Related Task Self-Supervised Learning for Rock Climbing Route Rating

Jack Hlavka
Stanford University
jhlavka@stanford.edu

Ethan Harianto
Stanford University
eharianto@stanford.edu

Abstract

Rock climbing is an increasingly popular activity, leading to indoor gyms creating their own routes. The grading of these routes on a scale from V0 to V17 is a controversial and subjective process. Further, no large dataset exists to learn this problem. This paper proposes a method of using self-supervised learning on related tasks in adjacent labeled datasets to learn representations to allow training with fewer examples. This method is based on previous research that found that self-supervised learning on hard tasks related to the target task leads to better results. Using this method, a YOLOv5 object detection model was trained on a dataset of climbing holds, and then given a new regression head to predict the grade of a climbing route. With just 52 training examples, this model reached a test MSE of 1.47, comparable to a beginner or intermediate human climber given the same information. This method is a powerful way to learn problems with limited data through related tasks on similar datasets.

1. Introduction

Rock climbing has grown increasingly popular in the last few decades, especially in indoor gyms, eliminating the barrier of traveling to remote locations. While outdoor climbing routes are typically graded by the person making the first ascent, then verified by others who climb it after, this grading system is quite subjective requiring multiple attempts and climbers to bring a level of objectivity to the given grade [4]. Indoor climbing is even more subjective, as often the only person who has input on the grade is the route setter, someone who does not have the ability to approach the route from an uninformed climber's perspective due to their creation of the route. Therefore, there is often a significant difference in difficulty between routes of the same grade and especially, inconsistency between grades at different locations.

Grading is very subtle, and for a human, generally requires climbing the route. The difficulty is highly dependent on the exact angles at which it's possible to apply force and the various body positions that the holds force the climber into. It is quite possible for the same set of physical holds to be used to construct two routes of extremely different difficulty, just based on their arrangement. For example, a hold that has a good grip on one side but is smooth on the other could make the route much easier if oriented to take advantage of its good side but help very little if rotated just ninety degrees.

This paper will focus on bouldering routes, which are climbed without ropes and are no more than about 15 feet high. In indoor gyms, these routes are usually distinguished by their color, such that a route is an isolated group of holds all of the same color. The grading of these routes is usually done using the Vermin scale, in which routes are graded from V0, the easiest route, to V17, the currently hardest climbed route. While generally, routes are only integer values, some gyms have 'slash grades' (e.g. V6/7) or 'plus grades' (e.g. V5+ or V4-). Most indoor climbing facilities have no routes above V10, with the majority of the routes in the V3 to V6 range, the range of an intermediate climber.

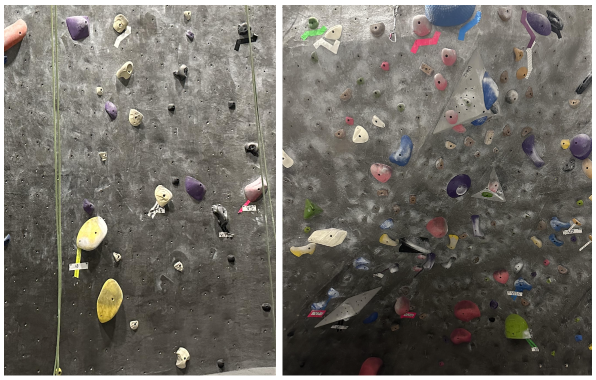
Figure 1. An example V0 in white with large, easy-to-grab holds (left) and an example V8 in green, with very few, far-apart, and small holds (right)

To attempt to standardize and simplify the grading process, this paper attempts to create an automated grading system. The input to this algorithm is an image of a complete climbing route. We then use a CNN to output a Vermin scale grade. This is treated as a regression problem, as V grades are in theory continuous, and are presented in a limited number of forms to reduce complexity and represent the difference in difficulty that people with different body types experience. For example, the algorithm may take in a given image and output the value 3.2, which would, for gym purposes, be considered a V3.

This is a difficult problem due to the importance of hold type and orientation, as well as subtle relationships between different holds that affect whole-body movements. A successful model must be able to understand not just what parts of the image are holds, but also develop some understanding of how holds relate to each other. One such understanding might be that a very small hold is very hard to use for a handhold but perfectly acceptable as a foothold. A much more nuanced understanding will be necessary for an accurate model.

As only limited data is available for this task, the focus will be improving results over a simple CNN. The primary method tested will be self-supervised learning to pre-train models using a domain-specific task to improve model understanding.

2. Related Work

Self-supervised learning is a well-researched area of computer vision, with consistent results showing that it can boost the results of pre-trained models. Zhai et. al. found that self-supervised learning on ImageNet resulted in competitive results using the labels of just 1% of the dataset. They also attempted self-supervised semi-supervised learning, where labels for each image were predicted, and then used for supervised training on the entire dataset. This approach reached a state-of-the-art top-5 accuracy of over 91% using a fraction of 10% of the data [19].

Newell et. al. examined the utility gained by self-supervised learning. They quantified each scenario by how many labeled examples are 'saved' by pre-training, representing the budget saved while labeling. They determined that utility is highest in the low data scenario, finding diminishing returns as the labeled dataset grows. Further, they determined that while large models perform worse than smaller ones in low data scenarios, pre-training allows larger models to become better than smaller models [15].

El-Nouby et. al. compared pre-training on an unrelated dataset, as in the common paradigm of using a model pre-trained on ImageNet, and self-supervised learning on the target dataset, finding that the self-supervised learning offers superior results on some datasets. This result is significant as these datasets outperform ImageNet for pre-training despite being much smaller and not using labels [5].

Finally, Ericcson et. al. compared 40 state-of-the-art models with both self-supervised learning and transfer learning techniques from ImageNet. They determined that self-supervised learning has generally surpassed generic pre-trained transfer learning, indicating that learning representations closer related to the target task are more valuable than having the extremely large ImageNet dataset [6].

Some authors tested the effect of the closeness of the target dataset to the pre-training dataset. Goyal et. al. tested the possibility of self-supervised pre-training using images unrelated to the target task. While they achieved comparable results by pre-training on random images, doing so required a total of one billion images in the pre-training dataset, which would be quite unfeasible to use [7].

Xu et. al. tested domain adaptation between the self-supervised and supervised tasks, determining that further work in optimizing the training algorithm was needed for comparable results when the domain was different [18]. Similarly, Bucci et. al. studied self-supervised learning of different datasets in different domains, achieving fairly significant results even when the target was different from the self-supervised datasets [2].

Finally, Guillaumin et. al. tested self-supervised pre-training on a related task. While attempting an image classification task, they pre-train with tags associated with the images. Even though many of the tags are unrelated to the image classes, pre-training still produced a sizeable positive effect on the classification accuracy, though it failed to have an effect on some classes [9]. This previous research concurs that pre-training is more effective when the task it is learning is as close as possible to the target task.

Goyal et. al. tested different self-supervised learning tasks and found that self-supervised learning becomes more effective when the self-supervised task is harder. They argue that common tasks like orientation classification are too easy to achieve optimal results and that, with harder tasks, better accuracy on the supervised task would be achieved [8].

Zoph et. al. tried a combination of pre-training and self-supervised training. They determine that whenever pre-training on a generic dataset is useful, fine-tuning a pre-trained model on the self-supervised task and then tuning that model on the supervised task results in even better results [20]. This is a significant result, as it indicates that even when self-supervised training is useful, it can still be correct to use a pre-trained model for the self-supervised task.

This previous research indicates that self-supervised learning is a powerful technique able to surpass pretraining, even in the general case. Further, self-supervised learning is successful even when the self-supervised dataset is unrelated to the target dataset. However, when the datasets are more related, the success rises, indicating that self-supervised learning should match the target task if possible. Self-supervised learning is also more effective when the self-supervised task is harder, suggesting that a task matching the target task rather than just, for example, detecting rotation, is more effective. Finally, even in the case when self-supervised learning is superior to pretraining, doing both is better than doing either individually. These results suggest that an effective use of self-supervised learning would use a similar dataset to the target dataset, use a task that is both difficult and contains elements of the target task, and would use a pretrained model to begin self-supervised learning.

3. Data

The algorithm in this paper utilizes two datasets, one target dataset and one self-supervised learning dataset. The target dataset is a collection of images of complete bouldering routes. As no such dataset exists online, this dataset was hand-collected and hand-labelled. With permission, pictures were taken of bouldering routes at the Stanford Climbing Gym, a relatively small gym with a high density of routes. Images were taken of all wall space in the gym, to cover every possible route. Images were cropped to show each route individually and then resized to the appropriate sizes for the model inputs. While this may appear to distort absolute distances in the images, all routes have the same starting and ending heights, meaning the resizing preserves distance relatively well, and is not worse than the effect caused by taking photos from different distances from the wall.

There were a total of 88 images of routes collected, encompassing all of the routes in the gym. To augment this extremely small dataset, each image was flipped horizontally and recolored in two new ways, resulting in a total of 528 images in the dataset. Further augmentations are not possible as all other orientations and image adjustments fundamentally change the identity of the route. Horizontal reflections don't do this as any route should be equally difficult as its symmetric counterpart, as humans are generally symmetrical.

Each route is associated with a grade, the ground truth label, which for the Stanford Climbing Gym is an integer between 0 and 8 inclusive. These grades were assigned by a team of route setters of different genders and body types, which, at least in theory, should remove some of the subjectivity of grading.

As this dataset is quite small, a 60-20-20 train-val-test split was used, with all versions of an image in the same split to avoid data leakage. This resulted in splits of size 312-108-108, meaning that the train split contained augmentations of 52 routes, and the val and test splits contained augmentations of 18 routes each.

The self-supervised learning dataset is the Blackcreed Climbing Holds and Volumes Dataset [1]. This is an object detection dataset, composed of images of climbing holds. Some images are of single holds, some are of sets of holds not attached to anything, and some are of entire routes. There are a wide variety of images in the dataset, from local gyms to images of international-level competition routes. Each image is resized to 640x640 pixels for processing. This resizing presents no issue as the distances between holds is not important and an image of a distorted hold is indistinguishable from an image of a hold shaped to be elongated or compressed.

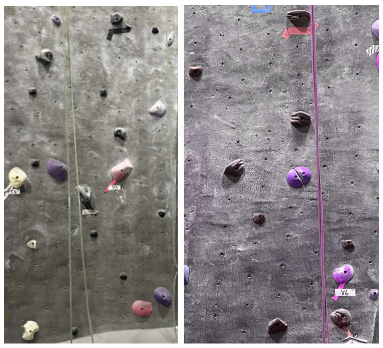
Figure 2. Two examples images from the dataset, one unmodified (left) and one color shifted (right)

There are a total of 4301 images in the dataset. Each image is augmented in three ways, through a combination of flips, rotations, and recoloring. In this case, vertical flips and rotations are acceptable as the overall identity of the route has no bearing on identifying holds. The dataset comes with a recommended 3876-277-148 train-val-test split, which was used for this project.

For each image that contains one or more climbing holds, bounding boxes were identified for each hold. These boxes are minimally sized to encompass the entire hold, and many of them overlap each other. Each box is assigned a class, though the only classes in the dataset are 0 and 1, which have no separate meanings. As such, all holds were unified into a single class for the purposes of this algorithm.

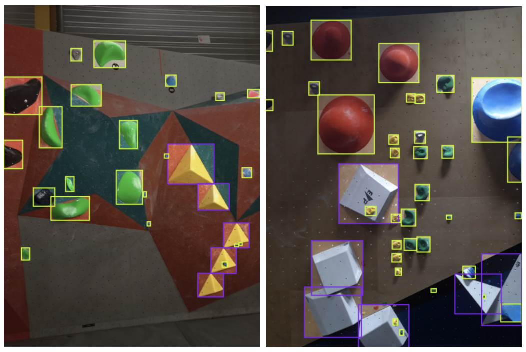
Figure 3. Two example images from the Blackcreed Climbing Holds Dataset with bounding boxes displayed. The colors of the bounding boxes represent different classes, though these classes are arbitrarily and inconsistently assigned.

4. Methods

4.1. Baseline Models

The baseline models in this paper are CNNs trained directly on the target dataset. This is the most straightforward method for a regression task on a dataset. These CNNs use mean squared error, which is the following loss function.

$$\ell = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

The architecture of such a CNN is a series of convolutional layers with normalization and pooling layers interspersed, then a head with a series of fully connected layers, the last of which has only a single output node, which has value $\hat{y}_i$.

The model architecture used is the ResNet architecture [11]. ResNet is a classical architecture in which the residual of the previous layer is repeatedly predicted, reducing the prediction error at each step. This architecture was state of the art when first developed and remains a powerful architecture.

For baseline results, we test both a ResNet trained from scratch on the target dataset as well as a ResNet pre-trained on the ImageNet image classification dataset. The latter receives a new set of fully connected layers to train from scratch, but every other layer remains from the pre-trained version to take advantage of learned features from the much larger dataset.

4.2. Proposed Methods

As no climbing route dataset existed before this project, and the ability to create such a dataset was limited by the fact that no gym has sufficient numbers of routes, the target dataset is extremely small. Even with augmentation, the training set has just over 300 images. This would already pose a significant challenge in training a CNN without overfitting. Complicating this already challenging task, climbing routes mostly only depends on the holds of the route; the wall and anything above, below, or to the side of the route has very little to do with the difficulty of the route. Therefore, a large amount of each image is irrelevant to the target task and allows for overfitting. As the relevant features are quite specific, it is expected that training a model from scratch would require a significant amount of training data to achieve a low prediction error.

To address these challenges of the low data scenario, we choose to use self-supervised learning on an adjacent dataset. Related work has shown the extreme efficacy of self-supervised learning in pre-training a model on unlabeled data.

Self-supervised learning is a technique in which a larger unlabeled dataset is leveraged to create a labeled dataset to allow the model to learn useful representations. For example, one common self-supervised learning method is predicting the correct orientation of each image in the unlabeled dataset. Such a model becomes a four-class (0°, 90°, 180°, 270°) classifier.

While this technique is useful for learning representations on large datasets, the climbing hold dataset is still small on the scale of image datasets. Traditional self-supervised learning may not be effective due to the size of the dataset. Further, as climbing holds can be placed in any way, the orientation of the image will be primarily determined by the floor, ceiling, edge of the wall, or ropes, none of which are useful for route grading. Therefore even if self-supervised learning can learn good representations for the rotation task, they likely won't be useful for the grading task.

While other traditional self-supervised learning objectives exist, like filling in a hole, recoloring, or ordering a jigsaw, no objectives force the model to learn any representations of the holds. As the holds can be placed anywhere in any orientation, any truly unlabeled dataset will not be suitable for learning representations that help with grading routes.

Therefore, we propose a form of related task self-supervised learning. This technique shares similarities with transfer learning. In particular, we propose training a CNN on a related dataset using useful labels that will help learn good representations. As we have a dataset of object detection bounding boxes on climbing holds, which are an important part of understanding a route, we pre-train our model on the object detection task. This is an application of self-supervised learning but with a task with proper labels. It is similar to transfer learning in that a model trained for one task is used for another, except that the final layers will be replaced and retrained for the new task.

We hypothesize that this approach will be effective due to past results indicating that self-supervised learning is very effective, as well as research showing that such learning is more effective when the task is hard [8] (as hold detection is in comparison to image rotation) and the dataset and task are related to the final dataset [18].

As the climbing hold dataset has object detection labels, a YOLOv5 model was used as the network for this algorithm. YOLO is a family of single-shot object detection pipelines [13]. To only use a single pass over the input image, YOLO uses a grid bounding box selection instead of a proposal network. Within each grid cell, the network identifies every object whose center lies within the grid cell. The model then proposes several bounding boxes that may contain objects. Each grid box has a predicted confidence score, which is used to select boxes that are likely to contain objects. The coordinates of the boxes are then adjusted with a regression head. Finally, for each adjusted bounding box class probabilities are predicted. For inference, the model selects the bounding boxes with confidence scores over a certain threshold, adjusts the bounding box with the regression head, and then chooses the class with the highest conditional probability. The result is a list of bounding boxes with confidence scores and class predictions. For training purposes, each head contributes to the loss, producing the following loss function

$$\begin{aligned}\ell =& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\ &+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ &+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{\text{c} \in \text{ classes}} (p_i(c) - \hat{p}_i(c))^2\end{aligned}$$

Importantly for this application, YOLO networks use a single CNN backbone that feeds into each of the different heads. This is useful for this application as it means that the CNN in the YOLO model must learn all representations necessary for all the sub-tasks in the object detection task. That means that the CNN must be able to distinguish generally where objects on the wall are and also their exact boundaries and identities. This algorithm is thus constructed to take maximal advantage of the self-supervised learning that occurs on the hold detection dataset.

The YOLOv5 model used has pre-trained weights, allowing the algorithm to take advantage of the stacking benefits of pre-training and self-supervised learning.[20] Those weights were first fine-tuned on the climbing hold dataset to learn more accurate representations of climbing holds. Then the various heads of the YOLO model were removed, and a convolutional layer was added to reduce the number of channels, then finally a fully connected layer was added to produce an appropriate regression head for the grading task. This regression head uses a mean squared error loss like the baseline model, though it updates only two layers' worth of weights, freezing the rest.

In summary, the model begins as a YOLOv5 object detection model, pre-trained on the large COCO dataset. This dataset contains nothing about climbing but teaches the model many useful representations of objects and shapes. Then the architecture is preserved but the weights are updated when the model is tuned on the hold detection dataset. While this doesn't directly teach anything about the target task, it teaches the CNN component of the YOLO model useful representations that allow it to identify climbing holds, especially their edges. Finally, the weights of the CNN are frozen, having learned the best representations they can, but the regression, confidence, and class heads are removed from the model, and replaced by convolutional and fully connected layers, which are tuned on the target dataset. While the target dataset isn't large enough to train a full model, it may be able to take advantage of previously learned representations and learn an effective convolutional and fully connected layer, creating a more effective regression head.

5. Experiments

All of the approaches in this paper ultimately trained directly on the target dataset using the mean squared error (MSE) loss. As this metric is relatively divorced from understanding the performance of the various models, some benchmarks need to be set. The first benchmark is random guessing. A model that only guesses the sample mean grade achieves an MSE of 4.40 on the test set. Any model that claims to hold predictive power must beat that benchmark.

The second benchmark is human performance on the task. When allowed to climb the route, a human will generally never be off by more than one grade, meaning the MSE should be less than 1. However, in cases where an image is the only thing provided, an error of two grades is plausible, though anything worse would be quite unlikely. In such a case, the mean squared error of human performance would likely be between 1 and 2.

5.1. Baseline

Two baseline models were trained directly on the target dataset without using the climbing holds data at all. Both models had their hyper-parameters lightly tuned on the validation set. Both models were trained for 5 epochs with an Adam optimizer with a learning rate of 0.001. A batch size of 4 was used.

The ResNet network trained from scratch achieved a test MSE of 4.87, while the ResNet network pre-trained on ImageNet achieved a test MSE of 5.23. This is significantly below even the benchmark of guessing the sample mean, indicating that these models are inaccurate, as seen in Figure 4, which shows predictions for the pre-trained models.

These results demonstrate that the extremely small dataset didn't contain enough examples to learn anything meaningful. As the test MSEs were not below that obtained by guessing, the models contained no meaningful predictive power. This is not a large surprise, as the dataset is so small, and the baseline models are not specialized for the task. These results justify the need for an improved model, one that uses another source of data in order to learn meaningful representations

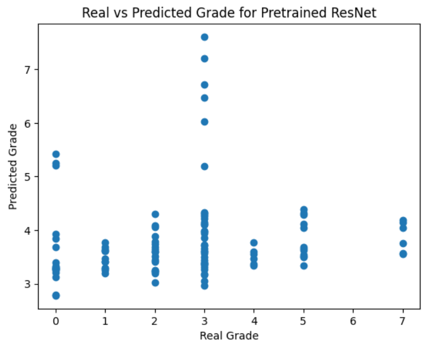

Figure 4. A graph demonstrating the very weak relationship between real grade and predicted grade when using the pre-trained ResNet model with no self-supervised learning.

5.2. Proposed Method

The proposed method had a first step, supervised learning on the climbing holds dataset. This object detection task, unlike the grading task, used the metric mAP50 (mean Average Precision). This metric measures the value of precision for each recall value from 0 to 1. The YOLOv5 pretrained model was trained for 10 epochs with an Adam optimizer with a learning rate of 0.01 and a batch size of 16. These hyper-parameters were tuned on the validation set. After training, the YOLOv5 model achieved a 0.78 mAP50 score, indicating a relatively strong ability to detect climbing holds. A visual examination confirmed, as shown in Figure 5, that the detection is consistent and accurate.

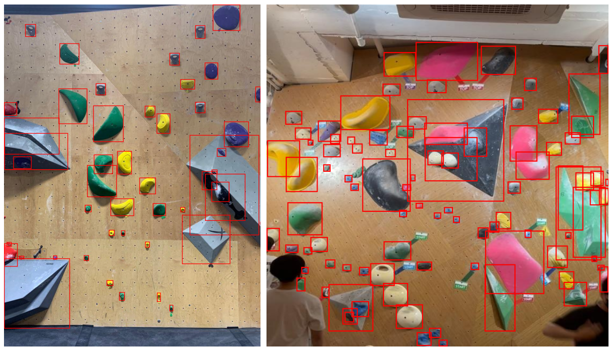

Figure 5. Two example images of climbing hold detection by the pre-trained YOLOv5 model.

The pre-trained YOLO model with self-supervised learning was compared with another pre-trained YOLO model that was trained on the target dataset without self-supervised learning. Both models had their CNN backbones used, with the various heads discarded and replaced by regression heads. These models were trained for 2 epochs with an Adam optimizer with a learning rate of 0.0001 and a batch size of 1. These hyper-parameters were tuned on the validation set and displayed significant effects from small changes.

The model that didn't undergo pre-training on the climbing hold dataset achieved a test MSE of 2.05, while the one that did reach a test MSE of just 1.47. These test errors are much lower than random chance would allow for and comparable to a human level. As seen in Figure 6, there is a strong correlation between the actual grade and the predicted grade. The general trend is for the predictions to be closer to the 'center' of the common grades (3-4), which makes sense as the MSE metric punishes extreme predictions far more than conservative ones.

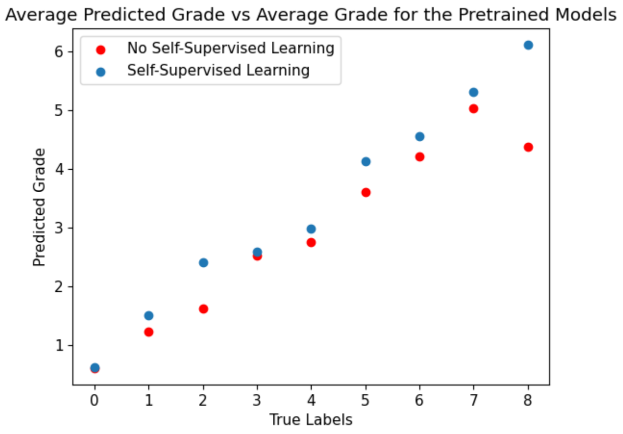

Figure 6. A plot of the average predicted grade for each ground truth grade for both YOLO models

Looking at examples of very wrong predictions reveals more about the nature of the model. The most misclassified example in the test set was the route depicted in purple in Figure 7. This route is a V7, but the proposed network predicted a grade of 2.1. This is a reasonable prediction. The holds appear large and close together, hallmarks of an easier route. The route is extremely difficult because the holds barely protrude from the wall, giving little to work with, a fact that is not clear from the image. This mistake is not only one that a human would make but also demonstrates that our network understands some important aspects of grading a route.

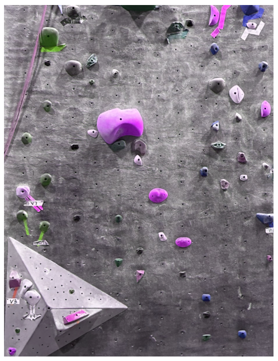

Figure 7. An example route (in purple) that was misgraded

5.3. Discussion

Model Architecture	Pre-Trained	SSL	Test MSE
ResNet	No	No	4.87
ResNet	Yes	No	5.23
YOLOv5	Yes	No	2.05
YOLOv5	**Yes**	**Yes**	**1.47**

Table 1. Table of test MSEs of the models tested in this paper

The ResNet architecture failed to do better than predicting the sample mean, even when pre-trained. This is likely due to the small size of the target dataset. Even when allowed to train for longer, these models simply were unable to form good predictors for this dataset.

The YOLOv5 architecture surprisingly was quite effective even without the additional learning from the climbing hold dataset. This major jump in predictive power over the ResNet may be due to the YOLO model's additional heads directing focus to the edges of objects, rather than the image as a whole. This is much more suited for a task like grading a route because, even though it requires information from the entire image, grading a route requires knowledge of subtle details about each individual hold, and needs very little else.

This model, even without any domain-specific pre-training, was able to reach the level of perhaps a beginner climber. While mistakes are still very common, most predictions understand the broad strokes of the route, and very few predictions are off.

As hypothesized, and consistent with previous research that both pre-training and self-supervised learning boost model performance, the YOLO model utilizing both techniques was the most successful. This aligned with the reviewed literature suggesting that a hard self-supervised learning task in an adjacent domain would be the most effective training method. This aligns with the understanding of CNNs. The pre-training forces the model to learn to identify a climbing hold, especially its edges. This is a necessary skill for route grading, and one that on its own provides a significant advantage.

This model displays a high level of sophistication and can grade routes at a similar level to that of perhaps an intermediate climber. The most significant mistake it made was understandable to a human and reflected its understanding of what makes a route difficult or easy. This understanding also aligns with the pre-training, where the model focuses on the holds it learned to detect, and makes its decisions based on those holds.

6. Conclusion

As hypothesized and supported by previous research, the most effective model was one that utilized both pre-training on unrelated datasets as well as self-supervised learning on a related but distinct task on an adjacent dataset. This model was quite effective, surpassing a beginning human level with just 52 training examples, due to the boost from the other learning techniques. This result indicates that significant results can be achieved even on small datasets, so long as there is another method to learn important representations.

Given more time and resources, we would test the effect of the size of the target dataset on the test error to determine whether there are any thresholds for dataset size. Further research could also investigate whether these techniques can be applied to other problems with limited data. If so, this could be a promising avenue of research for fields in which labeling data is expensive or otherwise difficult.

7. Contributions and Acknowledgements

Jack collected, processed, and hand-labeled the route grading dataset, trained the YOLO model on the hold identification dataset, trained the baseline models on the target dataset, wrote the code to modify YOLO models into regression models, and wrote the paper.

Ethan trained and fine-tuned the hyper-parameters of the proposed models on the target dataset, visualized the results, and wrote the paper.

We used the GitHub library Ultralytics (`https://github.com/ultralytics/ultralytics`), an implementation of the YOLO model family into PyTorch.

References

[1] Blackcreed. Climbing holds and volumes dataset. https://universe.roboflow.com/blackcreed-xpgxh/climbing-holds-and-volumes, February 2024.

[2] S. Bucci, A. D'Innocente, Y. Liao, F. M. Carlucci, B. Caputo, and T. Tommasi. Self-supervised learning across domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5516–5528, 2021.

[3] A. Clark. Pillow (pil fork) documentation, 2015.

[4] N. Draper. 14 climbing grades. *The Science of Climbing and Mountaineering*, page 227, 2016.

[5] A. El-Nouby, G. Izacard, H. Touvron, I. Laptev, H. Jegou, and E. Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

[6] L. Ericsson, H. Gouk, and T. M. Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[7] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[8] P. Goyal, D. Mahajan, A. Gupta, and I. Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the ieee/cvf International Conference on computer vision*, pages 6391–6400, 2019.

[9] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 902–909, 2010.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[13] G. Jocher. YOLOv5 by Ultralytics.

[14] T. maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[15] A. Newell and J. Deng. How useful is self-supervised pretraining for visual tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7354, 2020.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[17] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[18] J. Xu, L. Xiao, and A. M. López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7:156694–156706, 2019.

[19] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. S4l: Self-supervised semi-supervised learning, 2019.

[20] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845, 2020.