COMP90086 Computer Vision Project Report

Thomas Choi
1202247

Angela Yifei Yuan
1269549

I. Introduction

This project targets a visual physical reasoning task, predicting the stability of a stack of blocks from a single image. We use a subset of the ShapeStacks dataset [1], featuring synthetic scenes produced by a 3D rendering software. Each scene displays a vertical stack of 2 to 6 blocks, with the blocks varying in 6 possible colors and 5 possible shapes. The task is to develop a system to predict the stable height of the stack, defined as the number of consecutive blocks starting from the bottom that satisfies two criteria: (1) planar surface criterion, where the block must be placed on a flat surface, and (2) centre of mass criterion, such that its centre of mass must be supported by the surface below. The violations of the two criteria are denoted as VPSF and VCOM respectively. Some images contain a fully stable stack, where the stable height is equivalent to the stack height. In other images, there is exactly one block placement that violates the rules. The bottom block of the stack is always stable, so the stable height across all scenes ranges from 1 to 6.

We propose a detection pipeline capable of predicting the stable height of a stack with no block limit, and demonstrate empirically that it outperforms a simple direct stable height prediction pipeline when using same backbone architectures.

II. Methodology

A. Classification Pipeline

1) Direct Stable Height Prediction (P_{direct}): P_{direct} is a single-staged pipeline where the model is trained to directly predict the stable height of a block stack in an image. Given 6 possible stable heights, the classification layer of the neural network (NN) model requires 6 neurons with softmax activation function to perform multi-class classification. This is a baseline method selected for its simplicity and time efficiency. However, the need to retrain the model for larger stack heights limits its scalability and generalizability.

2) Iterative Stability Prediction (P_{iter}): P_{iter} utilizes a binary stack-stability classifier to iteratively predict whether a given image x_i contains a fully stable block stack. If the stack is classified as unstable, the top block is removed, and the stability check is repeated. This process continues until the stack is predicted to be stable or only one block remains, and the remaining stack height is returned as the predicted stable height. Figure 1 illustrates the prediction process.

In our pipeline implementation, the first iteration involves preprocessing steps including background removal, and color-based instance segmentation to identify the number and positions of all blocks. The information obtained from segmentation is then used in all iterations to remove the top block by replacing corresponding pixel values with zero, and to assess the remaining stack height. Object centralization is also performed during each iteration. Details for the preprocessing steps can be found in Section II-B.

This pipeline was inspired by [1], in which a new stack was built by iteratively stacking objects at the position with the highest stability score evaluated by a binary stability classifier. Rather than predicting the stable height directly, this pipeline reduces the workload of the classifier to binary classification, and improves the interpretability of the classifier. However, relying upon 2 systems (instance segmentation and binary stability classifier) to perform a single task implies that we have to ensure each system is optimised in order to maximise the overall performance in predicting the stable height.

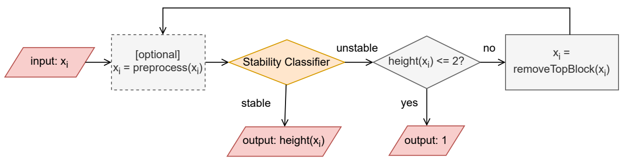

Fig. 1. Prediction process of P_{iter}, where x_i is a block stack image. The optional preprocess step can be customised and may differ across iterations.

B. Image Preprocessing

1) Background Removal: For stack stability classification, background refers to the non-stack pixels that do not contribute generalizable information toward the prediction. The rembg package [2] was used for background removal, which replaced these pixels with zero values to prevent model from over-fitting to the background environment, and help the detection focus on the block stack only. A common failure occurred with the scene depicted in Figure 2, where the striated floor posed a challenge for background removal.

2) Object Centralization: After background removal, the block stack pixels with non-zero values are translated to the centre of the image. This reduces the variability in the object's location, and aims to help the model learn more efficiently as it needs to account less for position shifts.

3) Image Augmentation: Image augmentation generates variations of the original images. By increasing diversity in the training dataset, we aim to improve the generalizability and robustness of the models. A key criterion in selecting data augmentation technique is ensuring that the resulting images preserve the correct orientation and structure of block stacks.

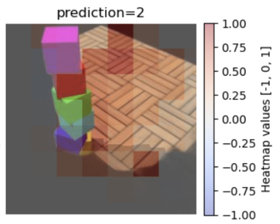

Fig. 2. When background is not fully removed, an occlusion test reveals that the model attended to the background in predicting the stable height.

During training, we apply several online data augmentation techniques, including horizontal flip, zooming in and out within a factor of 0.1, and slight rotations within 10 degrees.

4) Instance Segmentation: This step only applies to P_{iter} to support the progressive removal of blocks from the top.

Since blocks are only in 6 defined colors, we leverage this characteristic and extract objects by the 6 defined colors. Concretely, the background-removed image is converted into the HSV space to compute 6 color masks, each of which corresponds to a potential block and includes pixels within a specific hue range to account for the variations under different lighting and near edges. Within each mask, pixels with intensities higher than the threshold are retained (valid pixels) to filter out background noises. In order to reduce false positives, the largest contour of valid pixels is computed using OpenCV and color masks with an overly small contour size or an insufficient number of valid pixels are discarded. By grid searching over the hue range, intensity threshold and valid pixel count threshold, the extracted number of blocks (number of valid color masks) using this method achieved an accuracy of 0.84 in terms of the true stack height in the training set. Based on the position of every block, which is determined by the lowest position of the largest contour of valid pixels, they are re-stacked together for the classifier input and we can also remove blocks from the top to generate new stability samples.

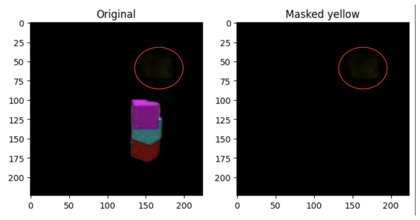

Fig. 3. An example of a false positive in block segmentation.

However, despite the constraints based on color, intensity and contour to detect valid objects, this traditional way to extract objects by colors is not robust to dark pixel noise in the background and false positives still occur (one example is in Figure 3), resulting in a total height higher than the ground truth. Moreover, we realised the errors in background removal that would miss some existing objects might propagate to object segmentation, hindering the segmentation result inputted to the classifier. In light of this, we have also tried to execute object segmentation with the entire background. However, the accuracy dropped significantly, which highlights the need to de-noise the background by removal.

C. Pre-trained Backbone

To leverage transfer leaning, we use models pre-trained on ImageNet, as they have already learned rich feature representations from a large and diverse dataset, which can often help accelerate training and improve performance of downstream tasks. A consistent approach is taken to fine-tune different pre-trained models: after the last convolutional layer in the pre-trained backbones, we added a global average pooling layer, followed by a fully-connected layer and a dropout layer, before the final classification layer.

We experimented with three backbone architectures: InceptionV3, ResNet, and Swin Transformer.

1) InceptionV3: InceptionV3 [3] uses multiple convolutional filters of different sizes in parallel, which enables the model to capture features at various scales. This is useful for our task as the images are taken from different distances to the block stack and from different camera angles, such that the blocks appear as different sizes in different scenarios. Additionally, Inception model was previously applied in the task of binary stability prediction [1] and achieved performance above 77% accuracy on the ShapeStack dataset.

2) ResNet: ResNet [4] is a deep convolutional NN (CNN), selected as it effectively addresses challenges associated with training very deep networks. It introduces the concept of residual learning through shortcut connections that bypass one or more layers, which enables the network to learn identity mappings and helps mitigate the vanishing gradient problem.

3) Swin Transformer: More recently, transformer models have been used in image classification and achieve state-of-the-art performance. These models process images by splitting them into patches and applying self-attention mechanisms to capture long-range dependencies between patches, rather than relying on the localized convolutional filters used in CNNs. We selected Swin Transformer [5] which employs shifted windows for efficient local and global representation learning. By restricting self-attention to local windows, it addresses the computational complexity when applying transformer models to visual classification tasks with large pixel counts. Additionally, by shifting the window partitioning in each layer, allowing windows to cross the boundaries of local windows in previous layer, it establishes connections between windows and facilitates global feature learning.

III. Experimentation

This section outlines the experimentation process and details the methods employed during model training.

Our experimentation follows an iterative cycle alternating between method implementation, evaluation, and subsequent enhancement. To support this process, we divide the training dataset into 2 parts: 80% for training, and 20% for validation, where model performance is assessed on the validation set.

The experimentation steps mainly focused on three aspects, including backbone model selection, image pre-processing and augmentation, and pipeline optimization.

1) We began by implementing P_{direct}, experimenting with three backbone architectures described in Section II-C. We conducted hyperparameter tuning, mainly focusing on the dropout ratio (0.3), number of neurons in the hidden fully-connected layer (256), number of layers to be retrained (all), and learning rate (0.00001). Based on experimental results, and due to the substantial difference between our task and the ImageNet object classification task, we fine-tune the entire backbone model for better performance. We employ early stopping to terminate model training when the validation accuracy does not improve for two epochs, and the model with the highest validation accuracy is restored and evaluated.
2) To address class imbalance, we employed random upsampling to ensure that all classes had equal representation. This is combined with online image augmentation to increase training dataset diversity and to prevent over-fitting caused by repeated samples. We also attempted alternative approach such as random downsampling and reduced the number of samples in larger classes to match the minority class size, but this approach yielded comparatively lower performance.
3) Upon conducting error analysis via occlusion test, we observed that the model frequently focus on background pixels during predictions (Figure 2). Background removal is applied to prevent the model from over-fitting to environmental cues that would not generalize to other samples. This step also facilitated object centralization, as the background was replaced with zero values, simplifying the positioning of the target objects.
4) We experimented with a stacking algorithm ($Stack_{aug}$) to aggregate the stable height predictions of the original image and augmented variations via soft voting. Each image undergoes four offline augmentations: horizontal flip, color randomisation, varying aspect-ratios, and minimal rotation. This process improved the accuracy of certain backbone models within the P_{direct} pipeline. However, it was not applied to the later P_{iter} pipeline for stacking the binary stability predictions, as it introduces additional computational complexity and increases the inference time by a factor of K, where K represents the number of augmentation techniques used. This is undesirable for P_{iter} which needs to repeat the detection process for multiple iterations.
5) We designed and implemented P_{iter}, motivated by the hypothesis that a binary stability classification task would be simpler than multi-class stable height prediction. Experiments are conducted using InceptionV3 and ResNet considering tradeoffs in performance and time efficiency. During inference, the classifier is provided with images where the top blocks are iteratively removed. To support this process, modifications to the training and validation sets are necessary so that the model becomes familiar with these perturbed images.
   After oversampling by stable height, each original sample, preprocessed through background removal and block segmentation, is assigned a binary stability label. The label depends on whether the number of blocks is less than or equal to the true stable height, and the sample is added to the new training or validation dataset. The process continues iteratively: if the current image is unstable, the top block is removed and the resulting image is added to the dataset with an updated stability label. The process terminates if the current image is stable, simulating the inference process and balancing the training and validation datasets in terms of stability. Online augmentation and object centralization continued to be executed before feeding the images into the classifier to improve generalizability and robustness.

IV. Evaluation

A. Result and Discussion

Validation set and the partial Kaggle test set accuracies of our proposed methods are reported in Table I. We report the best performance of different backbone models and classification pipelines from our experiments after steps 4 and 5.

When applying the same P_{direct} pipeline, the Swin Transformer model outperforms both CNN models. Nevertheless, the computation cost of the Swin Transformer is overwhelming as it often takes 1.5 hours to train for 1 epoch. Considering the trade-off between efficiency and performance, we decided to continue with ResNet and InceptionV3 for P_{iter}.

Our proposed method, P_{iter}, trained on a carefully constructed dataset, demonstrates improved performance over P_{direct} for both backbone models. This supports our hypothesis that binary stability classification is a simpler task than multi-class stable height prediction. A possible reason is that the binary stability detection requires the model to focus on smaller local patches, whereas stable height prediction necessitates a more global understanding of the image. Consequently, the CNN models using localised convolutional filters exhibit enhanced performance.

TABLE I
Performance comparison of different methods. Models with an asterik (*) uses $Stack_{aug}$ for performance enhancement.

Pipeline	Backbone Model	Val Accuracy	Test Accuracy
P_{direct}	InceptionV3*	0.4609	0.4750
	ResNet*	0.4766	0.4938
	Swin Transformer	0.5352	0.5313
P_{iter}	InceptionV3	0.5143	0.5177
	ResNet	0.5176	0.5302

B. Error Analysis and Future Improvement

Considering the promising result of P_{iter} and balancing between performance and efficiency, we proposed P_{iter} with ResNet to be our solution and are going to analyse the errors of it. First of all, as shown in Figure 4, the stability classifier tends to underestimate stability and demonstrates a lower recall in stable stacks. The pessimism is also propagated to the overall stable height prediction by the pipeline. In Figure 5, higher stable height classes generally obtain lower recalls compared

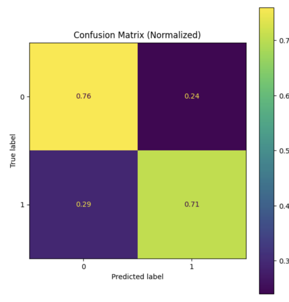

Fig. 4. Normalized confusion matrix of the stability classifier in P_{iter}. 0: unstable; 1: stable.

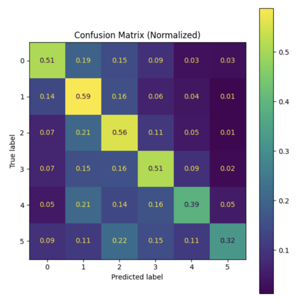

Fig. 5. Normalized confusion matrix of P_{iter}. Stable height number is shifted by -1.

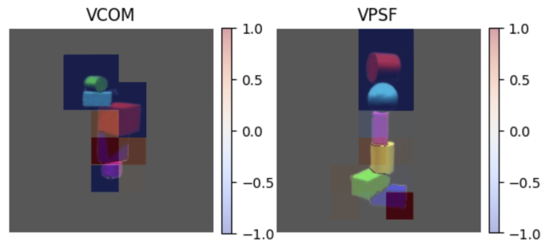

Fig. 6. Occlusion test of the stability classifier by instability type.

to lower stable height classes, reaching the lowest recall of 0.32 in class 6. This suggests that the stability classifier seemed to be more pessimistic about the stability of taller stacks due to the lack of tall stable stacks in training; during the construction of the training set, for unstable stacks, we iteratively removed the top block and added it to training until it is stable, introducing stability bias to lower stacks. A possible approach to address this is balancing training samples across stack heights within each of the 2 stability classes by oversampling. However, given the limited number of tall stable stacks, the diversity in the upsampled training set of the minorities is questionable.

To understand where the stability classifier focuses when determining instability, an occlusion test is performed for each type of instability. As seen from Figure 6, red indicates a decrease in stability score and blue indicates an increase in stability score if the patch is occluded by black pixels. In predicting VPSF, the classifier successfully attends to the violating areas (the tale sphere and red cylinder); when we occlude them, the stability increases significantly. Similarly in determining VCOM, however, the result is more noisy and the classifier is more uncertain about its attention. This result seems to suggest that after removing uninformative variability by background removal and centralization, the stability classifier can focus on critical areas to predict stability, particularly for VPSF.

To further investigate the strengths and weaknesses of P_{iter}, we leverage the provided training metadata and examine its performance under different scene settings. From Table II, the pipeline exhibits lower performance in scenes with a high camera angle and scenes categorised as hard to predict stability.

TABLE II
STABLE HEIGHT PREDICTION ACCURACY OF P_{iter} BY PROPERTIES.

Property	Property Value	Val Accuracy
shapeset	cubes only	0.4512
	all	0.5508
type	easy	0.6059
	hard	0.4342
cam angle	low	0.5419
	high	0.4462
instability type	stable	0.5013
	VCOM	0.3810
	VPSF	0.8283

Under a high camera angle, lower blocks are often occluded by higher blocks and the block segmentation may overlook small existing blocks due to the valid pixels constraint. Furthermore, determining the relative position of the blocks is hard given the similar lowest position of the contours, leading to block disarrangement when it attempts to re-stack the blocks. This difficulty in finding the spatial relationships among the blocks also applies to the ResNet classifier. To improve this, the depth map of the stack can be estimated using advanced pretrained models and fed into the classifier to provide more information on the spatial positions of the blocks.

On the other hand, P_{iter} demonstrates an outstanding stable height prediction performance in VPSF samples. This reveals that rounding surfaces, represented by gradual changes in shadow and curved contours in images, provide informative signals for stability prediction and the model can effectively detect VPSF by attending to spheres or cylinders. This strength is further demonstrated by the increased accuracy when evaluating all types of shapes, as opposed to only evaluating samples containing cubes, which aligns with the results in [1]. However, the classifier cannot effectively classify VCOM samples, revealing the failure to understand the relative position of the blocks in stack stability and learn physical intuition in the centre of mass of the stack due to the large variations in camera angles and block positions. This performance difference between VCOM and VPSF samples is supported by the occlusion test results above. A possible improvement would be providing auxiliary signals, such as the relative positions and centres of mass of the blocks, to the classifier to assist with stability prediction in addition to the image.

V. CONCLUSION

In conclusion, we have introduced the iterative stability prediction pipeline to predict the stable height of a stack, which is not only generalizable to any stack height but also outperforms the direct stable height classification strategy using the same backbone model.

In future studies, one with greater computational resources may consider adopting the Swin Transformer in P_{iter} to possibly improve the stability classification and thus, the overall stable height prediction. The block segmentation can also be enhanced by applying a more sophisticated instance segmentation technique such as pretrained deep networks to optimise the performance of every stage in the pipeline.

REFERENCES

[1] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi, “Shapestacks: Learning vision-based physical intuition for generalised object stacking,” in *Proceedings of the european conference on computer vision (eccv)*, pp. 702–717, 2018.

[2] D. Gatis, “Rembg,” Oct 2022.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[4] 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*, pp. 770–778, 2016.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.