Original Research Article

Exploration of Generative Adversarial Networks for Smart Rock Climbing Training Systems

Matthew Acs[ab], Richard Acs[ab*]

[a] Department of Computer Science, Florida Atlantic University, Boca Raton, Florida
[b] Co-first authors
*Corresponding author

ABSTRACT

This study explores the application of deep convolutional generative adversarial networks (DCGANs) to generate synthetic rock climbing problems on the Kilter Board, a modern interactive climbing system. Departing from traditional NLP-based methods, we employed three DCGAN models tailored for different difficulty levels. Our exploratory data analysis (EDA) revealed distinct patterns in climbing grades, with easier climbs showing specific hold usage and harder climbs displaying greater variability. A classification model validated these characteristics with over 95% accuracy. While the GAN models successfully produced a significant portion of valid and interpretable climbs, they faced challenges such as mode collapse and repetitive sequences, mitigated by using an ensemble approach. Despite some generated climbs not matching the intended difficulty, this study demonstrates the feasibility of using DCGANs for synthetic climbing problem generation, providing valuable insights into AI integration in smart climbing systems and laying a foundation for future advancements to enhance training and performance for climbers.

KEYWORDS

Kilter Board, GAN, Smart Rock Climbing, Smart Systems, Generative Adversarial Networks, Artificial Intelligence, Neural Networks

INTRODUCTION

Rock climbing has evolved significantly over the years, incorporating advancements in technology to enhance safety, performance, and overall experience. One of the latest developments in this field is the integration of smart systems, which leverage sensors, data analytics, and internet connectivity to provide feedback and insights to climbers [1]. These systems are designed to monitor and allow control over various parameters such as movement patterns, environmental conditions, and grip types. This offers valuable information that can help climbers optimize their techniques, prevent injuries, and improve their climbing efficiency.

One of the most important smart system training tools in rock climbing is the standardized interactive climbing board. These systems work by placing a grid of standardized rock climbing holds on a fixed wall. Each hold has LEDs built into the base that allow them to light up to one of four different discrete

colors, indicating the hold type. Through an app, climbers can connect to the board and a database of climbs online to illuminate specific holds and emulate a certain route. Additionally, the app can control how overhung the wall is by using an actuator to adjust the angle of the board. Although several of these systems exist, the two most popular are the MoonBoard [2] and Kilter Board [3] climbing systems. Each system has slightly different grid layouts, sizes, and degrees of overhang available, but both provide an app that allows users to select holds and create routes using the app's setting mode. Due to the widespread popularity of these systems in climbing gyms, both the MoonBoard and Kilter Board systems have climbing databases with thousands of climbs at different difficulties. However, it is often difficult to create climbs that effectively train a desired movement pattern and difficulty of climbing. With the large existing database of climbs, the challenge of generating routes for standardized climbing training boards has been approached from a data synthesis perspective.

Recently, machine learning approaches for generating synthetic board climbs have been studied. Duh and Chang utilized an LSTM approach along with climbing sequence pre-processing to generate MoonBoard climbs [4]. Their research framed board climb synthesis as an NLP-like problem as opposed to a computer vision problem due to the sparse nature of the image representation as well as the logical flow that synthesized climbs must follow [4]. Stapel also utilized a similar NLP-like approach to synthesize MoonBoard climbs, but used additional feature engineering to allow the integration of different climbing techniques and skills in the generator [5]. In both of these studies, routes were generated by building sequences bottom up and taking move types, distances, and other heuristics into account to produce natural, human-like movement of a specific type. This NLP-like approach has even been utilized to create a user website that allows for automatic generation and grading of routes alongside the MoonBoard app [6]. Non neural-network based methods have also been studied, including evolutionary algorithms [7] and chaotic variations approaches [8]. Notably, all of the existing approaches utilize a ground up approach, generating move sequences into a route, grading the route, and then presenting the route to the user. Additionally, all existing literature utilizes the MoonBoard climbing system, which has lower hold density and a more regular structure than the Kilter Board system.

This research aims to utilize deep convolutional generative adversarial networks (DCGAN) to synthesize novel routes for the Kilter Board training system. Our work expands current literature by attempting to generate problems on the Kilter Board platform, which presents a larger challenge for route generation due to the higher density and non-gridlike structure of the holds. We present three separate generator models, one trained to synthesize easier climbs (VEasy), one trained to synthesize harder climbs (VDifficult), and one trained on the entire database of both hard and easy climbs used for this study (VAll). To better understand the characteristics of the generated climbs, we also present a convolutional neural network classifier that we used to distinguish between harder and easier synthetic problems. This is compared with a subjective analysis of the generated routes to better understand the quality of the generated problems on a broader scale. The purpose of this study is twofold; to prove the feasibility of generating training climbs from a DCGAN model perspective, and to provide an in-depth analysis of the data publicly available for the Kilter Board system. To our knowledge, our study is the first to report an image based generative adversarial network for the application of generating rock climbing routes on a Kilter Board. Although NLP approaches have shown promise and dominated existing literature [4-6], we attempt to show the feasibility of framing this problem from a computer vision perspective.

METHODS

We began our study by creating and preprocessing a novel Kilter Board dataset. Subsequently, we performed an exploratory data analysis, trained both classification and generative adversarial models, and evaluated the outcomes. The specific techniques employed are detailed in the subsections below, and the overall procedure is described in the final subsection.

1.1 Dataset

The dataset for this study was sourced from the Kilter Board application, available on the Apple App Store [9]. To compile the dataset, we systematically captured screenshots of 12’ x 12’ Kilter Board climbs. These screenshots included only graded boulder climbs (problems) ranging from V1 (easier) to V6 (harder), providing a diverse range of difficulty levels for analysis. Screenshots were taken in descending order of popularity, starting with the most frequently ascended climbs and moving towards those with fewer ascents. This approach ensured that the dataset comprised climbs that are well-tested and validated by the climbing community. Figure 1.a shows the number of screenshots for each grade.

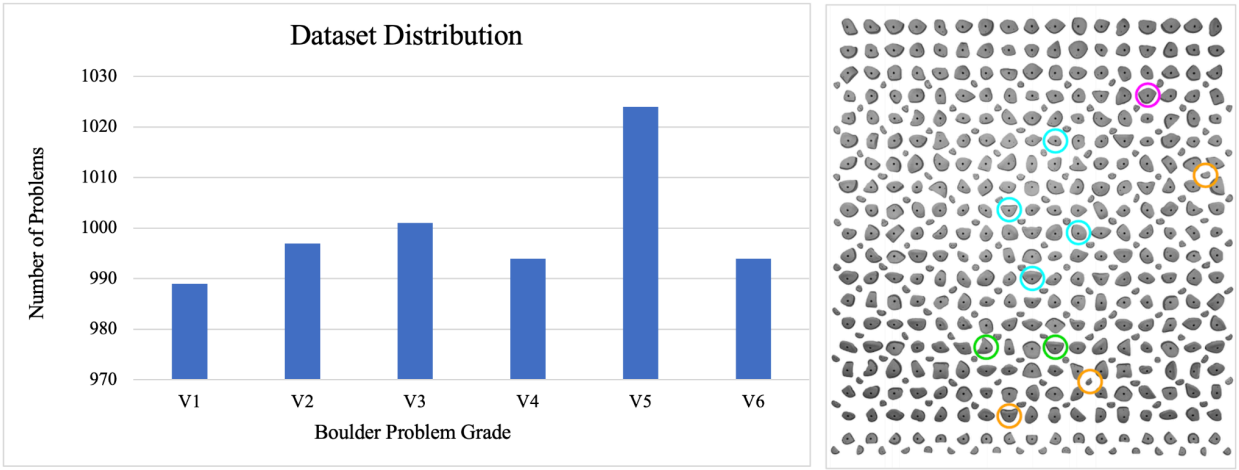


FIGURE 1. a) The number of climbs for each grade. b) An initial screenshot from the Kilter Board app.

Figure 1.b shows an initial screenshot from the Kilter Board app. As illustrated, each screenshot accurately captures the layout and position of holds on the Kilter Board. This step was crucial, as the subsequent analysis relied heavily on the precise location and color of each hold.

1.2 Data Preprocessing

The captured data underwent a preprocessing phase to prepare it for analysis, utilizing Python's PIL [10] and OpenCV [11] libraries. The steps were as follows:

1. **Color Detection:** Using OpenCV, we applied color thresholding to identify and isolate the colored holds in each screenshot.
2. **Marking Holds:** Holds were annotated with filled circles using OpenCV, ensuring clear identification.

3. **Background Removal:** Background elements were removed to isolate the holds and reduce noise, focusing on regions of interest.

Subsequently, all images were rescaled to a standardized resolution of 128 x 128 pixels. This rescaling ensured uniformity across the dataset and optimized computational efficiency for the machine learning models later on. Finally, all captured images were manually screened to remove duplicates. Images from V1-V3 were grouped together to form VEasy and images from V4-V6 were grouped together to form VDifficult.

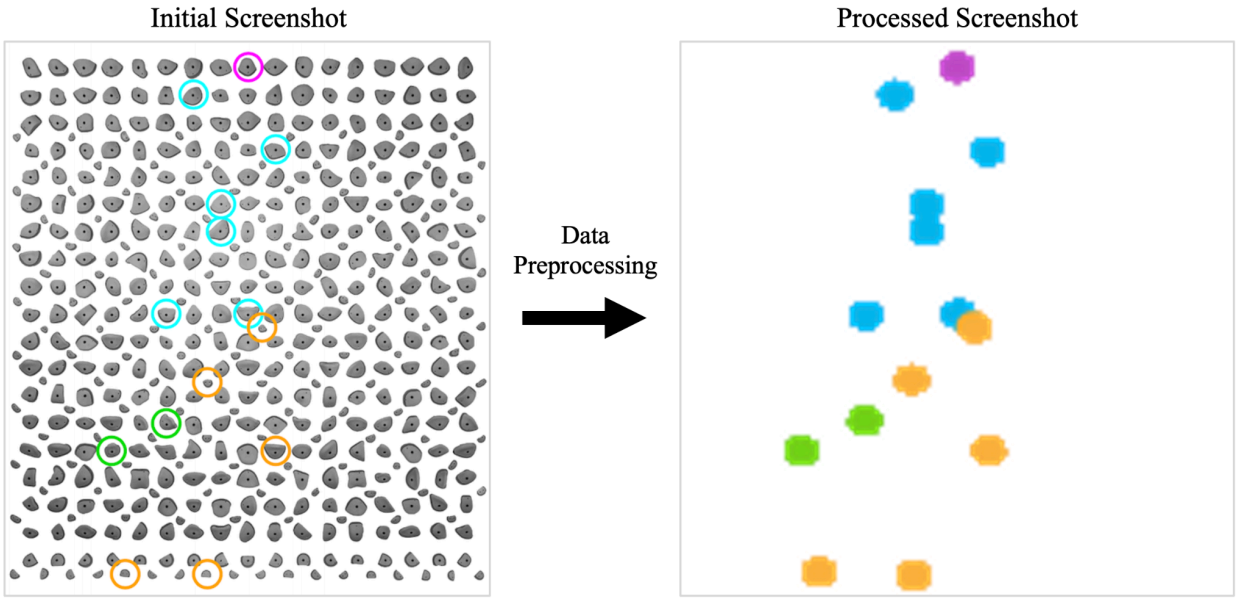


FIGURE 2. Data preprocessing transformations removed unnecessary background noise and highlighted important information.

Through these preprocessing steps, we transformed raw screenshots into a clean, annotated, and standardized dataset, ready for further analysis, as shown in figure 2. The preprocessing steps retained important information while reducing the image complexity and removing unnecessary background noise.

1.3 Exploratory Data Analysis

The exploratory data analysis (EDA) for the novel Kilter Board dataset involved several steps to better understand the data and gain insights into its features. Two key steps included creating average images and performing kernel principal component analysis (KPCA) for 3D visualization.

To understand the typical appearance of climbs at different difficulty levels, average images were generated. This process involved computing the mean of the pixel values for all images within each difficulty level category. The resulting average images provided a visual representation of the general characteristics and features associated with each difficulty level, highlighting common patterns or structures.

Kernel principal component analysis (KPCA) was performed to reduce the dimensionality of the data and visualize it in three dimensions. This step helped in understanding the underlying structure and distribution of the data points in a reduced space. Specifically, a polynomial kernel was used for the KPCA, which is effective in mapping the data into a lower-dimensional space where it becomes more separable. The SciKit Learn implementation of KPCA was utilized, as demonstrated in their documentation [12]. The resulting 3D scatter plot allowed for the visualization of clusters and patterns within the data, aiding in the identification of potential correlations between difficulty levels and the spatial distribution of the data points.

1.4 Classification Models

In order to provide a quantitative success metric for the generated images, we trained a convolutional neural network to classify Kilter Board climbs into either VEasy (V1-V3) or VDifficult (V4-V6). Initially, the Kilter Board dataset was normalized to pixel values in the range [0, 1]. The dataset was then split into training and validation sets using 5-fold cross validation to ensure unbiased evaluation of the models.

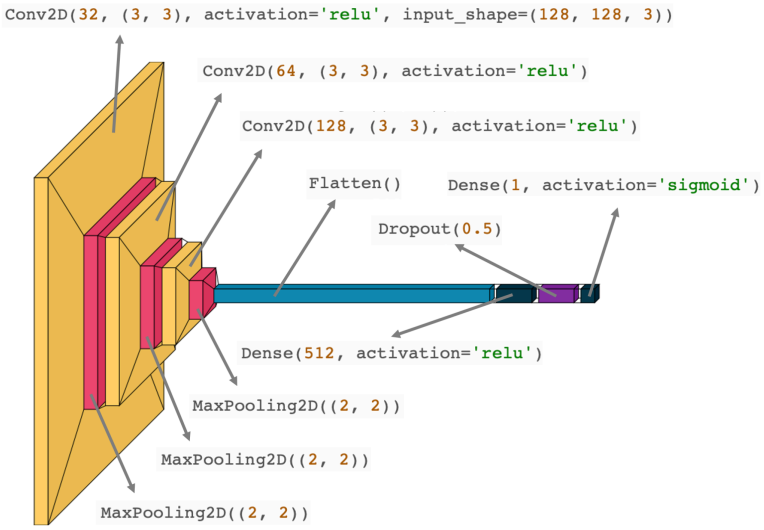


FIGURE 3. Convolutional neural network classifier architecture containing three convolutional layers.

For the classification model, a custom convolutional neural network (CNN) was used. The custom CNN architecture consisted of multiple convolutional layers with ReLU activation functions, max-pooling layers, and a dropout layer to prevent overfitting. Two dense layers were used with the final fully connected layer having a sigmoid activation function to output the class probabilities. The architecture of the CNN is shown in figure 3. Training of the models was conducted using the training dataset with a binary cross-entropy loss function and the Adam optimizer. The learning rate was set to 0.0001, the batch size was 16, and the number of epochs was 20.

The performance of the trained model was evaluated on the validation dataset using accuracy as a success metric. Confusion matrices were generated to visualize the classification performance across different difficulty levels and loss curves were generated to confirm training convergence. The CNN model was trained for both the original 128 x 128 images as well as rescaled 64 x 64 images. After training using 5

fold cross validation, the 64 x 64 model was trained using the entire dataset for 20 epochs and saved for later use. Tensorflow [13] and Keras [14] were used as the python libraries for implementing the CNNs.

1.5 Generative Adversarial Model

For our image generation approach, we utilized a deep convolutional generative adversarial network (DCGAN). The GAN architecture utilizes two separate networks, the discriminator and generator, that compete against each other in a process that generates increasingly convincing images relative to the original training dataset. The generator takes a latent space of noise, and outputs an image after passing through numerous convolutional layers. The discriminator takes an image as input and outputs a probability of the image being a real image or a fake image generated by the generator. Overall, as the generator and discriminator are trained, the generator begins to generate increasingly convincing images, while the discriminator learns to better differentiate between fake and real images. Our model architectures and parameters are based upon the original DCGAN paper [15]. The architecture of the generator and discriminator are shown in figures 4 and 5 respectively. Notably, both architectures utilize convolutional layers in place of fully connected layers to allow for greater depth, and pooling layers are replaced by strided convolutions. Additionally, batch normalization is utilized by both models, and LeakyReLU activation is used in all layers of the discriminator except for the final sigmoid to produce a probability of being real or fake. The generator utilizes ReLU for all layers except the final tanh to ensure that the pixel values remain in the normalized range of [-1, 1]. We utilized a latent space size of 100 based on the original DCGAN architecture, and our generator/discriminator generates and classifies 64 x 64 input image sizes for complexity considerations.

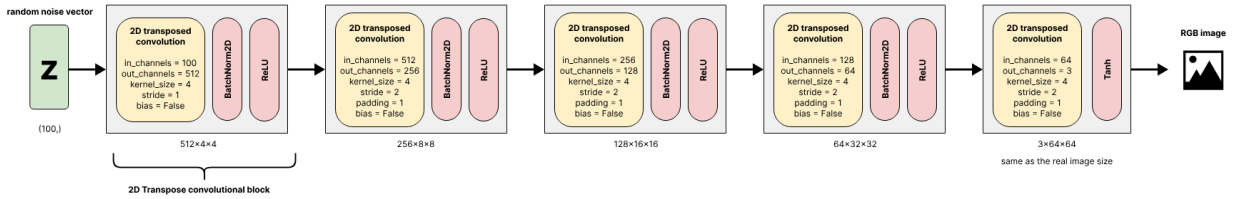


Figure 4. Generator architecture [16]

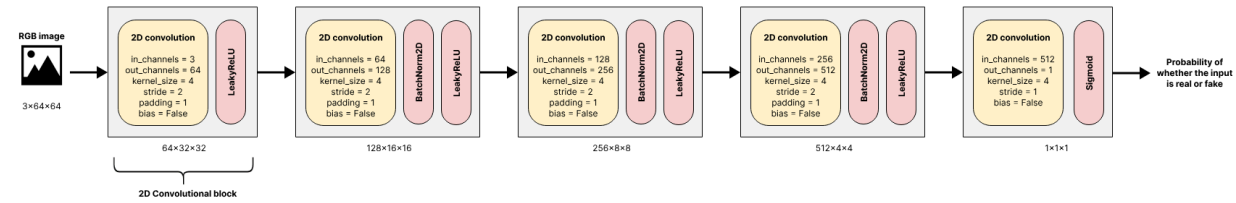


Figure 5. Discriminator architecture [16]

For our hyperparameters, we utilized a learning rate of 0.0002, Adam optimizer with beta1 parameter of 0.5 and beta2 of 0.999, and we conducted training for 500 epochs. We also initialized the weights of our generator's and discriminator's convolutional layers with a normal distribution, and batch normalization layers with a mean of 1.0 and bias to 0. Additionally, we utilized binary cross entropy (BCE) as the loss function to quantify the difference between the predicted probabilities and the actual binary labels. The equation for BCE is shown below (equation 1), with y being the target label of the sample, and $\hat{y}$ being the discriminator's probability output. The goal of the discriminator and generator is to minimize this loss,

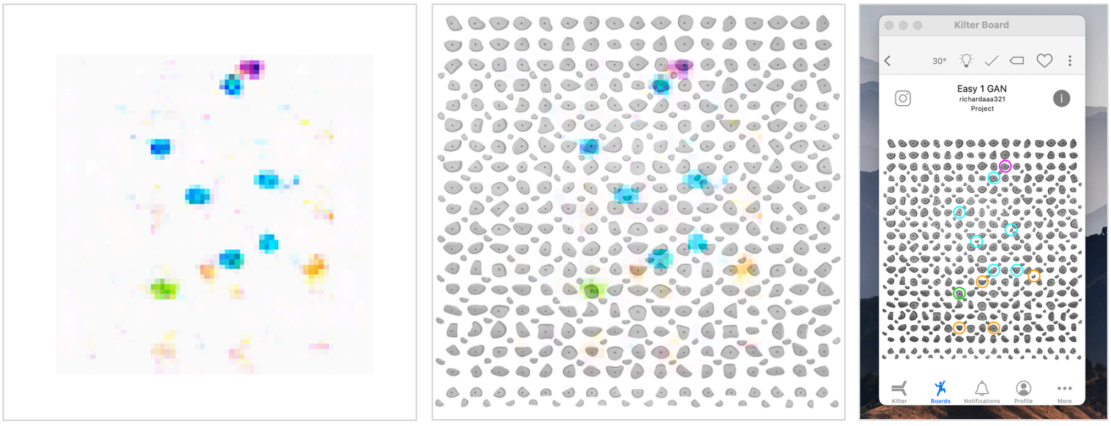
allowing the discriminator to learn to correctly classify real and generated samples, and the generator to learn to generate samples that can fool the discriminator into classifying them as real. In our study, the discriminator is first trained on an all real batch of samples, followed by an all generated batch of samples each with their respective loss calculations. To facilitate learning and prevent the discriminator from learning too quickly which would prevent the generator from learning, we randomly swap the labels of real images and fake images with a five percent probability. Additionally, based on recommendations by [17, 18], we followed the convention of labeling real images as 0 and fake images as 1, which helps improve the flow of gradients during learning.

$$(1)\ L(y, \hat{y}) = -(y * log(\hat{y}) + (1 - y) * log(1 - \hat{y}))$$

Using the architecture detailed above, we trained three separate ensemble models. The first model was trained on all climbs of difficulty V1 through V3, which we refer to as easy climbs. The second model was trained on all climbs of difficulty V4 through V3, known as harder climbs, and a third model was created with all data combined. Before training each model, since our dataset was in 128 x 128 resolution, we resized the data to 64 x 64 as expected by the model architecture and normalized each image pixel in the range [-1, 1]. To combat any potential mode collapse, we trained an ensemble of 10 models with different random weight initializations for each model category and used the ensemble when generating climbs for evaluation.

1.6 Processing of Generated Samples

100 samples were generated from each fully trained GAN for a total of 300 generated climbs. Each climb was manually evaluated for validity and classified as either invalid, valid, or interpretable. The remaining valid and interpretable climbs from each GAN were grouped into 5 classes each based on climb similarity. Finally, 3 samples were chosen at random from each of the 5 classes to be set onto a Kilter Board and evaluated. These selected climbs were overlaid onto a base Kilter Board, to show which holds were included in the climbs, then manually set using the Kilter app, as shown in figure 6.



(a) (b) (c)

FIGURE 6. a) A climb generated by the GAN. b) The generated climb overlaid onto a blank Kilter Board background resized to a square. c) The climb manually set on the Kilter Board app.

1.7 Evaluation Methods

The evaluation of the generated climbs involved classifying the valid and interpretable climbs. Each of the 45 selected climbs were rated in terms of difficulty via consensus of both authors after both having attempted to climb them. Additionally, the climbs were classified using the trained 64 x 64 CNN and the output was recorded.

1.8 Procedure

We initially followed the steps outlined in subsection 1.1 to obtain our raw dataset. This dataset was then preprocessed according to the steps outlined in subsection 1.2, resulting in a preprocessed dataset at a resolution of 128 x 128 pixels. The preprocessed dataset was saved, and an exploratory data analysis (EDA) was conducted as described in subsection 1.3. Insights from the EDA informed the methods in subsections 1.4 and 1.5. Subsequently, the classification model and generative adversarial models were developed concurrently, as detailed in subsections 1.4 and 1.5, respectively. We then followed the steps in subsection 1.6 to generate a set of climbs, which were evaluated using the procedure in subsection 1.7. The results of this procedure are reported in the results section below, in the order presented in this section.

RESULTS

The following subsections report the results of the experiments outlined in the methods section. The results show the outcomes of the EDA, classification models, three GAN models, and our evaluation and analysis of generated climbs.

2.1 Kernel Principal Component Analysis and Average Image

The 3D KPCA graph and average image from the exploratory data analysis are shown in figures 7 and 8 respectively. The KPCA graph shows a general clustering of climbs with easier grades being found higher along PC3 in the graph and harder grades being found lower along PC3. Additionally, the average image for each class shows that easier grades have holds that are more frequently found among climbs in that class, while harder grades have a more distributed hold frequency among all available holds.

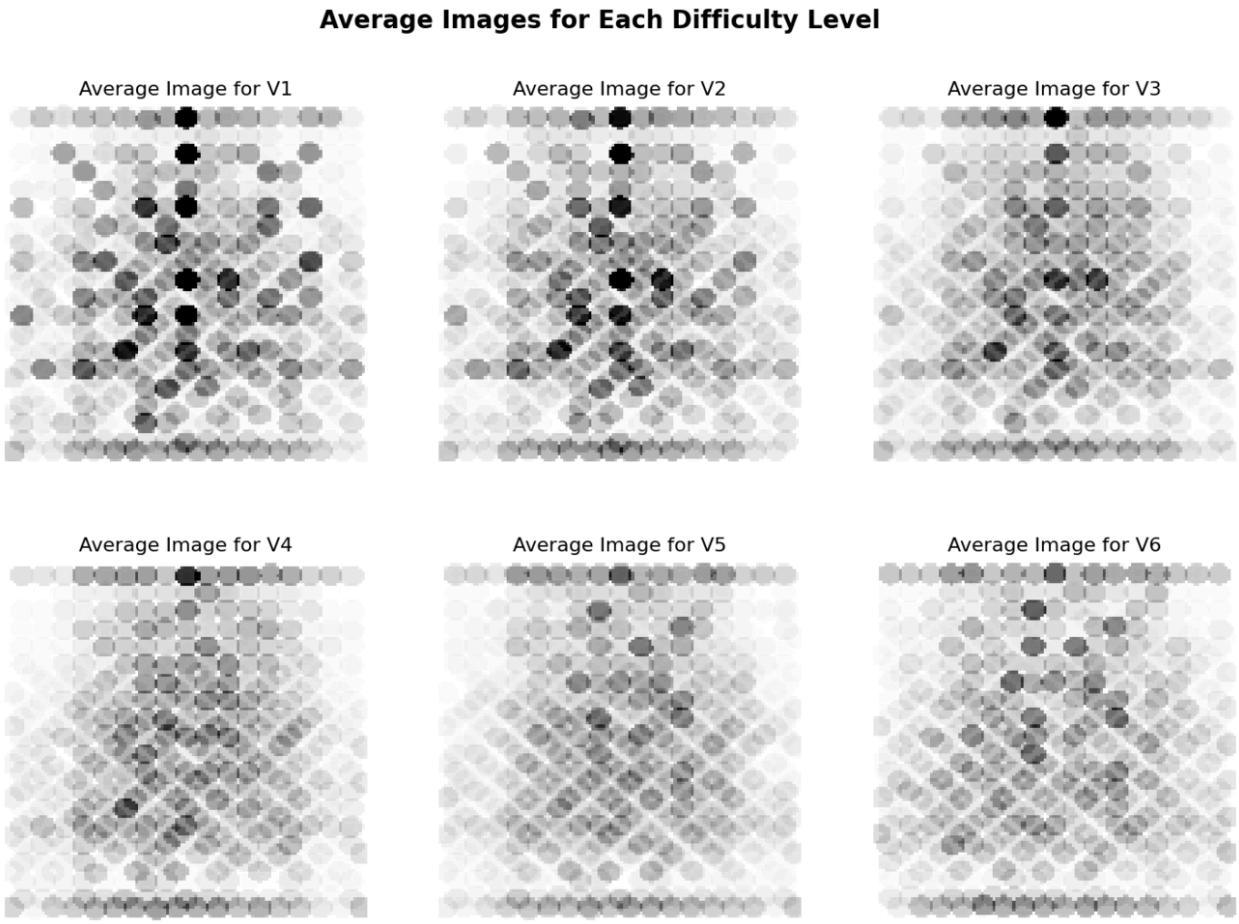


FIGURE 7. Average image across each class for the dataset.

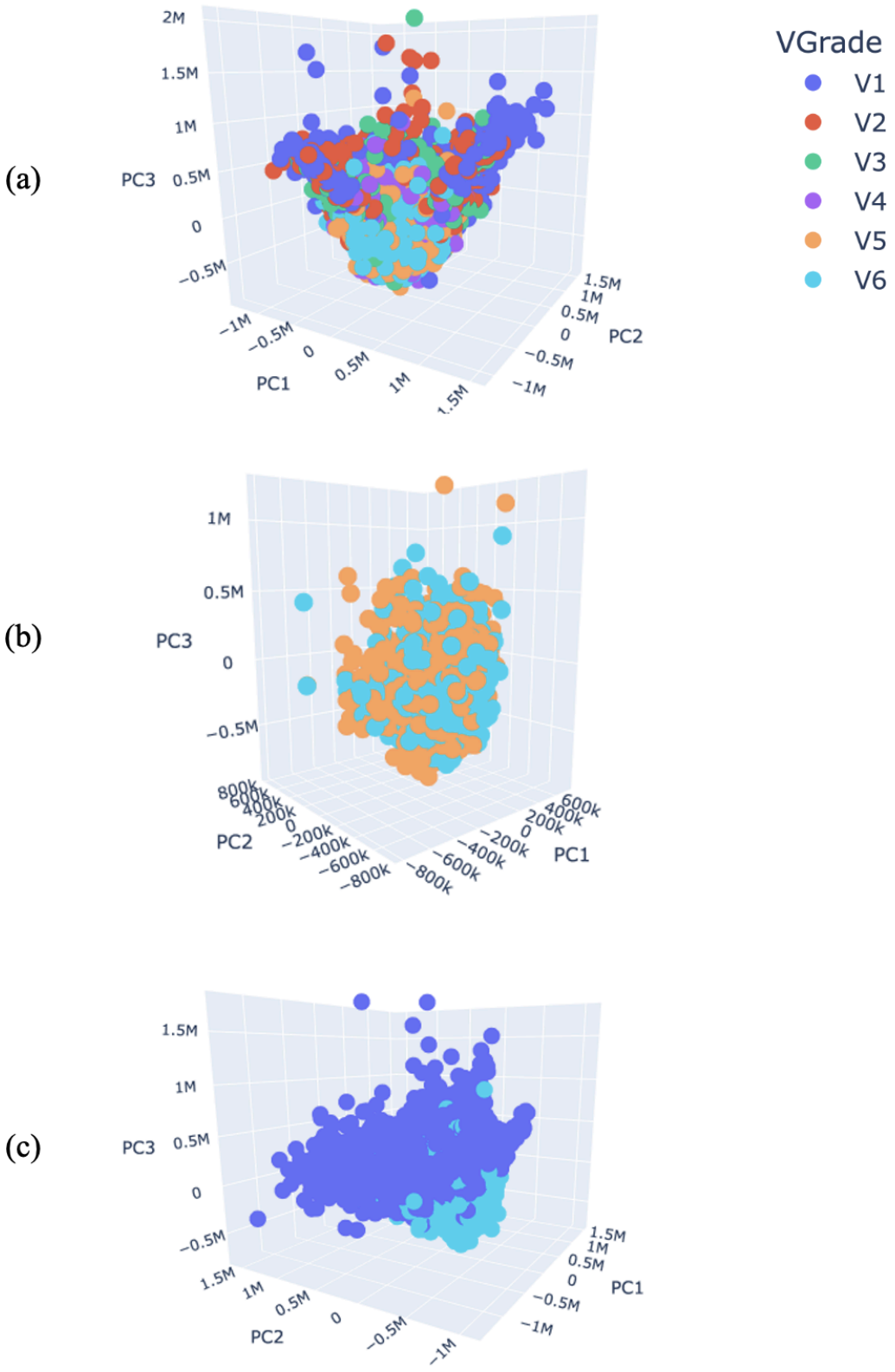


FIGURE 8. a) 3-dimensional KPCA of the full dataset. b) 3-dimensional KPCA of V5 and V6 datapoints. c) 3-dimensional KPCA of V1 and V6 datapoints.

2.2 Classification Model Success Metrics

The classification model training and success metrics are reported in Table 1. Overall, the 128 x 128 classifier achieved an average 5-fold cross validation accuracy of 95.70%, while the 64 x 64 classifier achieved an average 5-fold cross validation accuracy of 95.73%. The confusion matrices and training curves for the last fold of both models are shown in figure 9. The loss curve shows training convergence for both models. Finally, the 64 x 64 model trained on the entire dataset achieved a training loss of 0.0221 and training accuracy of 99.47%.

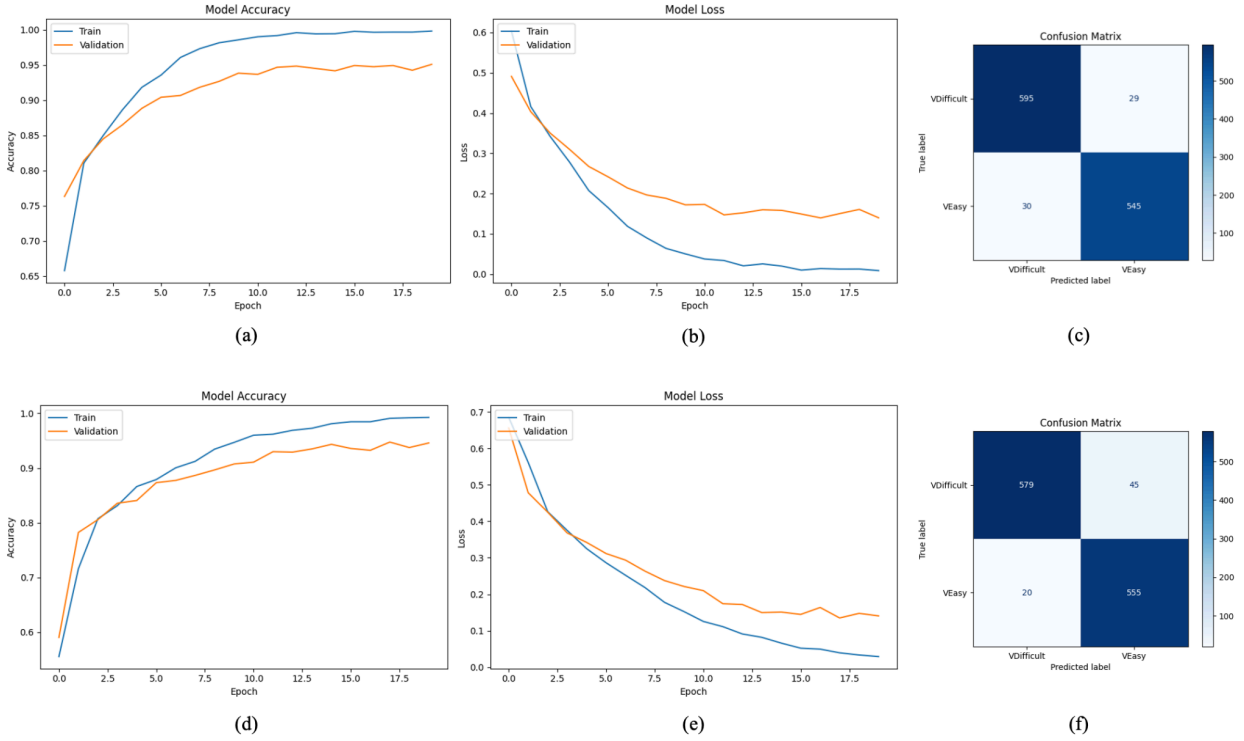


FIGURE 9. a) Accuracy curve of the 128 x 128 model training for the 5th fold. b) Loss curve of the 128 x 128 model training for the 5th fold. c) Confusion matrix of the 128 x 128 model using validation data for the 5th fold. d) Accuracy curve of the 64 x 64 model training for the 5th fold. e) Loss curve of the 64 x 64 model training for the 5th fold. f) Confusion matrix of the 64 x 64 model using validation data for the 5th fold.

TABLE 1. Classification model training and evaluation metrics.

Model	Average Validation Accuracy	Average Training Accuracy	Average Validation Loss	Average Training Loss
128 x 128	95.70%	99.81%	0.1210	0.0091
64 x 64	95.73%	99.56%	0.1109	0.0221

2.3 Generative Adversarial Model

The loss curve for the last model of the ensemble VDifficult GAN is shown in figure 10 below. Although only one of the loss curves are reported, each loss curve follows a near identical structure which would make reporting all 30 loss curves redundant. All loss curves exhibit the same trends of initial noise followed by balance and stabilization, with the generator loss being noisier and above that of the discriminator. Figure 11 shows random generations from the ensemble VAll and VEasy GANs. The raw generated data contained minor random noise that were not clear holds, some ambiguity in hold selection/shape, and contained very similar sequences to one another within a single model. These trends were once again the same for all 30 models trained in our experiments. Figure 12 shows generated climbs set on the Kilter app for evaluation.

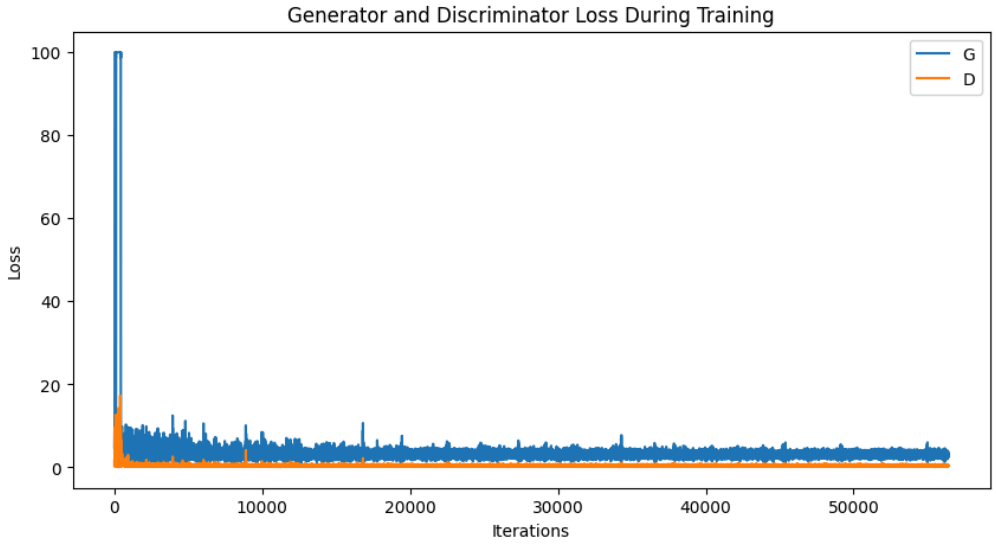
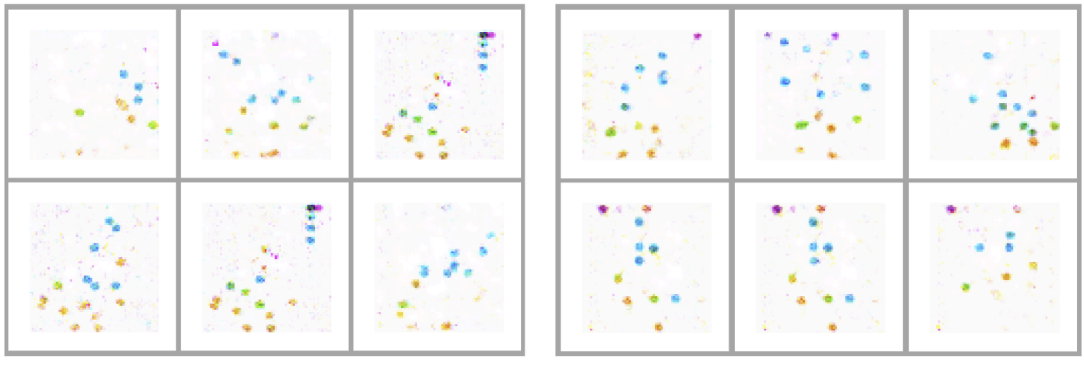


FIGURE 10. Loss curve for generator and discriminator during training of the last model of the ensemble VDifficult GAN.



(a) (b)

FIGURE 11. a) Raw generations from the fully trained VAll model. b) Raw generations from the fully trained VEasy model.

(a) (b) (c) (d)

FIGURE 12. a) A generated problem from the VAll GAN. b) Another generated problem from the VAll GAN. c) A generated problem from the VEasy GAN. d) A generated problem from the VDifficult GAN.

2.4 Generated Climb Evaluation Metrics

The evaluation metrics for the generated climbs are shown in tables 2 and 3. Table 2 shows the percentage of valid, interpretable, and invalid climbs produced by each generator out of the 100 climbs that each model created. Table 3 shows the percentage of the 15 chosen climbs that were graded to be the intended difficulty of that model.

TABLE 2. Generated climb validity.

GAN Model	Percentage Valid	Percentage Interpretable	Percentage Invalid
VEasy	49%	15%	36%
VDifficult	37%	45%	18%
VAll	47%	23%	30%

TABLE 3. Percentage of chosen generated climbs that are graded to be the intended difficulty.

GAN Model	Subjective Grading	Classifier
VEasy	20%	40%
VDifficult	60%	46.67%

DISCUSSIONS

This study focused on generating Kilter Board climbs through the computer vision perspective of a deep convolutional generative adversarial network. Additionally, we conducted an exploratory data analysis to understand underlying patterns among the data available through the Kilter Board database. This work was the first to utilize a computer vision based approach to generate interactive training board climbs at large, and the first to utilize the more challenging layout of the Kilter Board system over existing literature's use of the Moon Board [4-8]. Generative integration into existing interactive training systems will allow climbers to isolate specific movements, environmental conditions, and grip types. This makes characterizing existing board climb data and developing effective generative methods important to improving climbing training efficiency in such smart climbing systems.

Firstly, the results of our exploratory data analysis highlight the differences between climbing difficulties. The average images from easier grades had more specific holds being frequently used compared to more difficult climbs. Additionally, the average image from easier grades was more sparse. This reveals that easier climbs utilize a set of specific holds on the board, which implies that easier climbs are composed of easier holds. More difficult climbs have less of an apparent pattern, which implies that more difficult climbs have more variation in the holds used. The KPCA further showed that the different difficulty levels have specific qualities that make them separable. The KPCA plots show that the difficulty levels follow an imperfect gradient that creates regions where climbs of certain difficulties reside. When looking only at two climbs with similar difficulties, such as V5 and V6 shown in figure 8.b, the separability became less notable. However, when looking at two climbs with vastly different difficulty levels, such as V1 and V6 shown in figure 8.c, they occupy almost completely different spaces. This signifies that climbs of different difficulty levels are indeed separable, and that climbs of neighboring difficulties have overlap. This overlap is likely due to climbs being graded by humans on consensus, which is an imperfect system that will inevitably lead to certain climbs being rated as more difficult than they should be and vice versa.

We implemented a GAN model to generate climbs due to its simple architecture which captures sequential patterns well in lower resolution images [15], which is practical for our relatively small dataset of 6,000 climbs. Overall, the loss curve shown in figure 10 demonstrates that the discriminator and generator were able to converge to a stable state and successfully balance learning. Upon inspection of the generated problems in figure 11, the GAN models were able to learn the patterns present in Kilter Board problems and generate many viable looking climbs. However, a majority of the climbs from a single model have very similar if not identical sequences to a few overarching categories of climbs. This is known as mode collapse of the latent input space, and our data may have been especially prone due to the sparse nature of our preprocessed routes and similar images due to the inherent layout of start and finish holds for a climb. Utilizing an ensemble of 10 models resulted in significantly more variety of generated problems, but identical sequences and those with only slight modifications were still prevalent. Our architecture was relatively simple and followed the architecture presented by the original DCGAN paper operating on 64 by 64 images [15]. Creating a more complex and deep architecture, training for more epochs, and increasing the size of images accepted by the network may all help capture a larger complexity and bandwidth of patterns and help combat mode collapse.

Table 2 showed that the trained GANs created valid or interpretable images at a relatively high percentage, with 64%, 70%, and 82% of climbs being valid or interpretable for the VEasy, VAll, and VDifficult generators respectively. However, Table 3 showed that a lower percentage of the climbs generated actually represented the intended grade. This means that the generator was successfully able to produce climbs, however, they were not often at the intended grade. To grade the climbs, we used both subjective consensus and the 64 x 64 classifier, which achieved a 95.73% 5-fold cross validation accuracy. The high validation accuracy confirms the EDAs finding that different V-grades are indeed separable and represent distinct categories. The classifier graded 40% and 46.67% of generated climbs as being the appropriate grade for VEasy and VDifficult respectively, which highlights that the generator was able to successfully generate climbs at the intended grade. Subjectively, we graded 20% and 60% of generated climbs as being the appropriate grade for VEasy and VDifficult respectively. These numbers closely follow the results of the classifier. Finally, upon climbing the generated routes, we found a lack of flow that clearly felt awkward and unnatural.

Compared to existing ground up LSTM approaches, our results show that a GAN approach may generate less natural problems of a more repetitive nature. This may be due to the sparse and sequentially important nature of Kilter Board climb data. Utilizing image data lacks the inherent sequential information that can be encoded in sequential pattern representations, which allow for ground up creation of routes with more logical sequences. In our case, due to a lack of this information, the results showed generated climbs of strange sequences, such as putting the start and end holds next to each other (useless problem), or putting the end hold far below the start hold (unnatural and potentially injury-inducing problem). However, the focus of this work was to demonstrate the viability of framing this problem from a computer vision perspective, which did successfully result in climbable problems.

Although our study demonstrated the potential of utilizing generative adversarial networks to generate novel, human-like climbs for the Kilter Board, we used a relatively small subset of the Kilter Board database. The Kilter Board database has over 100,000 reported problems [19], and utilizing a larger subset could help combat mode collapse and create fewer strange/unnatural sequences. Additionally, our study omitted climbs from angles other than the 30 degree configuration of the Kilter Board, potentially losing sequence variability. As shown in figure 11, the low resolution of the generated climbs did introduce some ambiguity due to the non-circular nature of the holds generated at 64 x 64. Future efforts should focus on utilizing larger input/generated image sizes to decrease the level of hold ambiguity.

CONCLUSIONS

In this study, we explored the use of deep convolutional generative adversarial networks (DCGANs) to generate synthetic rock climbing problems on the Kilter Board, marking a novel application of computer vision in this domain. Our research employed three separate DCGAN models to create climbs of varying difficulty, moving away from the traditional NLP-like methods. The exploratory data analysis revealed distinct patterns across climbing grades, with easier climbs showing specific hold usage and harder climbs displaying greater variability. Our classification model, with over 95% accuracy, validated these distinct characteristics. While the GAN models faced challenges like mode collapse and repetitive sequences, the ensemble approach improved the variety and quality of generated climbs.

Despite the successes, our study identified areas for improvement. The generated climbs often did not align with the intended difficulty, highlighting the need for more sophisticated models or larger datasets to capture the nuances of climbing sequences better. The low-resolution images introduced ambiguity, suggesting future research should focus on higher resolution inputs. Nevertheless, this study demonstrates the feasibility of using DCGANs for generating synthetic climbing problems, contributing valuable insights into AI integration in smart rock climbing systems and laying a foundation for future advancements to enhance climbers' training and performance.

DISCLOSURE

The authors report no proprietary or commercial interest in any product mentioned or concept discussed in this article.

FUNDING

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

SOURCE CODE

The source code, datasets, and all results are publicly available at the following GitHub repository: https://github.com/matthewaaa123/KilterGAN

AUTHOR CONTRIBUTIONS

M.A. and R.A. contributed the study design and implementation, study selection, manuscript drafting, manuscript revisions, codebase, and a literature review of generative methods. M.A and R.A provided equal contributions. R.A contributed to the initial data analysis, and M.A contributed to the analysis of results.

ACKNOWLEDGMENTS

We would like to thank Kilter for providing the climbing route data and interface that we used for this study. We would also like to thank Google Colab for providing cloud hosted Jupyter Notebooks for the codebase. Finally, we appreciate the help of any climbers that we consulted for grading and assessment of generated climbs.

REFERENCES

1. M. S. W. Blunt, "A Survey of Low Cost Vision-Based Localization Techniques for Autonomous Mobile Robots," *Sensors*, vol. 23, no. 11, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/11/5080
2. "MoonBoard," Moon Climbing. [Online]. Available: https://moonclimbing.com/moonboard
3. "The Kilter Board," Setter Closet. [Online]. Available: https://settercloset.com/pages/the-kilter-board
4. H. O. L. Rosenberg, "Exploring Neural Network Training Strategies with TensorFlow," Stanford University, 2020. [Online]. Available: https://cs230.stanford.edu/projects_spring_2020/reports/38850664.pdf
5. R. W. Stapel, "Machine Learning Techniques for Predictive Maintenance," University of Twente, 2023. [Online]. Available: https://essay.utwente.nl/94487/1/Stapel_MA_EEMCS.pdf
6. A. Houghton, "Moon," A. Houghton, 2023. [Online]. Available: https://ahoughton.com/moon
7. D. Tyebkhan, "Deep Learning Techniques for Image Classification," Union College, 2023. [Online]. Available: https://arches.union.edu/do/53357/iiif/2efc327b-9f81-413e-ad18-97cb358e2b47/full/full/0/2023_TyebkhanD.pdf
8. I. S. Dhillon and S. Sra, "Generalized Nonnegative Matrix Approximations with Bregman Divergences," *arXiv preprint*, arXiv:1110.0532, 2011. [Online]. Available: https://arxiv.org/pdf/1110.0532
9. "Kilter Board," Apple Inc. [Online]. Available: https://apps.apple.com/us/app/kilter-board/id1215919336
10. "Pillow Documentation," Pillow. [Online]. Available: https://pillow.readthedocs.io/en/stable/
11. "opencv-python," PyPI. [Online]. Available: https://pypi.org/project/opencv-python/
12. "Kernel Principal Component Analysis," scikit-learn. [Online]. Available: https://scikit-learn.org/stable/auto_examples/decomposition/plot_kernel_pca.html
13. "TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/
14. "Keras API Documentation," Keras. [Online]. Available: https://keras.io/api/
15. A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv preprint*, arXiv:1511.06434, 2015. [Online]. Available: https://arxiv.org/pdf/1511.06434
16. "Understanding Image Generation: A Beginner's Guide to Generative Adversarial Networks (GAN)," OVHcloud, 2023. [Online]. Available: https://blog.ovhcloud.com/understanding-image-generation-beginner-guide-generative-adversarial-networks-gan/
17. I. Goodfellow et al., "Generative Adversarial Nets," *arXiv preprint*, arXiv:1406.2661, 2014. [Online]. Available: https://arxiv.org/pdf/1406.2661
18. C. Ledig et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," *arXiv preprint*, arXiv:1606.03498, 2016. [Online]. Available: https://arxiv.org/pdf/1606.03498
19. "Kilter Board Rules and Tips," Setter Closet. [Online]. Available: https://settercloset.com/pages/kilter-board-rules-tips