series forecasting in A-Trader system*. in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. 2017.
31. Rosenblatt, F., *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 1958. **65**(6): p. 386.
32. Karlik, B. and A.V. Olgac, *Performance analysis of various activation functions in generalized MLP architectures of neural networks*. International Journal of Artificial Intelligence and Expert Systems, 2011. **1**(4): p. 111-122.
33. Nicoletti, G.M., *An analysis of neural networks as simulators and emulators*. Cybernetics \& Systems, 2000. **31**(3): p. 253-282.
34. Hanin, B., *Universal function approximation by deep neural nets with bounded width and relu activations.* arXiv preprint arXiv:1708.02691, 2017.
35. Bengio, Y., P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult.* IEEE transactions on neural networks, 1994. **5**(2): p. 157-166.
36. Xie, C., et al., *Learning and teaching engineering design through modeling and simulation on a CAD platform.* Computer Applications in Engineering Education, 2018. **26**(4): p. 824-840.
37. Xie, C., et al., *A time series analysis method for assessing engineering design processes using a CAD tool.* International Journal of Engineering Education, 2014. **30**(1): p. 218-230.
38. Gero, J.S., *Design prototypes: a knowledge representation schema for design.* AI magazine, 1990. **11**(4): p. 26.
39. Verleysen, M. and D. François. *The curse of dimensionality in data mining and time series prediction*. in *International Work-Conference on Artificial Neural Networks*. 2005. Springer.
40. Potdar, K., T.S. Pardawala, and C.D. Pai, *A comparative study of categorical variable encoding techniques for neural network classifiers*. International Journal of Computer Applications, 2017. **175**(4): p. 7-9.
41. Kohavi, R. and others. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. in *Ijcai*. 1995.
42. Chollet, F., *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. 2018: MITP-Verlags GmbH \& Co. KG.
43. Friedman, J., T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Vol. 1. 2001: Springer series in statistics New York.
44. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.
45. Lee, K.C., S. Phon-Amnuaisuk, and C.Y. Ting. *A comparison of HMM, Na}ve Bayesian, and Markov model in exploiting knowledge content in digital ink: A case study on handwritten music notation recognition*. in *2010 IEEE International Conference on Multimedia and Expo*. 2010.
46. Fawcett, T., *An introduction to ROC analysis.* Pattern recognition letters, 2006. **27**(8): p. 861-874.
47. Srivastava, N., et al., *Dropout: a simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 2014. **15**(1): p. 1929-1958.