Climb a-GAN: Generation of Rock Climbing Problems

Cove Geary*
cove.geary@yale.edu
Yale University
New Haven, Connecticut

Joseph Valdez
joseph.valdez@yale.edu
Yale University
New Haven, Connecticut

Figure 1: Example of a climber using a Moonboard

ABSTRACT

Many rock climbing gyms have adopted a new type of rock climbing wall: the MoonBoard. The MoonBoard is a standardized interactive training wall of 11 x 18 holds that are identical in each gym. The MoonBoard website provides a space for users to create new custom paths, upload them, and even rate other paths. Using an ACGAN, it is possible for a user to be able to input a difficulty for a desired rock climbing path and have it generate a new suggested path according to that difficulty that is compatible with the MoonBoard. Current GitHub projects for MoonBoards generate a random path and then classify the difficulty, so this would be approaching it in reverse. By generating a path based on a given difficulty, it will be easier for rock climbing users to find a path they would be able to attempt (and enjoy). The classifiers in the ACGAN are the different levels of difficulties of the rock climbing routes. The ACGAN will output an image of a MoonBoard with the starting holds circled in green, the intermediate holds circled in blue, and the ending holds circled in red.

KEYWORDS

datasets, CGAN, conditional GAN, path generation

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/0000000.0000000

ACM Reference Format:
Cove Geary and Joseph Valdez. 2019. Climb a-GAN: Generation of Rock Climbing Problems. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/0000000.0000000

1 INTRODUCTION

Our project is motivated by two challenges in indoor rock climbing: generating new climbing paths (route- or problem-setting); and understanding what makes a particular problem challenging and fun.

Although the full space of possible mediums (wall size, arrangement, etc.) for a climbing problem is huge, perhaps the best candidate medium for starting to approach this algorithmically would be the MoonBoard. The MoonBoard is a standardized bouldering wall located in gyms throughout the world, such that every MoonBoard has the same features for hands and feet (holds) in the same place, arranged in a 11x18 grid. Climbing paths can be customized and created by users on an online community. The MoonBoard website hosts an online community where users can create problems, assign them a difficulty, and share them publicly for others to view. A problem on the MoonBoard is defined as a sequence of holds along the grid. Each problem is given a grade (difficulty level) and a rating (0-3 stars). There is even an app where users can upload paths so there exists a healthy amount (many thousands) of accessible data!

However, since users currently have to manually design paths and rate their difficulty, we were hoping this process could instead be streamlined with the aid of technology. It would be incredibly helpful if instead of creating your own path or even searching through thousands of paths to find one that matches your proficiency with bouldering, a user could simply input their desired difficulty and have a path automatically generated for them.

This might be able to be achieved through a GAN, or Generative Adversarial Network. GANs are given training data, then using two networks working against each other, eventually learns to generate new data similar to that of the training set. The two neural networks are referred to as the generator and the discriminator. The job of the generator is to learn to generate new data capable of fooling the discriminator into thinking it was from the initial training data. The job of the discriminator is to be able to determine where the generated data is "real" or "fake", or whether the data is from the initial training data or fabricated by the generator. After many iterations of this training, the generated data should so similar to the training data that it fools the discriminator. The model architecture is shown in the figure below.

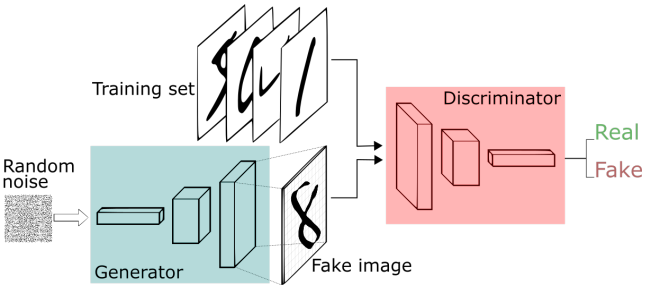

Figure 2: Model of a GAN from https://pathmind.com/wiki/generative-adversarial-network-gan

Using this method, we could train a GAN using the thousands of user-created MoonBoard problems with the intention of being able to generate new problems similar (and equally traversable) to the problems in the online community.

1.1 Related Work: Moonboards

As far as we can tell, very little work has been done thus far in the space of climbing problem/route generation. Further, anecdotally speaking, the task of route-setting tends to be seen by climbers as one that requires more than just practical skill—it is one that draws on creativity. For this reason, MoonBoard problem generation poses an exciting challenge for generative models.

As mentioned, the space of climbing problem generation, and even classification, seems to be relatively untouched. Some smaller GitHub-hosted projects seem to apply neural networks to classify the difficulty of existing MoonBoard problems (e.g. with an MLP, with a CNN, with an LSTM). We found one one GitHub-hosted project that seems to have experimented with generating novel MoonBoard problems. However, each of these repositories appears minimally-documented, and the results of each project is unclear. For the one generative project, problem generation and difficulty classification are done in two separate steps, so the starting "hold" is chosen at random to begin problem generation, and problem difficulty is assessed post-generation.

In one paper from an undergraduate machine learning final project, the authors present a more methodological approach to classifying MoonBoard problems [1]. Comparing Naive Bayes, Softmax Regression, and CNN (with ordinal output), they found the CNN to be the best-performing.

1.2 Related Work: ACGANs

Taking this further, it would be helpful for users to be able to generate a path according to their desired difficulty, making it easier to find a path they will enjoy. This could be possible by using an ACGAN, shown in the following figure.

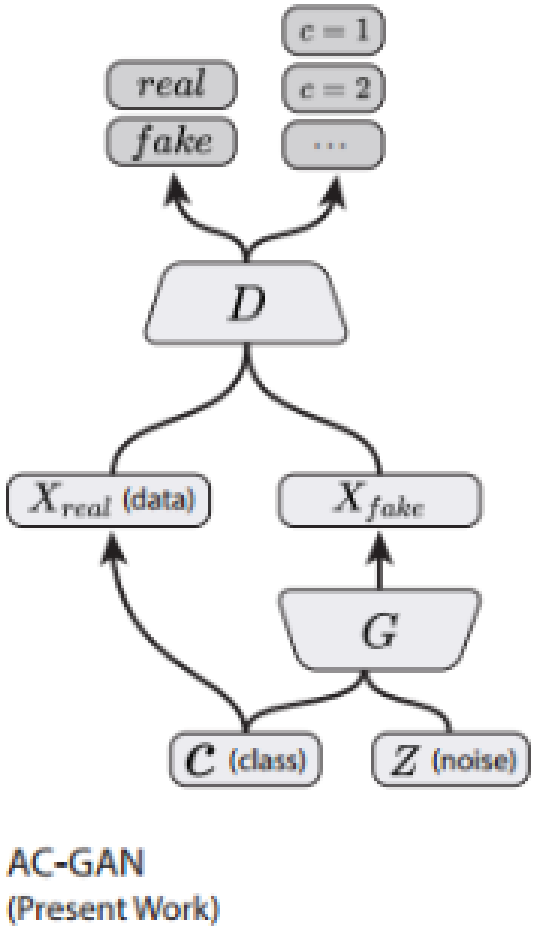

Figure 3: Model of an ACGAN

An AC-GAN, or Auxiliary Classifier Generative Adversarial Network, is an extension of the classic GAN method. In an ACGAN, the data is also given a classifier or label. The discriminator then has one more job in addition to being able to determine whether data is "real" or "fake". The discriminator now must also be able

to predict the class label of the given data. This then allows the generation of data of a specific type, according to the class label.

In the case of rock climbing problems, the different difficulties of the climbing problems could be used as the different classes. The ACGAN could then generate paths similar to those of the same difficulty in the training data set, which in this case would be the collection of user-made paths on the MoonBoard website. The generated image will then be a generic image of the 11 x 18 MoonBoard grid, and the holds that are used in the path will be circled.

2 METHOD

For our approach to the problem, we decided to use an auxiliary classifier GAN [2] with training in Tensorflow/Keras. ACGAN builds on "cGAN" by also coercing the discriminator to output classification, not just real/fake. Our extra classifiers are the different difficulty ratings of the climbing paths. These classifiers are based off the Fontainebleau system for classifying the difficulty of climbing paths, as this is a format used worldwide and supported by MoonBoard. This system grades paths on a range from 6B+ to 8B+ with 13 classes (labeled 1 - 13).

As previously mentioned, the MoonBoard website hosts thousands of examples of custom paths for MoonBoards. We gathered sets of example paths for each difficulty grade from 6B+ to 8B+ (some prior Github-hosted projects had already gathered sets of these paths according to difficulty grade to use in their projects on automatically grading the difficulty of a MoonBoard path). We trained our ACGAN with these sample paths according to their difficulty.

The different difficulties, however, are on more of an arbitrary scale, so the different difficulties are more relative than a gradual scale. We opted to try using ordinal regression in the classification of the classes to deal with this difference. The class labels will then be represented with modified one-hots; e.g. 3 = [1, 1, 1, 0, ...].

The MoonBoard itself will be modeled computationally as a three dimensional array: [x, y, z]. x and y will refer to the x and y coordinates on the 11 x 18 MoonBoard grid, and z will refer to the depth. Depth in this case refers to the order of the holds: starting, intermediate, and ending holds. 0 represents the starting holds depth, 1 represents the intermediate holds depth, and 2 represents the ending holds depth. At each point in this vector, the hold is represented as either a 0 or 1, signifying that the hold is not present in the path or it is used, respectively.

To visualize the climbing path, we first started with a blank image of a MoonBoard. Then, using the vectors output by the ACGAN, we circled the holds that would be used in the route. The colors of the circles depended on the depth, or order of the holds. The starting holds (depth 0) were circled in green, the intermediate holds (depth 1) were circled in blue, and the ending holds (depth 2) were circled in red. To be able to circle the correct position, the coordinates of each hold on the image were hard coded. The sample output was something similar to the following figure.

2.1 Data

As discussed, all MoonBoard data is user-generated, thus there are thousands of problems which we can access for training data.

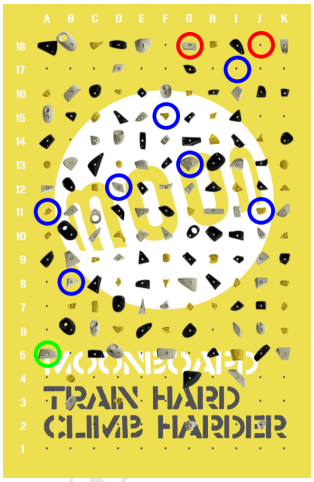

Figure 4: Example of a MoonBoard climbing path. "Bowl of Ramen," 6B+, by user catplan

Labeled training data was pulled from the MoonBoard website for all problems using the "2016 Holdset," the oldest MoonBoard configuration which also has the most training data available at present (n=31335). Table 1 displays the distribution of problems by grade (difficulty), with 6A+ being the easiest problem MoonBoard and 8B+ being the hardest.

Table 1: Distribution of Problem Data

Grade	n.	Grade	n.	Grade	n.
6A+	3	7A	4278	7C+	429
6B	2	7A+	3264	8A	213
6B+	10544	7B	1488	8A+	56
6C	3459	7B+	1705	8B	33
7C+	4510	7C	1320	8B+	31

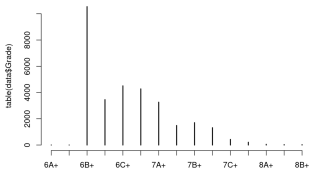

Figure 5: Distribution of Problem Data

2.2 Training Details

Through preliminary testing with handwritten digit generation, we identified a number of strategies to help stabilize training: one-sided label smoothing (with alpha=0.9) and noise on the ground truth data fed to the generator (with a probability of being flipped of 0.05). We also found that passing the training data as binary (i.e., black and white as opposed to greyscale) was sufficient to train the generator to produce nearly-binary output. Lastly, for

the MoonBoard GAN, we experimented with training the classifier using ordinal regression as opposed to categorical crossentropy.

3 EXPERIMENTS

We proceeded to search through a large number of possible architectures and hyperparameters in hopes of developing a suitable GAN. Tables 2 and 3 show the primary architecture modifications that we first attempted. For each, we also varied the dimensionality of the latent space (between 60 and 120) and whether to train the classifier with ordinal regression.

Table 2: Generator Architectures

	G Kernels	G Strides	G Filters
A	[3, (6, 4), (11, 7)]	[1, 1, 1]	[32, 16, 3]
B	[6, 6]	[1, 1]	[16, 3]
C	"	"	[32, 3]
D	[6, 6, 6]	[1, 1, 1]	[32, 16, 3]
E	[7, 7, 7]	"	"
F	[3, 6, 7]	[1/2, 1, 1]	[32, 16, 3]

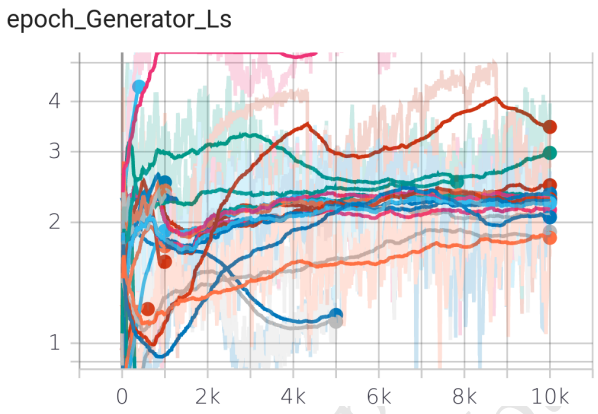

Figure 6: Generator Loss

Figures 6 and 7 show the overlaid losses of all of the discriminators and generators tested. As can be seen, the majority of all configurations reached some degree of stability (indicated by a convergence, rather than a divergence towards 0 or infinity).

After these models, we also tested a number of multi-layer perceptron models and searched through a variety of learning rates.

Unfortunately, despite much searching, all of our models suffer from mode collapse. In other words, despite reaching some amount of convergence, they all produce nearly-identical output regardless of noise that is input. For example, Figure 8 displays an output of one model after 10k iterations. Each column represents a difficulty level, and each row should contain a unique problem. However, all problems in a given column are identical.

3.1 User Survey

After generating paths that are traversible, we will have a few rock climbers actually attempt them on a local MoonBoard. After attempting the generated path, the climbers will fill out a short survey based on their experience with the generated path. They will be asked to rate their enjoyment of the path on a scale of 1 - 5 and how accurately the path matched the input difficulty on a scale of 1 - 5.

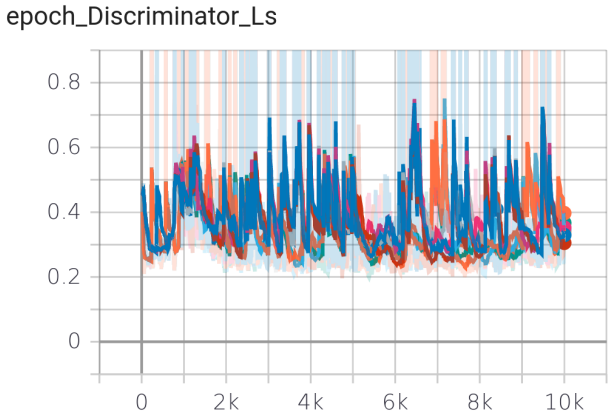

Figure 7: Discriminator Loss

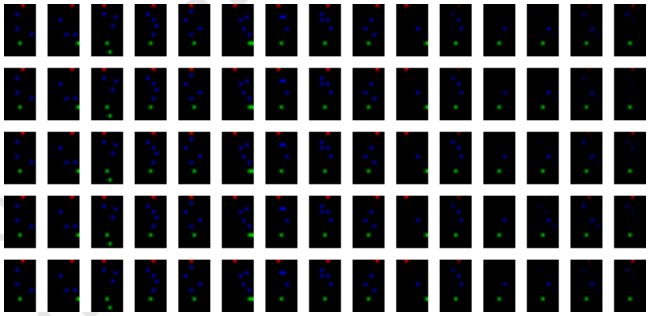

Figure 8: Example of Mode Collapse

Table 3: Discriminator Architectures

	D Kernels	D Strides	D Filters
A	[(11, 7), (6, 4), 3]	[2, 1, 1]	[8, 16, 32]
B	[4, 4]	[1, 1]	[8, 16]
C	"	"	[16, 32]
D	[4, 4, 4]	[1, 1 ,1]	[8, 16, 32]
E	[(9, 6), (9, 6), (9, 6)]	"	"
F	[7, 4, 3]	[2, 1 ,1]	[8, 16, 32]

4 CONCLUSION

4.1 Future Work

As discussed, so far we were unable to finish creating a stable generator for rock climbing problems. Future work could work on adjusting the hyperparameters or the algorithm to better work with the discrete differences between the problem difficulties, and ensure that there exist some starting and ending holds. The rest of the future work ideas is dependent on first fixing the system itself.

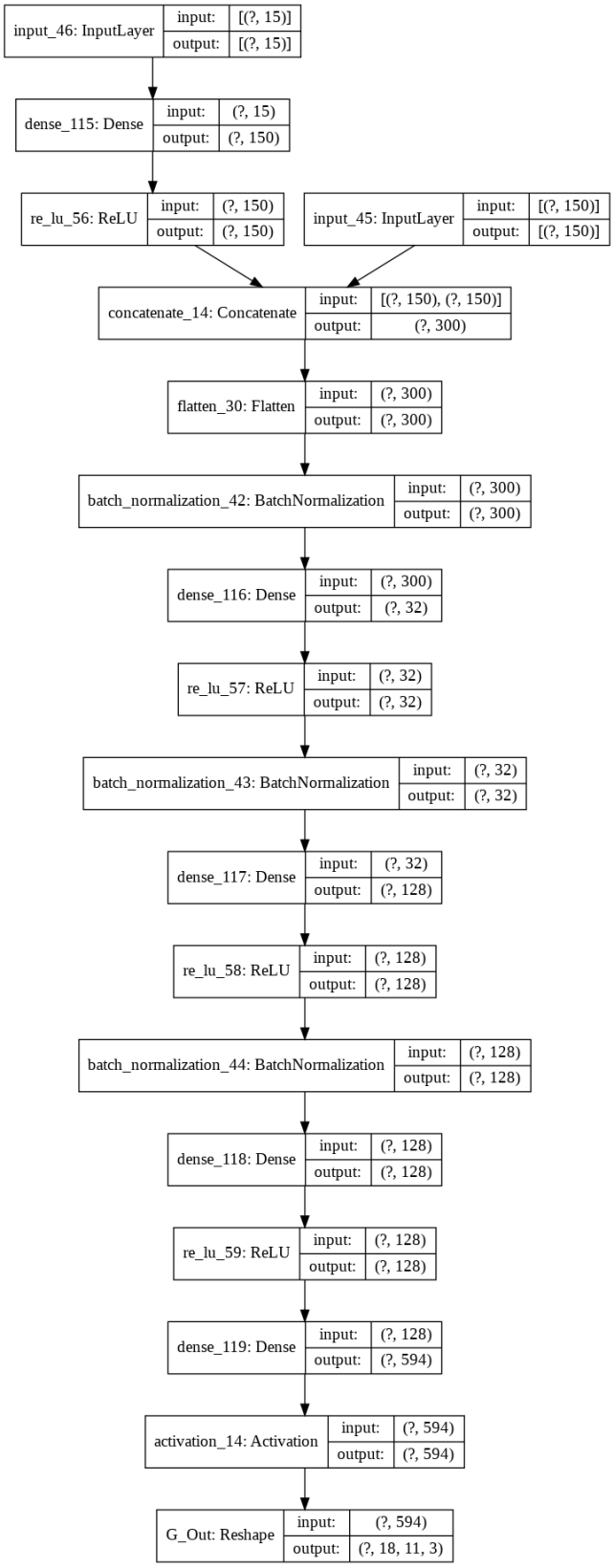

Figure 9: Example G Architecture

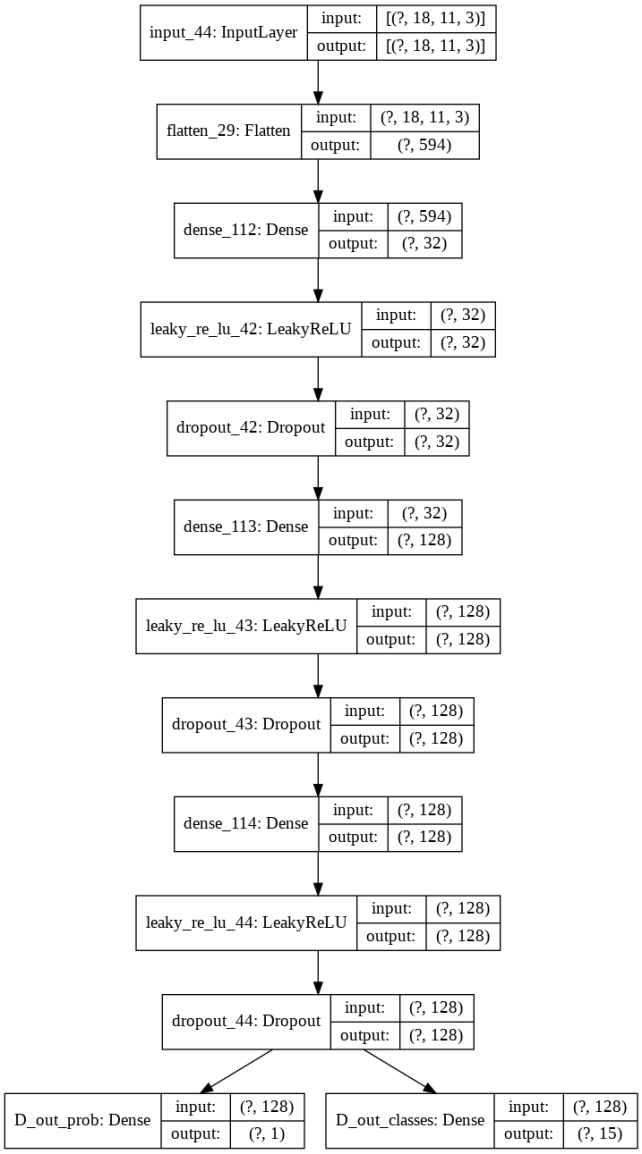

Figure 10: Example G Architecture

For the evaluation, we would like to be able to first generate a set of example paths using our trained ACGAN. Using this set of paths, we would test them with both (1) our discriminator and (2) one of the aforementioned Github-hosted projects (if we can find one that works reliably); this is to verify proper route-generation for the grade. We would also search for nearest-neighbors of the generated problem to ensure that the generator is not simply memorizing the training set, as well as if the generated collapsed and is just outputting the same path. Then we would move to evaluating the paths in person. The generated paths would be evaluated by members of a local rock climbing club (Cove is a member of this club!). After completing the suggested path, the members would respond with

whether the path matched the given difficulty and their rating of how much they liked the given path (how challenging/fun it was). Again, this is all once a working rock climbing problem generator is established.

Future work could then craft this system into a fully functioning tool for climbers to use on the go. This could include creating a website or app that allows user interaction and access to the online community. This means a rating or star system for the generated paths could also be included, allowing users to know which paths were most enjoyable. Problems are usually given fun, quirky names, as shown in the problem in Figure 2 which was named "Bowl of Ramen." There may be a way to even generate these names alongside the generated problems!

This solution could also be improved in many ways. As discussed, there were many difficulties with generating traversable paths of the different difficulties, especially with those with insufficient training data. Other generative models may do a better job at generating these problems with better accuracy. Experimenting with different models and different approaches may result in a working (and maybe even easier) solution to the problem.

ACKNOWLEDGMENTS

Acknowledgements

REFERENCES

[1] Sarmiento Dobles and Satterthwaite. 2017. *Machine Learning Methods for Climbing Route Classification.* Technical Report. Stanford University, Stanford, CA.
[2] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2016. Conditional Image Synthesis With Auxiliary Classifier GANs. arXiv:stat.ML/1610.09585