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

Eamon Whalen*
Graduate Research Assistant
Computational Science and Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
Email: ewhalen@mit.edu

Caitlin Mueller
Associate Professor
Civil and Environmental Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
Email: caitlinm@mit.edu

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 paper 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, the paper 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.

Keywords: graph neural network; geometric deep learning; transfer learning; surrogate model; metamodel; structural design;

Nomenclature

DM Design Model
GNN Graph Neural Network
GSM Graph-based Surrogate Model
MAE Mean Absolute Error

*Address all correspondence to this author.

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 is produced by running a design of experiment (DOE) 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 which 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 becomes 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 work explores the use of graph neural networks 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 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 work are as follows:

1. Graph-based Surrogate Models (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 paper is organized as follows: section 2 reviews related work, section 3 introduces the methodology of the GSM and a few naive alternatives used for comparison, section 4 outlines data generation methods and presents experimental results, section 5 introduces transfer learning and presents further results, and section 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 [1–3] for a review). Some of the most common surrogate modeling algorithms include polynomial regression [4], kriging (also known as Gaussian processes) [5], radial basis functions [6], random forest [7] and neural networks [8]. [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 parametric design features, restricting the feasible designs to some pre-determined space. This work proposes a surrogate model that operates on the geometry directly and is thus not limited to a particular 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. [13] proposed using "knowledge-based" characteristics, which are independent 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 functions, point clouds, and graphs (see [14, 15] for a review). Surrogate 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 natural representations of geometry. [26] trained a surrogate model to predict lift and drag coefficients from 3D point clouds. While potentially useful for solid bodies, point clouds are not an adequate representation of trusses because they lack topological information. Other works have represented designs as graphs. [25] used a graph-based convolutional model to learn fluid dynamics on meshed surfaces, [27] used a similar approach to learn the structural behavior of a thin shell, and [28] learned material properties from graph-based microstructures. The closest existing work to this one is probably [29], in which graph representations of trusses were used to optimize cross section sizes for struc-

tural loads. The structures in [29] had constant loads and geometry (apart from the cross sections), whereas this study explores the flexibility of graph-based networks to generalize across various geometries, topologies and loads.

Other notable engineering applications of graph-based learning include feature recognition on 3D CAD [30], shape correspondence for additive manufacturing [31], and generation of design decision sequences [32]; however, these do not directly address surrogate modeling.

2.3 Geometric deep learning: learning on graphs

Graph-based learning, both for shape analysis as well as other tasks, has recently received a lot of attention. [33] introduced the term *geometric deep learning* to mean learning from non-Euclidian data structures such as graphs and point clouds. See [34,35] for a general survey on graph neural networks (GNNs). MoNet [36] was the first framework to apply a GNN to meshed surfaces by leveraging convolutions over local geodesic patches. ACNN [37] defined similar patches based on anisotropic heat kernels, while GCNN [38] generalized these patches to user-defined pseudo coordinates. 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 hierarchical part representations. The aforementioned frameworks were applied to geometry processing tasks including shape correspondence, classification, and segmentation, whereas this work focuses on structural surrogate modeling.

2.4 Transfer learning: Recycling data

Transfer learning, where predictive models previously trained on source data are re-trained on target data from a different domain, task, or distribution, 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]. [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. [18] trained a convolutional autoencoder on 2D wheel designs before retraining the encoder as a surrogate model, reducing the required number of simulations. [19] first trained a model to predict the original parametric design features of an artery before retraining it to predict the location of maximum stress. [45] used a clustering algorithm to identify which designs would make for useful source data when applying transfer learning to microprocessor performance prediction. [25] trained a surrogate model to predict the drag coefficient of 2,000 primitive shapes before tuning the model on 54 car designs. This paper differs from previous 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.

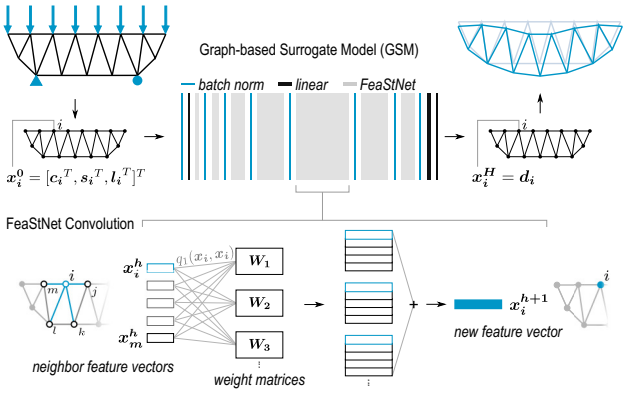

Fig. 1. The graph-based surrogate model (GSM) learns to predict nodal displacements given only geometry, supports and loads as inputs. Structures are represented as undirected graphs, where each vertex is assigned a feature vector consisting of a joint's spatial coordinates and binary variables indicating the presence of supports or loads. Graph convolutional layers utilize the FeaStNet operator [39].

3 Methodology: Surrogate modeling with graphs

The following section presents a new graph-based surrogate model (GSM) for predicting the displacement of trusses under static load.

3.1 Data representation: Trusses as graphs

This paper proposes a graph-based representation of trusses, where a set of vertices $V = \{v_1, ..., v_n\}$ represent the joints and a set of edges $E \in V \times V$ 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 $s_i \in \{0,1\}^2$ or load $l_i \in \{0,1\}^2$ 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 $G_H = (V_H, E)$, 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 Euclidian 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 pre-computation 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 (GSM)

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 $G_0 = (V_0, E)$ to a topologically identical output graph $G_H = (V_H, E)$. The surrogate model is implemented as a graph-based convolutional neural network built from a single 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 and 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 multi-topology 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 paper, 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 GSM

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

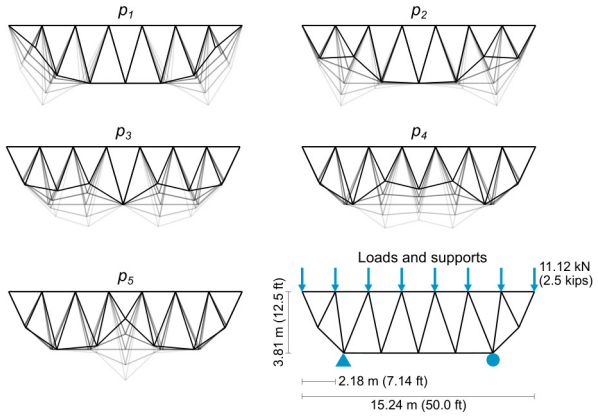

Fig. 2. A parametric design model of a truss. Data sets are created by perturbing design variables p_1-p_5. Each design is loaded with a uniformly distributed vertical load across the top and simply supported on the bottom. This particular design model is referred to as DM7.

4.1 Data generation and filtering

Surrogate modeling is most advantageous for computationally-intensive simulations; however, this work focuses on relatively simple designs because they more effectively depict the specific design scenarios used to evaluate the GSM (more on this in section 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 ($A = 0.29$ m^2, $I = 2.3\times10^{-3}$ m^4). 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-p_5, each perturbing the truss geometry in a particular way. Next, the design model was sampled 1,000 times using a Latin Hypercube and the resulting designs were simulated with bar elements using linear elastic Finite Element Analysis (FEA). Finally, the 10% of trusses with the largest maximum displacement (i.e. the worst-performing designs) were discarded. For the remainder of the paper, this design model will be referred to a design model 7 (DM7).

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 fea-

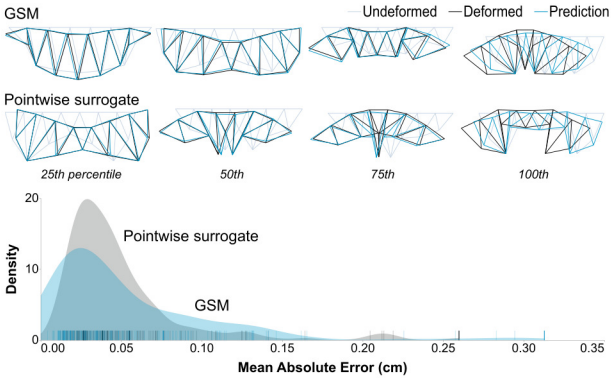

Fig. 3. The GSM and pointwise surrogate achieve comparable predictive performance on the test designs. Both error distributions are left-skewed, with 85% of designs producing a mean average error of less than 0.1 cm on either model.

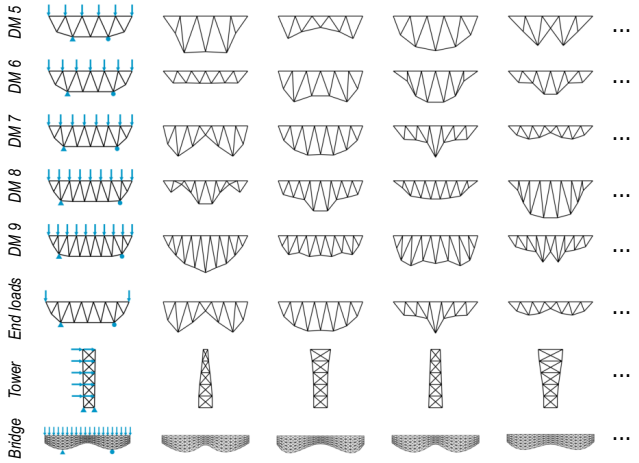

Fig. 4. Each row shows a few designs generated from one of the eight design models used in this paper. Loads and supports are omitted on all but the first column for clarity. The design models were selected to test specific scenarios that commonly arise in engineering design.

ture vectors after passing through a given layer. Similarly, the optimal learning rate was found to be 1×10^{-3} and the optimal number of FeaStNet heads was found to be 8 (see appendix Table 2 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 1×10^{-3}, 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 (appendix Table 3). 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 the support and load features in the feature vector improves prediction accuracy, despite the fact that all trusses were loaded identically (appendix Table 4). This is understandable, since the convolution can be thought of as acting on one vertex at a time.

4.3 Comparing the GSM to the pointwise surrogate

Both the GSM and pointwise surrogate successfully learn to predict a wide range of structural behaviors. Figure 3 shows the distribution of prediction errors for both models evaluated on the test set. The predictive performance of the two models is roughly comparable: the mean absolute error (MAE) over the entire test set is 0.049 cm for the GSM and 0.053 cm for the pointwise surrogate (30% and 33% of the baseline respectively). As the average maximum displacement of the trusses is 0.56 cm, these prediction errors are acceptable for most applications. The error distributions for both models are skewed left, implying that the models perform well on most of the designs but poorly on a few. Interestingly, it is observed that many of the designs for which prediction accuracies are low tend to also exhibit poor structural performance (i.e. large displacements).

4.4 Studying generalizability

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.

Towards 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 section 4.1 (DM7) was modified to create four new design models. The new design models, named DM5, DM6, DM8, 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 Figure 5 for an overview of the trials used throughout the rest of the paper. 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 was 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 Figure

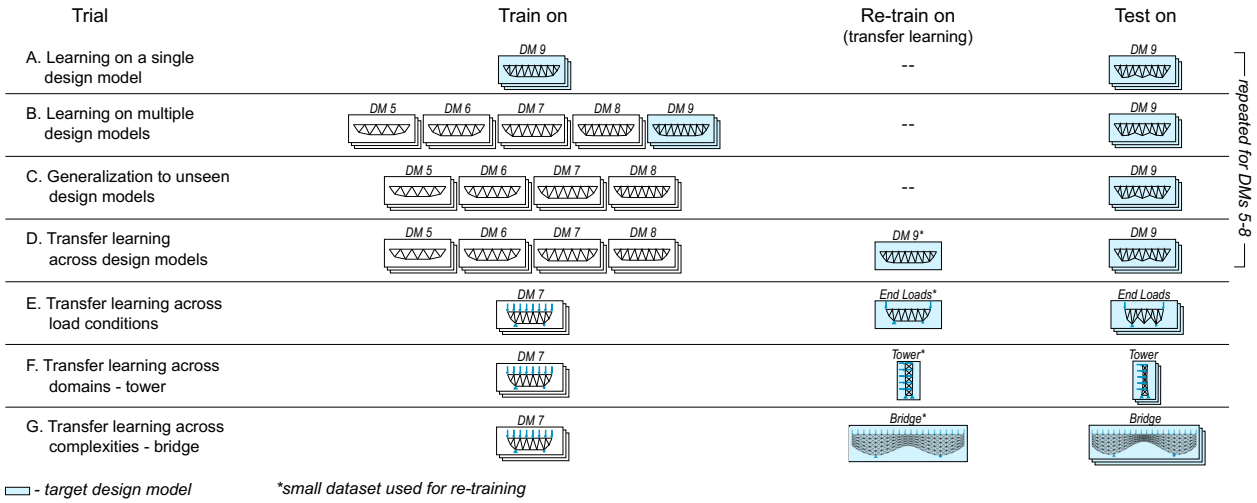

Fig. 5. An overview of the trials used to assess the GSM's generalizability across seven specific scenarios. The first three trials involve a single training, while the remainder of the trials leverage transfer learning to repurpose a previously trained GSM for new tasks. Note that trials *Trials B-G*) would not be possible with a traditional surrogate model because the data does not conform to the same parametrization.

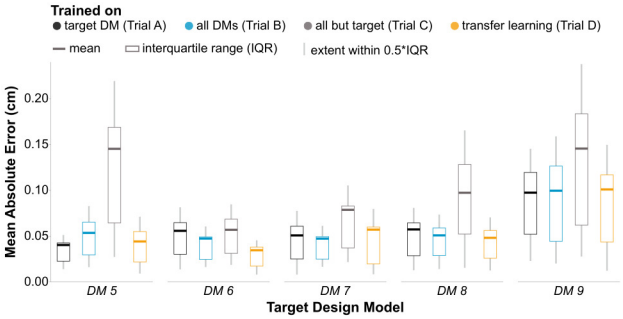

Fig. 6. The GSM can learn on data from multiple design models at once (*Trial B*), and doing so is sometimes advantageous even for cases when only a single design model is of interest. The GSM does not seem to generalize well to unseen design models (*Trial C*); however, transfer learning is an effective remedy (*Trial D*) and requires a fraction of the data required to train a GSM from scratch.

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 they have the fewest and most bars, respectively. The degree to which including off-target designs in the training data benefits training may therefore depend on how similar those designs are to the target.

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 in order to learn such generalization.

5 Transfer learning: repurposing the GSM

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 paper 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 which 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 sections 3 and 4. Let these design models now be referred to as the *source*. The subsequent trials demonstrate the performance of this GSM when re-trained 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

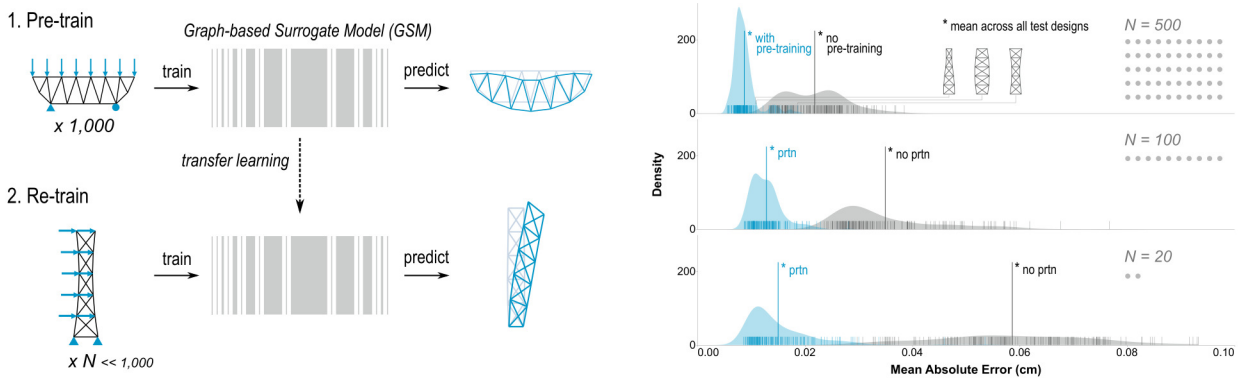

Fig. 7. Left: A previously trained Graph-based Surrogate Model (GSM) can be re-trained on a new data set with differing geometry, loads or topology. Right: Pre-training significantly increases the data efficiency of the GSM. In these results from *Trial F*, a pre-trained GSM trained on 20 designs (N=20) outperforms a fresh GSM trained on 500.

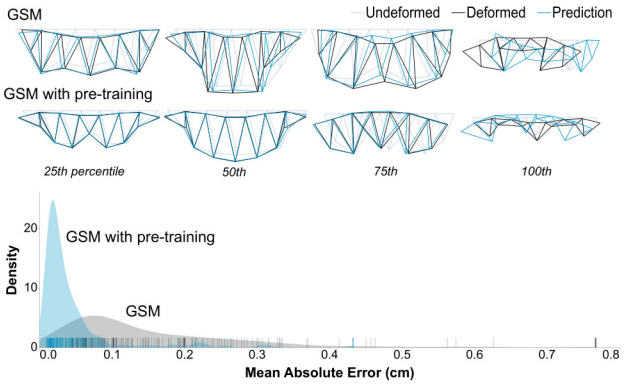

Fig. 8. A comparison of prediction error distributions between a pre-trained GSM and a GSM trained from scratch (*Trial D*, DM7). Both GSMs were trained on N=200 target designs. Visualizations of predictions representing each error quartile are also shown. Transfer learning reduces both the mean and standard deviation of prediction errors across the test designs.

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 future work.

In *Trial D*, the GSMs that were previously trained (pre-trained) on all design models but the target (*Trial C*) are re-trained on a small dataset (N=200) from the target model. The results can be seen in the final series of Figure 6. The re-trained GSMs produce significantly better predictions than those in *Trial C*. In fact, the re-trained 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 pre-trained GSM in *Trial D* was directly compared to that of a GSM trained only on the 200 design training set (without pre-training). The distributions can be seen in Figure 8. Pre-training 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 acceptable prediction error and available data are both application dependent. Data efficiency is particularly important in engineering design, where quality design data is 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 in section 4.1, and the full 1,000-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 1,000-design training set to ensure that training and testing sets did not overlap. In addition to the pre-trained GSM from *Trial D*, a second (not pre-trained) 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 Figure 9. For all models, prediction error correlates negatively with training size, which is expected. In nearly all cases, the pre-trained 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 pre-trained GSM requires only 200 designs to achieve an MAE that is within 10% of the MAE produced by training on 1,000 designs. For DM7, 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

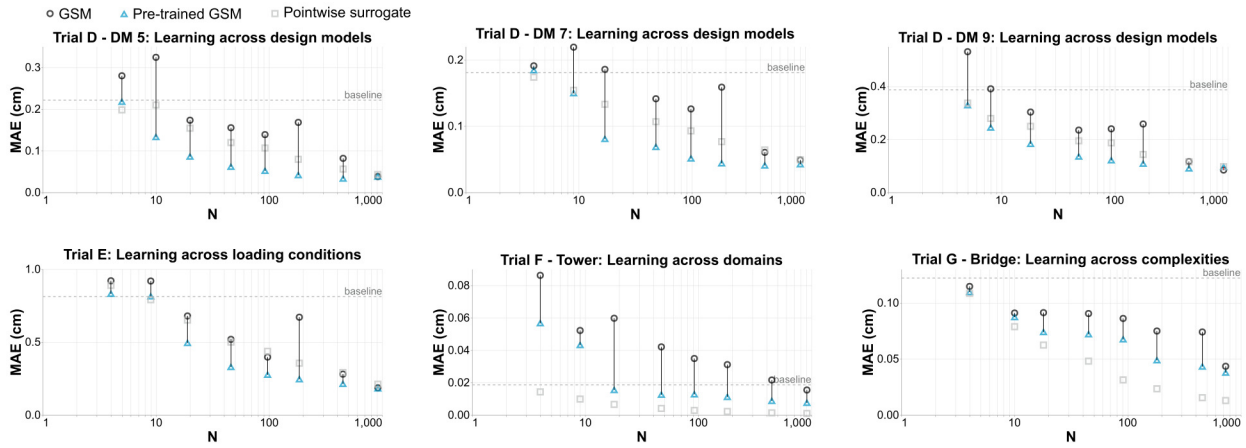

Fig. 9. Transfer learning consistently improved the GSMs data efficiency, reducing the amount of training data required to achieve a given prediction accuracy. The baseline refers to a naive model which always predicts the mean displacement from the 1,000 design training set.

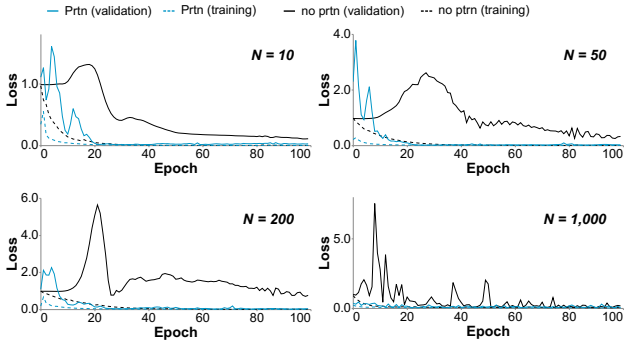

Fig. 10. Pre-trained (prtn) GSMs converge faster and to a lower loss value than those trained from scratch, particularly when the training size (N) is small. All curves taken from *Trial D*, design model 7 (DM7).

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 DM9 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 re-train 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 design models DM5-9 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 pre-trained on 1,000 designs from DM 7 and re-trained 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 pre-trained 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 pre-trained 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 section 4.2 were used for all trials with the exception of *Trial G*, which used a batch size of 128 and learning rate of 5×10^{-4}.

The results from *Trials E*, *F* and *G* are shown in the bottom row of Figure 9. In *Trial E*, pre-training on the same geometry but different load cases improved MAE by an average of 25.5% across all training sizes. In *Trial F*, pre-training 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 pre-trained on trusses achieves better prediction accuracy when re-trained 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.

Table 1. The difference in mean absolute error (Δ MAE) between the pre-trained GSM and a GSM trained from scratch, averaged across all training sizes. Transfer learning improved prediction accuracy by 19-48%.

Trial	Target	Δ MAE (cm)	Δ MAE %
D	DM5	-0.087	-48.6%
D	DM7	-0.0586	-40.0%
D	DM9	-0.106	-34.1%
E	End Loads	-0.148	-25.5%
F	Tower	-0.0219	-54.1%
G	Bridge	-0.0157	-19.8%

6 Conclusions and future work

The proposed Graph-based Surrogate models (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 pre-trained 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. Though 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 re-training. 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.

Acknowledgements

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.

References

[1] Wang, G. G., and Shan, S., 2007. "Review of Metamodeling Techniques in Support of Engineering Design Optimization". *Journal of Mechanical Design,* **129**(4), Apr., pp. 370–380.

[2] Forrester, A. I. J., Sóbester, A., and Keane, A. J., 2008. *Engineering design via surrogate modelling: a practical guide*. J. Wiley, Chichester, West Sussex, England ; Hoboken, NJ.

[3] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Kevin Tucker, P., 2005. "Surrogate-based analysis and optimization". *Progress in Aerospace Sciences,* **41**(1), Jan., pp. 1–28.

[4] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., 1989. "Design and Analysis of Computer Experiments". *Statistical Science,* **4**(4), Nov., pp. 409–423.

[5] Cressie, N., 1988. "Spatial prediction and ordinary kriging". *Mathematical Geology,* **20**(4), May, pp. 405–421.

[6] Dyn, N., Levin, D., and Rippa, S., 1986. "Numerical Procedures for Surface Fitting of Scattered Data by Radial Functions". *SIAM Journal on Scientific and Statistical Computing,* **7**(2), Apr., pp. 639–659.

[7] Tin Kam Ho, 1995. "Random decision forests". In Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, IEEE Comput. Soc. Press, pp. 278–282.

[8] Papadrakakis, M., Lagaros, N. D., and Tsompanakis, Y., 1998. "Structural optimization using evolution strategies and neural networks". *Computer Methods in Applied Mechanics and Engineering,* **156**(1-4), Apr., pp. 309–333.

[9] Tseranidis, S., Brown, N. C., and Mueller, C. T., 2016. "Data-driven approximation algorithms for rapid performance evaluation and optimization of civil structures". *Automation in Construction,* **72**, pp. 279–293. Publisher: Elsevier.

[10] Brown, N. C., and Mueller, C. T., 2019. "Design variable analysis and generation for performance-based parametric modeling in architecture". *International Journal of Architectural Computing,* **17**(1), Mar., pp. 36–52.

[11] Danhaive, R., and Mueller, C., 2021. "Design subspace learning: Structural design space exploration using performance-conditioned generative modeling". *Automation in Construction (in press)*.

[12] Xu, J., and Duraisamy, K., 2020. "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics". *Computer Methods in Applied Mechanics and Engineering,* **372**, p. 113379. Publisher: Elsevier.

[13] Xuereb Conti, Z., and Kaijima, S., 2018. "A flexible simulation metamodel for exploring multiple design spaces". In Proceedings of IASS Annual Symposia, Vol. 2018, International Association for Shell and Spatial Structures (IASS), pp. 1–8. Issue: 2.

[14] Ruizhongtai Qi, C., 2020. "Deep Learning on 3D Data". In *3D Imaging, Analysis and Applications*, Y. Liu, N. Pears, P. L. Rosin, and P. Huber, eds. Springer International Publishing, Cham, pp. 513–566.

[15] Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., and Ottersten, B., 2019. "A survey on Deep Learning Advances on Differ-

ent 3D Data Representations". *arXiv:1808.01462 [cs]*, Apr. arXiv: 1808.01462.

[16] Jiang, H., Nie, Z., Yeo, R., Farimani, A. B., and Kara, L. B., 2020. "StressGAN: A Generative Deep Learning Model for 2D Stress Distribution Prediction". In ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection.

[17] Messner, M. C., 2020. "Convolutional Neural Network Surrogate Models for the Mechanical Properties of Periodic Structures". *Journal of Mechanical Design,* **142**(2), Feb., p. 024503.

[18] Yoo, S., Lee, S., Kim, S., Hwang, K. H., Park, J. H., and Kang, N., 2021. "Integrating Deep Learning into CAD/CAE System: Generative Design and Evaluation of 3D Conceptual Wheel". *arXiv:2006.02138 [cs]*, Feb. arXiv: 2006.02138.

[19] Madani, A., Bakhaty, A., Kim, J., Mubarak, Y., and Mofrad, M. R. K., 2019. "Bridging Finite Element and Machine Learning Modeling: Stress Prediction of Arterial Walls in Atherosclerosis". *Journal of Biomechanical Engineering,* **141**(8), Aug., p. 084502.

[20] Garland, A. P., White, B. C., Jensen, S. C., and Boyce, B. L., 2021. "Pragmatic generative optimization of novel structural lattice metamaterials with machine learning". *Materials & Design*, Mar., p. 109632.

[21] Guo, X., Li, W., and Iorio, F., 2016. "Convolutional Neural Networks for Steady Flow Approximation". In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 481–490.

[22] Zhang, Z., Jaiswal, P., and Rai, R., 2018. "FeatureNet: Machining feature recognition based on 3D Convolution Neural Network". *Computer-Aided Design,* **101**, Aug., pp. 12–22.

[23] Williams, G., Meisel, N. A., Simpson, T. W., and McComb, C., 2019. "Design Repository Effectiveness for 3D Convolutional Neural Networks: Application to Additive Manufacturing". *Journal of Mechanical Design,* **141**(11), Nov., p. 111701.

[24] Umetani, N., 2017. "Exploring generative 3D shapes using autoencoder networks". In SIGGRAPH Asia 2017 Technical Briefs on - SA '17, ACM Press, pp. 1–4.

[25] Baque, P., Remelli, E., Fleuret, F., and Fua, P., 2018. "Geodesic Convolutional Shape Optimization". In Proceedings of the 35th International Conference on Machine Learning, J. Dy and A. Krause, eds., Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 472–481.

[26] Cunningham, J. D., Simpson, T. W., and Tucker, C. S., 2019. "An Investigation of Surrogate Models for Efficient Performance-Based Decoding of 3D Point Clouds". *Journal of Mechanical Design,* **141**(12), Dec., p. 121401.

[27] Danhaive, R., 2020. "Structural Design Synthesis Using Machine Learning". PhD thesis, Massachusetts Institute of Technology, Sept.

[28] Vlassis, N., Ma, R., and Sun, W., 2020. "Geometric deep learning for computational mechanics Part I: Anisotropic Hyperelasticity". *Computer Methods in Applied Mechanics and Engineering,* **371**, Nov., p. 113299. arXiv: 2001.04292.

[29] Chang, K.-H., and Cheng, C.-Y., 2020. "Learning to simulate and design for structural engineering". *arXiv:2003.09103 [cs, stat]*, Aug. arXiv: 2003.09103.

[30] 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". In Volume 11A: 46th Design Automation Conference (DAC), American Society of Mechanical Engineers, p. V11AT11A003.

[31] Huang, J., Sun, H., Kwok, T.-H., Zhou, C., and Xu, W., 2020. "Geometric Deep Learning for Shape Correspondence in Mass Customization by Three-Dimensional Printing". *Journal of Manufacturing Science and Engineering,* **142**(6), June, p. 061003.

[32] Raina, A., McComb, C., and Cagan, J., 2019. "Learning to Design From Humans: Imitating Human Designers Through Deep Learning". In Volume 2A: 45th Design Automation Conference, American Society of Mechanical Engineers.

[33] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P., 2017. "Geometric deep learning: going beyond Euclidean data". *IEEE Signal Processing Magazine,* **34**(4), July, pp. 18–42. arXiv: 1611.08097.

[34] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S., 2020. "A Comprehensive Survey on Graph Neural Networks". *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21. arXiv: 1901.00596.

[35] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., 2019. "Graph Neural Networks: A Review of Methods and Applications". *arXiv:1812.08434 [cs, stat]*, July. arXiv: 1812.08434.

[36] Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P., 2018. "Geodesic convolutional neural networks on Riemannian manifolds". *arXiv:1501.06297 [cs]*, June. arXiv: 1501.06297.

[37] Boscaini, D., Masci, J., Rodoià, E., and Bronstein, M., 2016. "Learning Shape Correspondence with Anisotropic Convolutional Neural Networks". In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., pp. 3197–3205. event-place: Barcelona, Spain.

[38] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M., 2016. "Geometric deep learning on graphs and manifolds using mixture model CNNs". *arXiv:1611.08402 [cs]*, Dec. arXiv: 1611.08402.

[39] Verma, N., Boyer, E., and Verbeek, J., 2018. "FeaSt-Net: Feature-Steered Graph Convolutions for 3D Shape Analysis". In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp. 2598–2606.

[40] Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D., 2019. "MeshCNN: A Network with an Edge". *ACM Transactions on Graphics,* **38**(4), July, pp. 1–12. arXiv: 1809.05910.

[41] Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., and Guibas, L. J., 2019. "StructureNet: Hierarchical Graph Networks for 3D Shape Generation". *arXiv:1908.00575 [cs]*, Aug. arXiv: 1908.00575.

[42] Pan, S. J., and Yang, Q., 2010. "A Survey on Transfer Learning". *IEEE Transactions on Knowledge And Data Engineering,* **22**(10), p. 15.

[43] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C., 2018. "A Survey on Deep Transfer Learning". *arXiv:1808.01974 [cs, stat]*, Aug. arXiv: 1808.01974.

[44] Lee, J., Kim, H., Lee, J., and Yoon, S. "Transfer Learning for Deep Learning on Graph-Structured Data". p. 7.

[45] Li, D., Wang, S., Yao, S., Liu, Y.-H., Cheng, Y., and Sun, X.-H. "Efficient Design Space Exploration by Knowledge Transfer". p. 10.

[46] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research,* **12**, pp. 2825–2830.

[47] Rutten, D. Grasshopper 3D. v6. Robert McNeel & Associates.

[48] Huang, Y., 2020. pyconmech - https://pypi.org/project/pyconmech/.

[49] Fey, M., and Lenssen, J. E., 2019. "Fast Graph Representation Learning with PyTorch Geometric". In ICLR Workshop on Representation Learning on Graphs and Manifolds.

[50] Kingma, D. P., and Ba, J., 2017. "Adam: A Method for Stochastic Optimization". *arXiv:1412.6980 [cs]*, Jan. arXiv: 1412.6980.

Appendix: GSM tuning results

Table 2. Grid search results used to tune the GSMs architecture, learning rate, and number of attention heads. A9 corresponds to *L16/C32/C64/C128/C256/C256/C128/L64/L2*, A10 corresponds to *L16/C32/C64/C128/C256/C512/C256/C128/L64/L2* and A11 corresponds to *L16/C32/C64/C128/C256/C512/C512/C256/C128/L64/L2*. The selected configuration (bold) produced the second-lowest mean squared error (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 section 4.2. Each configuration was averaged over three trials on a single Tesla K80 GPU.

Architecture	Learning Rate	# of Heads	Parameters	Train Time (s)	Train MSE (cm^2)	Test MSE (cm^2)
A9	1×10^{-4}	4	581406	1.11×10^{2}	7.44×10^{-3}	1.59×10^{-2}
A9	1×10^{-4}	8	1151734	1.19×10^{2}	6.75×10^{-3}	1.30×10^{-2}
A9	1×10^{-4}	12	1722062	1.30×10^{2}	8.70×10^{-3}	1.52×10^{-2}
A9	1×10^{-3}	4	581406	1.07×10^{2}	5.51×10^{-3}	9.13×10^{-3}
A9	1×10^{-3}	8	1151734	1.15×10^{2}	5.37×10^{-3}	9.25×10^{-3}
A9	1×10^{-3}	12	1722062	1.26×10^{2}	5.88×10^{-3}	9.23×10^{-3}
A9	1×10^{-2}	4	581406	1.04×10^{2}	1.02×10^{-2}	1.26×10^{-2}
A9	1×10^{-2}	8	1151734	1.15×10^{2}	1.03×10^{-2}	1.25×10^{-2}
A9	1×10^{-2}	12	1722062	1.26×10^{2}	1.36×10^{-2}	1.75×10^{-2}
A10	1×10^{-4}	4	1371426	1.31×10^{2}	5.04×10^{-3}	1.32×10^{-2}
A10	1×10^{-4}	8	2730238	1.54×10^{2}	4.83×10^{-3}	1.30×10^{-2}
A10	1×10^{-4}	12	4089050	1.76×10^{2}	4.90×10^{-3}	1.35×10^{-2}
A10	1×10^{-3}	4	1371426	1.30×10^{2}	4.69×10^{-3}	8.79×10^{-3}
A10	$\mathbf{1\times10^{-3}}$	**8**	**2730238**	$\mathbf{1.54\times10^{2}}$	$\mathbf{5.01\times10^{-3}}$	$\mathbf{8.83\times10^{-3}}$
A10	1×10^{-3}	12	4089050	1.76×10^{2}	6.45×10^{-3}	1.02×10^{-2}
A10	1×10^{-2}	4	1371426	1.31×10^{2}	9.54×10^{-3}	1.13×10^{-2}
A10	1×10^{-2}	8	2730238	1.53×10^{2}	1.04×10^{-2}	1.28×10^{-2}
A10	1×10^{-2}	12	4089050	1.75×10^{2}	1.56×10^{-2}	1.81×10^{-2}
A11	1×10^{-4}	4	2423590	1.61×10^{2}	5.01×10^{-3}	1.66×10^{-2}
A11	1×10^{-4}	8	4833030	1.99×10^{2}	4.49×10^{-3}	1.42×10^{-2}
A11	1×10^{-4}	12	7242470	2.38×10^{2}	5.13×10^{-3}	1.37×10^{-2}
A11	1×10^{-3}	4	2423590	1.59×10^{2}	5.40×10^{-3}	9.95×10^{-3}
A11	1×10^{-3}	8	4833030	1.99×10^{2}	5.44×10^{-3}	9.51×10^{-3}
A11	1×10^{-3}	12	7242470	2.36×10^{2}	6.72×10^{-3}	1.04×10^{-2}
A11	1×10^{-2}	4	2423590	1.59×10^{2}	1.30×10^{-2}	1.53×10^{-2}
A11	1×10^{-2}	8	4833030	1.97×10^{2}	1.54×10^{-2}	2.00×10^{-2}
A11	1×10^{-2}	12	7242470	2.35×10^{2}	1.99×10^{-2}	2.29×10^{-2}

Table 3. Grid search results used to assess various data transformations applied to the displacements before learning. The training and testing data were the same as in section 4.2. Each configuration was averaged over three trials. The standard scaler resulted in the smallest test MSE.

Standard scaler	Log transform	Train MSE (cm^2)	Test MSE (cm^2)
False	False	5.17×10^{-2}	5.66×10^{-2}
False	True	2.48×10^{-2}	2.97×10^{-2}
True	**False**	$\mathbf{3.02\times10^{-3}}$	$\mathbf{6.71\times10^{-3}}$
True	True	7.50×10^{-3}	1.30×10^{-2}

Table 4. Grid search results used to assess whether it is advantageous to include binary features indicating supports or loads. The training and testing data were the same as in section 4.2. Each configuration was averaged over three trials. Note that including support and load information is beneficial despite the fact that all designs are loaded identically.

Include supports	Include loads	Train MSE (cm^2)	Test MSE (cm^2)
False	False	4.40×10^{-3}	1.08×10^{-2}
False	True	5.49×10^{-3}	1.12×10^{-2}
True	False	6.04×10^{-3}	1.02×10^{-2}
True	**True**	$\mathbf{4.65\times10^{-3}}$	$\mathbf{9.38\times10^{-3}}$