Machine Learning for Climbing

Route Generation and Grade Classification on the MoonBoard

Junran Shi

Minerva University

Advisor: Carl Scheffler

Second Reader: Patrick Watson

Abstract

This project applies machine learning methods to generate novel climbing routes and classify their difficulty in the context of the MoonBoard training board for climbing. The goal of this project is to explore the effectiveness of artificial creativity in the field of "route-setting," which describes using limited holds to create climbing routes on a wall. This paper details the architecture, training process, and results of the generative and classification models, as well as the route validation process involving surveys and collecting feedback from experienced climbers. We found that the generated routes are of decent quality and can almost be mistaken for human-generated routes, though still lacking in creativity and elegance. Additionally, we attempt to improve the objectivity and accuracy in grading the difficulty of routes using classification models. We demonstrate improved results by using class weights and incorporating machine-generated climbs in the training data. Lastly, the outcomes of this project are presented using a website that allows climbers to view machine-generated climbs of their desired grades and two classification models' grade predictions.

The project's GitHub repository is hosted at https://github.com/junranshi/capstone_moonboard, and users can visit https://junranshi.github.io/moonboard_website/ to generate routes of their desired grades and see their predicted grades.

Table of Contents

1. Introduction

In indoor climbing, route-setting describes the process of using holds to compose routes on a wall to be climbed. To many, route-setting is more of an art than a science due to the high level of creativity involved. The goal of this project is to apply machine learning methods to explore the extent to which we can employ artificial creativity in the field of setting bouldering routes. Additionally, we attempt to improve the objectivity and accuracy in grading the difficulty of routes using classification models.

The data is sourced from the MoonBoard, a standardized indoor climbing wall with 25,000 existing routes used for training. Variational Autoencoders are used for route generation, and new routes are evaluated by experienced climbers. Enhancements to the generation method configurations lead to a significant improvement in route quality, and generated routes improve the classification accuracy for higher-difficulty routes compared to previously published papers.

1.1 Climbing

Indoor climbing is a relatively new sport. The first indoor climbing hall opened in 1974 in Bolzano, Italy. Since the turn of the millennium, the sport of climbing has undergone a growth spurt [1]. The growing number of indoor climbing gyms opens the doors to the sport to people of all demographics, demonstrated by a 7.9% increase [2] in the number of indoor climbers from 2019 to 2021. Since its introduction as a medal sport to the Tokyo Olympics in 2021, the sport has grown faster than ever. However, as a result of its late popularity compared to other sports, there hasn't been nearly as much investigation into the application of modern technology to analyzing the sport and athletes' performances.

1.1.1 Bouldering

Bouldering is a type of climbing that differs from traditional bolted climbing. It is performed at a lower altitude and uses soft crash pads for falls, thus eliminating the need for harnesses or ropes. Climbers can train at indoor bouldering gyms where they can find many routes, or "problems," made up of plastic "holds" on a climbing wall. The climber completes a route using the designated holds (Fig 1).

Outdoor Climbing Indoor Climbing

Fig 1. Outdoor climbing vs. indoor climbing.

1.1.2 Route-setting

In outdoor climbing, the natural formation of the rock determines the availability of holds, and climbers use them to form routes. Regarding indoor climbing, the routes are no longer naturally constrained and thus allow for human creativity in the combinations of holds and movements. Typically, each bouldering gym has "route-setters" who regularly design, test, and put up new routes. Generating these problems is a creatively complex task for which route-setters must consider many factors, such as the wall angle, the climber's abilities, physical

limitations, the route style, etc. Route-setting is a critical element of climbing since the formation of routes largely determines how much the climber enjoys and improves.

1.1.3 Grades

Each climbing route, either outdoor or indoor, is given a difficulty grade. The grade is typically agreed upon by the route-setter(s) for indoor routes. Many factors contribute to a route's grade, including hold types, hold placements, wall angle, required movements, etc. Given climbers' differences in physical abilities and strengths, grading is often subjective, personal, and situational, making it more art than science. Since climbers typically try routes in a certain grade range they are comfortable with, it can be frustrating when a route is "sandbagged" — harder than its grade suggests — and when different route-setters and gyms have their subjective grading standards.

Two popular grade scales for bouldering are the Font Scale, more popular in Europe, and the V Scale, more popular in the Americas and Asia. Fig 2 shows the conversion between the two scales.

BOULDERING GRADES	
V Scale	Font Scale
V0	4
V0+	4+
V1	5
V2	5+
V3	6A/6A+
V4	6B/6B+
V5	6C/6C+
V6	7A
V7	7A+
V8	7B/7B+
V9	7B+/7C
V10	7C+
V11	8A
V12	8A+
V13	8B
V14	8B+
V15	8C
V16	8C+
V17	9A

Fig 2: Conversion between V Scale and Font Scale for bouldering grades.

1.2 MoonBoard

The MoonBoard, created in 2005, is the oldest, most effective, and most popular training board for climbing. It is a standardized interactive training wall [3] that can be found in gyms across the world, connecting a community of climbers who actively contribute new routes to MoonBoard's growing database. The MoonBoard, instead of other training boards or typical gym routes, is chosen for the purpose of this project for the following reasons:

1. The problem space for routes on typical gym walls is complex and non-discrete, thus necessitating the need for an approximation. A standardized training board such as the MoonBoard allows us to simply represent holds and locations with numbers to be processed for machine learning without introducing the complexities of typical indoor climbing.

2. As the most popular and accessible training board, the MoonBoard hosts the biggest dataset of routes that can be used for model training. Its popularity also allows this project to generate the most value for users.
3. The MoonBoard dataset is the most readable available. Until 2021, the MoonBoard website permitted users access to all available routes. Prior studies could use web scrapers to gather routes, while the process would be more challenging for other boards which users could only access via mobile apps.

1.2.1 Setup

Each MoonBoard is an identical short and tilted climbing wall with 18 rows and 11 columns of holds (Fig 3). Users can browse, filter, and upload problems in the MoonBoard app. After the user connects to a MoonBoard and a route is selected in the app, LED lights on the MoonBoard point out holds that constitute the route. The MoonBoard has several setups (2016, 2017, and 2019) that differ in the holds used. I will use routes on the 2016 setup for this project since it contains the most routes.

For each route, a climber can only use circled holds for both hands and feet (Fig 4), with those in green as starting holds for hands, blue as intermediate holds, and red as finishing holds. A route is considered complete if the climber can "match" the finishing hold(s), meaning holding it with both hands in a stable position. Each route is graded with either Font or V grades. Due to its overhanging angle and small holds, MoonBoard routes have V4 as the easiest grade as opposed to V0 in typical indoor climbing. The highest grade a route can have on the MoonBoard is V14, approaching the highest difficulty in established outdoor bouldering routes.

Climbers can design their own routes and upload them to the MoonBoard database via the app. Currently, there are more than 56,000 routes from the community. Each route is given a grade by the route-setter and an average grade by climbers who complete the climb.

Fig 3: A climber using the MoonBoard training board.

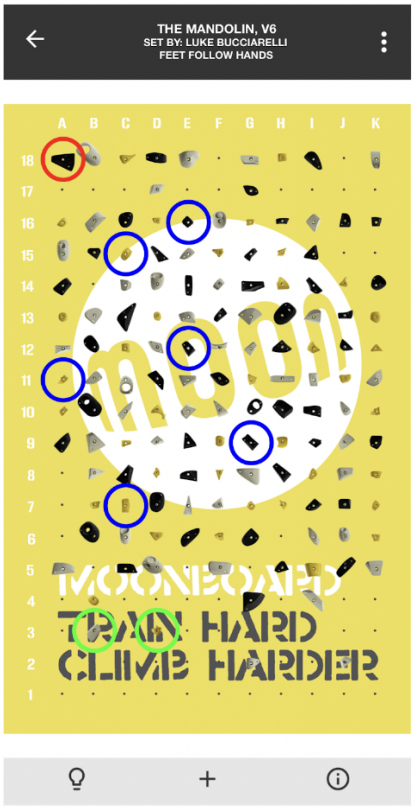


Fig 4: Example of a MoonBoard route as displayed on the app.

1.2.2 Legal Routes

A route can only be logged into the MoonBoard database only if it adheres to the following rules:

1. A route must have at least 1 starting hold, 1 intermediary hold, and 1 finishing hold.
2. A route has a maximum number of 14 holds.

3. A route has at most 2 starting holds. A starting hold must be on row 6 or lower.
4. A route has at most 2 finishing holds. A finishing hold must be on row 18 (the final row).
5. An intermediary hold must be located under row 18.
6. A hold's location must have an actual physical hold. See Fig 3 for some locations with no hold.

These rules should be considered during route generation to ensure new routes are legal.

2. Problem Statement

Research Question:

How can we use machine learning to approximate human-level classification of climbing grades and generate new, enjoyable routes of a given grade?

2.1 Capstone Motivation

There are several motivations for me to pursue this project. Personally, as a climber, I have felt frustrated by route-setters' subjective grading at climbing gyms when a route feels too hard for its given grade because of my build and reach, which makes me wonder if technology can be used to grade routes more objectively while considering differences in climbers' strengths and builds. Secondly, I have always been fascinated by route-setting, which involves designing climbing routes by fixing holds onto a wall. The process engages a high level of creativity that I hope to replicate using algorithms.

Additionally, based on my research, my work can create value for the climbing community in the following ways:

Firstly, climbers can benefit significantly from a system that objectively and accurately grades the difficulty of climbs. Routes that are poorly graded can often pose challenges for climbers, especially those whose training relies heavily on the MoonBoard. Accurate grading can make it easier to track progress, find climbs that fit training goals, and generally facilitate a more enjoyable experience.

Secondly, users can easily design new routes and upload them to the MoonBoard app. However, most climbers do not have enough experience to grade their difficulty accurately. This process can instead be streamlined with the aid of technology in two ways: when a user designs a new route, they can reference a suggested grade from the model(s); when a user does not want to go through filtering and browsing the app for a new climb, they can simply generate a new climb of their desired difficulty. Such a process would also lower the bar for those interested in route-setting on the MoonBoard.

Thirdly, despite the MoonBoard hosting more than 56,000 problems, less than 2% are of grades V10 and above. Although there are drastically fewer climbers at this level, they could benefit from a more comprehensive collection and styles of routes to try and train with, especially since the challenging nature of the MoonBoard makes it a vital training tool for professional climbers.

Lastly, I want to use this project to explore some of these questions:

- Are machines able to understand and replicate the structure of a route? Are they able to generate creative and fun routes in a similar manner to humans?
- Can route-setting be boiled down to rules, heuristics, and patterns, at least on the MoonBoard?
- Can we draw inspiration from MoonBoard routes to be applied to the problem space of general indoor route-setting?
- Is there such a thing as “objective grading?” Are machine-predicted grades truly more accurate than human-predicted grades?

3. Related Research

3.1 Non-Machine Learning Attempts

The earliest research on this topic was published in 2012 by Phillips et al. [4], an application of chaotic variations to create novel indoor rock-climbing routes, with the resulting variation generator named Strange Beta. The software aims to assist human route-setters in creating new routes with chaos to introduce novelty, and benchmarks showed that the software could assist setters in producing high-quality routes that sometimes exceeded those produced in a traditional manner. However, the authors' approach significantly differed from this project's, as the software's goal was not to fully generate routes but to *assist* human route-setters.

Phillips et al. was the only study using general indoor climbing routes. After the introduction and growth in popularity of the MoonBoard, studies in subsequent years, starting in 2017, all used the MoonBoard database as an approximation. The evolution of indoor climbing to use larger holds in various shapes and incorporate dynamic, parkour-like movements also contributed to the preference for a standardized training board since indoor climbing routes have become harder to generalize.

All other research has used machine learning save for Stapel (2020), who used a greedy algorithm with heuristics based on existing climbing routes to generate multiple routes and choose the best one(s) for the user based on their input preferences. The algorithm performed well on grade classification but produced routes that were enjoyed by only 40% of climbers surveyed, which is lower than Benchmark routes. The routes were straightforward and lacked flow since the heuristics only considered the simple “normal move.” The study's use of heuristics

and classification of the problem space can be combined with machine learning methods to understand the features of a route better.

3.2 Grade Classification

Dobles et al. (2017) [6] applied three simple models to the grade classification problem: Naive Bayes, softmax regression, and a convolutional neural network (CNN) with an ordinal output. All models produced ~35% accuracy but lagged behind human classification performance. CNN showed the best combination of accuracy and similar reproduction of underlying distribution, measured by KL-divergence (Fig 5). The imbalance in classes for the training data posed a significant challenge for the classifiers, as over-prediction of more common categories is inherent to Naive Bayes and softmax classifiers.

	User Ratings	Naive Bayes	Softmax	CNN
Accuracy	93.4%	34.0%	36.5%	34.0%
MAE	0.12	1.73	1.37	1.40
KL-divergence	0.0071	0.41	0.078	0.020

Fig 5: Dobles et al. (2017) results.

Tai et al. (2020) 's [7] Graph Neural Networks approach adapted the Text GCN framework used in document classification and showed significantly improved results (Fig 6), reaching roughly 73% average AUC. The only GCN model featuring four convolutional steps yielded the best averaged F1 and AUC scores. However, it's worth noting that logistic regression didn't lag far behind, with 70% AUC. An important finding from this study is that GCNs are less susceptible to data imbalance than baseline models such as logistic regression. The authors'

decision to only report F1 and AUC makes it difficult to compare their results with other attempts, which use accuracy as the primary metric.

Table 1: Summary of experiment results with per-class F1 scores and averaged accuracy, F1, and AUC scores

	Per-Class F1 Scores							Avg.	
Experiment	**V4**	**V5**	**V6**	...	**V12**	**V13**	**V14**	**F1**	**AUC**
Logistic Regression	0.52	0.35	0.25	...	0.22	0.00	0.00	0.23	0.70
SVM	0.53	0.40	0.31	...	0.34	0.33	0.00	0.29	0.66
Random Forest	0.66	0.36	0.26	...	0.24	0.00	0.44	0.28	0.67
Gradient Boosting	0.59	0.35	0.28	...	0.27	0.25	0.00	0.27	0.62
MLP	0.58	0.38	0.34	...	0.00	0.00	0.00	0.22	0.66
Dense Shallow, MH	0.61	0.38	0.32	...	0.37	0.00	0.00	0.26	0.67
Dense Deep, MH	0.48	0.40	0.23	...	0.21	0.00	0.00	0.22	0.65
GCN (S) 2S, OH, PMI	0.36	0.28	0.22	...	0.48	0.24	0.10	0.24	0.63
GCN (L) 2S, OH, PMI	0.33	0.27	0.23	...	0.48	0.25	0.13	0.24	0.64
GCN (S) 2S, MH, PMI	0.57	0.38	0.33	...	0.49	0.00	0.00	**0.29**	**0.73**
GCN (L) 2S, MH, PMI	0.58	0.38	0.33	...	0.45	0.00	0.00	**0.29**	**0.72**
GCN (S) 2S, MH, Binary	0.54	0.38	0.33	...	0.46	0.00	0.00	**0.28**	**0.72**
GCN (S) 2S, MH, Self-Binary	0.58	0.41	0.30	...	0.14	0.00	0.00	0.25	0.70
GCN (S) 2S, MH, Self-PMI	0.59	0.38	0.29	...	0.56	0.00	0.00	0.27	0.68
GCN (L) 4S, MH, PMI	0.52	0.34	0.33	...	0.50	0.00	0.37	**0.31**	**0.73**
GCN (S) 2S, MH, Win-PMI	0.55	0.37	0.33	...	0.48	0.00	0.00	**0.28**	**0.73**
GCN (L) 2S, MH, Win-PMI	0.57	0.38	0.34	...	0.47	0.00	0.00	**0.29**	**0.72**

Fig 6: Results from baseline experiments and graphical convolutional network by Tai et al. (2020)

Most recently, Duh, Y.-S., & Chang, R. (2021) [8] applied Recurrent Neural Networks, specifically Long Short Term Memory (LSTM) to both classification and generation, producing results outperforming previous attempts (Fig 7). The models are trained with “BetaMoves” — a move preprocessing pipeline developed by the authors that mimic a climber’s hand sequence. The authors describe the complex pipeline as “injecting human insights.” However, their model still suffers from the training data’s class imbalance as it underestimates the difficulty of harder routes. Their highest classification accuracy on the test set achieved 46.7%, which is only slightly higher than human-level performance.

	HLP	GradeNet Training	GradeNet Dev set	GradeNet Test set	Naive RNN Dev set	Ref[1] CNN	Ref[2] Graph NN	Ref[3] MLP
Accuracy	45.0%	64.3%	47.5%	46.7%	34.7%	34.0%	Not reported	35.6%
±1 accuracy	87.5%	91.3%	84.8%	84.7%	Not evaluated	Not reported	Not reported	74.5%
F1 score	-	0.506	0.242	0.255	0.165	Not reported	0.310	Not reported
AUC	-	0.898	0.764	0.773	Not evaluated	Not reported	0.73	Not reported

Fig 7: Performance 'GradeNet,' the classifier built on Chang & Duh's use of RNN, compared to other studies.

3.3 Route Generation

The two most popular methods for route generation are Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), but validating the generated routes is difficult due to the need for experienced climbers to attempt them. This has led to a heavy reliance on the authors' judgment or a small sample of climbers to validate the models.

Variational Autoencoders (VAEs)

In 2019, van den Brand [9] applied both GAN and VAE to route generation on the MoonBoard and concluded that VAEs performed better while GANs suffered from mode collapse. His VAE setup included eight layers and 50 nodes for the latent space. According to the author, the implementation produced promising results — that the model performed within acceptable bounds to human-generated routes. However, the validity of this statement is questionable. Among the generated routes, many seem to be "unclimbable," having holds too far apart or virtually impossible to catch; some are also "illegal," as in not following MoonBoard rules, for example, having three finish holds. Additionally, there were only seven respondents to the route validation survey, an extremely small sample to draw conclusions. One interesting idea

proposed by van den Brand is to adjust the difficulty of a route while retaining its structure and style by manipulating the threshold used in the VAE's output matrix.

Lo's implementation of VAE [10] used the 2017 setup on the MoonBoard. Lo used only 16 nodes for the latent space. Of the 50 problems generated by the model, 22 satisfied the author's requirement and adhered to the MoonBoard's route rules. The results were promising for a relatively simple model, and the author concluded that the model picked up on existing patterns and rules. However, the paper did not describe any validation other than the author's visual evaluation. A more in-depth comparison between machine versus human-generated routes is needed to evaluate the quality of the routes properly.

Generative Adversarial Networks (GANs)

Geary and Valdez (2019) attempted to apply an Auxiliary Classifier Generative Adversarial Network (ACGAN) to build a route generator. Despite reaching some amount of convergence, all of their models suffered from mode collapse. That is, they all produced nearly-identical output. Geary and Valdez suggested that adjusting the hyper-parameters of the algorithm may improve the results and did not validate the generated routes with any climbers. This confirms Van den Brand [9]'s result that GANs performed worse than VAEs, but it may still be a viable model with different parameters.

Recurrent Neural Network

Chang and Duh's (2021) work [8], as mentioned previously, used "BetaMoves" to train an RNN for route generation. The authors compared their generated routes with ones from Houghton et al. (2020)'s project [9], which shows a drastic improvement in feasibility, quality,

and enjoyability. However, the significance of their improvement may be questioned, as the conclusion was based on the evaluation of a single "expert climber," and the authors only display two generated routes, which could be the best ones cherry-picked out of many lower-quality routes.

	total problems evaluated	redundant/ holds	weird sequence/ ugly moves/ lack of flow	reasonable problems	high quality problems
Latest MoonBoard problems	50	10%	6%	84%	60%
Deep RouteSet	40	0%	5%	95%	80%
Model Ref[3]	40	35%	35%	40%	20%

Fig 8: performance of DeepRouteSet, the generator from Chang and Duh's application of RNN, compared to routes generated by Houghton et al.

4. Data

4.1 Cleaning

MoonBoard's website displayed all problems to users before an update in 2021, which allowed prior projects to use a web scraper to gather information from the complete database. Since that is no longer possible, I used the data scraped in 2020 by user jrchang612 [12] for an attempt with recurrent neural networks with the authors' permission [8]. This dataset was scraped using the web scraper developed by user gestalt-howard [13] for a GNN attempt.

The raw dataset contains 30,642 user-generated routes. I performed the following steps to clean the data:

1. Remove all climbs with grade 8B+ / V14 (24 problems) since they are a small class and most routes are clearly mislabeled (e.g., obviously an easier climb with many holds).
2. Remove all illegal climbs with more than 2 starting or finishing holds.
3. Remove all climbs with no repeats since their quality hasn't been validated by any user from the community, usually indicating lower quality and/or questionable grading.

After cleaning, the dataset contains 25,099 routes.

4.2 Exploratory Data Analysis

MoonBoard allows users to use both V Scale and Font Scale when creating and grading routes. I converted the routes in the dataset from Font Scale to V Scale since the V Scale has fewer classes and can reduce ambiguity (both 6C and 6C+ in the Font scale map to V5 and both 7B and 7B+ map to V8). After grade conversion, the routes range from V4 to V13.

The distribution of grades (Fig 9) is heavily skewed towards the easier end of the scale, with V4 and V5 routes accounting for over 50% of all routes and V10-V13 accounting for less than 2%. This distribution is representative of climbers' skills. In terms of successful repeats — the number of times that users have successfully climbed a route — V4 and V5 route account for more than 60% of all repeats, and V10-V13 account for less than 1% (Fig 10). Such a class imbalance will pose a significant challenge for our models since they will have less exposure to harder routes and consequently find them more challenging to classify and generate.

Due to the small sample sizes, it is unlikely that the models will learn the distinctions between the higher grades of V10-V13. It is also possible that the grading for those routes is more arbitrary since they have been validated by fewer people. Therefore, I combined the V10-V13 classes into one V10+ class, which now accounts for 1.5% of all routes. This modification will help with reducing the severity of the class imbalance.

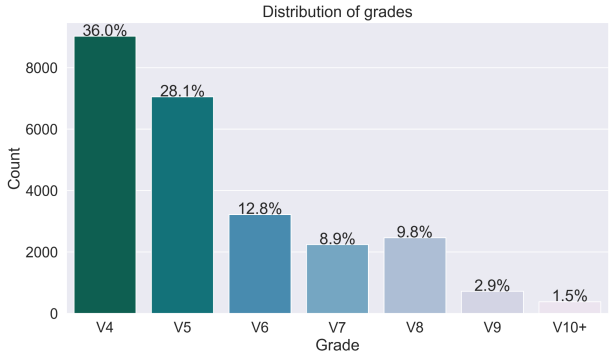


Fig 9: Distribution of grades.

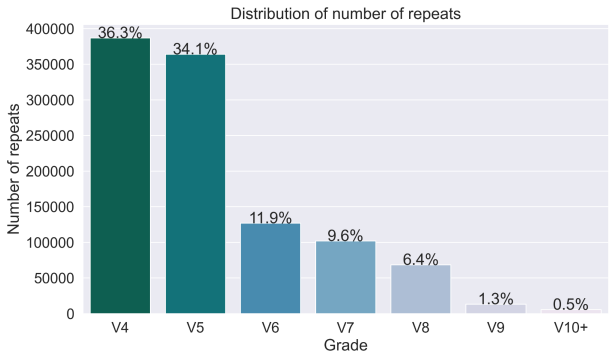


Fig 10: Distribution of the number of repeats.

4.2.1 Heuristics

Two heuristics can be considered for classification and route generation:

Heuristic 1: number of holds.

Fig 11 shows the distribution of the number of holds for each grade. For easier routes of grade V4-V6, the average number of holds is roughly 8 holds per route. The distribution for V4 has a long tail to the right but less so for V6+ routes. It's not rare for V4 routes to have more than 10 holds but far less likely for harder routes. For harder routes of grade V8-V10+, the average number of holds is roughly 6-7 holds per route, with some having as few as 4 holds. Intuitively, having fewer holds makes a route more likely to be difficult. A principal components analysis on all routes showed roughly the same pattern along the first component (Fig 12), suggesting that the number of holds is the most significant component that connects a route to its grade.

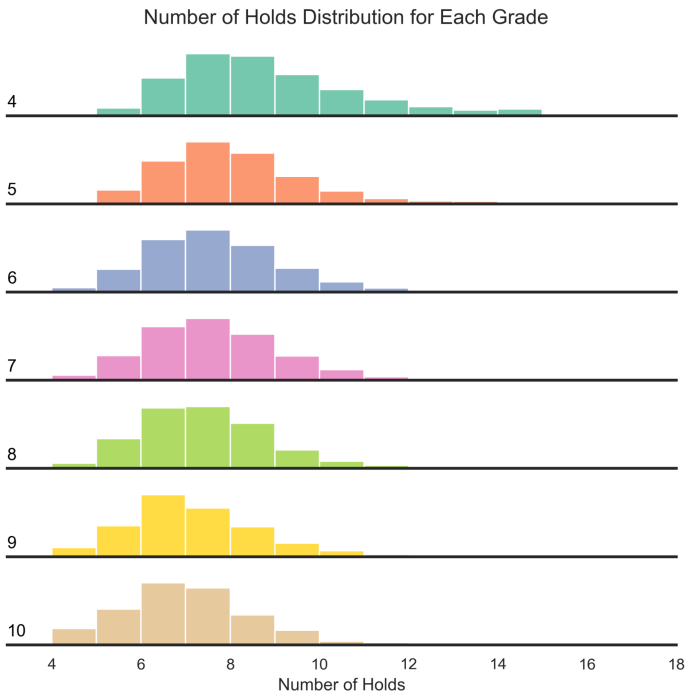


Fig 11: Distribution of the number of holds for each grade.

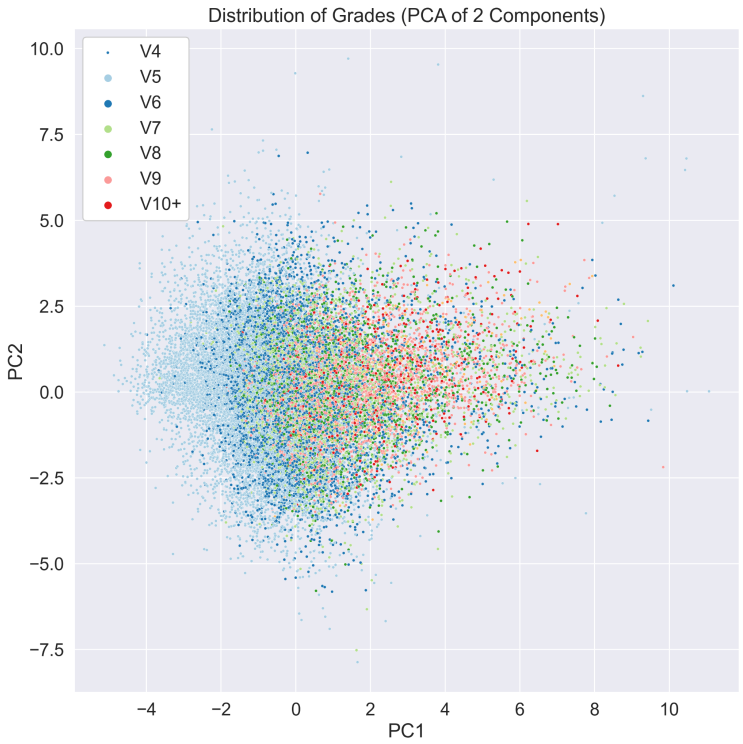


Fig 12: Principal component analysis of routes with grades as classes and two principal components.

Heuristic 2: type of hold.

The MoonBoard has holds of various types and sizes, some easier than others. For example, 13G is relatively larger and positive, making it a “jug” (Fig 13 right). In comparison, 15F is barely a nib, and a climber would be lucky to fit more than two finger pads on the hold (this holds would be considered a “crimp”). We can study which holds are used more frequently in each grade range using heat maps (Fig 13 left). The difference between 13G and 15F is evident in how frequently they are used in routes of grades V4-V5 to V10-V13. Interestingly, the three grade ranges differ significantly in their choice of holds. For example, 2G is included more frequently in V4-V5 routes as a foothold, whereas in harder routes, the climber may be required to “cut feet” and “campus” (not using footholds for support) by removing 2G. Popular holds in

V4-V5 routes are almost *never* present in V10-V13 routes since the introduction of easier holds would downgrade the route. In conclusion, the frequency of hold usage in routes of various difficulties can be a useful heuristic for determining the difficulty of a route.

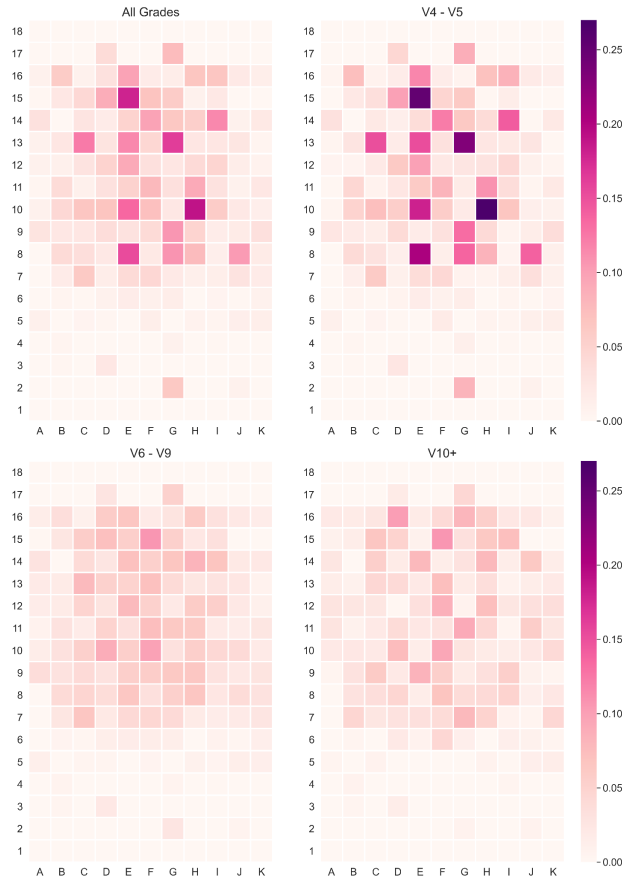


Fig 13 (left): Heat map of use frequency of holds (excluding start and finish holds).

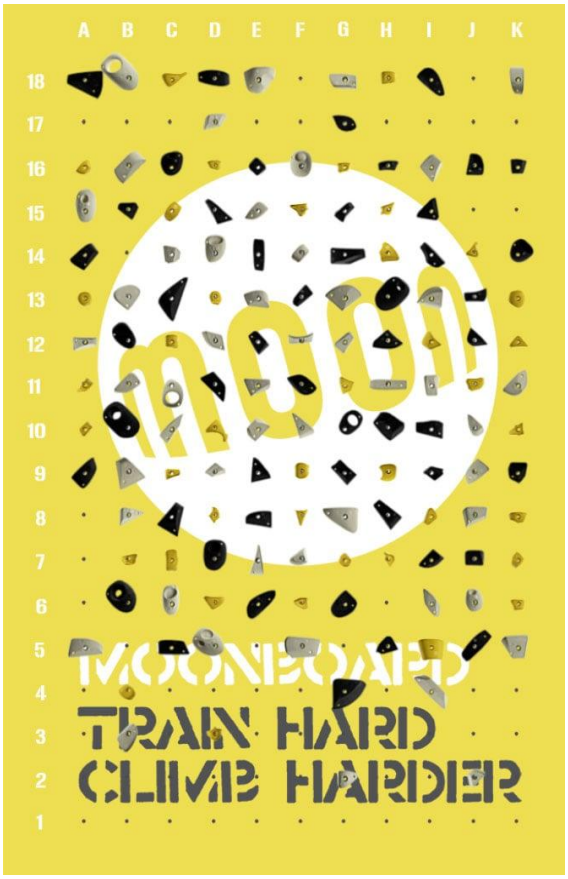


Fig 13 (right): MoonBoard 2016 setup and holds.

4.3 Data Representation

Routes have differing numbers of holds. Therefore a coordinate-list representation would have variable length. Our classification and generative models require fixed-length inputs. For a generalizable representation, every route is one-hot encoded to use as input for the models, as

shown by the example with the route below. 1s represent holds that are on, 0s represent holds that are off.

Each route can be represented using an 18×11 matrix or an array of size 198 reshaped from the matrix (Fig 14). In the sections below, we will specify the representation used for each model.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig 14. One-hot encoded representations of routes.

5. Route Generation

We use variational autoencoders (VAEs) for route generation, for it has shown the most promising results among existing literature [9, 10]. Generative adversarial networks (GANs) were considered but not pursued since both attempts by Geary & Valdez [11] and van den Brand [9] encountered mode collapse.

5.1 Variational Autoencoders (VAE)

A variational autoencoder is a generative model that combines deep learning with variational inference to learn a compressed representation of data (via the encoder) and generate new data from that representation (via the decoder). It was first proposed by Kingma et al. (2013) [14]. VAEs learn a latent space of data, meaning that they are able to represent data points in a low-dimensional space and generate new data points from the same space. This makes them useful for tasks such as image generation, text generation, and dimensionality reduction.

The network consists of an encoder and a decoder. The encoder $e_\theta(x)$ maps the inputs data x to variables z from a probability distribution (chosen to be a Gaussian distribution represented using (μ_x, σ_x) for each latent variable. The decoder $d_\phi(z)$ takes samples $z \sim \mathcal{N}(\mu_x, \sigma_x)$ from this distribution and maps it to an output x' with the goal for it to be as similar to the original input as possible (Fig 15).

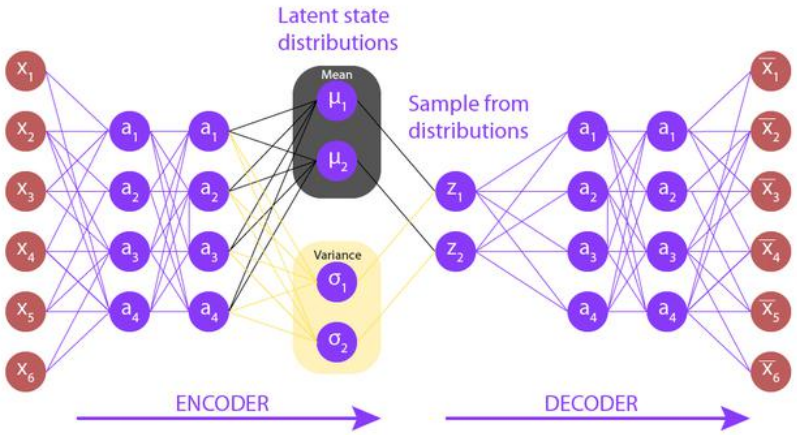


Fig 15. VAE structure

To generate new data samples, we simply sample from the latent distribution and run it through the decoder network.

Loss Function

The loss function of a VAE consists of two parts: a reconstruction loss, which measures how well the VAE was able to reconstruct the input data, and a similarity loss, which measures the complexity of the latent space.

The ***reconstruction loss*** typically measures the distance between the input data and the reconstructed data. Just like in a normal autoencoder, it is the mean squared loss of the input and reconstructed output. The reconstruction loss is used to ensure that the VAE learns to encode and decode the input data accurately.

The ***similarity loss*** is the Kullback-Leibler (KL) divergence between the latent space distribution and standard gaussian (zero mean and unit variance). Minimizing it encourages the learned distribution to be close to the known distribution, which helps prevent the model from

overfitting to the training data. It also helps ensure that the latent space is smooth and continuous, which can be useful for generating new data samples. The Gaussian distribution is chosen because it is a convenient and flexible distribution that can capture a wide range of data. It also has a simple mathematical form that allows for efficient computation. Additionally, the Gaussian distribution is easy to sample from, which is essential for generating new data samples from the learned latent space.

$$\begin{aligned}
&\text{reconstruction loss } = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi\left(\mu_x + \sigma_x \epsilon\right)\|_2 \\
&\quad \mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&\text{similarity loss } = KL \text{ Divergence } = D_{KL}\left(\mathcal{N}\left(\mu_x, \sigma_x\right) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\right)
\end{aligned}$$

$$\text{loss } = \text{ reconstruction loss } + \text{ similarity loss}$$

During training, the model aims to achieve equilibrium between the two losses, ultimately resulting in a latent space distribution that resembles a unit norm with clusters that contain similar input data points.

5.2 Model Architecture

Input

Each route is represented with an array of size $(18,\ 11, 3)$, made up of three 18×11 binary matrices. Each route's start, intermediate, and finish holds are one-hot encoded separately into each matrix.

For example, a hold that is off: $(0, 0, 0)$

A starting hold: $(1, 0, 0)$

A middle hold: $(0, 1, 0)$

A finish hold: $(0, 0, 1)$

In the final implementation, we train a model for each grade level. For example, a V5 model is only trained on V5 routes as inputs so that its outputs resemble that grade. See 5.4 for details.

Network

The final implementation of VAE consists of 6 hidden layers in total (3 for the encoder and 3 for the decoder) and uses 50 latent variables. As previously discussed, the input and output size is $(18, 11, 3)$, requiring the one-hot-encoding of the route to separate start, mid, and finish holds into three separate channels, then be flattened into a binary array with 594 numbers. The network structure can be found in Fig 16.

Outputs

The outputs of this network are in the shape of $(18, 11, 3)$. However, the values in this array are no longer binary but continuous values in the range of $[0, 1]$. The bigger the number, the higher confidence the model has in including the hold in the route.

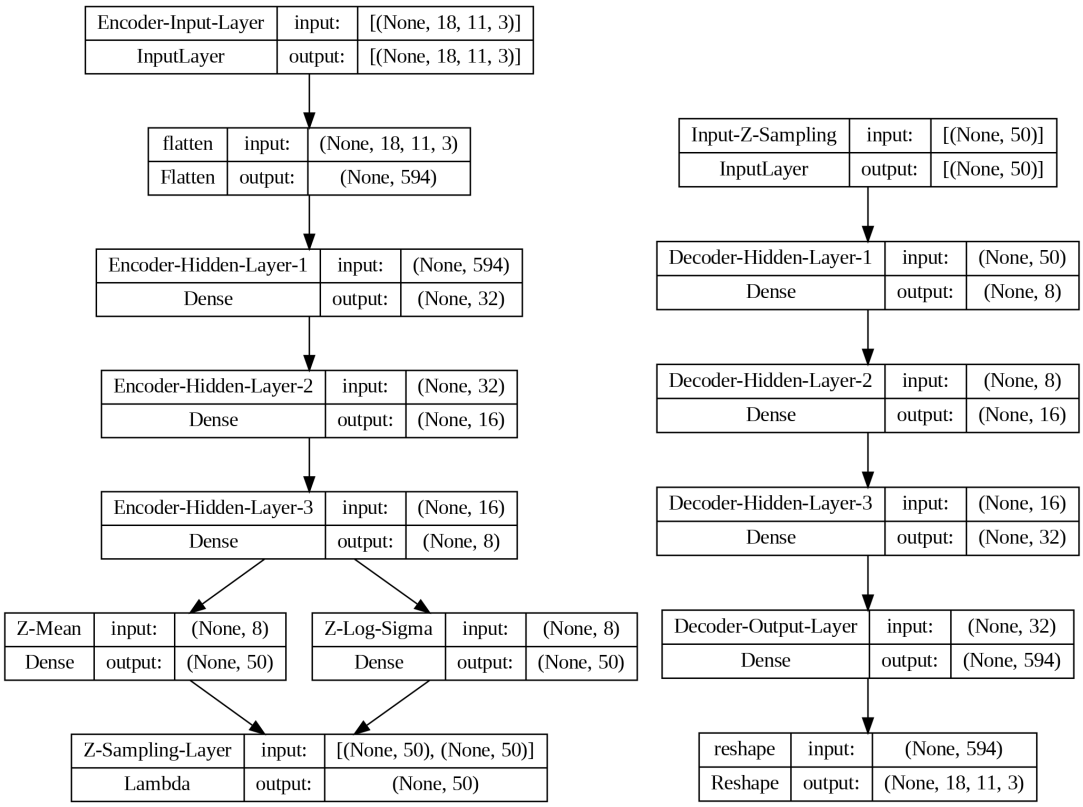


Fig 16. Left: VAE encoder; Right: decoder

5.3 Interpreting Outputs

The output format from our VAE differs from that of real MoonBoard routes in which a hold is strictly either 1 (legal) or 0 (illegal). Converting outputs into MoonBoard routes with binary representations poses a challenge. As discussed in section 4.2.1, the number and type of holds greatly determine the difficulty of a route. The VAE's outputs indicate the type of holds that should be included in the route — the higher a hold's number, the more confident we can be in including it in the route. Thus the final task is to determine how many start/intermediate/finish holds to include by filtering from each channel.

Too many holds make the route redundant; too few holds can make climbing entirely impossible. Additionally, an unfitting number of holds might cause the route to be not reflective of its grade and not feel “flowy” or elegant. The most appropriate number of holds is entirely determined by the climb itself and would require human judgment or complex computations that analyze the sequence of the climb. Acknowledging these difficulties, we decided to use a *heuristic*: randomly sample from the existing dataset of all climbs of the desired grade and use the same number of holds. We can use an example to illustrate this process. After we generate a route supposed to be V5, we take a sample from the distribution of all V5 climbs and filter the output using the same number of holds. If we were to generate a large number of V5 climbs, then the distribution of their number of holds would resemble that of the original V5 dataset.

For intermediate holds, we always take the holds with the highest values. However, for start and end holds, this method would lead to generated routes of a certain grade almost always having the same setup for start and end. As displayed in Fig 17, all V6 routes generated using the same model invariably use hold F5 as a start hold and D18 as a finish. This outcome is to be expected: since these holds are very popular among the training samples, the model almost always gives them the highest predicted values.

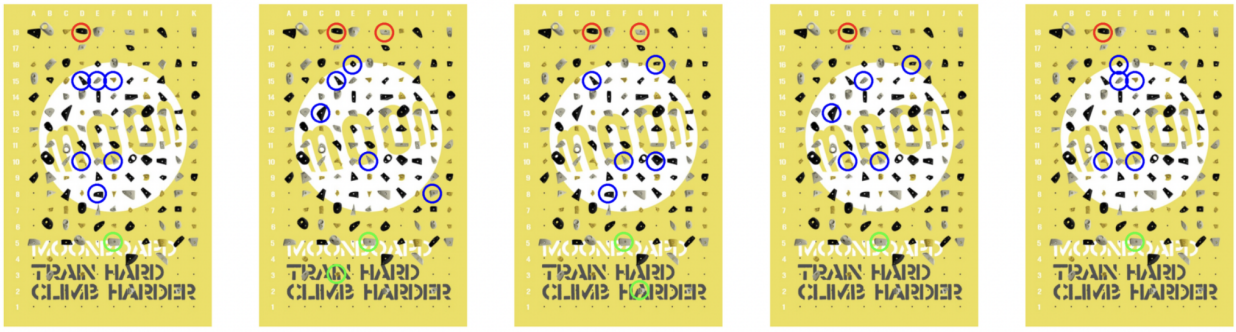


Fig 17. Generated V6 routes with a simple heuristic of choosing start/finish holds with the highest values.

We can prevent this issue with the following strategy: if we were to choose n start/finish holds, instead of directly choosing the n holds with the highest number, we find more than n and randomly select n holds from that pool. As MoonBoard routes typically have 1 or 2 start holds and 1 finish hold, we choose them from 3 potential start holds and 2 finish holds respectively. The introduction of randomness to the algorithm adds more variability to our start/finish holds, as reflected by the generated V6 routes in Fig 18.

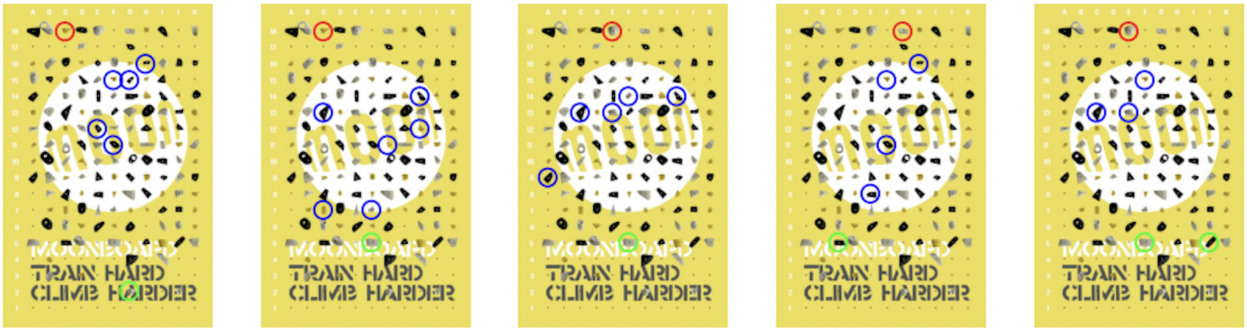


Fig 18. Generated V6 routes with improved heuristic for start/finish hold selection.

The output format makes an additional feature possible: we can change the number of holds selected to change the difficulty of a route but retain its general style, as shown in Fig 19. In this way, a climber has more flexible options when climbing a single route, for example, adding more holds to use as a warm-up or taking away holds to experiment with more challenging moves. We did not include this feature in our final presentation of generated routes, but it may serve as an interesting extension for future work.

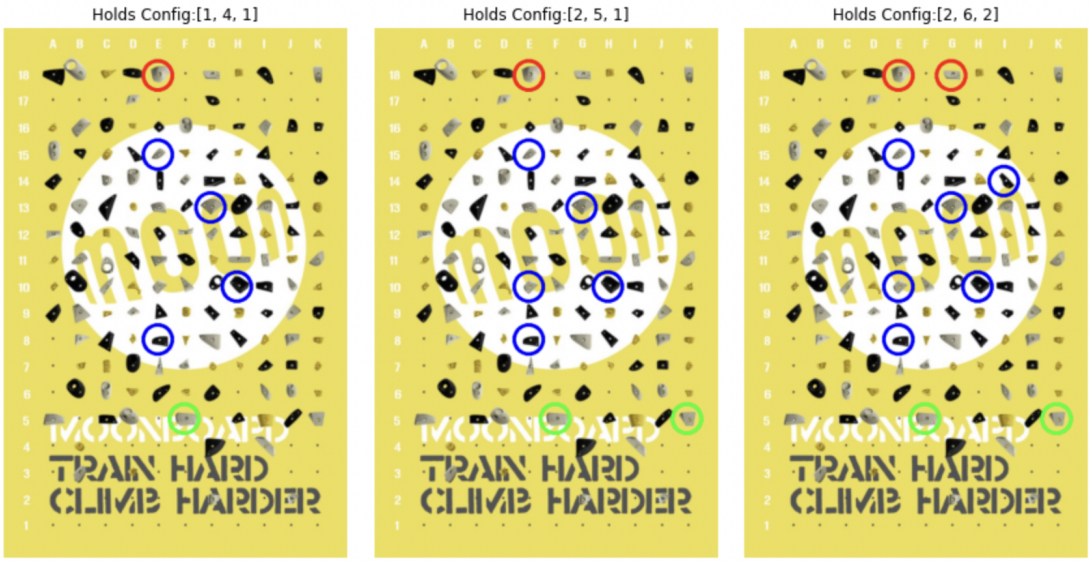


Fig 19. Changing the number of holds for a single output to create three routes of a similar style.

5.4 Experiments

I ran several experiments to find the ideal architecture and training method for VAEs.

Flattened input versus three-channel input

As mentioned in 5.1.1, we use a three-channel representation so the VAE can learn specifically about a route's start, intermediate, and finish holds. It helps to ensure that the generated routes are legal — as in, the start and finish holds are within two holds and on the correct rows.

In my experiment, I found that the model wouldn't necessarily learn those requirements if we simply used a flattened 18 by 11 one-hot-encoded matrix as input. Fig 20 shows an illegal generated route (it has more than two finish holds). Using separate channels for each set of holds solves this problem, as we can select start, intermediate, and finish holds independently.

Fig 20. An illegal route that is generated from flattened input.

Training with all data versus training by class

Due to the overrepresentation of V4 and V5 climbs in the dataset, when the model is trained on ***all*** data, the generated routes represented almost solely V4 climbs and share the same style. The model never generates any challenging routes.

We decided to train 7 models, one for each grade level, using only routes of that grade. This design adds a feature to the generator: we can generate a route of a designated grade. In contrast, all previously published attempts trained models using all data and then generated random routes. If a user wanted a route of a specific grade, the model would keep generating routes until the classifier predicts one of them is of that grade. Our method significantly increases efficiency and accuracy as we are more confident that a model trained on only routes of a particular grade will generate routes that resemble that grade.

The grade-specific models can generate higher-grade routes (Fig 21 shows an example of a V9 route that is evidently more difficult than previously generated routes with a more challenging selection of holds), yet for lower-grade routes, the outputs still end up converging to look similar.

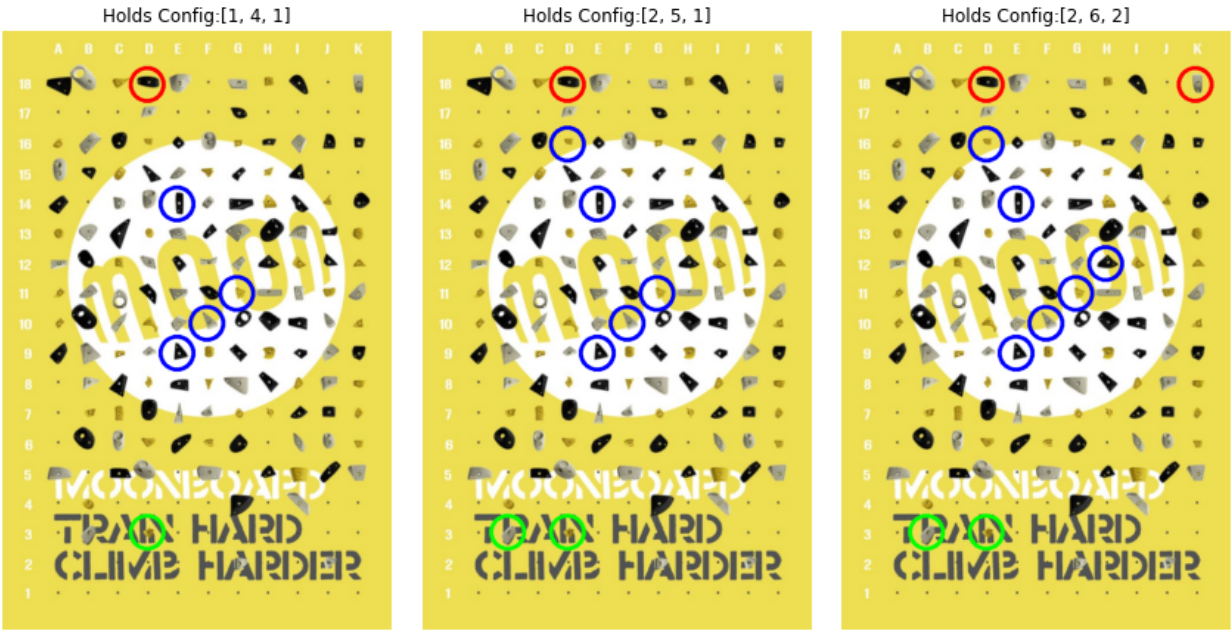


Fig 21. Generated novel V9 route from a generator trained only on V9 routes.

Training with all data versus training in small samples

Our hypothesis for the model's tendency to converge for lower grades is that it struggled to learn patterns from a sizable training dataset and thus simply learned the most frequently used holds. To test this hypothesis, we trained multiple models, each using a small sample of 200 routes. The results showed that each model generates different routes for the same latent variable input (Fig 22). Since each model is trained on a random selection of training data, we can theoretically get almost infinitely many models, allowing us to generate many routes that differ in style and show higher levels of creativity.

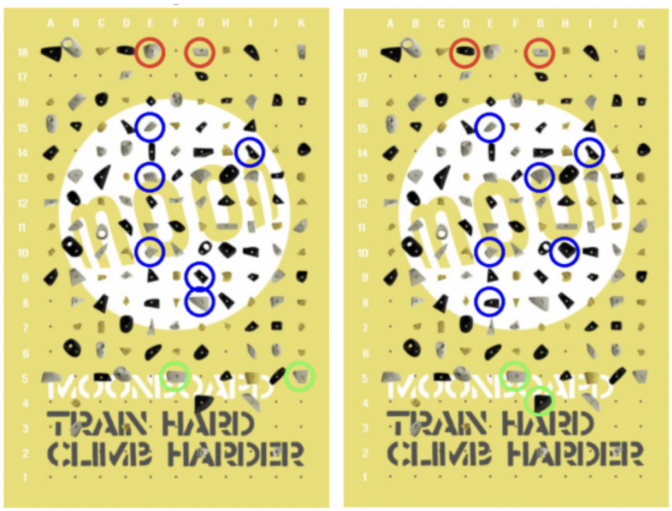


Fig 22. Small sample training outputs for the same latent vector. The left and right are produced using two different models, each trained on 200 routes respectively.

We also compared different numbers of samples to train on. We found that 100 samples are insufficient for the model to learn route patterns. Fig 23 shows a V4 route generated after training on 100 samples for 20 epochs. The model hasn't learned the rule that the finish hold must be on the 18th row. The models consistently produce legal routes when trained on 200 samples and don't show notable improvements in route quality as we increase the number of samples up to 500. Therefore we chose to train the models using 200 samples for efficiency.

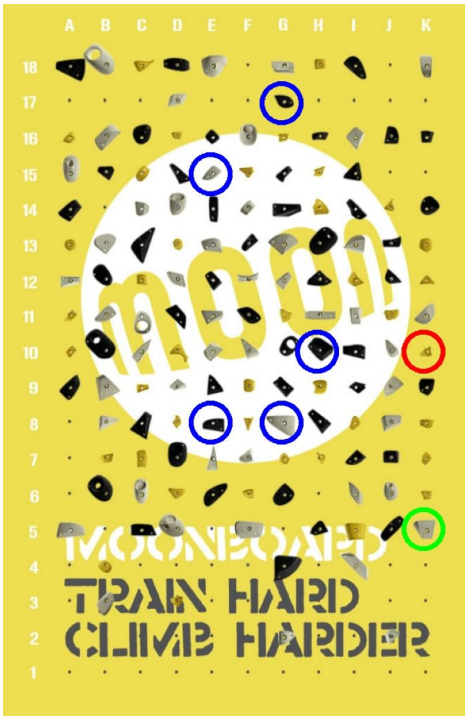


Fig 23. An illegal V4 route generated by a model trained on 100 samples.

Different dimensions for the latent space

After experimenting with various dimensions for the latent space, I found that with a higher dimension, the model learns more from the dataset. This observation intuitively makes sense since we are compressing the input data into more dimensions, thereby retaining more information. As Fig 24 demonstrates, when we only use two dimensions, the outputs look extremely similar since the input data was more compressed; when we increase the dimensions to 20, the outputs have far more variability. The improvements are less drastic when we increase the dimensions beyond 20, but we set the latent space dimension to 50 to generate high-quality outputs for our purposes.

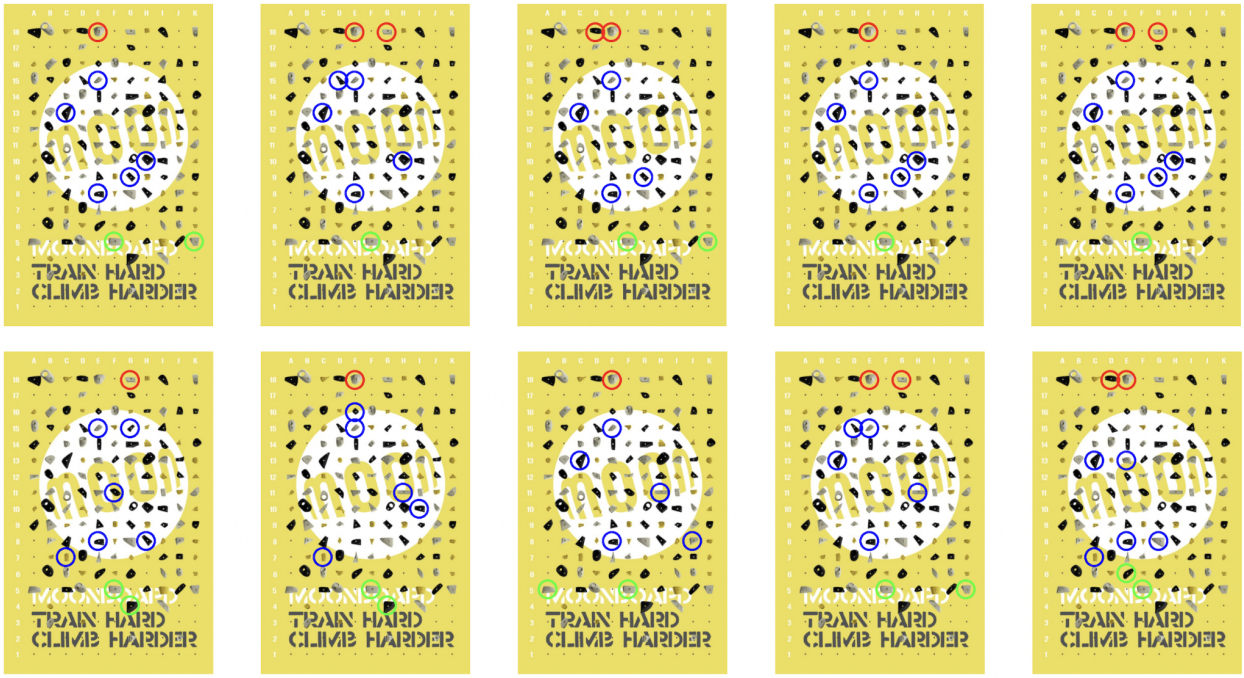


Fig 24. Top: Generated V5 routes with 2 latent dimensions; Bottom: Generated V5 routes with 20 latent dimensions.

Different numbers of epochs

Typically, the more epochs a model is trained for, the more it minimizes the loss function until it stabilizes around a level. Fig 25 shows the loss curves of training and test data loss curves as we train a V10+ model. The loss drastically decreases in the first 10 epochs, then the rate of reduction decreases until loss roughly stabilizes at around the 18th epoch, meaning training our model for more epochs than that may produce only marginal improvements in quality, if any.

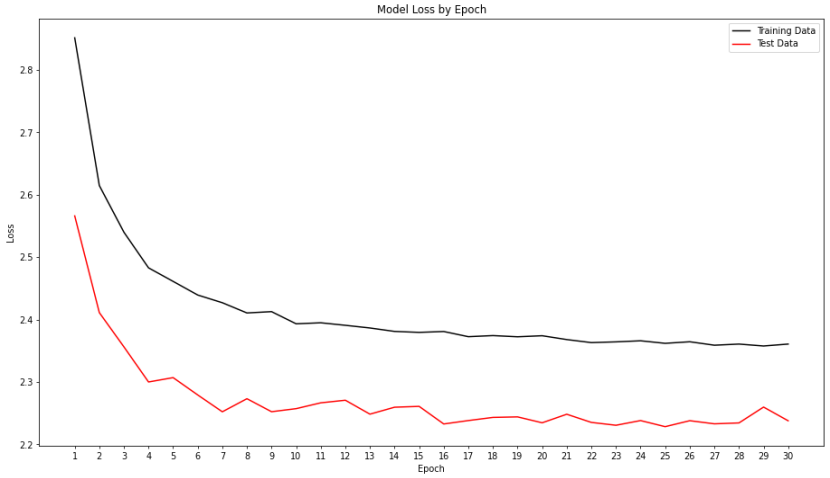


Fig 25. Loss vs. epochs, V10+ model

We experimented with generating routes after training with various numbers of epochs. As shown in Fig 26, the results confirm our guesses based on the loss vs. epochs graph. Routes generated after training for 5 epochs disregard MoonBoard rules and route patterns, and those generated after training for 10 epochs are legitimate routes, but some don't appear to be V6 routes. Routes generated after training for 20 epochs seem to have an almost similar quality to those with 50 epochs. To reduce training time while ensuring route quality, we train each model for 20 epochs.

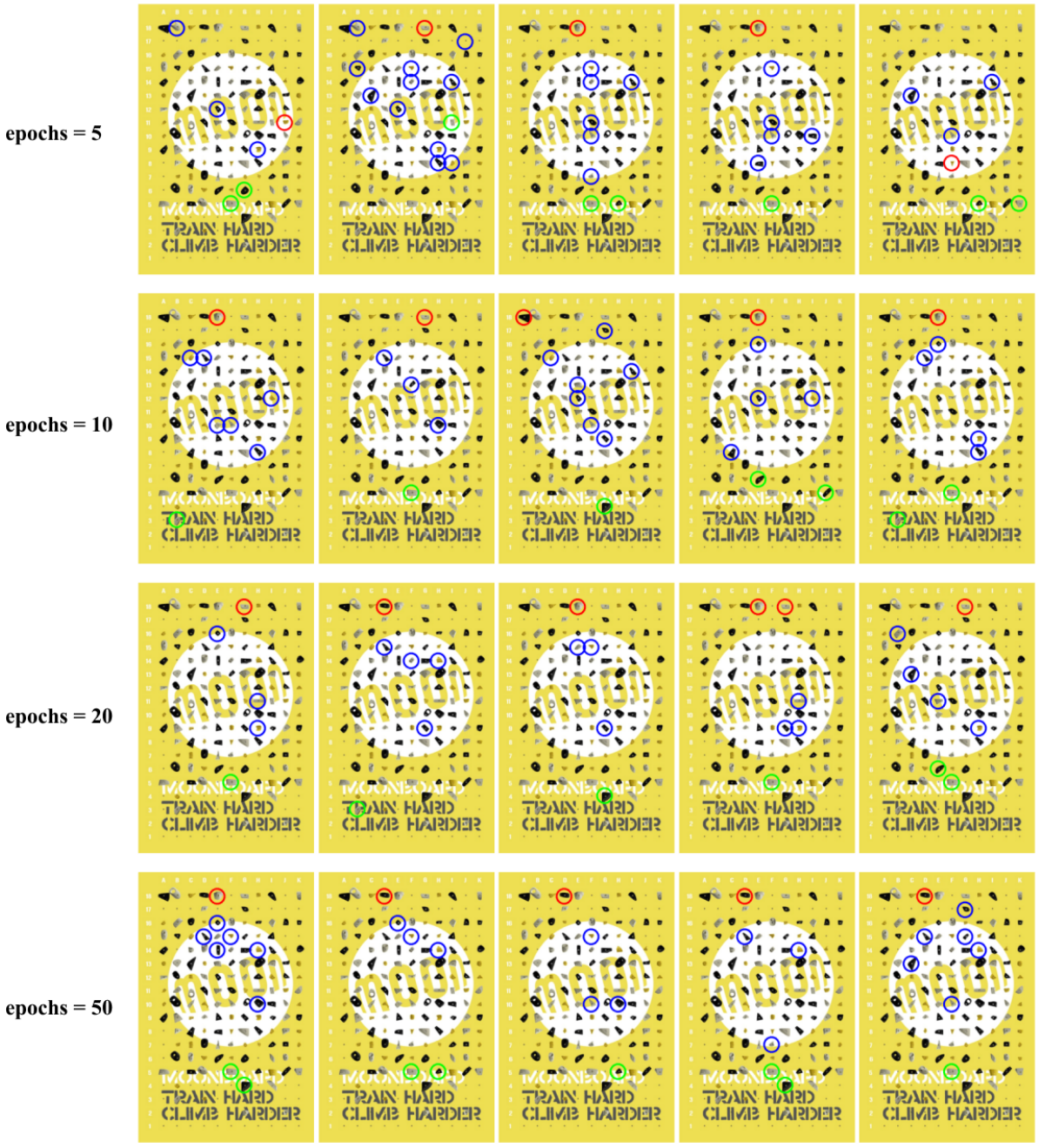


Fig 26. V6 routes generated after training for various numbers of epochs.

Experiments Conclusion

Combining our findings from these experiments, our final configuration is training a model for each grade level using small samples of 200 routes each for 20 epochs. This design satisfies our goal of producing legal routes that are accurate for their grade without taking an unnecessarily long time to train. We discuss our results in the following section.

5.5 Results

With the model architecture shown in Fig 16 and configuration concluded in 5.4, I trained a model for each grade and generated 6 routes each using randomly sampled latent vectors from the latent space (Fig 27).

All models successfully generated legal routes. Most of them seem to reflect their designated grade when judged by the number of holds and the difficulty of the holds used, which are the two heuristics we identified in 4.2.1 for evaluating a route's grade. We can observe that routes of higher grades (V6+) appear more creative and varied than easier ones. We believe this is due to two factors:

1. Easier routes can only choose from a limited set of holds since most holds on the MoonBoard are notoriously difficult to grab. The routes we can generate from these holds are thus limited.
2. The training data for easier routes show more obvious patterns than that for harder routes; for example, easier routes are more often concentrated in the center of the board and have holds closer to each other. The model learns these patterns easily and generates routes confined to these observations, whereas harder routes share fewer obvious patterns.

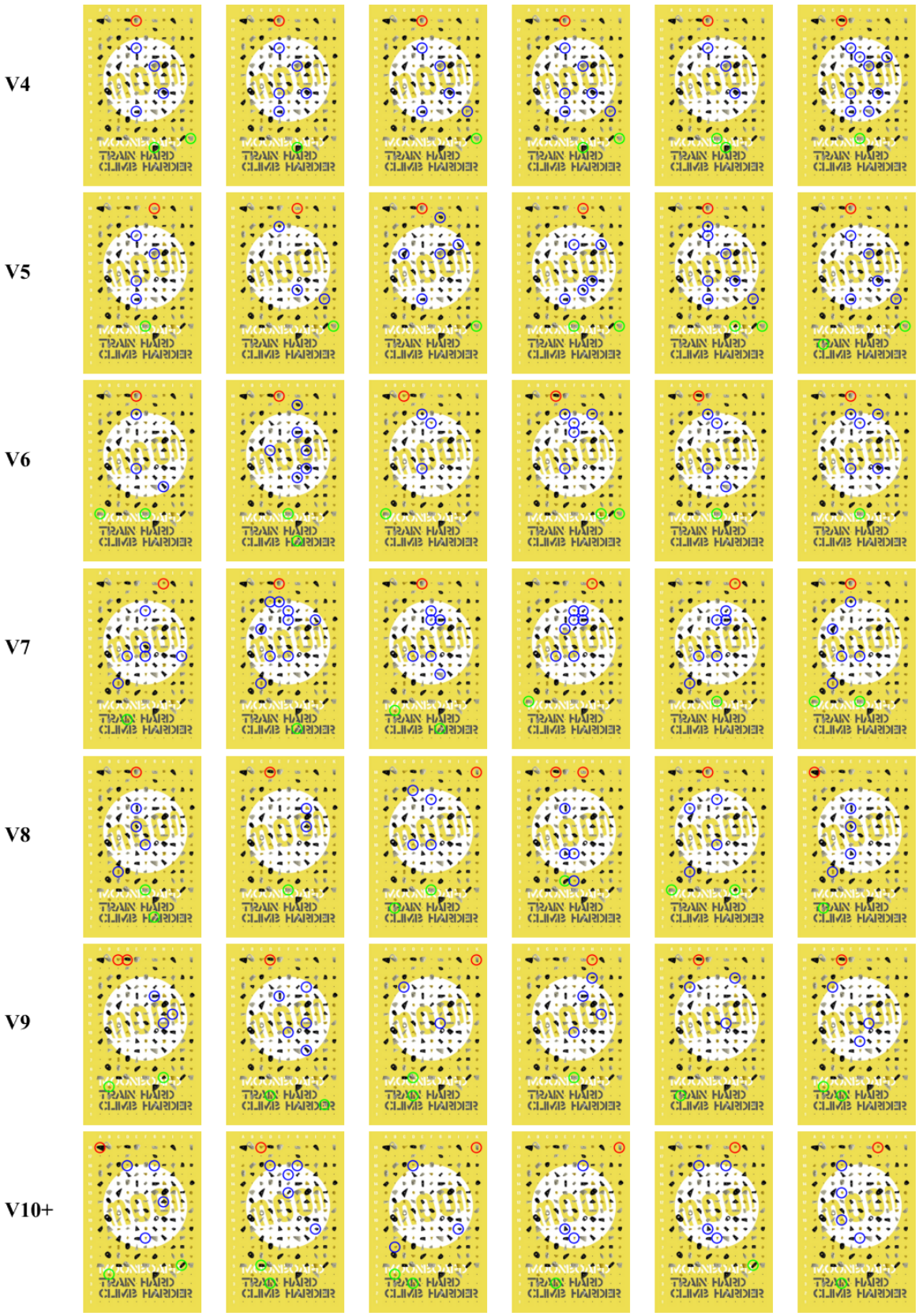


Fig 27. Generated routes of grades V4-V10+ with six routes for each grade.

With our small-sampling training approach, we can simply train multiple models using different samples for easier routes to compensate for this weakness. Each model may generate similar routes, but models will differ from one another.

Almost all routes appear climbable upon visual inspection. However, a few routes are not satisfactory, such as:

- The 3rd and 4th intermediate holds in the second V5 routes are too far apart. Perhaps the route would make sense if it were a much higher grade if it's climbable at all.
- The fifth V9 route is clearly unclimbable since the first move from the start to the small hold is too far.

Based on the generated routes' overall quality, we can conclude that our heuristic for determining the number of holds for each route is effective. It allowed us to avoid the additional layers of complexity of analyzing each route to determine the optimal number of holds. In other words, we could work with limited knowledge of the model's output and still produce a reasonable route. Approaching the start, intermediate, and finish holds independently also allowed us to *always* generate legal routes, a significant improvement from previous attempts with VAE in which a "threshold" is used, which does not guarantee validity.

However, the existence of routes of lower quality (unclimbable or astray from their grades) is also attributable to our heuristic. Sampling from the existing dataset's number of holds doesn't regard the generated route, and the relations between the chosen holds. Therefore, we sometimes get routes with holds too far apart — perhaps four holds were sufficient for one V10+ route in the dataset, but that does not imply we can simply use the same configuration for any randomly generated V10+ route (Fig 28 left). For future work, we can implement a system that

checks for the validity of generated routes: if holds are too close or too far, or the difficulty/orientation of the hold makes it impossible to move to, we change the number of holds in the route or generate a new one.

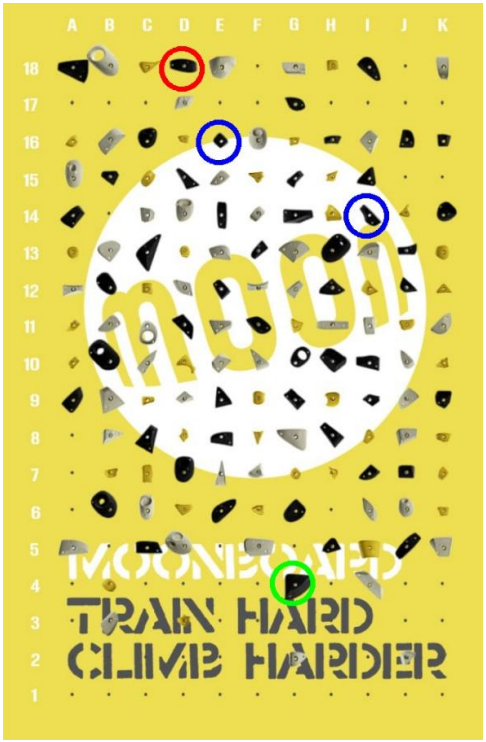


Fig 28 (left): Generated V10+ route with holds too far apart for it to be climbable.

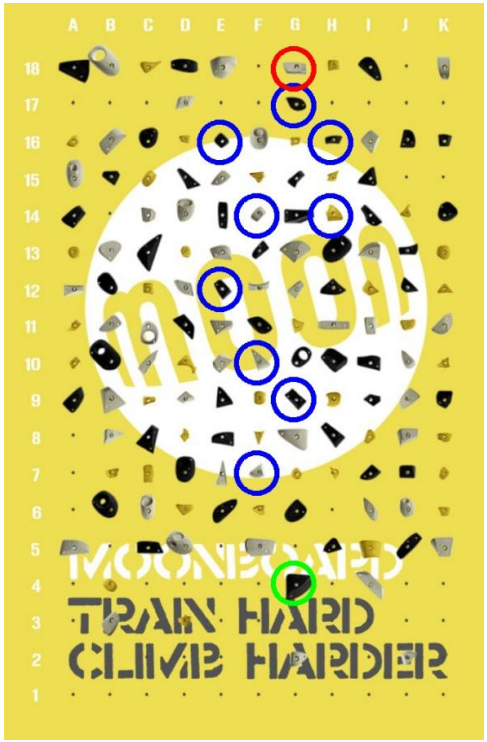


Fig 28 (right): Generated V6 routes with too many chosen holds, making it appear more like a V4 route.

6. Grade Classification

The goal of grade classification is to predict the grade of a given MoonBoard route out of 7 possible grades as accurately as possible. According to Duh, Y.-S., & Chang, R. (2021)[8], the human-level performance of this classification task (based on three climbing experts) is 45.0% accuracy for exact grade and 87.5% when allowed one grade difference.

As discussed in 4.2, the dataset is heavily skewed towards easier routes. The classifier will therefore find it challenging to accurately identify harder routes, as it won't be able to learn the decision boundary effectively. We will apply several classification models to the dataset to generate baseline metrics, then investigate two methods to handle class imbalance.

6.1 Metrics

To examine the classification results, we choose the following metrics:

1. ***Accuracy*:** describes the proportion of correct predictions the model makes compared to the total number of predictions. Also known as the recall.

   Accuracy = (Number of correct predictions) / (Total number of predictions)
2. ***Balanced accuracy:*** a variation of accuracy that takes into account class imbalance. It is calculated as the average of the recall (sensitivity) for each class.

   Balanced accuracy = (Recall for class 1 + Recall for class 2 + ... + Recall for class n) / n

   Where n is the number of classes.
3. ***F1 score:*** a metric that combines precision and recall to provide a balanced measure of a model's performance. Precision is a measure of how well a model identifies positive cases, while recall is a measure of how well it identifies all relevant cases.

   The F1 score is the harmonic mean of precision and recall, and it is calculated as:

F1 score = 2 * (precision * recall) / (precision + recall)

4. ***AUC (Area under the curve):*** The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for different classification thresholds. TPR is the same as recall, while FPR is the proportion of negative cases that are incorrectly classified as positive. AUC provides a measure of the overall performance of a model, and it is calculated by computing the area under the ROC curve. The AUC value ranges from 0 to 1, where a value of 1 indicates perfect performance and a value of 0.5 indicates random guessing.

Since we are interested in simply taking a route and accurately predicting its grade, as in our ability to make true positive predictions, the metric we are most interested in is accuracy (or recall). To study the effect of our methods to handle class imbalance, balanced accuracy is the most informative metric as it is not impacted by the bias caused by the overrepresentation of easier grades. If the model performs poorly for a minority class, it will be reflected in the balanced accuracy just as much as the majority class. Since the F1 score and AUC consider false positives, they will mainly be used to compare our baseline models to reflect their tendency to incorrectly predict minority classes as V4/5.

6.2 Baseline Results

We trained five classifiers on the original dataset — logistic regression, Naive Bayes, random forest, XGBoost, and a simple convolutional neural network (the CNN architecture can be found in Fig 29). Their results are shown in Table 1.

	HLP	Logistic Regression	Random Forest	Naive Bayes	XGBoost	CNN	Ref [7] GNN	Ref [8] GradeNet (RNN)
Accuracy	0.45	0.50	0.40	0.36	0.50	0.50	-	0.47
+/- 1 Accuracy	0.83	0.83	0.70	0.64	0.83	0.84	-	0.85
F1 Score	-	0.47	0.37	0.20	0.48	0.49	0.31	0.26
AUC	-	0.76	0.69	0.65	0.75	0.77	0.73	0.77

Table 1. Baseline classification results compared to human-level performance and previous studies.

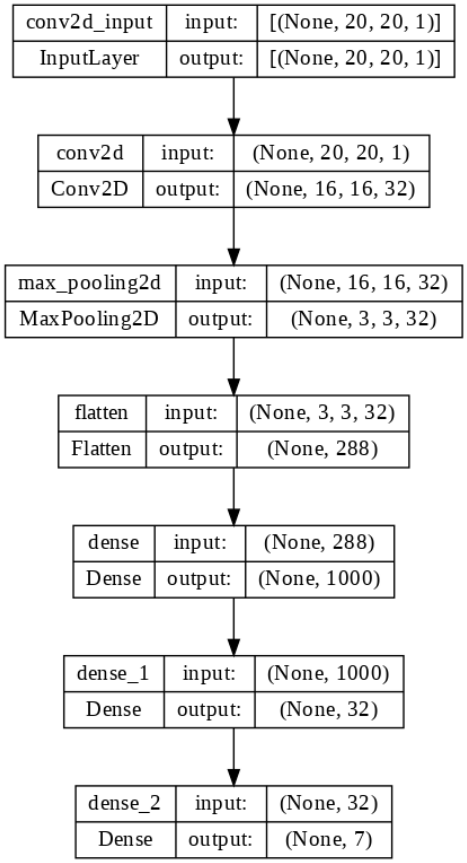


Fig 29. Simple CNN model architecture[1].

[1] The input to the CNN model is a (20, 20) matrix that is the original (18, 11) MoonBoard one-hot encoded matrix after padding for ease of implementation. The padding process does not impact the model's learning process as the padded areas are all 0's.

In all metrics, the top-performing methods are logistic regression, XGBoost, and simple CNN. Since we combined classes above V10, the models achieved higher accuracy, F1 score, and AUC even than GNN by Tai et al. [7] and RNN by Chang & Duh [8]. Not only did random forest and Naive Bayes perform poorly on accuracy, but their low F1 score and AUC also reflect that these two models suffer greatly from class imbalance and tend to overpredict V4/5 for harder routes. That is not to say we do not observe this behavior in the better-performing models. Fig 30 displays the confusion matrix for logistic regression. Overall, for V6+ routes and especially V9 and V10+ routes, the model exhibits a bias to underpredict the grade.

Since our exploration focuses on methods to handle class imbalance, not to optimize a complicated neural network that has previously been attempted with limited success, we will focus our efforts on logistic regression and XGBoost for simplicity.

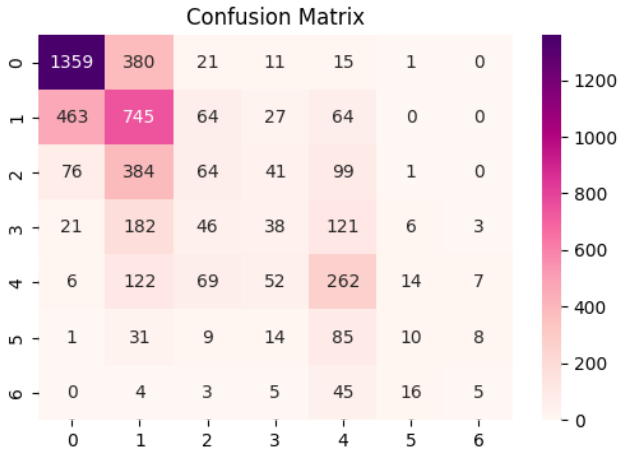


Fig 30. Logistic regression confusion matrix for original dataset.

6.3 Handling Class Imbalance

We study the effects of two methods to handle class imbalance: using class weights and oversampling the minority classes using routes generated from the VAE models. Our main metric of concern is balanced accuracy since it treats all classes equally.

6.3.1 Class Weights

Machine learning algorithms treat each data point equally, which causes problems when the data has biased classes. We can modify the algorithm by giving different weights to each class and thereby influencing the classification of the classes during the training phase. We do this by penalizing misclassifications of the minority class using a higher class weight while reducing the class weight for the majority class.

Applied to our dataset, we use "balanced" class weights, meaning they are inversely proportional to their respective frequencies. The weight for class *j* can be calculated using the formula below:

$$w_j = \frac{n_{\text{total samples}}}{n_{\text{classes}} \times n_{\text{class=j}}}$$

This significantly reduces the weights for frequent classes (V4 and V5) while increasing the weights for infrequent ones (V8, V9, V10+). If the model misclassifies a V10+ route, it will be penalized much more severely than misclassifying a V4 route. As a result, the model is expected to show better performance for classifying minority classes than not using class weights.

6.3.2 Oversampling with Generated Routes

A popular method to deal with imbalanced datasets is to oversample the minority class by repeating samples and undersampling the majority class by removing samples with the goal of exposing the model to each class in a more balanced manner.

With a working route generator, we propose oversampling using novel generated samples instead of repeating existing samples. A driving motivator is that our class imbalance is extremely severe: the V10+ class barely has 400 samples, while the V4 class has more than 9,000. To balance the dataset would require us to repeat that class many times, which can lead to overfitting. The model may become too specialized to the training data due to repeated exposure to the same samples and does not generalize well to new data. Instead, using generated samples for oversampling can help to mitigate this issue by introducing new and varied data to the model.

6.4 Approach

We use our trained VAE models to generate novel routes in three batches. Each batch includes 200 V8 routes, 400 V10 routes, and 600 V10+ routes. We continuously add batches to the existing, original training dataset and train logistic regression and XGBoost models respectively (see Table 2 for numbers of routes used for training), both with and without using weights.

	V4	V5	V6	V7	V8	V9	V10+
Original Dataset	7220	5644	2576	1791	1968	572	305
+ One Batch	7220	5644	2576	1791	2168	972	905
+ Two Batches	7220	5644	2576	1791	2368	1372	1505
+ Three Batches	7220	5644	2576	1791	2568	1772	2105

Table 2. Number of routes in training data per class.

6.5 Results

Combining the original training data with three batches of generated routes, we have four datasets for which we train with and without class weights, with results summarized in Table 3.

		Original	Original + W	+ 1 Batch	+ 1 Batch + W	+ 2 Batches	+ 2 Batches + W	+ 3 Batches	+ 3 Batches + W
Logistic Regression	Accuracy	0.50	0.45	0.50	0.46	0.50	0.46	0.50	0.46
	Balanced Accuracy	0.31	0.36	0.34	0.37	0.35	0.37	0.36	0.37
	Accuracy within 1 grade	0.83	0.80	0.83	0.80	0.83	0.81	0.82	0.81
	Accuracy within 2 grades	0.94	0.93	0.94	0.94	0.94	0.94	0.93	0.94
	AUC	0.76	0.79	0.76	0.78	0.76	0.78	0.77	0.78
	V4 recall	0.76	0.68	0.76	0.68	0.76	0.69	0.76	0.69
	V5 recall	0.55	0.40	0.55	0.41	0.55	0.40	0.47	0.40
	V6 recall	0.10	0.26	0.10	0.25	0.09	0.24	0.13	0.24
	V7 recall	0.07	0.23	0.05	0.23	0.06	0.25	0.09	0.25
	V8 recall	0.50	0.22	0.46	0.29	0.45	0.32	0.38	0.34
	V9 recall	0.08	0.30	0.13	0.34	0.20	0.32	0.22	0.30
	V10+ recall	0.07	0.45	0.29	0.40	0.37	0.40	0.30	0.35
XGBoost	Accuracy	0.50	0.48	0.50	0.48	0.49	0.47	0.50	0.48
	Balanced Accuracy	0.31	0.37	0.34	0.36	0.33	0.37	0.35	0.36
	Accuracy within 1 grade	0.84	0.82	0.83	0.82	0.83	0.82	0.82	0.83
	Accuracy within 2 grades	0.94	0.93	0.94	0.94	0.93	0.94	0.93	0.94
	AUC	0.75	0.77	0.76	0.77	0.76	0.78	0.76	0.77
	V4 recall	0.75	0.71	0.77	0.71	0.75	0.69	0.76	0.71
	V5 recall	0.56	0.46	0.55	0.45	0.54	0.45	0.53	0.45
	V6 recall	0.10	0.20	0.11	0.21	0.09	0.22	0.10	0.19
	V7 recall	0.10	0.23	0.09	0.23	0.07	0.25	0.08	0.24
	V8 recall	0.44	0.29	0.42	0.33	0.41	0.35	0.43	0.36
	V9 recall	0.09	0.30	0.16	0.25	0.16	0.24	0.22	0.24
	V10+ recall	0.15	0.39	0.27	0.36	0.32	0.41	0.33	0.36

Table 3. Results of classification experiments using logistic regression and XGBoost. “W” stands for class weights.

The table shows that although both methods lead to more balanced classification accuracy across classes, using class weights is the driving contributor behind most

improvements. Initially, both logistic regression and XGBoost achieved 31% balanced accuracy. Comparing the results for purely adding batches of generated routes, our hypothesis that increased representation of minority classes can improve balanced accuracy. The more batches, the stronger the improvement, with the two models reaching 36% and 35% balanced accuracy respectively, after adding three batches.

Adding class weights can quickly achieve the same results. Using the original dataset with balanced weights, logistic regression and XGBoost reach 36% and 37% balanced accuracy respectively, roughly the same overall effect as oversampling the minority classes. However, it appears that the severe underrepresentation of the minority classes may lead to too heavy of a weight when using balanced weights since they are inversely proportional to the classes' frequencies. The V9 and V10+ classes achieve very high accuracy (30% and 45% for logistic regression and 30% and 39% for XGBoost), while the other classes see less improvement. The results may be improved to achieve even more balanced accuracy across classes if we manipulated the class weights more instead of using balanced weights, which may be an interesting topic of exploration for future work.

In our results, the combination of both methods showed more balanced results as well. Using three batches and class weights, we are able to improve balanced classification accuracy from 31% to 37%. Although the use of oversampling and class weights achieved similar improvements in balanced accuracy, oversampling did so without losing accuracy, while class weights resulted in a decrease in accuracy from 50% to 46%. This observation is to be expected since the use of class weights penalizes mistakes made in those classes far less than in higher-grade classes. Since the majority classes account for more than half of test data, the loss

in accuracy in those classes exceeds the gain for minority classes, thereby sometimes reducing overall accuracy.

Each model has merits and weaknesses depending on the metric we use. For our purposes, we would like the classifier to have consistent performance across classes. A mistake of +1/-1 difference in grade prediction can be overlooked, but the classifier would entirely fail at its role if it were to predict V4/5 for very hard routes, as seen in the baseline results. Therefore, we choose to focus on "balanced accuracy" as our primary metric of concern and conclude that a classifier trained with class weights, added by 2 or 3 batches of generated routes, provides ideal results. If we were to focus on only overall accuracy, our conclusion would be drastically different, choosing instead to train the classifier using the original dataset without class weights.

7 Presentation

A collection of generated routes is hosted at https://junranshi.github.io/moonboard_website/ (Fig 31) for users to view. Each route is generated using a model trained on 200 samples of that grade for 20 epochs. Users can specify a desired grade or choose a random grade. The website also displays the predicted grade from the classifiers using logistic regression and XGBoost respectively. Of the generated routes, the classifiers' predictions are mostly accurate, typically within a 1-grade difference. The decision to host and load the websites on the website instead of generating them on the spot is for two main reasons:

1. The routes have been reviewed previously to ensure that users see relatively high-quality routes from the generator and prevent the occasional bad route;
2. The website serves only as a demo, not a fully-fledged route generator to be used. Therefore, only a handful of routes are sufficient to demonstrate the performance of the generator and classifier.

If we were to generate models on the spot, the ideal implementation would be to pre-train many models and sample a latent vector to generate a route every time the user demands. It prevents the website from taking too long to produce results due to training time.

This website is built upon the code from Andrew Houghton's Github project moon-board-climbing, with the author's permission.

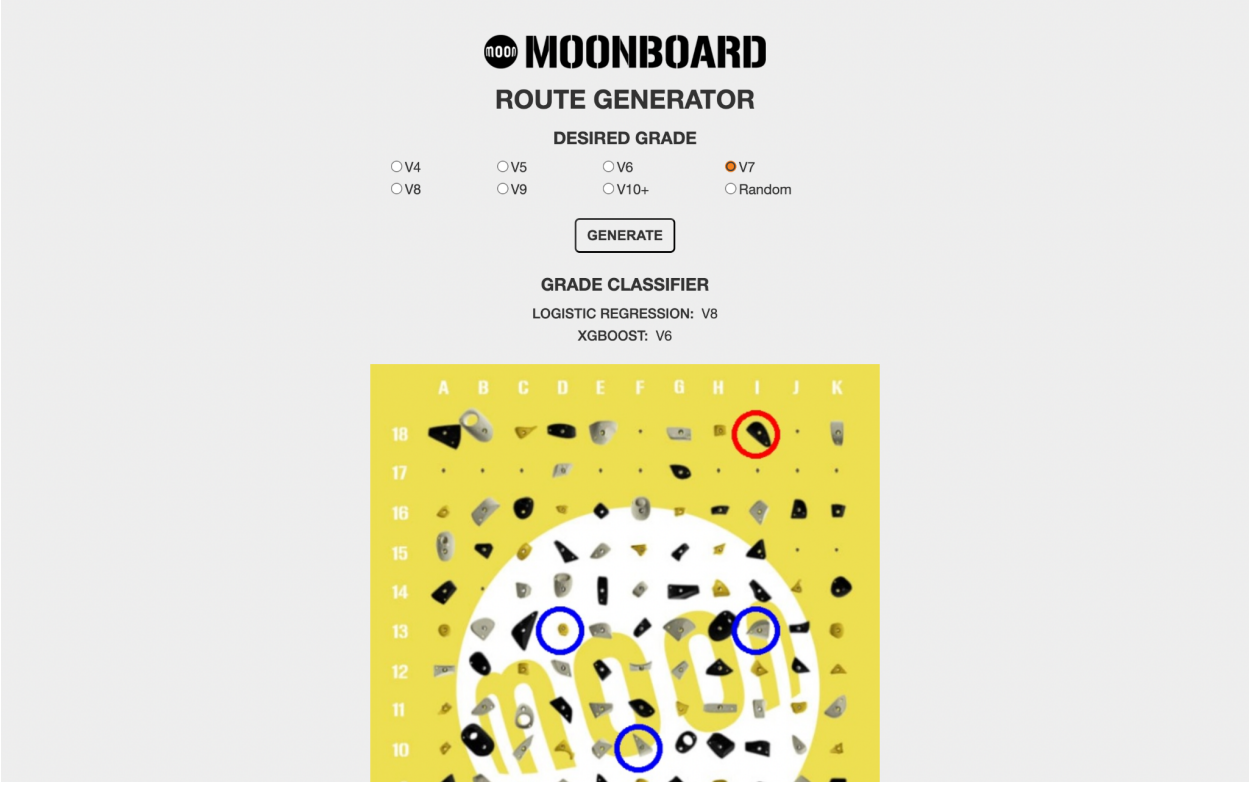


Fig 31. Preview of the demo website.

8 Route Validation

It is impossible to truly evaluate the quality of our generated routes with a mere visual inspection. While we can easily assess whether a route is legal, it requires physically trying routes to test if they are reflective of its grade, whether the holds can support required body positions, and whether the route is enjoyable to climb. Thus we invite experienced climbers to give subjective assessments of route quality in order to meaningfully understand the performance of our route generator.

8.1 Survey Design

Each participant is sent a Google Form that can be found here: https://forms.gle/cPazxetRv38SZQGW8 (Appendix A). The respondent chooses a grade range they typically climb on the MoonBoard and is presented with 8 routes of that range. Half of the routes are generated using our models and the other half are existing "benchmark" routes — ones that have been climbed by a large number of climbers, are "true to the grade," and agreed to be high-quality routes.

For each route, the climber answers a list of questions to describe their experience on the route:

- Were you able to finish this route after less than 5 quality tries? [Yes/No]
- On a scale of 1 to 5, did you enjoy this route? [1: Hated it!; 5: Loved it!]
- On a scale of 1 to 5, how would you rate the "flowiness" of this route? As in, did the moves feel intuitive and natural? Were there holds where you needed them? [1: It felt extremely unnatural; 5: It felt incredibly flowy]

- On a scale of 1 to 5, how would you rate the creativeness of this route? As in, did it feel very similar to many routes you've climbed, or did it take you by surprise with some element for example with a hold selection? [1: This route felt too boring; 5: This route had high creativity]
- Do you think this route was generated by a human or machine? [Human/Machine]
- On a scale of 1-5, how confident are you in your answer to the previous question? [1: Not confident at all, I can't tell!; 5: Extremely confident]

We took measures to ensure that participants were entirely blinded to the origins of each route to avoid potential biases. If the generated routes were publicly published, it would be easy for the climber to distinguish between generated routes (published by one particular user, no repeats) versus existing routes (published by different users, many repeats as a benchmark route). Publishing the benchmark routes using a new username is not an option as MoonBoard rejects users publishing existing routes. Van den Brand's [9] approach to mitigate this potential bias is to publish the existing routes with a subtle change in the finish hold. However, we reject this method as MoonBoard routes can change significantly in style and difficulty with even a small change in one hold. Instead of asking users to search for published routes, we present users with an image of the route and ask them to select the holds locally on their app and directly attempt the route on their MoonBoard without publishing it. This method fully blinds participants to any information regarding the route.

When a participant finishes evaluating routes of their comfortable grade range, they are presented with the option to evaluate routes of lower difficulties. Thus one relatively more advanced climber can contribute input on more than just their typical grade range.

8.2 Results

Route validation proved to be a challenge in previous works mainly due to the small sample size.

24 MoonBoard climbers of differing skill levels responded to our survey, providing input on 16 out of 20 generated routes (V10+ routes remain unattempted due to their difficult nature). Our sample size is a significant increase from van den Brand's analysis with 7 respondents but still insufficient for reaching statistically significant conclusions. We describe the survey responses below (Table 4) to reach some understanding of our models' performance.

		Responses	Completed?	Completed?	Enjoyment (1-5)	Flowiness (1-5)	Creativeness (1-5)	% Generated
Generated Routes	V4	16	43	67.19%	3.16	2.09	2.03	67.19%
	V4-6	11	21	47.73%	3.18	1.98	2.33	63.64%
	V6-8	7	12	42.86%	3.25	3.43	2.98	67.86%
	V8-10	4	6	37.50%	3.02	3.15	2.71	62.50%
	V10+	0	0					
	Total	**38**	**82**	**53.95%**	**3.17**	**2.42**	**2.36**	**65.79%**
Benchmark Routes	V4	16	41	64.06%	3.55	2.30	2.55	57.81%
	V4-6	11	19	43.18%	3.23	2.67	2.59	40.91%
	V6-8	7	11	39.29%	3.05	3.10	2.98	53.57%
	V8-10	4	4	25.00%	3.76	3.54	3.57	56.25%
	V10+	0			-			
	Total	**38**	**75**	**49.34%**	**3.39**	**2.69**	**2.75**	**51.97%**

Table 4. Route validation survey results.

The number of completions for generated routes is only slightly higher than benchmark routes, meaning the generated routes are roughly reflective of their supposed grade. In terms of enjoyment, flowiness, and creativeness, generated routes score lower than benchmark routes by 6.5%, 10%, and 14% respectively. Though the difference is not drastic, we can be fairly confident that generated routes lack behind slightly on these "humanistic" metrics that require creativity and a movement-oriented understanding of routes. We can observe that the discrepancy is relatively larger for easier routes (V4-6) than harder routes, meaning that our models perform

better at generating novel, high-quality difficult routes. Since the existing MoonBoard is heavily skewed towards easier routes, this finding supports our hypothesis that a route generator can provide value by adding more hard routes to the pool to support high-level climbers' and athletes' training. This observation may be attributable to higher-grade routes displaying less discernable patterns while lower-grade routes are constrained by the limited selection of holds they can use, which leads to the model learning a rigid set of patterns for generating V4/5 routes that appear less creative and flowy to experienced climbers.

With the knowledge that there are generated routes in the survey, respondents appear to be skeptical even of benchmark routes, with over half of them being guessed to be generated. The difference of 65.79% vs 51.97% isn't as big of a difference as we expected, meaning that participants in fact find it rather difficult to distinguish between generated routes and benchmark routes and the model was able to learn accurate patterns from the training data to generate similar routes. This is a promising outcome. To produce more statistically significant results, we would need to acquire more samples.

Overall, the survey responses show that our generated routes are of higher quality than previous attempts. We also validate the quality of our routes with greater credibility using a larger sample size and survey design that minimizes confirmation bias. Though still lacking in creativity measures, our routes are generally well-enjoyed by climbers. Additionally, we would like to add that benchmark routes are typically agreed to be of higher quality than average MoonBoard routes. Given more time, we may be able to compare our routes against average MoonBoard routes in addition to benchmark routes and acquire more samples for analysis.

9 Discussion

In this paper, we presented an approach using machine learning to generate climbing routes and classify their difficulty levels on the MoonBoard. We designed several experiments to explore the optimal setup for variational autoencoders and trained them using available MoonBoard data to generate legal and novel routes. We also trained classification models to predict the difficulty level of a given route. We found that combining our generative and classification models, as well as oversampling with generated routes and using class weights, improved the balanced accuracy of difficulty classification, realizing an improvement in results from previous efforts to handle the class imbalance in the MoonBoard dataset.

We also conducted a survey to validate the quality of our generated routes and found that they are generally well-enjoyed by climbers, significantly more so than the routes produced by previously published attempts. Our approach provides a promising solution for generating high-quality climbing routes and can assist in diversifying the MoonBoard's selection of routes, especially for higher grades. Our balanced classifier can also provide an additional grade when climbers approach a climb, either for an “objective” grade or to simply have another point of reference.

Through this process, we have been able to gain some initial insight into our motivating questions. We have found relevant heuristics that can assist us in assessing the difficulty of a route and converting a VAE output into a legal MoonBoard route. Our generative models can learn the patterns of existing routes to generate novel ones that appear and feel similar to human-generated routes. It seems that route-setting on the MoonBoard can be boiled down to rules, heuristics, and patterns to be learned by a machine-learning model. However, it is

questionable whether these insights can be transferred to the problem space of general indoor route-setting, as the MoonBoard is limited to a specific style of overhung climbing on small holds. Understanding general indoor route-setting requires the involvement of a far greater selection of holds, wall angles, possible movements, and route legality. Lastly, our exploration of grade classification sheds some light on the role of grades in climbing. When introduced to machine-predicted grades, climbers whom I surveyed expressed no fewer complaints of grade inaccuracy or "subjectivity." It seems that grades are almost always inaccurate for some and accurate for others since each route is so dependent on a climber's style, build, and strengths. Perhaps there is no "objective" grade to a route, but only a range of possible grades that can be used for reference. If this is the case, then our trained classification models can provide more grades for reference to climbers.

The problem space of this project shows potential for further findings. Future research can be done to improve the creativity of our generated routes, continue to validate the quality of our models, and add additional features to the generator such as adding/subtracting holds, generating routes including a particular hold, checking whether a route is climbable, etc.

References

[1] *Rock Climbing Statistics: Accidents, Injuries, Deaths & Demographics—99Boulders*. (n.d.). Retrieved December 9, 2022, from https://www.99boulders.com/the-growth-of-climbing

[2] Journal, C. B. (2022, September 23). New Survey Reports on Climbing's Consistent Growth. *Climbing Business Journal*. https://www.climbingbusinessjournal.com/new-survey-reports-on-climbings-consistent-growth/

[3] *MoonBoard—MoonBoard—What is the MoonBoard?* (n.d.). Retrieved December 9, 2022, from https://www.moonboard.com/what-is-the-moonboard

[4] Phillips, C., Becker, L., & Bradley, E. (2012). strange beta: An assistance system for indoor rock climbing route setting. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *22*(1), 013130. https://doi.org/10.1063/1.3693047

[5] Stapel, F. T. A. (2020, January 31). *A Heuristic Approach to Indoor Rock Climbing Route Generation* [Info:eu-repo/semantics/bachelorThesis]. University of Twente. http://essay.utwente.nl/80579/

[6] Dobles, A., Sarmiento, J. C., & Satterthwaite, P. (n.d.). *Machine Learning Methods for Climbing Route Classification*. 6.

[7] Tai, C.-H., Wu, A., & Hinojosa, R. (n.d.). *Graph Neural Networks in Classifying Rock Climbing Difficulties*. 6.

[8] Duh, Y.-S., & Chang, R. (2021). *Recurrent Neural Network for MoonBoard Climbing Route Classification and Generation* (arXiv:2102.01788). arXiv. http://arxiv.org/abs/2102.01788

[9] Brand, H. van den. (2019). *Climbing Creativity: Teaching a Neural Network to Create Routes for the Moonboard Training Board*. https://theses.ubn.ru.nl/handle/123456789/10867

[10] Lo, K. H. (2020). *Embedding and generation of indoor climbing routes with variational autoencoder* (arXiv:2009.13271). arXiv. https://doi.org/10.48550/arXiv.2009.13271

[11] Geary, C., & Valdez, J. (2017). *Climb a-GAN: Generation of Rock Climbing Problems*. 6.

[12] Chang, R. (2022). *MoonBoardRNN* [Jupyter Notebook]. https://github.com/jrchang612/MoonBoardRNN (Original work published 2020)

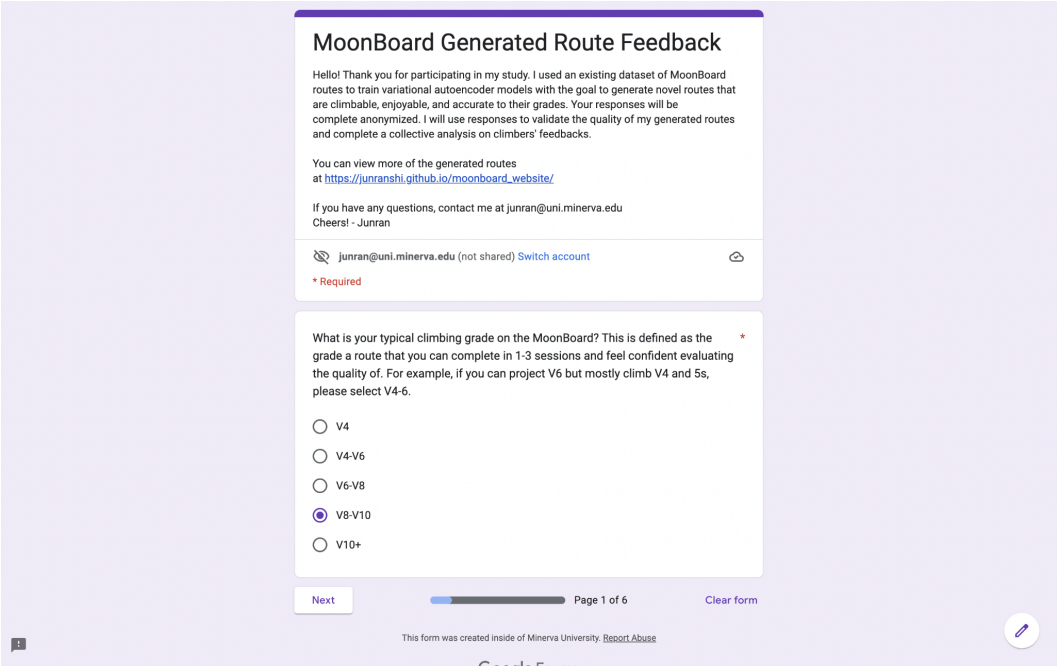
[13] Tai, H. (Cheng-H. (2022). *MoonGen* [Python]. https://github.com/gestalt-howard/moonGen (Original work published 2020)

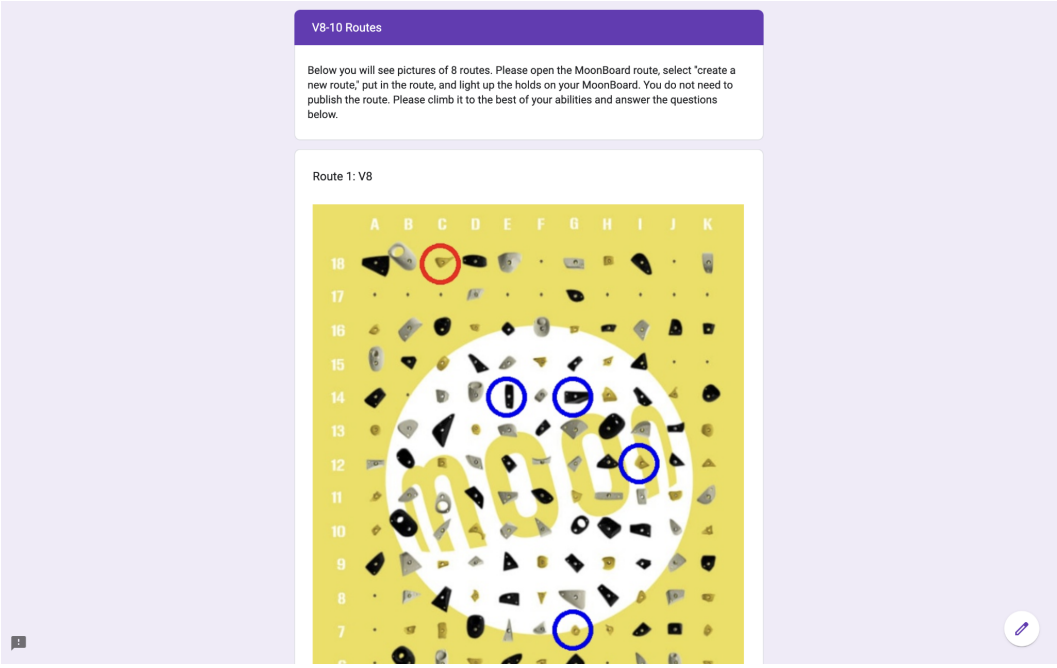
[14] Kingma, D. P., & Welling, M. (2014). *Auto-Encoding Variational Bayes* (arXiv:1312.6114). arXiv. https://doi.org/10.48550/arXiv.1312.6114

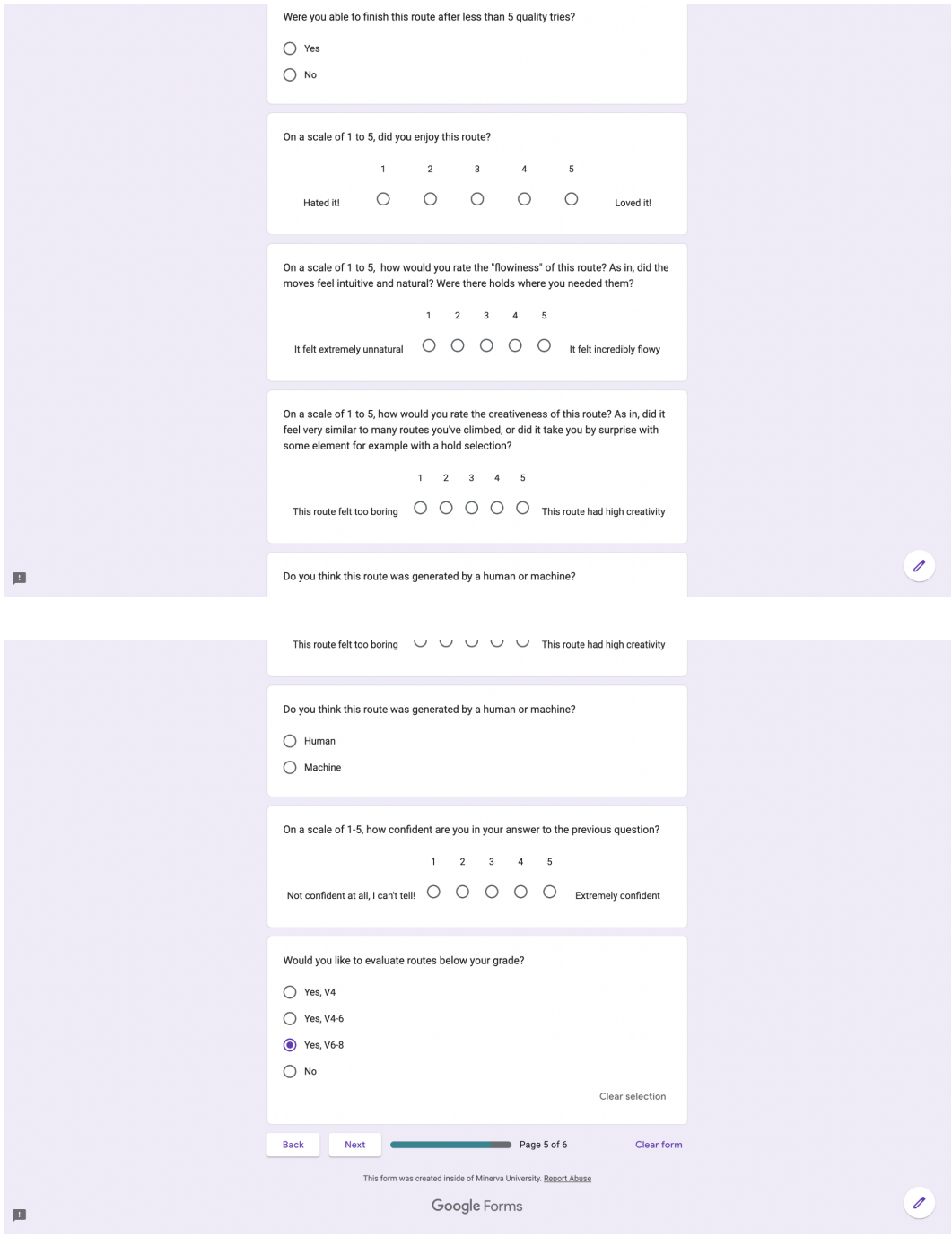
Appendices

A. Survey Form

Full form can be found here: https://forms.gle/cPazxetRv38SZQGW8






Were you able to finish this route after less than 5 quality tries?
Yes
No
On a scale of 1 to 5, did you enjoy this route?
1 2 3 4 5
Hated it!
Loved it!
On a scale of 1 to 5, how would you rate the "flowiness" of this route? As in, did the moves feel intuitive and natural? Were there holds where you needed them?
1 2 3 4 5
It felt extremely unnatural
It felt incredibly flowy
On a scale of 1 to 5, how would you rate the creativeness of this route? As in, did it feel very similar to many routes you've climbed, or did it take you by surprise with some element for example with a hold selection?
1 2 3 4 5
This route felt too boring
This route had high creativity
Do you think this route was generated by a human or machine?

This route felt too boring
This route had high creativity
Do you think this route was generated by a human or machine?
Human
Machine
On a scale of 1-5, how confident are you in your answer to the previous question?
1 2 3 4 5
Not confident at all, I can't tell!
Extremely confident
Would you like to evaluate routes below your grade?
Yes, V4
Yes, V4-6
Yes, V6-8
No
Clear selection
Back
Next
Page 5 of 6
Clear form
This form was created inside of Minerva University. Report Abuse
Google Forms

B. Capstone LOs

#qualitydeliverables	I submitted deliverables that show my work for these two semesters and demonstrate the appropriate scope and rigor for a Capstone project. I have achieved the goals that I set for myself at the end of the third year, including completing classification and generation that produced better results than previous attempts, making a website to show my results, and using surveys to validate my generated routes. All of my deliverables are exemplary of the quality and level of completion demanded by a Capstone-scale project.
#curation	My Capstone is presented in a well-structured manner that facilitates the reader's understanding. My deliverables include a paper that details my goal, approach, and findings; a user-facing website that showcases my route generation and grade classification results; and a code base that demonstrates the technicalities behind my project. They are all clearly labeled and organized so that readers can easily find the information that they need. For example, my paper is clearly organized using titles and includes a table of contents of all key content; my code base is introduced with an informative README that explains the structure and purposes of each folder. Content that is not directly relevant to my final presentation is included in appendices, such as my survey and iterative process to design the models.
#outcomeanalysis	My deliverable included a complete HC/LO list that satisfies the requirements for a full Capstone. The HCs and LOs that I have identified can adequately and appropriately serve as rubrics for measuring the quality of my work. For example, I have chosen CS HCs/LOs to describe my code-base and website, as well as Business LOs to analyze my design thinking approach and route validation process. My footnotes are concise but detailed, citing specific sections of my deliverable where skills and concepts are applied. I implemented the feedback received from the last submission to identify specific sections for applications in order for the grader to easily find them. Lastly, I also qualitatively assess the outcomes of my work, for example by using surveys to validate the quality of my generated routes and discussing the findings.
#navigation	Throughout the fourth year, I followed my intended timeline which was updated with each submission and hit the milestones that I had set for myself. I met every deadline and commitment with work products and presentations that exceed expectations. I broke down the project into several components to make them more approachable, including the following: completing a literature review to understand the problem space, replicating studies, implementing simple models, iteratively improving results, completing surveys for route validation, and creating a website to demonstrate my work. Based on the realities of my availabilities and priorities as well as my progress on the project, I adjusted my timeline and expectations with flexibility and communicated any changes and challenges to my advisor. Throughout the past two months, I worked consistently by devoting several hours per week to my Capstone. My route validation efforts required me to start the process early and reach out to contacts, therefore I prioritized my work to finalize the models first, send out

	surveys, then build the website. I made and followed realistic plans to complete my Capstone on time while remaining flexible and adapting to realities and challenges.

C. HCs

1 #professionalism	Throughout the time I worked on my Capstone, I demonstrated a high level of professionalism. I reached out proactively to my committee, never missed a meeting, and always came prepared with my progress and questions. My Capstone deliverables are well-organized and professionally formatted, following nuanced conventions for the audience, context, and discipline. For my paper, I made sure there were no grammatical mistakes, the citations were correctly formatted, the paper was properly structured, etc.
2 #responsibility	I demonstrated a high level of responsibility throughout these two semesters. I proactively scheduled meetings with my committee and always showed up on time and thoroughly prepared. During the Fall semester, I was able to shift my priorities and adjust my timeline quickly as I changed my plans for the Spring semester while communicating my plans to the relevant stakeholders of my Capstone. I took responsibility for my Capstone by following my timeline, meeting milestones, and reaching out for support from peers and professors when I faced difficulties.
3 #organization	My deliverables demonstrate clear organization that makes them easily understandable and digestible for readers. My paper is well-sectioned and follows a clear line of logic — reviewing existing literature to identify areas of improvement, gathering and analyzing data, performing route generation, using generated routes and class weights to perform classification, presenting my work using a website, and completing route validation using surveys. My code base is organized and well-commented, with a detailed README that explains how to navigate the repo.
4 #composition	Throughout my paper, I communicate with clarity and precision. For example, in my literature review (Section 3), I provide a summary of each study and identify their approach, strengths, and weaknesses; in my experiments section (5.4), I use subtitles to organize my communications and clear composition to describe my experiment, goals, and findings. Overall, my write-up is concise and uses professional language.
5 #audience	My deliverable includes a complete introduction (Section 1) in which I introduce the sport of climbing, grading in bouldering, the MoonBoard, etc. I understand that my audience may not be familiar with the subject and thus provided sufficient context with my explanations and visual aids (for example, using Fig 3 & 4 in section 1.2.1 to portray the usage of a MoonBoard). I also know that my audience should have a base level of understanding of machine learning. Therefore, I ensured that my explanation of VAE (Section 5.1) and model-building process (Section 5.2-5.4) was sufficiently informative without being too verbose. I used data visualizations to aid my explanations in all parts of my write-up to make the content more understandable for the audience (for example, displaying the routes to explain my experiments in section 5.4).
6 #critique	In Section 3, I completed a literature review of current approaches in which I noted which merits I could use for my models and what they could improve

	on (for example, for route validation in Section 3.3, I pointed out small samples, insufficient comparison, etc.). I highlighted the negative effects of their deficiencies on the credibility and validity of their conclusions. My critique is well-justified and evidence-based. Additionally, I adjusted my approach to address the weaknesses shown in other models — such as having a bigger sample size in route validation and mitigating biases in my survey design after critiquing van den Brand's design in Section 8.1.
7 #gapanalysis	My literature review in Section 3 serves as a gap analysis in the problem of MoonBoard route generation and grade classification. I evaluated the existing solutions (previously published attempts) and identified gaps to be filled (weaknesses in their approach). For example, I highlighted the difficulty in acquiring a large enough sample to validate the quality of generated routes and the insufficiently addressed problem of imbalanced classes in grade classification. My gap analysis allowed me to conclude that I need to implement novel solutions and produce improved results with much better validation.
8 #dataviz	I used data visualizations in several areas to aid my explanations and showcase my findings. In the exploratory data analysis in Section 4.2, I used histograms (figs 9, 10, & 11) to portray the dataset's class imbalance and heatmaps (fig 13) to describe the association between certain holds and difficulty grades. Since my project is very technical, I also used data visualizations to help explain my methods and findings, for example using a confusion matrix (Section 6.2, fig 30) to showcase that logistic regression shows a bias to underpredict the grade for V6+ classes despite having a high accuracy score.
9 #descriptivestats	In Section 4.2, I performed exploratory data analysis on the MoonBoard dataset and discussed my results using descriptive stats. For example, using the histograms in fig 11, I used compared the average number of holds between classes to illustrate that the number of holds can be used as a heuristic for predicting the grade of a route. I observed that the easier classes of V4/5 showed a much longer tail to the right, suggesting it's more likely for those routes to have more than 12 holds.
10 #heuristics	I used heuristics in two parts of my Capstone. The first is in Section 4.2.1, where I identified the number of holds and the types of holds as viable heuristics for evaluating the difficulty of a route. The second and more important application uses a heuristic in Section 5.3 to determine the number of holds to choose out of VAE output. My heuristic is to sample from the existing dataset of routes of the same grade and apply the same number of holds. Additionally, I slightly changed the heuristic to add more variability to the choices of start and finish holds. Not only did I justify why such a heuristic is needed, but I also evaluated its effectiveness in Section 5.5, noting its strengths and weaknesses, in particular, its disregard of the current route and potential to choose too few holds that the route is no longer climbable. Lastly, I suggested a method for future exploration to make up for this weakness.

11 #breakitdown	In Section 5.4, instead of approaching the configuration of the VAE generative model as a wholistic and daunting problem, I broke it down into tractable problems by running experiments to study the effects of each variable. I broke down the question of "how do we set up training configurations to produce high-quality routes accurate to its grade?" to the following questions: - what input format allows the model to learn MoonBoard rules? - should we train one model on all data or train different models for each grade? - would using smaller samples prevent the models from converging? - what latent space dimension allows the model to learn sufficient information from data? - how many epochs should we train for? My approach allowed me to study the effects of these factors independently to arrive at a full solution and determine the best configuration for training instead of trying different combinations to find a working setup by chance.
12 #constraints	I identified the constraints of the dataset's severe class imbalance in Section 4.2 and applied constraint satisfaction during the route generation and grade classification process. This is a constraint because we cannot simply acquire more samples for the minority classes, and the imbalance is due to the nature of the training board and climbers' skills. However, I employed strategies to work around this constraint. In route generation (Section 5.4), I proposed building independent models for each grade level to mitigate the previous model's tendency to generate only V4/5 routes due to overexposure in the training data. In grade classification (Section 6.3), I used two methods — class weights and oversampling with generated routes — to satisfy the constraints. I was able to significantly improve the classification accuracy for the minority classes despite their underrepresentation in the original training data, which is a successful application of constraint satisfaction.
13 #evidencebased	All of my conclusions are based on concrete evidence derived from my models and testing. My application of this HC is focused primarily in my evaluation of my generated routes. In Section 5.5, I first evaluate the quality of routes upon visual inspection, noting deficiencies such as holds being too far apart and using that as evidence to assess the effectiveness of my heuristic for choosing the number of holds. In Section 8, I use surveys to gather qualitative feedback about the routes from experienced climbers. Their feedback serves as evidence for my initial conclusions that machine-generated routes are of decent quality and can almost be mistaken for human-generated routes but lack behind in creativity. I also apply #evidencebased in my discussion of classification results in Section 6.5, in which I analyzed how balanced accuracy reacted to my implementation of the two methods to handle class imbalance and reached reasonable conclusions about their efficacy.
14 #biasmitigation	In Section 8.1, I identified the possible confirmation bias that climbers may have if they are not blinded to whether a route is generated by a human or an algorithm. This would be reflected in the route's creator username when they access it in the MoonBoard. Knowing a route is generated by a machine, the climber may give it lower scores because they are biased that machine-generated routes are inferior to human-generated ones. To mitigate this bias, instead of uploading routes to the MoonBoard database, I showed

	climbers screenshots of routes and asked them to “create” them on the MoonBoard app to try them without publishing them. This is an effective bias mitigation strategy because climbers are completely blinded to the route’s origins except if they have climbed the human-generated routes before (which is a rare case). I also critiqued van den Brand’s bias mitigation strategy by explaining that changing the finish hold of a route can lead to significant changes that the author did not account for.
15 #interviewsurvey	I designed a survey for climbers to validate the quality of my routes. The design and results are shown in Section 8, and the survey itself is linked in Appendix A. The survey asks questions that explore the study's goal and phrase them in a clear way, thus following best practices. For example, I clarified the definitions of 1 and 5 whenever asking climbers to use a scale of 1 to 5; I defined what a “comfortable grade range” means in context so that climbers are presented with climbs they can complete and evaluate. Moreover, my survey design minimizes potential biases by asking climbers to display routes locally on their MoonBoard instead of searching in a public database, as explained in Section 8.1 (more in #biasmitigation tag).
16 #algorithms	My project demonstrates a deep understanding and correct application of algorithms in many ways, including these two algorithms (found in the ‘Generation Finalized.ipynb’ file in the ‘models’ folder on GitHub): - the ‘train_generator’ function that takes in the desired grade and number of samples used to train a generative model and returns the trained decoder; - the ‘display_generated_route’ function the takes an input latent vector, trained decoder, and associated grade as input, uses the decoder to generate a route with the latent vector and returns the output in various formats including a list of the holds and an image of the route. I use these algorithms effectively to simplify my code and prevent repetitive work, as seen in the sections where I train generators and display routes. For all of the algorithms I designed, I also included comments to identify the required inputs, expected outputs, and the functions of the algorithms.
17 #designthinking	Throughout my time working on this project, I applied iterative design thinking to refine my solutions continuously. Before committing to this project, I conducted initial user research to understand the problem space and identify key features that are needed to create value. It allowed me to recognize stronger climbers’ needs for more diverse high-difficulty routes, which was not considered previously. As I designed and trained my models, I changed my parameters and setups iteratively (Section 5.4), gathering initial feedback from two relatively strong climber friends at each step to evaluate the results together and implement changes. This “rapid prototyping” process allowed me to efficiently arrive at the best design. Lastly, I implemented my solution by building a website (Section 7) and allowing users to climb the generated routes, and identifying areas of improvement that can be considered for future work.

D. LOs

#MLCode (CS156)	I produced working, readable, and performant Python implementations of various machine learning systems. I included comments that explain my code at a high level for the reader to understand the technical implementation and for ease of replication. I use functions whenever necessary to avoid unnecessary repetition in my code. Most of this application is in the generation notebook for VAE ('Generation_Experiments.ipynb' and 'Generation_Finalized.ipynb'), where I documented each of the experiments I did and the final models.
#MLExplanation (CS156)	I provided a clear explanation of variational autoencoders in Section 5.1, where I explained how VAEs work theoretically, explained the mathematics behind the loss function, and the architecture of my specific implementation. In Section 5.2, I also explained in detail the input and output structure of my models and justified my design decisions using findings from my experiments, for example why training for 20 epochs is sufficient and why training using smaller samples can result in more creative routes.
#MLFlexibility (CS156)	I apply machine learning skills and concepts in a way that addresses the goals of this project. For example, in Section 5, I leverage variational autoencoders in a substantial manner that shows a thorough understanding of the subject. My application shows a deep grasp of the concept as I justify the decisions I make for training, ie using experiments (Section 5.4) to determine the optimal input, number of epochs, latent space dimension, etc. I also demonstrate flexibility in the use of generated routes to overcome the class imbalance in grade classification (Section 6.3). The application of this strategy shows that I have a deep understanding of oversampling methods and can leverage the results from one half of my project to aid the other half to prevent overfitting when oversampling using original data.
#webstandards (CS162)	In Section 7, I created a demo website where climbers can generate new routes of a given grade and see the classified grades. I built systems that correctly use standard web technologies such as HTTP and HTML with the support of CSS and JavaScript. In my write-up, I justified my decision to display pre-generated routes to the user instead of generating new routes on the spot and described the optimal setup for higher performance if that were to be implemented. The code for the website can be found in the 'website' folder in my repo.
#whyExist (B110)	In Section 3, I leveraged my research into the space and analysis of existing literature to identify space for better work in route generation and grade classification, providing sufficient evidence for why my work is relevant and useful. I also identified how exactly my solution can add value for users in Section 2.1, such as adding to the lacking collection of harder routes, lowering the bar of entry for those interested in

	route-setting, etc.
#implementation (B144)	In Section 8, I conduct user testing with the use of generated routes and benchmark routes mixed together in a survey with the goal to determine the quality of machine-generated routes. I used a design thinking approach: using results from the user testing, I identified possible areas of improvement, such as checking a generated route for feasibility based on the types of holds and distance between holds and using a smaller sample size for training V4-5 models.