Stack Stability Prediction with Convolution Neural Network and Vision Transformer

Wing Lam Ip
1149602
winglami@student.unimelb.edu.au

Yu-Tse Ling
1466145
yutse@student.unimelb.edu.au

I. Introduction

Recognition of the physical relationship between objects and the environment has always been an emphasis in computer vision research. Real-world applications including automated vehicles, self-flying drones, and facial recognition highlight the importance of object detection and object stacking with technology nowadays as it facilitates the convenience of human living.

This project focuses on a specific aspect of physical reasoning, object stacking, that is discussed in [1]. In the original work, the task is a binary classification problem, where the aim is to determine whether the stack presented in the image is stable or not. This project is an extension to the challenge, where given a stack of blocks, the aim is to determine the stable height of the stack. This means that the implemented algorithm should identify not only whether it is stable but also count the number of blocks presented. In cases where instability is detected, the algorithm should identify the specific block contributing to the failure of stability. This is a challenging task as the implemented system should be able to perform multiple computer vision tasks including object classification, localization, and segmentation. At the same time, the system should understand intuitive physics such as whether the center points of each object align within the range and whether the surfaces of objects are flat or smooth to stabilize the stack.

Therefore, a robust algorithm is required to further understand the scenario presented in the images under diverse conditions. In this project, we aimed to compare the performance of two state-of-the-art deep learning algorithms, namely Convolutional Neural Networks (CNN) and Vision Transform (ViT), on this particular task in order to gain insight into the effectiveness of different deep learning approaches for visual physical reasoning.

II. Relevant Literature

A. CNN

CNN is a type of neural network that is designed primarily for computer vision tasks. The main feature of CNN is the use of convolutional layers that apply kernels to input images and automatically detect important features such as edges, textures, and patterns. Besides the benefit of automatic feature extraction, the use of kernels also allows local connections which significantly reduces the number of model parameters and therefore speeds up convergence[2]. Another important feature is the pooling layers that are applied to reduce the dimensions and allow the model to be more robust and invariant to small transformations in the input image.

Since the proposal of AlexNet and its exceptional performance on image classification tasks[3], researchers in this field have been experimenting with different architecture variations to further utilize the benefit of CNN. For example, Zhang et al.[4] implemented a visual manipulation relationship network based on CNN and showed the effectiveness of CNN on object detection tasks and its potential to capture the relationships between objects in terms of object stacking.

However, despite the exceptional performance on various computer vision tasks, Rangel et al.[5] suggested that there are some drawbacks of CNN, including the need for a large amount of labeled data for training, vulnerability to adversarial examples, and the limitation in learning spatial relationships and orientations.

B. ViT

Transformer is a recently developed deep learning architecture that utilizes the self-attention mechanism to generate a dense representation of the model's input and output[6]. Different from sequential neural networks and CNN, self-attention allows the transformer to learn the global context of a token by attending all positions in the input token sequence at once. An additional learnable classification token is added to the beginning of the sequence and the embedding of it is used for downstream classification tasks. Inspired by the successful use of the transformer model in natural language processing, recent research attempts to explore the possibility of the use of transformers in vision tasks.

ViT[7] adapts the idea by first turning an image into a sequence input by splitting an image into fixed-size patches. The patches are linearly aligned, each with a position embedding that indicates the relative position of the patches. Then the patches are fed into the transformer encoder along with the extra classification token. The original paper showed that after pretraining the ViT on large amounts of data, it managed to outperform state-of-the-art CNN in various image recognition benchmarks with lower computational costs.

Apart from the ability to understand the global context, Tuli et al.[8] found that ViT approaches vision tasks in a more similar way than humans compared to CNN. They found that

ViT is shape-biased, in which ViT processes objects mainly based on shape, while CNN processes objects mainly based on texture. This makes ViT more robust against occlusion, non-salient background regions, and domain shifts[9].

III. Methodology

A. Dataset

The dataset includes a train set with 7,680 images and a test set with 1,920 images. It is the subset of ShapeStack [1] that contains of RGB images of synthetic scenes with a variety of vertical stack from 2 to 6 blocks (shapes include cubes, rectangular solids, spheres, or cylinders) with dimensions of 224 * 224. The data in train set is annotated with integer labels ranging from 1 to 6 that correspond to the stable height. Stable height is the number of blocks that have been placed properly so that the stack remains stable. A properly placed block should meet two criteria: 1) The block should not be placed on top of a curved surface (i.e., a cylinder placed horizontally or a sphere), and 2) The block should align with the center of mass (CoM) criterion mentioned in [1]. For each stack, there is at most one block that violate the criteria.

B. Data preprocessing

TABLE I: Class Distribution on Training Set

Class Label	1	2	3	4	5	6
Count	1920	1920	1536	1152	768	384

Table I displays the class distribution on the training set. The provided training set was first split into 80% of training set and 20% of validation set. To improve the generalizability of the trained model, we augmented the training images by random vertical flipping and randomizing the brightness, contrast, saturation, and hue of the images. The augmentation steps were done to increase the variety of the images but at the same time ensure that the stack follows the physics rule as an upright stack. After that, images in both sets were normalized.

C. Algorithm 1: CNN

For the CNN algorithm, VGG19 [10] that was pre-trained on the ImageNet dataset was used as the base model. VGG19 is a very deep CNN model with 19 weight layers with the use of small-size 3×3 convolution filters. The use of a smaller kernel allows benefits including having the more non-linear variation but keeping the same receptive field as a larger kernel and significantly reducing the number of model parameters. The original paper [10] showed that increasing the depth of the model resulted in better visual representations. And the representation allows achieving the top accuracy on ILSVRC classification and localization tasks during the publish time, and the performance is suggested to be generalizable to different tasks and datasets.

The algorithm is implemented by connecting VGG19 with extra fully connected layers. The preprocessed images were first fed to VGG19. The output of the base model VGG19 was then passed to a flatten layer followed by two fully connected layers with 128 and 64 neurons, both using the ReLU activation function. The final layer is an output layer with 6 neurons with a softmax activation function that outputs the probability distribution across 6 classification labels. The model is configured with 40 epochs, a batch size of 32, and a dropout rate of 0.3. For optimization, the Adam optimizer is used with a learning rate of 2×10^{-5}, and the model's loss is computed using the categorical cross entropy function. Early stopping was applied, in which the model weights from the epoch with the best validation accuracy were saved.

D. Algorithm 2: ViT

For the ViT algorithm, the DINO V2[11] checkpoints that is hosted on Hugging Face [1] was used. DINO V2 is a self-supervised ViT that utilizes the technique of knowledge distillation to train the model. This allow the model to learn transferable image features and generate image embedding that results in state-of-the-art performance on benchmarks in different computer vision tasks such as image classification and semantic segmentation.

The algorithm was implemented by connecting the output image embedding of DINO V2 into an output layer with 6 neurons with a softmax activation function for classification. The model is configured with 20 epochs and a batch size of 32. For optimization, the AdamW optimizer is used with a learning rate of 5×10^{-5}. A weight decay of 0.01 and warm-up were applied to the learning rate for the first 10% of the training steps. The model's loss is computed using the categorical cross-entropy function. Early stopping was applied, in which the model weights from the epoch with the best validation accuracy were saved.

E. Evaluation metrics

The algorithms' performance on the validation set is reported. The overall performances of the algorithms are compared based on accuracy, precision, recall, and f1-score. Due to the class imbalance, macro precision, recall, and f1-score are reported as they give equal weight to each class regardless of the number of instances included in that class. This ensures that the algorithm performance on minority classes is also taken into account and avoid information bias. Also, the f1-score of each class label is reported to provide a more in-depth analysis of the algorithms' performance.

In addition, the train CSV file contains extra metadata columns including 1) whether the stack contains only cubes or multiple shapes, 2) instability type (no instability, unstable due to unsupported center of mass, or unstable due to stacking on the non-planar surface), and 3) camera angle (low or high). Further evaluation of the algorithms is done by comparing the accuracy among each group to see in which scenario the algorithms performed the best and in which scenario they underperformed.

[1]https://huggingface.co/facebook/dinov2-base

IV. Result

A. Overall Result

TABLE II: Overall Result

Algorithm	Accuracy	Precision	Recall	F1-Score
DINO V2	0.61	0.56	0.57	0.56
VGG19	0.53	0.42	0.44	0.43

1 Precision, recall, and F1-Score are macro

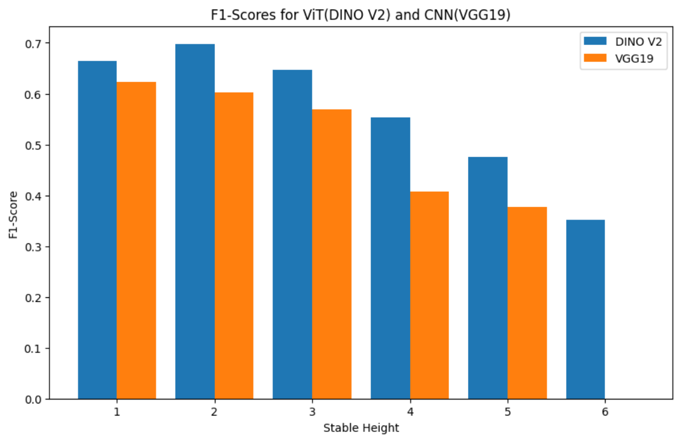

Fig. 1: F1-scores for each class labels

Table II and Fig. 1 compare the overall performance of the two deep learning algorithms DINO V2 (ViT) and VGG19 (CNN). As shown in Table 2, DINO V2 significantly outperforms VGG19 in all evaluation metrics, including accuracy (0.61 vs. 0.53), macro precision (0.56 vs. 0.42), macro recall (0.57 vs. 0.44), and macro F1-score (0.56 vs. 0.43). Figure 1 further breaks down the F1-scores across the 6 class labels. DINO V2 resulted in higher f1-scores in all six classification labels compared to VGG19. Similar trends in F1-scores were found between the two algorithms, in which both algorithms showed higher F1-scores in the majority classes (stable height of 1 to 3), and lower F1-scores in minority classes (stable height of 4 to 6). One noticeable difference is the algorithms' performances on the minority class. For stacks with a stable height is 6, DINO V2 resulted in a 0.35 F1-score. On the other hand, VGG19 failed to classify any of the stacks, resulting in a zero F1-score in this class.

This result showed that VGG19 is weaker in physical reasoning compared to DINO V2 in terms of object stackability. Furthermore, the prediction performance when a stable height is 6 indicates that VGG19 is weaker in counting the object occurrence. This is because of the maximum height of the stack is 6, and having a stable height of 6 means that there is no stackability criteria is volated. Therefore, this suggests a possible limitation of counting object occurrence in VGG19's ability.

B. Shape of Blocks in the Stack

According to Fig. 2, the DINO V2 achieved both higher accuracy in detecting stacks with only cubes and multiple

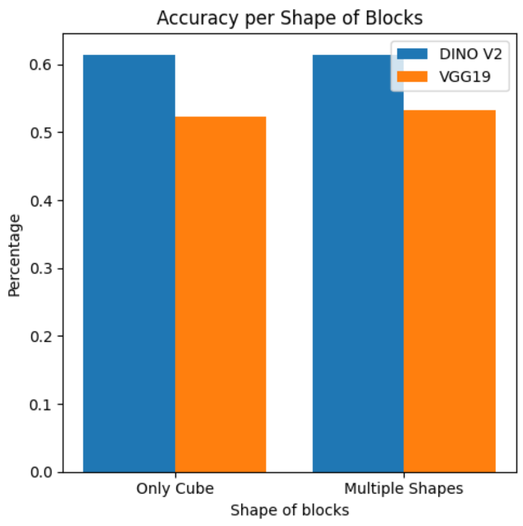

Fig. 2: Accuracy per Shape of Blocks

shapes compared to VGG19. For the detection of only cubes, the DINO V2 and VGG19 have an accuracy of 0.62 and 0.52, respectively. For stacks with different shapes, the DINO V2 and VGG19 have accuracy of 0.63 and 0.52, respectively. The similarity between these two groups suggests that the shape of the blocks does not affect the prediction performance of both models.

C. Instability Type

TABLE III: Instability Type Distribution in Training Set

Instability Type	Count
Stable	1920
Unstable (CoM Violation)	3840
Unstable (Non-planar Surface)	1920

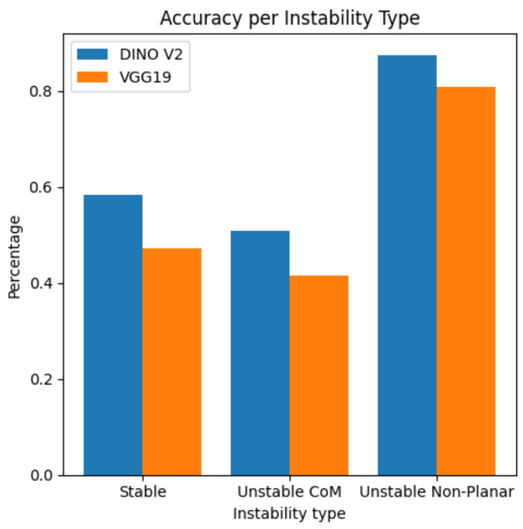

Fig. 3: Accuracy per Instability Type

According to Fig.3, both models performed exceptionally well in predicting the stable height for stacks containing a block that was placed above a non-planar surface, with accuracy of 0.87 and 0.80 for DINO V2 and VGG19, respectively. The model performance on another unstable criterion, violation of CoM, is significantly lower than the non-planar surface criterion, with only accuracy of 0.51 and 0.42 for DINO V2 and VGG 19, respectively. The performance on stable stacks is just slightly higher than the violation of CoM stack, with an accuracy of 0.58 and 0.47 for DINO V2 and VGG 19, respectively. Considering the distribution of instability type presented in Table III, stacks with non-planar surface violation is half of the number of stacks with violation of CoM, yet it is the instability type that the models performed the best. This suggested that the presence of a block on a non-planar surface is a visual feature that is easy to learn by the models. On the other hand, the violation of CoM is a more difficult feature to capture by the model as it requires the model to consider the entire stack and learn the object relationship between the blocks. Similarly for stable stacks, the relatively low accuracy suggested that models need to learn more complicated features in order to identify the stack as stable and output the total height of the stack.

D. Camera Angle

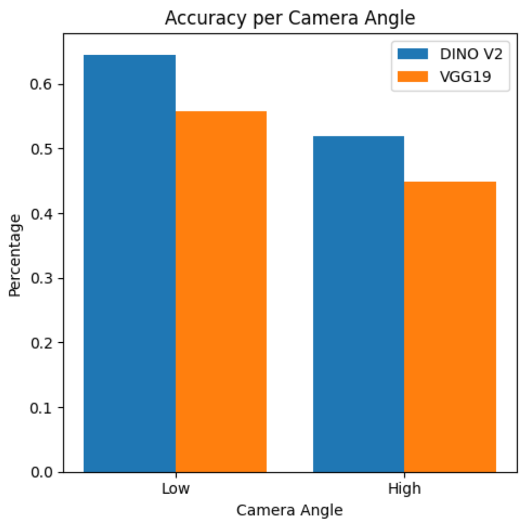

Fig. 4: Accuracy per Camera Angle

Camera angle is an important factor in image display. Shooting from different angles distorts the objects and can sometimes block views and hide the edges of the blocks. The results showed that both VGG19 and DINO V2 have better accuracy in lower camera angle images. This matches our expectations because overlooking objects often changes the normal formation of objects. Additionally, the blocks placed at the top are likely to obstruct or cover up the blocks underneath and therefore make it harder to predict correctly. Similar to the result from previous groups, DINO V2 performs better with 0.65 (low angle) and 0.51 (high angle) compared to 0.56 (low angle) and 0.45 (low angle) for VGG19.

V. Discussion and Conclusion

The experiment results show that DINO V2 outperforms VGG19 in the ShapeStack task as DINO V2 consistently shows higher accuracy compared to VGG19 in all kinds of scenarios presented, and also a higher accuracy and f1-score compared to VGG19. The result suggested that DINO V2 has a better physical reasoning ability specifically in the detection of object stackability compared to VGG19. This result aligns with the recent discussion regarding the performance difference between CNN and ViT [8][9].

There are two possible explanations for the performance differences between the two algorithms. First, the ability to capture the global context may be important for this object stackability identification task. VGG19 is built with multiple convolutional layers and MaxPooling layers inserted in between every 2 or 4 convolutional layers sequentially. This architecture enables the model to capture local spatial features effectively but sometimes struggles to identify long-range dependencies between elements. On the other hand, for DINO V2, the transformer architecture allows the model to capture the global spatial features that are essential to the understanding of the relationship between the blocks in the stacks.

Second, the randomly generated background and the lighting in the image introduce extra texture information to the image that is not relevant to the stackability task. VGG19, as a texture-bias CNN model [8], is more likely to be affected by the background noise. Small changes in the intensity of brightness and slight modifications in angles can significantly affect its ability to discriminate [5]. However, the transformer-based DINO V2 is a shape-bias model[8] that is more likely to focus on the relevant features on the stacks and more robust to the noise introduced by the background texture.

For future work, one can try to apply extra preprocessing steps to the dataset with techniques like locating stacks with bounding boxes or removing the background to reduce the noise of the image. Also, one can experiment with different CNN architectures and configurations such as adding one more fully connected layer or changing the number of neurons in the fully connected layers, to achieve the optimum CNN architecture for this task. Last but not least, one can extend the comparison to more available CNN and ViT models to obtain a more generalizable conclusion about the difference of physical reasoning ability between these two deep learning algorithms.

To conclude, both the VGG19 and DINO V2 models we implemented obtained accuracy above the baseline of 40%, indicating that they hold the ability to identify stable heights in object stacking. The Kaggle test set performance for DINO V2 and VGG19 are 63.853% and 54.583%, respectively. As the DINO V2 outperforms the VGG19, we selected the DINO V2 transformer model as our final submission to the Kaggle competition.

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)*, 2018, pp. 702–717.

[2] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[4] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship network for autonomous robotics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2018, pp. 118–125.

[5] G. Rangel, J. C. Cuevas-Tello, J. Nunez-Varela, C. Puente, and A. G. Silva-Trujillo, "A survey on convolutional neural networks and their performance limitations in image recognition tasks," *Journal of sensors*, vol. 2024, no. 1, p. 2 797 320, 2024.

[6] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[7] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] S. Tuli, I. Dasgupta, E. Grant, and T. L. Griffiths, "Are convolutional neural networks or transformers more like human vision?" *arXiv preprint arXiv:2105.07197*, 2021.

[9] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 296–23 308, 2021.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.