Software Impacts 17 (2023) 100567

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

AddLat2D the 2D Lattice Generator

Martha Baldwin [a], Nicholas A. Meisel [b], Christopher McComb [a,*]

[a] Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States of America
[b] School of Engineering Design and Innovation, The Pennsylvania State University, University Park, PA 16802, United States of America

ARTICLE INFO

Keywords:
Lattice design
Design for additive manufacturing (DfAM)
Unit cells
Shape design
Mesostructure
Data generation

ABSTRACT

This work addresses the challenges of acquiring additive manufacturing data, given the complexities and design possibilities of such structures. Researchers in additive manufacturing struggle with scarcity and unsuitability of 2D datasets which pose further difficulties. To overcome these concerns, this research presents an application, AddLat2D, for generating 2D lattice structure datasets tailored to user specifications. Building upon a previous version of the application (Baldwin et al., 2023, 2022), this work highlights our development and usage of AddLat2D to generate datasets that have custom image size and pixel intensity values.

Code metadata

Current code version	*v1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-259
Permanent link to reproducible capsule	https://codeocean.com/capsule/2039426/tree/v1
Legal code license	*MIT*
Code versioning system used	*git*
Software code languages, tools and services used	*Python, Hugging Face, Streamlit*
Compilation requirements, operating environments and dependencies	
If available, link to developer documentation/manual	
Support email for questions	ccm@cmu.edu

1. Introduction

Data collection for additive manufacturing applications is often difficult to achieve due to the inherent complexity and size of the possible structures [1]. Additive manufacturing provides extreme manufacturing flexibility, which enables countless possibilities for creating structures that can be classified as lattices [2,3]. Concurrently, to develop effective machine learning algorithms, it is often important to have a carefully curated and large dataset, especially when applying them to additive manufacturing tasks [4]. Therefore, it is appealing to generate synthetic data to mimic the lattice structures used when developing machine learning solutions for additive manufacturing applications.

Recent research has explored various tools specifically designed to develop diverse shape and property spaces of unit cells for additive manufacturing [5,6]. However, broader machine learning applications require datasets to have a specific data size and labeled data points [4]. To address these concerns, we have developed dataset generation software specifically tailored to create a strut-based 2D unit lattice cell dataset, allowing customization of data size and pixel intensity. This software builds upon previous works that necessitated data generation [7,8]. The software, referred to as AddLat2D [9,10], enables the generation of a CSV file containing a dataset of 2D shapes, with options to modify the data format according to user preferences. The primary objective of this software is to provide users with access to customizable data that aligns with their specific needs. Moreover, the resulting datasets encompass labels for all the data points, fostering their utilization in machine learning applications.

2. Software description

The purpose of AddLat2D [9,10] is to create a dataset of 2D lattice cells based on a specified image size and list of density values. Where

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: ccm@cmu.edu (C. McComb).

https://doi.org/10.1016/j.simpa.2023.100567
Received 5 July 2023; Received in revised form 3 August 2023; Accepted 5 August 2023

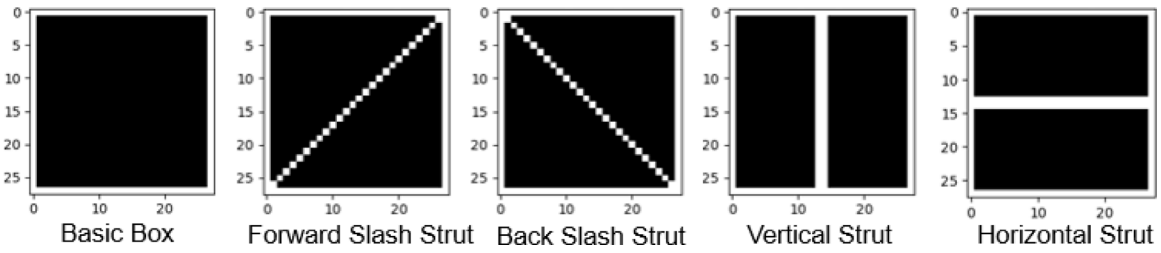


Fig. 1. Main strut types.

Fig. 2. Hugging Face: Image size selection slider.

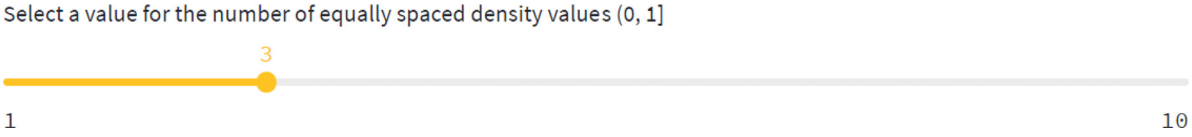


Fig. 3. Hugging Face: Density values selection slider.

the image size defines the pixel width and height of each image, and the density values define the pixel intensity of each activated pixel in the unit cell. The final output is a CSV file with columns containing the arrays of the images, the 'Density' value, and the respective 'Thickness' values that define the shape, and a visualization of the data distribution. This section will outline the possible methods for accomplishing this data generation, using a Python script or a Hugging Face application.

2.1. Python script

This section outlines the main functions that exist within the backend Python script for this software [9]. The Python script is comprised of two key components for data generation: the data generation function and the piecewise functions that define the main shapes present in the dataset (see Fig. 1). These piecewise functions consist of the mathematical representations of individual line components seen in Fig. 1, which constitute the building blocks of the dataset. Finally, the data generation function is created by systematically combining the piecewise functions.

During the data generation process, the data generation function operates by cycling through each of the piecewise functions. For each piecewise function, the 'Thickness' and 'Density' parameters are applied, allowing customization of the strut's thickness and density attributes. This iterative process creates a factorial dataset, as it contains all combinations of shapes. This approach provides flexibility in defining the characteristics of the dataset, enabling users to tailor the generated data to suit their specific research needs. To begin the data generation process, the user must provide the image size and a list of density values to the function. The function will then output the CSV file as described previously.

One notable advantage of the script's design is its flexibility in incorporating additional shapes. Users can easily define new shapes they wish to include by adding them to the set of piecewise functions. Subsequently, the data generation function must be modified to incorporate the newly defined shape. This intentional design choice enables users to expand the applicability of the software when developing datasets, granting them the freedom to include a diverse range of custom shapes according to their research requirements.

Once the dataset has been generated, the user can choose to apply a t-distributed neighbor embedding function on the dataset to create a visualization of the distribution of the dataset. An example of how to execute this is incorporated in the code, but it is not a feature this work focused on exclusively.

The Python script implementation facilitates the development of datasets encompassing various strut-based shapes, while also allowing users to extend its functionality by incorporating their own shapes. This feature broadens the script's potential for application among different users, enabling them to create datasets that suit their specific research needs and contribute to the advancement of their respective fields.

2.2. Hugging Face application

To enhance accessibility for users who may not be familiar with Python, a user-friendly graphical user interface has been developed using Hugging Face, expanding the software's availability to a broader audience of researchers and practitioners[1] [10]. This section outlines how to utilize this software for future research. To begin the data generation process, the user should select the image size using the slider bar (see Fig. 2). The image size defines the height and width of the data points generated and has a maximum value of 16 to prevent the application from exceeding its computational restrictions on Hugging Face.

Next the user should determine the number of density values they would like their dataset to include for the pixel intensity of the activated data points (see Fig. 3). These values are equally spaced from 0 (exclusive) to 1 (inclusive).

To visualize the current types of data being produced, the user can test the 'Generate Samples' button (see Fig. 4). This feature should assist users to better understand how the various settings affect their data. For example, the values for density are displayed above the figures in Fig. 4, which emphasizes that 0 is not included in the dataset.

[1] This tool can be found at https://huggingface.co/spaces/cmudrc/AddLat2D.

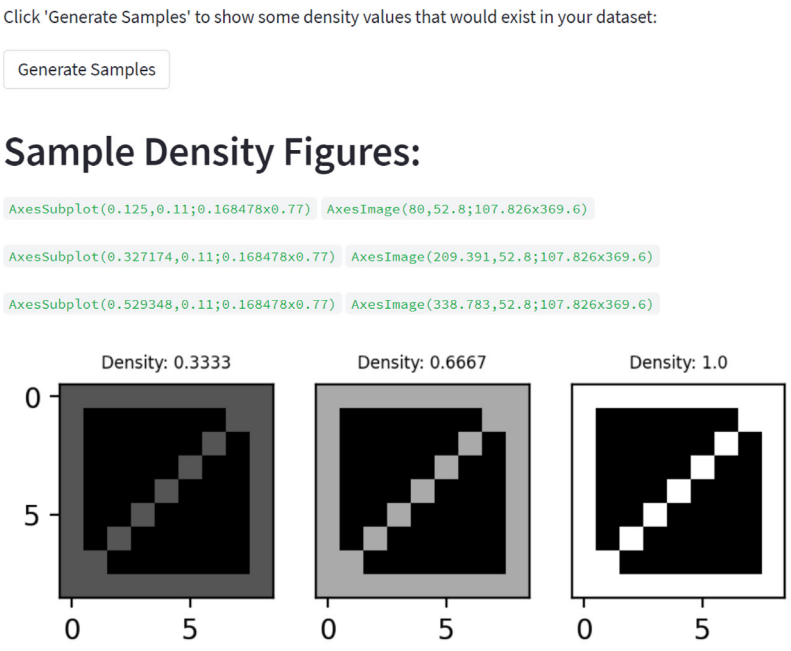


Fig. 4. Hugging Face: Sample figures.

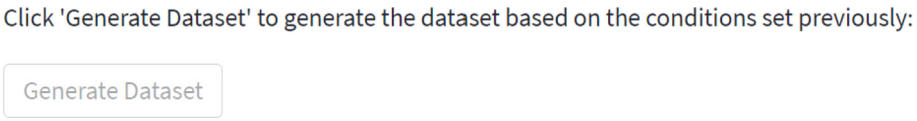


Fig. 5. Hugging Face: 'Generate Dataset' button.

If the user is satisfied with the type of data generated, then they can begin the data generation process by selecting 'Generate Dataset' (see Fig. 5).

The application will display two figures after the dataset is generated (see Figs. 6 and 9). See the section 'Data Produced' below for further discussion on the resulting figures. Following the generation of the data, the option to download the data is available (see Fig. 6), which will automatically download a CSV file of the entire generated dataset.

The implementation using Hugging Face will enable users with a variety of coding experiences and backgrounds to generate data for various needs. Additionally, using a web-based platform to support the software ensures compatibility on a variety of devices. This application has already eased the data generation process for prior works that utilized similar data generation techniques [7,8].

2.3. Data produced

This section will outline how to interpret the data produced, regardless of whether the user chooses to use the Python script or the Hugging Face application. The result is a dataset with 6 labels, and a visualization of the data using a t-distributed neighbor embedding.

As discussed, the output of the script and software is a CSV file containing data points with 6 labels (see Figs. 6 and 7). The labels in Fig. 7 are further explained by Table 1, but is also discussed in this section. The 'Array' is a combination of all the strut types (see Fig. 1)

Here is what a portion of the generated data looks like (double click on the 'Array' cells to view the full array):

	Array	Density	Basic Box Thickness	Forward Slash Strut Thickness	Back Slash Strut Thickness	Vertical Strut Thickness	Horizontal Strut Thickness
0	[[0.3333333333333333, 0.3	0.3333333	1	0	0	0	0
1	[[0.3333333333333333, 0.3	0.3333333	1	0	1	0	0
2	[[0.3333333333333333, 0.3	0.3333333	1	0	2	0	0
3	[[0.3333333333333333, 0.3	0.3333333	1	0	3	0	0
4	[[0.3333333333333333, 0.3	0.3333333	1	0	4	0	0
5	[[0.3333333333333333, 0.3	0.3333333	1	0	5	0	0
6	[[0.3333333333333333, 0.3	0.3333333	1	1	0	0	0
7	[[0.3333333333333333, 0.3	0.3333333	1	1	1	0	0

Click 'Download' to download a CSV file of the dataset:

Download Dataset

Fig. 6. Hugging Face: Sample of generated data and the 'Download Dataset' button.

	Array	Density	Basic Box Thickness	Forward Slash Strut Thickness	Back Slash Strut Thickness	Vertical Strut Thickness	Horizontal Strut Thickness
0	[[0.2, 0.2, 0.2, 0.2, 0.2,	0.2	1	0	0	0	0
1	[[0.2, 0.2, 0.2, 0.2, 0.2,	0.2	1	0	1	0	0
2	[[0.2, 0.2, 0.2, 0.2, 0.2,	0.2	1	0	2	0	0
3	[[0.2, 0.2, 0.2, 0.2, 0.2,	0.2	1	0	3	0	0
4	[[0.2, 0.2, 0.2, 0.2, 0.2,	0.2	1	0	4	0	0

Fig. 7. Sample of raw data.

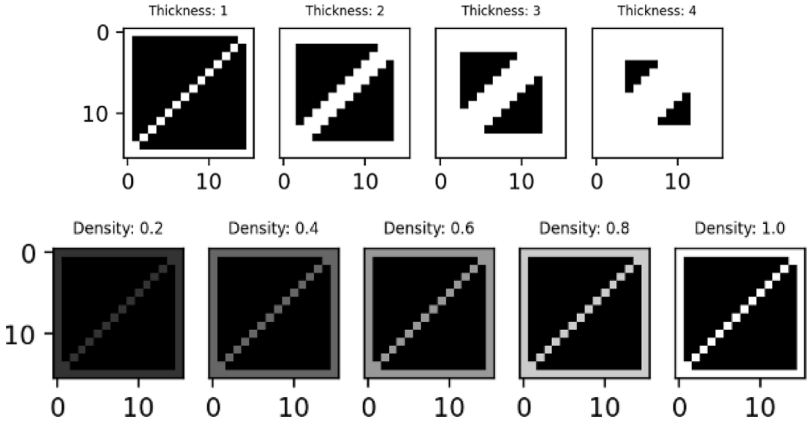
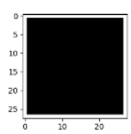
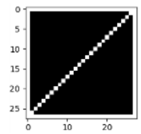
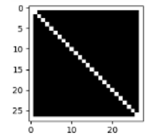
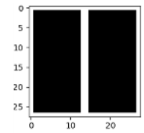
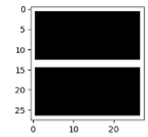


Fig. 8. Sample shape with possible thickness (top) and density (bottom) variations.

that are defined using a combination of 'Density' and 'Thickness', this is the set of unique data ponts generated by the application. The 'Density' represents the value of each activated pixel in the base structure, which the user can define as a list of values between (0,1] (see Fig. 8). The 'Thickness' describes the number of pixel layers in the existing base structure (see Fig. 8). Fig. 8 displays the arrays as the 'Thickness' of the forward slash strut and the basic box have equal thickness values in each step. A 'Thickness' of 0 represents the lack of that strut, and 1 represents a single pixel thickness which is displayed in the set of base images (see Fig. 1).

After generating the dataset, users can gain insights into the distribution of the data by utilizing the data plotting code. This code enables quick visualization of the generated dataset by employing t-distributed neighbor embedding, a dimensionality reduction technique that projects the data into two dimensions. By visualizing the data in this reduced dimensionality space, users can better grasp the diversity and patterns within the dataset. The resulting plot provides a visual representation where each image in the dataset is depicted as a point, color-coded based on its average pixel intensity. By observing the distribution of points and their corresponding color patterns, users can gain valuable insights into the characteristics and variations present in the dataset (see Fig. 9). This visual exploration aids in data analysis and supports decision-making processes when generating subsequent datasets or designing downstream tasks and experiments.

Table 1
Description of data.

Array	Density	Basic box thickness	Forward slash strut thickness	Back slash strut thickness	Vertical strut thickness	Horizontal strut thickness
Square arrays	Positive real number					
The other variables serve to define the parameters of the output of this array.	The values of the activated pixels in the array, ranged from (0,1].	Integer values that define the thickness of the shape, where 1 is the example value shown above.				
Example: [[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]]	1.0	1	0	0	0	0

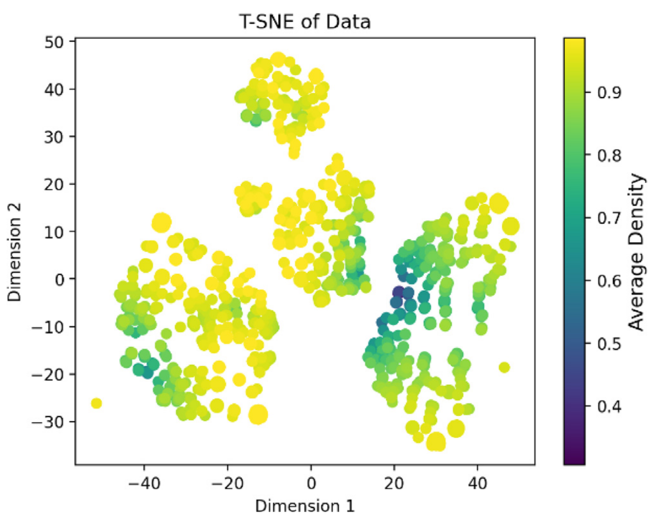


Fig. 9. Example of a T-distributed Neighbor Embedding applied to data.

In summary, the software provides users with the capability to generate a customizable dataset of 2D lattice cells. Users can define various parameters, such as image size and density, to create a unique dataset of strut-based shapes. Additionally, the software offers functionality to analyze the distribution of the generated data through visualization techniques, employing t-distributed neighbor embedding for dimensionality reduction and visual representation.

3. Impact overview

This research presents a data generation software, AddLat2D, for creating 2D lattice structure datasets tailored to user specifications. By using our software, users can develop a multitude of related datasets by changing only two data parameters. The datasets generated by this software have been utilized in previous work by the authors [7,8] and hold potential value for use by others.

3.1. Impact of AddLat2D on current multi-lattice research

The software was designed as a secondary objective to prior research in design for additive manufacturing. The authors used a subset of a generated dataset to explore the potential for using variational autoencoders for developing multi-lattice structures [7,8]. The work ultimately proved as motivation to use variational autoencoders for developing multi-lattice transition regions. Additionally, the work proved the presence of potential relationships that exist between geometric smoothness and latent space properties.

3.2. Potential impact of AddLat2D

The software has significant potential to provide flexible datasets for use in machine learning research for additive manufacturing. There is value in being able to generate synthetic data of various sizes and characteristics. For instance, the combination of multiple related simple datasets can be useful for model development and debugging, benchmarking [4], proof of concept development [11], and assessment of overfitting. Additionally, other similar datasets have been used in feature selection [12] and classification algorithms, which demonstrate possibilities for future work.

Model development and debugging is one of the most time-consuming parts of machine learning applications. Using a simple dataset allows researchers to decrease computation time needed to test the architecture of their model by using scaled data. When thousands of operations are being performed on data, it is important to ensure that the code is optimized effectively. This was evident in our previous work, given the small sample size to prove our model architecture was efficient [7,8]. During the development of our model, having a smaller dataset enabled shorter training times in order to optimize the model architecture. Given that the dataset can be modified in size without significant changes to the data, the datasets generated could be extremely useful for model development.

Benchmarking is a widely accepted practice in the field of computer science, which involves the systematic evaluation and comparison of algorithms, models, or systems using standardized datasets and metrics [13]. For datasets to be effective for benchmarking, they need to contain a subset of the global data domains [4] The dataset generated from this software, with its simplicity and focus on 2D strut shapes within lattice structures, offers an ideal platform for benchmarking studies. Researchers can leverage this dataset to evaluate and compare different algorithms, enabling objective and evidence-based decisions on algorithm selection and improvement strategies. Other binary image datasets have proven extremely popular for benchmarking, including MNIST [14] and Fashion-MNIST [15].

Testing the functionality of new algorithms requires an extremely simple dataset that the user knows extremely well, to interpret the results from the algorithm. For instance, Berthelot et al. used a dataset

of lines to measure the quality of interpolations from their machine learning model [11]. Using lines meant that they could measure the change of the angles between each image to serve as the metric for the machine learning model. Using a similar dataset generated from this software, we determined the scale of data necessary to produce results in prior work [7,8].

Feature selection algorithms could be applied to the dataset to narrow down the important data points that affect training. Feature selection is a process used in machine learning to remove redundant and irrelevant data [16,17]. This allows potential machine learning models to train faster using a smaller equivalent dataset. Since the size of the data can be easily changed, it could be used to measure the performance of the feature selection algorithms. A similar type of dataset was used by Wang et al. to extract critical features from microstructures [12]. Future work could use a smaller dataset to find the optimal feature selection technique for binary image data.

Classification algorithms play a crucial role in identifying labels or categories within a dataset based on specific input parameters [17,18]. The dataset's simplicity and focused nature provide a solid foundation for exploring classification algorithms across various domains. Other similar datasets used for classification applications consist of MNIST [14] and Fashion-MNIST [15], which both utilize of binary images with sets of labels. With a minimum of 5 labels, the datasets generated offer versatility for multi-label prediction tasks in different research applications. By utilizing this software, researchers can accelerate their classification-related research, enhance decision-making processes, and contribute to advancements in their respective fields.

4. Limitations

While the software offers significant benefits, there are certain limitations that warrant consideration for further improvement. Some major areas we have noted include data generation diversity, computational efficiency, and data duplication.

Diverse datasets are important for some applications and these needs may not be addressed solely using this software to generate data. The possible arrays generated were restricted to strut-based shapes for simplicity of development. To address this, future work could involve incorporating additional shape types and employing data augmentation techniques to expand the diversity of the data domain. Notably, for applications in additive manufacturing, alternative approaches have demonstrated advantages in terms of dataset diversity [5].

Computational efficiency is an important aspect of data generation tools to avoid wasting resources. The code for generating the dataset was not fully optimized, therefore, it may be time consuming to generate datasets with large image sizes and many density values. This had a significant impact on the options available through the GUI on Hugging Face. The Hugging Face application is limited due to the computational restrictions imposed on users. Specifically, the size of the unit cells and density options are restricted in the Hugging Face implementation of the software, thereby limiting larger dataset generation to users with a Python background.

Finally, the software inherently contains duplicates within the dataset to ensure retention of all generated labels. While this duplication facilitates indexing of every unit cell, it can lead to overfitting if substantial duplicate data is present. Nevertheless, the presence of duplicates becomes necessary for indexing purposes, as some unit cells may only be indexable through specific combinations of thickness values.

Declaration of competing interest

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.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT [19] in order to improve quality of writing. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Acknowledgments

This paper is based on work that was recently accepted for publication at the Solid Freeform Fabrication Symposium [8]. This material is based upon work supported by the National Science Foundation, United States through Grant No. CMMI-1825535. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2023.100567.

References

[1] P. Yadav, O. Rigo, C. Arvieu, E. Le Guen, E. Lacoste, In Situ monitoring systems of the SLM process: On the need to develop machine learning models for data processing, Crystals (Basel) (2020) http://dx.doi.org/10.3390/cryst10060524.

[2] B. Hanks, J. Berthel, M. Frecker, T.W. Simpson, Mechanical properties of additively manufactured metal lattice structures: Data review and design interface, Addit. Manuf. 35 (2020) http://dx.doi.org/10.1016/J.ADDMA.2020.101301.

[3] J. Chen, M. Patterson, A.M. Mirzendehdel, M. Behandish, Automatic shape modification for self-supporting structures in additive manufacturing, in: Proceedings of the ASME Design Engineering Technical Conference. 3-A, 2022, http://dx.doi.org/10.1115/DETC2022-89054.

[4] G. Williams, N.A. Meisel, T.W. Simpson, C. McComb, Deriving metamodels to relate machine learning quality to design repository characteristics in the context of additive manufacturing, in: IDETC/CIE, 2020, http://dx.doi.org/10.1115/1.4044199.

[5] Y.-C. Chan, F. Ahmed, L. Wang, W. Chen, METASET: Exploring shape and property spaces for data-driven metamaterials design, J. Mech. Design 143 (2021) http://dx.doi.org/10.1115/1.4048629.

[6] D. Lee, Y.-C. Chan, W.W. Chen, L. Wang, A. van Beek, W. Chen, t-METASET: Task-aware acquisition of metamaterial datasets through diversity-based active learning, 2022, http://dx.doi.org/10.1115/1.4055925.

[7] M. Baldwin, N.A. Meisel, C. McComb, Smoothing the rough edges: Evaluating automatically generated multi-lattice transitions, 3D Print. Addit. Manuf. (2023) http://dx.doi.org/10.1089/3dp.2023.0008, ahead of print.

[8] M. Baldwin, N.A. Meisel, C. McComb, A data-driven approach for multi-lattice transitions, in: Solid Freeform Fabrication Symposium, 2022, http://dx.doi.org/10.26153/tsw/44429.

[9] Design research collective, AddLat2D, 2023, http://dx.doi.org/10.57967/hf/0683.

[10] Design research collective, AddLat2D hugging face space, 2023, https://huggingface.co/spaces/cmudrc/AddLat2D. (Accessed 6 June 2023).

[11] D. Berthelot, C. Raffel, A. Roy, I. Goodfellow, Understanding and improving interpolation in autoencoders via an adversarial regularizer, in: ICLR, 2019, http://dx.doi.org/10.48550/arXiv.1807.07543.

[12] L. Wang, Y.-C. Chan, F. Ahmed, Z. Liu, P. Zhu, W. Chen, Deep generative modeling for mechanistic-based learning and design of metamaterial systems, Comput. Methods Appl. Mech. Engrg. 372 (2020) http://dx.doi.org/10.1016/J.CMA.2020.113377.

[13] R.S. Olson, W. La Cava, P. Orzechowski, R.J. Urbanowicz, J.H. Moore, PMLB: a large benchmark suite for machine learning evaluation and comparison, BioData Min. 10 (2017) 36, http://dx.doi.org/10.1186/s13040-017-0154-4.

[14] L. Deng, The MNIST database of handwritten digit images for machine learning research, IEEE Signal Process. Mag. 29 (2012) 141–142, http://dx.doi.org/10.1109/MSP.2012.2211477.

[15] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017, http://dx.doi.org/10.48550/arXiv.1708.07747.

[16] J. Cai, J. Luo, S. Wang, S. Yang, Feature selection in machine learning: A new perspective, Neurocomputing. 300 (2018) 70–79, http://dx.doi.org/10.1016/j.neucom.2017.11.077.

[17] J. Chen, H.T. Ilies, C. Ding, Graph-based shape analysis for heterogeneous geometric datasets: Similarity, retrieval and substructure matching, Comput. Aided Des. 143 (2022) 103125, http://dx.doi.org/10.1016/J.CAD.2021.103125.

[18] G. Williams, N.A. Meisel, T.W. Simpson, C. Mccomb, Design repository effectiveness for 3D convolutional neural networks: Application to additive manufacturing, Mech. Design (2019) http://dx.doi.org/10.1115/1.4044199.

[19] OpenAI, ChatGPT, 2023, https://chat.openai.com/chat. (Accessed 7 June 2023).