MIT Open Access Articles

Toward Reusable Surrogate Models: Graph-Based Transfer Learning on Trusses

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

Citation: Whalen, Eamon and Mueller, Caitlin. 2022. “Toward Reusable Surrogate Models: Graph-Based Transfer Learning on Trusses.” Journal of Mechanical Design, 144 (2).

As Published: 10.1115/1.4052298

Publisher: ASME International

Persistent URL: https://hdl.handle.net/1721.1/145570

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of Use: Article is made available in accordance with the publisher’s policy and may be subject to US copyright law. Please refer to the publisher’s site for terms of use.
Toward Reusable Surrogate Models: Graph-Based Transfer Learning on Trusses

Surrogate models have several uses in engineering design, including speeding up design optimization, noise reduction, test measurement interpolation, gradient estimation, portability, and protection of intellectual property. Traditionally, surrogate models require that all training data conform to the same parametrization (e.g., design variables), limiting design freedom and prohibiting the reuse of historical data. In response, this article proposes graph-based surrogate models (GSMs) for trusses. The GSM can accurately predict displacement fields from static loads given the structure’s geometry as input, enabling training across multiple parametrizations. GSMs build upon recent advancements in geometric deep learning, which have led to the ability to learn on undirected graphs: a natural representation for trusses. To further promote flexible surrogate models, this article explores transfer learning within the context of engineering design and demonstrates positive knowledge transfer across data sets of different topologies, complexities, loads, and applications, resulting in more flexible and data-efficient surrogate models for trusses.

[DOI: 10.1115/1.4052298]

Keywords: graph neural network, geometric deep learning, transfer learning, surrogate model, metamodel, structural design, approximation-based optimal design, conceptual design, design methodology, design of experiments, design representation, expert systems, machine learning, metamodeling, simulation-based design, structural optimization

1 Introduction

Surrogate models, also known as metamodels, response surfaces, reduced-order models, approximation models, or emulators, are used extensively in engineering to approximate complex systems. In a typical workflow, training data are produced by running a design of experiment of physics-based simulations, after which a surrogate model is trained in a supervised manner to predict one or more of the simulated quantities. The trained surrogate model might then be used to speed up design optimization, estimate gradients (if the surrogate is differentiable), or share the system (e.g., to a lightweight/web platform or collaborator) easily and without divulging intellectual property. Generally, these surrogate modeling methods require that each design be represented as a fixed-length vector of design parameters (e.g., design variables). This requirement restricts the surrogate model to a single-design space, requiring the user to train a new surrogate model every time the parametrization changes.

Ideally, surrogate models would operate on more organic representations of geometry, enabling learning across design data from multiple sources. Many design processes are incremental in nature. The result is often several small, disjoint design studies that differ slightly in geometry, topology, or loading conditions. A more flexible surrogate model could be trained across design iterations, perhaps supplemented with historical designs from previous projects, and could be continuously updated as new data become available. The ability to learn across related projects would not only save computational resources but might also yield powerful insights that could not have been inferred from a single-design space. Such models would also grant engineers greater design freedom since design changes would not be restricted to the parametrization used to generate training data.

One challenge in developing such a model is choosing a geometry representation. An ideal representation would accommodate arbitrary changes to the geometry or topology and encode loads and supports to enable learning across load cases. A second challenge is quantifying the extent to which such a surrogate generalizes to new designs. Unlike with traditional parametrization, the notions of interpolation and extrapolation are not well defined for representations that span the set of all possible shapes. How might one determine which inputs are “safe” and which are not likely to yield quality predictions?

This study explores the use of graph neural networks (GNNs) as surrogate models for trusses. The proposed graph-based surrogate model (GSM) learns to predict a displacement field given only the geometry, supports, and loads as inputs. It is shown that the GSM can be trained on data from multiple design models (DMs) simultaneously, often outperforming GSMs trained on a single source. Transfer learning is then explored as an effective method to repurpose previously trained GSMs to new tasks. Both the generalizability and data efficiency of the GSM are improved with transfer learning, with positive transfer being observed across varying topologies, loads, complexities, and even different applications.

The key contributions of this study are as follows:

(1) GSMs, which operate directly on the geometry and do not require parametric design features, are proposed for the modeling of trusses.

(2) Transfer learning is shown to improve the GSMs data efficiency and generalizability, leveraging historical data to reduce the required number of simulations by one or two orders of magnitude.

(3) Various source/target pairs that arise naturally in a design context, including design data of varying topologies, loads, complexities, and applications, are used to demonstrate the utility of transfer learned GSMs in a real-world setting.

The remainder of this article is organized as follows: Sec. 2 reviews related work, Sec. 3 introduces the methodology of the GSM and a few naive alternatives used for comparison, Sec. 4...
outlines data generation methods and presents experimental results. Sec. 5 introduces transfer learning and presents further results, and Sec. 6 contains conclusions and ideas for future work.

The following terminology is used throughout the paper: Let design refer to a specific design concept of a structure (i.e., something that could be built), design model (DM) refer to a hand-parametrized design space, which can be sampled to generate designs, and surrogate model refer to a data-driven predictive model that learns to predict a structure’s engineering performance.

2 Related Work

Engineering surrogate modeling is a thoroughly explored topic with applications dating back to the 1980s. Conversely, transfer learning and geometric deep learning are relatively young research areas with hundreds of papers published in the last few years alone. The following is a brief review of what are considered to be the most relevant works to this one, but is by no means comprehensive.

2.1 Surrogate Modeling With Parametric Design Features. Surrogate models have been used in engineering design for several decades (see Res. [1–3] for a review). Some of the most common surrogate modeling algorithms include polyno-

mial regression [4], Kriging (also known as Gaussian processes) [5], radial basis functions [6], random forest [7], and neural networks [8]. Reference [9] compared several of these algorithms for civil engineering problems. Dimensionality reduction techniques have been used to derive more suitable parametrizations [10,11] and quantities of interest [12]. All of the aforementioned methods require that a design be represented as a fixed-length vector of para-

metric design features, restricting the feasible designs to some pre-
determined space. This study proposes a surrogate model that operates on the geometry directly and is thus not limited to a particu-

lar parametrization.

2.2 Surrogate Modeling Without Parametric Design Features. Recently, a few surrogate models have been proposed that do not rely on handcrafted design parameters. Reference [13] proposed using “knowledge-based” characteristics, which are inde-

pendent of design variables, as features. While this may enable the combination of training data from multiple design spaces, it still relies heavily on the user to craft useful characteristics. Other approaches have sought to learn on the geometry itself. The pursuit of deep learning methods for shape data has led to the ability to learn on several geometry representations, including shape descriptors, images, voxels, polycubes, signed distance func-

tions, point clouds, and graphs (see Refs. [14,15] for a review). Sur-

rogate models have been trained on images [16–21], voxels [22,23], and polycubes [24,25]. Images and voxels suffer from resolution problems and data loss due to rasterization. Polycubes solve this problem by mapping the geometry to a regular grid but are limited to fixed-topology data sets.

The advent of geometric deep learning techniques has enabled learning on non-Euclidian domains, which are generally more

2.3 Geometric Deep Learning: Learning on Graphs. Graph-based learning, for both shape analysis and other tasks, has recently received a lot of attention. Reference [33] introduced the term geometric deep learning to mean learning from non-Euclidian data structures such as graphs and point clouds. See Refs. [34,35] for a general survey on GNNs. Geodesic convolutional neural nets-

works (GCNN) [36] was the first framework to apply a GNN to meshed surfaces by leveraging convolutions over local geodesic patches. Anisotropic convolutional neural networks (ACNN) [37] defined similar patches based on anisotropic heat kernels, while MoNet [38] generalized these patches to user-defined pseudo coor-

dinates. FeaStNet [39] introduced an attention mechanism to perform “feature steering,” which acts as dynamic filtering over neighbors. Other notable extensions of GNNs to shapes include MeshCNN [40], which introduced learnable edge pooling and StructureNet [41], which introduced a graph-based encoder for hier-

archical part representations. The aforementioned frameworks were applied to geometry processing tasks including shape correspon-
dence, classification, and segmentation, whereas this study focuses on structural surrogate modeling.

2.4 Transfer Learning: Recycling Data. Transfer learning, where predictive models previously trained on source data are retrained on target data from a different domain, task, or distribu-
tion, is a widely applied concept in machine learning [42]. Deep learning models in particular often benefit from transfer learning due to their data-intensive nature [43]. Reference [44] addressed some of the particular challenges of transfer learning in graph neural networks. A few works have explored transfer learning in the context of engineering design. Reference [18] trained a convolu-
tional autoencoder on 2D wheel designs before retraining the encoder as a surrogate model, reducing the required number of simulations. Reference [19] first trained a model to predict the origi-
nal parametric design features of an artery before retraining it to predict the location of maximum stress. Reference [45] used a clus-
tering algorithm to identify which designs would make for useful source data when applying transfer learning to microprocessor perfor-

mance prediction. Reference [25] trained a surrogate model to predict the drag coefficient of 2000 primitive shapes before tuning the model on 54 car designs. This article differs from previ-
ous works in that it seeks to systematically quantify the effects of transfer learning on data efficiency and generalizability across several common source/target pairs in structural design.

3 Methodology: Surrogate Modeling With Graphs

The following section presents a new GSM for predicting the dis-
placement of trusses under static load.

3.1 Data Representation: Trusses as Graphs. This article proposes a graph-based representation of trusses, where a set of vertices $V = \{v_1, \ldots, v_n\}$ represent the joints and a set of edges $E \in V \times V \times V$ represent the bars. The set of vertices that share an edge with $v_i$ is referred to as its neighborhood and is understood to include $v_i$ itself. Each vertex $v_i$ is assigned a feature vector $x_i$ of length $r (x_i \in \mathbb{R}^r)$. The geometry of the truss is encoded by using the joints’ spatial coordinates $c_i \in \mathbb{R}^2$ as vertex features. Additional binary features indicate the presence of a support $s_i \in \{0, 1\}$ for load $l_i \in \{0, 1\}$ for each degree-of-freedom. The geometry, supports, and loads are thus encoded by the graph $G_0 = (V_0, E)$. The deformed structure
is represented by a topologically identical graph \( G_H = (V_H, E_H) \), where now the vertex features encode the displacements \( d_i \in \mathbb{R}^2 \) of each joint under static load. The proposed graph representation has three main advantages:

1. It encodes the exact spatial coordinates of the geometry
2. It facilitates arbitrary topologies
3. It does not rely on handcrafted design parameters

In contrast with Euclidean representations like images, (1) implies that there is no information loss when converting the geometry to or from the deep learning representation. (2) and (3) enable learning across multiple design spaces.

### 3.2 Convolutions on Graphs

The GSM’s primary mechanism is a graph-based convolutional layer. The FeaStNet [39] convolution was selected because it extends to arbitrary graph topologies, does not require the selection and precomputation of pseudo coordinates, and can be made transformation invariant in feature space. The latter implies that raw spatial coordinates can be used directly as input features without having to learn spatial invariance or transform all designs to a common pose. Geometric deep learning is an active field; it is likely that other graph-based learning methods are also suitable for this context and should be considered as future research.

### 3.3 The Graph-Based Surrogate Model

The proposed surrogate model learns to predict joint displacements given the geometry, supports, and loads as inputs. It does so by learning a map from an input graph \( G_0 = (V_0, E_0) \) to a topologically identical output graph \( G_H = (V_H, E_H) \). The surrogate model is implemented as a graph-based convolutional neural network built from a simple sequence of \( H \)-linear and FeaStNet convolutional layers (Fig. 1). All layers except the final one are followed by a rectified linear (ReLU) activation function. It is observed that batch normalization applied to the input, after each convolutional operation significantly improves prediction accuracy. The network architecture, layer dimensions, and number of attention heads per FeaStNet layer dictate the total number of learnable parameters.

### 3.4 A Naive Alternative: The Pointwise Surrogate

A second, simpler type of surrogate model was used to compare against the proposed graph-based method. This pointwise surrogate consists of several simple regression models, which each take the spatial coordinates of the structure’s joints (flattened into a vector) as inputs and predict a single scalar quantity. For a 2D truss with 15 nodes, this corresponds to training 30 regression models (for the \( x \) and \( y \) displacement of each node). The random forest algorithm was selected for this study, but any regression technique (e.g., Kriging, polynomials, radial basis functions) could be used. Note that the pointwise surrogate relies on a fixed ordering of joints and thus cannot be extended to multitopology data sets. Also, note that in the case where all designs are identically loaded, there is no benefit to including support or load information in the input, since the designs are represented by a single vector. The pointwise surrogate was implemented using the scikit-learn [46] random forest class using default settings.

### 3.5 A Baseline: Predicting the Mean

As an additional reference point, consider an even simpler predictive model that simply predicts the mean displacement across each joint in the training set. Throughout the article, the performance of this naive model is referred to as the baseline. Models that fail to beat the baseline effectively have no predictive value.

### 4 Characterizing the Graph-Based Surrogate Model

The following section presents a series of trials designed to characterize the prediction accuracy and generalizability of the proposed graph-based surrogate model.

#### 4.1 Data Generation and Filtering

Surrogate modeling is most advantageous for computationally intensive simulations; however, this study focuses on relatively simple designs because they more effectively depict the specific design scenarios used to evaluate the GSM (more on this in Sec. 5). A set of truss designs was generated as follows. First, a parametric design model of a simple two-dimensional truss was built using a combination of commercial [47] and open source [48] software. The truss is made of steel (\( E = 30.5 \) Msi) and consists of beams with constant cross section (\( A = 0.29 \) m², \( I = 2.3 \times 10^{-3} \) m⁴). A vertical static load of 11.1 kN is applied to all joints on the top of the truss, and simple supports are applied to two of the bottom joints (Fig. 2). The truss was parametrized using five handcrafted design variables \( p_1 \sim p_5 \), each perturbing the truss geometry in a particular way. Next, the design model was sampled 1000 times using a Latin hypercube, and the resulting designs were simulated with bar elements using linear elastic finite element analysis. Finally, the 10% of trusses with the largest maximum displacement (i.e., the worst-performing designs) were discarded. For the remainder of this...
Table 1 The difference in mean absolute error (Δ MAE) between the pretrained GSM and a GSM trained from scratch, averaged across all training sizes

<table>
<thead>
<tr>
<th>Trial</th>
<th>Target</th>
<th>Δ MAE (cm)</th>
<th>Δ MAE %</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>DM5</td>
<td>−0.087</td>
<td>−48.6</td>
</tr>
<tr>
<td>D</td>
<td>DM7</td>
<td>−0.0586</td>
<td>−40.0</td>
</tr>
<tr>
<td>D</td>
<td>DM9</td>
<td>−0.106</td>
<td>−34.1</td>
</tr>
<tr>
<td>E</td>
<td>End loads</td>
<td>−0.148</td>
<td>−25.5</td>
</tr>
<tr>
<td>F</td>
<td>Tower</td>
<td>−0.0219</td>
<td>−54.1</td>
</tr>
<tr>
<td>G</td>
<td>Bridge</td>
<td>−0.0157</td>
<td>−19.8</td>
</tr>
</tbody>
</table>

Note: Transfer learning improved prediction accuracy by 19–48%.

4.2 Training and Tuning. A GSM was trained to predict joint displacements given a truss design as input. The truss designs were randomly partitioned such that 68% were used for training, 12% were used for validation, and 20% were reserved for testing. The GSM was implemented with Pytorch Geometric [49] and trained for 100 epochs on a Tesla K80 GPU using the ADAM optimizer [50] and a mean squared error (MSE) loss function. Through a series of grid searches, the optimal architecture was found to be L16/C32/C64/C128/C256/C512/C256/C128/L64/L2, where L denotes a linear layer, C denotes a FeaStNet convolutional layer, and the numbers represent the length of the vertex feature vectors after passing through a given layer. Similarly, the optimal learning rate was found to be $1 \times 10^{-3}$ and the optimal number of FeaStNet heads was found to be 8 (see Table 2 in the Appendix for details).

The resulting model has 2.7 million training parameters. Throughout all trials, batch normalization was applied to the input, and after each convolutional layer, the ADAM weight decay was set to $1 \times 10^{-5}$, and the batch size was set to 256.

Four data transformation strategies were studied: standardization, log transformation, standardization followed by log transformation, and no transform. It was found that standardization alone yields the lowest testing MSE (Table 3 in the Appendix). To study the effects of including support and load information in the feature vectors, the model was trained once using spatial coordinates alone as features and compared to when spatial coordinates are used in addition to binary support or load features. It was found that including both support and load features in the feature vectors improves prediction accuracy despite the fact that all trusses were loaded identically.

The trained GSM was implemented with Pytorch Geometric [49] and trained for 100 epochs on a Tesla K80 GPU using the ADAM optimizer [50] and a mean squared error (MSE) loss function. Through a series of grid searches, the optimal architecture was found to be L16/C32/C64/C128/C256/C512/C256/C128/L64/L2, where L denotes a linear layer, C denotes a FeaStNet convolutional layer, and the numbers represent the length of the vertex feature vectors after passing through a given layer. Similarly, the optimal learning rate was found to be $1 \times 10^{-3}$ and the optimal number of FeaStNet heads was found to be 8 (see Table 2 in the Appendix for details).

The resulting model has 2.7 million training parameters. Throughout all trials, batch normalization was applied to the input, and after each convolutional layer, the ADAM weight decay was set to $1 \times 10^{-5}$, and the batch size was set to 256.

Four data transformation strategies were studied: standardization, log transformation, standardization followed by log transformation, and no transform. It was found that standardization alone yields the lowest testing MSE (Table 3 in the Appendix). To study the effects of including support and load information in the feature vectors, the model was trained once using spatial coordinates alone as features and compared to when spatial coordinates are used in addition to binary support or load features. It was found that including both support and load features in the feature vectors improves prediction accuracy despite the fact that all trusses were loaded identically (Table 4 in the Appendix). This is understandable since the convolution can be thought of as acting on one vertex at a time.

Effective surrogate models should generalize well to unseen designs. For surrogate models that rely on bounded, handcrafted design parameters, one might assess generalizability simply by sampling the design space with sufficient density. In contrast, graph representations span the set of all conceivable trusses and thus a bounded design space does not exist. Developing practical intuition regarding the extent to which graph-based surrogate models generalize to new designs is an open challenge.

Toward this end, a series of data sets and trials were designed to test the generalizability of the GSM under a variety of conditions. The truss design model from Sec. 4.1 (DM7) was modified to create four new design models. The new design models, named DM5, DM6, DM8, and DM9 for the number of bars along the top, have identical outer profiles as DM7 but differing topologies (Fig. 4).

The following trials were designed to test the generalizability of the GSM. The reader is referred to Fig. 5 for an overview of the trials used throughout the rest of this article. Let the term target refer to the design model of interest to the user, that is, the design model from which the test set was generated. In Trial A, a GSM was trained and tested on designs generated from the target design model. Note that there is no overlap between the training and testing sets. In Trial B, training data from all of the design models were combined to train the GSM. The GSM was then tested on designs from the target design model as in Trial A. Trial B thus quantifies the GSM’s ability to learn on multiple design models simultaneously. Note that this would be impossible with the pointwise surrogate, which is limited to fixed-topology data. In Trial C, designs originating from the target design model...
were removed from the training set, thus testing the GSM’s ability to generalize to unseen design models. Trials A–C were repeated with each of the five design models (DMs 5–7) as the target, the results of which can be seen in Fig. 6.

In Trial A, the GSM archives a MAE of less than 0.1 cm for all design models, confirming the previous conclusion that the GSM effectively approximates single-design model data. Trial B also produced MAEs less than 0.1 cm across each design model, indicating that the GSM can learn on data from multiple design models simultaneously. Interestingly, for three of the design models (DMs 6–8), the inclusion of data from other design models actually improved predictions on the target. These results indicate that it is sometimes beneficial to add designs to the training data even if they are not from the design model of interest. Note that this did not hold true for DMs 5 or 9, which might be considered the most different from the rest of the design models in that the GSM does not seem to generalize well to unseen design models. It is possible that greater topological variation in the training set is required to learn such generalization.

Since the GSM is able to learn on multiple design models simultaneously, one might hope that the model generalizes well to previously unseen design models; however, this was not the case. The MAEs produced in Trial C were on average 76% higher than those in Trial B. While the mid-range topologies (DMs 6–8) showed better generalization than the extremes (DMs 5, 9), the general trend was that removing all target designs from the training data significantly reduces predictive performance. In other words, the GSM does not seem to generalize well to unseen design models. It is possible that greater topological variation in the training set is required to learn such generalization.

### 5 Transfer Learning: Repurposing the Graph-Based Surrogate Model

In light of the GSM’s poor generalization to unseen design models, one might conclude that a separate GSM must be trained for each potential target; luckily this is not the case. This article proposes transfer learning as a means of repurposing previously trained GSMs to new targets, using a fraction of the training data required to train a GSM from scratch. Transfer learned GSMs thus reduce the number of required simulations and training epochs and enable learning across design models, load conditions, and even separate applications. This section demonstrates the benefits of applying transfer learning to GSMs through four trials that emulate common design scenarios, namely, learning across small data sets that vary in topology, loads, application, or complexity.

#### 5.1 Effects on Generalizability

Consider a GSM that has been trained to predict the performance of one or more design models as described in Secs. 3 and 4. Let these design models now be referred to as the source. The subsequent trials demonstrate the performance of this GSM when retrained on a small training set from a new target design model (Fig. 7). Multiple strategies exist for applying transfer learning to neural networks. This study employs what is perhaps the most basic: simply retraining all learnable parameters on the target data set for an additional 100 epochs. The further exploration of transfer learning strategies, for example, those that freeze parameters or add new ones, for engineering design encouraged as the future work.

In Trial D, the GSMs that were previously trained (pretrained) on all design models, but the target (Trial C) are retrained on a small dataset (N = 200) from the target model. The results can be seen in the final series of Fig. 6. The retrained GSMs produce significantly better predictions than those in Trial C. In fact, the retrained GSMs on average produce 5.5% lower errors than those in Trial B and use less than a third of the training data. To further analyze the effects of transfer learning on prediction accuracy, the error
distribution from the pretrained GSM in Trial D was directly compared to that of a GSM trained only on the 200 design training set (without pretraining). The distributions are shown in Fig. 8. Pretraining on related source models reduces the average MAE across the test set by 70% and the standard deviation by 54%, resulting in a more accurate and robust surrogate.

5.2 Studying Data Efficiency. Effective surrogate models should achieve a useful level of prediction accuracy with a minimum amount of training data. The amount of both acceptable prediction error and available data are application dependent. Data efficiency is particularly important in engineering design, where quality design data are often scarce or prohibitively expensive to generate. On the other hand, deep learning methods, with their large number of trainable parameters, are notorious for requiring large data sets. This section explores the effects of transfer learning on the GSM’s data efficiency.

Trial D was repeated for a variety of target data set sizes. Each data set was generated as explained in Sec. 4.1, and the full 1000-design set was reserved for testing. To ensure that all data sets were similarly distributed, any designs with maximum displacements exceeding the 90th percentile from the test set were discarded. A different random seed was used in sampling the 1000-design training set to ensure that training and testing sets did not overlap. In addition to the pretrained GSM from Trial D, a second (not pretrained) GSM and a pointwise surrogate were trained on the target sets for comparison.

The mean absolute prediction errors as a function of training set size ($N$) can be seen in the top row of Fig. 9. For all models, prediction error correlates negatively with training size, which is expected. In nearly all cases, the pretrained GSM achieves the lowest prediction errors, followed by the pointwise surrogate and finally by the GSM trained from scratch. Transfer learning improves prediction MAEs by 48.6%, 40.0%, and 34.1% for DMs 5, 7, and 9, respectively. The implication is that the amount of training data required to achieve a given predictive performance is reduced by roughly one or two orders of magnitude. For DM 5, a pretrained GSM requires only 200 designs to achieve an MAE that is within 10% of the MAE produced by training on 1000 designs. For DM 7, just 100 designs were sufficient to achieve a similar result.

Interestingly, transfer learning was most beneficial for the medium-sized training sets. It is presumed that the smallest training sets do not sufficiently represent the differences between source and target distributions, while the largest training sets are sufficiently large to train a GSM to its predictive limit from scratch. Positive transfer was observed across all design models and training sizes, with the exception of the largest training set for DM 9 in which transfer learning increased MAE by 13.7%. This was the only observed case of negative transfer throughout all trials.

The loss histories of both GSMs reveal further insights about the effects of transfer learning. Figure 10 shows the evolution of training and validation losses for both GSMs, plotted for four training set sizes. Note that the validation losses for the transfer learned GSM at epoch zero are initially high and comparable to an untrained model. At this point, the conditions are quite similar to those in Trial C. The model is attempting to generalize to an unseen topology. However, as training progresses, the transfer learned validation losses converge faster and to a lower value than those of the models trained from scratch. Roughly 30 epochs are sufficient to retrain a model, compared to 100 epochs without transfer learning, representing further computational savings.

Encouraged by the positive transfer observed in Trial D, one might ask “for which source and target data sets is transfer learning useful?” The DMs 5–9 differ in topology but have the same outer profile, supports, and loads. The following trials were designed to test other source/target differences that might occur in a design process. In Trial E, a GSM is pretrained on 1000 designs from DM 7 and retrained on identical geometry but with different loads (point loads at the ends as opposed to a uniform load across the top). Thus, Trial E tests the ability to transfer-learn across load cases. The remaining two trials test the ability to transfer-learn across domains. In Trial F, a GSM is again pretrained on DM 7 and retrained on a set of trussed towers. The towers were generated by sampling three handcrafted design parameters. Each is pinned at
the bottom and loaded horizontally on the remaining joints. The spanning trusses (DM5-9) and towers differ in topology and outer profile, but have a similar number of bars (27 and 26, respectively).

In Trial G, the GSM is pretrained on DM 7 and re-trained on a set of densely trussed bridges. The bridges each consist of 404 bars, making them significantly more complex than the trusses. The bridges are uniformly loaded across the top and simply supported at the bottom. The hyperparameters described in Sec. 4.2 were used for all trials with the exception of Trial G, which used a batch size of 128 and learning rate of $5 \times 10^{-4}$.

The results from Trials E, F, and G are shown in the bottom row of Fig. 9. In Trial E, pretraining on the same geometry but different load cases improved MAE by an average of 25.5% across all training sizes. In Trial F, pretraining on trusses improved MAE predictions on towers by an average of 54.1%, and in Trial G, the same process improved MAE predictions on bridges by 19.8%. The result is a significant reduction in the amount of required training data. For example, in Trial E, a GSM pretrained on trusses achieves better prediction accuracy when retrained on 20 tower designs as a GSM, which was trained on 500 towers from scratch. Table 1 summarizes the findings from Trials D–G. Positive transfer was observed across all trials and training sizes, although to varying degrees. As before, the medium-sized training sets generally showed largest benefit and the smallest and largest data sets showed the least. These results further motivate the use of transfer learning to repurpose design data and surrogate models for new tasks.

6 Conclusions and Future Work

The proposed GSMs learn to predict displacement fields given a structure’s geometry, supports, and loads as inputs. Since the GSM does not rely on handcrafted design parameters, it can be trained on data from multiple design spaces simultaneously and often benefits from doing so. Transfer learning was presented as an effective method for repurpose GSMs to new tasks by leveraging historical data. GSMs that are pretrained on a related data set achieve 19–48% lower prediction errors than those trained from scratch. The result is a more flexible, general, and data-efficient surrogate model for trusses.

Future work could consider the increasingly wide array of graph-based learning methods and assess their suitability for trusses. A similar analysis could be performed for surface and volumetric meshes. Although both are easily represented as graphs, meshes differ from trusses in that the topology is not physically meaningful. In terms of transfer learning, further work is required to be able to predict the most effective sources for a given target. One might also explore alternative transfer learning strategies in which learnable parameters are added or frozen during retraining. The ability to learn across designs of varying complexity (Trial G) might support hierarchical learning strategies in which models are progressively trained on higher complexity designs. Finally, future work could explore ways of making GSMs generalize to unseen topologies, perhaps by leveraging alternative sources of training data like design competitions for more geometrically diverse data sets.

Acknowledgment

This research was supported by the Engineering Data Science group at Altair Engineering Inc. and is based upon work supported by the National Science Foundation under Grant No. 1854833. Also a special thanks to Renaud Danhaive and Yijiang Huang for their mentorship.

Conflict of Interest

There are no conflicts of interest.


Table 2 Grid search results used to tune the GSMS architecture, learning rate, and number of attention heads

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Learning rate</th>
<th>No of heads</th>
<th>Parameters</th>
<th>Train time (s)</th>
<th>Train MSE (cm²)</th>
<th>Test MSE (cm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>8</td>
<td>581406</td>
<td>1.11 × 10²</td>
<td>7.44 × 10⁻³</td>
<td>1.59 × 10⁻²</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>12</td>
<td>1151734</td>
<td>1.19 × 10²</td>
<td>6.75 × 10⁻³</td>
<td>1.30 × 10⁻²</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>4</td>
<td>581406</td>
<td>1.07 × 10²</td>
<td>5.51 × 10⁻³</td>
<td>9.13 × 10⁻³</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>8</td>
<td>1151734</td>
<td>1.15 × 10²</td>
<td>5.37 × 10⁻³</td>
<td>9.25 × 10⁻³</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>12</td>
<td>1722062</td>
<td>1.26 × 10²</td>
<td>5.88 × 10⁻³</td>
<td>9.23 × 10⁻³</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>4</td>
<td>581406</td>
<td>1.04 × 10²</td>
<td>1.02 × 10⁻²</td>
<td>1.26 × 10⁻²</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>8</td>
<td>1151734</td>
<td>1.15 × 10²</td>
<td>1.03 × 10⁻²</td>
<td>1.25 × 10⁻²</td>
</tr>
<tr>
<td>A9</td>
<td>1 × 10⁻⁴</td>
<td>12</td>
<td>1722062</td>
<td>1.26 × 10²</td>
<td>1.36 × 10⁻²</td>
<td>1.75 × 10⁻²</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>4</td>
<td>1371426</td>
<td>1.31 × 10²</td>
<td>5.04 × 10⁻³</td>
<td>1.32 × 10⁻²</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>12</td>
<td>4089050</td>
<td>1.76 × 10²</td>
<td>4.90 × 10⁻³</td>
<td>1.35 × 10⁻²</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>4</td>
<td>1371426</td>
<td>1.30 × 10²</td>
<td>4.69 × 10⁻³</td>
<td>8.79 × 10⁻³</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>8</td>
<td>2730238</td>
<td>1.54 × 10²</td>
<td>4.83 × 10⁻³</td>
<td>1.30 × 10⁻²</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>12</td>
<td>4089050</td>
<td>1.76 × 10²</td>
<td>4.90 × 10⁻³</td>
<td>1.35 × 10⁻²</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>4</td>
<td>1371426</td>
<td>1.30 × 10²</td>
<td>4.69 × 10⁻³</td>
<td>8.79 × 10⁻³</td>
</tr>
<tr>
<td>A10</td>
<td>1 × 10⁻⁴</td>
<td>8</td>
<td>2730238</td>
<td>1.54 × 10²</td>
<td>5.01 × 10⁻³</td>
<td>8.83 × 10⁻³</td>
</tr>
</tbody>
</table>

Note: A9 corresponds to L16/C12/C64/C128/C256/C256/C128/L64/L64/L2, A10 corresponds to L16/C12/C64/C128/C256/C512/C256/C128/C64/L64/L2, and A11 corresponds to L16/C12/C64/C128/C256/C512/C512/C128/L64/L64. The selected configuration (bold) produced the second-lowest MSE and had twice as many learnable parameters (and hence more expressivity) as the configuration with the lowest (underlined). The training and testing data were the same as in Sec. 4.2. Each configuration was averaged over three trials on a single Tesla K80 GPU.

Table 3 Grid search results used to assess various data transformations applied to the displacements before learning

<table>
<thead>
<tr>
<th>Standard scaler</th>
<th>Log transform</th>
<th>Train MSE (cm²)</th>
<th>Test MSE (cm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>5.17 × 10⁻²</td>
<td>5.66 × 10⁻²</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>2.48 × 10⁻²</td>
<td>2.97 × 10⁻²</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>3.02 × 10⁻¹</td>
<td>6.71 × 10⁻¹</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>7.50 × 10⁻⁴</td>
<td>1.30 × 10⁻⁴</td>
</tr>
</tbody>
</table>

Note: The training and testing data were the same as given in Sec. 4.2. Each configuration was averaged over three trials. The standard scaler (bold) resulted in the smallest test MSE.

Table 4 Grid search results used to assess whether it is advantageous to include binary features indicating supports or loads

<table>
<thead>
<tr>
<th>Include supports</th>
<th>Include loads</th>
<th>Train MSE (cm²)</th>
<th>Test MSE (cm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>4.40 × 10⁻³</td>
<td>1.08 × 10⁻²</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>5.49 × 10⁻³</td>
<td>1.12 × 10⁻²</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>6.04 × 10⁻³</td>
<td>1.02 × 10⁻²</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>4.65 × 10⁻³</td>
<td>9.38 × 10⁻³</td>
</tr>
</tbody>
</table>

Note: The training and testing data were the same as given in Sec. 4.2. Each configuration was averaged over three trials. Note that including support and load information (bold) is beneficial despite the fact that all designs are loaded identically.

References