INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.180

Assessing Machine Learnability of Image and Graph Representations for Drone Performance Prediction

B. Song [1,✉], C. McComb [2] and F. Ahmed [1]

[1] Massachusetts Institute of Technology, United States of America,
[2] Carnegie Mellon University, United States of America

✉ binyangs@mit.edu

Abstract

Deep learning (DL) from various representations have succeeded in many fields. However, we know little about the machine learnability of distinct design representations when using DL to predict design performance. This paper proposes a graph representation for designs and compares it to the common image representation. We employ graph neural networks (GNNs) and convolutional neural networks (CNNs) respectively to learn them to predict drone performance. GCNs outperform CNNs by 2.6-8.1% in predictive validity. We argue that graph learning is a powerful and generalizable method for such tasks.

Keywords: engineering design, artificial intelligence (AI), design evaluation, design representation, graph convolutional networks

1. Introduction

Designers operate on a variety of representations. Ideally, a design representation should capture the nature of the design to support formal and functional reasoning. The common design representation methods include images (Burnap et al., 2016), voxels (Khokhlov et al., 2019), point clouds (Park et al., 2019), meshes (Zhang et al., 2020), tabular parameters (Kulfan, 2012), design grammars (Stump et al., 2019), graphs (Cao et al., 2020), and so on. For the same design, different representations can capture the elements of the design and the interactions between pairs of elements in different ways. These representations are often interchangeable with effort. Moreover, different representations afford the use of different computational models for design assistance. For example, we can employ convolutional neural networks (CNNs) and graph neural networks (GNNs) to learn features of designs represented by images and by graphs, respectively. While machine learning has made strides towards supporting designers, there has not been a critical comparison of the machine learnability of different representations for design. To bridge the gap, we use two common representation approaches, image-based and graph-based representations, to represent drone designs and compare their learnability with deep neural networks (DNNs) in this paper.

Meanwhile, design is undergoing a revolutionary shift driven by the recent advances in machine learning (ML), which aims to automate time-consuming, complex design tasks. A variety of ML models have been developed to support designers at different stages, such as user need elicitation (Song, Meinzer, et al., 2020), design synthesis (Chen and Ahmed, 2021), and product platform design (Song et al., 2018). Among them, a valuable focus of ML models is on design performance evaluation. Since theoretical and experimental approaches are often time-consuming and resource-demanding, accurate and fast ML-based design performance evaluation can lead to scalable evaluations, broader explorations, more informed decision making, and significant design cost reduction. For example, effective ML-based surrogate models for performance evaluation can benefit the development of deep generative design (GD) models (Regenwetter et al., 2021). Specifically, GD models based on various deep generative algorithms aim at synthesizing highly optimized and customized designs by learning from existing designs (Burnap et al., 2016; Chen and Ahmed, 2021). Design performance evaluation is a core component for training GD models to generate feasible, usable, and creative designs (Regenwetter et al., 2021). Many current models rely on external evaluation modules (e.g., physics-based simulation modules) to evaluate the generated designs (Stump et al., 2019), which slows down the training process. ML-based performance evaluation models that can be embedded into GD models can speed up the training process and promote the practical uses of GD models. Accordingly, this paper aims to study the potential of DNN models for design performance evaluation, which is specifically enabled by the machine learnability of graph and image representations.

In this paper, we develop the DNN models based on a set of drone designs from a prior study (Song et al., 2022) to predict multiple drone performance metrics, including flying range, velocity, and cost. The contributions of this paper include: (1) We propose a homogeneous graph representation enabling the use of GNNs for various design tasks, which applies to broad design problems involving graph-like structures and components with various size options; (2) We develop a GNN-based and a CNN-based regression models for accurate and fast design performance prediction of drones. The models are differentiable and can be integrated into other deep models (e.g., GD models) seamlessly to support complex design tasks (e.g., design synthesis, optimization); (3) We compare the machine learnability of the proposed graph representation to that of the commonly used image representation for design evaluation and demonstrate the advantages of the graph representation for such tasks. To this end, we provide an overview of design representation methods and corresponding DNN models for learning from them in Section 2, describe the method and data of this study in Section 3, report and discuss the results in Section 4, and conclude the study in Section 5.

2. Background

In this section, we review the design representations and the corresponding DNN models to capture features from different representations.

2.1. Design Representations

Designs can be represented in both structured and unstructured ways. As one type of common structured representation, image data is typically structured as rectangular matrices of square pixels, taking different values. Herein, we use the word image in a broader sense of any 2D figures, such as pictures of physical objects (Burnap et al., 2016), 2D renderings of 3D computer-aided design (CAD) models (Wen et al., 2016), and freehand sketches (Hannah et al., 2012). An image is often read by DNN models as a 3D tensor $[height \times width \times number\ of\ channels]$ (Regenwetter et al., 2021). The number of channels depends on the color scheme. Black-and-white images are represented by a single channel with Boolean values for pixels; grayscale images are represented by a single channel with integers from 0 to 255 for pixels; color images are represented by 3 or 4 channels, each representing one primary color channel, with integers from 0 to 255 for pixels. Image representations can capture detailed information of a design but pose challenges for downstream tasks, such as performance evaluation (Regenwetter et al., 2021). Besides, other structured design representations include 3D voxelized geometry data and parametric design data. 3D voxelized geometries are represented as 3D grids of voxels, which typically take Boolean values (Khokhlov et al., 2019). Researchers also use the corresponding key design parameters often structured in a tabular form to represent a design (Kulfan, 2012).

Multiple unstructured methods have also been employed to represent designs. Among them, graph representation is an attractive option because of its flexibility in representing complex systems and the interactions between elements within such systems (Barnes and Harary, 1983). Within a graph, each node represents an element of the system, which is often represented by a vector that indicates all the features of the node. The edge between a pair of nodes indicates the relations between the corresponding elements, which can be static or dynamic over time. Engineering researchers have employed graphs to represent CAD models (Cao et al., 2020) and mechanisms in machine design (Yang et al., 2018).

https://doi.org/10.1017/pds.2022.180 Published online by Cambridge University Press

In addition, other unstructured ways to represent designs include rule-based design grammars, point clouds, and meshes. The rules of a design grammar define how symbols and variables of the grammar indicate the spatial, configurational, and other design information (Stump et al., 2019). Design grammars have been applied in a broad context. A point cloud describes a 3D object as a set of points that define the external surface of the object, which are often generated by 3D scanning tools (Daneshmand et al., 2018). Similarly, 3D designs can also be represented by meshes, especially triangle meshes, for finite element analysis (Zhang et al., 2020).

In this paper, we employ two representations, images and graphs, to represent drone designs. In the following, we review the DNN models that are commonly used for learning these two types of representations, respectively.

2.2. Deep Neural Network Models

CNNs are commonly applied to image learning and generation tasks and have achieved great success. Convolutional layers are the core building blocks of a CNN, which uses a set of feature detectors (a.k.a., kernels or filters, 2D array of weights) to capture whether the features are present in the input data and generates a feature map as an output. This convolution process provides an effective and scalable way to capture features from images. A series of well-known nets signify the evolution of CNNs. LeNet-5 was one of the earliest CNNs that inspired the following work (LeCun et al., 1998). AlexNet improved computational efficiency by utilizing multiple GPUs and the Rectified Linear Units (ReLU) instead of Tanh as activation functions (Krizhevsky et al., 2012). Then, global average pooling was proposed in GoogLeNet to mitigate overfitting (Szegedy et al., 2015). Inception CNNs brought along multitasking that stacks filters with multiple sizes in a single layer to improve both performance and efficiency (Szegedy et al., 2016). VGGNet proposed to use low-dimensional filters (e.g., 3×3), marking the beginning of very deep CNNs (Simonyan and Zisserman, 2014). ResNet introduced residual learning and shortcut connections, improving accuracy and alleviating the vanishing-gradient problem (He et al., 2016). DenseNet proposed connecting each layer to every other layer to encourage feature propagation and reuse for reducing parameters and handling vanishing gradients (Huang et al., 2017). Now CNN is becoming the default method for image-related tasks (Voulodimos et al., 2018).

GNNs are a class of DNNs for extracting information from data represented by graphs and making predictions. As GNNs take graphs as input, the basic idea of GNNs is to refine the original node representations through propagating information between connected nodes. Accordingly, the learned node representations convey both the features of individual nodes and the topological relations between the nodes (Zhou et al., 2021). Within each layer, the representation of a node is updated by combining its own features with features of its neighboring nodes (Battaglia et al., 2018). The convolutional propagation operators are the core building blocks of a GNN, which generalizes convolutions from structured grid data (e.g., images) to unstructured graph data (Zhou et al., 2021). According to the approaches of generalizing convolutions, one can classify convolution operators as spectral operators and spatial operators. The spectral operators work with spectral representations of graphs through graph Fourier transform (Shuman et al., 2013). Graph convolutional network (GCN) simplified the convolution operation of the spectral representations to alleviate overfitting, which has become a widely used algorithm for learning graphs (Kipf and Welling, 2016). Spatial operators define convolutions according to the topology of graphs to overcome the differently sized neighborhoods for maintaining the permutation invariance (Zhou et al., 2021). Along with this approach, researchers have proposed to sample a fixed number of neighbors (Hamilton et al., 2017), to select a given number of top features through max pooling (Gao et al., 2018), or to normalize node degrees (i.e., number of edges of the node) through a transition matrix for updating node representations (Atwood and Towsley, 2015). One can use the obtained node representations directly for node-level tasks or pool them to generate graph representations for graph level tasks. Studying complex systems using GNNs is attracting growing research interest (Zhou et al., 2021).

According to the literature, little understanding has been developed about how effectively CNNs and GCNs can learn from images and graphs that represent the same designs, especially for performance prediction in the engineering design domain.

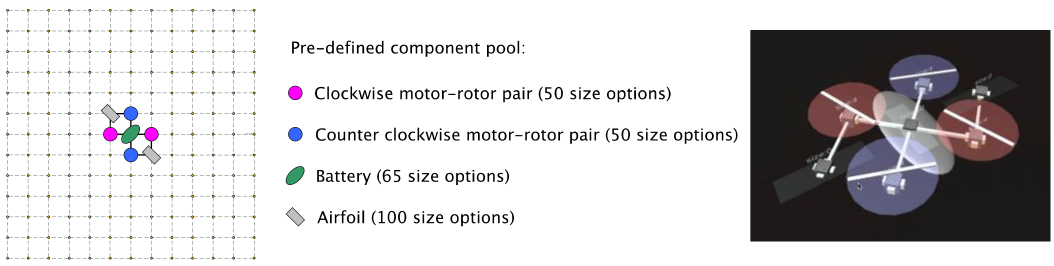
https://doi.org/10.1017/pds.2022.180 Published online by Cambridge University Press

3. Drone Representations and Computational Models

This paper studies the machine learnability of image and graph representations using a set of drone designs to predict drone performance. The dataset consists of 1,699 drone designs generated by human designers in response to a drone design challenge via a drone design research platform (Song, Soria Zurita, et al., 2020). Each drone was evaluated through multi-physics simulation[1], which returned three performance metrics, namely, flying range, velocity, and cost. The DNN models take the drone configuration and the payload carried by the drone as input. We train them on the drones to predict the simulated performance metrics, which is posed as a regression task. This paper uses images and graphs to represent the drone designs and employs CNNs and GCNs to learn the image and graph representations, respectively, for drone performance prediction.

3.1. Drone Representations

The studied drones were initially designed through the Unity game engine using a set of pre-defined components. The drone design space was defined by a 13×13 grid, as shown in Fig. 1(a). The pre-defined component pool contains four types of components, including clockwise motor-rotor pairs with 50 size options, counter-clockwise motor-rotor pairs with 50 size options, batteries with 65 size options, and airfoils with 100 size options. For constructing a drone, users can add a component to any grid node and connect it to other components. They can change the size of the component as well. A valid drone design is one that generates a balanced thrust enabling the drone to stay stable in the air and move forward. In the dataset, each drone consists of 9 components on average, and the most complex drone is composed of 31 components. Such a pre-defined drone design space enables us to represent different drone designs using images and graphs. An example drone is shown in the design space in Fig. 1(a) and the 3D model of the drone is shown in Fig. 1(b).


Pre-defined component pool:
Clockwise motor-rotor pair (50 size options)
Counter clockwise motor-rotor pair (50 size options)
Battery (65 size options)
Airfoil (100 size options)

(a) Drone design space (b) The drone model

Figure 1. Drone design space with an example drone and the pre-defined component pool

3.1.1. Image representation

Via image representation, we use matrices of pixels to represent the drone configuration and payload. Specifically, a drone configuration conveys information regarding the positions and sizes of different components composing the drone. Accordingly, three 13×13 matrices are used to specify the corresponding component type, size, and the associated payload at each grid node within the design space, respectively. In the component type matrix T, T_{ij} takes the value of 0 if there is no component at the node (i, j), the value 0.25[2] for a clockwise (CW) motor-rotor pair, the value 0.5 for a counter-clockwise (CCW) motor-rotor pair, the value of 0.75 for a battery, and the value of 1 for an airfoil. In the size matrix S, S_{ij} indicates the size of the component, which is normalized to the range (0,1][3]. In the payload matrix P, P_{ij} denotes the normalized average payload held by the component that is calculated by dividing the total

[1] https://github.com/hyform/drone-testbed-local-evaluation
[2] Originally, 1, 2, 3, and 4 respectively indicate clockwise motor-rotor pair, counter-clockwise motor-rotor pair, battery, and airfoil. These indicators are normalized into the range (0, 1].
[3] The maximal size option of each component corresponds to 1, while the smallest size option corresponds to $1/\text{number of size options}$.

payload by the number of components and normalizing it to the range (0, 1][4]. S_{ij} and P_{ij} take the value of 0 if no component is attached to the node (i, j). The drone representation using three matrices resonates with the image representation with red (R), green (G), and blue (B) channels. Fig. 2 demonstrates the component type, size, and payload matrices of the example drone shown in Fig. 1.

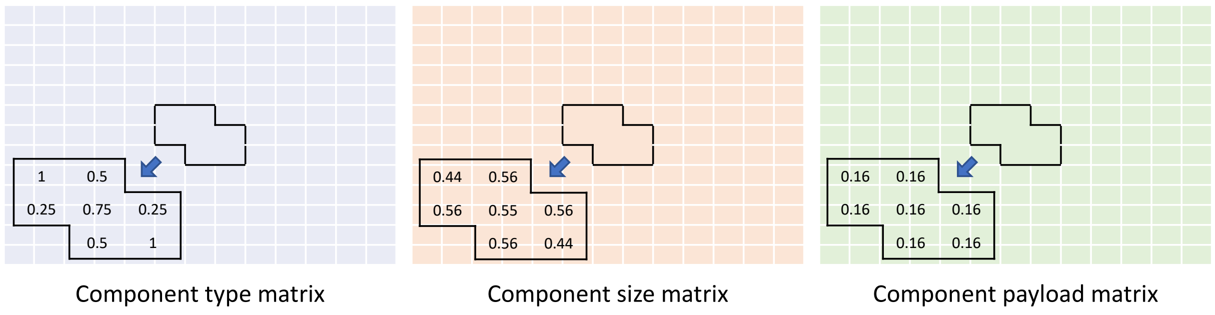


Figure 2. The component type, size, and payload matrices. The values in the highlighted area are shown in the enlarged view and all other cells take the value of 0.

3.1.2. Graph representation

In the graph representation, we embed the drone configuration and payload information in the graph's node features and topology. Each node within the graph represents a component of a drone. The features of the component are encoded into a vector with a dimension of 9, as illustrated in Fig. 3(a). Specifically, the first four elements of the vector are the one-hot encodings of the component type, each corresponding to one component type. The fifth to eighth elements indicate the normalized component size, each corresponding to one component type. The last element denotes the normalized average payload of each component. Fig. 3(b) presents the graph representation of the example drones shown in Fig. 1. In the drone graph, each node is connected to all other nodes, and the edge weights are calculated through three steps. First, we calculate the distance D_{mn} between a pair of nodes m and n using Equation 1, where the distance between two adjacent nodes in the grid is 1. Then, we normalize the distance against the maximal node distance D^{max} in the dataset following Equation 2. Third, we assume the edge weights follow a Gaussian distribution with a Gaussian width $\sigma = 0.2\pi$. Accordingly, the final edge weight is calculated through Equation 3:

$$Distance: D_{mn} = \sqrt{(m_i - n_i)^2 + (m_j - n_j)^2}; \tag{1}$$

$$Normalized\ distance: D_{mn}^{norm} = D_{mn}/D^{max}; \tag{2}$$

$$Weight = \exp\left(-D_{mn}^{norm\,2}/\sigma^2\right). \tag{3}$$

The obtained weights are within the range (0,1). The edges between nodes that are closer to each other present higher weights, as indicated by the edge width in Fig. 3(b). This is only one way to define the edge weight. The adjacency matrix A denotes the connectivity of a graph, where A_{mn} denotes the edge weight between nodes m and n. Through the approaches described above, we can represent each drone using an image or a graph.

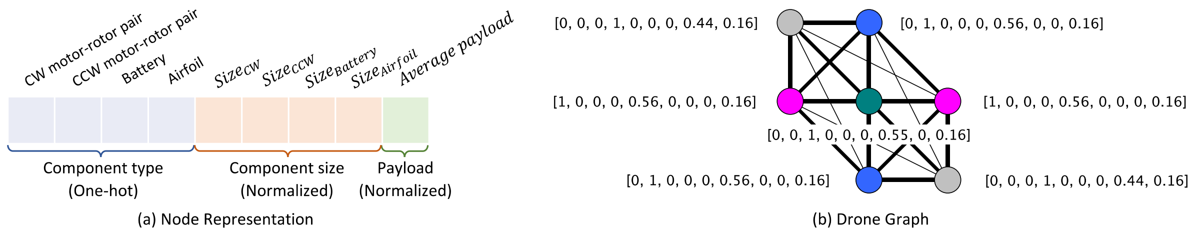


Figure 3. Drone graph representation.

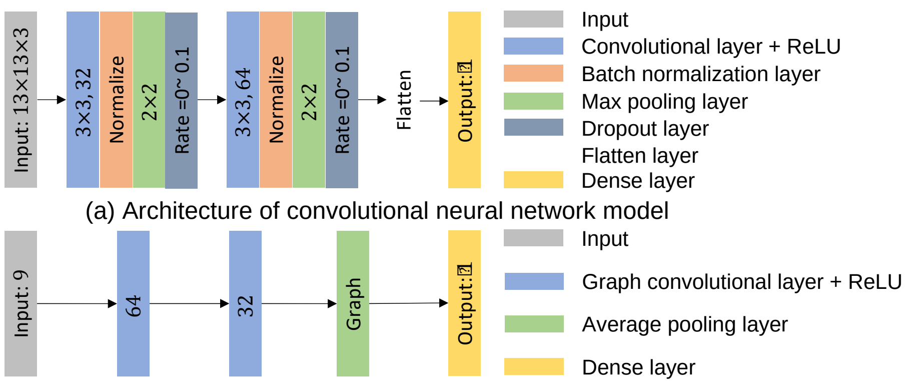
[4] The average payload was calculated for all drones and normalized together to the range (0, 1] by dividing each average payload value with the largest average payload value of all drones.

https://doi.org/10.1017/pds.2022.180 Published online by Cambridge University Press

3.2. Deep Neural Network Models

3.2.1. Convolutional neural network

We construct a CNN to learn the image representations of the drones. As shown in Fig. 4(a), the network consists of eleven layers, including an input layer, nine hidden layers, and an output layer. The input layer takes inputs with a shape of $13 \times 13 \times 3$. Among the hidden layers there are two convolutional layers, which respectively employ 32 and 64 kernels with a kernel size of 3×3 and use the ReLU activation function. Each of the convolutional layer is followed by a batch normalization layer, a max pooling layer with a pooling size of 2×2, and a dropout layer which takes varying dropout rate for different performance metric prediction[5]. A flatten layer is added after the convolutional modules. In the end, a dense output layer with the ReLU activation function returns a scalar value as the predicted performance score for each input.



(a) Architecture of convolutional neural network model

(b) Architecture of graph convolutional network model

Figure 4. Drone graph representation.

3.2.2. Graph convolutional network

Similarly, a GCN is developed to learn the graph representations of the drones. As shown in Fig. 4(b), the network consists of five layers, including an input layer, three hidden layers, and an output layer. The input layer takes a graph's node feature matrix with a shape of $n \times 9$ and the adjacency matrix with a shape of $n \times n$ as input, where n is the number of nodes of the graph and 9 is the dimension of the node feature vector. There are two graph convolutional layers with the ReLU activation function among the hidden layers, which respectively output feature vectors with dimensions of 64 and 32. Following the convolutional modules is an average pooling layer for generating the graph-level representation based on the representations of all nodes in the graph. Then, a dense output layer with the ReLU activation function returns a scalar value for each graph as the predicted performance score.
When training the models, we experimented with different hyperparameter values for both models. The selected values in Fig. 4 returned the best prediction outcomes compared with other options.

4. Results and Discussion

To train and validate the constructed CNN and GCN models, we divide the set of 1,699 drones into a training set, a validation set, and a test set following a split ratio of 0.7:0.15:0.15. The CNN and GCN models use the same data split for training and testing to predict flying range, payload, and cost. We train the models using the Adam optimizer and adopting the mean squared error (MSE) as the loss function. The CNN models employ a learning rate of 0.0005[6] for flying range and velocity and 0.005 for cost, while the GCN models use a learning rate of 0.01 for flying range and velocity and 0.1 for cost. The maximal number of training epochs is 3000 for both CNN and GCN models, but the training is ended earlier if the validation loss does not reduce within 50 consecutive epochs. We assess the performance of the DNN models in terms of their explanatory power for drone performance variabilities, i.e., r^2 values (Fig. 5 and

[5] The dropout rate is 0.1 for flying range, 0 for velocity, and 0.03 for cost.
[6] A lower value did not improve the results.

Fig. 6(b)), and the MSEs (Fig. 6(a)) between the predicted and simulated performance metrics. Pairwise two-tailed Student's t-tests are conducted to examine the statistical significance of the MSE differences between the CNN models and the GCN models across all drones (Fig. 6(a)). In the following, we report the performance of both models for predicting each drone performance metric.

4.1. Flying Range

During training, the CNN and GCN models improve dramatically in the first bunch of epochs for predicting flying range. The CNN model stops improving much earlier (about 70 epochs) than the GCN model, while the GCN model only improves slowly after the first 500 epochs until the end of the 3000 epochs. Then, we run the trained models to predict the flying ranges of the drones in the test set. Figs. 5(a) and 5(b) visualize the prediction results of the two models, respectively. We can see that there are more nodes locating closer to the $y = x$ line. Accordingly, the GCN model ($r^2 = 0.794$) exhibits a slightly higher r^2 value than the CNN model ($r^2 = 0.768$), indicating that the GCN model has higher predictive validity for flying range. Accordingly, the CNN model returns an MSE value of 65.7, which is slightly higher than that the GCN model (MSE = 59.1). The MSE difference between the two models is not statistically significant, as shown in Fig. 6(a).

4.2. Velocity

When training the models to predict velocity, we observe that the CNN model stops improving early (around 60 epochs). In contrast, the GCN model improves obviously during the first 200 epochs and improves slowly after that until near 1400 epochs. Then, we predict the velocities of the drones in the test set with the trained models. Figs. 5(c) and 5(d) show the prediction results of the two models, respectively. The nodes in the GCN plot follow the $y = x$ line better than those in the CNN plot as the simulated velocity increases. This is in line with the higher r^2 value exhibited by the GCN model ($r^2 = 0.596$) than that by the CNN model ($r^2 = 0.559$). In addition, the MSEs achieved by the CNN and GCN models are 33.9 and 30.12, respectively. The MSE difference between the two models is marginally significant ($p = 0.085$), as shown in Fig. 6(a).

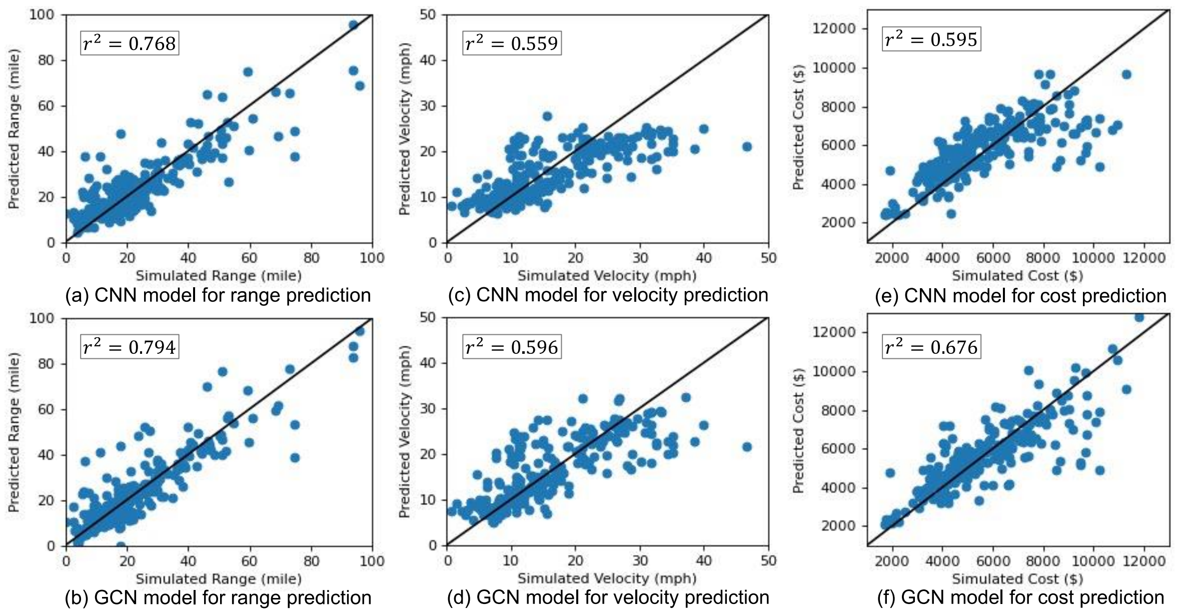


Figure 5. The prediction results of the CNN and GCN models.

4.3. Cost

During training for cost prediction, the CNN model improves obviously in the first 50 epochs and continues improving until near 200 epochs. In comparison, the GCN model improves prominently in the first 500 and slowly after that until near 1200 epochs. Then, we examine the performance of the trained models with the test set. Figs. 5(e) and 5(f) show the prediction results of the two models, respectively. On average, the nodes in the GCN plot are closer to the $y = x$ line than those in the CNN

https://doi.org/10.1017/pds.2022.180 Published online by Cambridge University Press

plot, especially as the cost approaches high values. This resonates with the higher r^2 value exhibited by the GCN model ($r^2 = 0.676$) compared to the CNN model ($r^2 = 0.595$). On average, the GCN model returns a lower MSE (1285040.5) than the CNN model (MSE = 1639504.8). According to the pairwise two-sided t-test, the MSE difference between the models is significant ($p = 0.015$) as shown in Fig. 6(a).

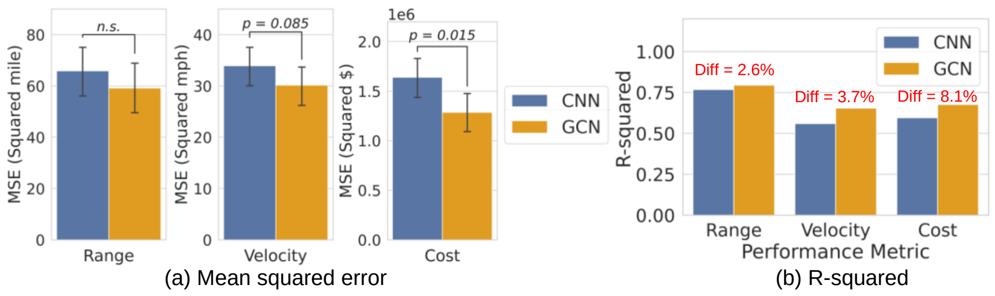


Figure 6. R-squared and mean squared error of the prediction results.

Fig. 6(b) compares the r^2 values between the CNN models and GCN models. In terms of model predictive validity (i.e., r^2 value), the GCN models that learn drone graph representations outperform the CNN models that learn drone image representations by 2.6-8.1% for drone performance prediction. There are two potential reasons. Firstly, the CNN models are subject to overfitting to a larger extent than the GCN models. During training the models for predicting each performance metric, the validation loss of the CNN model stops reducing much earlier than that of the GCN model, which indicates the overfitting of the CNN models is more severe than that of the GCN models. We can explain this by comparing the complexity of the CNN models and the GCN models. Fig. 4 shows that the architecture of the GCN models (3 hidden layers) is much simpler than that of the CNN models (9 hidden layers), which results in a big difference in the number of trainable parameters for the two sets of models - 2,945 for the GCN models and 19,814 for the CNN models. More trainable parameters require a larger volume of data to train. When we train the CNN and GCN models on the same dataset, the GCN models can avoid overfitting better.

Secondly, the GCN models can potentially learn features from the farther neighborhood. One CNN convolutional layer often capture features from elements in the near neighborhood. In contrast, one GCN convolutional layer can learn features from all elements in the neighborhood from near to far. When CNNs are not deep and the kernel size is small, their learning can be limited to only the near neighborhood. This can make the CNN models less effective in capturing relations between distant drone components. We observe that the performance of the GCN models seems to have more advantage over the CNN models when the flying range, velocity, and cost approach larger values, where the drones probably consist of more components and each component tends to have farther neighbors. The observation indicates that the GCN models are more capable of learning features embedded in the farther neighborhood or interactions between elements farther from each other. This resonates with technical literature on drone design, which has shown that rotors in a relatively long distance also affect each other (Aleksandrov and Penkov, 2012).

This study contributes to the design community in three distinct ways. First, it provides a homogeneous graph representation for engineering designs, which enables the use of GCNs to learn the designs for various tasks. It is straightforward to generalize the proposed graph representations to a broad variety of design problems involving graph-like structures and components with various size options, such as configuration design (e.g., linkage, truss) and robot design. Second, we develop a GNN-based and CNN-based regression models to embed drone designs represented by images and graphs for performance prediction. The results from this study show the great potential of the DNN models for accurate and fast design performance evaluation. Moreover, because the DNN models for design performance evaluation are differentiable, they can be integrated seamlessly into other deep models (e.g., GD models) for various complex design tasks, such as design synthesis and optimization. This integration can speed up the training process of such models and promote their practical applications. Third, we compare the machine learnability of the proposed graph representation to that of the image representation, which is commonly used in many machine-learning tasks, and demonstrate that the

https://doi.org/10.1017/pds.2022.180 Published online by Cambridge University Press

graph representation is more effective for such tasks. This finding is in line with other studies published recently that show the superiority of graph representations in different contexts (Pfaff et al., 2020; Shi and Rajkumar, 2020). We anticipate the application of graph representations to more design scenarios to verify their effectiveness more broadly.

However, it is also subject to a few limitations: (1) The dataset used to train and test the CNN and GCN models is small. A larger dataset can mitigate the overfitting problem to an extent. (2) It is hard to understand which of the two graph representation attributes (i.e., avoiding overfitting and capturing distant relations) contributes more to the strengths of the GCN models. Future work will train the models with varying data sizes to study the influence of overfitting and understand its contribution to the total improvement. (3) This study only examines one method of defining the edge weight in the graph representation. Future work should experiment with different methods and study how different edge weight definitions affect the effectiveness of the GCN models.

5. Conclusion

Designers operate on a variety of representations, including images and graphs. This paper proposes an image representation and a graph representation for drone designs to compare their machine learnability. Accordingly, we construct a CNN architecture and a GCN architecture to respectively learn the two representations for predicting multiple performance metrics of drone designs. The findings show that the graph representation exhibits higher machine learnability than the image representation. The GCN models outperform the CNN models, which are most commonly used in various deep learning tasks, by 2.6-8.1% in terms of predictive validity. The difference in performance can be explained by the effectiveness of graph representation in capturing distant relations between design components and the simple architecture of the GCN models that helps avoid overfitting.

Acknowledgement

This material is based upon work supported by the Defense Advanced Research Projects Agency through cooperative agreement N66001-17-1-4064. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

Aleksandrov, D. and Penkov, I. (2012), "Optimal gap distance between rotors of mini quadrotor helicopter", Proceedings of the International Conference of DAAAM Baltic, pp. 251–255.

Atwood, J. and Towsley, D. (2015), "Diffusion-Convolutional Neural Networks", Advances in Neural Information Processing Systems, pp. 2001–2009.

Barnes, J.A. and Harary, F. (1983), "Graph theory in network analysis", Social Networks, Vol. 5 No. 2, pp. 235–244.

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., et al. (2018), "Relational inductive biases, deep learning, and graph networks", https://arxiv.org/abs/1806.01261v3.

Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R. and Papalambros, P.Y. (2016), "Estimating and Exploring the Product Form Design Space Using Deep Generative Models", Proceedings of the ASME Design Engineering Technical Conference, Vol. 2A-2016, https://doi.org/10.1115/DETC2016-60091.

Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A.R. and Pan, W. (2020), "Graph Representation of 3D CAD Models for Machining Feature Recognition with Deep Learning", Proceedings of the ASME Design Engineering Technical Conference, Vol. 11A-2020, https://doi.org/10.1115/DETC2020-22355.

Chen, W. and Ahmed, F. (2021), "MO-PaDGAN: Reparameterizing Engineering Designs for augmented multi-objective optimization", Applied Soft Computing, Vol. 113, p. 107909.

Daneshmand, M., Helmi, A., Avots, E., Noroozi, F., Alisinanoglu, F., Arslan, H.S., Gorbova, J., et al. (2018), "3D Scanning: A Comprehensive Survey", https://arxiv.org/abs/1801.08863v1.

Gao, H., Wang, Z. and Ji, S. (2018), "Large-scale learnable graph convolutional networks", Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, Vol. 18, pp. 1416–1424.

Hamilton, W.L., Ying, R. and Leskovec, J. (2017), "Inductive Representation Learning on Large Graphs", Advances in Neural Information Processing Systems, Neural information processing systems foundation, Vol. 2017, pp. 1025–1035.

Hannah, R., Joshi, S. and Summers, J.D. (2012), "A user study of interpretability of engineering design representations", Taylor & Francis , Vol. 23 No. 6, pp. 443–468.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), "Deep residual learning for image recognition", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2016, pp. 770–778.

Huang, G., Liu, Z., Maaten, L. Van Der and Weinberger, K.Q. (2017), "Densely Connected Convolutional Networks", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2017, pp. 2261–2269.

Khokhlov, M., Koh, I. and Huang, J. (2019), "Voxel Synthesis for Generative Design", Design Computing and Cognition '18, Springer International Publishing, pp. 227–244.

Kipf, T.N. and Welling, M. (2016), "Semi-Supervised Classification with Graph Convolutional Networks", 5th International Conference on Learning Representations, ICLR 2017.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012), "ImageNet Classification with Deep Convolutional Neural Networks", Vol. 25, pp. 1–9.

Kulfan, B.M. (2012), "Universal Parametric Geometry Representation Method", Journal of Air Craft, Vol. 45 No. 1, pp. 142–158.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), "Gradient-based learning applied to document recognition", Proceedings of the IEEE, Vol. 86 No. 11, pp. 2278–2323.

Park, J.J., Florence, P., Straub, J., Newcombe, R. and Lovegrove, S. (2019), "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation".

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. W. (2020). Learning Mesh-Based Simulation with Graph Networks. ArXiv:2010.03409 [Cs.LG].

Regenwetter, L., Nobari, A.H. and Ahmed, F. (2021), "Deep Generative Models in Engineering Design: A Review", https://arxiv.org/abs/2110.10863v1.

Shi, W., & Rajkumar, R. (2020). Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1711–1719.

Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A. and Vandergheynst, P. (2013), "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains", IEEE Signal Processing Magazine, Vol. 30 No. 3, pp. 83–98.

Simonyan, K. and Zisserman, A. (2014), "Very Deep Convolutional Networks for Large-Scale Image Recognition", 3rd International Conference on Learning Representations, ICLR.

Song, B., Luo, J. and Wood, K. (2018), "Data-Driven Platform Design: Patent Data and Function Network Analysis", Journal of Mechanical Design, Vol. 141 No. 2, p. 021101.

Song, B., Meinzer, E., Agrawal, A. and McComb, C. (2020a), "Topic Modeling and Sentiment Analysis of Social Media Data to Drive Experiential Redesign", Proceedings of the ASME Design Engineering Technical Conference, Vol. 11A-2020.

Song, B., Soria Zurita, N. F., Nolte, H., Singh, H., Cagan, J., & McComb, C. (2021). When Faced with Increasing Complexity: The Effectiveness of Artificial Intelligence Assistance for Drone Design. Journal of Mechanical Design, 144(2). https://doi.org/10.1115/1.4051871.

Song, B., Soria Zurita, N. F., Zhang, G., Stump, G., Balon, C., Miller, S. W., … McComb, C. (2020b). Toward hybrid teams: a platform to understand human-computer collaboration during the design of complex engineered systems. Proceedings of the Design Society: DESIGN Conference, 1, 1551–1560. https://doi.org/10.1017/dsd.2020.68.

Stump, G.M., Miller, S.W., Yukish, M.A., Simpson, T.W. and Tucker, C. (2019a), "Spatial Grammar-Based Recurrent Neural Network for Design Form and Behavior Optimization", Journal of Mechanical Design, Vol. 141 No. 12, p. 124501.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., et al. (2015), "Going deeper with convolutions", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), "Rethinking the Inception Architecture for Computer Vision".

Voulodimos, A., Doulamis, N., Doulamis, A. and Protopapadakis, E. (2018), "Deep Learning for Computer Vision: A Brief Review", Computational Intelligence and Neuroscience, Vol. 2018, https://doi.org/10.1155/2018/7068349.

Wen, R., Tang, W. and Su, Z. (2016), "A 2D engineering drawing and 3D model matching algorithm for process plant", ICVRV 2015, pp. 154–159.

Yang, W., Ding, H., Zi, B. and Zhang, D. (2018), "New Graph Representation for Planetary Gear Trains", Journal of Mechanical Design, Vol. 140 No. 1.

Zhang, Z., Wang, Y., Jimack, P.K. and Wang, H. (2020), "MeshingNet: A New Mesh Generation Method Based on Deep Learning", Lecture Notes in Computer Science, Springer, Cham, Vol. 12139 LNCS, pp. 186–198.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., et al. (2021), "Graph neural networks: A review of methods and applications", https://doi.org/10.1016/j.aiopen.2021.01.001.