What a drag! Streamlining the UAV design process with design grammars and drag surrogates

Michael Sandborn, Carlos Olea, Anwar Said, Mudassir Shabbir, Peter Volgyesi, Xenofon Koutsoukos, Jules White
{michael.sandborn, carlos.d.olea, anwar.said, mudassir.shabbir,
peter.volgyesi, xenofon.koutsoukos, jules.white}@vanderbilt.edu
Vanderbilt University, USA

***Abstract*—Unmanned Aerial Vehicles (UAVs) continue to proliferate, revolutionizing tasks such as cargo transport, surveillance, and search and rescue operations. With the discovery of novel use cases or specialized tasks for aerial vehicles, there is an increased need for improved design space exploration and performance estimation techniques for candidate UAV designs. Typical pipelines for this design process rely on time-consuming human efforts to identify productive design geometries or expensive computational approaches for performance analysis to reconcile aerodynamic, electrical, and physical interactions.**

In this work-in-progress paper, we propose the use of a design process that uses a design grammar for UAV design generation and a Graph Neural Network (GNN)-based drag surrogate trained on simulation data for accelerated UAV design space exploration. We formulate a UAV design grammar and provide preliminary performance results from the GNN drag surrogate for randomly generated designs. We expect our approach to accelerate the exploration of UAV design geometries using a learned surrogate drag model to circumvent resource-hungry Computer-aided design (CAD) and simulation routines.

***Index Terms*—Design Space Exploration, Design Grammar, Graph Neural Network, Drag Surrogate, Unmanned Aerial Vehicle**

I. Introduction

Unmanned Aerial Vehicles (UAVs) continue to revolutionize areas such as emergency response, shipping, transportation, and military technology. The market for UAVs is expected to grow to roughly $78 billion by 2030 [1] and the market for UAV simulation is expected to grow to $1.5 billion by 2027 [2]. As novel applications for UAVs are identified, the need for identifying diverse yet feasible and performant design geometries will increase.

Currently, the development process for UAVs relies heavily on human designers to draft vehicle geometries (e.g. create CAD models) based on domain expertise and then iteratively refine design candidates using simulation-based evaluation techniques (e.g. flight dynamics) to make small changes to the overall design geometry. However, human engineering time is expensive and simulation throughput may be infeasible for evaluating hundreds or thousands of designs in a time-sensitive manner. Additionally, bottlenecks exist in CAD modeling software which must assemble a model from its components for each design of interest which can be resource-intensive.

There is also a tradeoff between design diversity and feasibility in the design space— it may be desirable to either (1) alter a known good design (e.g. quadcopter) such that it remains in a feasibility region for flight stability or (2) disregard flight feasibility to prioritize novelty of vehicle geometries. Design space exploration approaches must reconcile these two conflicting objectives. If flight feasibility is the sole objective, the space of vehicle geometries when incorporating current aerodynamics knowledge is relatively rigid and UAV designs may lack geometric diversity. Conversely, if geometric novelty is the main objective, there is more legwork for evaluating the airworthiness of a generated design with novel geometry since it would likely differ greatly from incumbent aerial vehicle shapes (e.g. airplane, quadcopter, helicopter, hexacopter). It is also possible that geometry-based heuristics such as axial symmetry and vehicle spanning area can help guide the generation of nearly airworthy designs, for example in an evolutionary setting.

We aim for a process to address the conflicting geometry and flight feasibility objectives, ideally with minimal human intervention for design generation and heuristic evaluation. There are two main goals to consider. First, we would like to compactly express the geometry of a UAV design including its components and how they are connected (e.g. a motor connected to a propeller). We use the term *geometry* to refer to both the overall shape of a UAV including the orientation and connectivity of its components. Second, we would like to quickly estimate the performance of a UAV to assess whether the UAV geometry is airworthy without relying on substantial computation (e.g. simulation).

In particular, given a UAV design sampled from a design grammar, we would like a surrogate drag model that provides a reliable estimate of a vehicle's drag profile *without the use of a CAD assembly process or CFD simulation*, since not every design that is explored will be worth assembling for flight simulation. We consider a *drag profile* to be a real-valued vector that captures information such as the centers of drag, the magnitudes of drag forces, or the vehicle area that is sensitive to drag in the x, y, z directions. Drag is chosen as the performance heuristic because it is directly related to the overall geometry of the vehicle and is also indicative of the performance or efficiency of a design. Patterns learned by the drag surrogate can then inform updates to the production rules in the design grammar in order to produce increasingly stable or feasible designs over time. This approach is summarized in

Figure 1. The contributions of this work are:

- We formulate a string-based design grammar for Unmanned Aerial Vehicles (UAVs)
- We produce an initial dataset of randomly generated UAV designs along with drag profiles from simulation as labels for estimating UAV design feasibility
- We provide preliminary results from a GNN-based drag surrogate which predicts the drag profile of a UAV design from its representative design graph

The remainder of this paper is organized as follows: Section II briefly outlines related work, Section III introduces our proposed methods for rapid generation and exploration of UAV designs using a design grammar and drag surrogate, Section IV provides preliminary experimental results, and Section V gives concluding remarks.

II. Related Work

A. Design Grammars

Design grammars are sets of productions rules that define a valid design in some domain and are powerful tools for both analyzing existing systems and generating instances of data that are constrained by a set of production rules. Recently, efforts such as Zhao et al. [3] have had success in applying the use of a design grammar, particularly a graph grammar, to the space of underwater vehicle design. These efforts were inspired by the work of Sims [4], Zhao et al. [5] and Stockli et al. [6]. In this work, we develop a design grammar for UAVs.

B. Graph Neural Networks

Graph Neural Networks (GNNs) provide a framework for learning on structured data represented as nodes connected by edges. These networks capture latent local and global patterns in graph data by aggregating neighborhood information and propagating node and edge level features for tasks such as graph classification and graph regression [7]. Recently, GNNs have enabled breakthroughs in areas from protein folding prediction [8] to social networks analysis [9] to traffic forecasting [10]. In this work, we propose the use of GNNs for predicting the drag profile of a UAV design by leveraging vehicle connectivity and component information. Although GNNs have been investigated in the context of coordinating UAV communication [11], to the best of our knowledge, GNNs have not previously been studied as drag surrogates to aid in design space exploration and heuristic evaluation of UAVs.

C. Physics-Guided Learning

Physics-guided learning combines equations that govern dynamical systems (e.g. the Navier-Stokes equations for fluid flow) with the power of deep learning to improve the fidelity of physical simulation especially in areas with high dimensionality and uncertainty (e.g. fluid dynamics). Recent work has examined the use of learning drag coefficients of obstacles using Multilayer Perceptrons (MLP) [12], estimating drag force for 2D objects using convolutional networks (CNNs) [13], and predicting the drag force of particles in moving fluids [14] or on airfoils [15]. More closely related to our work, Sanchez-Gonzalez et al. [16] and Ogoke et al. [17] investigate the prediction of drag force and flow field characteristics using GNNs but do not focus on vehicles. We focus on the use of GNNs to predict drag profiles of arbitrary UAV geometries.

III. Proposed Method

In this section we outline the proposed approach for learning drag profiles of UAV designs represented as graphs. Because UAVs capable of stable flight likely occupy only a small fraction of the UAV design space, there is a need for a productive, heuristic-based search strategy. The design space for UAVs grows rapidly even without considering non-geometric parameters such as electrical or control parameters. Consider a vehicle with n parameterized components represented as graph nodes, and m inter-component connections represented as graph edges. Assume that each component must be connected to the vehicle which imposes the requirement of $m_{min} = n-1$ edges to represent a valid vehicle, and that a vehicle on average contains $n_{avg} = 20$ components. Also, assume that a vehicle has at most a quarter of the number of possible edges connecting components $m_{max} \leq n_{avg}(n_{avg}-1)/8$. The number of possible graphs satisfying these requirements is given by $\binom{\binom{n_{avg}}{2}}{m_{max}} \approx 9.71^{44}$ graphs for vehicles with exactly $n_{avg} = 20$ components and exactly $\lfloor n_{avg}(n_{avg}-1)/8 \rfloor = 47$ edges for component connections. The count of all graphs containing n_{avg} components (ignoring isomorphisms) with connections ranging from m_{min} to m_{max} edges is given by $\sum_{k=m_{min}}^{m_{max}} \binom{\binom{n_{avg}}{2}}{k}$. Clearly, a heuristic search of this space is necessary to identify feasible designs.

A. Problem Formulation

Given a UAV design geometry represented as a graph that originates from our design grammar, we aim to train a GNN model that reliably estimates the drag profile of a UAV design, a performance heuristic which we hypothesize correlates strongly with flight stability. We use *geometry* to refer to the shape of a UAV including its components and their connections.

We consider UAV geometries that are produced from a design grammar over strings whose production rules encode allowable component types, connections and relative positions on a UAV. Given a design string generated from the grammar, we form the corresponding graph by creating nodes for each of n parameterized (e.g. length) components. The m edges of the graph indicate which components are connected to which other components on the UAV.

We indicate the i^{th} UAV design graph with $\mathcal{G}_i(n, m)$. From here, we obtain the "ground truth" from the simulation denoted as f as $\mathcal{Y}_i = f(\mathcal{G}_i(n, m))$ which is a vector $d \in \mathbb{R}^6$ that represents the drag profile and includes the centers and areas affected by drag in the x, y and z directions for the vehicle represented by $\mathcal{G}_i(n, m)$.

After obtaining a set $\mathcal{S} = (\mathcal{G}_j, \mathcal{Y}_j)$ of randomly generated UAV design graphs (training samples) and their drag profiles (labels) we train a GNN drag surrogate $f_s(\mathcal{S})$ to obtain drag predictions on new UAV designs. Figure 1 summarizes

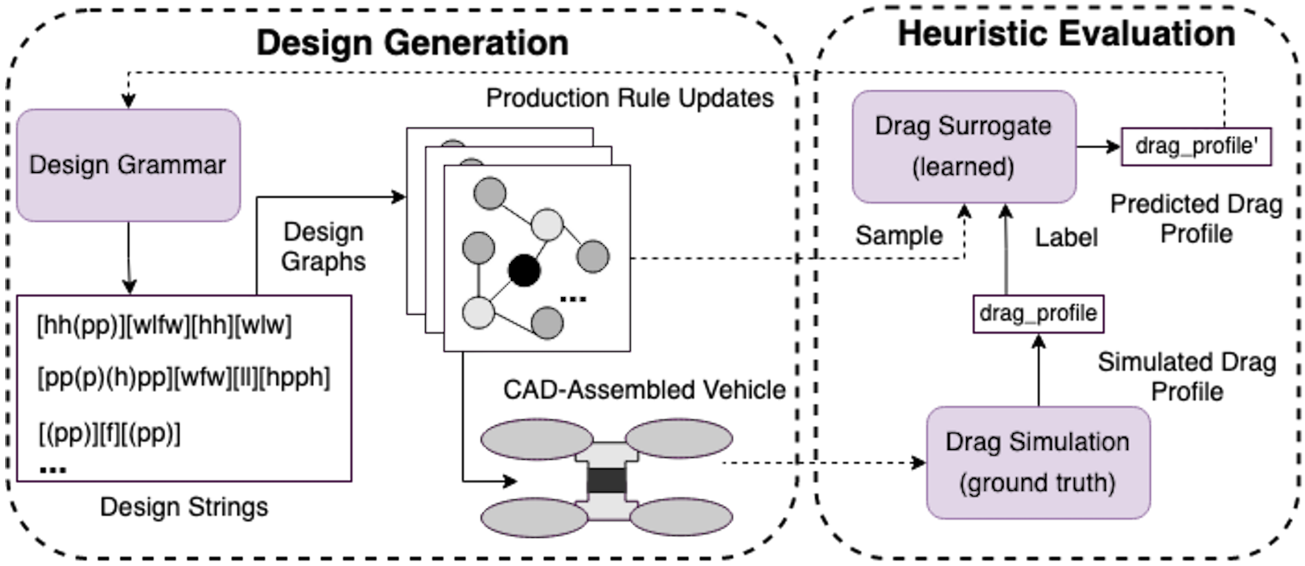

Fig. 1. The proposed approach includes 2 main stages: (1) Design Generation and (2) Heuristic Evaluation. In stage (1) we produce designs from a design grammar and convert them into graphs to represent vehicle connectivity and component attributes. The assembled design (e.g. CAD model) is then provided as input to the drag simulation. In stage (2) the drag simulation computes the drag profile for the assembled UAV in the x, y, and z directions. The UAV design graph is the input sample and the ground truth drag profile is the label for the GNN drag surrogate.

⟨*terminal-token*⟩ ::= h | p | l | w | f

⟨*non-terminal-token*⟩ ::= (|) | [|]

⟨*propeller*⟩ ::= h | p

⟨*fuselage-cluster*⟩ ::= [,[w], w, [l, p], f, w, [w],]

⟨*wing-cluster*⟩ ::= w, [l], w | l [l]

⟨*sub-cluster*⟩ ::= (, {⟨*propeller*⟩}, ⟨*propeller*⟩, {⟨*propeller*⟩},)

⟨*prop-cluster*⟩ ::= [,{⟨*prop-cluster*⟩},{⟨*sub-cluster*⟩}, {⟨*propeller*⟩}, {⟨*sub-cluster*⟩}, {⟨*prop-cluster*⟩},]

⟨*cluster*⟩ ::= ⟨*wing-cluster*⟩ | ⟨*propeller-cluster*⟩

⟨*design*⟩ ::= {⟨*cluster*⟩}, ⟨*fuselage-cluster*⟩, {⟨*cluster*⟩}

Fig. 2. EBNF representation of the UAV design string grammar

this process. The GNN approach is detailed further in Section III-C.

B. UAV Design Grammar

When exploring a design space, two conflicting objectives must be addressed: maximizing the proportion of the space explored containing performant designs and minimizing the proportion of the space containing non-performant designs. These two axes are affected by the amount and complexity of the restrictions on design generation. For the UAV design space, we select a string grammar to encode UAV designs.

Our string grammar functions by first generating strings according to the production rules shown in Figure 2. Although one-dimensional, each string is embedded with three dimensional consideration through the use of framing by non-terminal tokens and generation by cluster. Clusters are denoted by brackets, while sub-clusters (necessarily within clusters) are denoted by parentheses. The dimensions can be considered as follows: the leftmost cluster of any given string represents the frontmost cluster in 3D space. Similarly, the rightmost cluster of any given string represents the rearmost cluster in 3D space. Within the bounds of a given cluster, the (right)leftmost part of the cluster in the string represents the (right)leftmost part of the cluster in 3D space. Sub-clusters are assumed to deviate either up or down from the cluster they are enclosed by, though the polarity and magnitude of the deviation is left to be decided by the interpreter and allows for additional design freedom.

After a string is generated, it is augmented and transformed into a graph. The interpreter takes in the generated string and a set of options for each component type (e.g. propeller sizes), information on connection ports and allowable connections for each component (e.g. motor connected to propeller). Component attributes (e.g. length, width) are used to calculate bounding boxes and prevent collisions when placing parts in 3D space. Connection ports and allowable connection types specify all possible sets of valid connections given the structure and content of a design string. For example, should the placements of sub-clusters be above or below their base cluster, and by how much? Should sub-cluster placement be grouped or alternating? Other possible parameters could be whether a design string should be interpreted as having one, two, or more principal axes for connectors.

Once all the required elements have been provided, the interpreter places all the elements of the design string into three dimensional space, creating a graph of nodes consisting of both parts and connectors, and edges denoting connections between ports of parts and connectors.

One may ask why we use a string grammar as opposed to graph grammar. Similar efforts, such as Zhao et al. [3], have explored the use of graph grammars for designing underwater vehicles. While there is certainly merit in using a graph grammar for vehicle design, especially as vehicles are often most easily represented in detail via graphs due to their ease of encapsulating parts and their connections, there are some drawbacks. The most notable being that a graph grammar in this case would be monolithic, in that the production rules must encapsulate substantial domain knowledge in order to appropriately restrict the state space to valid designs. By comparison, a string grammar with an interpreter is more modular allowing for granular changes. While it does increase the steps required to produce a design, it lends itself to streamlined adjustments later, for instance using by reinforcement learning techniques or fitness-based evolutionary methods.

C. Graph Representation Learning

In this section, we describe how we leverage the graph learning approach to solve this problem. In particular, we discuss the Deep Graph Convolutional Network (DGCNN) [18] framework in a graph regression setting as the learned drag surrogate for UAVs.

The last few years have observed the growing prevalence of deep learning (DL) methods on various application domains such as computer vision, natural language processing and graph representation learning [19]. Because graphs provide a powerful and general formalism for representing a wide range of real-world systems, graph representation approaches have seen a surge in recent years. In particular, Graph Neural Networks (GNNs) learn graph structure by leveraging both the topology and node information. GNNs are flexible to several downstream machine learning tasks such as node and graph classification, graph regression, link prediction, and community detection [20], [21]. Since the performance estimation is a graph regression task, we define it as follows.

Problem 3.1: (Graph Regression) Let $\mathcal{G} = \{G_1, \ldots, G_N\}$, be a set of graphs representing UAVs and $\mathcal{Y} = \{y_1, \ldots, y_N\}$ be their corresponding drag profiles. Given $\mathcal{G}$ and $\mathcal{Y}$, we aim to learn a representation vector h_G that helps in predicting the estimation $y_{G'}$ for an unseen graph G'.

The most common approaches for graph-level learning typically involve aggregation after extracting node-level features. However, such aggregation results in significant information loss, reducing model performance. When dealing with the graph regression problem, it is critical to keep as much vertex information as possible. Maintaining such information allow models to learn both local and global-level information and thus obtain improved performance. Keeping this in view, the authors in [18] proposed DGCNN where a new pooling layer was introduced that arranges node features in a consistent order to further use convolutional neural networks (CNNs). Unlike the traditional GNNs, the architecture of DGCNN also involves 1D convolutional layers to extract expressive graph-level representations and has shown outstanding performance. In this work, we employ DGCNN in our experimental setup with a slight variation to perform the desired graph regression task. In the following, we briefly discuss the layer-wise architecture of the DGCNN model.

D. Graph convolutional Layers

The graph convolution layers usually involve a message passing mechanism to learn node features. Given the adjacency matrix A, and feature matrix X, DGCNN considers the following form of convolution.

$$H = f(\tilde{D}^{-1}\tilde{A}XW) \tag{1}$$

Where $\tilde{D}$ is a normalized degree matrix, $\tilde{A}$ is the adjacency matrix with self-loops, and W is the matrix of trainable parameters. f can be any nonlinear function. Having the generalized convolutions similar to GCN [22], DGCNN accommodates a variety of graph convolution mechanisms.

E. The SortPooling Layer

The main idea behind SortPooling layer is to bring the node features into a consistent order so that any type of Neural Networks or CNNs can be applied. To do so, the author use Weisfeiler Lehman (WL) [23] coloring scheme to sort node features. Given H^l where l is the last layer of GNN convolutions, SortPooling first sorts H (row-wise) in descending order and then sort the vertices in the same order accordingly. For the scale-invariance, the sorted features are further trimmed so only k features are chosen. This mechanism provides scale invariance and allows DGCNNs to impose ordering in the feature space.

F. Other Layers

DGCNN employs additional CNNs layers after sort pooling to produce expressive graph representations. It flattens the feature matrix first, then applies two one-dimensional convolution layers and several MaxPooling layers. Finally, a fully connected layer with the desired activation function is applied.

We use the same architecture as DGCNN, with the exception of the base convolution model and the activation function for the final layers. We use GraphSAGE graph convolution [24] instead of GCN [22] because it performs better in our case. We also removed the Softmax from the final layer because of the regression task.

IV. Experimental Evaluation

Using DGCNN, we run experiments on our generated design topologies to estimate performance on different test designs. In our experimental setup, we use publicly available DGCNN implementations and the same architecture. Because of very limited data, we consider quite a slim model to experiment with. For training, we use a 70:30 train-test split ratio and L1 Loss with a learning rate of $1e^{-4}$. We used GraphSAGE convolution with three layers and 32 hidden channels. The number of neurons in the final fully connected layers was set to $416, 16$, and 1, respectively. We also performed MinMax normalization on labels and trained the model for 200 epochs. We show the performance of DGCNN in terms of L1 loss

in Figure 3. The x-axis indicates the number of epochs and the y-axis shows loss value at each epoch. Our initial results of decreasing loss on both train and test sets show promise for our approach of learning drag profiles of UAV geometries using GNNs.

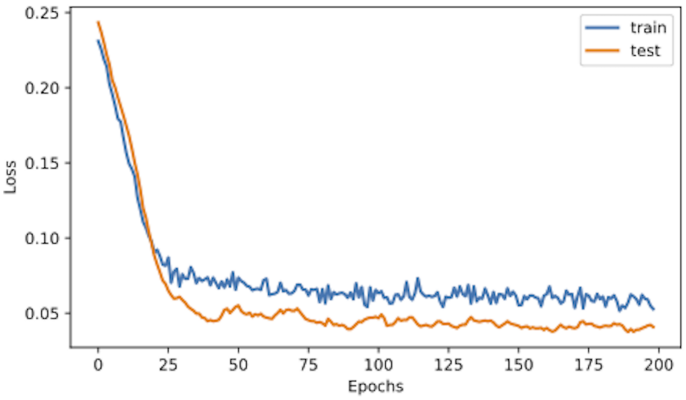

Fig. 3. DGCNN loss curve

V. Concluding Remarks

We investigate the problem of UAV design space exploration for rapid generation and heuristic evaluation of vehicle geometries. To achieve this, we propose the combination of a design grammar and a graph neural network-based drag surrogate for generation and surrogate evaluation of many UAV designs, respectively, to circumvent expensive CAD assembly and simulation routines.

There are 3 main next steps for this work: (1) generation of additional UAV design data, (2) exploration of additional node attributes such as component locations and centers of gravity, and (3) refinement of the grammar production rules for generating increasingly stable UAV designs using insight from predictions of the learned GNN drag surrogate.

VI. Acknowledgement

This work is supported by DARPA's Symbiotic Design for CPS project and by the Air Force Research Laboratory (FA8750-20-C-0537). Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or AFRL.

References

[1] https://www.prnewswire.com/news-releases/unmanned-aerial-vehicle-market-size-to-reach-usd-77-69-billion-in-2030–increasing-demand-for-uavs-in-military-applications-is-one-of-the-key-factors-driving-industry-demand-says-emergen-research-301528073.html.

[2] https://www.prnewswire.com/news-releases/drone-simulator-market-worth-1-501-million-by-2027–exclusive-report-by-marketsandmarkets-301630849.html.

[3] A. Zhao, J. Xu, J. Salazar, W. Wang, P. Ma, D. Rus, and W. Matusi, "Graph grammar-based automatic design for heterogeneous fleets of underwater robots."

[4] S. K, "Evolving virtual creatures," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, vol. 21, 1994.

[5] Z. A, X. J, K.-L. M, H. J, S. A, R. ´ D, and M. W, "Robogrammar: graph grammar for terrainoptimized robot design," in *ACM Transactions on Graphics*, vol. 39, no. 6, 2020.

[6] F. R. Stockli and K. Shea, "A simulation-driven graph grammar," in *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection*, 2015.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, Jul. 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03819-2

[9] Z. Wang, T. Derr, D. Yin, and J. Tang, "Understanding and predicting weight loss with mobile social networking data," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1269–1278. [Online]. Available: https://doi.org/10.1145/3132847.3133019

[10] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *CoRR*, vol. abs/2101.11174, 2021. [Online]. Available: https://arxiv.org/abs/2101.11174

[11] P. Li, L. Wang, W. Wu, F. Zhou, B. Wang, and Q. Wu, "Graph neural network-based scheduling for multi-uav-enabled communications in d2d networks," 2022. [Online]. Available: https://arxiv.org/abs/2202.07115

[12] K. U. Rehman, A. B. Çolak, and W. Shatanawi, "Artificial neural networking (ann) model for drag coefficient optimization for various obstacles," *Mathematics*, vol. 10, no. 14, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/14/2450

[13] J. Viquerat and E. Hachem, "A supervised neural network for drag prediction of arbitrary 2d shapes in laminar flows at low reynolds number," *Computers & Fluids*, vol. 210, p. 104645, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045793020302164

[14] N. Muralidhar, J. Bu, Z. Cao, L. He, N. Ramakrishnan, D. Tafti, and A. Karpatne, "Physics-guided design and learning of neural networks for predicting drag force on particle suspensions in moving fluids," 2019. [Online]. Available: https://arxiv.org/abs/1911.04240

[15] W. Peng, Y. Zhang, E. Laurendeau, and M. C. Desmarais, "Learning aerodynamics with neural network," *Scientific Reports*, vol. 12, no. 1, Apr. 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-10737-4

[16] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," 2020. [Online]. Available: https://arxiv.org/abs/2002.09405

[17] F. Ogoke, K. Meidani, A. Hashemi, and A. B. Farimani, "Graph convolutional networks applied to unstructured flow field data," *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045020, sep 2021. [Online]. Available: https://doi.org/10.1088/2632-2153/ac1fc9

[18] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," vol. 32, no. 1. Proceedings of the AAAI conference on artificial intelligence, 2018.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[21] A. Said, S.-U. Hassan, W. Abbas, and M. Shabbir, "Netki: a kirchhoff index based statistical graph embedding in nearly linear time," *Neurocomputing*, vol. 433, pp. 108–118, 2021.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[23] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Series*, vol. 2, no. 9, pp. 12–16, 1968.

[24] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.