



PAPER • OPEN ACCESS

Graph convolutional networks applied to unstructured flow field data

To cite this article: Francis Ogoke *et al* 2021 *Mach. Learn.: Sci. Technol.* **2** 045020

View the [article online](#) for updates and enhancements.

You may also like

- [Enhancing Galaxy Classification with U-Net Variational Autoencoders for Image Denoising](#)
S. S. Mirzoyan
- [Toward Characterizing Dark Matter Subhalo Perturbations in Stellar Streams with Graph Neural Networks](#)
Peter Xiangyuan Ma, Keir K. Rogers, Ting S. Li et al.
- [Transferring predictions of formation energy across lattices of increasing size](#)
Massimiliano Lupo Pasini, Mariia Karabin and Markus Eisenbach



PAPER

OPEN ACCESS

RECEIVED
20 June 2021REVISED
16 August 2021ACCEPTED FOR PUBLICATION
20 August 2021PUBLISHED
8 September 2021

Original Content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.



Graph convolutional networks applied to unstructured flow field data

Francis Ogoke¹ , Kazem Meidani¹ , Amirreza Hashemi² and Amir Barati Farimani^{1,3,4,*} ¹ Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, United States of America² Department of Computational Modeling and Simulation, University of Pittsburgh, Pittsburgh, PA, United States of America³ Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, United States of America⁴ Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, United States of America

* Author to whom any correspondence should be addressed.

E-mail: barati@cmu.edu**Keywords:** data-driven prediction, deep learning, unstructured data, flow field measurements, graph neural network

Abstract

Many scientific and engineering processes produce spatially unstructured data. However, most data-driven models require a **feature matrix** that enforces both a **set number** and **order of features** for each sample. They thus cannot be easily constructed for an unstructured dataset. Therefore, a graph based data-driven model to perform inference on fields defined on an unstructured mesh, using a graph convolutional neural network (GCNN) is presented. **The ability of the method to predict global properties from spatially irregular measurements with high accuracy is demonstrated by predicting the drag force associated with laminar flow around airfoils from scattered velocity measurements.** The network can infer from field samples at different resolutions, and is invariant to the order in which the measurements within each sample are presented. The GCNN method, using inductive convolutional layers and adaptive pooling, is able to predict this quantity with a validation R^2 above 0.98, and a Normalized Mean Squared Error below 0.01, without relying on spatial structure.

1. Introduction

Due to recent advances in data-driven methods and the **proliferation** of scientific data, there has been a significant amount of attention toward data-driven inference to model or predict system properties. This is particularly relevant in **fluid mechanics**, where large amounts of data are needed to understand **potentially complex, multiscale flow phenomena**. The success of deep learning (DL) in computer vision has inspired its application in studying physical phenomena. **Physics informed Neural Networks** are used to learn the physics behind and solutions to high-dimensional partial differential equations (PDEs) [1–5]. Deep neural networks have been of particularly high interest in **surrogate modeling** and predicting complex transport phenomena [6, 7]. Wiewel *et al* used a **latent space learning** to efficiently simulate the temporal evolution of the pressure field [8]. Farimani *et al* applied **conditional generative adversarial networks** to solve the physics of transport phenomena without knowledge of the governing equations [9].

Machine learning has seen success in generating flow fields based on data collected from experiments, and noisy data from numerical simulations [10–13]. For example, **particle image velocimetry (PIV)**, where velocity fields are generated by tracking the movement of tracer particles, has been introduced as a non-intrusive technique for analyzing flow behavior and measuring the forces interacting with an immersed object [14, 15]. The analysis of PIV data using Machine Learning has allowed for more efficient flow field reconstruction and prediction. Rabault *et al* used a convolutional neural network (CNN) architecture to surrogate PIV by cross-correlating point locations between two frames, therefore predicting the flow velocity field [16]. Morimoto *et al* applied a CNN to artificial PIV data to develop a method for reconstructing flow field from snapshots with missing regions [17].

Various machine learning algorithms have recently been applied to structured flow field data to facilitate predictions based on immersed flows around streamlined objects, such as airfoils. For example, the leading-edge suction parameter (LESP) and angle of attack (AoA) are of high importance for discrete vortex methods to be effective. Hou *et al* used a combination of CNNs and recurrent neural networks to predict these parameters based on time-dependent surface pressure measurements [18]. In related work, Provost *et al* used the same method to optimize the number of sensors necessary for LESP and AoA prediction [19].

Zhang *et al* applied CNNs on image representations of various airfoils and their surrounding flow to predict the lift coefficient. The airfoils are immersed in different flow conditions, and the parameters of the flow (e.g. Mach Number) are encoded as pixel intensities [20]. Viquerat and Hachem used an optimized CNN to estimate the drag coefficient of several arbitrary 2D geometries in laminar flow. A large training sample of random 2D shapes along with their drag forces computed by immersed mesh method were used to increase the prediction accuracy of realistic geometries such as NACA airfoils [21]. Yilmaz and German used a CNN to predict airfoil performance directly from the geometry of the airfoil, replacing cumbersome surrogate methods that required manual parameterizations of the airfoil using shape functions [22].

Guo *et al* trained a deep CNN to make fast but less accurate visual approximations of the steady state flow around 2D objects which improves the design process by expediting the alternatives generation [23]. Miyanawala and Jaiman used a CNN to predict aerodynamic coefficients for several bluff body shapes at low Reynolds numbers. They used structured data of an encoded distance function to predict unsteady fluid forces [24]. Bhatnagar *et al* also used a signed distance function as well as a limited range of both Reynolds numbers and angles of attack to predict flow field velocities and pressure for several airfoils [25].

These methods, however, are limited in their ability to generalize to unstructured data. As traditional machine learning methods require the creation of a feature matrix with both a specific size and order of input samples, they cannot be applied to unstructured data. Flow field data, however, can be highly unstructured due to the use of irregular meshes to define curved or complex geometries.

Recent interest in manipulating unstructured data has led to the development of both mesh-free inference methods for point cloud representations and reduced-order models based on graph-based representations of fluid data [26]. Several works have used graph theory-based methods to identify coherent structures within turbulent flow [27, 28]. Hadjighasem *et al* examined the generalized eigenvalue problem of the graph Laplacian to develop a heuristic for determining the locations of coherent structures [29]. This work is extended to extract coherent structures from the vortical behavior of the flow by Meena *et al*, where a graph is constructed to represent the mutual interaction of individual vortex elements, and larger vortex communities are identified with network theory-based community detection algorithms [30]. To perform mesh-free inference, Trask *et al* introduce the idea of GMLS-Nets. GMLS-Nets parameterize the generalized moving least-squares functional regression technique for application on mesh-free, unstructured data. They abstract the GMLS operator to perform convolution on point clouds. They demonstrated success in both uncovering operators governing the dynamics of PDEs and predicting body forces associated with flow around a cylinder based on point measurements [31].

In this paper, we present a method for data-driven prediction from flow fields defined on irregular and unstructured meshes, using a graph convolutional neural network (GCNN) framework. GCNNs have been applied to problems dealing with unordered data points where specific relationships between the points encode important information and thus are often used in applications such as natural language processing, traffic forecasting, and material property prediction [32–39]. Graph Neural Networks have also previously been used to model spatiotemporal phenomena and surrogate physics simulators [40, 41]. Our approach exploits the irregular mesh that the velocity field is defined on to regress from unordered data across both varying resolutions and non-uniform spatial distributions.

In section 2, we first present our methodology of using graph representation of the unstructured mesh. The learning algorithm based on an inductive graph convolution method is then discussed and finally, the structure of the implemented network is explained. In section 3, The data which is used in our study and the results for the corresponding experiments are discussed and compared to other traditional methods. A conclusion of the work and suggestions for some possible future directions are also provided in section 4.

2. Methods

2.1. Graph representation

The key idea behind our approach is to use a graph representation to describe the connectivity of unstructured data points. In order to resolve flow fields around complex geometries in computational fluid dynamics, an irregular mesh around the immersed object is created. Next, numerical methods are applied to calculate the flow field data. Any mesh structure around the immersed object in the flow field can be represented as a graph, considering mesh nodes as vertices and using edges to connect the neighbors.

Therefore, one can construct a graph representation of the unstructured mesh with different complexities or number of nodes around arbitrary objects.

Specifically, we define an undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, with \mathbf{V} vertices describing the nodes of the mesh, and \mathbf{E} edges representing the connectivity of the mesh. The flow field data is defined on mesh nodes, resulting in a feature matrix that contains the input features for V graph vertices. The edge connections are encoded as the adjacency matrix, a binary $V \times V$ matrix indicating whether any given pair of vertices are connected.

2.2. Graph convolution

GCNNs are the generalization of CNNs for operation on graphs. GCNNs, like CNNs, are able to extract multi-scale spatial features through the use of shared weights and localized filters [42]. However, as discussed earlier, traditional CNNs are unable to work with unstructured data. GCNNs can bypass this limitation by defining the convolution operation based on the structure of the graph. By propagating information through each node's local neighborhood as defined by the adjacency matrix, GCNNs are invariant towards the order in which the nodes are specified in the feature matrix. GCNNs are often used for tasks such as node classification, link prediction [43], and graph classification.

GCNNs can be described in terms of a general framework for learning on graph-structured data, called message passing neural networks (MPNNs) [44]. MPNNs develop hidden state embeddings, h_v for each node v during the training process. Supervised training of a graph neural network aims to learn a state embedding from the features defined on the nodes and edges of the graph to have the best possible mapping to the output. The training process consists of two phases, the message passing phase where hidden states aggregate information from their surrounding nodes, and the readout phase where a feature vector for the graph is computed from the hidden states. The message passing process is parameterized by two functions, the message function M_t and the node update function U_t , while the readout function is given by R . R , M_t , and U_t are all learned differentiable functions that are updated during the training process.

Defining e_{vw} as the edge connecting node v to node w and $\mathcal{N}(v)$ as the neighborhood of node v , the message passing phase can be formalized as:

$$m_v^{t+1} = \sum_{w \in \mathcal{N}(v)} M_t(h_v^t, h_w^t, e_{vw}) \quad (1)$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}). \quad (2)$$

Based on h_v , the readout phase computes a feature vector as:

$$\hat{y} = R(h_v^T | v \in G). \quad (3)$$

A standard framework used is the Laplacian based GCNN, detailed in [42], where M_t and U_t are defined as the following:

$$M_t(h_v^t, h_w^t) = \tilde{A}_{vw} (\deg(v) \deg(w))^{-\frac{1}{2}} h_w^t \quad (4)$$

$$U_t(h_v^t, m_v^{t+1}) = \sigma((W^t)^T m^{(t+1)}) \quad (5)$$

where \tilde{A}_{vw} is the adjacency matrix describing the connectivity of the graph, assuming that each node is connected to itself, σ is a nonlinear activation function, and $\deg(v)$ is the number of nodes connected to node v [44].

The GraphSAGE method, introduced by [45] is in close relation to the Laplacian based GCN layers described in the message passing formalization above. In the version of GraphSAGE implemented in this paper, the GraphSAGE algorithm is the inductive variant of this message passing network, with specific modifications to improve the accuracy and efficiency of the model. GraphSAGE acts in an inductive manner, operating on each node rather than on the entire graph, as it uses each node's local neighborhood in order to learn a function that can generate appropriate node embeddings. This method first samples a fixed number of nodes from the k nearest neighbors of each node and then applies an aggregation operator to transfer information to the node itself (algorithm 1). The aggregator can be a weighted averaging operation with trainable parameters. This inductive framework is especially helpful in the case of large graphs where low-dimensional embeddings of the nodes are more important [45]. In this regression application, the readout phase consists of a fully connected neural network that predicts a single global value for each graph based on the hidden state embeddings.

Algorithm 1. GraphSAGE embedding generation algorithm, reproduced from [45].

Input: Graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$; input features $\{x_v, \forall v \in \mathcal{V}\}$; depth K ; weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; neighborhood function $\mathcal{N}(v) = \{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}$
Output: Vector representations z_v for all $v \in \mathcal{V}$

```

1  $h_v^0 \leftarrow x_v, \forall v \in \mathcal{V}$ 
2 for  $k = 1 \dots K$ 
3   for  $v \in \mathcal{V}$ 
4      $h_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{MEAN}(\{h_u^{k-1}\} \cup \{h_v^{k-1}, \forall u \in \mathcal{N}(v)\}))$ 
5    $h_v^k \leftarrow h_v^k / \|h_v^k\|_2, \forall v \in \mathcal{V}$ 
6  $z_v \leftarrow h_v^K, \forall v \in \mathcal{V}$ 

```

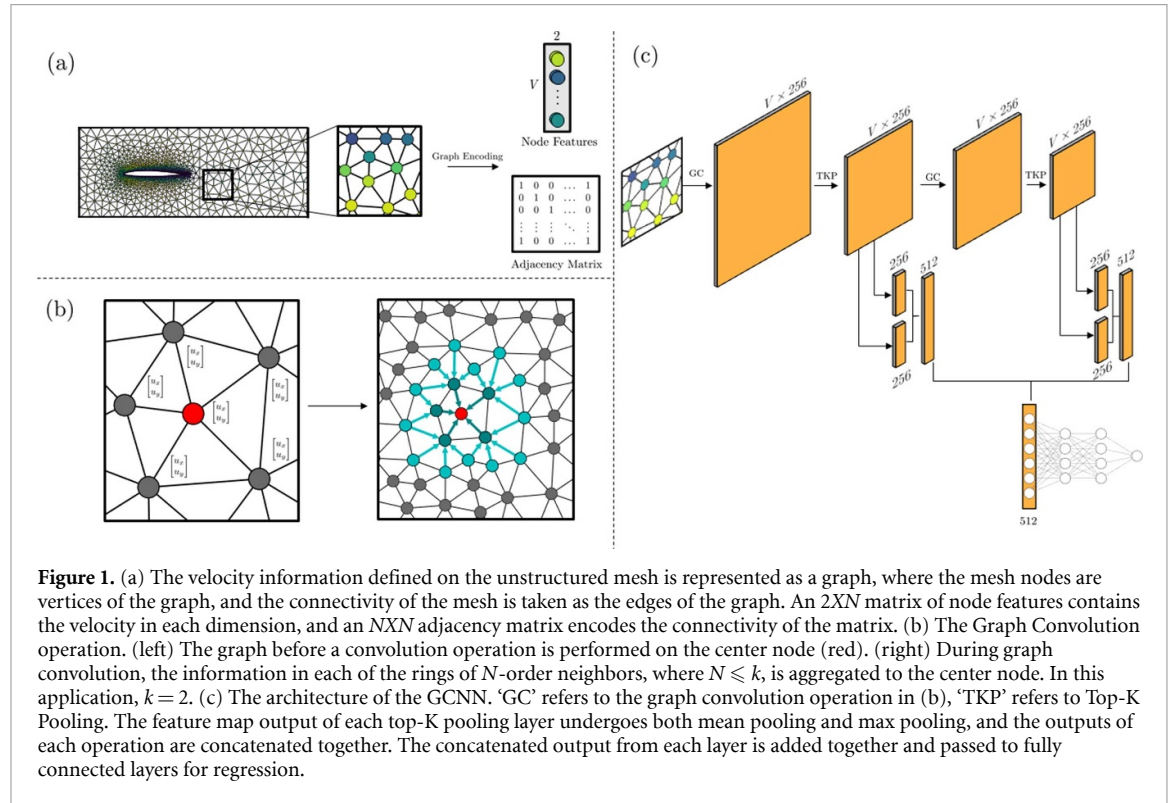


Figure 1. (a) The velocity information defined on the unstructured mesh is represented as a graph, where the mesh nodes are vertices of the graph, and the connectivity of the mesh is taken as the edges of the graph. An $2 \times N$ matrix of node features contains the velocity in each dimension, and an $N \times N$ adjacency matrix encodes the connectivity of the matrix. (b) The Graph Convolution operation. (left) The graph before a convolution operation is performed on the center node (red). (right) During graph convolution, the information in each of the rings of N -order neighbors, where $N \leq k$, is aggregated to the center node. In this application, $k = 2$. (c) The architecture of the GCNN. ‘GC’ refers to the graph convolution operation in (b), ‘TKP’ refers to Top-K Pooling. The feature map output of each top-K pooling layer undergoes both mean pooling and max pooling, and the outputs of each operation are concatenated together. The concatenated output from each layer is added together and passed to fully connected layers for regression.

Flow field meshing usually results in a relatively large graph around the objects in comparison to other common applications such as molecular graphs. Hence, we use a node level embedding graph convolution operator that is based on the average aggregator in the GraphSAGE framework. Since the mesh has a specific edge connection pattern in which each node is only connected to a few neighbors, the sampling operator is not used. In our convolution operation, the features in the k nearest rings in the neighborhood of each node are transferred to the center node by a trainable aggregation operation (figure 1(b), line 4 in algorithm 1). Here, we use the Pytorch Geometric [46] library to load the data and implement the graph convolutional layers and pooling.

2.3. Network architecture

For this problem, we implement a GCNN using GraphSAGE convolutional layers and top-K pooling steps, similar to the approach described in [47]. Different from [47], we use an inductive convolution as opposed to a transductive convolution. Inductive methods are capable of generalizing to graphs with different structures, here allowing for prediction on meshes with varying resolutions. The inputs of the network are graphs with node level velocity features, while the output is the value of the predicted drag force. Specifically, we use two GraphSAGE layers, each followed with a top K pooling layer. Top K pooling is a downsampling method to reduce the size of the layers by selecting the most important features. In top K pooling layer, a learned score is assigned to each node, and the nodes with the K highest scores are selected to be passed to the next layer [48]. The output of each pooling layer is pooled twice, once using global mean pooling and then using global max

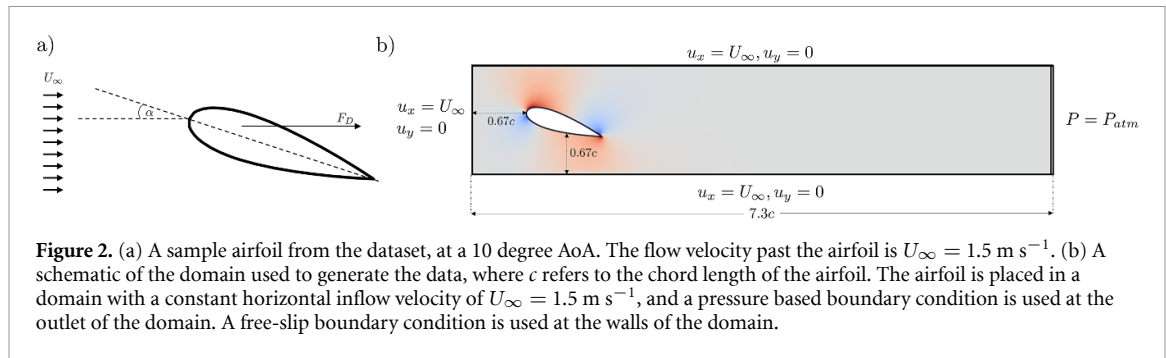
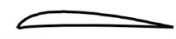



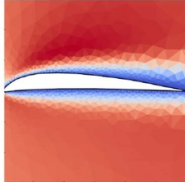
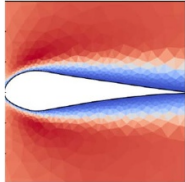
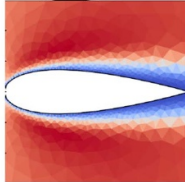
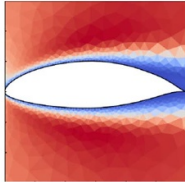


Table 1. The velocity field and drag force for four different airfoil samples from the dataset.

Airfoil Geometry				
Velocity Field				
F_D	1.478	1.983	2.274	2.462

pooling, and the output from each pooling operation are concatenated together. While the input size to the network can vary in different samples as they have various number of nodes, the output size should be the same. By pooling along the dimension of vertices, the global pooling operations result in the same output size. The pooled, concatenated vectors are added together as a ‘skip connection’, to reinforce the information contained in the sparse convolved feature maps. The output from this step is then fed to a fully connected network with three hidden layers, that predicts the drag force (figure 1(c)). The training details of the GCNN are provided in table A1 and the parameters are defined in [46, 49] for interested readers. The effect of some of the hyperparameters and their selection are also examined in table A3, and figure 1.

3. Results

3.1. Data

In order to test the performance of our method, we aim to predict the drag force on the airfoils directly from the unstructured flow field velocity. To generate the airfoils, coordinate files are extracted from the UIUC airfoil database which contains the cartesian coordinates outlining the shape of the airfoil [50]. The incomplete or non-meshable samples are then removed from the dataset. In addition, the geometries are normalized to have a unit chord length. Next, each airfoil coordinate file is imported to the open-source mesh generator GMSH [51]. Meshes are created to reflect the variation in the density of information contained in the domain, with a finer mesh on the area close to the airfoil that resolves the complex boundary layer effects, and a coarser mesh further from the airfoil, where the flow is minimally affected by the presence of the airfoil.

To compute the velocities at mesh nodes around each object and the corresponding drag force, we perform CFD simulations using the FEniCS [52] package. FEniCS supports the DOLFIN PDE solver, which is used to solve the incompressible Navier–Stokes equations with an incremental pressure correction scheme (IPCS) method [53]. The boundary conditions for these CFD simulations are a uniform velocity input of 1.5 m s^{-1} at the inlet (left), a far-field pressure condition at the outlet (right) and slip conditions at the top and bottom interfaces (figure 2). The viscosity and the density of the flow are 0.001 Pa s and 1 kg m^{-3} respectively. Selected airfoil samples with their velocity magnitude field along with their drag force are provided in table 1. There is a positive correlation between the thickness of the airfoil and the corresponding drag force.

The drag on an airfoil A is calculated as follows:

$$F_D = \int_A (\sigma \cdot n) \cdot e_x dS \quad (6)$$

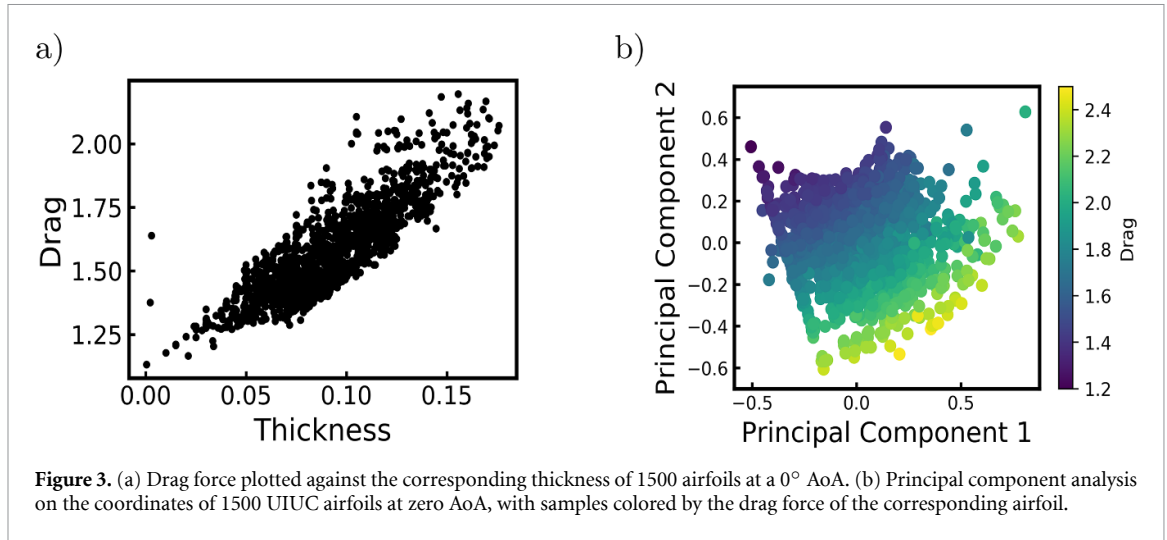


Figure 3. (a) Drag force plotted against the corresponding thickness of 1500 airfoils at a 0° AoA. (b) Principal component analysis on the coordinates of 1500 UIUC airfoils at zero AoA, with samples colored by the drag force of the corresponding airfoil.

where σ is the Cauchy stress tensor, e_x is the horizontal unit vector, n is the unit vector normal to the airfoil surface.

A grid convergence study is used to choose a specific mesh size to fully resolve the boundary layer effect while minimizing the computational time required. The resulted meshes contain 900–1500 mesh points and 3000–4000 edges, generated in effectively random spatial positions surrounding the airfoil.

The incompressible Navier–Stokes equation are given by

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = \nabla \cdot \sigma(\mathbf{u}, p) + f \quad (7)$$

$$\nabla \cdot \mathbf{u} = 0.$$

To predict the velocity field at the next time-step (u^{n+1}) from an existing time-step (u^n) while enforcing mass conservation, an IPCS is used to iteratively solve equation (6). A detailed description of the IPCS method can be found in [53].

The CFD outputs of each sample are then processed to generate the matrix of node level horizontal and vertical velocities as well as the adjacency matrix. However, storing an $N \times N$ adjacency matrix is memory-intensive. To bypass this issue, we instead store a matrix of dimension $2 \times E$, where E is the number of edges, compactly encoding the adjacency matrix. This compact representation only stores the two nodes that each edge connects in the graph. This representation is specifically helpful where the adjacency matrix is sparse which is true for the graphs in our dataset, as there are over 1200 nodes in the graph, and each node is only connected to five other nodes on average.

We perform our approach on two different sets of data. The first study is supposed to only examine the geometry of the airfoils. Therefore, the dataset consists of 1550 different airfoils. The second dataset, however, covers not only different geometries but also various angles of attack. 21 angles of attack are considered for 522 airfoils (10 962 samples in total). Angles of attack are changed in the range of -10° to 10° with an increment of 1° . Given the relatively low velocity in the domain and small angles of attack, the flow regime is laminar, and no significant flow separation occurs.

3.2. Experiments

Before implementing our method on the airfoil dataset, we perform a study of the relationship between the airfoil geometry and its drag force while other parameters are held constant. Without considering the effect of the AoA, we show the drag force to have a positive correlation with the thickness of the airfoil (figure 3(a)). However, this correlation, on its own, cannot cover a sufficient portion of the variance in the drag force. In order to determine the geometric features influencing the magnitude of the drag force, we conduct a principal component analysis (PCA) on the geometry of the airfoil and label the samples by their drag forces. By PCA, which is a linear dimensionality reduction technique, we extract components that can mostly describe the data variance in an unsupervised manner. While PCA is not generally interpretable, here we can observe the correlation of main components with the drag labels (figure 3(b)).

For the first experiment, we use the aforementioned GCNN architecture to predict the drag force for the dataset of airfoils with zero angles of attack. 80% of the samples are randomly selected for training and the

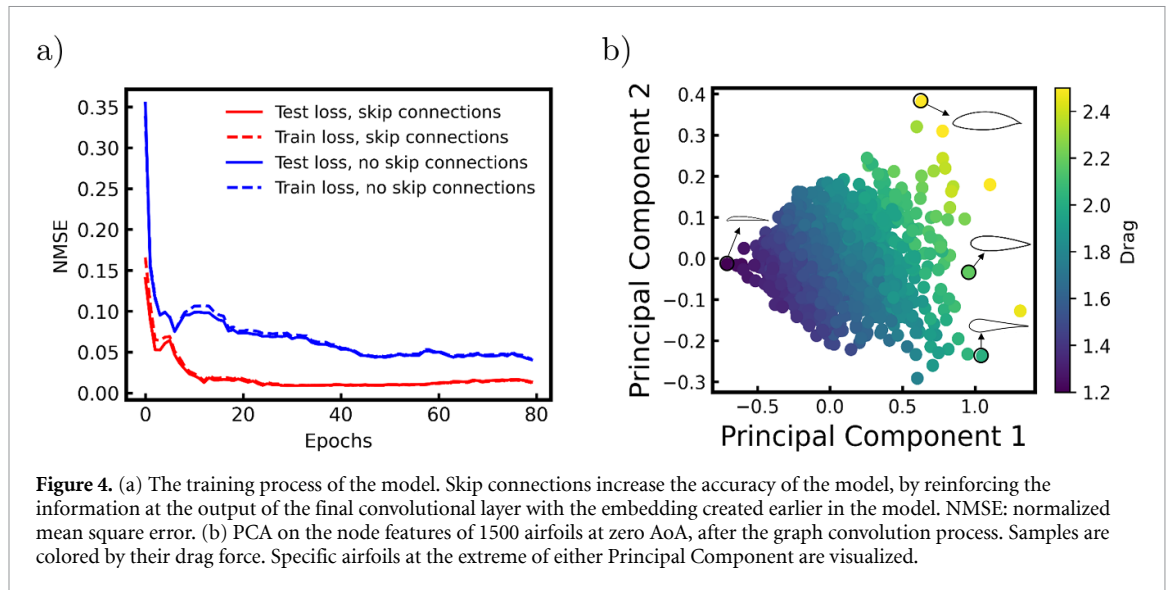


Figure 4. (a) The training process of the model. Skip connections increase the accuracy of the model, by reinforcing the information at the output of the final convolutional layer with the embedding created earlier in the model. NMSE: normalized mean square error. (b) PCA on the node features of 1500 airfoils at zero AoA, after the graph convolution process. Samples are colored by their drag force. Specific airfoils at the extreme of either Principal Component are visualized.

remainder are used as a test set. Two complementary metrics of mean squared error (MSE) of drag prediction and the coefficient of determination (R^2) are used to quantitatively evaluate the performance. Figure 4(a) shows the evolution of the loss metric as training progresses. The use of skip-connections in the architecture improves the model's accuracy.

Using node level velocities as the input to the network, the GCNN is anticipated to detect the most important features from the flow field data to accurately estimate the drag force. To illustrate the node embeddings produced by the convolution network, we analyze the values at the input to the first fully connected layer in the trained network which is the averaged output of the convolution layers.

To do so, we perform a PCA on the features and to detect the two most important geometric components that determine the drag force. It is noteworthy to emphasize that there is no geometrical feature directly encoded in the input of the network. Smooth transition of drag values with two components and depiction of the geometry of samples indicate that the network could learn meaningful geometrical features from flow field data. Here, the first two principal components can explain more than 90% of the variance in the dataset. The first component encodes a measure of airfoil thickness and the second component is an approximate measure of how quickly the airfoil tapers (figure 4(b)).

The network is also implemented on the second dataset, containing airfoils that vary in geometry and AoA, adding complexity to the prediction task. A comparison of the ground truth values of the drag forces from CFD result and the predictions from the network qualitatively shows the high accuracy of the model for both datasets (figure 5).

In addition to GCNNs, other machine learning and neural network algorithms can be applied on the flow field velocity data to solve a regression problem of drag prediction. Notice that the graph size and node order is not a matter of importance for the GCNN as we pass the adjacency and feature matrix directly to the model. However, non-graph based methods require a specified input size, as well as an identical node order between samples. Since the model perceives each input element as a different feature, it cannot be trained unless the order of the node elements are consistent between samples.

In order to benchmark the GCNN's performance against traditional, structured machine learning methods, we construct a node ordering that is consistent across samples. Traditional machine learning methods require the construction of a feature matrix, where the information described by a single feature—i.e., a single column of the feature matrix—must be consistent from sample to sample. In this formulation, feature i in sample x represents the same information as feature i in sample y . The node ordering is provided by the mesh generation software based on the order in which the nodes are generated, and is the same for each individual sample. Therefore, since the spatial density of the nodes in each sample is similar, this creates a matrix where Node i in sample x is relatively close in space to Node i as it appears in all other samples. To create a matrix with the same number of features for each sample, the closest 1000 nodes to the center of the airfoil are taken as the feature vector, as there are at least 1000 node measurements in each individual sample. To quantify the spatial similarity of nodes of the same index in this dataset, the distance between nodes of the same index are computed. After comparing the distance from node i in sample x with node i across all other samples, 98% of these distances are smaller than $5 \times 10^{-3}L$, where L is the length of

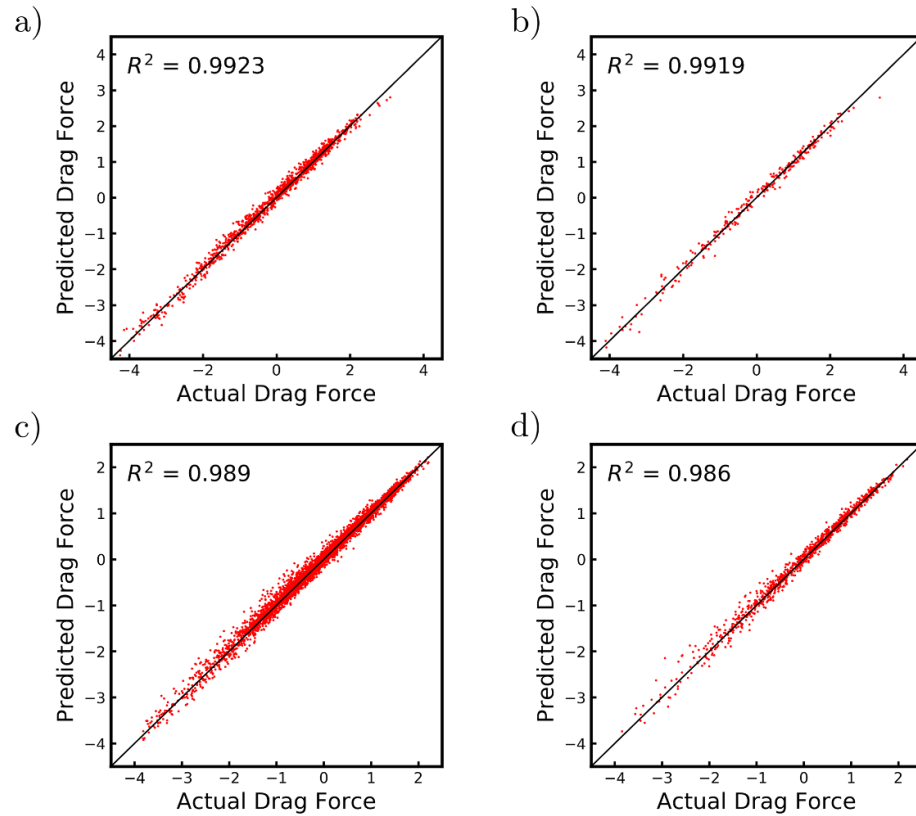


Figure 5. A comparison of GCNN predictions with the ground truth drag force. The first dataset contains 1500 samples of different airfoils at zero angles of attack. (a) Data from the training set. (b) Data from the test set. The second dataset consists of 5000 samples from 500 airfoils at angles of attack ranging from -10° to $+10^\circ$, with intervals of 1° . (c) Data from the training set. (d): Data from the test set.

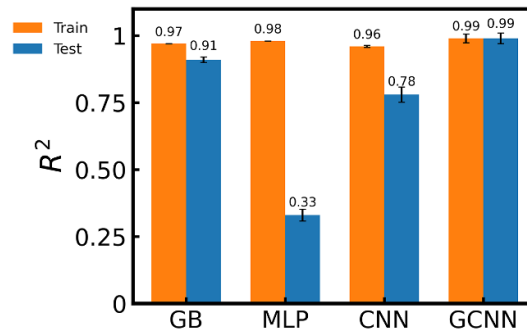


Figure 6. A comparison of the performance of different prediction algorithms. The dataset consists of 1500 airfoils at zero AoA (GB: gradient boosted random forest, MLP: multilayer perceptron, CNN: convolutional neural network, GCNN: graph convolutional neural network).

the domain. This indicates that the position of an arbitrary node is approximately consistent across samples.

To test the performance of non-graph based methods, we select a variety of the most used methods for performing prediction. Therefore, we compare the performance of Gradient Boosted Random Forest regression, a fully connected neural network, and a two-dimensional CNN on predicting the drag force based on a matrix of node features that adhere to the previously defined structure. Some basic details of these models are provided in table A2. The models have undergone hyperparameter tuning using the package HyperOpt [54], which uses a Tree-Structured Parzen Estimator based algorithm to search for the optimal hyperparameters given a suitable range. The results of the tuning process are tabulated in table A3. A comparison shows that the GCNN approach outperforms the non-graph based methods (figure 6). It is noteworthy that the consistent feature order used in this experiment allows the GB algorithm to achieve a

comparable accuracy to the GCNN, but this may not be applicable in cases where this information is not readily available. In fact, when the features are not provided to the model in a consistent order, the performance of the Gradient Boosting model sharply drops (figure 2).

4. Conclusion

We have introduced a novel approach based on GCNNs for data-driven prediction using unstructured field information. This method is able to take advantage of the properties of convolution, such as automatic feature detection and parameter sharing while being applied to unstructured data. Flow field properties are usually measured on sparsely scattered points, leading to unstructured data that are incompatible with traditional machine learning algorithms as they only can be applied to structured data. To evaluate the proposed model, the drag force of two-dimensional airfoils are estimated based on the horizontal and vertical components of the flow velocities, measured on the nodes of the irregular mesh around the airfoils. The result of this experiment demonstrates the capability of this approach for global property prediction based on flow field data in similar scenarios. Our model can potentially be extended to experimental cases where access to certain flow information is not readily available. With the currently implemented framework, only velocity information is used to calculate the drag. For instance, the required velocity information can be determined experimentally by analyzing the motion of a sparse set of tracer particles in the flow. By formalizing the edge connection between tracer observations as a connection from each measurement to the k nearest measurements, one can extend the framework of the GCNN to predict body forces.

The proposed idea of graph representation of the flow field data can be further used for prediction or classification of other field properties whether they are global, such as the drag force, or locally defined on the field. The algorithm can also be used for optimizing the desired properties for design and control applications.

Funding sources

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability statement

The related code will be available at <https://github.com/BaratiLab/Airfoil-GCNN> upon publication. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

The authors would like to thank Rishikesh Magar and Yuyang Wang for valuable comments and edits.

Appendix

Table A1. GCNN Parameters.

Layers	Parameters	Settings	Layers	Parameters	Settings
Graph convolution	Convolution width	64	Fully connected	Depth	3
	Depth (neighbor rings)	2		Width	[256,128,64,1]
	Activation	ReLU		Activation	ReLU
Top-K pooling	Ratio (k)	0.5	Others	Loss	MSE
				Optimiser	Adam

Table A2. Details of the trained models.

Model	Parameters	Settings	Model	Parameters	Settings
CNN	Num conv layers	5	MLP	Num layers	4
	Conv kernel	3*3		Width layers	512
	Depth Conv layers	[64,64,128,256,512]		Loss	MSE
	Max pooling	2*2	GB	Optimiser	SGD with momentum
	Num FC layers	3		Num estimators	500
	Width FC layers	[768,4096,1000,1]		Max depth	1

Table A3. Hyperparameter optimization (HPO) for the baseline GB, MLP, and CNN models as well as the proposed GCNN framework.

Model	Hyperparameter	Range	Selected
GB	No. estimators	[1,500]	500
	Max depth	{1,2,3,4,5}	1
MLP	Learning rate	[0.001,0.05]	0.00355
	Dropout	[0,0.9]	0.045
	Weight decay	$[1 \times 10^{-5}, 1 \times 10^{-3}]$	0.00016
	Activation	{tanh, Relu, sigmoid}	Relu
	No. layers	{1,2,...,10}	4
CNN	No. neurons	{64,128,256,512}	512
	Momentum	{0.5,0.6,0.7,0.8,0.9,0.95,0.99,0.999}	0.9
	Weight decay	{0.1,0.2,0.3,...,0.9}	0.2
GCNN	Dropout	{0.1,0.2,0.3,...,0.9}	0.4
	Learning rate	$[1 \times 10^{-7}, 1 \times 10^{-1}]$	0.0005
	Convolution width	{16,32,64,128,256}	64
	topK Ratio	{0.1,0.2,0.3,...,0.9}	0.5

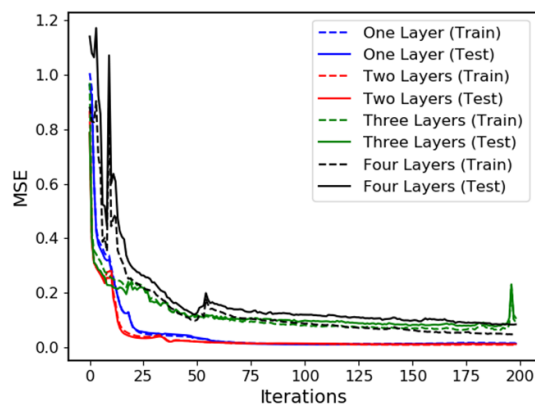


Figure A1. The effect of number of GCNN layers as a hyperparameter on the model's training and test errors. Based on the observed errors, two number of layers are selected and used for the experiments.

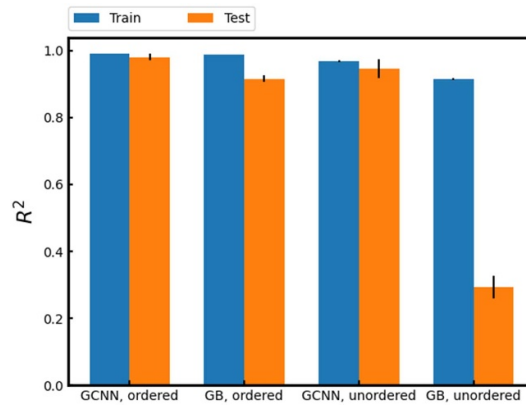


Figure A2. A comparison of the performance of our Gradient Boosting model and the GCNN model, for samples with a consistent feature order and samples without a consistent feature order.

ORCID iDs

Francis Ogoke  <https://orcid.org/0000-0002-2432-7783>

Kazem Meidani  <https://orcid.org/0000-0001-8441-8700>

Amir Barati Farimani  <https://orcid.org/0000-0002-2952-8576>

References

- [1] Han J, Jentzen A and Ee W 2018 Solving high-dimensional partial differential equations using deep learning *Proc. Natl Acad. Sci.* **115** 8505–10
- [2] Lye K O, Mishra S and Ray D 2020 Deep learning observables in computational fluid dynamics *J. Comput. Phys.* **410** 109339
- [3] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
- [4] Pang G, D'Elia M, Parks M and Karniadakis G E 2020 nPINNs: nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator algorithms and applications *J. Comput. Phys.* **422** 109760
- [5] Jagtap A D, Kharazmi E and Karniadakis G E 2020 Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems *Comput. Methods Appl. Mech. Eng.* **365** 113028
- [6] Sharma R, Farimani A B, Gomes J, Eastman P and Pande V 2018 Weakly-supervised deep learning of heat transport via physics informed loss (arXiv:1807.11374)
- [7] Kim J and Lee C 2020 Prediction of turbulent heat transfer using convolutional neural networks *J. Fluid Mech.* **882** A18
- [8] Wiewel S, Becher M and Thuerey N 2018 Latent-space physics: towards learning the temporal evolution of fluid flow (arXiv:1802.10123)
- [9] Farimani A B, Gomes J and Pande V S 2017 Deep learning the physics of transport phenomena (arXiv:1709.02432)
- [10] Kissas G, Yang Y, Hwuang E, Witschey W R, Detre J A, Perdikaris P 2019 Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks (arXiv:1905.04817)
- [11] Maulik R, Fukami K, Ramachandra N, Fukagata K, Taira K 2020 Probabilistic neural networks for fluid flow model-order reduction and data recovery (arXiv:2005.04271)
- [12] Yan S, He Y, Tang T and Wang T 2019 Drag coefficient prediction for non-spherical particles in dense gas–solid two-phase flow using artificial neural network *Powder Technol.* **354** 115–24
- [13] Cai S, Zhou S, Xu C and Gao Q 2019 Dense motion estimation of particle images via a convolutional neural network *Exp. Fluids* **60** 73
- [14] Noca F, Shiels D and Jeon D 1999 A comparison of methods for evaluating time-dependent fluid dynamic forces on bodies, using only velocity fields and their derivatives *J. Fluids Struct.* **13** 551–78
- [15] Graham W R, Ford C P and Babinsky H 2017 An impulse-based approach to estimating forces in unsteady flow *J. Fluid Mech.* **815** 60–76
- [16] Rabault J, Kolaas J and Jensen A 2017 Performing particle image velocimetry using artificial neural networks: a proof-of-concept *Meas. Sci. Technol.* **28** 125301
- [17] Morimoto M, Fukami K, Fukagata K 2020 Experimental velocity data estimation for imperfect particle images using machine learning (arXiv:2005.00756)
- [18] Hou W, Darakananda D and Eldredge J 2019 *AIAA Scitech 2019 Forum* p 1148
- [19] Le Provost M, Hou W and Eldredge J 2020 *AIAA Scitech 2020 Forum* p 0799
- [20] Zhang Y, Sung W J and Mavris D N 2018 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Conf. p 1903
- [21] Viquerat J, Hachem E 2019 A supervised neural network for drag prediction of arbitrary 2D shapes in low Reynolds number flows
- [22] Yilmaz E and German B 2017 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf. p 3660
- [23] Guo X, Li W and Iorio F 2016 *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD '16* (New York, NY: Association for Computing Machinery) pp 481–90
- [24] Miyanawala T P and Jaiman R K 2017 An efficient deep learning technique for the Navier–Stokes equations: application to unsteady wake flow dynamics (arXiv:1710.09099)

- [25] Bhatnagar S, Afshar Y, Pan S, Duraisamy K and Kaushik S 2019 Prediction of aerodynamic flow fields using convolutional neural networks *Comput. Mech.* **64** 525–45
- [26] Gross B, Trask N, Kuberry P and Atzberger P 2020 Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: a generalized moving least-squares (GMLS) approach *J. Comput. Phys.* **409** 109340
- [27] Schlüter-Kuck K L and Dabiri J O 2017 Coherent structure colouring: identification of coherent structures from sparse data using graph theory *J. Fluid Mech.* **811** 468–86
- [28] Padberg-Gehle K and Schneide C 2017 Network-based study of Lagrangian transport and mixing *Nonlinear Process. Geophys.* **24** 661
- [29] Hadjighasem A, Karrasch D, Teramoto H and Haller G 2016 Spectral-clustering approach to Lagrangian vortex detection *Phys. Rev. E* **93** 063107
- [30] Meena M G, Nair A G and Taira K 2018 Network community-based model reduction for vortical flows *Phys. Rev. E* **97** 063103
- [31] Trask N, Patel R G, Gross B J, Atzberger P J 2019 GMLS-Nets: a framework for learning from unstructured data (arXiv:1909.05371)
- [32] Battaglia P W et al 2018 Relational inductive biases, deep learning, and graph networks (arXiv:1806.01261)
- [33] Bastings J, Titov I, Aziz W, Marcheggiani D, Sima'an K 2017 Graph convolutional encoders for syntax-aware neural machine translation (arXiv:1704.04675)
- [34] 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)
- [35] Yu B, Yin H and Zhu Z 2018 *Proc. Twenty-Seventh Int. Conf. on Artificial Intelligence, IJCAI-18* (Int. Joint Conferences on Artificial Intelligence Organization) pp 3634–40
- [36] Yao L, Mao C and Luo Y 2019 *Proc. Conf. on Artificial Intelligence* vol 33 pp 7370–7
- [37] Xie T and Grossman J C 2018 Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties *Phys. Rev. Lett.* **120** 145301
- [38] Chen C, Ye W, Zuo Y, Zheng C and Ong S P 2019 Graph networks as a universal machine learning framework for molecules and crystals *Chem. Mater.* **31** 3564–72
- [39] Duvenaud D K, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A and Adams R P 2015 Convolutional networks on graphs for learning molecular fingerprints vol 28 *Advances in Neural Information Processing Systems* pp 2224–32
- [40] Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J and Battaglia P W 2020 Learning to simulate complex physics with graph networks (arXiv:2002.09405)
- [41] Alet F, Jeewajee A K, Bauza M, Rodriguez A, Lozano-Perez T and Kaelbling L P 2019 Graph element networks: adaptive, structured computation and memory (arXiv:1904.09019)
- [42] Kipf T N, Welling M 2016 Semi-supervised classification with graph convolutional networks (arXiv:1609.02907)
- [43] Zhang M and Chen Y 2018 *Proc. 32nd Int. Conf. on Neural Information Processing Systems, NIPS'18* (Red Hook, NY: Curran Associates Inc.) pp 5171–81
- [44] Gilmer J, Schoenholz S S, Riley P F, Vinyals O, Dahl G E 2017 Neural message passing for quantum chemistry (arXiv:1704.01212)
- [45] Hamilton W L, Ying R and Leskovec J 2017 Inductive representation learning on large graphs (arXiv:1706.02216)
- [46] Fey M, Lenssen J E 2019 Fast graph representation learning with PyTorch geometric (arXiv:1903.02428)
- [47] Cangea C, Velićović P, Jovanović N, Kipf T, Liò P 2018 Towards sparse hierarchical graph classifiers (arXiv:1811.01287)
- [48] Gao H, Ji S 2019 Graph u-nets (arXiv:1905.05178)
- [49] Paszke A et al 2019 *Advances in Neural Information Processing Systems 32 H Wallach, H Larochelle, A Beygelzimer, F d' Alché-Buc, E Fox and R Garnett* pp 8024–35
- [50] Selig M 1996 *UIUC Airfoil Data Site* (Department of Aeronautical and Astronautical Engineering University of Illinois at Urbana-Champaign)
- [51] Geuzaine C and Remacle J-F 2009 Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities *Int. J. Numer. Methods Eng.* **79** 1309–31
- [52] Logg A and Wells G N 2010 Dolfin *ACM Trans. Math. Softw.* **37** 1–28
- [53] Thomadakis M and Leschziner M 1996 A pressure-correction method for the solution of incompressible viscous flows on unstructured grids *Int. J. Numer. Methods Fluids* **22** 581–601
- [54] Bergstra J, Yamins D and Cox D D 2013 *Proc. 30th Int. Conf. on Int. Conf. on Machine Learning - Volume 28, ICML'13* (JMLR.org) p I-115-I-123