An Investigation of Surrogate Models for Efficient Performance-Based Decoding of 3D Point Clouds

James D. Cunningham
Computer Science and Engineering,
Penn State University,
PA 16802
e-mail: jdc5549@psu.edu

Timothy W. Simpson
Engineering Design and Industrial and Manufacturing Engineering,
Penn State University,
PA 16802
e-mail: tws8@psu.edu

Conrad S. Tucker
Engineering Design and Industrial and Manufacturing Engineering,
Penn State University,
PA 16802
e-mail: ctucker4@psu.edu

This work investigates surrogate modeling techniques for learning to approximate a computationally expensive function evaluation of 3D models. While in the past, 3D point clouds have been a data format that is too high dimensional for surrogate modeling, by leveraging advances in 3D object autoencoding neural networks, these point clouds can be mapped to a one-dimensional latent space. This leads to the fundamental research question: what surrogate modeling technique is most suitable for learning relationships between the 3D geometric features of the objects captured in the encoded latent vector and the physical phenomena captured in the evaluation software? Radial basis functions (RBFs), Kriging, and shallow 1D analogs of popular deep 2D image classification neural networks are investigated in this work. We find the nonintuitive result that departing from neural networks to decode latent representations of 3D objects into performance predictions is far more efficient than using a neural network decoder. In test cases using datasets of aircraft and watercraft 3D models, the non-neural network surrogate models achieve comparable accuracy to the neural network models. We find that an RBF surrogate model is able to approximate the lift and drag coefficients of 234 aircraft models with a mean absolute error of 1.97×10^{-3} and trains in only 3 seconds. Furthermore, the RBF surrogate model is able to rank a set of designs with an average percentile error of less than 8%. In comparison, a 1D ResNet achieves an average absolute error of 1.35×10^{3} in 38 min for the same test case. We validate the comparable accuracy of the four techniques through a test case involving 214 3D watercraft models, but we also find that the distribution of the performance values of the data, in particular the presence of many outliers, has a significant negative impact on accuracy. These results contradict a common perception of neural networks as an efficient "one-size-fits-all" solution for learning black-box functions and suggests that even within systems that utilize multiple neural networks, potentially more efficient alternatives should be considered for each network in the system. Depending on the required accuracy of the application, this surrogate modeling approach could be used to approximate an expensive simulation software, or if the tolerance for error is low, it serves as a first pass which can narrow down the number of candidate designs to be analyzed more thoroughly. [DOI: 10.1115/1.4044597]

Keywords: computer-aided design, conceptual design, design automation, generative design, metamodeling

1 Introduction

Performance evaluation is a critical component of the design process as it enables designers to receive feedback with respect to performance criteria for candidate designs. However, because building and testing every candidate design is too expensive, computer simulations that approximate the test environment are typically used as an initial performance evaluation that is less expensive than building and testing. When the test environment is sufficiently complex, then even computer simulation can prove to be prohibitively costly for a large number of candidate designs. Additionally, computational fluid dynamics (CFD) is notorious for being computationally expensive. One CFD calculation can take days or even weeks to complete, even with modern computing power [1,2].

To mitigate the challenges of simulation, a further approximation of the testing environment is made by constructing a surrogate model that approximates the simulation environment. This surrogate model typically uses function approximation techniques such as polynomial regression, splines, radial basis functions (RBFs), neural networks (NNs), support vector machines, or Kriging (also called spatial correlation modeling) to approximate the simulation output [3].

Surrogate models are typically able to approximate a simulation with a high degree of accuracy such that they can serve as a substitute for the simulation. However, these existing surrogate modeling techniques often rely on a parameterized design space in which a complex 3D shape is distilled down to dozens or fewer design variables. The purpose of design parameterization is to characterize parameters that control the shape of a geometric model [4]. This frees the designer from the traditionally laborious task of directly manipulating the design geometry. Figure 1(*a*) shows a parameterized surrogate model, in which a set of human-defined parameters define the design space. This is contrasted in Fig. 1(*b*), in which the design space is instead represented by a set of points.

Design parameterization is a laborious task that requires a high degree of domain expertise, and design of methods to optimize parameterizations for specific applications is the subject of research in itself [4,6,7]. This makes it prohibitive to use parameterized

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received March 1, 2019; final manuscript received August 12, 2019; published online September 9, 2019. Assoc. Editor: Samy Missoum.

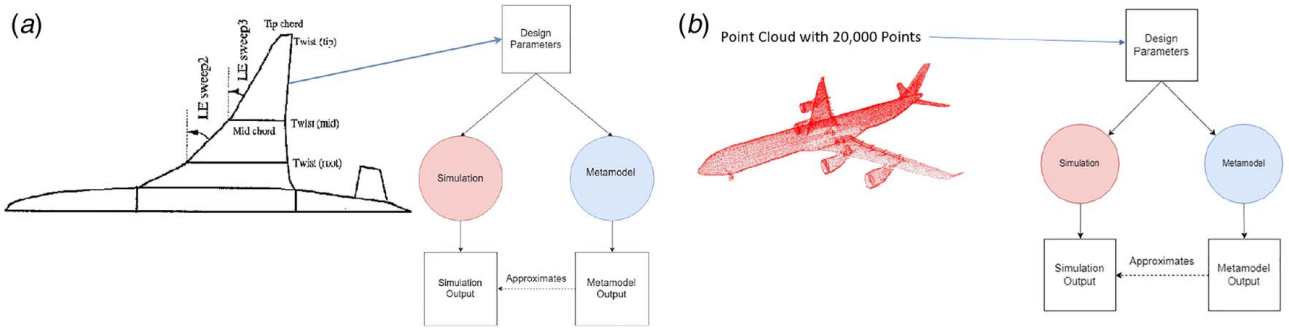

Fig. 1 Parameterized metamodeling techniques are well-suited to (*a*) parameterized design problems, but do not scale to (*b*) designs generated non-parameterized. For non-parameterized designs, a meta-modeling technique is required which can learn a surrogate function of high-input dimensionality [5].

surrogate modeling techniques when given a database of 3D shapes that are not parameterized such as in online shape repositories like ShapeNet [8] or ModelNet40 [9]. Additionally, the advent of deep generative neural networks (DGNNs) allows the direct generation of non-parameterized 3D designs [10,11]. These DGNN approaches to design have not used objective performance evaluation because of the difficulty in evaluating their large datasets of 3D models.

One popular approach to 3D object generation is autoencoding [12–14]. The idea is to take a 3D point cloud and learn to map it into a latent vector that preserves geometric information about the object. Then, in the case of PointNet [12], features from this latent vector are extracted for classification, and in the case of AtlasNet [13], the object is "decoded" back into the original object. While no notion of performance is embedded in this training process, we propose a surrogate model that can be thought of as replacing the decoders in each of these approaches with one that decodes the latent vector into a performance vector instead of back into a 3D object. This latent vector is typically of length 1024, and it represents a learned parameterization of the 3D space that can be difficult for a designer to intuitively describe. However, the dimensionality reduction of the 3D object provided by the autoencoder allows non-parameterized surrogate modeling techniques to learn form-function relationships to be employed on data in this format. Once trained, a surrogate model would allow for a large number of 3D models to be evaluated in seconds. In cases where the tolerance for error is very low, this approach can serve as a first pass estimate of the performance criteria and eliminate the need to do expensive simulation for a portion of the candidate designs. We postulate that this approach would be especially well-suited to evaluating designs from a non-parameterized generative model, such as a DGNN, as it would allow for many designs to be sampled without incurring the cost of evaluating every design in an expensive simulation environment.

Given that the form of a 3D object can be represented as a 1D latent vector of geometric features, and given that the function that the 3D object performs can be represented as a performance score that is acquired from a physics-based simulation, we investigate the following research questions:

- RQ1 Can the relationship between the 1D latent vector and the performance score be learned by a surrogate model?
- RQ2 Which surrogate modeling technique is most efficient at learning this form-function relationship?

The knowledge gained from answering these two research questions will enable the efficient evaluation of large datasets of non-parameterized 3D models. The rest of this paper is organized as follows. Section 2 reviews related literature and Sec. 3 details our method. Section 4 then applies this method to two case studies involving 3D models of aircraft and watercraft. Section 5 discusses the results before the conclusions and future work are discussed in Sec. 6.

2 Literature Review

In this section, related work in 3D object feature extraction and surrogate modeling is reviewed.

2.1 3D Feature Extraction. Deep neural networks have shown the ability to automatically extract useful features from high-dimensional input data. Architectures such as ShapeNet [8], VoxNet [15], and 3D ShapeNets [9] have achieved high classification accuracy on voxelized 3D objects, but because they rely on a voxelized representation of 3D space, the dimensionality of the network increases cubically with the voxel resolution. Approaches such as OctNet [16] and Vote3D [17] introduce methods to mitigate this problem, but they still have trouble processing large point clouds.

Another strategy is to convert a 3D object into a vector and then perform feature extraction on that vector for classification. Fang et al. [18] combined traditional shape feature extraction techniques with fully connected neural networks to classify 3D objects. Nash et al. [14] developed a variational autoencoder which models both 3D surface points and surface normals to generate surfaces.

Qi et al. [12] develop a method to directly convert unordered point sets into an encoded feature vector, which is invariant under transformations and captures local geometric structures of points. They show that this feature vector can be used to achieve state-of-the-art 3D object classification, part segmentation, and semantic segmentation. Groueix et al. [13] base their 3D shape encoder off of PointNet, which has achieved state-of-the art 3D object generation results by decoding the feature vector back into a 3D point cloud. Our work uses the AtlasNet point cloud encoder and then introduces a novel performance-based decoder that learns features that relate directly to a design's ability to meet a performance objective function and that can be evaluated far more efficiently than a computationally expensive performance evaluation environment. This allows designers to efficiently evaluate thousands of 3D models.

2.2 Non-Parameterized Surrogate Modeling. Metamodels approximate the output of costly simulations to allow a large sampling of the candidate design space to be explored. In their review of metamodeling techniques across many applications, Wang et al. [3] motivated metamodeling through the discussion of computationally expensive simulation. Many commonly used classical function identification techniques used in surrogate modeling are discussed, including RBFs, Kriging, and NNs. These three techniques are also identified by Mullur and Messac [19] as the most commonly used non-parameterized surrogate modeling techniques. We discuss related work for each of these techniques in Secs. 2.2.1–2.2.3.

2.2.1 Radial Basis Functions. RBF methods use linear combinations of a radially symmetric functions based on the Euclidean distance of training points. RBF methods have the desirable

properties of being able to accurately model arbitrary functions, handle scattered data points in multiple dimensions, and are known for their relatively simple implementation [19]. RBF methods have been demonstrated to be a highly effective surrogate modeling technique in terms of accuracy and robustness, and perform particularly well given a small number of training samples [20].

Ellbrant et al. [21] used an RBF-based metamodel to approximate the output of a CFD analysis tool that evaluates blade geometries given a design parameterized by ten variables. Their approach was shown to reduce the total design time from approximately 2 weeks to 3.5 days.

2.2.2 Kriging. Kriging models the response of a computer model as a linear model plus stochastic systematic departure [22]. Kleijnen [23] discussed the advantages of Kriging over classic linear regression models, he expands the discussion to included metamodeling of random simulation, in which the output of a computer simulation is not deterministic for a given input. Kriging is formulated as a generalized linear regression model that accounts for correlation in the residuals between the regression model and the direct data from the simulation. Simpson et al. [24] show the advantages of Kriging compared with second-order polynomial response surface models. Jia and Taflanidis [25] apply a Kriging metamodel to approximate simulations of high-dimensional wave and surge responses in real-time storm and hurricane risk assessment. The model takes five parameters and returns a risk estimate across many time instants and over a large coastal area, which is a high-dimensional output.

However, standard Kriging models are not suited to problems with high-input dimensionality, because a large covariance matrix must be inverted several times to estimate the model's parameters. We therefore employ Kriging with partial least squares (KPLS) for this application [26]. The PLS method is an established tool for high-dimensional problems that project the input and output variables into a smaller space using principal components [27]. By incorporating this information into the Kriging correlation matrix, the number of hyper-parameters is reduced and can be solved much more quickly [26].

2.2.3 Neural Networks. Neural networks have been well explored as non-parameterized surrogate models. For example, Caixeta and Marques [28] use a neural network metamodel-based multidisciplinary design optimization for wing design. Ferreiro et al. [29] developed a neural network metamodel that can make multi-criteria decisions to optimize for low environmental impact in the design of one-way slabs. Sreekanth and Datta [30] employ a neural network-based surrogate model for the multi-objective management of saltwater intrusion in coastal aquifers.

In recent years, neural networks have been improved to be able to achieve state-of-the-art results in classification tasks [31–35]. These improvements have also been leveraged for regression tasks such as speech enhancement [36,37], human pose estimation [38], and object tracking [39]. Kossaifi et al. [40] showed that by augmenting state-of-the-art classification networks such as ResNet [35] and VGG [34], their high performance will transfer to regression tasks.

These image classification networks have also been adapted to 1D data. Solovyev et al. [41] proposed 1D ResNet and VGG architectures for understanding simple speech commands. They classify a 16384×1 dimensional waveform into one of 12 categories. Hurtado et al. [42] also adapt ResNet and VGG to one dimension in order to classify protein models. These approaches both utilize datasets comparable in size to image datasets. Our application is unlike these approaches in that 3D model datasets are much smaller than image datasets, on the order of thousands instead of millions. Deep neural networks are non-parameterized models that are able to achieve extremely high accuracy across many applications, but they have the drawback of requiring a large amount of data. To mitigate this problem for our application, we implement our own shallower versions of ResNet and VGG adapted for one-dimensional data.

3 Method

This section details our proposed method to use a surrogate model to replace the decoder of a 3D autoencoder neural network to predict the performance of a 3D model from only a point cloud of the model. The surrogate modeling problem that our performance-based decoder solves is significantly different than the classification and reconstruction tasks addressed by the neural network decoders of other approaches [12,13]. This motivates the exploration of alternate decoder models that tie directly to performance. Section 3.1 describes the formulation of the overall performance prediction model, and Sec. 3.2 describes the surrogate modeling techniques explored in this work.

3.1 Performance Prediction Model Formulation. The prediction model is tasked with approximating the input–output relationship implied by an evaluation environment $F(\cdot)$, which is explicitly defined as follows:

$$\mathbf{y} = F(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ is a 3D model belonging to a dataset of encoded models $\mathbf{X}$, $\mathbf{y}$ is the performance metric vector, and $F(\cdot)$ is the function implied by the evaluation environment

Using this definition, the performance prediction network then implies the following function:

$$\hat{\mathbf{y}} = F^*(\mathbf{x}) \tag{2}$$

where $\hat{\mathbf{y}}$ is the estimated performance metric vector and $F^*(\cdot)$ is the approximation of $F(\cdot)$ implied by the surrogate model, and the task of the performance predictor is the optimization problem given by

$$\hat{\mathbf{y}}^* = \min_{\hat{\mathbf{y}}} |\mathbf{y} - \hat{\mathbf{y}}| \ \forall \mathbf{x} \in \mathbf{X} \tag{3}$$

As shown in Fig. 2, our performance prediction surrogate model takes as input an autoencoded latent representation of the 3D point cloud, which we represent as $\tilde{\mathbf{x}}$. This latent representation comes from the encoder portion of AtlasNet [13], which itself is based on PointNet [12], a network that provides state-of-the-art point cloud analysis results. This encoder network transforms a point cloud into a 1024 latent vector through a series of multilayer perceptrons as well as a single symmetric function, max pooling, to make

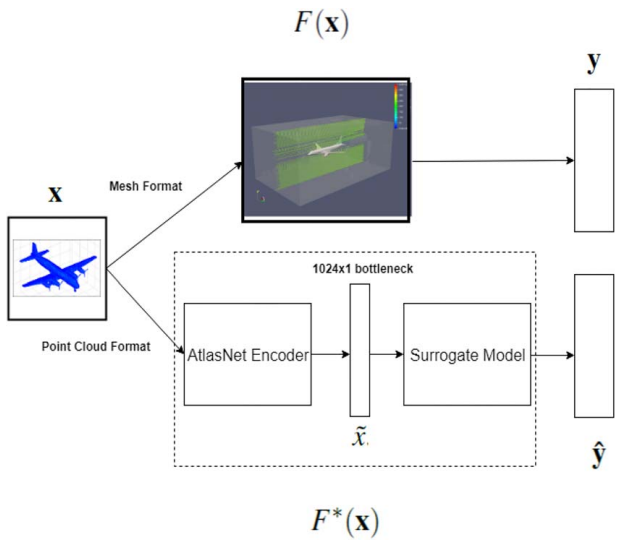

Fig. 2 Prediction model training iteration example

the model invariant to permutation of the input point cloud data [12]. This transformation can be thought of as a learned way to summarize a shape by a sparse set of keypoints. Once the geometric information of the point cloud has been compressed by the autoencoder, it is of a tractable dimension for surrogate modeling techniques.

The performance prediction network uses the regression loss function of mean absolute error (MAE) which allows the network to directly estimate the performance metric values. The MAE is defined as

$$MAE(\mathbf{z}, \hat{\mathbf{z}}) = \frac{1}{N_{samples}} \sum_{i=0}^{N_{samples}-1} |\mathbf{z}_i - \hat{\mathbf{z}}_i| \tag{4}$$

where $\mathbf{z}$ is the true performance metric vector and $\hat{\mathbf{z}}$ is the predicted performance metric vector

The prediction model training phase consists of computing the loss function $MAE(z, \hat{z})$ defined in Eq. (4) over a batch of designs from the training dataset and then repeating for many epochs. This process is illustrated in Fig. 2. In Sec. 3.2, we present the formulation of each of the surrogate models that will be used to decode the autoencoded geometry into a performance vector as shown in Fig. 2.

3.2 Surrogate Modeling Techniques. In this section, we discuss the surrogate modeling techniques explored for the performance decoder, which learns to transform $\mathbf{x}$ into $\tilde{\mathbf{x}}$. Sections 3.2.1 and 3.2.2 describe the implementations of RBF and KPLS, respectively, used in this work. We use the open-source Surrogate Modeling Toolbox [43] implementations of both of these methods. Section 3.2.3 describes the architectures of our customized 1D implementations of ResNet and VGG. By comparing each of the surrogate modeling techniques in this section, we will be able to answer RQ2 by examining whether any of the proposed models outperform the others with respect to learning form-function relationships from autoencoded latent vectors of 3D point clouds.

3.2.1 Radial Basis Function. The RBF model uses a linear combination of basis functions, one for each training point. Coefficients of the basis functions are computed during the training stage. The RBF model prediction equation is:

$$\mathbf{y} = p(\tilde{\mathbf{x}})\mathbf{w}_p + \sum_{i=1}^{d} \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}\mathbf{t_i}\mathbf{w}_r) \tag{5}$$

where $p(\tilde{\mathbf{x}})$ is the vector mapping polynomial coefficients to the prediction output. This polynomial is added to the basis functions to capture global trends, $\tilde{\mathbf{x}}$ is the input prediction vector, $\tilde{\mathbf{x}}\mathbf{t_i}$ is the ith training input vector, $\mathbf{w}_p$ is the vector of polynomial coefficients, and $\mathbf{w}_r$ is the vector of basis function coefficients.

The coefficients $\mathbf{w}_p$ and $\mathbf{w}_r$ can be computed by solving the following linear system:

$$\begin{bmatrix} \phi(\tilde{\mathbf{x}}\mathbf{t_1}, \tilde{\mathbf{x}}\mathbf{t_1}) & \dots & \phi(\tilde{\mathbf{x}}\mathbf{t_1}, \tilde{\mathbf{x}}\mathbf{t_d}) & p(\tilde{\mathbf{x}}\mathbf{t_1})^T \\ \vdots & \ddots & \vdots & \vdots \\ \phi(\tilde{\mathbf{x}}\mathbf{t_d}, \tilde{\mathbf{x}}\mathbf{t_1}) & \dots & \phi(\tilde{\mathbf{x}}\mathbf{t_d}, \tilde{\mathbf{x}}\mathbf{t_d}) & p(\tilde{\mathbf{x}}\mathbf{t_d})^T \\ p(\tilde{\mathbf{x}}\mathbf{t_1}) & \dots & p(\tilde{\mathbf{x}}\mathbf{t_d}) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}_{r1} \\ \vdots \\ \mathbf{w}_{rd} \\ \mathbf{w}_p \end{bmatrix} = \begin{bmatrix} yt_1 \\ \vdots \\ yt_d \\ 0 \end{bmatrix} \tag{6}$$

We use the Gaussian basis function:

$$\phi(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \exp(\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2) \tag{7}$$

3.2.2 Kriging With Partial Least Squares. As discussed in Sec. 2.2.2, we choose to employ KPLS because of its rapid convergence speed given our high-dimensional data [26]. Kriging considers $\mathbf{y} = F(x)$ a realization of the stochastic process:

$$Y(\tilde{\mathbf{x}}) = \sum_{j=1}^{m} \beta_j f_j(\tilde{\mathbf{x}}) + Z(\tilde{\mathbf{x}}) \tag{8}$$

where for $j = 1, \dots, m$, f_j is a known independent basis function, β_j is an unknown parameter, and Z is a zero-mean Gaussian random variable with a stationary covariance function k.

This stationary covariance function k (also called a covariance kernel) is given by:

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sigma^2 r_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}'} \forall \mathbf{x}, \quad \mathbf{x}' \in \mathbf{X} \tag{9}$$

where σ^2 is the process variance and $r_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}'}$ is the correlation function between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$.

Because the true covariance function is not known in practice, a common estimate is provided by the following Gaussian exponential kernel:

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sigma^2 \prod_{i=1}^{d} \exp(-\theta_i(\tilde{x}_i - \tilde{x}_i')^2) \tag{10}$$

where $\boldsymbol{\theta}$ is the vector of hyper-parameters and d is the dimension size of $\tilde{\mathbf{x}}$. Estimating these hyper-parameters in cases where d is large is prohibitively computationally expensive [26]. KPLS mitigates this problem by projecting input variables onto a new space formed by the principle components of the input variables. This leads to the following kernel for KPLS:

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sigma^2 \prod_{i=1}^{d} \prod_{l=1}^{h} \exp(-\theta_l(w_i^{(l)}\tilde{x}_i - w_i^{(l)}\tilde{x}_i')^2) \tag{11}$$

where h is the number of principal components and $\mathbf{w}_i$ is a vector of weights for each principle component for a given data sample. This leads to considerably fewer hyper-parameters than the standard Kriging case. The derivation of this kernel can be found in Ref. [26].

3.2.3 Neural Networks. This section details the two neural network (NN) architectures we implement for the surrogate model: a 1D ResNet architecture and a 1D VGG architecture. These two NN architectures are considered state-of-the-art for general function regression, and the depths of the networks are chosen in consideration of the small size of 3D model datasets compared with image datasets.

First, we define operations that are used in both NN architectures, the 1D convolution and batch normalization. 1D convolutions extract learned spatial features from the latent bottleneck representation of AtlasNet, and batch normalization acts as a regularizer and allows for stable training at higher learning rates [44].

$$Conv1d(N_i, C_{out_j}) = \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \bigoplus input(N_i, k) \tag{12}$$

where $\oplus$ is the cross-correlation operator, N is the batch size, C is the number of channels, and L is the number of input features.

$$BatchNorm(x) = \frac{x - E[x]}{\sqrt{Var[x] + \varepsilon}} \tag{13}$$

where ε is a small positive constant.

Next, we discuss the architectures of the 1D ResNet and 1D VGG neural network models.

1D ResNet: This 1D ResNet implementation is based on the popular 2D ResNet for image classification tasks [35], with modifications for this low data one-dimensional case such as modified filter lengths and shallower network architectures. Figure 3 shows the ResNet building block that is sequentially placed to create networks of varying depths. The left path consists of two sequential convolutions, and the right path is the "identity" connection that adds the unmodified (except for proper scaling) previous layer

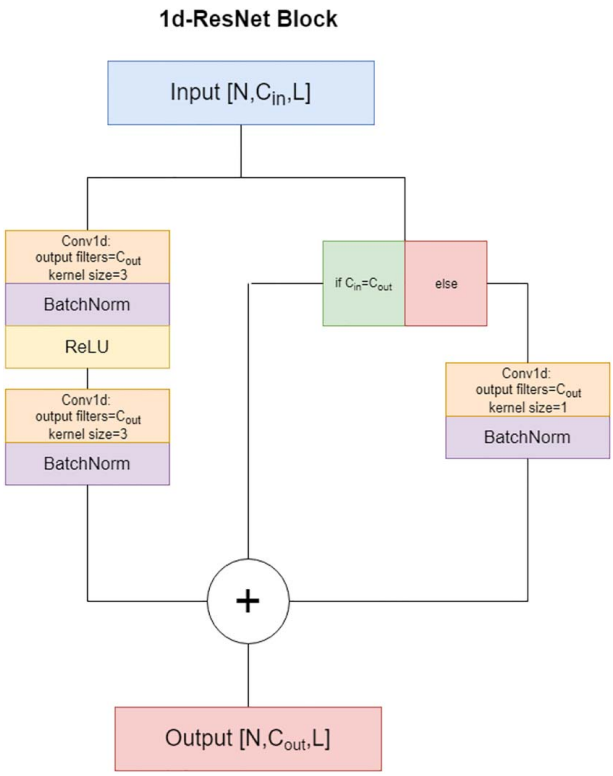

Fig. 3 The building block for our 1D ResNet, where *N* is batch size, C_{in} is the number of input filters, C_{out} is the number of output filters, and *L* is the number of input features

and improves the ability of neural networks to exploit increased depth even with less training data. Convolution kernel sizes in our 1D case are selected to match the 2D case; so, 3×3 2D convolutions become 1D convolutions of length 3.

He et al. [35] observed that the accuracy of ResNet increased with network depth, but with the caveat that a deeper network requires more training data. The image classification domain in which this approach was originally proposed is data-rich, and thus, the shallowest networks proposed in Ref. [35] is an 18-layer network. Given that our dataset is considerably smaller than image classification datasets, we extrapolate to a 12-layer case that reduces the depth of the network while maintaining the ResNet structure. Table 1 provides the details of each of this architecture.

1D VGG: This 1D VGG network is based on Simonyan and Zisserman's very deep convolutional neural network structure [34], once again with modifications for a small dataset and a one-dimensional input vector. Figure 4 shows the architecture of a VGG building block, which is a series of 1D convolutions followed by a max-pooling layer. The block size *B* indicates the number of consecutive 1D convolutions before max pooling occurs. Once again, convolutional kernel sizes are selected to match their 2D counterparts. For these networks, a pooling size of 4 is chosen for the first two layers to more quickly reduce the dimensionality of our filters and then subsequent layers use a pooling size of 2. These pooling sizes were chosen such that the flattened layer has a dimension of 4096, which matches the flattened layer size in the original 2D network.

Once again, a deeper network is observed to provide better accuracy given a large enough supporting dataset, and we once again extrapolate to a shallower seven-layer network that preserves the VGG structure, in consideration of our small dataset. Table 2 shows the details of this seven-layer architecture. Dropout regularization of 50% is employed between the first two dense layers to improve generalizability to the testing dataset as in the 2D analog.

Table 1 Architecture for each of the implemented 12-Layer 1D ResNet Network. The layer "res3" consists of two ResNet blocks of the same size in sequence, with a max pooling layer in between. The variable z in the final layer is the performance criteria vector. The length of the final layer should match the length of this vector.

Layer name	Number of filters	Output size	Sequential ResNet blocks in layer
conv1	64	256	
maxpool	64	128	
res1	64	128	1
res2	128	64	1
maxpool	128	32	–
res3*	256	16	2
res4	512	16	1
avgpool	512	1	–
dense	–	length(**z**)	

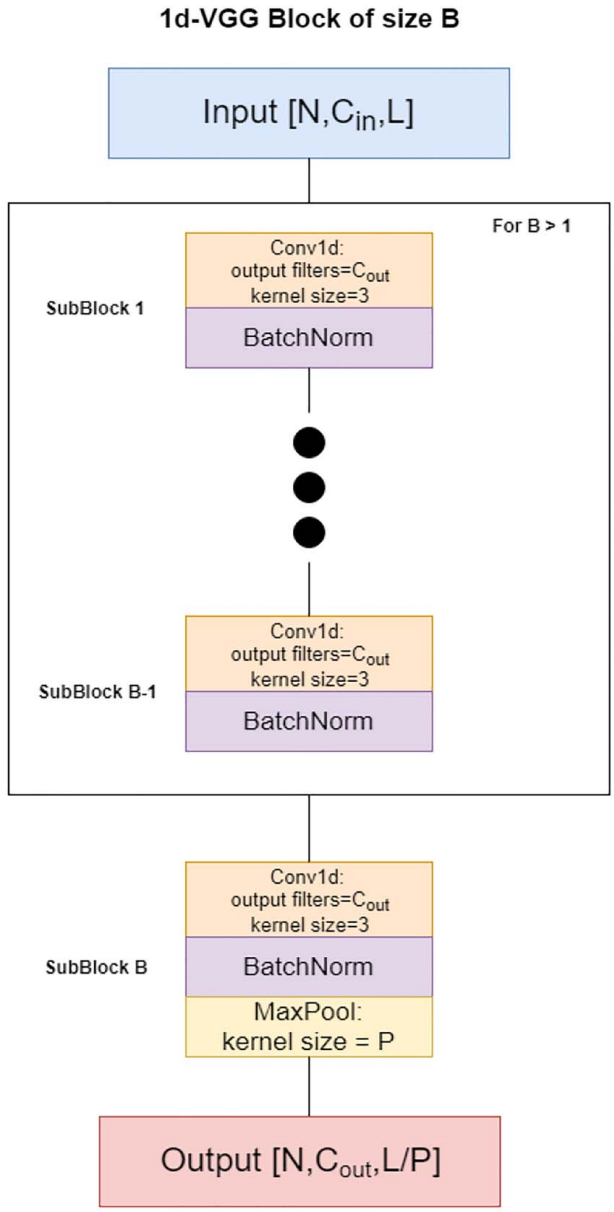

Fig. 4 The building block for our 1D VGG network, where *N* is the batch size, C_{in} is the number of input filters, C_{out} is the number of output filters, and *L* is the number of input

Table 2 Architecture for each of the implemented 7-Layer 1D VGG Network. The variable z in the final layer is the performance criteria vector. The length of the final layer should match the length of this vector.

Layer name	Number of filters	Output size	VGG block size
vgg1	64	256	1
vgg2	256	32	2
vgg3	512	8	1
flatten	–	4096	
dense1	–	1024	50% Dropout
dense2	–	256	50% Dropout
dense3	–	length(**z**)	

In Sec. 4, two case studies in which each of these surrogate modeling techniques is applied to predict the performance of 3D models of both aircraft and watercraft are presented. These two case studies will give us insight to be able to answer RQ1, whether these surrogate modeling techniques can accurately and efficiently predict design performance from a latent vector, which would enable the use of large-scale 3D datasets in engineering design contexts. The case studies will also address RQ2, by allowing us to identify which surrogate modeling technique(s) perform the best in terms of accuracy and efficiency.

4 Application

The surrogate modeling techniques described in Sec. 3 are applied to case studies using 3D point clouds of 1250 aircraft and 250 watercraft from the ShapeNet dataset. The objective is to use these surrogate modeling techniques to accurately predict the coefficients of lift and drag for an aerodynamic simulation in OPENFOAM for the aircraft test case, and the coefficient of drag only in a hydrodynamic simulation for the watercraft test case. Section 4.1 describes the ShapeNet Dataset, and Sec. 4.2 describes the aerodynamic simulation software OPENFOAM.

4.1 3D Model Dataset. The dataset used for this experiment was derived from the ShapeNet dataset [8]. Specifically, 1250 models from the aircraft category were sampled to create our aircraft dataset, and 250 models from the watercraft categories were sampled to create the watercraft dataset. The models were available in mesh format as well as in point cloud format. The mesh models are required for the OPENFOAM evaluation software, and the point clouds are used for the prediction model. Note that in the general case, any mesh model can be easily converted to a point cloud by stripping away the face connectivity information. The point clouds are normalized to each containing 2500 points. A pre-trained AtlasNet encoder made publicly available by Groueix et al. [13] was used to convert the 3D models to 1024 latent vectors. Figure 5 shows some of the objects in each category in mesh format.

One limitation of this dataset includes the relatively small size which is on the order of thousands of designs. This is small compared with many image datasets.

4.2 Model Evaluation Method. We use the open-source CFD software OPENFOAM [45,46]. OPENFOAM can perform simulations of basic CFD, combustion, turbulence modeling, electromagnetics, heat transfer, multiphase flow, and stress analysis [47]. OPENFOAM is based on the finite volume method, using C++ and object-oriented programming to develop a syntactical model of equation mimicking and scalar-vector-tensor operations [47]. OPENFOAM has been used in research for simulating coastal engineering processes [48], realistic wave generation and active wave absorption [49], boiling fluid flows [50], and many other applications [51–56].

For our investigation, we use OPENFOAM 5.0's *simplefoam* motorbike tutorial case for incompressible fluids using Reynolds-averaged

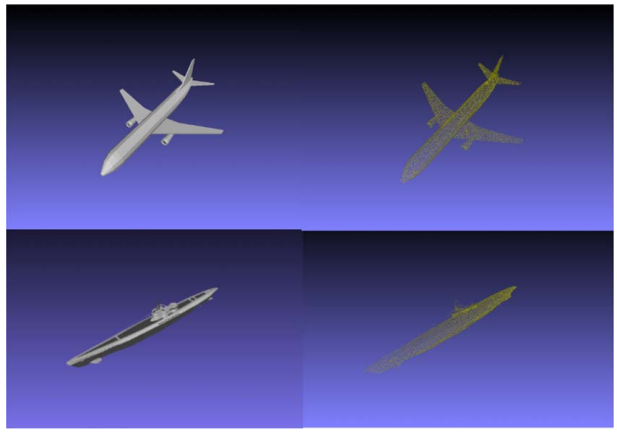

Fig. 5 Sample of mesh and corresponding point cloud models from the aircraft (top) and watercraft (bottom) datasets

simulation turbulence modeling. This tutorial solves for incompresisble fluid flow over a 3D object and calculates coefficients of lift and drag as well as pressures on the object surface and flow streamlines by solving a system of Navier–Stokes equations:

$$\nabla \cdot \mathbf{u} = 0 \tag{14}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \mathbf{v}\nabla^2\mathbf{u} - \nabla \mathbf{p} + \mathbf{g} \tag{15}$$

$$\mathbf{v} = \frac{\mu}{\rho} \tag{16}$$

where **u** is the velocity vector, **p** is the static pressure, **g** is the gravity force vector, **v** is the kinematic viscosity, μ is the dynamic viscosity, and ρ is the fluid density.

For the aircraft test case, we use the kinematic viscosity of air at 20 °C $v = 1.516e-5$ m^2/s. Likewise, for the watercraft test case, we use the kinematic viscosity of water at 20 °C $v = 1.004e-6$ m^2/s. We set the flow velocity to $u = 250$ m/s for the aircraft test case and $u = 17m/s$ for the watercraft test case. We use an angle of attack $\alpha = 0$ for all objects in both simulations. The inlet and outlet dimensions were 12×8, and the total size of the domain was a $12 \times 8 \times 20$ rectangular prism. For each test case, 200 simulation time steps were calculated, and the coefficient values at time step 199 were used for the coefficient ground truth data, after being scaled by a factor of 156.25.

5 Results and Discussion

In this section, we discuss the results of the OPENFOAM aerodynamic evaluation as well as the prediction model training and regression accuracy.

5.1 OPENFOAM Evaluation Results. Of the 1250 Shapenet models that were evaluated in OPENFOAM for the aircraft test case, only 64 of them were non-manifold objects, which returned 0 for both coefficients of lift and drag. Furthermore, the 15 models that had one (or both) coefficient's absolute value greater than 6 were excluded from training as extreme outliers. Figure 6 shows the histograms for the coefficients of lift and drag for the remaining 1170 models. The mean coefficient of drag was $\mu_d = 0.175$, and the standard deviation was $\sigma_d = 0.162$. For the coefficient of lift, the mean value was $\mu_l = 7.75 \times 10^{-3}$, and the standard deviation was $\sigma_l = 0.149$.

For the 250 models evaluated according to C_D in the watercraft test case, 23 were nonmanifold and 8 were outliers according to the aforementioned criteria. Figure 7 shows the histogram of the coefficient of drag for the remaining 214 models. The mean

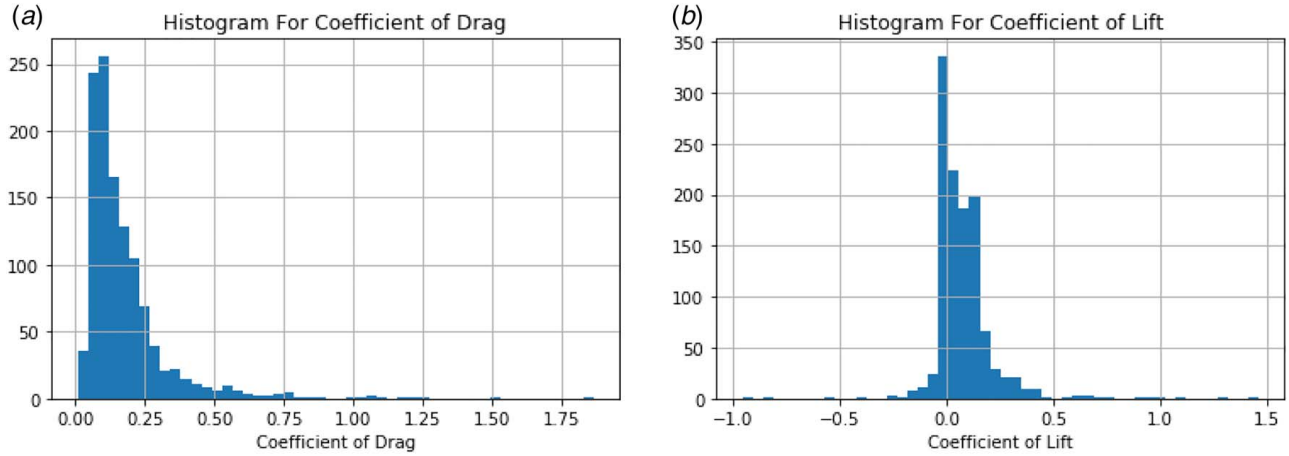

Fig. 6 Histograms of (*a*) coefficient of drag and (*b*) coefficient of lift for the aircraft dataset

coefficient of drag was $\mu_d = 0.746$, and the standard deviation was $\sigma_d = 0.750$. This large mean and standard deviation comes from the fact that the watercraft dataset consists of both submersible and non-submersible watercraft. The non-submersible watercraft models return a large coefficient of drag in an underwater flow simulation (Fig. 8).

5.2 Prediction Model Regression Results. In this section, we compare the convergence time and prediction accuracy of each of the surrogate modeling techniques discussed in Sec. 3.2 for the main test case involving 1170 aircraft models, as well as the smaller validation test case involving 171 watercraft models.

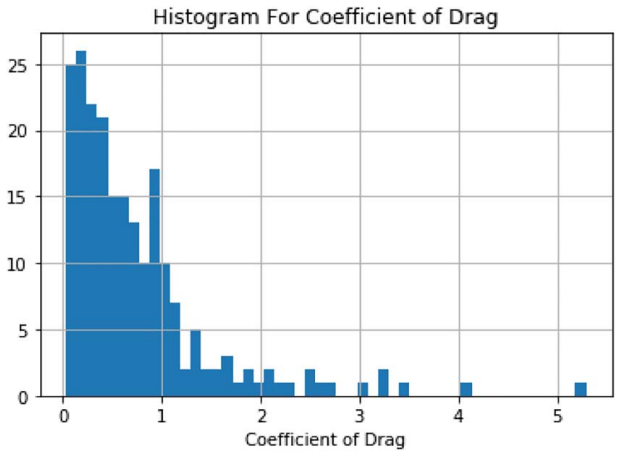

Fig. 7 Histogram of the coefficient of drag for the watercraft dataset

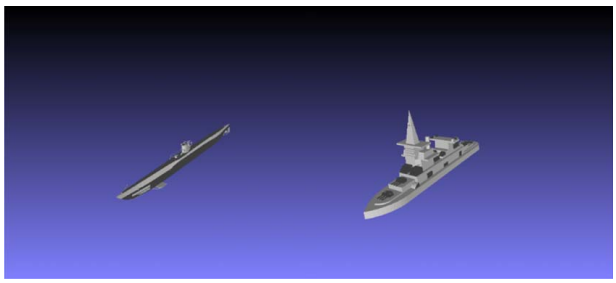

Fig. 8 Example of submersible watercraft in dataset with $C_D = 5.31 \times 10^{-3}$ (left) and non-submersible watercraft in dataset with $C_D = 0.6912$ (right)

Table 3 shows the convergence times for each method for the aircraft test case. All experiments were run using a GeForce GTX 1080 GPU and a 12-core Intel i7-8700 CPU. KPLS, and RBF both converge on a time scale of seconds, whereas the neural network approaches take minutes. Both neural network approaches were trained for 250 epochs. Once trained, all methods are able to evaluate the testing models in less than 10 s. By contrast, based on the average OPENFOAM model evaluation time of 7 min, evaluating the 234 testing models for the aircraft test case in OPENFOAM would have taken 43.05 h. Even for the neural network models, training the surrogate model and then using it to evaluate the remaining models reduces the evaluation time for the remaining models by a factor of 70 compared with evaluating them in OPENFOAM.

5.2.1 Aircraft Test Case. The 1170 samples were randomly split 80–20% into training and testing data, respectively. The prediction model was trained on the 936 training models and then evaluated for generalizability on the 234 testing models. Experiments with two mutually exclusive splits of training and testing data were conducted, and the average metrics of the two experiments recorded. MAE values are reported after being scaled back to the original coefficient value range.

In addition to MAE, we also examine the explained variance for the prediction model, which is defined as

$$ExplainedVar(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{Var[\mathbf{y}] - Var[\hat{\mathbf{y}}]}{Var[\mathbf{y}]} \tag{17}$$

where $\mathbf{y}$ is the true coefficient vector and $\hat{\mathbf{y}}$ is the predicted coefficient vector.

The best possible score is 1.0, and lower values indicate a less predictive model. Table 4 shows the MAE and explained variance for C_L and C_D in the testing dataset, as well as the p-value for statistical difference between the lowest error method and all others.

We see that only KPLS has a statistically significant larger error than the other three methods for C_L and that ResNet does not

Table 3 Convergence time for each method for the aircraft test case

	Convergence time
KPLS	12 s
RBF	3 s
ResNet	38 min
VGG	34 min

Table 4 Tables of MAE and explained variance

	MAE	MAE p-val	ExplainedVar
(a) Coefficient of lift			
RBF	2.06×10^{-3}	N/A	0.901
KPLS	5.32×10^{-3}	1.47×10^{-5}	0.657
ResNet	2.69×10^{-3}	0.888	0.812
VGG	3.06×10^{-3}	0.404	0.809
(b) Coefficient of drag			
RBF	1.88×10^{-3}	0.147	0.967
KPLS	1.97×10^{-3}	0.110	0.964
ResNet	1.54×10^{-3}	N/A	0.964
VGG	1.96×10^{-3}	0.249	0.961

Table 5 Statistical significance p-values for the differences in error between C_L and C_D for each method

	p-value
RBF	0.0429
KPLS	6.39×10^{-9}
ResNet	0.0207
VGG	0.0295

outperform the other methods by a statistically significant margin with respect to C_D error. Notably, in all cases, the surrogate model is more accurate with respect to C_D than the coefficient of C_L by a statistically significant margin as shown in Table 5. This is likely due to the fact that all C_D values are greater than zero, whereas the C_L values can be negative, and in fact, the C_L distribution has large outlier values in both the positive and negative directions. The fact that the neural network and KPLS methods have a more severe reduction in accuracy with respect to C_L suggests that they are comparatively less robust to outliers than the RBF model used in this study. We speculate that the input dimension reduction performed in these two methods limit the degree of overfitting to dense regions of the input space. This goes against the intuition that a neural network would be best suited to decode an object encoded by a neural network, as well as the perception that neural networks are the best at black box function approximation.

Given the fact that samples with large coefficient values are sparse, we would expect that any model would be considerably more accurate at predicting models with coefficient values near the center of the distribution than those at the extremes. However, it would still be useful to a designer if the networks were able to provide insight to how these models with outlier coefficients compare with the more common models. Therefore, in addition to evaluating the accuracy of the performance prediction network with respect to the absolute value, we are also interested in evaluating how well it is able to rank a set of designs. We again calculate the MAE, but now $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the true and predicted percentile, respectively, of the object coefficients with respect to the entire testing dataset. Table 6 summarizes the resulting errors.

With this analysis, we see that the neural network methods are superior in terms of ranking designs with respect to both coefficients, and they can achieve accuracy to within 7% of the true percentile. This suggests that while the neural network methods may not have been able to achieve as high regression accuracy on outlier data, they were still able to recognize when a data point was an outlier and rank it accordingly.

Next, we investigate the prediction performance of the surrogate models on outlier data. We separately analyze the accuracy for two outer-most and two inner quartiles of coefficients for the highest performing network architectures. This allows us to directly see the difference in how the model predicts outlier data and data close to the median. Table 7 shows the MAE in terms of both raw coefficient value and percentile for these categories.

The raw MAE values in Table 7 confirm that for the outer-most quartiles, the model has significantly lower precision with respect to the coefficient values than the inner-most quartiles, but the outlier accuracy is closer to the inlier accuracy in terms of percentile, especially with respect to C_D. The intuitive interpretation of this result is that the models are better at recognizing outliers than predicting the absolute value of their performance.

This is to be expected given the histograms in Fig. 6, which show that the sample size of models with coefficients of absolute value greater than 100 is small. Likewise, the inverse result in the inner quartiles can be explained by the large concentration of samples with coefficient values less than 0.33. The large number of samples indicates that the prediction models have learned to predict the absolute value of the coefficients better, but because these true values are much closer together than in the outer-most quartiles, the task of ranking them becomes more difficult. Overall, it appears that the improved ability to learn the relationships in the inner quartiles overcomes the increased difficulty of the ranking task, as the ranking accuracy in the inner quartiles is still superior.

Finally, we investigate the reason that the RBF and KPLS models were able to achieve comparable accuracy to the neural network methods despite being comparatively simple computationally. Inspecting the latent vectors produced by the autoencoder reveals that they are sparse, and on average, their ℓ_0 norm is 370. This means that, on average, 64% of the latent variables are zero. This suggests that the RBF and KPLS methods, which rely on projecting the input vectors to a lower-dimensional space, were able to compress the latent vectors without losing much information and efficiently solve the problem without sacrificing accuracy.

While this test case validates the performance of the surrogate models with respect to a certain category of geometries, we would like to see if they are robust to multiple types of geometries. In order to examine this, we investigate a second test case using a dataset of watercraft models.

Table 6 Table of MAE metric in terms of percentile ranking

	MAE (percentile)
(a) Coefficient of lift	
RBF	8.27%
KPLS	11.29%
ResNet	6.89%
VGG	7.69%
(b) Coefficient of drag	
RBF	7.43%
KPLS	7.45%
ResNet	5.67%
VGG	6.56%

Table 7 Outer-most and inner-most quartiles separately calculated MAE

	Outer	Outer %	Inner	Inner %
(a) Coefficient of lift				
RBF	0.131	22.13	2.06×10^{-3}	8.73
KPLS	0.171	24.69	2.27×10^{-3}	10.41
ResNet	0.135	20.81	1.74×10^{-3}	7.17
VGG	0.138	20.77	2.13×10^{-3}	7.95
(b) Coefficient of drag				
RBF	8.71×10^{-3}	14.36	1.68×10^{-3}	8.39
KPLS	8.77×10^{-3}	14.04	1.66×10^{-3}	8.60
ResNet	8.03×10^{-3}	13.36	1.36×10^{-3}	6.47
VGG	8.56×10^{-3}	13.76	1.79×10^{-3}	7.57

5.2.2 Watercraft Test Case. As with the aircraft test case, the 214 watercraft samples were randomly split 80–20% into training and testing data, respectively. This gives 171 training models and 43 testing models. Once again, two experiments were conducted, each with a different split of training and testing data.

Table 8 shows that, in this test case, all methods perform worse than the aircraft test case, especially the neural network methods. This raises the question of whether the performance suffers due to the different distribution of the latent vectors (geometry of the data) or due to differences in the distribution of the performance vectors, as the boat test case includes significantly more designs with $C_D > 1.5$ than the aircraft test case. An additional experiment was performed in which all designs with $C_D > 300$ were removed from the dataset, leaving 197 watercraft models. The results of this experiment are shown in Table 9.

We see that once these outliers have been removed, the MAEs are more in line with the values in the aircraft test case. ResNet achieves the best accuracy, same as in the aircraft C_D test case shown in Table 4(*b*).

We also see that the percentile accuracy is better than the aircraft test case regardless of whether additional extreme outliers are removed. Looking at the difference in distributions between Figs. 6(*a*) and 7, we see that the watercraft C_D values are more evenly distributed, as opposed to the aircraft test case in which most values are clustered between 0 and 0.33 with a small number of outliers. As we found in Sec. 5.2.1, a tight cluster of values presents difficulty in ranking, because the true values are very close together. On the other hand, ranking outliers is an easier task, but the results suffer from the fact that there are less training examples of outliers by definition. Both of these facts suggest that having data that is more spread out, but with sufficient samples for learning in each region of values would be most conducive to ranking accurately. The watercraft performance value distribution meets both of these criteria when compared with the aircraft distribution.

Modifying the distribution of the performance vectors to be more similar to the aircraft test case's distribution had a large impact on the regression accuracy, but changing the latent vector distribution to an entirely different category of geometries resulted in a similar performance. This suggests that the surrogate models are robust to different encoded geometry representations, but that the regression is sensitive to performance vector distributions with a large number of outliers.

With respect to RQ1, we found that all of the surrogate models investigated in this work are able to learn the form-function relationship between a geometry encoded in a 1D latent vector and a performance score, and they are robust to multiple categories of geometries. However, care must be taken to remove extreme outliers from the performance metric distribution, as we have shown this to have a severe negative impact on the accuracy of all of the surrogate models.

With respect to RQ2, we found that moving away from neural network approaches (i.e., RBF and KPLS) was far more efficient than using neural network models. However, none of the methods consistently outperformed the others in terms of accuracy. The relative accuracy of the surrogate models seems to be most heavily affected by the distribution of the performance values, and thus RBF and KPLS are both viable candidates for efficient surrogate models to learn form-function relationships from autoencoded latent vectors, enabling large-scale efficient evaluation of 3D model datasets.

Table 8 Tables of MAE, MAE *p*-values, and MAE percentile of C_D for the watercraft test case

	MAE	MAE *p*-val	MAE (percentile)
RBF	4.63×10^{-3}	0.0169	1.46%
KPLS	2.95×10^{-3}	N/A	1.51%
ResNet	0.107	2.35×10^{-23}	1.57%
VGG	6.02×10^{-3}	3.01×10^{-12}	1.78%

Table 9 Tables of MAE, MAE *p*-values, and MAE percentile of C_D for the reduced watercraft test case. This test case removed samples in which $C_D > 1.92$.

	MAE	MAE *p*-val	MAE (percentile)
RBF	2.47×10^{-3}	4.19×10^{-5}	1.00%
KPLS	1.55×10^{-3}	0.289	1.63%
ResNet	1.35×10^{-3}	N/A	0.938%
VGG	2.80×10^{-3}	9.09×10^{-6}	1.13%

6 Conclusions and Future Work

We investigated the use of four surrogate modeling techniques for estimating performance features of 3D models from a neural network autoencoded representation of a point cloud, namely RBF, KPLS, a shallow 1D ResNet architecture, and a shallow 1D VGG architecture. We show that departing from a neural network structure for decoding this representation into performance predictions is far more efficient than state-of-the-art neural network techniques, yet comparable in accuracy. In all experiments, the neural network methods converged on the timescale of minutes, and the non-neural network approaches converged on the timescale of seconds. In a test case involving 1170 aircraft models, the RBF surrogate model achieved an absolute performance prediction error of 1.97×10^{-3}, on average, and converged in 3 seconds, although all methods were within the margin of statistical significance in terms of accuracy. In a test case with 214 watercraft models, ResNet achieved the highest accuracy, and KPLS was within the margin of statistical significance after eliminating outliers with $C_D > 1.92$. We found that while the surrogate models were robust to encoded geometries of a different form (both aircraft and watercraft), they were sensitive to large numbers of extreme outliers with respect to regression accuracy.

However, for all methods, prediction accuracy in terms of the raw coefficient value is much better for models with coefficients close to the median of the dataset; however, the models are much more consistent at predicting where an object's coefficient values fall relative to other objects in the dataset and achieve percentile rank errors of less than 9%. In particular, the models are most accurate at identifying data with high coefficients of drag with respect to the rest of the dataset. In applications in which performance metrics must be gathered for a large amount of 3D models of varying quality, this prediction model can serve as an adequate filter to narrow down which models will be tested more rigorously, and depending on the tolerance for error of the application, it could replace the evaluation environment altogether. This surrogate modeling approach would be best suited for this purpose in cases where the models of interest are near the center of the coefficient distribution, and outliers can safely be eliminated. This surrogate modeling method allows for large 3D model datasets to be efficiently evaluated for performance, enabling their use in engineering design applications.

The neural network results have the potential to be greatly improved by introducing more data for training. While the non-neural network approaches had consistent performance across the two test cases, going from 214 training models to 936 training models greatly improved the performance of the neural network approaches. Nonetheless, using a very shallow architecture to manage the number of trainable weights, 1170 data points is very small compared with the amount of data available in applications where these deep neural network approaches are common, which is usually on the order of millions of data points. While datasets of this size for 3D point clouds are not readily available, an increase

in data would enable effective training of deeper network architectures. Given the promising performance of these networks on a comparably small dataset, it is reasonable to speculate that as large 3D model datasets become available, a surrogate model could serve as a substitute for an expensive performance evaluation software with a high degree of accuracy, and the long convergence time compared with the RBF and KPLS methods could prove worthwhile.

Another area that could be explored is the generalization of one surrogate model to another application. For example, the geometric features learned to predict aerodynamic coefficients will have some overlap with the geometric features needed to predict these coefficients in an underwater environment. Therefore, we would expect that some transfer learning would occur when using the aerodynamic prediction model as a starting point, which would reduce the data load required for the new application.

Acknowledgment

This research is funded in part by DARPA HR0011-18-2-0008 (Funder ID: 10.13039/100000185). Any opinions, findings, or conclusions found in this paper are those of the authors and do not necessarily reflect the views of the sponsors. The authors would like to acknowledge Dule Shu and Haoyuan Meng for their contributions to this work.

References

[1] Naboni, E., Zhang, Y., Maccarini, A., Hirsch, E., and Lezzi, D., 2013, "Extending the Use of Parametric Simulation in Practice Through a Cloud Based Online Service," IBPSA Italy-Conference of International Building Performance Simulation Association, Bozen, Italy, Jan. 30–Feb. 1.
[2] Rößger, P., and Richter, A., 2018, "Performance of Different Optimization Concepts for Reactive Flow Systems Based on Combined Cfd and Response Surface Methods," Comput. Chem. Eng., **108**, pp. 232–239.
[3] Wang, G. G., and Shan, S., 2007, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," ASME J. Mech. Des., **129**(4), pp. 370–380.
[4] Hardee, E., Chang, K.-H., Tu, J., Choi, K. K., Grindeanu, I., and Yu, X., 1999, "A CAD-Based Design Parameterization for Shape Optimization of Elastic Solids," Adv. Eng. Softw., **30**(3), pp. 185–199.
[5] Samareh, J. A., 1999, "A Survey of Shape Parameterization Techniques," CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics, Williamsburg, VA, June 22–25.
[6] Sripawadkul, V., Padulo, M., and Guenov, M., 2010, "A comparison of airfoil shape parameterization techniques for early design optimization," 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, p. 9050.
[7] Shahrokhi, A., and Jahangirian, A., 2007, "Airfoil Shape Parameterization for Optimum Navier–Stokes Design With Genetic Algorithm," Aerosp. Sci. Technol., **11**(6), pp. 443–450.
[8] Chang, A. X., Hanrahan, P., and Savva, M., 2015, "Semantically-Enriched 3D Models for Common-sense Knowledge," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, June 8–10.
[9] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., 2015, "3D Shapenets: A Deep Representation for Volumetric Shapes," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, June 7–12, pp. 1912–1920.
[10] Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R., and Papalambros, P. Y., 2016, "Estimating and Exploring the Product Form Design Space Using Deep Generative Models," Ann. Arbor, **1001**, p. 48109.
[11] Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L., 2017, "Representation Learning and Adversarial Generation of 3D Point Clouds," e-print arXiv:1707.02392.
[12] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., 2017, "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation," Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, July 21–26, IEEE, New York, Vol. 1(2), p. 4.
[13] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M., 2018, "Atlasnet: A Papier-Mâché Approach to Learning 3D Surface Generation," e-print arXiv:1802.05384.
[14] Nash, C., and Williams, C. K., 2017, "The Shape Variational Autoencoder: A Deep Generative Model of Part-Segmented 3D Objects," Computer Graphics Forum, **36**(5), pp. 1–12.
[15] Maturana, D., and Scherer, S., 2015, "Voxnet: A 3D Convolutional Neural Network for Real-time Object Recognition," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, Sept. 28–Oct. 2, IEEE, New York, pp. 922–928.
[16] Riegler, G., Osman Ulusoy, A., and Geiger, A., 2017, "Octnet: Learning Deep 3D Representations at High Resolutions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, July 21–26, pp. 3577–3586.
[17] Wang, D. Z., and Posner, I., 2015, "Voting for Voting in Online Point Cloud Object Detection," Robotics: Science and Systems, **1**(3), pp. 10–15607.
[18] Fang, Y., Xie, J., Dai, G., Wang, M., Zhu, F., Xu, T., and Wong, E., 2015, "3D Deep Shape Descriptor," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, June 8–10, pp. 2319–2328.
[19] Mullur, A. A., and Messac, A., 2005, "Extended Radial Basis Functions: More Flexible and Effective Metamodeling," AIAA J., **43**(6), pp. 1306–1315.
[20] Jin, R., Chen, W., and Simpson, T. W., 2001, "Comparative Studies of Metamodelling Techniques Under Multiple Modelling Criteria," Struct. Multidisciplinary Optim., **23**(1), pp. 1–13.
[21] Ellbrant, L., Eriksson, L.-E., and Mårtensson, H., 2012, "CFD Optimization of a Transonic Compressor Using Multiobjective Ga and Metamodels," Proceedings of the 28th International Congress of the Aeronautical Sciences, Brisbane, Australia, Sept. 23–28, pp. 23–28.
[22] Sacks, J., Schiller, S. B., and Welch, W. J., 1989, "Designs for Computer Experiments," Technometrics, **31**(1), pp. 41–47.
[23] Kleijnen, J. P., 2009, "Kriging Metamodeling in Simulation: A Review," Eur. J. Oper. Res., **192**(3), pp. 707–716.
[24] Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F., 2001, "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization," AIAA J., **39**(12), pp. 2233–2241.
[25] Jia, G., and Taflanidis, A. A., 2013, "Kriging Metamodeling for Approximation of High-Dimensional Wave and Surge Responses in Real-Time Storm/hurricane Risk Assessment," Comput. Methods Appl. Mech. Eng., **261**, pp. 24–38.
[26] Bouhlel, M. A., Bartoli, N., Otsmane, A., and Morlier, J., 2016, "Improving Kriging Surrogates of High-Dimensional Design Models by Partial Least Squares Dimension Reduction," Struct. Multidisciplinary Optim., **53**(5), pp. 935–952.
[27] Wold, H., 1975, "Soft Modelling by Latent Variables: the Non-linear Iterative Partial Least Squares (nipals) Approach," J. Appl. Probab., **12**(S1), pp. 117–142.
[28] Caixeta, P., Jr., and Marques, F., 2010, "Neural Network Metamodel-based MDO for Wing Design Considering Aeroelastic Constraints," 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 18th AIAA/ASME/AHS Adaptive Structures Conference 12th, Orlando, FL, Apr. 12–15, p. 2762.
[29] Ferreiro-Cabello, J., Fraile-Garcia, E., de Pison Ascacibar, E. M., and de Pison Ascacibar, F. J. M., 2018, "Metamodel-Based Design Optimization of Structural One-Way Slabs Based on Deep Learning Neural Networks to Reduce Environmental Impact," Eng. Struct., **155**, pp. 91–101.
[30] Sreekanth, J., and Datta, B., 2010, "Multi-Objective Management of Saltwater Intrusion in Coastal Aquifers Using Genetic Programming and Modular Neural Network Based Surrogate Models," J. Hydrol., **393**(3–4), pp. 245–256.
[31] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, "Imagenet Classification With Deep Convolutional Neural Networks," Twenty-sixth Conference on Neural Information Processing Systems, Lake Tahoe, NV, Dec. 3–8, pp. 1097–1105.
[32] Howard, A. G., 2013, "Some Improvements on Deep Convolutional Neural Network Based Image Classification," International Conference on Learning Representations 2013, Scottsdale, AZ, May 2–4.
[33] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S., 2017, "Dermatologist-Level Classification of Skin Cancer With Deep Neural Networks," Nature, **542**(7639), p. 115.
[34] Simonyan, K., and Zisserman, A., 2014, "Very Deep Convolutional Networks for Large-scale Image Recognition," e-print arXiv:1409.1556.
[35] He, K., Zhang, X., Ren, S., and Sun, J., 2016, "Deep Residual Learning for Image Recognition," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, June 26–July 1, pp. 770–778.
[36] Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H., 2015, "A Regression Approach to Speech Enhancement Based on Deep Neural Networks," IEEE/ACM Trans. Audio Speech Language Process., **23**(1), pp. 7–19.
[37] Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H., 2014, "An Experimental Study on Speech Enhancement Based on Deep Neural Networks," IEEE Signal Process. Lett., **21**(1), pp. 65–68.
[38] Toshev, A., and Szegedy, C., 2014, "Deeppose: Human Pose Estimation via Deep Neural Networks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, June 24–27, pp. 1653–1660.
[39] Held, D., Thrun, S., and Savarese, S., 2016, "Learning to Track at 100 fps With Deep Regression Networks," European Conference on Computer Vision, Amsterdam, Netherlands, Oct. 8–16, pp. 749–765.
[40] Kossaifi, J., Lipton, Z. C., Khanna, A., Furlanello, T., and Anandkumar, A., 2017, "Tensor Regression Networks," e-print arXiv:1707.08308, 2(4).
[41] Solovyev, R. A., Vakhrushev, M., Radionov, A., Aliev, V., and Shvets, A. A., 2018, "Deep Learning Approaches for Understanding Simple Speech Commands," e-print arXiv:1810.02364.
[42] Hurtado, D. M., Uziela, K., and Elofsson, A., 2018, "Deep Transfer Learning in the Assessment of the Quality of Protein Models," e-print arXiv:1804.06281.
[43] Bouhlel, M. A., 2018, "Surrogate Modeling Toolbox," https://github.com/SMTorg/smt.
[44] Ioffe, S., and Szegedy, C., 2015, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," International Conference on Machine Learning, Lille, France, July 6–July 11, pp. 448–456.
[45] Jasak, H., Jemcov, A., and Tukovic, Z., 2007, "Openfoam: A C++ Library for Complex Physics Simulations," International Workshop on Coupled Methods in Numerical Dynamics, Dubrovnik, Croatia, Sept. 19–21, pp. 1–20.
[46] Jasak, H., 2009, "Openfoam: Open Source CFD in Research and Industry," Int. J. Naval Arch. Ocean Eng., **1**(2), pp. 89–94.

[47] Chen, G., Xiong, Q., Morris, P. J., Paterson, E. G., Sergeev, A., and Wang, Y., 2014, "Openfoam for Computational Fluid Dynamics," Not. AMS, **61**(4), pp. 354–363.
[48] Higuera, P., Lara, J. L., and Losada, I. J., 2013, "Simulating Coastal Engineering Processes With Openfoam®," Coastal Eng., **71**, pp. 119–134.
[49] Higuera, P., Lara, J. L., and Losada, I. J., 2013, "Realistic Wave Generation and Active Wave Absorption for Navier–Stokes Models: Application to Openfoam®," Coastal Eng., **71**, pp. 102–118.
[50] Kunkelmann, C., and Stephan, P., 2009, "CFD Simulation of Boiling Flows Using the Volume-of-Fluid Method Within Openfoam," Numer. Heat Transfer, Part A: Appl., **56**(8), pp. 631–646.
[51] Petit, O., Page, M., Beaudoin, M., and Nilsson, H., 2009, "The Ercoftac Centrifugal Pump Openfoam Case-study," Proceedings of the Third IAHR International Meeting of the Workgroup on Cavitation and Dynamic Problem in Hydraulic Machinery and Systems, Brno, Czech Republic, Oct. 14–16.
[52] Lysenko, D. A., Ertesvåg, I. S., and Rian, K. E., 2013, "Modeling of Turbulent Separated Flows Using Openfoam," Comput. Fluids, **80**, pp. 408–422.
[53] Kassem, H. I., Saqr, K. M., Aly, H. S., Sies, M. M., and Wahid, M. A., 2011, "Implementation of the Eddy Dissipation Model of Turbulent Non-Premixed Combustion in Openfoam," Int. Commun. Heat Mass Transfer, **38**(3), pp. 363–367.
[54] Lysenko, D. A., Ertesvåg, I. S., and Rian, K. E., 2012, "Large-Eddy Simulation of the Flow Over a Circular Cylinder At Reynolds Number 3900 Using the Openfoam Toolbox," Flow Turbulence Combust., **89**(4), pp. 491–518.
[55] Iturrioz, A., Guanche, R., Lara, J., Vidal, C., and Losada, I., 2015, "Validation of Openfoam® for Oscillating Water Column Three-Dimensional Modeling," Ocean Eng., **107**, pp. 222–236.
[56] Silva, L. F. L., and Lage, P. L., 2011, "Development and Implementation of a Polydispersed Multiphase Flow Model in Openfoam," Comput. Chem. Eng., **35**(12), pp. 2653–2666.