Physical Reasoning: Predicting the Stability of Block Towers with Computer Vision Methods

Laradell Tria
Student ID 1417478

Gia Han Ly
Student ID 1074109

I. Introduction

As computer vision systems are increasingly being adopted into tasks that require maneuvering around physical spaces, physical reasoning and intuition are fundamental to facilitating machines' understanding of object relations. [1]. One key challenge in physical reasoning is to predict the stability from a visual input. This will involve making accurate judgments and decisions about the balance and supports of the objects found in a scene. The problem of stability prediction is inherently challenging due to the diverse appearances of the objects found in the images, like the camera angle, lighting, and the laws of physics (gravity, inertia) that can govern the object's interaction [2]. Therefore, this task becomes a useful benchmark for testing the physical reasoning capabilities of deep learning neural network models in learning physical intuition from the visual dataset alone [3]. Previous work in stability prediction often utilised physics engines in training model predict [4], hence the goal of this paper is to examine and evaluate the machine learning approaches using both computer vision techniques and deep learning models in order to determine the effectiveness in predicting the stability of the block towers based solely on the visual cues on the image.

A. Dataset

The dataset used to train and test the methods is based on ShapeStacks [2], an annotated image dataset depicting stacks of geometric objects. This dataset consists of RGB images of block towers of various heights and shapes, taken from various angles and lighting conditions. Each image represents a vertical stack of blocks, where the task is to determine the stability of the stack and predict the number of blocks that remain stable before the structure becomes unstable and potentially collapses.

B. Task

The task explored in this paper is to classify the stable height of a given block model solely from its RGB images. This classification problem becomes a multi-class task, where the goal is to predict what is the stable height of the stack in each image and provide insights into the physical stability of the structure. This task challenges a model to learn the physical principles like gravity and balance from the visual cues, which requires advanced image processing and machine learning techniques for accurate prediction.

II. Methodology

This section outlines the methods for predicting block tower stability, covering data preprocessing, model selection, and evaluation.

A. Exploratory Data Analysis

An exploratory data analysis is conducted first to examine the current state of the given dataset, which can be seen below:

- Number of images of block towers: **7680**
- Total height of each image: **6**:2304, **5**:1920, **4**:1536, **3**:1152, **2**:768
- Number of classes (Stable Height): **6**
- Classes of each Stable Height: **1**: 1920, **2**: 1920, **3**: 1536, **4**: 1152, **5**: 768, **6**: 38

B. Model Selection

Three deep-learning models were considered for predicting the stability of block towers:

1) ResNet50: ResNet50 is a convolutional neural network model that introduced the concept of residue learning [5]. This helps to elevate the vanishing gradient issue, which allows the network to have deeper layers and therefore capture more features important to detect objects in images. ResNet50 is chosen as one of the base models as it was pre-trained on ImageNet [6], which allows it to generalise well to subsequent downstream tasks such as classifying stable height of blocks tower [5].

2) EfficientNet: EfficientNet [7] proposed the method of compound scaling, which, similar to MobileNet, is designed to reduce the overall computation complexity while preserving model accuracy. EfficientNet employed efficient scaling of model depth, width and resolution; which helps preserve model performance relative to input and computational constraints in a constant way.

3) MobileNet: MobileNet [8] is a convolutional neural network model designed for mobile devices with lower computation resources. MobileNet has a compatible performance with other complex models by utilising the inverted residual, therefore increasing the balance between efficiency and accuracy. Both MobileNetV3Small and MobileNetV3Large are tested in this report, but MobileNetV3Large is chosen as a base for the final architecture to ensure that the model has sufficient complexity and because of its superior result.

C. Image Preprocessing Methods

To improve the quality and relevance of visual features, image preprocessing techniques were applied to the dataset:

1) CLAHE: Contrast Limited Adaptive Histogram Equalization (CLAHE) is a method to improve contrast while limiting noise in an image by first equalising pixel intensities in divided regions and then reducing contrast enhancement in a homogenous area and distributing the contrast to all other areas equally [9].

2) Data Augmentation: The initial exploratory data analysis revealed a limited dataset, making it necessary to apply data augmentation [10] to generate new variations of the images and improve the model's understanding of the data. Augmentation techniques included geometric transformations (flipping, translations, rotations), colour space adjustments, kernel filtering, and feature space augmentations. These transformations help mitigate overfitting by increasing

the diversity of training samples, enabling the model to generalize better despite the small dataset.

3) Depth estimation: Another preprocessing method employed is depth estimation, which is obtained from inference using the Depth Anything V2 model [11]. This preprocessing is done in an attempt to help separate the object of interest from the background to perform instance segmentation, handle object occlusion and boost the model's performance and efficiency in detecting the region of interest.

D. Model Training and Evaluation

The dataset was split into training and validation sets with an 80-20 ratio since test set data was provided in the beginning. Each model was trained using categorical cross-entropy loss, with the stable height of the block towers as the target labels. The models were evaluated using an accuracy metric to determine the effectiveness of the predictions. Early stopping and model checkpoints were also employed during training to optimise performance and avoid overfitting.

III. Experiments

This section includes the description of the experiments conducted to evaluate the performance of different deep learning models and preprocessing techniques for the task of predicting block stack stability. Each experiment follows a structured pipeline that includes image preprocessing, model training, and performance evaluation.

A. Environment Details

To conduct the experiments, three different environments were used to run the procedures implemented for the model. One source is Kaggle notebook with the following specifications: RAM: 29GB, Disk size: 57.6 GB, GPU: T4 GPU, and CPU: 4 x Intel Xeon CPU. Other local environments are a Macbook Air M2 with RAM: 16GB, Disk size: 512GB and a Macbook Pro M1 with RAM: 16GB and Disk size: 1TB

B. Experimental Designs

1) Experimental Setup: Experiments were carried out using the ShapeStacks dataset as described in the previous section where it's split into training (80%) and validation (20%). Each model (ResNet50, EfficientNetB0, MobileNet) was trained on the same dataset split (random_state=42), allowing for a fair comparison across architectures. After that, the image processing techniques were applied consistently to the input data during the preprocessing stage. All models were trained for 50 epochs with early stopping (patience=15) and model checkpoint to prevent overfitting. The models were implemented using Python and TensorFlow, with training performed on the environment described above for efficient computation.

2) Baseline Experiment: The first experiment involved the pre-trained models without any model configuration or image preprocessing techniques. This baseline model served as a benchmark for comparison for the next experiments.

3) Model Configuration: Once the trained model was chosen, the best model was fine-tuned and improved to include additional layers for the classification task. Experiments on freezing layers, configuring the weights, and adjusting the training done in the model, were conducted along with the additional layers. The figure 1 below shows the final architecture of the model chosen as the best-performing model.

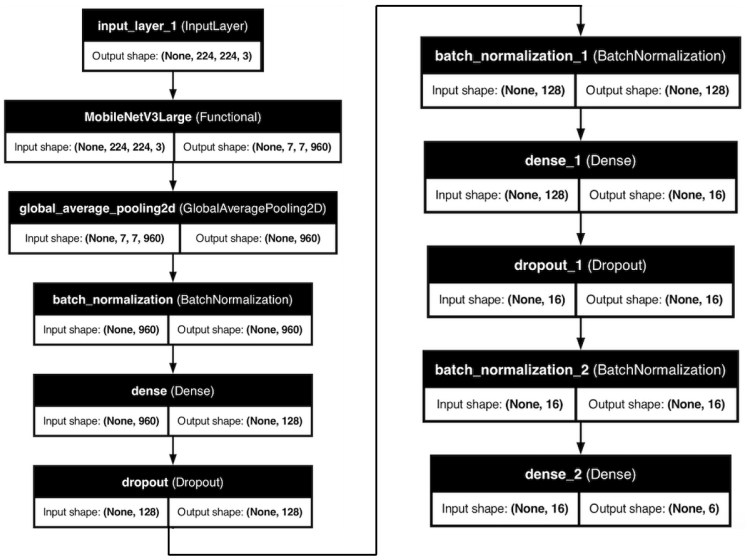

Fig. 1: Model architecture of the final model

4) Image Processing Structure: Subsequent experiments were conducted to evaluate the impact of various image preprocessing techniques which were discussed in the last section. These steps aimed to enhance visual features and improve model performance:

- **Data Augmentation:** To increase data variability and simulate different real-world conditions, augmentation techniques such as rotations, flips, translations and brightness adjustments were applied for this experiment.
- **CLAHE:** CLAHE was applied to the input images to enhance contrast in challenging lighting conditions. The results from the best-performing model chosen with CLAHE-processed images were compared to models without preprocessing.
- **Depth Estimation:** Depth map was computed from the image to segment the object of interest (block tower) from the background before being used as input for the pre-trained models.

C. Evaluation Methods

The evaluation metric used to examine the performance of each model is through a mean accuracy where correct predictions are considered 1 and 0 if incorrect. After the model is evaluated, the total score will be calculated on the test set to assess the model's generalisation to unseen data. To prevent overfitting, early stopping was applied during training, halting the process when the validation loss stopped improving for 15 consecutive epochs. Additionally, the model checkpointing was used to save the best model weights based on the lowest validation loss, ensuring better generalisation to unseen data. Both validation loss and accuracy were key metrics for selecting the final model for evaluation.

D. Hyperparameter Tuning

1) Batch Size: Experiments on batch sizes 16, 32, and 64 are conducted to see how it affects the training time and the convergence behaviour of the model. For each model, the batch size was selected based on the computational efficiency of the model and its performance on the validation set.

2) Optimisers:

a) Adaptive Moment Estimation - Adam: Adam [12] is an optimisation method that has an adaptive learning rate for every parameter, works well with large datasets and has a faster convergence

than other methods of gradient descent. All models were trained initially with Adam optimiser as a benchmark for optimiser tuning.

b) Root Mean Square Propagation - RMSProp: RMSProp [13] is an adaptive learning rate optimiser that was designed to prevent vanishing gradient problem while also offering a stable convergence rate. RMSProp was used to compare with Adam due to its inherent stability.

3) CLAHE: Two parameters were tuned for CLAHE application: clip limit and tile grid size

a) Clip limit: Clip limit refers to the amount of amplification or pixel intensity that CLAHE distributes to other regions of the image.

b) Tile grid size: Tile grid size refers to the number of tiles that CLAHE will divide the image into and apply the intensity redistribution.

4) Other Hyperparameter Tuning: In addition to optimisers, batch size, and CLAHE parameters, the following hyperparameters were also tuned:

- Dropout Rate: Regularisation was implemented using dropout to prevent overfitting, with dropout rates of 0.2 and 0.5 tested. These rates were chosen as they are the standard dropout rates and were appropriate to reduce model overfitting.
- Epochs: After initial experiments, the model was trained for 50 epochs, with early stopping applied if the validation loss plateaued for more than 15 epochs. This avoided unnecessary training and reduced the risk of overfitting.

IV. Results and Discussion

This section presents the results of the experiments conducted for block stability prediction. The performance done is evaluated using the metrics outlined in the experiment section, and the impact of various image processing techniques and hyperparameter tuning are included as well.

A. Comparison of Models

The table I compares the performance results between the 3 pre-trained models.

Pre-trained Models	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Test Acc.
Resnet50	25.45%	1.66	24.87%	1.67	25.63%
EfficientNetB0	43.88%	1.32	44.14%	1.33	25.28%
MobileNetV3Small	35.44%	2.65	34.63%	2.79	36.77%
MobileNetV3Large	**50.76%**	**1.53**	**49.80%**	**1.56**	**50.20%**

TABLE I: Performance of the Base Pre-trained Models

From the table, it is evident that MobileNetV3Large has superior generalisation and accuracy performance for the test set compared to EfficientNet and ResNet50. Despite ResNet50 having deeper layers and complexity compared to MobileNet; and being generally considered to be more accurate, ResNet50 has more trainable parameters and might require more data to properly fine-tune the model. One justification for this is that due to the small size of the training dataset, ResNet50 might have considered certain noises as features, leading to a worse generalisation when compared to the more shallow architecture of MobileNet. As a result, MobileNetV3 Large is chosen as the base model for the overall architecture.

B. Fine-tuning from the Pretrained Model

In this part, the following tables present the result of fine-tuning done on MobileNetV3Large. Table III shows the results on the different number of layers applied to the model, where the appropriate labels can be seen in table II . Table IV shows the results of experimenting with different batch sizes on the model.

Label	Description
6 Layers	Dense=128, Dropout=0.5, 2 BatchNormalizations
9 Layers V1	Dense1=128, Dropout1=0.5, Dense2=16, Dropout2=0.2, 3 BatchNormalizations
9 Layers V2	Dense1=128, Dropout1=0.5, Dense2=32, Dropout1=0.5, 3 BatchNormalizations

TABLE II: Performance of the Model Configuration on MobileNetV3Large

MobileNet Models	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Test Acc.
Freeze 100 Layers	45.42%	2.61	44.01%	2.70	44.27%
Freeze 50 Layers	40.65%	1.66	39.52%	1.73	39.69%
Output Layer Only	42.52%	1.82	43.61%	1.76	45.23%
6 Layers	50.76%	1.53	49.80%	1.56	50.21%
9 Layers V1	**51.22%**	**1.32**	**52.83%**	**1.34**	**52.50%**
9 Layers V2	39.30%	2.06	38.41%	2.03	37.29%

TABLE III: Performance of the Model Configuration on MobileNetV3Large

It can be seen that in Table III, the model with **9 Layers V1** performs the best, achieving the highest validation accuracy (52.83%) and test accuracy (52.50%). This suggests that the small second dense layer and lower dropout rates provide a better balance between complexity and regularisation hence, it contributes to the better generalisation of the predicting block stability task.

Batch Size	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Test Acc.
16	43.27%	1.75	44.08%	1.72	46.35%
32	**51.22%**	**1.32**	**52.83%**	**1.34**	**52.50%**
64	31.10%	1.82	31.45%	1.83	34.90%

TABLE IV: Performance of the Fine-tuning Batch size for the MobileNet

From Table IV, this evaluates the impact of the batch size on MobileNet's performance. Thus, the batch size of **32** yields the best results, where it achieves the highest validation accuracy (52.83%) and test accuracy (52.50%). This size allows the model to update its weight with sufficient variability, which tends to capture essential features in the image that can contribute to accurate predictions of stability in block towers.

C. Data Augmentation

Models	Train acc.	Train loss	Val. acc.	Val. loss.	Test acc.
MobileNetV3Large	29.68%	1.95	29.62%	1.94	30.73%
MobileNetV3Large (w/ Data Augmentation)	**54.02%**	**1.26**	52.15%	1.28	**55.72%**
MobileNetV3Large (w/ Data Augmentation) No flipping and rescaling	52.17%	1.37	**52.28%**	**1.38**	52.50%

TABLE V: Performance of Model Configurations with Data Augmentaion

From table V, it can be observed that having data augmentation helped the model to generalise better, especially since the input dataset is quite small and might not have sufficient samples for each of the classes of stable height. Furthermore, despite data augmentation including flipping and rescaling having a higher training performance, the results of the validation set show that this could lead to overfitting to a certain degree. This suggests that carefully selecting augmentations that better reflect real-world block scenarios is essential for achieving optimal generalisation in block stability prediction.

D. Inclusion of Image Preprocessing

As seen in the table VI, the optimal value of the clip limit is 5 when choosing to apply CLAHE for image preprocessing. Additionally,

image preprocessing using depth estimation with Depth Anything V2 was also experimented with, however, the result suggested that there is a variance in the dataset and differences in class distribution between the train, validation and test set. Nevertheless, the model with no preprocessing step of either CLAHE or depth estimation outperformed in training and testing, which indicates that extra preprocessing is unnecessary and could lead to overfitting, or the model's inability to generalise due to noise being captured in a small dataset.

Methods	Train acc.	Train loss	Val. acc.	Val. loss.	Test acc.
No preprocessing	54.02%	1.26	52.15%	1.28	55.72%
CLAHE Clip Limit = 3 Tile Grid Size = 16,8	52.17%	1.36	52.28%	1.38	54.68%
CLAHE Clip Limit = 5 Tile Grid Size = 16,8	52.37%	1.35	52.41%	1.37	52.60%
CLAHE Clip Limit = 6 Tile Grid Size = 16,8	50.57%	1.69	49.87%	1.71	51.56%
CLAHE Clip Limit = 7 Tile Grid Size = 16,8	29.54%	1.58	30.47%	1.60	-
Depth Estimation (w/ Depth AnythingV2)	33.28%	1.54	31.31%	1.54	51.25%

TABLE VI: Performance of Image Preprocessing Methods (w/ Data Augmentation)

E. Fine-tuning the Built Model w/ Image Preprocessing

In the last part of the experiments, we wanted to fine-tune the model with the inclusion of image preprocessing and how these configurations can impact the results of the model's performance.

Methods	Train acc.	Train loss	Val. acc.	Val. loss.	Test acc.
Adam	52.37%	1.35	52.41%	1.37	52.41%
RMSprop	50.57%	1.69	49.87%	1.70	50.93%

TABLE VII: Performance of the Optimisers applied on the model

The results indicated that while the Adam optimiser achieved a slightly better training accuracy of 52.37% compared to 50.57% for RMSprop, the overall improvements were marginal. This suggests that the choice of optimiser and preprocessing techniques had limited influence on the model's predictive ability. Consequently, the model's inherent limitations, such as struggles with physical reasoning and class imbalance, remain more significant challenges that need to be addressed for substantial performance gains.

F. Error Analysis on the Model

Following the evaluation of the model performance, an error analysis was conducted to better understand the underlying issues that led to suboptimal results in the block stability prediction task. This section explores the key factors contributing to the errors and the areas for potential improvements:

From the results in Table I, the MobileNetV3Large model achieved the best accuracy, with 50.20% on the test set. Despite this being the highest result, several misclassifications were still observed, particularly in the following areas:

- **Complex Tower Structures:** Misclassifications were common in images featuring complex block structures or towers that had small blocks at the bottom and larger blocks on top. Such configurations are physically unstable but could be visually misinterpreted by the models due to a lack of explicit physical rules, such as the centre of gravity or torque. Additionally, having a smaller base block compared to all blocks above can occlude the base block and resulting in the model misclassifying the tower height by 1.
- **Occlusion and Lighting Variations:** Block towers occluded by shadows or other objects in the scene led to incorrect predictions. In some cases, lighting differences between training and test data caused the model to misinterpret the tower's geometry, despite the image processing techniques like CLAHE being applied to correct these variations.
- **Camera Angle Variability:** Camera angle variability significantly affected model performance, with ResNet50 and EfficientNetB0 struggling to generalise across different perspectives, especially oblique angles (pictures from the top view of the block towers), leading to poor stability predictions. MobileNetV3Large, however, demonstrated better robustness to angle changes, aided by its compact design and data augmentation techniques like rotation and flipping, which improved its ability to recognise stability from varying viewpoints. However, recognising tower height from top views remains challenging as the model can not recognise and detect the number of blocks presented.
- **Height-Based Class Imbalance:** The dataset revealed a strong imbalance in class distribution for various stable heights. As observed in the exploratory data analysis, the stable height class "6" had only 38 instances compared to the much higher counts for other classes. This imbalance caused the models to favour predicting the more frequent stable heights (e.g., 1, 2, or 3), leading to lower precision in predicting higher stable classes like 5 or 6.
- **Preprocessing Effectiveness:** The image preprocessing methods like CLAHE and depth estimation improved overall performance but did not fully mitigate the model's confusion, especially in highly complex scenes. CLAHE was effective in enhancing contrast, but it did not address the models' struggle with spatial reasoning about block relationships.

V. Conclusion

From all the findings in this report, it can be concluded that MobileNetV3 Large with data augmentation is a good starting point to aid the model in detecting and understanding the physical interaction of objects in a given scene purely from visual cues. It also shown that MobileNetV3 Large was able to perform well when given a limited size dataset with class imbalance; especially when the dataset contains occluded or noisy objects of interest, the model showed better capability in generalising compared to other more complex models. The results also emphasised the importance of data augmentation, even more so when there is not enough sample for any given class in a dataset; as seen in a large improvement when moving from an original data-only fine-tuned model to a model fine-tuned using augmented data. Although depth estimation and CLAHE enhanced input data, the model still struggled with tasks requiring physical reasoning. In order to improve the performance of MobileNetV3 further, a multi-tasks approach can be explored, as discussed in [2], to train the model on the stability of local regions before classifying the stable height of the overall block tower. Additionally, integration physics-based modelling, targeted data augmentation, and contrastive learning methods to better handle these challenges and improve the block tower stability predictions.

REFERENCES

[1] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[2] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi, "Shapestacks: Learning vision-based physical intuition for generalised object stacking," in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2018, p. 724–739. [Online]. Available: https://doi.org/10.1007/978-3-030-01246-5_43

[3] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example," in *International conference on machine learning*. PMLR, 2016, pp. 430–438.

[4] W. Li, A. Leonardis, and M. Fritz, "Visual stability prediction for robotic manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2606–2613.

[5] 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*, 2016, pp. 770–778.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[7] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[8] S. Qian, C. Ning, and Y. Hu, "Mobilenetv3 for image classification," in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2021, pp. 490–497.

[9] K. Zuiderveld, *Contrast limited adaptive histogram equalization*. USA: Academic Press Professional, Inc., 1994, p. 474–485.

[10] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.

[11] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," *ArXiv*, vol. abs/2406.09414, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270440448

[12] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.