BIKED: A Dataset for Computational Bicycle Design with Machine Learning Benchmarks

Lyle Regenwetter
Dept. of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA, 02139
Email: regenwet@mit.edu

Brent Curry
BikeCAD.ca
Ottawa, Ontario, Canada
K1J 6E9
Email: info@bikecad.ca

Faez Ahmed
Dept. of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA, 02139
Email: faez@mit.edu

In this paper, we present "BIKED," a dataset comprised of 4500 individually designed bicycle models sourced from hundreds of designers. We expect BIKED to enable a variety of data-driven design applications for bicycles and support the development of data-driven design methods. The dataset is comprised of a variety of design information including assembly images, component images, numerical design parameters, and class labels. In this paper, we first discuss the processing of the dataset, then highlight some prominent research questions that BIKED can help address. Of these questions, we further explore the following in detail: 1) How can we explore, understand, and visualize the current design space of bicycles and utilize this information? We apply unsupervised embedding methods to study the design space and identify key takeaways from this analysis. 2) When designing bikes using algorithms, under what conditions can machines understand the design of a given bike? We train a multitude of classifiers to understand designs, then examine the behavior of these classifiers through confusion matrices and permutation-based interpretability analysis. 3) Can machines learn to synthesize new bicycle designs by studying existing ones? We test Variational Autoencoders on random generation, interpolation, and extrapolation tasks after training on BIKED data. The dataset and code are available at `http://decode.mit.edu/projects/biked/`

1 Introduction

Rapid advancements in machine learning research fields are often catalyzed by the introduction of quality publicly available datasets. Open source datasets often serve as benchmarks to evaluate the performance of different algorithms on a level playing field. In computer vision, for example, a set of well-known datasets (Imagenet [1], CIFAR-10 [2], MNIST [3], *etc.*) serve as the standard benchmarks for evaluating new techniques and methods and determining the state of the art. Data-driven design is a growing field that has increasingly tapped into machine learning and other data-driven methods to realize design goals, including automated design. New methods in data-driven design research are often tested on existing datasets like 3D model datasets and topology optimization (TO) datasets, however, these have drawbacks that limit the scope of their applications. We introduce the BIKED dataset with the primary intent of supporting new data-driven design methods both in general and in bicycle design specifically. In particular, we support the development of methods that can leverage detailed parametric design information, such as explicit dimensions and component categories.

Data-driven methods have the potential to enhance bicycle design exploration and generation, just as they have in other fields. We found the bicycle (bike) design problem to be an ideal domain for the testing and development of data-driven methods. Bicycles are machines with hierarchies of subcomponents, complex part dependencies, and large variations between designs. This variation largely stems from the broad set of use cases (road races, stunts, cargo transportation), physical considerations of the rider (body dimensions, flexibility, posture), and human preferences (style, aesthetics, budget). Due to this diverse set of constraints, bikes greatly benefit from customization. This makes the bicycle an excellent candidate for data-driven design, which typically leverages the design expertise contained in previous products, studies the diversity of existing designs, and captures the statistical trends of human preference. Compared to a manual design process, which is common in the bicycle industry, data-driven approaches can accelerate new bike design tasks and lead to novel and high-performing designs.

Significant advancements to the bicycle design field also have the potential to positively impact society. As of 2015, there were estimated to be at least 580 million privately owned bikes worldwide [4]. Despite the continued popularity of bicycles, further increasing bicycle use and ownership can improve public health [5], reduce traffic congestion [6], and positively impact the environment [7]. We hope that by developing new data-driven bicycle design methods, advancements in the field of bicycle design can be realized and bicycles can subsequently be made more accessible to more people. This impact will be especially pronounced if data-driven design methods can enable increased customizability for particular user groups for whom bicycles may not presently be a competitive transportation option.

BIKED—a dataset of bicycle models–is organized and curated specifically for data-driven design of bicycles[1]. Data is sourced from a rich archive of CAD files from the specialized BikeCAD software, which is primarily used by professional frame designers and bicycle enthusiasts. A sequential data curation process was carried out as detailed in Section 3 to extract parametric design data. As we demonstrate in Section 4.1, this parametric data captures the rich variations between designs in the dataset and is highly revealing of many parameters of interest like bicycle type.

While we present the dataset itself as the key contribution of this work, we also outline and explore several key research questions that BIKED may address: How can we explore, understand, and visualize the current design space of bicycles and utilize this information? When designing bikes using algorithms, under what conditions can machines understand the design of a given bike? Can machines learn to synthesize new bicycle designs by studying existing ones? We dedicate the latter portion of this paper to exploratory work on these research questions.

In Section 4.1 we consider the question of how we can explore, understand, and visualize the current design space of bicycles and furthermore utilize this information. We approach this problem through unsupervised embedding methods. These methods can generate easy-to-visualize low-dimensional maps of the design space. By studying the distribution of bicycles across the space, especially with bicycle class annotations, we can understand trends in the data such as what classes tend to overlap or distance from one another. By identifying sparse regions in the design space, we can potentially even identify design opportunities that are underrepresented in the current bicycle industry.

In Section 4.2 we explore the conditions necessary for machines to understand the design of a given bike. We formulate a classification problem to evaluate this question. Specifically, we investigate what quantity of data, type of data, and algorithm yields high classification performance. To do so, we evaluate and provide baseline results for 12 classification algorithms on several training set sizes. We then construct, tune, and train three deep neural networks to predict class results based on different types of input data provided within BIKED. These neural networks provide a baseline for higher performance models and yield valuable insight into the predictive information contained within the parametric and image data. In addition to maximizing performance, practitioners must understand the patterns identified by machine learning. We examine the behavior of the classifiers by generating and analyzing confusion matrices. Furthermore, to understand what design attributes are important for each bike style, we pose this problem as an interpretability analysis problem for a classification model. Using Shapley Additive Explanations [8], we explore how individual parameters impact class predictions and discuss the design implications of our results.

Finally, in Section 4.3, we address the last of our three big questions: When designing bikes using algorithms, can machines create a unique bicycle design that they have not seen before? We present applications and baseline results of two Variational Autoencoders (VAEs) applied to the parametric and image data respectively. VAEs support a host of applications including dimensionality reduction and novel data synthesis. We explore several types of data synthesis: random sampling (exploring the design space), interpolation (synthesizing new designs between two existing designs), and extrapolation (synthesizing new designs away from an existing design) in both the original parameter space and the latent space of the VAEs.

Inspiring other researchers to use standardized problems and datasets for data-driven design applications and algorithm development is one of the key goals of BIKED. To that end, we provide a suite of algorithms (Variational Autoencoders, convolutional neural networks, Gaussian processes, ensemble classifiers, *etc.*) and their performance values throughout our exploration of the aforementioned research questions.

2 Background and Previous Work

In this section, we discuss the background and related work in bicycle design optimization, compare BIKED to other datasets commonly used for data-driven design, and discuss the BikeCAD software and design archive. We also note that this paper is primarily an expansion of previous work [9].

2.1 Bicycle Design and Optimization

Bicycle design and optimization is a well-researched field. Numerous treatises have explored principles of bicycle design in the centuries since the earliest predecessors to the modern bicycle [10, 11]. Today, significant research effort is dedicated to improving bicycle aerodynamics [12–14] and structure [15]. Other studies explore practices of bicycle sizing and fitting [16, 17]. While some make use of the wide availability of anthropometric data, many are limited by the availability of bicycle design data. We expect that a quality dataset of bike designs including comprehensive parametric information will enhance future simulation-based bicycle design studies.

[1]Dataset and code available at: `http://decode.mit.edu/projects/biked/`

2.2 Comparison to Other Datasets

Research in this domain of data-driven design often taps into 3D model datasets like Shapenet [18] and Princeton ModelNet [19]. These datasets are particularly useful in 3D shape synthesis applications. Though some of this shape synthesis research considers adherence to physical laws, the nature of these datasets tends to emphasize research focused on visual fidelity rather than physical function [20–23]. In contrast, BIKED contains thousands of design parameters that explicitly specify design information ranging from local component geometry to overall design layout. Certain design information can be extracted from 3D models, but the level of detail falls short of the comprehensive numerical design parameters that BIKED provides.

Data-driven design research frequently uses Topology Optimization (TO)-based approaches, which often depend on TO datasets. While TO datasets can provide extensive information for models learning to optimize local geometry [24, 25], approaching full design synthesis tasks using TO is often impractical, and generalizing TO datasets to new design domains poses an additional challenge. As such, BIKED enables methods and functionality that would not be feasible to develop using TO datasets.

2.3 BikeCAD

BikeCAD is a parametric computer-aided design (CAD) software optimized for bicycle design. It features a live-updating model and numerous design menus that help users customize bicycle geometry and features, as shown in Figure 1. The BikeCAD website features an open archive of user-submitted BikeCAD designs [26]. As of July 2020, the archive had accrued roughly 5000 bicycle designs since its inception in 2011. Figure 2 shows tile images of several randomly sampled bikes from the dataset. Since designs span many iterations of the BikeCAD software, the advancement of the program over the years has enabled increased complexity in more recent designs on the archive. Each entry in the dataset contains the BikeCAD file, an image, a rating out of 5, the BikeCAD version used to create the design, and a design identification number (ID). Additionally, some designs in the archive contain the ID that they are based on and several parametric dimensions. Of this information on the archive, only the BikeCAD files themselves were used in BIKED. Rating information was not included since most designs are unrated and few designs have more than a single rating. Additionally, though tracking the design IDs that each design is based on would make for an interesting graph of design progression, many design ID links are no longer functional since their referenced designs have been removed from the archive.

Since BikeCAD's primary user base is comprised of frame builders and bicycle enthusiasts, many designs are high-quality and are furthermore self-selected as showcase designs. We also note that the BikeCAD software promotes quality through realistic default values, comprehensive design and analysis tools such as toe overlap and lean angle analysis, as well as a suite of tools to check the design's adherence to Union Cycliste Internationale (UCI) standards. All in all, this combination of factors suggests a reasonably high quality for the majority of dataset designs.

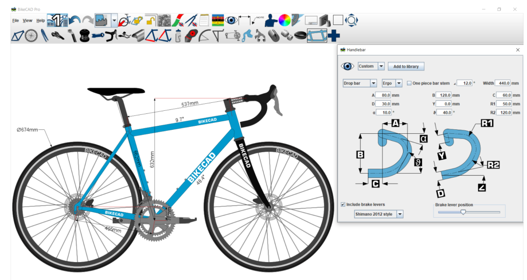


Fig. 1: Screenshot from BikeCAD software showing active model with some dimensional labels and an open menu.

Fig. 2: Tile images of sample bike designs. Backgrounds and dimensional labels are removed for visibility.

3 Methodology

In this section, we discuss the various steps taken to generate and process the dataset. An overall flow diagram of the data curation progression is shown in Figure 3 and the following labeled sections correspond to the processing operations listed.

3.1 Sourcing from Design Archive

To process the dataset locally, we downloaded designs from the BikeCAD design archive. We were able to download 4800 of the designs posted on the archive as of July 2020. Original ordering on the design archive was preserved in the subsequent processing of the data.

3.2 File Standardization in BikeCAD

BikeCAD files are XML files structured as a collection of data parameters with corresponding values. Certain parameters have different meanings depending on flags in other parts of the file. To ensure that the parameters in the files referred to the same geometric dimensions across different designs, BikeCAD's internal conversion formulas were applied using a custom version of the BikeCAD software developed by one of the authors. 9 files were corrupt or unprocessable leaving 4791 successfully standardized designs.

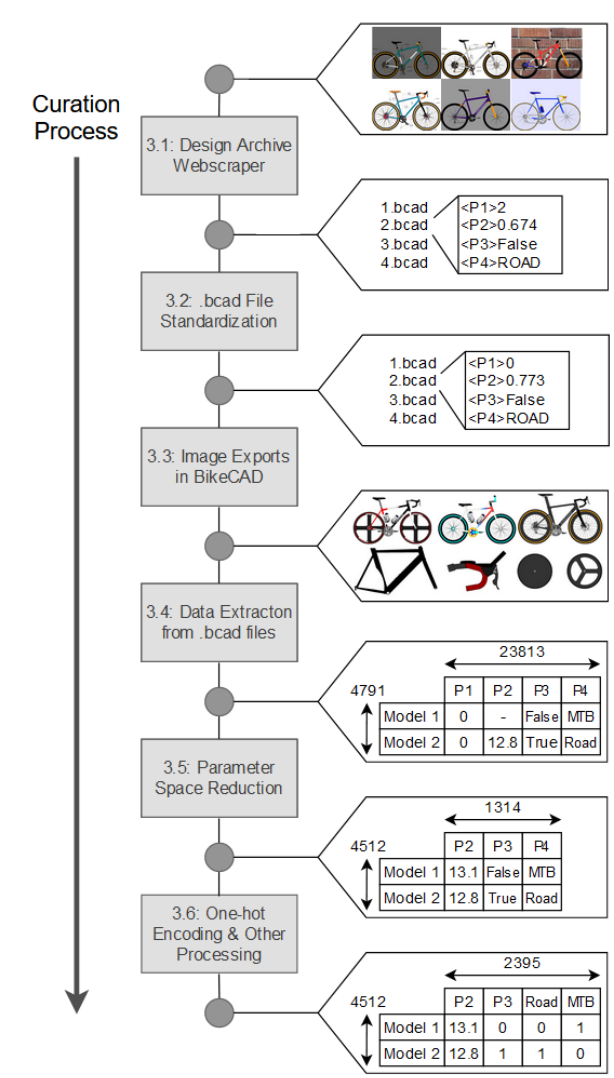
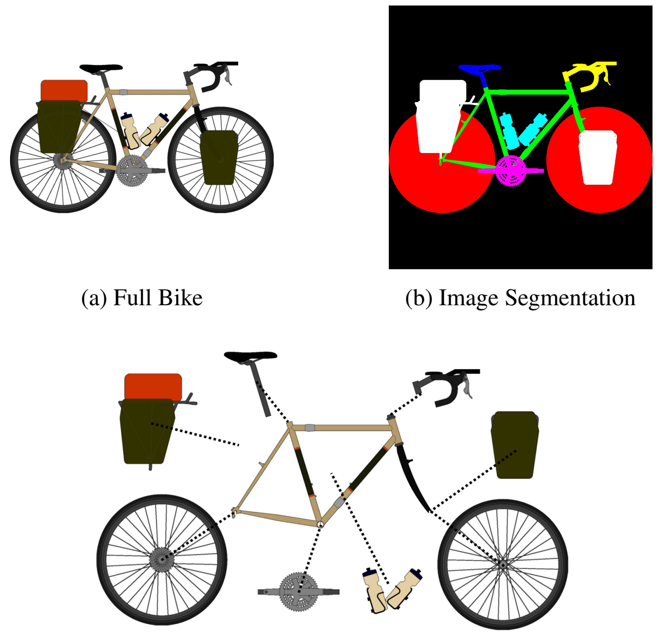


Fig. 3: Flow diagram of dataset curation process. Processing steps (gray boxes) are labeled according to the section in which they are discussed. Data snapshots (right) demonstrate sample data generated by the previous processing step.

3.3 Bicycle Assembly and Component Image Exports

To remove dimensioning lines and backgrounds from the bike models' corresponding images, we directly edited label visibility and set plain backgrounds in the BikeCAD files, then exported these clean bike images. Next, segmented component images were rendered for 5 essential components (frame, saddle, handlebars, wheels, cranks) and two nonessential components (cargo racks, bottles) that only appeared in some designs. BikeCAD models are structured as a collection of component parts that form an assembly. Each component image was rendered and exported from BikeCAD after editing the BikeCAD files to hide all other components. 4510 bikes were compatible with this process, yielding 5 sets of 4510 component images (essential components), one set of 965 (bottles), and one set of 346 (racks). An example segmentation on one bike design is demonstrated in Figure 4. The components in the segmented



(a) Full Bike

(b) Image Segmentation

(c) Exploded View

Fig. 4: Segmentation of a bicycle into 7 component parts. Figure 4a shows the bike assembly image and Figure 4b shows a semantic mask of its segmentation into components. Figure 4c demonstrates an exploded view showing the 7 component groups. Observe that portions of the bike that were covered in the original image are present in the component images, such as the portion of the wheels that were covered by the stem, chainstays, and racks. Exploded views are not included in the dataset.

images maintain the same position and scale as the original assembly images. In this sense, the segmented images can be interpreted as a "layering" of the full assembly image into bicycle component image slices. Semantic masks of the segmentation are also included.

3.4 Data Extraction from BikeCAD Files

For ease of processing, we parsed the individual standardized BikeCAD files and compiled all parameter data in a tabular data structure. In total, 23818 unique parameters (or features) were collected from the files, though many parameters were quite sparse across the design space since certain designs had no entries for many parameters.

3.5 Parameter Space Reduction

Lowering the dimensionality of a dataset can help reduce computational cost and memory requirements when applying data-driven models. Given that our original data was 23818 dimensional, dimensionality reduction was critical for efficient data-driven analysis. To reduce the dimensionality of the parameter space, several parameters were removed from consideration. First, parameters with no bearing on bicycle function such as color, text, and positioning of dimensional labels were identified and dropped from the data. Next, parameters defining additional non-standard tubes, water bottles, or tandem features were removed, and any bike with a data value under one of these fields was dropped. This removed some unique designs but drastically reduced the dimensionality of the data by dropping sparsely inhabited parameters. Additionally, each design was manually considered and any designs that were not bikes (scooters, motorcycles, wheelchairs, *etc.*) were dropped. Finally, any parameters that were empty for every design or for which every design had the same value were dropped. After these steps, 4512 bikes of 4791 and 1314 parameters of 23813 remained, resulting in a 94.5% reduction in dimensionality. We note that these 4512 designs constitute a different subset of the original 4800 than the 4510 designs compatible with the segmentation process.

Approximately 5% of designs lacked one or more essential components, which in practice implied that the designer turned off the visibility of said components in the software. For example, some designs originally had only a frame visible, presumably implying the designer was only intending to design a frame. Since we did not want incomplete designs in the dataset, critical components of bike designs were used in the processing of the dataset regardless of whether they were hidden or not. In other words, visibility parameters of critical components were dropped from the parameter space. In cases like the standalone bike frame designs, it was unclear if the hidden components like wheels, saddle, and handlebars received any design consideration, but were included nonetheless. As such, some designs may include components not intended to be part of a final design. In contrast, non-essential features like bottle holders and fenders were included or excluded in the parameter space based on the visibility in the original design. We acknowledge that manually overriding component visibility of critical design components could have unintentionally included components that were not intended to be part of the designer's vision, however, the number of affected designs was minimal. In these rare cases, the bicycle model could have one or more components whose parameters were not adjusted from the defaults or were set in an earlier draft of the model.

3.6 Other Processing Steps

Following the parameter space reduction, the parameter space consisted of four types of data: continuous, discrete, boolean, and categorical. Continuous and discrete parameters reflect some ordinal significance (i.e. similar numerical values imply actual real-world similarity in that parameter). In contrast, categorical values such as numerical class labels imply no such ordinal significance even if represented numerically. Some example parameters of each type are included in Appendix A1, and we refer the reader to the project page for a full list. Most data types in the parameter space could be easily classified (i.e. floating-point numbers are continuous). However, integer-valued parameters had to be individually considered and manually classified as categorical or discrete. Additionally, certain variables that would ideally be continuous had to instead be treated as categorical. For example, the bike "size" parameter was treated as categorical since it contained a medley of words and measurements with varied or ambiguous units ("xxl","623EET","Huge","52cm","26", *etc.*). Since many standard machine learning methods do not support categorical data, we applied a commonly used method called one-hot encoding, which converts a categorical variable with n unique categories into a set of n boolean parameters. These steps left us with the distribution of variables shown in Table 1.

Table 1: Assortment of types of parameters present in dataset before and after one-hot encoding

	Original		One-Hot	
Parameter Type	**Num.**	**Perc.**	**Num.**	**Perc.**
Cont. Variables	635	48.3%	635	26.5%
Discrete Variables	424	32.3%	424	17.7%
Booleans	148	11.3%	1336	55.8%
Catg. Variables	107	8.1%	0	0%

Next, missing and unknown values were imputed using k-nearest-neighbors imputation with $k = 5$ neighbors.[2] Values with a magnitude above a selected threshold were also discarded and then considered to be missing, as they were likely caused by corrupted data or data type issues. A cutoff magnitude of 100,000 was selected since most units were in millimeters or inches, and bikes were unlikely to be designed with component dimensions over 100 meters. Other outlier removal methods such as z-score cutoffs were examined and found to exclude quality designs without any discernible corrupted data. The final imputed data is the version of the data used in the discussion and sample applications discussed henceforth.

3.7 Image Regeneration

Because parametric data is difficult to quickly interpret, we developed a method to regenerate images from parametric data, as shown in Figure 5. We use this method to generate corresponding images from the fully processed parameter data. This method also applies to newly generated parametric data, which can assist in visualizing interpolation or novel bike generation results as we demonstrate in Section 4.3. In this process, the one-hot encoding is reversed by taking the most probable category to be the absolute truth. In the original dataset, one-hot vectors are populated with boolean values, but the method also supports probabilistic values to better accommodate generative methods. Next, bike data is inserted into a BikeCAD file template to generate new BikeCAD files from the standardized parametric data. Any fields from the template file that are present in the parametric data are overwritten and any absent fields are left at default values. These BikeCAD files are then opened in the

[2] Other imputation strategies such as mean imputation and median imputation are also provided as options in the associated code, but a detailed investigation contrasting imputation methods and hyperparameters was not performed in this study.

software to export corresponding images. Since certain features like component colorings were dropped as described in Section 3.5, this regeneration process yields images with standardized colors as dictated by the template file.

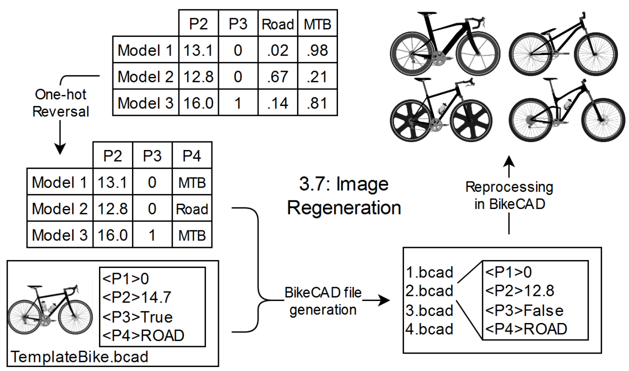


Fig. 5: Process to generate images from parametric data

3.8 Dataset Feature Summary

The data curation process yields several types of data at various levels of processing. The fully processed data consists of tabular parametric data after the processing steps detailed above and standardized images regenerated from curated parametric data. We also provide minimally processed data to support unforeseen applications that may require specialized curation steps. This data consists of raw tabular parametric data sourced directly from BikeCAD files with no processing and original bicycle images after background and dimensional label removal. Finally, we provide segmented bicycle component images to support component-wise learning tasks on images. We also note that the tabular data contains several parameters that can be used as labels for different learning tasks, such as bike class, model year, or individual component types. Note that these labels are neither rigorous nor standardized since all values are reported under the individual designers' discretion. For instructions on usage, we again refer the reader to the BIKED's project page, which links to a comprehensive usage guide.

4 Data-driven Design Applications

In this section, we consider three main research questions: How can we explore, understand, and visualize the current design space of bicycles and utilize this information? When designing bikes using algorithms, under what conditions can machines understand the design of a given bike? Can machines learn to synthesize new bicycle designs by studying existing ones? We address these research questions one by one in the following three subsections. While addressing these questions, we simultaneously hope to accomplish three goals:

1. Provide a starting point and inspiration for researchers interested in using BIKED to expand upon one of these research questions or pursue new research directions supported by BIKED.
2. Give baseline performance values for a variety of common data-driven tasks.
3. Uncover patterns, relationships, and trends within the various forms of data included in BIKED.

Discussion of other research questions that BIKED may help address are discussed in Section 7.

4.1 Bicycle Design Space Exploration and Visualization

Extensive design data can open doors to better understand, visualize, and explore a design domain. We consider methods to uncover these insights and discuss how to furthermore utilize this information in making design decisions. By visualizing the relationships between different classes of bicycles, designers can identify trends, patterns, and potential gaps in the design space. To meet a broader set of customer needs, new classes like hybrid bikes are often introduced. Understanding what new classes can be introduced in the market or what features from one class can be incorporated in another class of bicycle can help designers efficiently target underrepresented segments of the industry.

To explore and visualize the bicycle design space, we use unsupervised embedding algorithms on BIKED. Design space embeddings can allow researchers to visually identify groups of similar designs in a dataset or identify areas of the design space that are sparse or completely empty. Design space embeddings can also be helpful for designers to place their own designs within the context of existing ones.

Bicycle Class Labels: To assist in visualization, we label designs using the bicycle "style" parameter, which we refer to as the bicycle "class." Table 2 shows the distribution of bicycle classes in the dataset. We observe that road bikes are by far the most prevalent class (40.56%) as labeled by the designers. We note that using bicycle "styles" as class labels has several drawbacks. Labels are reported by designers and are subjective and unstandardized. Classes are also not mutually exclusive: bike designs may potentially fit into more than one class, and several of the classes are conventionally considered subclasses of others (e.g. cyclocross, time trial, and touring bikes are subclasses of road bikes) [27]. Despite this fact, each design in BIKED is assigned to exactly one class. Furthermore, several of the classes, like "hybrid" or "other" are not necessarily descriptive of an individual class of bike. Discovering new ways to represent bikes at a high level besides through mutually exclusive classes is a potential research direction that BIKED may support.

T-SNE embedding of full dataset: To visualize the designs, we apply a t-Distributed Stochastic Neighbor Embedding (t-SNE) method [28] to the parameter space of the processed dataset. t-SNE is a dimensionality reduction algorithm that projects samples onto a lower-dimensional embedding space while keeping similar designs together and distancing dissimilar ones. An annotated plot of such an embedding is included in Figure 6. The tendency of classes to

Table 2: Assortment of bike classes present in the processed dataset. Mountain bikes are abbreviated as "MTB." Bikes classified as "OTHER" are labelled as such by the designer. The "Remaining Label Categories" group contains the remaining 10 explicit categories in descending order of prevalence: BMX, CITY, COMMUTER, CRUISER, HYBRID, TRIALS, CARGO, GRAVEL, CHILDRENS, FAT.

Bike Class	**Count**	**Percent**
ROAD	1865	41.33
MTB	616	13.65
TRACK	470	10.42
OTHER	315	6.98
DIRT_JUMP	293	6.49
TOURING	201	4.45
CYCLOCROSS	151	3.35
POLO	128	2.84
TIMETRIAL	89	1.97
Remaining Label Categories	384	8.51

overlap illustrates the aforementioned non-mutually exclusive property of the bike classes and this embedding helps reveal the extent of the overlap and affected classes. We can also identify several unique clusters of bike designs.

PCA embedding of median designs: Next, we consider a more targeted approach by eliminating all parameters except the 50 deemed to be most impactful on bicycle classification. These specific parameters were determined using a SHAP analysis as discussed in Section 4.2. To better visualize the relationship between classes of bikes, we plot only the "median" bike of each class as a representative. We determine this "median" bike by applying min-max normalization to every parameter, taking the median of each normalized parameter over all bikes of that class, and finding the nearest design to this median value by euclidean distance. The results of a Principal Component Analysis (PCA) under these conditions are included in Figure 7. In this plot, we can observe several trends. For example, bikes with components and structure more similar to a conventional road bike tend towards the bottom right corner. Bikes with large wheels relative to absolute size tend toward the top.

We have demonstrated 2-Dimensional embedding examples that are particularly conducive to design space visualization, however, for tasks like identifying sparse regions in the design space, higher dimensional PCA embeddings may be more effective. In combination with the generative process described in Section 3.7, PCA can be a powerful tool. Designers can manually select points of interest in the embedding space, decode latent vectors, then directly generate full bicycle CAD models and images to evaluate these designs. We evaluate a similar generative process in Section 4.3 using the latent space of Variational Autoencoders, albeit using automated sampling instead of manual selection.

4.2 Bicycle Classification

To create meaningful designs, generative machine learning models must identify key characteristics which define a bicycle model and accurately predict which real-world class of design best describes it. In this section, we explore the conditions under which machines can understand the design of a previously unseen bike. Moreover, we examine how intelligent algorithms selectively process data to gain this understanding. To address these questions, we design a classification problem, experiment with various methods, document their performance, and analyze their behavior. In particular, we seek to understand how the algorithm, type of data, and quantity of data impact classification performance, as well as understand the behavior of high-performing classifiers.

Table 3: Classification accuracy for various classification models using differing quantities of training data points. Average scores are determined over 10 instantiations.

Model	**Train Size**				
	200	600	1200	2000	3000
Support Vector Clf.	57.4%	63.4%	66.3%	68.5%	69.6%
K-Neighbors	47.9%	52.3%	55.9%	58.3%	60.3%
Depth 8 Dec. Tree	52.1%	57.9%	61.9%	65.5%	66.5%
Random Forest	**59.0%**	**64.0%**	65.3%	66.5%	66.8%
AdaBoost	48.1%	50.1%	48.9%	49.8%	50.4%
Gaussian Pr. Clf.	47.9%	62.5%	**66.7%**	**69.5%**	**72.0%**
3-layer Neural Net	54.9%	61.6%	63.9%	66.8%	68.7%
6-layer Neural Net	55.3%	59.7%	62.7%	65.5%	68.7%

Baseline Classification Performance: In the first part of this study, we explore how algorithm selection and dataset size impact classification performance on BIKED's parametric data. Table 3 presents an excerpt of a study contrasting 12 untuned models and 5 train set sizes, with each combination instantiated 10 times. We report top and average performance for four classification metrics: Accuracy, precision, recall, and F1. Full results and further details about this study are included in Appendix B1.

We find that certain methods drastically outperform others in the classification task. Overall, the smaller neural network (3 layers) performs the best in the F1 metric and the Gaussian Process Classifier (GPC) performs best in overall accuracy. We note that the success of the GPC comes at the cost of significantly higher computational expense and this method would not be viable on a significantly larger dataset. Overall, the classification performance may seem underwhelming, but we discuss the challenges that come with this classification formulation later in this section.

Another key takeaway is the improvement of both accuracy and F1 score with increased dataset size, a trend that remains clear into training sizes on the order of thousands. This suggests that additional bicycle designs would allow machines to better understand the bicycle design space and continue to provide valuable information to classification models. As such, a future expanded dataset with new BikeCAD archive designs would likely be valuable.

Comparison of classification using image and parametric data: Next, we seek to understand how different varieties of data impact an algorithm's ability to understand a design. In particular, we examine the relative value of images and

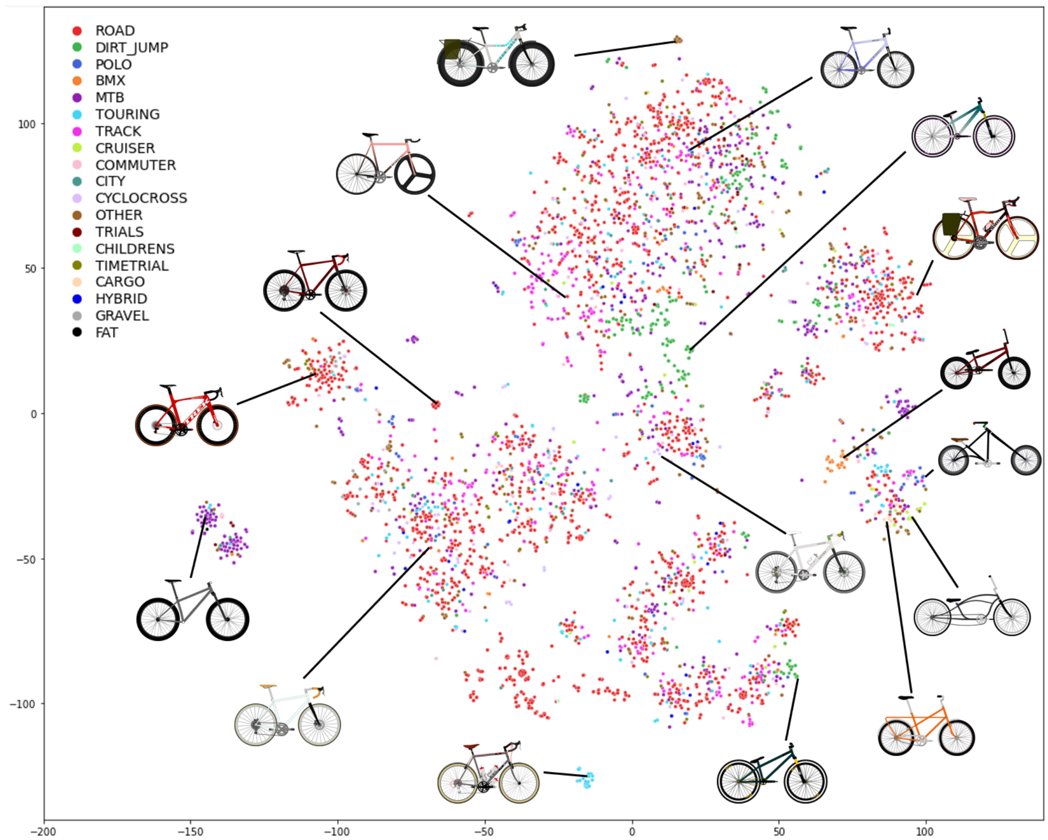


Fig. 6: Visualization of the 2-dimensional embedding generated through t-Distributed Stochastic Neighbor embedding of parametric data. Horizontal and vertical axes denote the two embedding dimensions.

Table 4: Classification accuracy and F1 scores for three custom deep networks. Top and average scores are determined over 10 instantiations.

Model	Average		Top		Train Time (s)
	Acc.	F1	Acc.	F1	
Parametric DNN	**72.0%**	**48.0%**	**73.1%**	**51.6%**	149.0
Image CNN	65.6%	44.6%	66.8%	47.9%	96.0
Combination NN	71.0%	47.8%	72.5%	50.0%	30.6

parametric data. Three deep neural networks are all trained for the bicycle classification task, however, the first uses only BIKED's parametric data, the second uses only BIKED's image data, and the third uses both simultaneously. Network architectures, training details, and training plots are included in Appendix B2. In this study, the average and highest classification accuracies and F1 scores over ten instantiations are reported in Figure 4. The parametric deep neural network's classification performance is notably better than the previous study's neural networks, primarily due to the custom architecture, which includes dropout and batch normalization layers to correct for overfitting. While the three-layer architecture was found to perform best in the previous study, a five-layer network is found to be preferable once measures to combat overfitting are imposed.

The classification accuracy using only parametric data is superior to the accuracy using only images, suggesting that the parametric data is richer in meaningful information than the image data. This is expected since the images are deterministically generated from the parametric data so should have strictly less information. In other words, different parametric vectors may map to the same image, but different images will never map to the same parametric vector. Though we would expect the classifier using both images and parametric data to have comparable performance, it ended up falling slightly short of the parametric classifier due to overfitting, which can be observed in the training plots included in Appendix B2. This overfitting may be explained by the drastically higher input dimensionality of the combined data over the pure parametric data.

Analysis of Classification Results with Confusion Matrices: Of the classifiers tested, none attained classification accuracy higher than 75% or F1 higher than 55%. For this reason, claiming that machines can truly understand the design of bicycles requires further justification. The formulated problem of bicycle classification is challenging because of the previously mentioned overlapping class labels and imbalance of classes. While aggregate metrics are handy for benchmarking algorithms, examining their exact behavior often requires a more detailed performance analysis. Confusion matrices are often used to report and visualize classification performance. We include confusion matrices that show the proportion of test set designs of a particular class

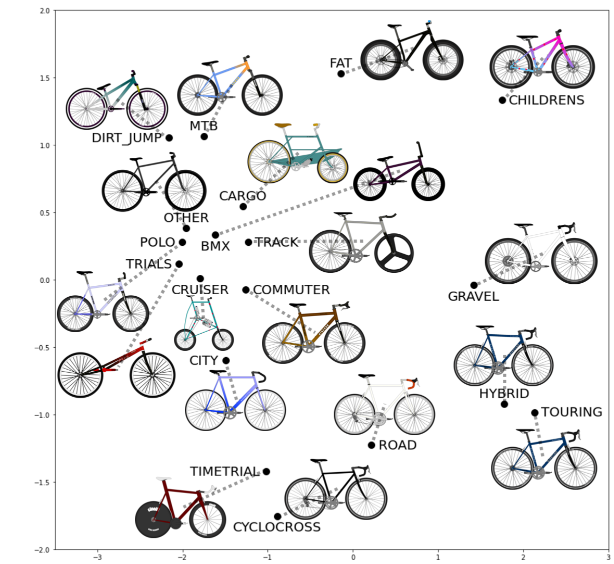


Fig. 7: Visualization of "median" bikes of each class in a 2-dimensional Principal Component Analysis embedding generated using the 50 most significant parameters. The principle PCA component is shown on the horizontal axis and secondary component is shown on the vertical axis.

(vertical axis) that are classified into a particular classification (horizontal axis). A perfect classifier's confusion matrix is the identity matrix. Shown in Figure 8 is the confusion matrix for the highest performing deep neural network trained on parametric data from the previous subsection. Confusion matrices for the deep image classifier and hybrid classifier are included in Appendix B3.

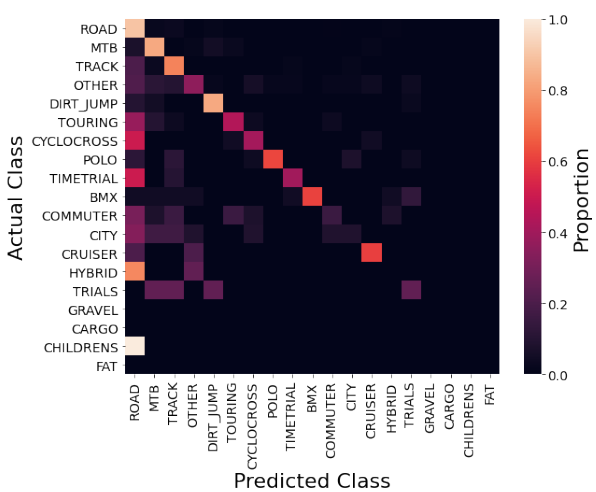


Fig. 8: Confusion matrix of parametric DNN predictions on a previously unseen test set. Actual classes are shown on the vertical axis and predictions are shown on the horizontal axis. Classes are sorted from most prevalent (top left) to least prevalent (lower right).

Examining the confusion matrix, we see that the classifier indeed seems to be struggling due to the overlap and poor definition of some class labels as well as significant class imbalance. Classes with very few designs (fat, children's, cargo, and gravel with 1, 10, 14, and 19 designs respectively) do not have enough training data to be classified well. In addition, three of these classes do not have a single representative model in the test set. This phenomenon significantly impacts F1 scores, which are calculated with macro averaging (see Appendix B1) and primarily explains the poor F1 performance. Subclasses of road bikes, like 'gravel', 'time-trial', 'cyclocross', and 'touring' are often categorized into the general 'road' class. Additionally, ambiguous classes like 'hybrid', 'other', and arguably 'city' and 'commuter' pose challenges for the classifier since a variety of bicycle styles may fit into these classes. This indicates that the classifier is struggling due to the overlap and poor definition of class labels. From these observations, we can conclude that the classifier's moderate performance is more an indication of the challenges brought about by the construction of the classification problem, rather than a difficulty to understand the design of the bicycle.

Interpretability analysis to identify design attributes: While we have demonstrated that neural networks can understand the design of bikes and predict class fairly well, they still act as black-box function approximators. Understanding the classification impact of individual parametric features can help designers appreciate the particular factors that define a specific class of bike. For example, one may question what specific features cause the network to assign a particular model to the road bike class instead of the mountain bike class. To study such questions, the final part of this classification study examines the results of a Shapley Additive Explanations (SHAP) analysis [8]. SHAP locally approximates model predictions over individual input values, effectively capturing the predictive significance of each input parameter. This analysis shows the average impact of a particular feature on the classification probability for each class type.

Figure 9 shows the results of the SHAP analysis on the highest performing deep neural network which scores 73.1% classification accuracy. We observe that each of the most significant parameters besides the number of cogs is a boolean type. Many directly relate to the style of a particular component. The significance of many of these parameters makes intuitive sense. Handlebar, fork, and dropout style are major distinguishing factors between bike classes and most designers of a particular class of bike will choose to use corresponding classes of components for their design. Humans would probably not immediately consider the number of cogs, material, presence of derailleur, or curvature of the fork when classifying bikes, but would likely acknowledge their significance. We can also observe that a few features have a particularly large impact on the probability that a bike is classified as a particular class, like track style dropout spacing for track bikes or suspension forks for mountain bikes. Such SHAP Analyses are reasonably consistent between different instances of the same network architecture, but the importance ranks of the top 20 parameters often fluctuate in ranking by 3-5 places.

4.3 Bicycle Synthesis using Variational Autoencoders

In this section, we consider the question of how algorithms can generate previously unseen bicycle designs. While generative methods have been used extensively on images, BIKED's parametric data opens possibilities to directly synthesize designs in the parameter space. We aim to contrast the two methods by considering a variety of bicycle synthesis methods including random generation, interpolation, and extrapolation using Variational Autoencoders (VAEs) [29, 30]. VAEs encode input data samples into a low dimensional latent space and subsequently decode samples from latent space vectors [29, 30]. The VAEs can be used for design synthesis by generating new latent space vectors, either through random sampling, interpolation, or extrapolation, then decoding the generated latent space vectors to synthesize new designs. We train two VAEs, one on images and one on parametric data. Since parametric design synthesis results are difficult to qualitatively evaluate, images are generated from synthesized parametric data using the method discussed in Section 3.7. Appendix C1 explains the architecture and training details of the VAEs in detail.

Evaluating trained VAEs using reconstruction quality: Examining a VAE's reconstructive ability by encoding and subsequently decoding samples is a common way to evaluate its performance. Figure 10 shows the trained VAEs' performance on a reconstruction task on previously unseen data. Compared to the slightly blurry images generated by the image VAE, the parametric VAE tends to generate more realistic images, largely due to the image generation process in BikeCAD. On the other hand, the image VAE tends to reconstruct the input more accurately. This is largely because the image VAE is directly trained to reconstruct visual appearance while the parametric VAE is only trained to do so indirectly. Critical to this observation is the fact that the parametric VAE knows no weighting of the relative importance (or particularly visual importance) of features. For example, the number of teeth on the third rear cog is just as important to the parametric VAE as the handlebar type. Weighting for this relative importance could be a future line of inquiry. Finally, we note that both VAEs struggle to reconstruct bikes with very unconventional features like the inverted handlebar and extra tube of the third bike on the right. This may be a limitation of the quantity of data—when an unconventional feature is present in extremely few designs, the VAEs do not have enough training data to learn to reconstruct these features, especially if these features do not appear in the training set. It may also be a limitation of the VAE model itself, since we experience that the parametric VAE sometimes struggles with training issues like posterior collapse, which can be sensitive to initialization and reduce reliability.

Bicycle synthesis through interpolation and random generation: Next, we consider the task of bicycle synthesis. Full design synthesis is a problem with many possible considerations and objectives, but as a starting point, we attempt a qualitative comparison of randomly generated designs from our machine learning models. We test two synthesis methods on each of the trained VAEs. The first of these is randomly sampling a vector from the latent space of the VAE and decoding this vector using the decoder. The random selection is performed by generating a probability distribution function for each value of the encoder's output based on the test data, then sampling from each distribution independently to generate a random latent vector. An alternate approach is to interpolate some arbitrary fraction of the distance between latent vectors encoded from existing designs in the dataset instead of sampling randomly. Finally, we also consider interpolation between dataset designs in the original parameter space. Several designs synthesized using each of these methods are shown in Figure 11. Though quantitative evaluation of synthesized designs is not considered in this work, future work may leverage quantitative evaluation metrics that have been used to evaluate Generative Adversarial Networks [31, 32]. We discuss other next steps in generative methods for bicycle design later in this section.

Comparison of interpolation and extrapolation in original parameter space and VAE latent spaces: Next, we dive deeper into interpolation options and further consider extrapolation for the three methods considered above (original parameter space, parametric VAE latent space, and image VAE latent space). We manually select a particularly demonstrative example to showcase, interpolating between a dirt jump and road bike design and extrapolating from the road bike away from the dirt jump bike[3]. Results are shown in Figure 12. In the first two columns, we note the discontinuities in the interpolation of categorical components which, due to the nature of the image regeneration process, abruptly switch styles as soon as one style's probability supersedes another. Interpolation between two designs in the parameter space typically works well, but extrapolation quickly becomes unrealistic, generating bikes that are sometimes disconnected, as shown in the bottom two images on the left. Interpolation and extrapolation both work well in the latent space of the parametric VAE, with extrapolation results generally being quite realistic. In the example shown, extrapolating on the road bike in a direction opposite the dirt jump bike shifts the design to begin resembling a timetrial (triathlon) bike. However, the limited reconstructive capability of the parametric VAE detracts from the fidelity of the newly generated designs with respect to the original designs. Extrapolation based on the image VAE is very poor and is further exacerbated by the limited fidelity of the image VAE's reconstruction, which is particularly poor in this example. Interpolation in the image VAE's latent space has other undesirable properties, such as the visual overlap of the source designs rather than a gradual change in geometry. Based on these qualitative results, interpolation based on a simple image VAE does not seem to be an effective method for quality bicycle synthesis.

[3] Interpolation/Extrapolation formulation included in Appendix C2

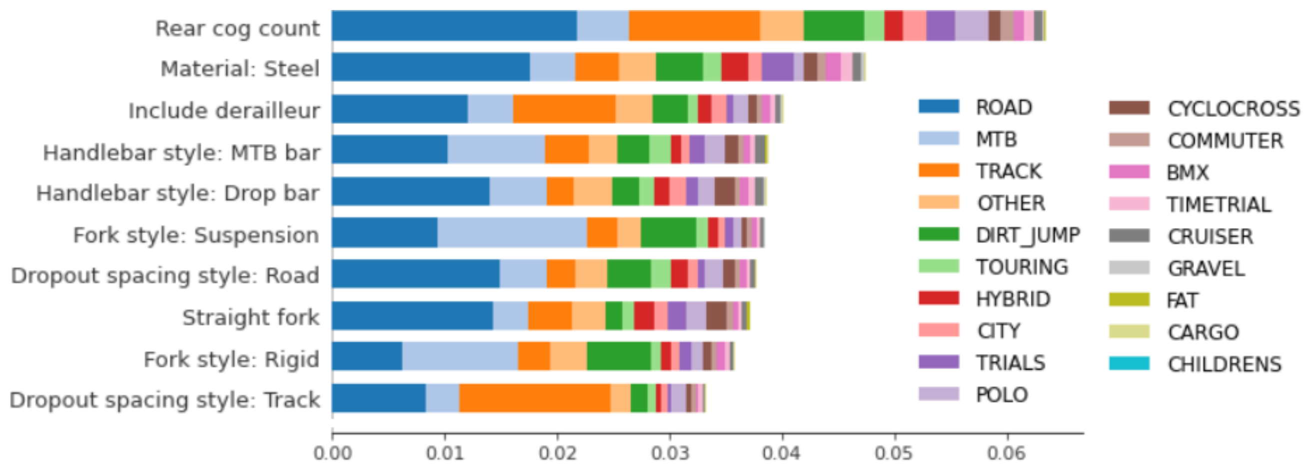
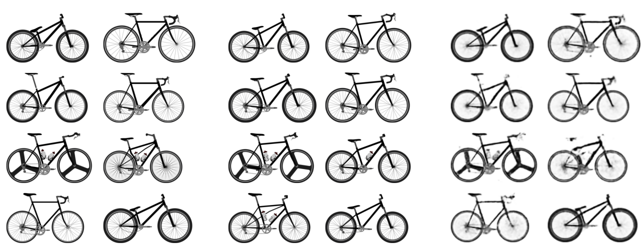
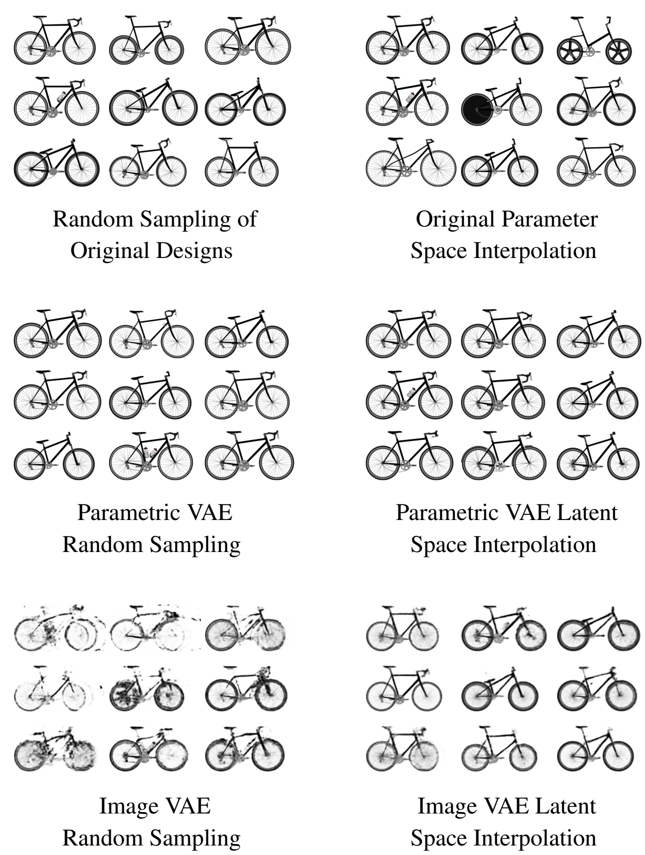


Fig. 9: Shapley Additive Explanations analysis of feature significance on design classification result. The top 10 most significant features are shown on the vertical axis and the mean impact to classification probability is shown on the horizontal axis. Since parameters are one-hot encoded, we list boolean values describing whether or not a categorical parameter falls into a particular class as "Parameter: Class." Note that a high significance does not imply a positive correlation.



(a) Images of Original Designs (b) Parametric VAE Reconstruction (c) Image VAE Reconstruction

Fig. 10: Original bike images and images reconstructed by the two Variational Autoencoders.



Random Sampling of Original Designs

Original Parameter Space Interpolation

Parametric VAE Random Sampling

Parametric VAE Latent Space Interpolation

Image VAE Random Sampling

Image VAE Latent Space Interpolation

Fig. 11: Bike Images randomly generated through random latent space sampling, random latent space interpolation, and random parameter space interpolation.

Further advancement of Generative Methods for Bicycles We envision many opportunities for further development of generative methods for bicycles. Other generative methods like Generative Adversarial Networks (GANs) [33] may be promising alternatives to Variational Autoencoders (VAEs). BIKED's component labels and images also open possibilities for hierarchical design methods or approaches based on symbolic reasoning. Conditional approaches based on bike class may also be viable. We also note that reliable synthesis of designs will likely require rule-based feasibility checking of geometric validity and manufacturing constraints in a sort of "expert-system-based" approach. In addition to feasibility checking, performance evaluation of considerations like structural stiffness, aerodynamics, weight, and ergonomics will be important in any performance-aware synthesis method and will be an interesting avenue for future research. Incorporating BIKED-based automated feasibility checking, performance evaluation, and optimization methods with advanced generative methods is a promising avenue towards the ultimate goal of full performance-aware bicycle synthesis.

5 Strengths and Novelty

Compared to existing 3D model and topology optimization datasets that are commonly used in the data-driven design field, BIKED emphasizes comprehensive design information. This makes BIKED well suited to a multitude of

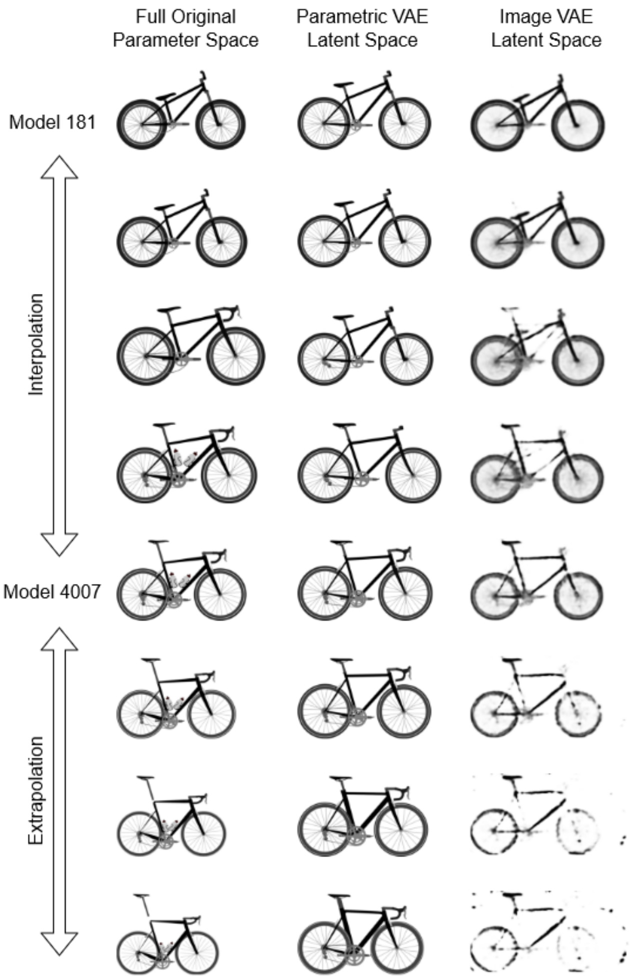


Fig. 12: Interpolation and extrapolation results between a road bike and dirt jump bike. The left side of the figure shows interpolation and extrapolation results carried out in the original parameter space between original designs. The center and right show results from interpolation and extrapolation carried out between latent space representations of the original designs encoded by the two Variational Autoencoders, then decoded and reprocessed (when applicable) into images.

data-driven design tasks that would not be feasible using other datasets. BIKED's detailed component images and parametric data may furthermore support hierarchical methods that would not be supported by other datasets.

BIKED's designs are all hand-designed and, as discussed in Section 2.3, the inbuilt software features help ensure that the majority of BIKED designs are of good quality. BIKED is extensively processed, organized, and documented to make BIKED as user-friendly as possible.

6 Limitations

As is the case in most datasets, BIKED has several sources of bias to note. The fact that all designs were modeled in BikeCAD is a potential source of bias, since BikeCAD's design tools and parameterization scheme might make certain designs more or less easy to model. Furthermore, there is a bias towards the default bike designs that often serve as the starting point in a user's design process. This bias may be noticeable at a high level in overall design characteristics, or at a low level with users leaving certain dimensions at their default values. This is especially true for more niche parameters that have little to no visual impact on the appearance of the bike model, where novice users may not even understand the parameter's meaning. This bias causes many parameters to have uniform values across the vast majority of designs, which can then cause issues, such as posterior collapse while training Variational Autoencoders. Additionally, BIKED has a temporal bias which is demonstrated and discussed in more detail in Appendix D1. Since model numbers largely correlate to the BikeCAD version originally used to create the design, updates to the software that introduced new parameters or updated old default values are likely the root cause.

We also reiterate some of the potential biases discussed in Section 3.5. Dropping bikes with particular features like tandem components or extra nonstandard tubes may have introduced biases. Additionally, manually overriding component visibility of critical design components could have unintentionally included components that were not intended to be part of the designer's vision. We have worked closely with bicycle experts to use domain knowledge to alleviate as many sources of bias as possible and plan to continue refining our methods and dataset to provide a rich and accurate resource to inspire research on data-driven bicycle design.

We also note that, although BikeCAD is used by many experienced designers, there is no guarantee of quality or performance of any bike design. Some designs are created to be facetious or highly unrealistic. We would also advise users that the raw unprocessed dataset may contain offensive or proprietary imagery and language since designers are free to use terminology and graphical designs of their choice.

7 Future Research Directions

Aside from the three key research questions discussed, we envision many exciting ways to harness BIKED in data-driven design research and bicycling science beyond the initial research presented in this paper. Researchers studying bicycle performance can leverage BIKED's parametric data as an input to multi-physics simulations or develop data-driven regression models. Surrogate models of bicycle frame stiffness or aerodynamic drag could accelerate an iterative design process by eliminating computational bottlenecks caused by Finite Element Analysis (FEA) or Computational Fluid Dynamics (CFD) simulation. Efficient surrogate models can enable high-level optimization tasks like Multidisciplinary Design Optimization (MDO). Rapid performance evaluation methods would be an integral component of any comprehensive performance-aware bicycle synthesis process. BIKED's diversity of bicycle designs also lends itself to studies of customer preference and modeling of this data in conjunction with bicycle classes and parametric properties. The fact that BIKED designs were created largely by bike enthusiasts and tradespeople may be of intrinsic use as a study of customer preference. A more advanced study may expand on BIKED's models with further customer preference surveys for a more detailed study of customer preference in the bicycle market.

8 Conclusion

In this paper, we have presented BIKED, a dataset for data-driven bicycle design. BIKED features 4500 bicycle designs providing full bicycle assembly images, segmented component images, and extensive parametric data to enable various data-driven design approaches. Throughout the paper, we discussed the various processing steps taken to curate the data, and discussed key data-driven applications that we expect BIKED to facilitate. We demonstrated some initial exploration in these directions while simultaneously uncovering features of the dataset and providing baseline performance for common machine learning methods. In the first of these studies, we considered unsupervised dimensionality reduction methods to gain various insights about the dataset. In the second, we trained numerous classification models to predict bicycle class, including 10 baseline models using parametric data and 3 tuned deep neural networks to contrast classification performance using different types of input data. Using one of these classification models, we performed a Shapley Additive Explanations analysis to understand the impact of individual parameters on classification predictions. Next, we trained two Variational Autoencoders to perform a variety of different tasks on both parametric and image data, including design reconstruction, latent space random sampling, interpolation, and extrapolation. Finally, we discussed key strengths and novelty as well as potential limitations and biases in the dataset.

BIKED goes above and beyond existing datasets, providing extensive design information of various types and promising to enable a slew of novel data-driven design applications. In our discussion, we documented the process to curate the dataset, explored BIKED's features, demonstrated baseline performance for common algorithms, and attempted to inspire researchers with sample applications. We hope that BIKED will prove a valuable resource to researchers in the data-driven design community and will support novel and innovative approaches in this rapidly growing field.

9 Acknowledgments

We would like to thank Professor Daniel Frey for his input and guidance throughout the project. We would like to thank Kris Vu for assisting with sourcing files from the BikeCAD archive and Amin Heyrani Nobari for assisting with the exporting of component images. Finally, we would like to acknowledge MathWorks for supporting this research.

References

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., 2009. "Imagenet: A large-scale hierarchical image database". In 2009 IEEE conference on computer vision and pattern recognition, Ieee, pp. 248–255.

[2] Krizhevsky, A., Hinton, G., et al., 2009. "Learning multiple layers of features from tiny images".

[3] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., 1998. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE,* ***86***(11), pp. 2278–2324.

[4] Oke, O., Bhalla, K., Love, D. C., and Siddiqui, S., 2015. "Tracking global bicycle ownership patterns". *Journal of Transport & Health,* ***2***(4), pp. 490–501.

[5] Oja, P., Titze, S., Bauman, A., De Geus, B., Krenn, P., Reger-Nash, B., and Kohlberger, T., 2011. "Health benefits of cycling: a systematic review". *Scandinavian journal of medicine & science in sports,* ***21***(4), pp. 496–509.

[6] Hamilton, T. L., and Wichman, C. J., 2018. "Bicycle infrastructure and traffic congestion: Evidence from dc's capital bikeshare". *Journal of Environmental Economics and Management,* ***87***, pp. 72–93.

[7] Edenhofer, O., 2015. *Climate change 2014: mitigation of climate change*, Vol. 3. Cambridge University Press.

[8] Lundberg, S. M., and Lee, S.-I., 2017. "A unified approach to interpreting model predictions". In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. Curran Associates, Inc., pp. 4765–4774.

[9] Regenwetter, L., Curry, B., and Ahmed, F., 2021. "BIKED: A dataset and machine learning benchmarks for data-driven bicycle design". In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC-21, ASME.

[10] Wilson, D. G., and Schmidt, T., 2020. *BICYCLING SCIENCE*, 4 ed. MIT Press.

[11] Sharp, A., 1977. *Bicycles and Tricycles: A Classic Treatise on Their Design and Construction*, 1 ed. Dover Publications Inc.

[12] Chowdhury, H., Alam, F., and Mainwaring, D., 2011. "A full scale bicycle aerodynamics testing methodology". *Procedia Engineering,* ***13***, pp. 94–99.

[13] Chowdhury, H., and Alam, F., 2012. "Bicycle aerodynamics: an experimental evaluation methodology". *Sports Engineering,* ***15***(2), pp. 73–80.

[14] Malizia, F., and Blocken, B., 2020. "Bicycle aerodynamics: History, state-of-the-art and future perspectives". *Journal of Wind Engineering and Industrial Aerodynamics,* ***200***, p. 104134.

[15] Suppapitnarm, A., Parks, G., Shea, K., and Clarkson, P., 2004. "Conceptual design of bicycle frames by multiobjective shape annealing". *Engineering Optimization,* ***36***(2), pp. 165–188.

[16] Laios, L., and Giannatsis, J., 2010. "Ergonomic evaluation and redesign of children bicycles based on anthropometric data". *Applied ergonomics,* ***41***(3), pp. 428–435.

[17] Swart, J., and Holliday, W., 2019. "Cycling biomechan-

ics optimization—the (r) evolution of bicycle fitting". *Current sports medicine reports,* ***18***(12), pp. 490–496.
[18] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al., 2015. "Shapenet: An information-rich 3d model repository". *arXiv preprint arXiv:1512.03012*.
[19] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., 2015. "3d shapenets: A deep representation for volumetric shapes". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920.
[20] Yang, J., Mo, K., Lai, Y.-K., Guibas, L. J., and Gao, L., 2020. "Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry". *arXiv preprint arXiv:2008.05440*.
[21] Gao, L., Yang, J., Wu, T., Yuan, Y.-J., Fu, H., Lai, Y.-K., and Zhang, H., 2019. "Sdm-net: Deep generative network for structured deformable mesh". *ACM Transactions on Graphics (TOG),* ***38***(6), pp. 1–15.
[22] Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., and Guibas, L. J., 2019. "Structurenet: Hierarchical graph networks for 3d shape generation". *arXiv preprint arXiv:1908.00575*.
[23] Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L., 2017. "Grass: Generative recursive autoencoders for shape structures". *ACM Transactions on Graphics (TOG),* ***36***(4), pp. 1–14.
[24] Sosnovik, I., and Oseledets, I., 2019. "Neural networks for topology optimization". *Russian Journal of Numerical Analysis and Mathematical Modelling,* ***34***(4), pp. 215–223.
[25] Zhang, Y., Peng, B., Zhou, X., Xiang, C., and Wang, D., 2019. "A deep convolutional neural network for topology optimization with strong generalization ability". *arXiv preprint arXiv:1901.07761*.
[26] Curry, B., 2018. Design archive.
[27] Inc, S. Bicycle types: How to pick the best bike for you.
[28] Van der Maaten, L., and Hinton, G., 2008. "Visualizing data using t-sne.". *Journal of machine learning research,* ***9***(11).
[29] Kingma, D. P., and Welling, M., 2013. "Auto-encoding variational bayes". *arXiv preprint arXiv:1312.6114*.
[30] Kingma, D. P., and Welling, M., 2019. "An introduction to variational autoencoders". *arXiv preprint arXiv:1906.02691*.
[31] Chen, W., and Ahmed, F., 2021. "Padgan: Learning to generate high-quality novel designs". *Journal of Mechanical Design,* ***143***(3).
[32] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K., 2019. "Modeling tabular data using conditional gan". *arXiv preprint arXiv:1907.00503*.
[33] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014. "Generative adversarial networks". *arXiv preprint arXiv:1406.2661*.
[34] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

A Appendix A1: Dataset Details

We include a table listing a few sample parameters from each type in Table 5. We note that any of these can be used as the labels in regression or classification tasks, depending on the researcher's goals, though certain parameters like "bike style" may make intuitive sense for many applications.

Appendix B1: Training Details and extended results of Simple Classifiers

We include a more detailed study of classification performance on BIKED's parametric data for a baseline set of 12 classifiers with 5 sizes of training data. For each classifier and train set size, we record the average and top accuracy, recall, precision, and F1, as well as the average evaluation time over 10 instantiations. Each classification model is trained on 200, 600, 1200, 2000, and 3000 randomly sampled designs, leaving the remainder of the 4512 processed designs for testing. We do not include a validation set since no tuning is performed. Only parametric data is used in this study. Each of the 10 instantiations consists of a new model initialization and a different random train-test split. Precision, recall, and F1 are calculated using macro averaging, i.e. calculating recall, precision, and F1 for each individual class and subsequently performing an unweighted average over the classes. If no representative models of a class appear in the test data, we set precision, recall, and F1 to 0 for the class. Table 6 contains the results of this study. The training was performed on a computer with Xeon Silver 4215R CPU, 64 Gb RAM, and two NVIDIA Quadro RTX4000 GPUs. Key parameters of classification models are listed below. For further implementation details of the methods tested, we refer the reader to our provided code.

1. Neural networks tested have 200 neurons per layer, use Rectified Linear Unit (ReLu) activation functions, and are trained using the adam optimizer [34].
2. K-Neighbors-Classifier uses 5 neighbors
3. Random Forest Classifier uses 50 estimators and a maximum depth of 7
4. Gaussian Process Classifier uses the Radial-basis function kernel with a length parameter of 1 and scaling factor of 1
5. SVC classifier uses a linear kernel

Appendix B2: Architecture and training details of deep classifiers

The architectures of the three custom neural networks are shown in Figure 13. The number of neurons in every fully connected layer is labeled in the diagram. Similarly, the number of filters (f), kernel size (k), and stride size (s) are shown for every convolution layer. Dropout layers are labeled according to dropout probability. Models were trained using the Adam optimizer [34] with a learning rate of 10^{-4} using a categorical cross-entropy loss. A batch size of 100 was used for all models. The 4512 models were split into training, validation, and test sets with a ratio of 70:15:15 (3158:677:677). Unlike the study in B1, only one train-test-validation set was used for every instantiation of every model. The training was performed on a computer with Ryzen 9 5950x, 32 GB RAM, and Nvidia RTX3080 GPU. Training plots of the top-performing instantiations of the networks are included in Figures 14 and 15. The training was halted after 100, 20, and 10 epochs without improvement for the parametric, image, and combination classifiers respectively.

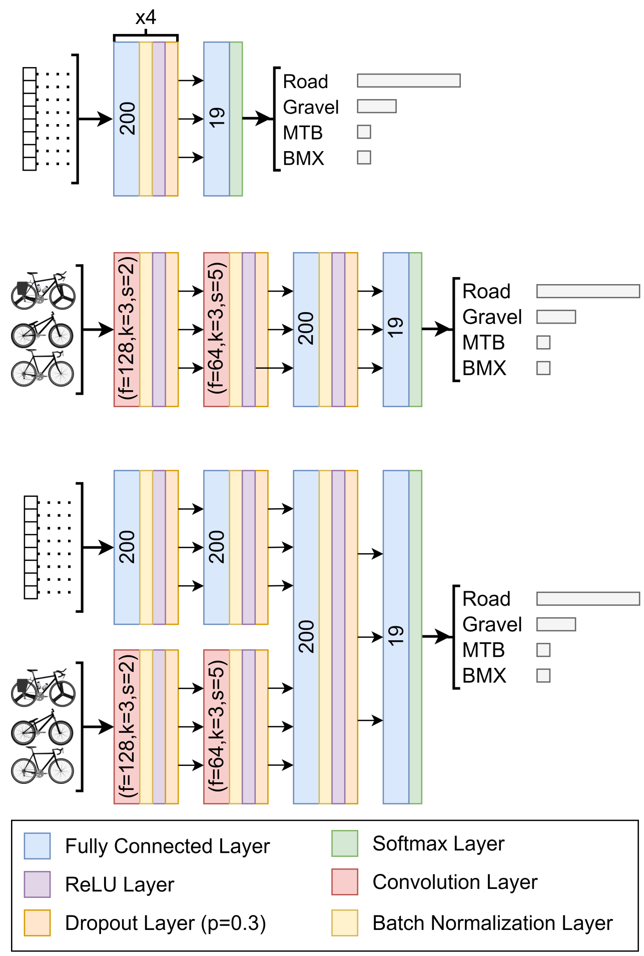


Fig. 13: Network diagrams of the three classification networks implemented.

Appendix B3: Classification Confusion Matrix

Shown in Figures 16 (repeated from the main text) through 18 are confusion matrices of the classification performance of the three custom-built deep classifiers. We can appreciate that all three deep classifiers struggle with the class imbalance and class overlap issues discussed in Section 4.2. We also note that classes of bikes that are visually similar to others are more difficult for the image-based CNN to classify. For example, track bikes are primarily identifiable by their single-speed drivetrain, but since the cassette of

Table 5: Example parameters of each of the four datatype present in BIKED.

Continuous	**Discrete**	**Categorical**	**Boolean**
Top tube length	Number of rear cogs	Wheel type	Include racks (Y/N)
Bottom bracket drop distance	Number of teeth on first rear cog	Handlebar style	Symmetric chain stays (Y/N)
Saddle height	Number of spokes on front wheel	Bicycle style	Include fenders (Y/N)

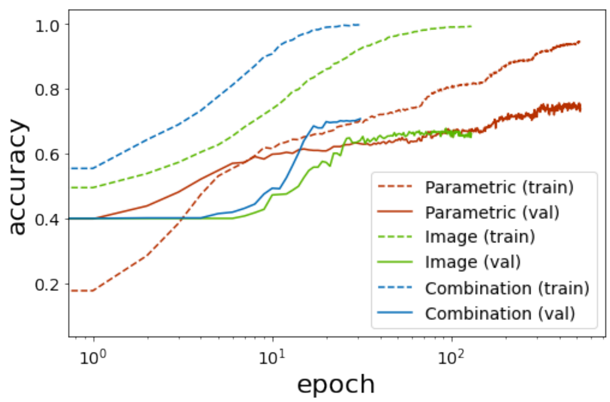


Fig. 14: Accuracy vs training epoch (log scaled) for the three tuned classification networks implemented.

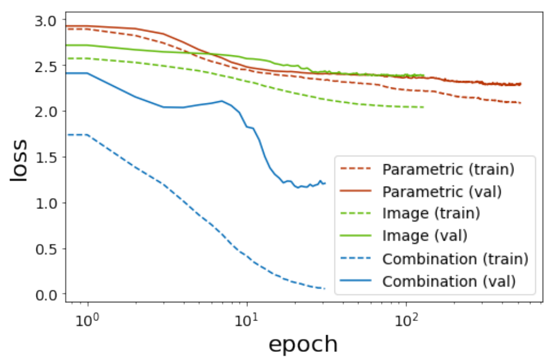


Fig. 15: Loss vs training epoch (log scaled) for the three classification networks implemented.

the bike is not visually prominent, track bikes are more challenging for the CNN compared to the other two classifiers.

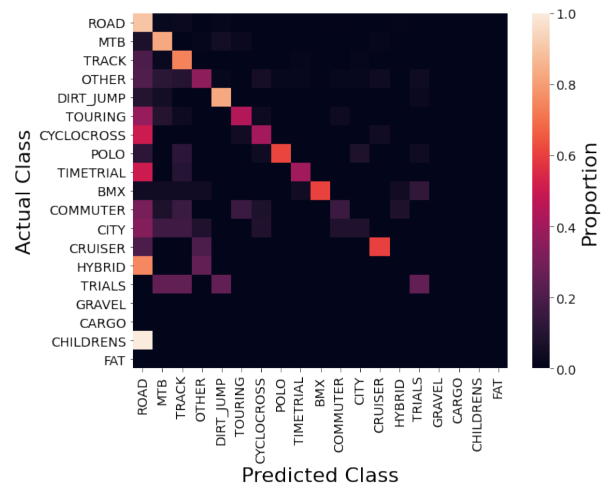


Fig. 16: Confusion matrix of Parametric DNN predictions

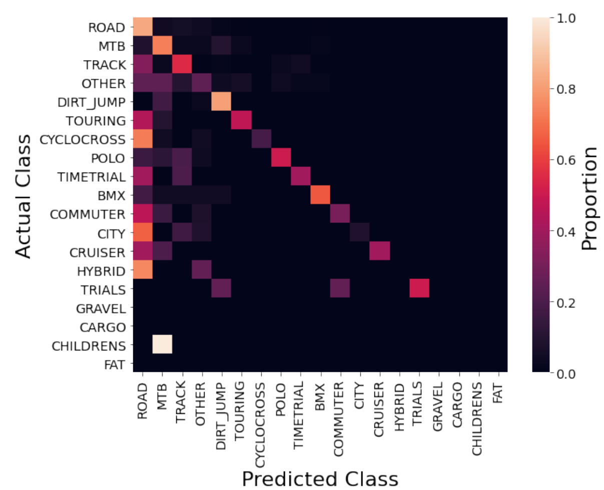


Fig. 17: Confusion matrix of Image CNN predictions

Appendix C1: Architecture and training details of VAEs

Figures 19 and 20 show the architecture of the image VAE and parametric VAE respectively using the same labeling scheme as the classifiers in Appendix B1. The top portion of each diagram represents the encoder that maps a design to a latent space vector. The bottom represents the decoder that maps a latent vector to a design. Models were trained using the Adam optimizer [34] with a learning rate of 10^{-3} for the parametric VAE and 10^{-4} for the image VAE. Training is halted after 200 consecutive training epochs with no im-

Table 6: Classification accuracy, precision, recall, and F1 for 12 classifiers using BIKED's parametric data.

Train Size	**Model**	**Average**				**Top**				**Average**
		Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1	**Train Time (s)**
200	Support Vector Clf.	57.4%	16.3%	33.8%	17.0%	59.9%	18.3%	41.3%	20.2%	9.3
	Naïve Bayes	43.7%	9.3%	30.7%	9.8%	46.9%	11.5%	41.0%	12.3%	4.3
	K-Neighbors	47.9%	13.3%	19.4%	12.8%	51.4%	17.1%	23.4%	15.2%	11.8
	Depth 5 Dec. Tree	52.0%	19.3%	22.4%	19.2%	55.7%	23.1%	30.4%	22.1%	**0.2**
	Depth 8 Dec. Tree	52.1%	20.9%	22.4%	20.7%	57.4%	**28.0%**	27.2%	**26.5%**	**0.2**
	Random Forest	**59.0%**	18.6%	**43.0%**	19.6%	**61.8%**	21.0%	**48.9%**	21.6%	0.4
	AdaBoost	48.1%	10.4%	10.7%	7.8%	51.1%	13.0%	18.3%	9.9%	4.7
	Gaussian Pr. Clf.	47.9%	9.9%	8.4%	7.9%	58.8%	23.5%	29.0%	21.4%	122.8
	3-layer Neural Net	54.9%	19.8%	32.7%	21.0%	56.2%	21.6%	40.0%	22.5%	4.2
	4-layer Neural Net	54.8%	19.5%	30.0%	20.6%	56.9%	22.5%	37.6%	23.1%	2.7
	5-layer Neural Net	54.6%	20.0%	27.5%	20.9%	56.9%	23.4%	32.5%	24.0%	2.5
	6-layer Neural Net	55.3%	**21.2%**	28.7%	**22.2%**	57.1%	25.4%	33.2%	25.5%	2.3
600	Support Vector Clf.	63.4%	23.4%	45.4%	25.6%	64.7%	25.9%	54.7%	28.7%	23.6
	Naïve Bayes	43.3%	14.7%	37.6%	16.8%	45.0%	16.9%	44.2%	19.0%	4.3
	K-Neighbors	52.3%	18.1%	27.9%	19.1%	53.9%	20.2%	34.9%	21.8%	34.3
	Depth 5 Dec. Tree	59.0%	24.1%	29.1%	23.9%	62.3%	28.5%	37.0%	27.6%	**0.2**
	Depth 8 Dec. Tree	57.9%	26.9%	30.2%	27.3%	59.6%	29.3%	36.4%	31.4%	0.3
	Random Forest	**64.0%**	23.6%	**50.3%**	25.5%	**65.4%**	27.1%	57.8%	28.5%	0.4
	AdaBoost	50.1%	10.8%	9.2%	8.2%	53.9%	15.9%	13.8%	12.3%	4.7
	Gaussian Pr. Clf.	62.5%	25.1%	39.9%	25.6%	65.0%	27.6%	**61.1%**	29.0%	454.8
	3-layer Neural Net	61.6%	29.0%	43.0%	**31.6%**	62.7%	30.9%	48.3%	**34.5%**	14.5
	4-layer Neural Net	60.8%	**29.0%**	42.1%	31.4%	61.7%	**32.0%**	47.7%	33.7%	10.1
	5-layer Neural Net	58.6%	27.4%	35.0%	28.7%	62.4%	30.7%	43.7%	32.9%	8.5
	6-layer Neural Net	59.7%	29.0%	38.3%	30.7%	62.0%	31.1%	46.6%	33.9%	8.5
1200	Support Vector Clf.	66.3%	28.5%	49.1%	31.4%	**68.4%**	31.9%	**60.5%**	36.0%	40.7
	Naïve Bayes	41.8%	20.8%	32.2%	22.2%	42.8%	22.7%	40.8%	23.8%	3.6
	K-Neighbors	55.9%	23.5%	37.3%	25.4%	56.7%	26.5%	43.6%	29.3%	59.1
	Depth 5 Dec. Tree	62.8%	26.3%	30.0%	26.1%	64.3%	28.6%	33.8%	29.5%	**0.3**
	Depth 8 Dec. Tree	61.9%	32.7%	38.3%	33.6%	63.5%	36.1%	42.7%	36.7%	**0.3**
	Random Forest	65.3%	25.2%	**51.6%**	26.8%	67.0%	26.3%	59.4%	28.8%	0.4
	AdaBoost	48.9%	10.5%	7.5%	7.8%	51.1%	12.0%	9.9%	9.1%	4
	Gaussian Pr. Clf.	**66.7%**	32.2%	46.4%	33.8%	67.8%	35.5%	59.2%	37.1%	1204.8
	3-layer Neural Net	63.9%	34.7%	50.4%	**38.4%**	65.9%	37.9%	53.0%	**41.8%**	28.9
	4-layer Neural Net	63.2%	34.7%	45.8%	37.6%	64.0%	37.5%	49.8%	40.5%	19.7
	5-layer Neural Net	63.3%	**34.9%**	45.3%	37.1%	64.9%	37.5%	53.1%	40.1%	16.3
	6-layer Neural Net	62.7%	34.2%	43.1%	35.7%	65.1%	**38.4%**	51.2%	40.3%	12.7
2000	Support Vector Clf.	68.5%	31.9%	54.9%	35.6%	69.9%	33.8%	**67.0%**	38.9%	54.1
	Naïve Bayes	40.0%	24.8%	29.6%	24.4%	41.8%	26.4%	35.7%	26.4%	2.2
	K-Neighbors	58.3%	27.3%	43.0%	30.2%	60.2%	29.7%	48.4%	33.6%	78
	Depth 5 Dec. Tree	63.7%	26.7%	32.6%	26.7%	66.3%	30.9%	40.2%	31.1%	**0.3**
	Depth 8 Dec. Tree	65.5%	32.8%	41.1%	34.8%	67.0%	35.2%	48.8%	38.2%	**0.3**
	Random Forest	66.5%	26.6%	51.3%	28.5%	67.7%	27.3%	60.9%	29.4%	0.4
	AdaBoost	49.8%	11.0%	7.9%	8.2%	52.9%	17.6%	13.7%	12.9%	4.1
	Gaussian Pr. Clf.	**69.5%**	37.9%	54.6%	39.7%	**71.8%**	43.3%	62.8%	46.2%	2686.4
	3-layer Neural Net	66.8%	**41.0%**	**55.3%**	**44.8%**	67.7%	43.5%	64.2%	47.8%	47
	4-layer Neural Net	65.7%	40.1%	53.1%	43.4%	67.6%	43.4%	62.0%	**48.0%**	26.1
	5-layer Neural Net	65.4%	38.3%	50.1%	40.6%	69.5%	42.8%	56.7%	45.3%	20.1
	6-layer Neural Net	65.5%	38.8%	48.0%	40.5%	68.4%	**46.5%**	53.0%	46.2%	20.1
3000	Support Vector Clf.	69.6%	34.3%	55.7%	38.0%	70.7%	37.8%	68.3%	43.1%	58.7
	Naïve Bayes	38.3%	31.5%	28.5%	27.9%	40.9%	36.3%	33.1%	32.3%	1.4
	K-Neighbors	60.3%	30.8%	45.2%	32.9%	61.4%	35.5%	52.1%	35.7%	64.3
	Depth 5 Dec. Tree	64.6%	29.9%	35.4%	29.7%	65.7%	32.6%	40.9%	31.1%	**0.3**
	Depth 8 Dec. Tree	66.5%	34.6%	43.6%	36.3%	67.3%	39.0%	47.2%	40.7%	0.4
	Random Forest	66.8%	27.2%	50.5%	28.9%	69.6%	31.5%	58.5%	33.9%	0.5
	AdaBoost	50.4%	10.6%	7.6%	7.8%	55.0%	14.0%	15.7%	11.4%	4.3
	Gaussian Pr. Clf.	**72.0%**	**42.7%**	**63.2%**	47.3%	**74.4%**	47.7%	**73.3%**	53.0%	5250.4
	3-layer Neural Net	68.7%	45.6%	60.6%	**49.9%**	71.2%	51.2%	66.4%	**55.0%**	50.9
	4-layer Neural Net	67.5%	44.1%	57.1%	47.4%	69.5%	49.8%	68.1%	53.5%	28.5
	5-layer Neural Net	68.6%	44.6%	55.0%	47.5%	69.9%	**51.6%**	62.2%	52.0%	31.7
	6-layer Neural Net	68.7%	47.2%	54.0%	48.7%	70.9%	51.5%	59.5%	53.2%	29.3

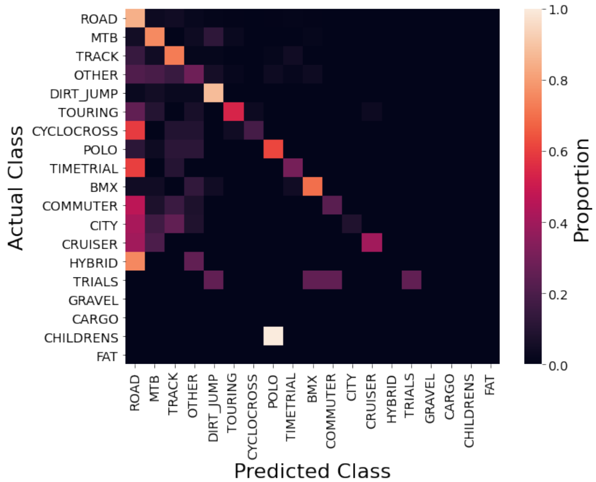


Fig. 18: Confusion matrix of Combination NN predictions

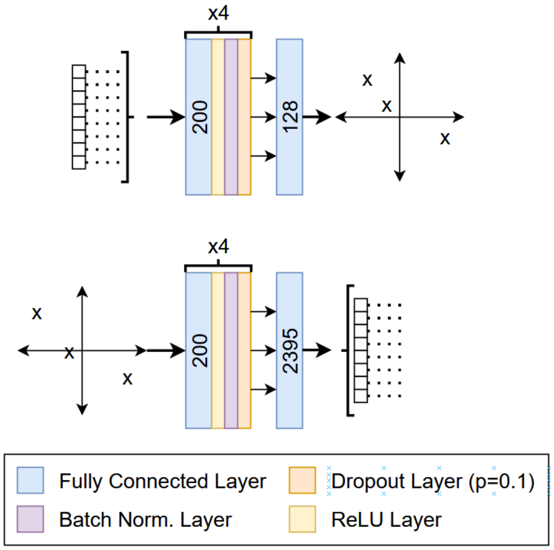


Fig. 20: Model architecture of Variational Autoencoder for parametric data.

provement for the parametric VAE and after 20 for the image VAE. Data dimensionality and train, validation, and test set splits are identical to the classifiers described in B2. The training was performed on a computer with Ryzen 9 5950x, 32 GB RAM, and Nvidia RTX3080 GPU. Training plots of

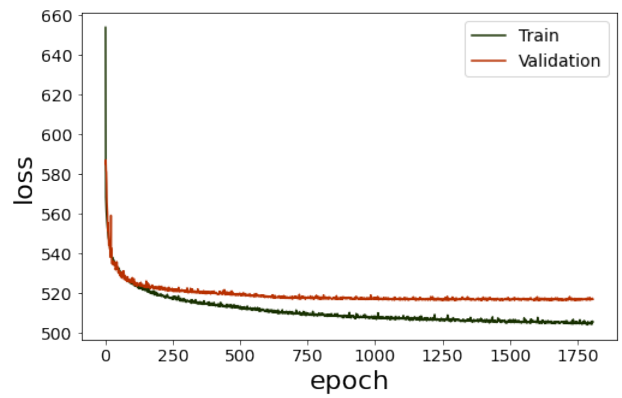


Fig. 21: Training of Parametric VAE

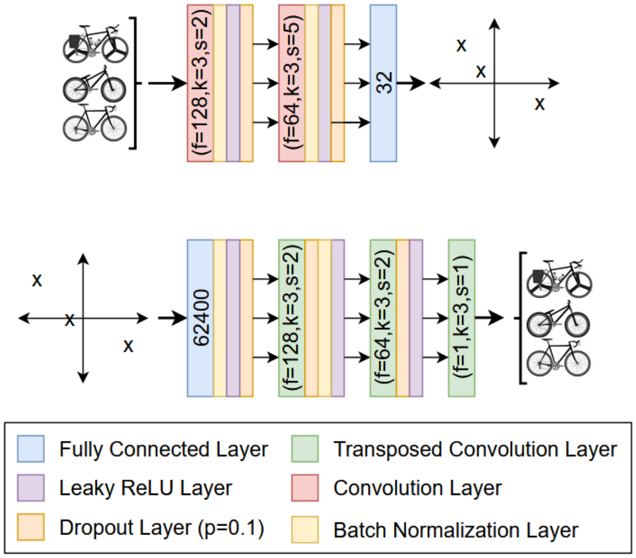


Fig. 19: Model architecture of Variational Autoencoder for images.

the image and data VAEs are included in Figures 21 and 22.

Appendix C2: Interpolation Setup

We define the interpolation and extrapolation vectors (i_n) and (e_n), respectively in terms of two original design vectors m_1 and m_2 as follows:

$$i_k = m_1 + k(m_2 - m_1) * s_i, \ \forall \ k \in [1, n_i] \tag{1}$$

$$e_k = m_2 + k(m_2 - m_1) * s_e, \ \forall \ k \in [1, n_e] \tag{2}$$

Here, s_i and s_e are the interpolation and extrapolation step sizes and n_i and n_e are the number of interpolation and extrapolation steps respectively. In this demonstration, we select $n_i = n_e = 3$. In the case of interpolation, our step size is determined by the number of interpolation steps, while in extrapolation, we choose to select the step size such that our last extrapolation point is equally far from m_2 as the distance between m_1 and m_2:

$$s_i = \frac{1}{n_i + 1} \tag{3}$$

$$s_e = \frac{1}{n_e} \tag{4}$$

Using this formulation, the results in Figure 12 are generated.

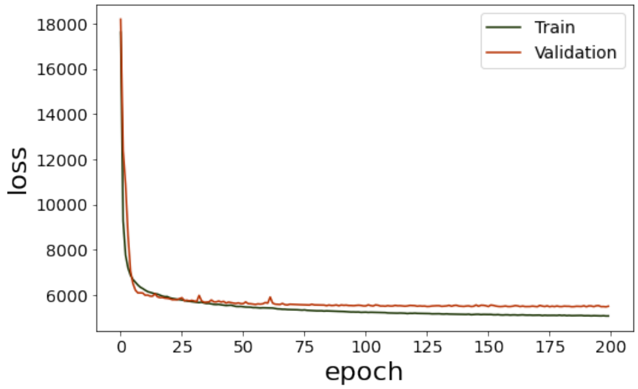


Fig. 22: Training of Image VAE

Appendix D1: Temporal Bias

Figure 23 contains a 2-Dimensional PCA embedding with models colored by model number. Clearly, there is a strong bias in the dataset based on model number, which is caused by the "status quo" of bike parameters evolving as new versions of the software were developed. This bias would not be apparent had we shuffled the data, and we intentionally left the data unshuffled in case the ordering of models proves valuable in future research.

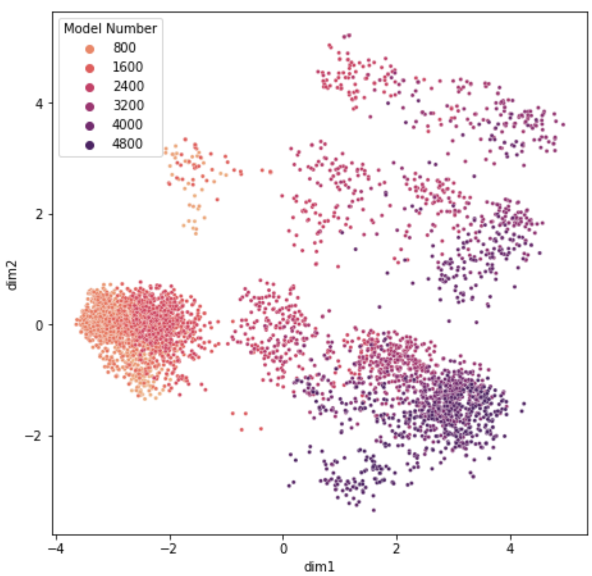


Fig. 23: PCA embedding with designs labeled by model number illustrating the temporal bias in the dataset