CS230

Recurrent Neural Network for MoonBoard Climbing Route Classification and Generation

Final Report

Yi-Shiou Duh
Department of Physics
Stanford University
allenduh@stanford.edu

Je-Rui Chang
Department of Bioengineering
Stanford University
jrc612@stanford.edu

1 Introduction

MoonBoard is one of the most effective tools for climbing training. Each MoonBoard is an identical short climbing wall, with 18 rows and 11 columns of holds (left panel of Figure 1). Climbers can only use circled holds to climb from the start (circled in green) to the goal (circled in red). Each problem has a suggested grade, and some high-quality problems are selected into the benchmarked list. MoonBoard comes along with a mobile app which allows users to access the global database of routes uploaded by the climbing community. The app also has filters that allow users to select benchmarked problems or problems within a certain grade.

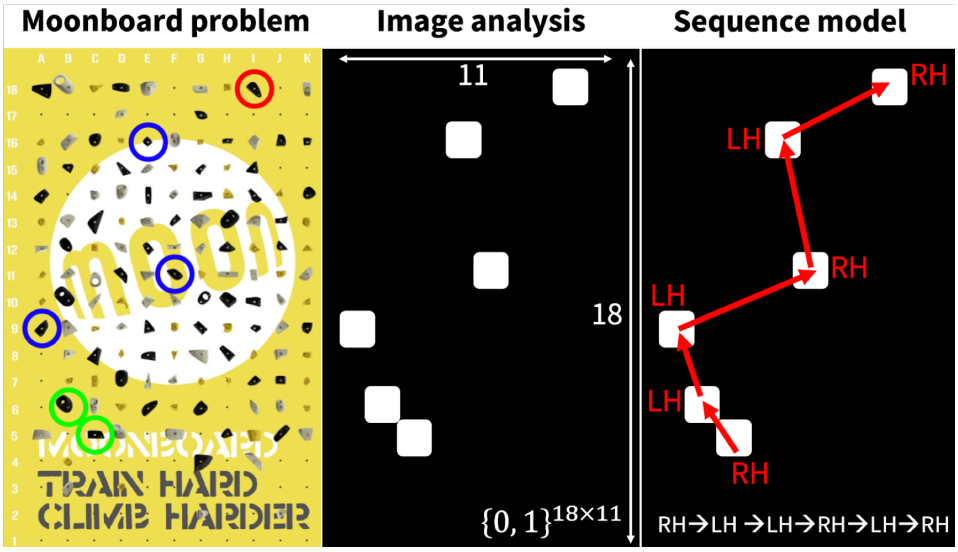

Figure 1: Example of a MoonBoard problem and the Machine learning model. Previous literature directly analyzed MoonBoard problems as a $\{0,1\}^{18\times11}$ matrix. Here we proposed a new sequence model as a more natural representation of a MoonBoard problem. The hand sequence was produced by BetaMove in order to mimic human climber's hand sequence from the start to the goal.

From a climber's point of view, it is beneficial to have a route generator that can produce more difficult problems and a grade predictor that can predict their grade. Machine learning (ML) has been widely applied to sport analysis. However, rock climbing has not been well-analyzed so far

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

because of the complexity of the climbing movement. Current ML models cannot accurately predict the problem's difficulty, not to mention classify climbing techniques or generate reasonable problems. We therefore propose to fill this gap using recurrent neural networks (RNN) with the aid of a large database from MoonBoard.

In this paper, we demonstrated an improved version of the grade predictor - **"GradeNet"** - using a new move preprocessing pipeline **"BetaMove"**(beta is a climbing jargon which means the method to climb a problem). This approach allows us to outperform other existing classifiers in literature and even reach human-level performance. This advance shows that move sequences generated by BetaMove is a more natural representation than unprocessed raw image. Based on this new sequence input, we then built a route generator **"DeepRouteSet"** and assembled our networks together into a complete pipeline: DeepRouteSet can generate new MoonBoard problems, and GradeNet can predict its grade. This work not only pioneers the RNN architecture for climbing route classification, but also allows future work to apply transfer learning to learn style-labeled data. The link to our Github repository can be found here: https://github.com/jrchang612/MoonBoardRNN, and a demo website showing our generated problems can be found here: https://jrchang612.github.io/MoonBoardRNN/website/.

2 Related Works

Table 1: The summary of previous grade predictors.

	Ref [2]	**Ref [5]**	**Ref [3]**	**Ref [1]**	**Ref [1]**	**Ref [1]**
Algorithm	CNN	GCN+text embedding	CRDL + DE-CTW	LSTM	MLP	random forest
Performance	Accuracy 34%	AUC 0.73	Differentiate hard vs easy 64%	Accuracy 29.9%	Accuracy 35.6%	Accuracy 16.5%

Previous literature applied end-to-end, non-sequential models directly to unprocessed MoonBoard problems, and we summarized their approaches in Table 1. For the grade classification, Convolutional neural networks CNN [2] directly uses a $\{0,1\}^{18\times 11}$ matrix as an input, and results in 34% accuracy. Tai et al also encoded the MoonBoard problem as a $\{0,1\}^{18\times 11}$ matrix, but used GCN and text embedding technique to obtain an AUC of 0.73 [5]. Andrew Houghton tried many different machine learning frameworks, including RNN[1]. However, in his work, the input sequences are not processed to mimic climbing sequences, which ended up with poor accuracy. Kempen used a non-machine-learning method that can only differentiate easy from difficult problems, but the accuracy of their algorithm was still only 64% [3].

Regarding route generation, Houghton created a MoonBoard route generator using LSTM [1]. Their training set input sequences are not preprocessed to mimic climbing sequence, so their generated problems include redundant holds and strange move sequence. For the non-machine-learning approach, Phillips et al developed an automatic route setter, "StrangeBeta", based on the mathematical characteristics of a strange attractor [4]. However, they described the generated routes with a special language called CRDL, which can be ambiguous even to climbing experts, and their system is not applied to MoonBoard.

In our opinion, MoonBoard problem is more similar to an NLP problem than a computer vision problem because of two major reasons: (1) With graphic representation, the $\{0,1\}^{18\times 11}$ matrix used in CNN is too sparse. (2) Climber follows a physically reasonable sequence to climb up, from the start hold to the goal. We therefore decided to implement RNN with proper preprocessing on the climbing route grading and route generation problem.

3 Methods

As shown in Figure 2, the overall pipeline of our project started with BetaMove, which preprocesses the image of a MoonBoard problem into a move sequence. BetaMove simulates a variety of climbing sequences and computes a success score, just like human climbers trying out climbing sequences to find out the easiest one. Each sequence is composed of many moves, in which each move's success score can be parameterized by the relative distance between holds and the difficulty scale of each hold. We then used beam search algorithm (with beam size = 8) to find the best sequence. The parameters

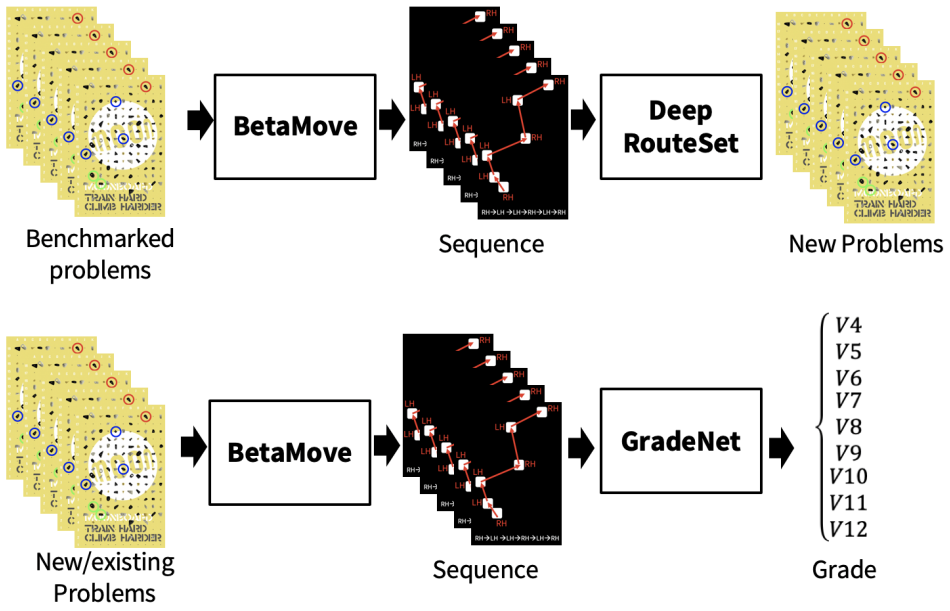

Figure 2: The AI pipeline of our project. We first use BetaMove to preprocess MoonBoard problems into a move sequence that is similar to human climber's prediction. The preprocessed move sequences were then used to train both a route generator, DeepRouteSet, and a grade predictor, GradeNet.

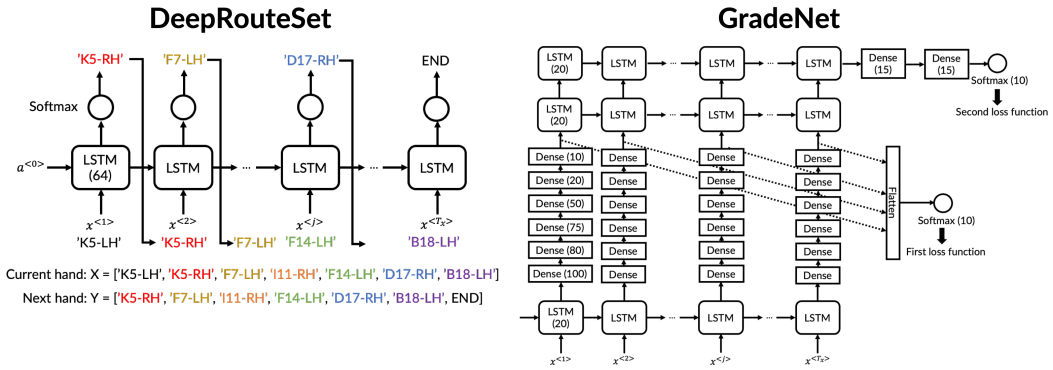

Figure 3: Model architecture of DeepRouteSet and GradeNet. DeepRouteSet is a route generation model which is trained with BetaMove-preprocessed sequences. GradeNet is a sentiment classification model trained with the mapping between BetaMove-preprocessed sequences and their corresponding grades. The trained GradeNet can predict the grade of both existing and generated problems. The number in each box indicates the number of neurons in each layer.

of BetaMove were tuned to match the 20 dev-set hand sequence predicted by a climbing expert, and we evaluated the performance of the model by comparing the predicted move sequences of BetaMove on another 20 MoonBoard problems with the ones predicted by a climbing expert. The prediction of BetaMove exactly matched the move sequences predicted by a climbing expert in 95% of the test problems (19/20). This physically meaningful move sequence injects human's insight to improve our input structure.

Both our route generator, DeepRouteSet, and our grade predictor, GradeNet, were trained using the sequence data generated by BetaMove. Similar to music generation, DeepRouteSet learns the pattern between adjacent moves from existing MoonBoard problems, and is able to generate new problems using these patterns. In GradeNet, We implemented 2 novel techniques. First of all, the input of GradeNet is a move sequence produced by BetaMove, and each move is embedded into a 22-dimensional vector. Secondly, to grade a MoonBoard problem, both hidden layer prediction and overall grade prediction need to be considered at the same time. This spirit was embodied in the

two-stage architecture of GradeNet , as shown in the right panel of Figure 3. In the first stage, the embedded input sequences pass through the LSTM layer, followed by 6 dense layers. The output sequence of the 6th dense layer is combined and flattened for the first grade prediction. In the second stage, the output of the 6th dense layer was fed into another two LSTM, followed by 2 dense layers for another grade prediction. The final loss function was the sum of the two categorical cross entropy. Batch normalization, dropout, L2 regularization, max pooling, and average pooling were all tested, but none of them shows significant improvement. The training history of GradeNet was described in Appendix A3.

4 Dataset and Features

MoonBoard's official website hosts a database of problems uploaded by the global climbing community. We scraped 30634 MoonBoard problems using modified Selenium scraping script based on gestalt-howard's Github project moonGen. Because everyone can upload problems and give it a grade, additional preprocessing is required to filter out problems from amateur users. First, we chose the Hueco scale grading system (scale from V4 to V14, and is more prevalent in Asia and in the US) instead of the Font scale (scale from 6B to 8B+ and is more prevalent in Europe) because both 6C and 6C+ in the Font scale map to V5 in the Heuco scale, and both 7B and 7B+ map to V8. Second, we excluded all V14 problems (24 problems) because many of them are clearly mislabeled with very easy moves. Third, we exclude some problems we found in the error analysis: 1 problem has mislabeled start holds, and 6 problems have unrealistically large numbers of holds. Finally, we removed all problems without repeats, which means no other users have verified the route and usually indicates low quality and questionable grade. The final grade distribution is shown in Appendix Figure A.1. The remaining 25096 problems were divided into training, dev, and test set, with 20157, 2442, and 2497 problems, respectively. The distribution is highly skewed toward easy problems, and we use weighted training to counteract this imbalance. BetaMove preprocessed the gleaned dataset into move sequence for all further training and testing.

5 Experiments, Results, and Discussion

To evaluate the grade prediction accuracy of GradeNet, we estimated human-level performance. As described in detail in Appendix A2, the human-level accuracy of exact match is only 45%. This is because it is difficult to assign a discrete grade from a continuous spectrum of climbing difficulties. If you allow the predicted grade to be off by 1 grade, the accuracy increases to about 87.5%.

Table 2: Performance of GradeNet and previous classifiers. The performance of GradeNet is close to human level performance (HLP). Results from Ref[2], Ref[5], and Ref[1] are listed for comparison. In Ref[1], the author tried LSTM, MLP, and random forest algorithm, and here we only reported their data from MLP as it had the best performance.

	HLP	**GradeNet Training**	**GradeNet Dev set**	**GradeNet Test set**	**Naive RNN Dev set**	**Ref[2] CNN**	**Ref[5] Graph NN**	**Ref[1] MLP**
Accuracy	45.0%	64.3%	47.5%	46.7%	34.7%	34.0%	Not reported	35.6%
$\pm$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

The performance of GradeNet, in comparison with existing MoonBoard grade predictors, are summarized in Table 2. GradeNet not only outperformed other classifiers including CNN[2], MLP[1], Graphical neural network[5], but also reached human-level performance. GradeNet also surpassed a naive RNN, in which the hold lists of the problems were directly fed as input without the preprocessing of BetaMove. This indicates that the improvement primarily comes from the injection of human insight through the BetaMove-preprocessed move sequence.

The confusion matrix in Figure 4 shows that the prediction of GradeNet mostly agrees with the actual grade within ± 1 error. For the majority of problems (V4-V7), the confusion matrix is symmetric. We believe that the deviation comes from the subjective nature of grading, and the difficulties to

categorize problems with difficulties between two grades. For problems harder than V8, however, our model underestimated their grade. This is probably because in those difficult problems, there are only 1 or 2 crux moves while most other moves are around V8. For instance, a V11 problem may only have one V11 crux move. We therefore expect an attention model can better tackle this challenge.

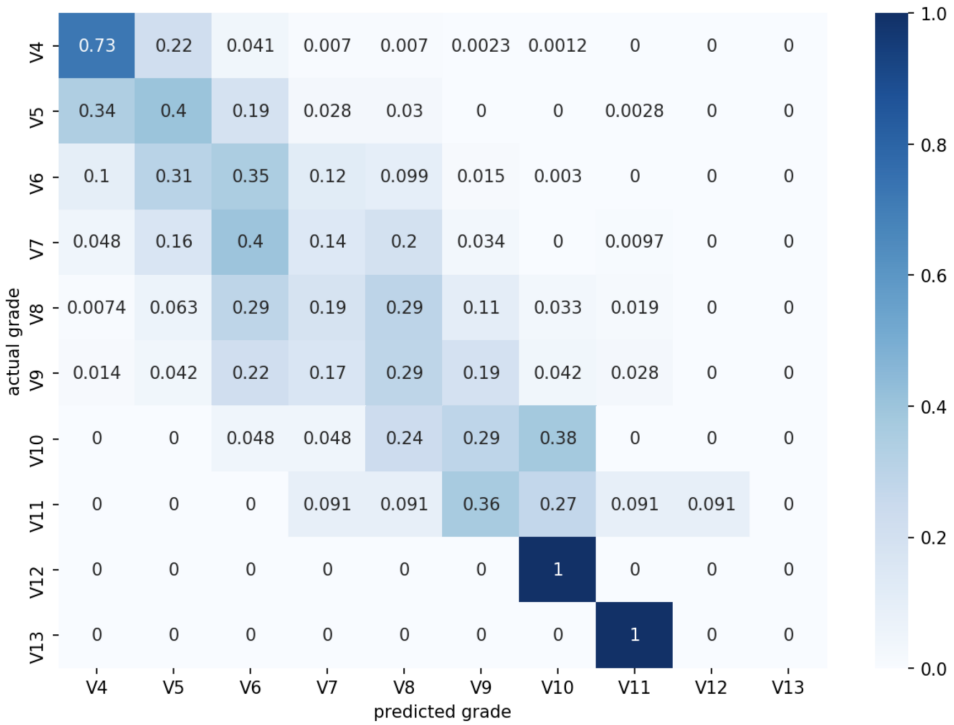

Figure 4: The confusion matrix of GradeNet. The predicted grade and actual grade distributed along the main diagonal. For problems easier than V8, GradeNet performs well. For harder problems, GradeNet underestimates their difficulty.

For our route generator, DeepRouteSet, we asked a climbing expert to evaluate the quality of generated problems by 2 criteria: (1) Is this problem decent/reasonable? A reasonable problem should not have obvious redundant/unused holds, weird sequences that can cause injury, ugly moves, or the coexistence of very easy and very hard moves. (2) Is this problem a high-quality one that can be a candidate for benchmark? High quality problems usually have a natural climbing flow, guidance for climbers to apply a natural posture, and consistent difficulties of moves. The comparison of our model with the route generator reported by Houghton et al[1] and the latest 50 MoonBoard problems is summarized in Table 3. The differences in the quality is striking. Problems generated by DeepRouteSet showed much less poor qualities compared to the existing model (examples shown in Figure A.4). Furthermore, DeepRouteSet produced 80% of high quality problems compared with 20% in Ref[1]. Note that in Ref[1], LSTM was also used for route generation. This striking difference indicates that the human insight from BetaMove can help improve the quality of a route generator.

Table 3: Performance of DeepRouteSet, in comparison with the route generator reported in Ref[1]

	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[1]	40	35%	35%	40%	20%

6 Conclusions and Future Work

In conclusion, we demonstrated that with BetaMove, we are able to inject human insights into the machine learning problems. By preprocessing a MoonBoard problem into a proper move sequence, the accuracy of our grade predictor, GradeNet, reaches near human-level performance. In addition, our strategy also optimizes our route generator, DeepRouteSet.

In the future, we hope to use the same framework to generate a climbing style classifier. However, currently we are not able to collect enough labeled data to train the model. We tried transfer learning using the pretrained weights from GradeNet, but the outcome was poor. We also hope to construct a user interface for displaying our generated problems and collect feedback.

7 Contributions

Je-Rui Chang adapted the source code from gestalt-howard's Github project moonGen to scrape the MoonBoard data and organized the dataset. Yi-Shiou Duh developed BetaMove and analyzed the performance of "BetaMove". Je-Rui Chang developed the GradeNet and analyzed the performance of GradeNet. Yi-Shiou Duh performed the analysis of human-level performance. Yi-Shiou Duh adapted the source code from a Coursera problem exercise "Improvise a Jazz Solo with an LSTM Network" and built DeepRouteSet. Yi-Shiou Duh collected the labeled data for style analysis from himself and his climber friends. Je-Rui Chang modified the source code from Andrew Houghton's Github project moon-board-climbing to set up the website. Je-Rui Chang maintained the GitHub repository. Yi-Shiou Duh and Je-Rui Chang contributed equally to the final report and final video.

8 Acknowledgement

The authors appreciated the advice from Colin Wei during the early phase of this project. The authors appreciated the permission from Howard (Cheng-Hao) Tai for using his scraping code. The authors appreciated the permission from Andrew Houghton for using his website code. The authors acknowledged the help from Zhi-Xuan Jian for evaluating the difficulties to grab holds and providing the estimate of human-level performance. The authors acknowledged the help from Ming-Shu Gan in filming and editing climbing videos and evaluated the estimate of human-level performance. The author acknowledged several climbers for providing their video and providing evaluation on generated moonboard problems: Maya Madere, Jiun-Jie Tzeng, Hung-Ying Lee, Keita Watabe, Chia-Hsiang Lin, Joey Wen, Jeff Lau, Hoseko Lee, Nathaniel Coleman. The authors acknowledged insightful discussion with Lucien Lo, Yi-Ting Duh, Liting Xu. The authors thanked the help from people who take part in style surveys.

9 Conflict of Interests

The authors declare no conflict of interests with any existing enterprises. This work is not funded or supported by MoonBoard Company.

References

[1] andrew-houghton/moon-board-climbing: Using Neural Networks to generate new rock climbs for the moon board (jointly developed by Andrew Houghton, Ryan Mann and Jemma Herbert).

[2] Alejandro Dobles, Juan Carlos Sarmiento, and Peter Satterthwaite. Machine Learning Methods for Climbing Route Classification. Technical report, 2017.

[3] Lindsay Kempen. A fair grade : assessing difficulty of climbing routes through machine learning, 2019.

[4] C. Phillips, L. Becker, and E. Bradley. Strange beta: An assistance system for indoor rock climbing route setting. *Chaos*, 2012.

[5] Cheng-Hao Tai, Aaron Wu, and Rafael Hinojosa. Graph Neural Networks in Classifying Rock Climbing Difficulties. Technical report.

A Appendix

Appendix A1: Grade Distribution of our Dataset

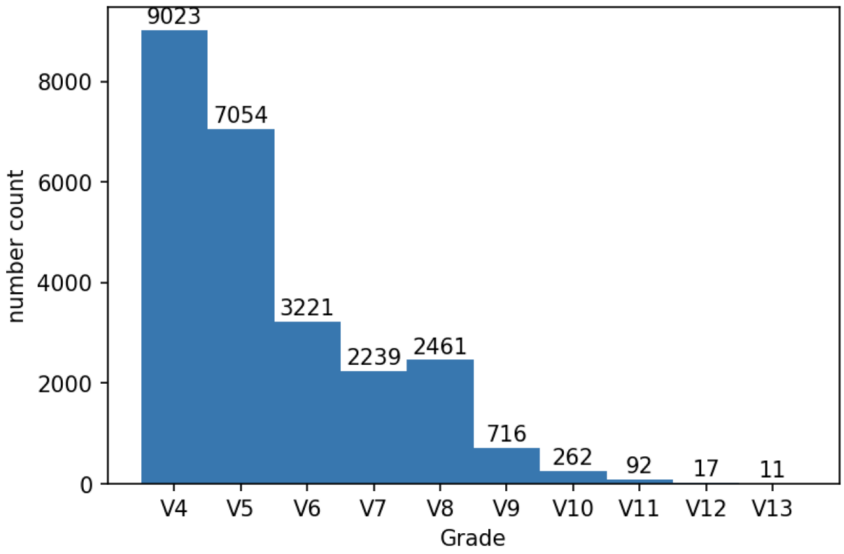

Figure A.1: Distribution of V-grade of the MoonBoard dataset.

Appendix A2: Human-level Performance

To help us evaluate the performance of GradeNet, we first estimated human-level performance on the MoonBoard grading problem. In the previous study [2], how much the user-rated grade on MoonBoard app matches the grade provided by route-setters is used as human-level performance. However, we don't think this is a fair comparison because climbers usually assign grades after they have climbed the problem, and there is a climbing convention to follow the original grade unless there is significant error as a respect to the route setter's effort.

To combat this problem and make a fair estimation of human-level performance, we instead asked 3 climbing experts to blindly estimate 40 climbing problem's grades without actually climbing it. As shown in Table A1, it is very difficult even for climbing experts to determine the grade without actually climbing it. However, if you tolerate one grade off from the predicted grade, the accuracy can be up to 87.5%. From the feedback of our climbing experts, there are several reasons for their low accuracy: (1) It is very difficult to grade a problem without actually climbing it. (2) Climbers of different body types are more suitable for different problems. (3) When a climber believes that the problem falls between 2 grades, they don't know which grade they should assign.

Table A.1: Human level performance of estimating the grade of a MoonBoard problem without actually climbing it.

	Exactly meet with the grade	**Allow one grade difference**
Climbing expert 1	47.6%	82.5%
Climbing expert 1 (second try)	30%	77.5%
Climbing expert 2	42.5%	87.5%
Climbing expert 3	45%	87.5%
Estimated HLP	45.0%	87.5%

Appendix A3: The training of GradeNet

The input of GradeNet is a move sequence produced by BetaMove. Each move is embedded into a 22-dimensional vector. For example, as shown in Figure A.2, BetaMove embedded a single move from hold 3 to hold 4 into a 22- dimensional vector that will be used in GradeNet. This vector includes the target hold's position (x_4, y_4), the relative distance to the previous 2 holds $(v_{34x}, v_{34y}, v_{24x}, v_{24y})$,

the difficulty scales of all three holds (f_2, f_3, f_4), the placement of feet, and the estimated success scores of each move.

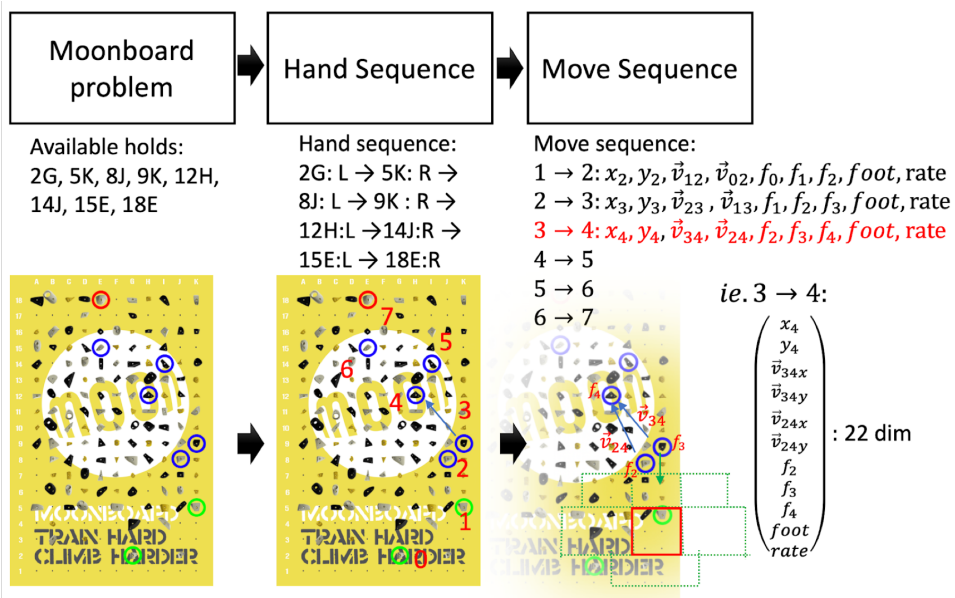

Figure A.2: The 22-dimensional embedding of BetaMove.

During the training process (Figure A.3), the input was weighted to combat the problem of uneven input class distribution (the weights were adjusted after training for 100 epochs). After the training for 200 epochs, the training results of the model are summarized in Table 1 of the main text.

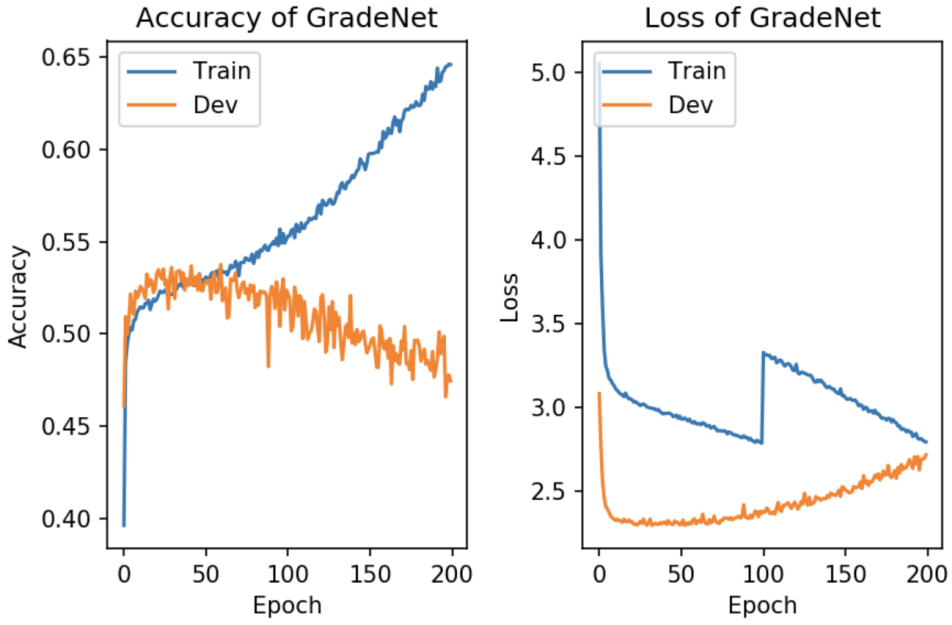

Figure A.3: Training curve of GradeNet. At Epoch 100, the class weight was adjusted, so the loss function of the training set suddenly increased.

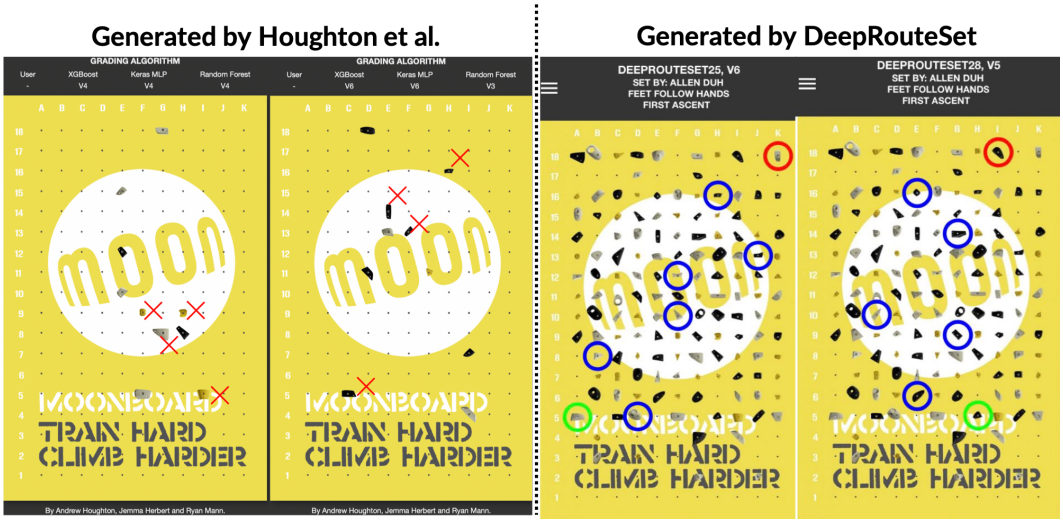

Figure A.4: Comparison between the generated problems by Houghton et al and DeepRouteSet. The holds that are labeled with "X" are the ones that are redundant.