Learning Time-series Data of Industrial Design Optimization using Recurrent Neural Networks

Sneha Saha, Thiago Rios, Stefan Menzel, Bernhard Sendhoff, Thomas Bäck, Xin Yao, Zhao Xu, Patricia Wollstadt

2019

Preprint:

This is an accepted article published in 2019 ICDM Workshop: Learning and Mining with Industrial Data (LMID). The final authenticated version is available online at: https://doi.org/10.1109/ICDMW.2019.00116

Learning Time-series Data of Industrial Design Optimization using Recurrent Neural Networks

Sneha Saha*, Thiago Rios*, Stefan Menzel* Bernhard Sendhoff*, Thomas Bäck†, Xin Yao‡§, Zhao Xu¶ and Patricia Wollstadt*
*Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 63073 Offenbach, Germany
†Leiden Institute of Advanced Computer Science (LIACS), Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
‡CERCIA, School of Computer Science, University of Birmingham, UK
§Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
¶NEC Laboratories Europe, Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
Email: {sneha.saha, thiago.rios, stefan.menzel, bernhard.sendhoff, patricia.wollstadt}@honda-ri.de,
t.h.w.baeck@liacs.leidenuniv.nl, x.yao@cs.bham.ac.uk, zhao.xu@neclab.eu

***Abstract*—In automotive digital development, 3D shape morphing techniques are used to create new designs in order to match design targets, such as aerodynamic or stylistic requirements. Control-point based shape morphing alters existing geometries either through human user interactions or through computational optimization algorithms that optimize for product performance targets. Shape morphing is typically continuous and results in potentially large data sets of time-series recordings of control point movements. In the present paper, we utilize recurrent neural networks to model such time-series recordings in order to predict future design steps based on the history of currently performed design modifications. To build a data set sufficiently large for the training of neural networks, we use target shape matching optimization as digital analogy for a human user interactive shape modification and to build data sets of control point movements in an automated fashion. Experiments show the potential of recurrent neural networks to successfully learn time-series data representing design changes and to perform single- and multi-step prediction of potential next design steps. We thus demonstrate the feasibility of recurrent neural networks for learning successful design sequences in order to predict promising next design steps in future design tasks.**

***Index Terms*—Recurrent neural networks, computer aided engineering, deep learning, optimization**

I. Introduction

Central to digital design development in the automotive domain is the variation of 3D shapes in order to fulfill requirements of the various stakeholders involved. Requirements comprise aesthetic design-guidelines, but also functional performance targets such as aerodynamic efficiency or structural stiffness. Even though various tools from computer aided design (CAD) and engineering (CAE) provide means for altering shapes, the necessity to simultaneously meet various—potentially conflicting—targets, leads to considerable complexity in the design process. To support a human designer in handling this complexity, we propose to learn a model of existing, successful shape manipulation sequences, in order to predict potential future design steps from the modifications currently applied. Such a model offers the possibility to provide real-time feedback on the current design and to collaboratively guide the designer through the design process.

On a 2D design level, recently two systems have been proposed, which provide real-time guidance to human users while drawing online. ShadowDraw [1] relies on a database of images and provides feedback to user sketches by offering opaque visual layers of potentially similar objects. SketchRNN [2] is a generative, deep neural network trained on a database of drawing sequences by human users. When presented with a novel sketch sequence for a pre-selected object class, the network proposes alternatives of next drawing steps to illustrate potential design directions.

The present paper aims at transferring the idea of guiding the design process using a model trained on existing data into 3D design space for engineering applications. Here, a number of challenges emerge. First, we need to select a meaningful representation capable of modifying objects with a high degree of flexibility and low number of parameters. In the present paper we rely on shape morphing methods such as free form deformation (FFD) [3]–[5] that are state-of-the-art techniques in engineering applications and allow human users to perform intuitive design changes [6]. Second, due to the high complexity of engineering designs, a large data set of design modification sequences is required as training data for our prediction model. Since it would be very costly (and in reality impossible) to generate this amount of data manually, we propose to utilize computational optimization methods for generating these sequences. We propose to tune the hyperparameters of an evolutionary optimization algorithm such that the optimizer's output closely matches user-generated deformation sequences. Based on these hyperparameter settings we generate a large number of deformation sequences from repeated optimization runs. Third, for learning the design sequences, we utilize multi-input recurrent neural networks (RNNs), which are capable to abstract temporal information contained in deformation time-series and enable us to predict future potential shape morphing directions based on a current state of design modifications.

The remainder of the paper is organized as follows: We

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186.

2375-9259/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDMW.2019.00116

review prior work on learning models for engineering tasks and on artificial neural networks in section II; in particular, we review RNNs as network architectures specialized on processing sequential data. We then introduce, in section III, an approach for data generation from a simplified 3D design task, resulting in a sufficient amount of multivariate sequences of design changes. In section IV, we present results on training an RNN on the generated design sequences in order to predict future design steps. In section V, we discuss our results in the light of providing real-time guidance to designers performing 3D design tasks in the engineering domain.

II. Prior Art

A. Metamodeling

Learning from data that results from optimizing a design to match a given design target, is also referred to *metamodeling* [7]–[9]. Commonly, metamodeling means training a model on the steps an optimizer performs with the goal to predict optimal design parameters, x, such that an objective function, $f(x)$, is optimized. Metamodels replace complex or computationally expensive aspects of an optimization, such as the evaluation of a gradient or cost function, by a simpler and cheaper model. Hence, metamodelling models the design process, with the goal of approximating a globally optimal model as accurately as possible at a reasonable cost. Furthermore, metamodels allow to explore the design space to obtain deeper insight into the design problem and a better formulation of the optimization task. In the context of metamodeling, the feasibility of artificial neural networks (ANN) for learning from engineering data has been explored in [9]. ANNs have shown the ability to learn from high-dimensional design spaces and a limited number of samples [8], suggesting ANNs as a promising approach to model engineering design data.

However, purely feed-forward ANNs assume independence between individual data samples and require fixed-size input. Both assumptions are violated if sequential inputs or inputs with temporal dependencies are considered. To handle these limitations, recurrent neural networks (RNN) have been proposed (see [10] for a review), which model time-dependency in the input explicitly. RNNs and recent extensions using deep neural networks (DNNs), for instance, long short-term memory (LSTM) networks [11], have been used successfully to learn from sequential data, while requiring little to no input of prior knowledge [12]. Furthermore, RNNs have been used as generative models [13], [14] which allows to use them to generate novel sequences based on the latent representations learned from data.

Hence, state-of-the art DNN architectures offer promising capabilities for modeling sequential engineering design data. Yet, current engineering applications make only limited use of such architectures, even though, ANNs, and in particular DNNs, not only offer greater flexibility when dealing with high-dimensional problem spaces, as commonly encountered in engineering contexts, but also provide generative models, which offer novel opportunities for exploring the design space. The latter has been demonstrated successfully for the 2D domain [1], [2], where recent applications use RNNs to model human sketches and offer guidance to human users during a design task.

B. Recurrent Neural Networks

RNNs are neural network architectures that model time-dependency, in particular long-term dependencies, in their inputs [10]. Compared to other modeling approaches for temporal data, RNNs have a high expressive power due to a high number of hidden states and non-linear activation functions [15]. Furthermore, networks are differentiable such that gradients for training are cheaply computed. RNNs model time-dependency by introducing recurrent or feedback connections that, when calculating the network's current output based on the current input, allow to include the network's previous states into the calculation.

Early RNN architectures suffered from practical problems during training, in particular, the calculation of the gradient for backpropagation can become unstable, such that gradients either "vanish" or "explode". To overcome these shortcomings of earlier architectures, current architectures like LSTMs propose improvements to the original architecture [11]: LSTM cells overcome training problems, by introducing self-feedback connections of constant weight that enable a gradient to propagate through a cell for multiple time steps without decreasing and therefore to vanish. An additional *multiplicative unit* learns when to open and close in- and output gates of the cell in order to allow for constant error flow over multiple time steps. Newer implementations introduce forget gates [16], which can reset the internal state of the cell. Based on these additions to classical RNNs, LSTM networks have shown to be able to learn long-range dependencies more successfully and robustly. Accordingly, LSTM networks achieve state-of-the-art performance in a variety of application domains (e.g., speech recognition [17], or video encoding [18], see also [10] for a review).

Modern RNNs, such as LSTMs, have been shown to handle complex interactions between multivariate input- and output sequences [19]. Hence, not only predictions of single but also multiple time steps, are possible. Furthermore, these networks are *generative*, i.e., they are able to generate new sequences by providing probabilistic predictions of future samples. By sampling repeatedly from the RNNs output distribution complete, novel sequences can be generated [13]. Recently, these generative capabilities together with the expressive power of RNNs have been used to model sequences from 2D-design tasks, such as human drawing [1], [2], and, in particular, the generative capabilities of RNNs have been used to guide users in drawing new designs. Similar approaches to generating temporal sequences have been explored in other domains, e.g., for the generation of music [20].

As a consequence, RNNs are a promising tool for learning design sequences in the engineering domain, in particular, for learning sequences describing complex tasks such as 3D design processes. Furthermore, their generative abilities can be exploited to guide novel users based on the learned design

experience. However, even though RNNs are a promising tool for learning from multivariate, sequential data, applications to engineering design tasks are lacking so far. In the present paper, we therefore explore the use of RNNs for learning sequences describing an engineering 3D design task.

III. Methods

In the spirit of progressing towards a 3D cooperative design tool for industrial data, we first describe how a sufficient amount of data for training an RNN were generated: to this end, we defined a shape-manipulation task solved by FFD, which was repeatedly carried out using an optimization algorithm, and resulted in a set of sequences of FFD control point movements. Next, we introduce the RNN architecture used to learn these sequences in order to predict future control point movements.

A. Shape Manipulation Task

To obtain a simplified, yet relevant shape manipulation process in the engineering context, we propose a target-shape matching task where a human designer or an optimization algorithm have to modify an initial shape such that it matches a predefined target shape. We use target-shape matching as a proxy for a design process, where a human user operates a CAD system with the intention of modifying a shape such that a design target is met. Such a target may be either given implicitly, e.g., based on aesthetic considerations, or explicitly, e.g., based on functional or structural requirements. An optimal result of the matching task is a design with minimal distance to the target shape.

Target-shape matching is a popular approach when evaluating otherwise time-consuming optimization methods in the engineering domain. Instead of performing an optimization task that requires the evaluation of a costly objective function, the optimization task is replaced by target-shape matching, assuming that calculating the distance between the current shape and the target shape surface is computationally cheaper. For example, optimizing a part for aerodynamic performance requires Computational Fluid Dynamics (CFD) simulations, where every simulation may take between a couple of minutes to hours. Hence, optimizing for aerodynamic performance requires iterative shape modification and CFD simulation until the optimization converges. This approach becomes infeasible if large numbers of optimization runs are to be carried out, where each run may require several hundreds of optimization and simulation steps. Thus, to test optimization approaches, target-shape matching replaces such costly objectives by first defining a target shape as the global optimum to be reached by the optimization, and second, using the optimization method to modify shape parameters to fit an initial, e.g. random, design to the target.

B. Free Form Deformation

As a tool to modify an initial shape such that the distance to a target shape is minimized, we use FFD [4]. FFD allows easy manipulation of a shape by providing the user with a set of control points around the shape, that can be moved in 3D space in order to deform the geometry (Figure 1). Also, by recording the sequence of control point movements, FFD offers a straightforward parameterization of the manipulation process in terms of direction and magnitude of control point movements. FFD deforms shapes by first embedding the geometry in a parallelepiped lattice such that the points defining the geometry can be expressed with respect to the local coordinate system spanned by the lattice. Deformations are then introduced by transforming the lattice and recomputing the points describing the geometry with respect to the transformed lattice.

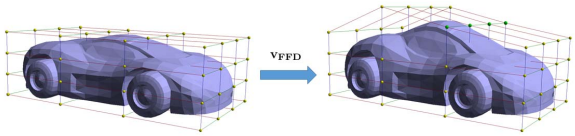

Fig. 1: Schematic representation of the car shape surface mesh [21] embedded in the FFD lattice. Spheres indicate control points of the lattice. Layers of control points can be grouped together to control planes to minimize the number of optimization parameters.

We represent a geometry as a polygon surface mesh, $K = (G, P)$, where the mesh connectivity is described by a graph $G = (V, E)$ with mesh vertices V and $|V| = N$, edges $E \subseteq V \times V$, and 3D coordinates P. The geometry is embedded into the control lattice's local coordinate system,

$$\vec{v} = \vec{v}_0 + s\vec{s} + t\vec{t} + u\vec{u}, \tag{1}$$

where $\vec{s}$, $\vec{t}$, and $\vec{u}$ are unitary vectors defining the coordinate system, $\vec{v}_0$ describes the system's origin, and s, t, and u are the resulting local coordinates.

Within the uniformly spaced lattice, we define control points, $\mathbf{p}_i$, which, when displaced, deform the geometry according to [4]:

$$\mathbf{v_{FFD}} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i \left\{ \sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j \left[\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^k \mathbf{p}_{ijk} \right] \right\} \tag{2}$$

where $\mathbf{v_{FFD}}$ is the deformed surface mesh vertex and l, m, n are the number of control planes in $\vec{s}$-, $\vec{t}$-, $\vec{u}$- direction, respectively. In our case, each plane is defined by a set of control points that are moved simultaneously. In the remainder of this article, we will use the movement of a whole control plane for the deformation of shapes. Formulation (2) ensures continuity within the lattice, such that control volumes with more than three planes impose continuity of curvature on deformed shapes, i.e., a smooth deformed surface, which is usually desired when modeling engineering components.

For data generation, we considered a benchmark car shape [21], represented by a mesh refined to $N = 8474$ nodes.

For FFD, we used a lattice with six planes each in x- and z-directions, and four planes in y-direction. Assuming symmetry of planes with respect to the geometric center of the vehicle, plane positions along the normal directions yield an eight-dimensional design space for deformations. We record sequences of shape deformations as the current distance of each control plane x_i ($i \in \{1, \ldots, 8\}$) to the geometric center of the shape.

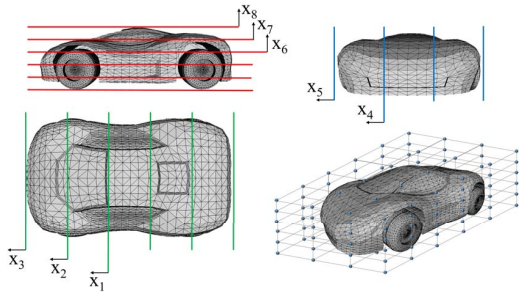

Fig. 2: Planes of the control lattice used for free form deformation of the car shape.

C. Optimization Algorithm used for Data Generation

In order to avoid the manual generation of a large data set of shape deformation sequences for training an RNN, we propose to generate a synthetic data set by solving the target-shape matching task using computational optimization. In such an optimization task, the objective is to minimize the geometric difference between the initial and the target shape, which can be measured by a modified version of the Hausdorff distance [22],

$$\mathcal{H}(V, V_0) = \frac{1}{2}\left(\sum_{i=1}^{N} \min_{a_i \in V_0} |a_i - \tilde{a}|^2 + \sum_{j=1}^{N} \min_{\tilde{a}_j \in V} |\tilde{a}_j - a|^2 \right) \quad (3)$$

where a_i denote mesh vertices of the reference shape, V_0, and $\tilde{a}_j$ denote vertices of the deformed geometry, V.

During the optimization, the initial shape is modified using FFD of the eight control planes defined above, where control planes are moved iteratively in each optimization step according to the optimization algorithm for search and minimization. The optimization algorithm used was the covariance matrix adaptation evolution strategy (CMA-ES) [23], following the implementation proposed in [24]. CMA-ES is a popular choice for optimization in the engineering domain, due to a high convergence ratio, also for small population sizes, an adaptive step-size, and a low number of hyperparameters [23], [25]–[27]. We set the hyperparameters such that the convergence behavior of the algorithm approximated shape modifications sequences as performed by a human designer, leading to a $(3, 10)$-evolutionary strategy, i.e., populations of ten offspring individuals and three parents, with initial size step of 0.01. Shape modifications performed by human users were investigated in a user study using a simplified shape-matching task (data not shown). Hyperparameters were set to generate design sequences close to operations performed by users experienced in the operation of CAD systems.

The optimization was performed over 70 generations for 150 different target shapes. Each target shape was generated as a random deformation of the initial car shape, obtained using FFD to scale the shape randomly in each direction. Deformations were sampled from a uniform distribution under the constraint that the scaling in each direction should be within $\pm$ $40\,\%$ of the shape's initial size. The feasibility of the optimized shape was evaluated through visual inspection. Each optimization run resulted in an eight-dimensional design sequence of individual control plane movements, x_i. Each sequence denotes the distance of the control point to the geometric center of the car shape in each generation of the optimization.

D. Recurrent Neural Network Model

For learning from generated data, we used an LSTM network [11] and trained it on the eight-dimensional sequences describing control plane position.

The network architecture consisted of a one-layer LSTM of 256 units, followed by one fully connected output layer containing eight cells, corresponding to the number of desired outputs. The learning rate was set to 0.01, batch size and number of epochs were set to 32 and 128, respectively. We used the Adam optimizer [28] and the mean square error as loss function for training. We implemented the LSTM network using the Keras framework [29] with a Tensorflow back-end [30].

To use the LSTM network for prediction, we converted the design sequences into individual *samples* using a sliding window of length w and step-size 1, leading to a three dimensional tensor with dimensions *samples*, *time steps* and *features*. Here, each sample corresponds to one window of w time steps and features denote the positions of the eight control planes, x_i. By associating each window with its next actual time step, we formulate a supervised learning task of predicting the next time step from the window of the immediate w past time steps (see Figure 3 for an illustration). Based on the training data, the LSTM layer abstracts a latent representation of the input time series, based on which the fully-connected output layer produces the predicted, eight-dimensional next step in the sequence.

The model was trained on optimization sequences from 150 geometries and tested on a hold-out set of 5 and 15 sequences of the total 150 geometries. We used single-step prediction to calculate the loss during training and testing of the model. For comparison, we used a moving average baseline model that performed single-step prediction as the average over the past w input samples. The window length was determined by choosing the maximum lag, d, at which the partial autocorrelation function (PACF) of each design sequence feature decayed below the threshold $1.96/\sqrt{T-d}$, where T is the length of the time series and 1.96 defines the critical alpha level at 5 under the null hypothesis that the PACF

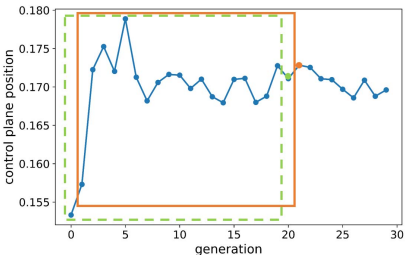

Fig. 3: Windowing approach used to formulate one-step prediction as a supervised learning problem. Boxes indicate windows used as training samples and colored markers indicate corresponding next time-steps used as labels.

is zero (assuming the PACF asymptotically follows a standard normal distribution) [31].

In addition to the single step prediction model, we also trained a model for multi-step prediction. Multi-step forecasting may be performed using a model that immediately produces a multi-step output or can be performed using a recursive strategy. The recursive strategy involves using a one step model iteratively on the next predicted step, while the predicted sample is used as a new input for making the following prediction. Due to feeding predictions back into the model as input, the recursive strategy may lead to an accumulation of predictions errors such that the performance quickly degrades as the prediction time horizon increases. Hence, we here used a multi-step prediction model. The architecture was identical to the single-step prediction model, however, the number of LSTM units was increased to 512, and the number of time steps predicted was set to 3 for each of the eight features, such that the fully connected output layer contained 24 cells.

IV. Results

A. Data Preprocessing

Prior to training the LSTM network on the design sequences, we truncated optimization runs that showed an early convergence towards the target shape (Figure 4). For these runs, later generations showed minimal to no control plane movement (see Figure 4A for an example). To exclude these non-informative epochs in the data, we determined a cutoff criterion for each geometry sequences based on the difference in the rolling mean, using a window size of 10. If the difference in the mean fitness fell below $5\,\%$ of the optimal fitness, the sequence was truncated (Figure 4B).

We determined the range of the window length, w, for defining labeled training samples using the PACF of each sequence [31]. Through cross-validation, we found the best prediction accuracy in the training data for $w = 22$, which was used for the single- as well as the multi-step prediction model.

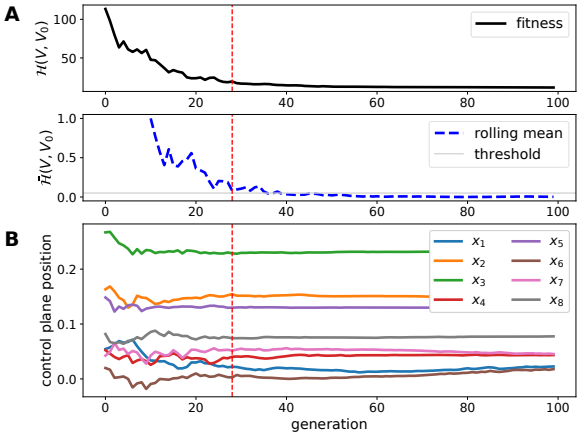

Fig. 4: Preprocessing of design sequences by truncating optimization runs showing early convergence. **A** Fitness (upper panel) and rolling mean of fitness (window size 10) over generations for a representative optimization run, where the dashed red line indicates cutoff based on the first mean value with 5% of optimal fitness. **B** Corresponding design sequence.

B. Single-step prediction

The LSTM network model trained to perform single-step prediction outperformed the baseline model for all eight features (Figure 5, Table II), demonstrating that the LSTM network successfully learned multivariate design sequences from data. We measured forecasting accuracy using mean absolute error and mean squared error between baseline- and model-prediction. In particular, the network was able to predict the multivariate output, i.e., the joint movement of all eight control planes, hence, successfully accounting for the multivariate nature of the data.

To evaluate the capability of the model to suggest next design steps to a user, we performed predicted modifications on a deformed car mesh from the beginning of an optimization run (see Figure 6 for a representative sample).

C. Multi-step prediction

To evaluate whether the LSTM network was able to predict multiple next design steps, we trained a second model to perform multi-step prediction for three time steps. Again, the LSTM model outperformed the baseline model (Figure 7, Table I). However, the model failed to consistently predict design changes that mirrored the behavior of the optimizer for all eight features (e.g., x_5, in Figure 7).

V. Discussion

We demonstrated the use of state-of-the-art RNN models for learning 3D design sequences in order to predict future design steps. We proposed a feasible parametrization of the design process in terms of FFD for manipulating 3D surface meshes, which was feasible for producing design sequences learnable

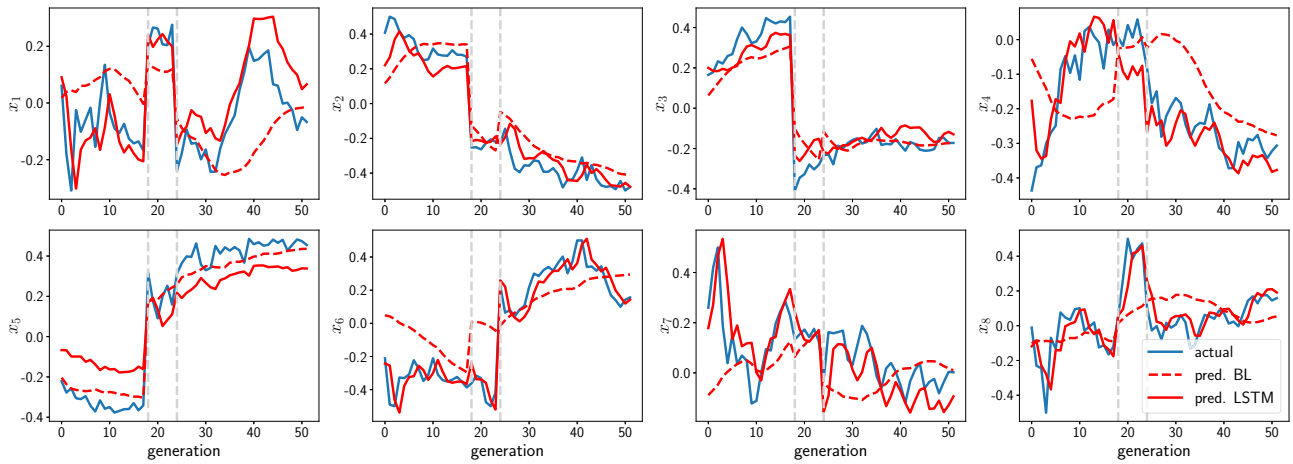

Fig. 5: Predicted (red lines) versus actual sequences (blue lines) for three test geometries. Comparison of single-step prediction using the LSTM network (solid red line) and single-step prediction using the baseline (BL) model (dashed red lines).

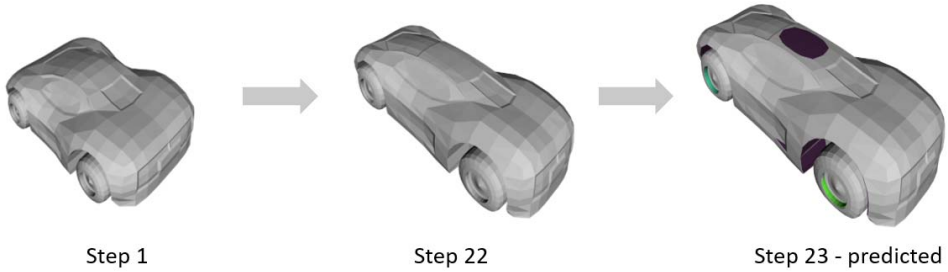

Fig. 6: Design changes predicted by the LSTM model for one sample of design changes: deformed car shape at time-step 1 (left) and time-step 22 (middle) of a representative window. Predicted design changes superimposed on the model from time-step 22 (right). The model proposes to move the roof downward (purple area) and to move the interior of the wheel cover backward (green area).

TABLE I: Mean absolute error (MAE) and mean squared error (MSE) plus standard deviations for baseline model (BL) and singlestep LSTM (LSTM) for 5 and 15 test set sequences

		BL		sLSTM	
		Training set	**Test set**	**Training set**	**Test set**
5/145 split	**MAE**	0.151 ± 0.002	0.154 ± 0.001	$\mathbf{0.117 \pm 0.003}$	$\mathbf{0.109 \pm 0.004}$
	MSE	0.038 ± 0.005	0.040 ± 0.002	$\mathbf{0.020 \pm 0.005}$	$\mathbf{0.019 \pm 0.007}$
15/135 split	**MAE**	0.148 ± 0.003	0.153 ± 0.002	$\mathbf{0.100 \pm 0.003}$	$\mathbf{0.120 \pm 0.001}$
	MSE	0.037 ± 0.005	0.040 ± 0.002	$\mathbf{0.016 \pm 0.004}$	$\mathbf{0.020 \pm 0.003}$

by an RNN. Furthermore, we proposed the use of a realistic design task from the automotive domain, which could be solved using computational optimization, such that a sufficient amount of data for training a DNN could be generated. We set optimization parameters such that the optimizer produced design sequences that mimicked the design process as carried out by an experienced human user.

We evaluate the potential of RNNs for learning a model of the generated, successful design sequences and explored how the trained model may be used to guide a (novice) designer in future design processes by proposing potential next design steps based on the history of currently performed design modifications. Such a model is an important step towards harnessing generative neural networks for modeling design experience in order to provide this experience to human designers, forming a human-machine system that is able to cooperatively support an engineer in a design task.

A. Modeling 3D design Processes

Deep neural networks targeted at sequence learning are a promising approach for modeling design processes in the engi-

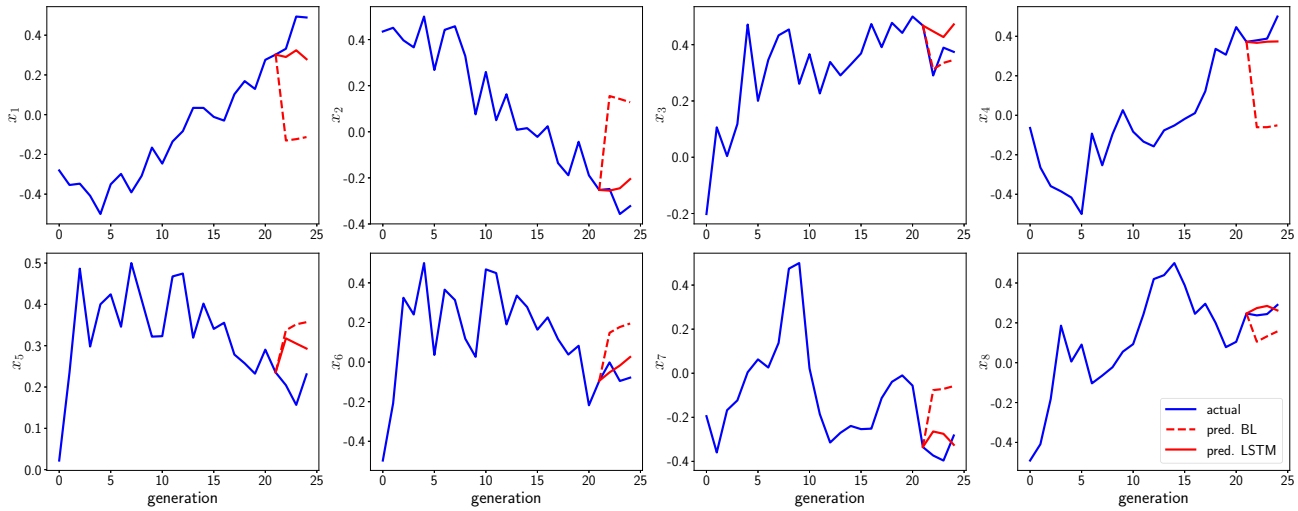

Fig. 7: Multi-step prediction (red lines) versus actual sequences (blue lines) and baseline (BL) model prediction (dashed red lines) for a representative geometry. Each panel shows one window of the design sequence and the corresponding three-step prediction.

TABLE II: Mean absolute error (MAE) and mean squared error (MSE) plus standard deviations for baseline model (BL) and multistep LSTM (mLSTM) for 5 and 15 test set sequences

		BL		mLSTM	
		Training set	**Test set**	**Training set**	**Test set**
5/145 split	**MAE**	0.804 ± 0.002	0.831 ± 0.001	$\mathbf{0.679 \pm 0.003}$	$\mathbf{0.700 \pm 0.004}$
	MSE	0.276 ± 0.005	0.286 ± 0.002	$\mathbf{0.095 \pm 0.005}$	$\mathbf{0.104 \pm 0.007}$
15/135 split	**MAE**	0.770 ± 0.003	0.800 ± 0.002	$\mathbf{0.657 \pm 0.002}$	$\mathbf{0.702 \pm 0.006}$
	MSE	0.269 ± 0.004	0.284 ± 0.003	$\mathbf{0.092 \pm 0.004}$	$\mathbf{0.108 \pm 0.002}$

neering domain, in particular, for handling multi-dimensional in- and outputs as they arise from 3D design processes. We showed that a specific type of RNN, an LSTM network, was able to learn and predict the next time step of the design process given the immediate history of modifications performed.

The prediction of multiple design steps showed promising performance compared to a baseline model. However, the predictions were not consistently in line with the design steps used by the optimizer for all control planes of the FFD. Here, future research should aim at improving the prediction of multivariate design changes for longer prediction horizons.

Furthermore, future work may use the trained RNN to explore the design space by exploiting the generative capabilities of RNNs and producing novel sequences of design changes.

In summary, RNNs showed the ability to learn from multivariate sequences representing shape modifications in a 3D design task. The networks' ability to generate novel design steps and to provide single- as well as multi-step prediction, make them promising candidates for learning from existing design data such as to model the design knowledge and experience contained within the data.

B. Towards Building Interactive Design Systems

The presented results are a promising step towards using DNNs for providing collaborative support to an engineer during a 3D design task. In particular, DNNs may be used as part of an interactive system that supports the design process by providing real-time feedback through predicting potential next modification steps.

Building interactive systems to support engineers in a design task is a long-standing topic in engineering research [32]. In particular, providing *guidance* during the design process has been identified as a central aspect in building such systems. State-of-the-art machine learning methods like DNNs, and in particular generative networks, provide an unprecedented opportunity to provide such guidance by modeling design experience learned from large amounts of realistic data and proposing next design steps. In particular, DNNs have been shown to provide *collaborative* assistance [2] and thus—instead of replacing human activity—amplify human ability such as to productively "couple agency and automation" [33]. Successful examples of such collaboration can be found in artificial intelligence methods integrated in human-machine interfaces [33], or knowledge bases that support ideation processes (see [34] and references therein). The present work

illustrates how DNNs for sequential data may be used to work towards collaborative assistance also for complex design tasks in the engineering domain.

C. Future Work

We presented the first evaluation of RNNs for learning from 3D design sequence data. Future work should expand this line of research by a more extensive investigation of neural network architectures for learning from such data. For example, windowed time series data may also be used to train convolutional neural networks, allowing for a further investigation of the benefit of recurrent neural network architectures over, e.g., convolutional ones. Furthermore, we may compare RNN models used here to classical time series models, such as autoregressive (AR) models, e.g. ARIMAX for multivariate time series [35]. Note, however, that the goal is to ultimately exploit generative capabilities of trained models, where (deep) neural networks have shown the most promising results [2]. Future work should therefore ultimately extend the presented models to generative architectures to allow for the creation of new design sequences from trained models. Future research should also explore the generalization capabilities of the trained model to previously unseen objects, also to provide support in the early modification steps. For practical application, retraining a model on large data sets may not be feasible. Hence, it is necessary to investigate how well a model trained on design data from one objects performs in the prediction of design sequences for a similar task on a novel object.

REFERENCES

[1] Y. J. Lee, C. L. Zitnick, and M. F. Cohen, "ShadowDraw: real-time user guidance for freehand drawing,"in *ACM Transactions on Graphics (TOG)*, vol.30,no. 4, Article 27, 2011.

[2] D. Ha and D. Eck, "A neural representation of sketch drawings," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[3] J. A. Samareh, "Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization," *AIAA Journal*, vol. 39, no. 5, pp. 877–884, 2001.

[4] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *ACM SIGGRAPH Computer Graphics*, 1986.

[5] S. Menzel and B. Sendhoff, "Representing the change–free form deformation for evolutionary design optimization," in *Evolutionary Computation in Practice*. Springer, vol. 88, pp. 63–86, 2008.

[6] S. Menzel, M. Olhofer, and B. Sendhoff, "A metamodel-driven interactive framework for a designer assistance system," in *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia*, pp. 1371–1380, 2010.

[7] T. Simpson, P. Koch, and J. Allen, "On the use of statistics in design and the implications for deterministic computer experiments," *Design Theory and Methodology-DTM'97*, pp. 14–17, 1997.

[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.

[9] G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design*, vol. 129, no. 4, pp. 370–380, 2007.

[10] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at Uber," in *International Conference on Machine Learning*, vol. 34, pp. 1–5, 2017.

[13] A. Graves, "Generating sequences with recurrent neural networks," *ArXiv preprint arXiv:1308.0850*, 2013.

[14] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

[15] H. T. Siegelmann, T. Hava, and E. Sontag, "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, no. 6, pp. 77–80, 1991.

[16] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[17] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[18] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International Conference on Machine Learning*, pp. 843–852, 2015.

[19] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," in *American Control Conference*, 2017.

[20] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio,"in The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, pp. 124, 2016.

[21] P. Prosic and P. Tiefenbacher, RepRap Action Car, 2008, [Online]. Available: https://www.thingiverse.com/thing:249/files.

[22] P. Zhang, X. Yao, L. Jia, B. Sendhoff, and T. Schnier, "Target shape design optimization by evolving splines," in *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 2009 – 2016, 2007.

[23] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[24] N. Hansen, Y. Akimoto, D. Brockhoff, and M. Chan, "CMA-ES/pycma: r2.7.0," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2651072.

[25] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Springer, pp. 75–102, 2006.

[26] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[27] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, vol. 2, no. 9, pp. 2–9, 1991.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[29] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[31] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. John Wiley & Sons, 2008.

[32] D. Ullman, "Toward the ideal mechanical engineering design support system," *Research in Engineering Design*, vol. 13, no. 2, pp. 55–64, 2002.

[33] J. Heer, "Agency plus automation: Designing artificial intelligence into interactive systems," in *Proceedings of the National Academy of Sciences*, vol. 116, no. 6, pp. 1844–1850, 2019.

[34] K. Fu, M. Fuge, and D. Brown, "Design creativity," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 32, no. 4, pp. 363–364, 2018.

[35] A. Pektaş and H. Cigizoglu, "ANN hybrid model versus ARIMA and ARIMAX models of runoff coefficient," *Journal of Hydrology*, vol. 500, pp. 21–36, 2013.