How can we optimise the performance of physical prediction models via perspective selection?

Paras Stefanopoulos (u7300546), Matthew Blau (u7398973), Angus Atkinson (u7117106)

Table of Contents

Question 1 (3 marks)

Please describe your research question. What is it you are trying to do? Please give enough details so that any of the tutors can understand it. (Max 250 words)

Within the Artificial Intelligence (AI) community, an active field of research is intuitive physics. This field explores how we can create AI systems capable of thinking about physical reasoning in a manner that resembles human intuition. AI systems which reason accurately about the world are essential to applications in robotics, autonomous vehicles, and answering Google CAPTCHA prompts.

A commonly used test of physical reasoning ability is the tower stability task, where AI models are presented with an image of a 3D block tower environment and asked to predict whether it will fall. Researchers have worked on this problem for some time, with models reaching accuracies of around 80% (Groth et al. 2018).

One problem that AI models face when solving this task is that it is very challenging to determine the stability of a tower from some perspectives. For example, Figure 1. shows two images of the same block tower taken from different perspectives. The image on the left suggests a stable block tower, while the image on the right suggests the opposite.

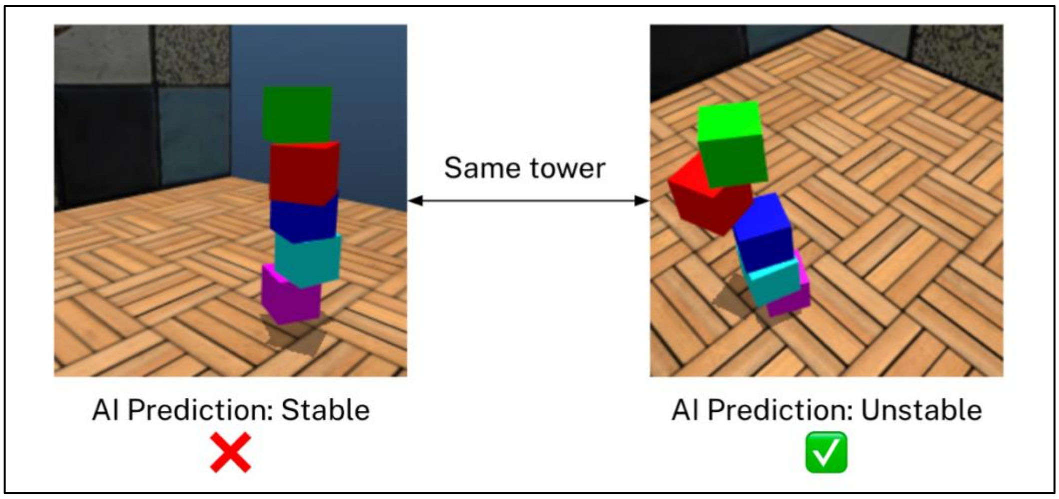

Figure 1: This figure depicts the same tower, from two different perspectives. The ShapeStacks model outputs different predictions for the two towers due to occlusion caused by the first view.

Inspired by examples like this, we hypothesise that the perspective significantly impacts the accuracy of existing tower stability models. To explore our hypothesis, we will develop a new model, which, given an image of a block tower, can predict the best perspective to view it from to optimise a stability prediction model's performance.

Ultimately, we will aim to answer the following question:

How can we optimise the performance of physical prediction models via perspective selection?

Question 2 (3 marks)

Please motivate your research question: Why did you think this question is interesting and important? (Max 500 words)

A primary goal of the intuitive physics field is to develop artificially intelligent systems that perform with high accuracy and are fit for real-world deployment. This motivation inherently influences the models produced in this field, particularly their input structure. With current technology, it is impractical and often impossible to capture complete symbolic representations of real-world environments, including the physical properties of all objects within the system. For this reason, many state-of-the-art models choose to use visual inputs (images), which are much easier to capture in the real world. These inputs are also more closely aligned with the underlying motivation of the intuitive physics field - replicating human-level intuition. Since humans generally base their physical reasoning on visual information, it seems logical that AI models inspired by human cognition should do the same.

While prediction models that take visual input are practical, there is a caveat: such models must select a perspective to view the 3D environment. As previously described, we hypothesise that choosing this perspective is no trivial decision. From our initial analysis of the ShapeStacks stability prediction model, we found that the model's accuracy varies according to the viewpoint of the image provided to it. For example, consider the tower in Figure 2 from the ShapeStacks dataset. When the 16 possible images of the block tower (taken from different perspectives) go through the stability prediction model, the model incorrectly predicts “stable” on three. We suspect this is because some images occlude information.

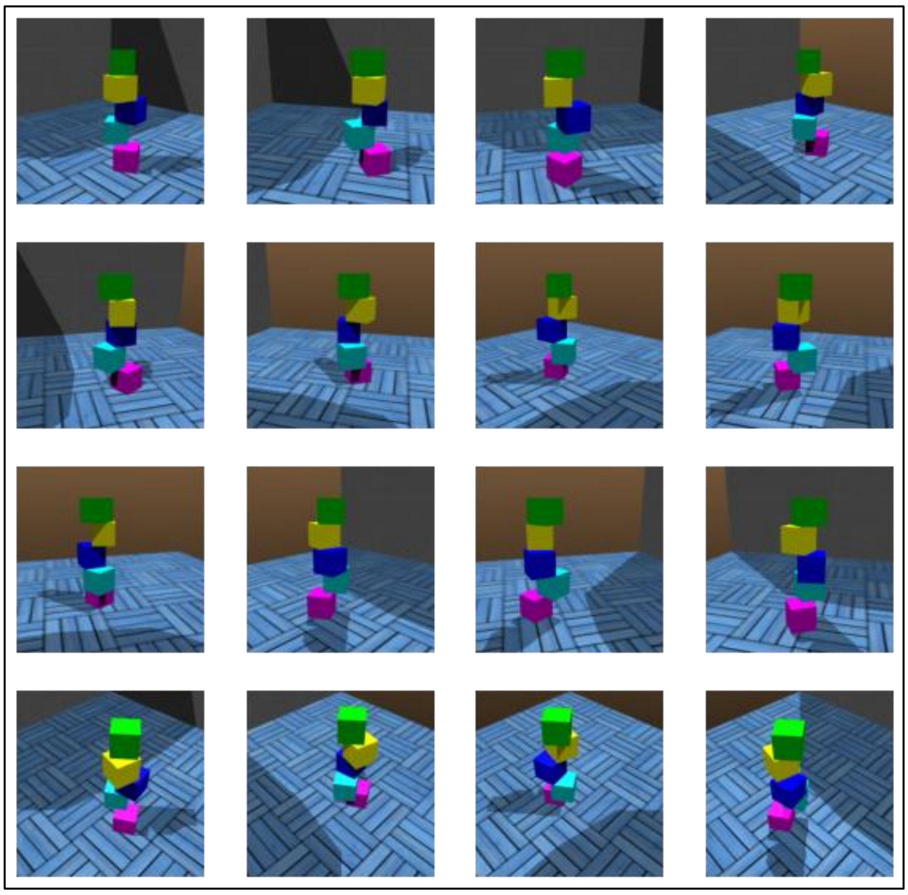

Figure 2: 16 images of the same tower, from different perspectives, as provided by ShapeStacks dataset. Images are sequential across rows, counting from perspective index 1 through to 16.

One way of eliminating the problem of perspective selection is to run a prediction model multiple times with inputs from arbitrary perspectives and apply some function to combine the outputs into a single prediction result. However, this approach is not ideal for several reasons:

1. It is time-consuming to capture multiple images from several perspectives, making this approach unsuitable for real-world applications that require a fast feedback loop.
2. Computation time and energy requirements scale linearly with the number of perspectives considered, posing a problem when running heavyweight prediction models on compute/power-constrained embedded devices.
3. This approach has diminishing returns as not all perspectives will yield additional information about the tower (particularly if certain viewpoints occlude information).

Motivated by the need for a more computationally efficient approach that better suits real-world applications, we are interested in exploring how one can apply perspective selection to this problem. If an AI system could predict the best perspective to view a tower from a single arbitrary image, it would only need to capture and process a single additional image to yield an accurate stability prediction. Our observations of human behaviour suggest that this approach may be viable: humans seem to be capable of intuitively identifying an optimal viewing angle when presented with an ambiguous perspective of a block tower. While we have focused the scope of our project on tower stability, this motivation is equally relevant to any prediction tasks influenced by perspective.

Question 3 (6 marks)

What is the current state of the art regarding your research question. Please give an overall summary (max 750 words, 3 marks) and describe the 2-3 (depending on the size of the team) most relevant state of the art papers in detail (max 500 words per paper, 3 marks). Please specify who summarised which paper, as the 3 marks per paper are for the person summarising it.

To the best of our knowledge, perspective selection has not yet been investigated as a method of improving the performance of physical prediction models. Although this means that there is no current state-of-the-art, related ideas have been explored in the literature of adjacent fields.

(Bonaventura et al. 2018) explore a variety of techniques for determining the best perspective of a 3D model. Many of the approaches rely on some metric for the quality of a perspective - such as the length and curvature of object silhouettes, variance in depth map data, or measures of (dis)similarity between perspectives, to name a few examples. However, the majority of these techniques are not directly applicable to the tower stability problem, because they either require the ability to render arbitrary perspectives of a scene, or are computed from the symbolic representation of the 3D model. For instance, the 'Viewpoint Entropy' metric (Vazquez et al., 2001) considers the projected area (i.e. the area visible to the camera) of each face in the scene from a variety of perspectives, generating a probability distribution. This is necessary to apply the Shannon entropy formula. This approach is not feasible for real-world tower stability prediction models, as many of the surfaces in the tower would be occluded from the camera.

Within robotics, active vision (the ability to manipulate the viewpoint of a camera) is essential for discovering unknown physical environments. For effective navigation, it is necessary to determine the optimal sequence of views from a camera or sensor to maximise the amount of useful information about the environment. This problem, known as *view planning*, is characterised by (Zeng et al. 2020) as one of the most challenging problems in the field, because each individual view only provides a limited subset of the necessary information to make an informed decision about the environment. A similar technique, known as *Next Best View Estimation*, is commonly applied to the field of 3D computer vision. Since 3D model reconstruction algorithms are very computationally expensive, it is essential to develop lightweight models which can determine the best sequence of input perspectives (Massiosy & Fisherz 1998). This allows the amount of *useful* data to be maximised over a minimal number of views. However, techniques from these fields cannot be directly applied to our perspective selection problem, due to the differences in input/output data, as contrasted below.

PERSPECTIVE SELECTION (OUR PROJECT)	VIEW PLANNING/NEXT BEST VIEW ESTIMATION
Minimises computational/energy usage of running a heavyweight prediction model	Minimise computational/energy usage of running a heavyweight prediction model
Identify all ‘good’ perspectives	Generate the best *sequence* of perspectives
Choose good perspectives based on single input image	Choose next best perspective based on all views seen so far
Maximise *total* information in a single view	Maximise *gain* in information from a corpus of previous views
View selected from one of 12 predefined candidate perspectives	Possible views limited to the range of motion of the robot
Loosely integrated with prediction model	Tightly integrated with prediction model
Applied to physical prediction problems	Applied to object reconstruction, scene reconstruction, object recognition, pose estimation

Table 1: Comparison of Perspective Selection, View Planning and Next Best View Estimation

View selection is also an important area of research for the 3D pose recognition community. Given a 2D depth map of a human hand, it is possible to re-project the same data points from a different 'virtual' perspective, which may improve the accuracy of pose recognition models. However, pose recognition is often applied in real-time to video data, making it computationally infeasible to rerun the heavyweight pose recognition model from many perspectives for each frame. Like in the tower stability prediction problem, this restriction on performance motivates the need for a view selection model. In the *Efficient Virtual View Selection* paper (Cheng et al. 2022), which is discussed in detail below, several methods for perspective selection are explored. One approach uses a lightweight confidence network based on the Resnet-18 (He et al. 2015) to predict the highest quality perspectives. This method is highly relevant to our research project because it demonstrates a potential model architecture for predicting one or more useful perspectives from a single input image, without any need to capture images from additional viewpoints. However, Cheng et. al. apply the pose recognition model to all of the identified best perspectives and combines the outputs to give a more accurate result. This is not likely to be possible for our perspective selection model, as the binary classification output from the prediction model does not contain enough data to combine multiple predictions.

Efficient Virtual View Selection for 3D Hand Pose Estimation (Cheng et al. 2022) - Paras

Cheng's paper explores how a single depth image can be used to produce "virtual views", which act like perspectives in order to improve the efficiency and accuracy of a 3D hand pose estimation model.

This problem and approach shares many similarities with that of predicting the quality of perspectives for a stability model. In both cases we take an input, estimate how good surrounding perspectives would be and evaluate by running those perspectives through an end model. With the idea of improving the efficiency of the complete pipeline for real world application.

Given a depth image, Cheng's three approaches are as follows, each is an iterative improvement on the last aiming for more efficiency:

1. Generate 25 "virtual views" by reprojecting the image to the different angles. Run all of these through the hand pose estimation model and aggregate the results. (Least efficient)
2. Generate the 25 virtual views, but this time use a view selection model to choose the best *n* options. Run the estimation model on this reduced number and combine to an aggregated result. (More efficient)
3. Given the single depth image, build a lightweight view selection model to choose the best *n* poses based on a "confidence" score without actually generating the virtual views. Then generate only these *n* virtual views and run the model on them, aggregating the result. (Most efficient)

The third approach is what the majority of the paper is focused on and is incredibly similar to what we are trying to achieve. First Cheng creates a dataset. To do this he first takes depth images of hands and reprojects them to the 25 views they want to investigate. Cheng then builds a sophisticated and expensive “confidence network” which can determine which views out of these 25 are the best. Passing the hands through this confidence network produces a dataset of “target” like vectors. Through the technique of distillation, these are used to train a much simpler, ResNet (He et al. 2015) based model to be able to take a depth image and predict which perspectives are the most useful. This ResNet model is considered “lightweight” and “*...effectively removes the necessity of re-rendering the input depth to all virtual cameras*” (Cheng et al. 2022).

The finalised “pipeline” is visible in Figure 3 below:

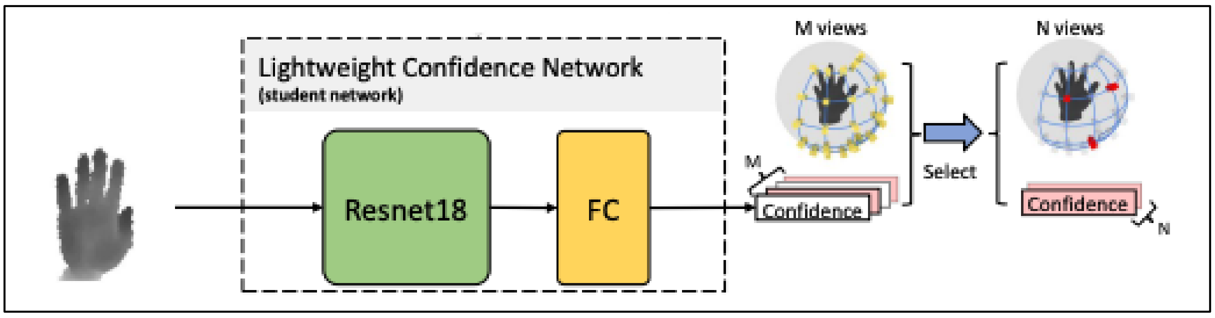

Figure 3: Model architecture for efficient virtual view selection hand pose estimation model

This pipeline performs very well. Using the ResNet, “... *student network almost doubles FPS,”* (Cheng et al. 2022) as opposed to the sophisticated confidence network. Along with this, the model with this view selection component performs better than randomly sampling the possible views in every benchmark.

Thus, Cheng’s model demonstrates that the idea is in fact valid. View/perspective selection in a pipeline of this form can improve computational performance relative to investigating all perspectives and improve accuracy relative to randomly sampling one perspective.

Next-best-view regression using a 3D convolutional neural network (Vasquez-Gomez et al. 2021) - Matthew

In this paper, Vasquez-Gomez et al. present a state-of-the-art model for next-best-view-estimation. Given a partial representation of a 3D object, this model predicts where to place an additional sensor (and at what angle) to best measure the maximum surface area that the partial model does not already capture. This paper is motivated by the goal of building real-world applications that can create accurate reconstructions of 3D objects, such as heritage sites (Vasquez-Gomez et al. 2021).

As input, the model takes a 'partial model' of a 3D object. A partial model is a probabilistic grid of cubes, where each cube contains a probability of being occupied. This probabilistic grid (dim. 32 x 32 x 32) goes through four 3D convolutional layers and five dense layers before the model predicts the position of the next best view. The idea is that "the 3D convolutional layers extract features from the grid to a 4D vector" (Vasquez-Gomez et al. 2021) before the features go through the dense layers to decide the next best view. The below diagram illustrates this model architecture:

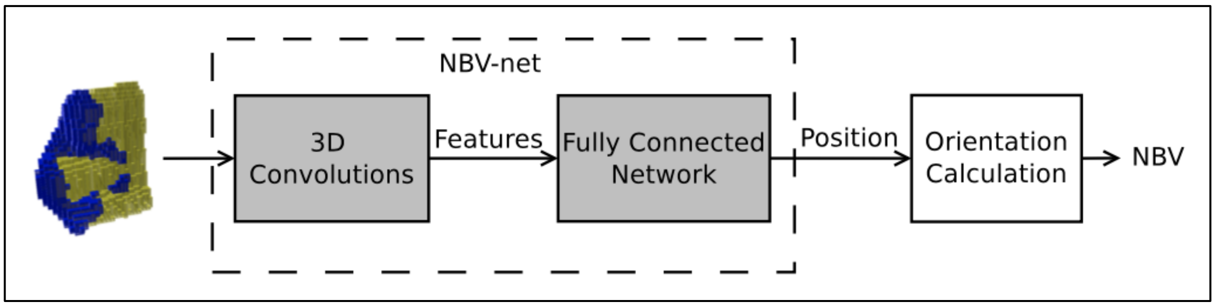

Figure 4: Model architecture for next best view regression

Vasquez-Gomez et al.'s model builds on previous state-of-the-art models in several ways:

1. Earlier data-driven models approach next-best-view as a classification problem, where they predict an optimal sensor location from a discrete set of views. On the other hand, Vasquez Gomez et al. 's model addresses the next-best-view as a regression problem,

outputting a sensor view in a continuous domain. While more complex, this approach is more effective than previous state-of-the-art models as it provides the absolute true next-best-view rather than the next-best-view from a set of views (which may not be very good).

2. When assessing Vasquez et al.'s model against a state of the art classification next-best-view model, it achieved better reconstruction coverage across ~77% of test object categories.
3. Vasquez Gomez et al.'s model shows a ~99% reduction in response time vs. a state-of-the-art GoFAI next-best-view model, indicating significantly better performance.

Vasquez et al.'s model is distinct from the model we intend to build. Firstly it is looking to ascertain the 'next-best-view' given a particular context, rather than a single optimum view from which to look at a 3D object. It also takes a 3D partial model as input rather than a 2D representation of a 3D environment.

Nonetheless, this paper is still relevant to our research as it provides insight into the field of next-best-view, the closest research problem to our perspective selection problem. Specifically, this paper is concerned with determining the best perspective from which to view a 3D object, which aligns closely with our research question. The greatest takeaway from Vasquez Gomez et al.'s paper (in the context of our research) is the idea behind their model architecture. Pairing a convolutional neural network (for feature extraction) with a series of dense layers seems like a promising approach for building our model.

ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking (Groth et al. 2018) - Angus

This paper introduces *ShapeStacks,* an open benchmark dataset for the tower stability prediction problem. It features 20,000 block towers ('scenarios'), each of which are captured from 16 different perspectives in RGB and depth map formats. The dataset is broken into two halves: 'Cubes' and 'CCS' (cuboids, cylinders and spheres). Similar to previous tower stability benchmarks (such as those in (Lerer, Gross & Fergus 2016) and (Wu et al. 2017)), the Cubes category only includes towers constructed entirely from cubes of the same. However, the CCS category extends beyond existing datasets by including additional object geometries such as cylinders and spheres. Furthermore, all images in the dataset are annotated with the stability status (stable/unstable), the number and type of stacking violations (e.g. unstable centre of mass, or object stacked on curved surface), and point-of-failure information. This dataset is perfect for training our perspective selection model because it provides a variety of perspectives of each tower, without any need to modify the benchmark or generate new images. The depth map and point-of-failure annotation data could be also useful for further research on this problem using more advanced model architectures.

To provide a baseline for this new dataset, the ShapeStacks paper trained two tower stability prediction models: one based on the AlexNet (Krizhevsky, Sutskever & E. Hinton 2012) convolutional neural network architecture, and another based on the Inception v4 (Christian et al. 2016) architecture. Both networks were separately trained on each half of the dataset to evaluate the overall effect of training dataset complexity on model performance. The researchers found that the Inception v4 model performed the best, with the Cubes/CCS variants scoring 77.7%/84.9% respectively on the Shapestacks test set, as opposed to AlexNet's 60.5%/58.8%. The higher accuracy of models trained on the CCS dataset was attributed to the greater diversity and variation in the training stacks. Although the Cubes-based model had lower performance, the training dataset was considerably less complex, which made it a good candidate for our initial attempt at the view selection problem.

Furthermore, when the researchers applied the Cubes/Inception v4 variant of the model to a common real-world tower stability benchmark (Lerer, Gross & Fergus 2016), it achieved 74.7% accuracy. This outperformed the baseline Physnet model (66.7%) for that dataset, and traded blows with the resource-intensive visual de-animation approach explored in (Wu et al. 2017). Therefore the ShapeStacks model achieved state-of-the-art performance in both real-world and synthetic benchmarks.

To prove that the ShapeStacks model learned an intuition for tower stability, the researchers applied it to several new tasks. First, a gaussian blur was applied to various parts of the image, and the model was rerun on each augmented input, allowing a heatmap to be generated representing the focus of the model. On average, there was a 79.9% correlation between the localisation heatmap and the violating region of unstable towers, suggesting that an intuition for stability was learned. Furthermore, the model was successfully adapted to actively stack blocks on top of one another. It performed this task quite well, routinely stacking towers up to 12 blocks high (well beyond the limits of the training dataset), and even counterbalancing unstable structures (a very difficult problem even for humans).

Question 4 (6 marks)

Please report in detail what you have done to solve your research question in chronological order. Please describe what worked, but also what didn't work, as well as why you decided to do the steps you described. (Max 2000 words)

The tower stability prediction problem is a common benchmark in the field of intuitive physics and physical reasoning. Many AI models attempting to solve this problem were discussed in the papers assigned for our tutorials and assignments. During these discussions, a common question was what the effect of perspective might be on visual prediction model's performance. We noted that humans typically view a tower from a variety of perspectives before making a prediction, whereas an AI model is often restricted to a single 2D image.

It was hypothesised that the performance of these sorts of AI models is dictated by the perspective of the input image. This gave us the idea of developing a perspective selection model capable of determining which views give the model the best chance of achieving optimal prediction.

To develop this model, we would need some notion of how good a given view is. However, in practice, this was not easy to quantify. Instead, we chose to define a good view as one where a tower stability prediction model performs well in comparison to other perspectives. Hence, two main resources would be essential for training a perspective selection model in this context:

1. A dataset of towers, along with:
    a. images of each tower from a variety of perspectives, or
    b. source code which can be used to generate images from new perspectives
2. An AI model which evaluates the stability of a tower-based on an input image

Whilst researching a variety of models and datasets, we came across the ShapeStacks paper (Groth et al. 2018). This met both of our key criteria, providing a pre-trained stability prediction model, and a variety of tower environments each containing 16 images from various perspectives.

However, after some time spent trawling the internet for these resources, we realised the relevant download links for the model and dataset were broken. Therefore, we chose to reach out to the paper's lead author, Oliver Groth, who is now a research scientist at DeepMind. Dr Groth kindly sent over the requested resources, and also provided the helpful suggestion that we take a look at the well-established related field of Next-Best-View-Estimation for further inspiration.

As described in the literature review section, whilst researching this field we came across (Cheng et al. 2022). This paper inspired us to investigate using a pre-trained convolutional network for feature extraction from the images and then training our own dense layers on the end.

We now felt we had the knowledge and resources to refine our research focus and shortly decided on, *"how can we optimise the performance of physical prediction models via perspective selection?",* as our research question. Our goal would be to develop a neural network that, given an image of a block tower, can predict the best perspective to view it.

Phase 1: Planning Our Approach

To get a sense of how we would build such a model, we sought the advice of Jack Miller, a machine learning researcher. Following a conversation with Jack, we came up with a high-level plan for the architecture of our model:

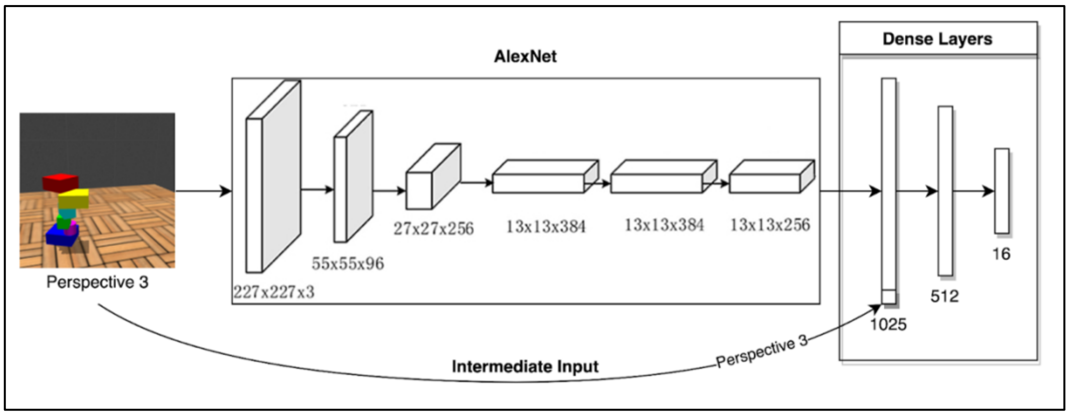

Figure 5: The above diagram depicts the initial architecture for our model, consisting of its input, a convolutional neural net as provided by AlexNet and a sequential system of dense layers which we will train.

Table 2 describes our justification and thoughts behind the more significant design decisions associated with the model:

MODEL INPUT
The model will take a single input image, represented as a vector of dimension 224x224x3 from the ShapeStacks dataset. This image depicts one of the ShapeStacks block tower environments from 1 of the 16 possible perspectives.
CONVOLUTIONAL NEURAL NETWORK (CNN) COMPONENT
The input image will be passed through a pre-trained image classification CNN for feature extraction. We will use the convolutional part of AlexNet (Krizhevsky, Sutskever & E. Hinton 2012) for our CNN, which was recommended to us by Jack. We are motivated to use AlexNet because it is a tried-and-tested model with plenty of online resources to support our implementation. While the ShapeStacks stability prediction model uses the Inception v4 network (Christian et al. 2016), which consistently outperformed AlexNet in the ShapeStacks paper, we think that the relatively complex architecture of Inception v4 could pose too many complications for the scope of this project.
INTERMEDIATE PERSPECTIVE INDEX INPUT
To deal with the problem of the model not knowing which of the 16 perspectives the input image corresponds to, we will pass in a perspective index value to the model at an intermediate layer (the first fully connected layer).
FULLY CONNECTED LAYERS
The CNN will connect to three fully connected layers of dimensions 1024→512→16. We will only train these fully connected layers, not altering the pre-trained parameters in AlexNet's convolutional component.
MODEL OUTPUT
The final output of the model will be a dimension 16 vector. Each feature corresponding to a different perspective of the inputted block tower environment (perspectives 1→16). Each value represents how good the model predicts the perspective is. As per this design, all input images that depict the same tower environment should result in the same output vector when fed to the model (Quality of perspectives should be constant).

Table 2: Justifications and thought process behind initial design decisions. Was written in dot points during the meeting with jack, later turned into cohesive paragraphs.

With this model architecture agreed upon, we needed to produce target vectors that would act as our training data, defining which perspectives are good and which are bad for each tower in the dataset. These vectors would be used to calculate our loss function, which was initially Mean Squared Error (MSE) as suggested by Jack. These target vectors were generated as follows:

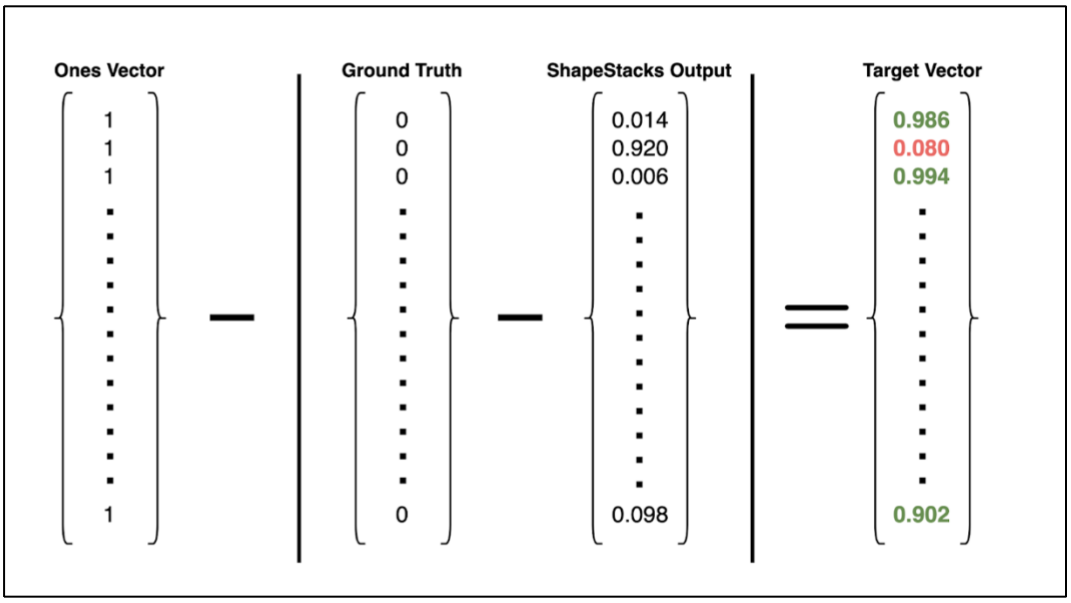

Figure 6: Example of formula used to generate target vectors for a given tower environment consisting of 16 perspectives.

Under this design, 'good' perspectives would have values closer to 1 while 'bad' ones would be closer to 0.

Ground Truth Vectors

Producing ground truth vectors was relatively straightforward since we already had the stability labels for each environment from the ShapeStacks dataset. Using these labels, we generated a dimension 16 ground truth vector for each environment/tower. All values in these vectors were the same, as the perspective from which one looks at a block tower does not change the tower's true stability.

Predicted Stability Vectors

We then needed to generate the predicted stability vectors, which would require forward passing all ShapeStacks images through the pre-trained ShapeStacks stability model. This would require around 100 gigabytes of disk space. We also needed a Nvidia GPU that supports CUDA to accelerate this computation. After investigating the limited options available to us, we decided that Google Collab Pro would be the most suitable environment and would be worth the minor investment.

Within Google Collab, we first rolled back the TensorFlow environment to V1 to match the version compatible with the pre-trained ShapeStacks model. We then developed a data-feeder and functions to collate outputs pertaining to the same tower into the desired vectors (with entries ordered by perspective index).

We wrote these raw output vectors to disk so that we didn't need to re-run this forward-pass when training our model. With both the model outputs and the ground truth vectors we then produced our final target vectors as previously described.

Phase 2: Implementing the Model

We now had training data and an idea of a suitable model architecture. At this point, we decided it would be wise to spend time increasing our knowledge of neural networks. Matthew and Angus specifically spent time watching lectures from MIT OpenCourseWare to learn more about the theoretical side of neural networks. Paras focused on the more practical side of things, reading PyTorch documentation and reviewing/implementing code examples.

We also discussed our plan with our tutor, Alexander Yang, to see if he could identify any flaws in our model architecture and provide helpful advice. During this conversation, we mentioned that we were planning on using AlexNet as the CNN component of our model. Alex's view was that AlexNet is a legacy CNN and that there are more contemporary CNNs that perform better and could be a better choice. He mentioned ResNet (He et al. 2015) as an example of one of these CNNs. This was consistent with the approach taken in (Cheng et al. 2022).

ResNet seemed to suit our project as there are several versions of ResNet with varying parameter sizes, giving us the option to start with a relatively small model and scale it up as needed. ResNet-18 in comparison to AlexNet has approximately 1/6th the number of parameters meaning it is more memory efficient and is more relevant given our previously motivated application of real-world AI and embedded systems.

Not surprisingly, our first working model did not converge. Table 3 describes the model's performance over 4 epochs using the MSE loss function.

Epoch	Loss (lower is better)	Change
1	0.1542	+0.0000
2	0.1536	-0.0006
3	0.1692	+0.0156
4	0.1531	-0.0161

Table 3: Depicts change in MSE loss over 4 epochs with initial model architecture

At first, we attempted to tune several of the model's hyperparameters, but this was unfortunately to no avail. We decided that we would need to make more drastic changes to make our model converge.

Phase 3: Iterative Adjustments

Removing intermediate perspective index input

One thought as to why our model wouldn't converge was that passing the perspective index of the input image as an intermediate input may be causing unintended problems. We also realised that this approach was slightly contrived when considering the real-world applicability of our model. A real-world AI agent would likely not have access to a perspective index to pass to their perspective-selection model. They would only have the input image.

For this reason, we decided to adapt the model so that it doesn't require a perspective index as an intermediate input. We took three steps to reconstruct the model such that it wouldn't need to know the perspective from which it is looking.

1. We observed that 12 of the 16 ShapeStacks perspectives were evenly sampled from an oblique circular path, while the remaining four were not. As we wanted to create a continuous perspective path to support the consecutive, relative ordering of perspectives, we decided to remove these entries. See Figure 7.

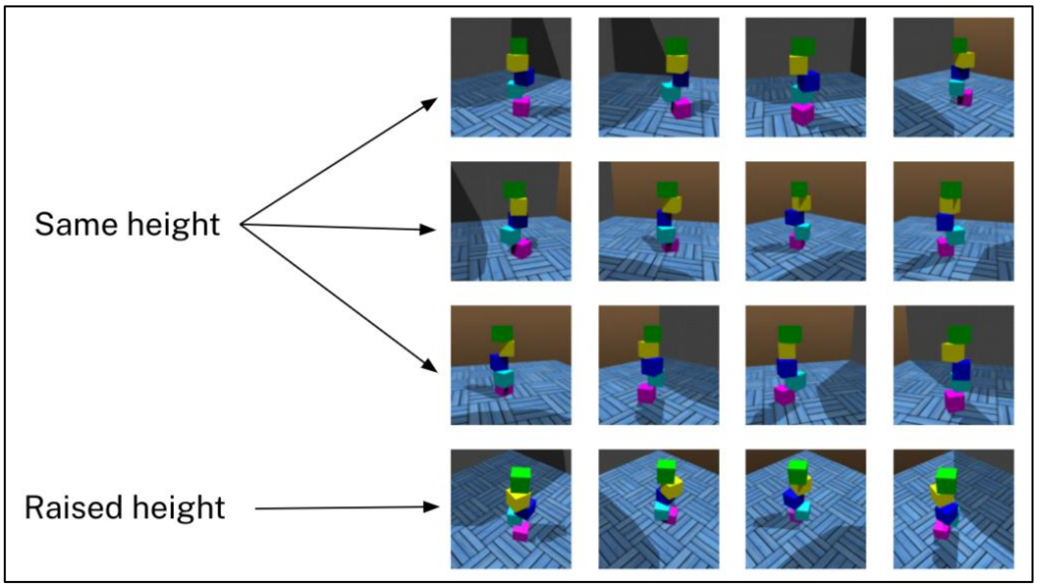

Figure 7: 16 images are of a tower environment provided by the ShapeStacks dataset. The difference in perspective heights is then pointed out.

2. We also adjusted the target vectors so that the ordering of the entries is consistent with the (clockwise) order of the perspectives around this continuous circular path. Perspectives aren't ordered 1, 2, 3, etc, with any notion of circular continuity.
3. Finally, we cycled the target vector so that the first value always corresponds to the perspective of the input image. The subsequent feature corresponds to the perspective immediately clockwise of the input perspective, and so on around the tower in a circular path. This ensures that the target vector identifies which perspectives are good, relative to the position of the input image.

Our final model and training architecture looked as follows:

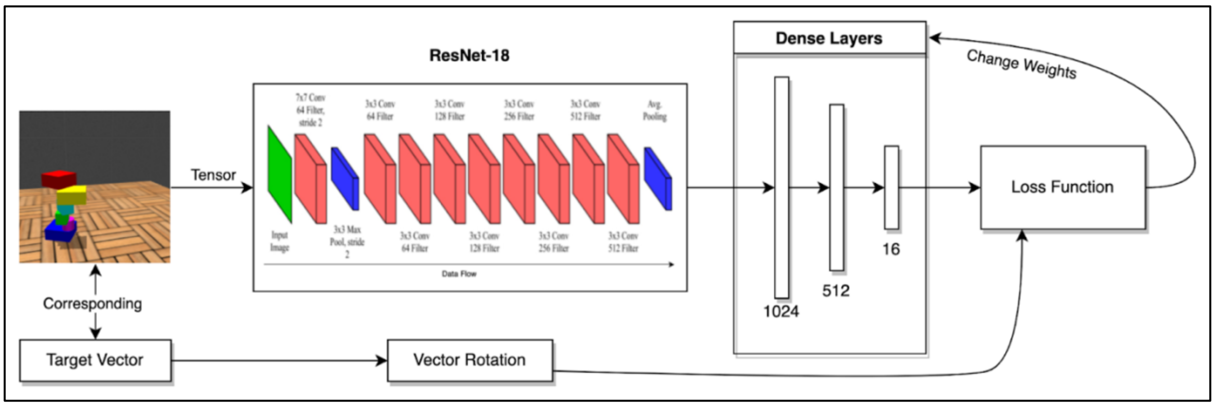

Figure 8: Final model and training architecture, depicting an input image, ResNet-18 feeding to a trained sequence of dense layers, feeding to a loss function. It also depicts how an image's corresponding target vector is produced and then rotated to make it a "relative" vector, before being passed to the loss function.

Increasing Density of Bad Perspectives

We also considered the possibility that our dataset was not rich enough for our model to converge. Our main concern was that there appeared to be a high proportion of block-tower environments in which perspective seemingly doesn't affect the ability of the stability model to predict stability. The graph below confirmed this by showing that the dataset contains over 9000 flawless block tower environments (76% of the dataset). We defined 'flawless' block tower environments as those in which the model's stability prediction did not vary regardless of the perspective, which would suggest that all viewing perspectives are equally valid. We theorized that our model would not be able to learn anything from these flawless block tower environments and therefore wanted to eliminate them.

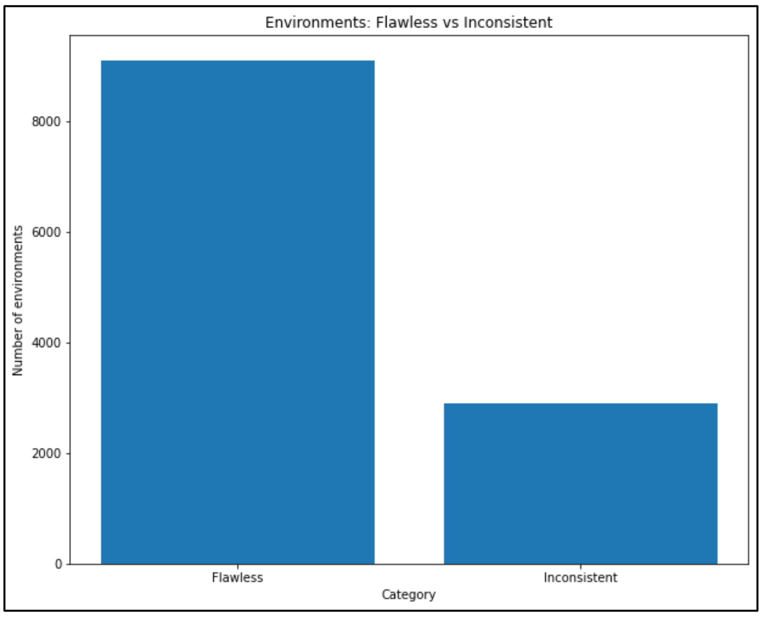

Figure 9: Frequency of "flawless" and "inconsistent" tower environments. A flawless tower environment is defined as a tower where ShapeStacks performs consistently on all perspectives. Inconsistent implies perspective influences ShapeStack's output.

Changing Loss Function

We also considered the possibility that Mean-Squared Error was not the appropriate loss function for our model, and that this was causing the lack of convergence. To solve this, we re-implemented our model with our model with two new loss functions that we perceived to be appropriate - Cross-Entropy Loss (CEL) and Euclidean distance. These two loss functions were chosen because of a brief conversation with Alex. They at first appeared to be somewhat applicable but did not result in convergence.

To implement CEL, standardisation of both the target vectors and the output layer of our PyTorch model was required. For this we used the SoftMax function, which converts a vector of numbers into a probability vector.

Scaling Target Vectors

Due to the negatively skewed nature of the target vectors (see Figure 10), we also attempted scaling the data such that across features, the mean was 0 and standard deviation was 1. The idea was that this may assist the MSE loss function however this also proved to no avail.

Phase 4: Evaluating where we went wrong

At this point in the project, we had no more timely ideas to practically resolve our inconclusive result. As such, we moved on to evaluating our work and identifying where we could have gone wrong.

Question 5 (3 marks)

How did you distribute the work among your team? Who did what? Why did you distribute the work in that way? (Max 500 words, ideally tabular form)

CONTRIBUTORS	ACTION
ALL	Analysed literature related to our research idea and shared knowledge
ALL	Identified and explored a suitable dataset and corresponding physical prediction model to help facilitate our research idea
Paras	Downloaded the ShapeStacks stability prediction model, implemented it in Google Collab and forward passed the ShapeStacks images.
ALL	Formed a precise research question that can be solved using machine learning. Investigated neural networks and their architectures.
ALL	Met with Jack Miller, an AI researcher at the ANU, to learn more about neural networks and how to implement them.
Paras & Matt	Familiarised ourselves with PyTorch and implemented a model, inc. a data loader and training code.
Angus & Paras	Worked on improving the model, identified problems with perspective index input and implemented code to improve upon this.
ALL	Analysed inconclusive results and collaborated to find reasons the model wasn't converging. Attempted new loss functions and other model tweaks.
Angus	Produced slides/speech and delivered a 3 minute presentation summarising progress.
Paras & Matt	Produced slides/speech and delivered a 15 minute presentation on our overall research.
ALL	Report writing $(Dx = Draft\ x)$: Q1: Angus (D1), Matt (D2) Q2: Matt (D1), Angus (D2), Matt (D3) Q3: Angus (literature overview), everyone (individual paper reviews) Q4: Matt (D1), Angus and Paras (D2) Q5: Paras (D1), Matt (D2) Q6: Paras (D1), Matt (D2) Q7: Paras (D1), Matt (D2) Q8: Paras (D1), Matt (D2)

Table 4: Contains details about the contributions of different team members. Contributor names in the first column, and the action they performed in the second. Contributions are ordered chronologically.

From a high level:

- Angus completed a literature review on the field of Next Best View Estimation and contributed to improving the model.
- Paras and Matt worked across all aspects of the project, including developing the model and writing the report. Paras primarily focused on the model, while Matt focused most of his effort on writing the research report.
- Paras and Matt collaborated to produce the 15-minute presentation, whilst Angus took care of the 3-minute presentation.

However, we collaborated across most aspects of the project, coming up with solutions to challenging problems through dialogue and debate.

Through regular meetings and constant Discord chat communication, the tasks were distributed as per the interests and declared skills of each group member as follows:

- Paras was most confident with the technical side of things and looked to develop his research abilities. Accordingly, he led the technical implementation of the project. He also worked with Matt on research and analysis, looking to improve his writing skills.
- Angus wanted to focus primarily on background research of the existing literature, solidifying the motivation and justification for the project. He also aimed to develop skills in machine learning by iteratively improving the initial model built by Paras.
- Matt felt most confident with the theoretical side of the research - exploring the motivation behind the research question and analysing ways of solving it. Matt also looked to develop his technical ML skills by working alongside Paras on developing the model.

Toward the latter half of the project, we spent most of our time working on tasks collaboratively in a shared space.

Question 6 (3 marks)

Please describe the outcome of your research? What positive or negative results have you achieved? Please note that we do not expect positive outcomes. (Max 500 words)

As our model did not converge, our results were inconclusive in answering our research question. However, our analysis of this inconclusive result has resulted in an in-depth understanding of the flaws in our approach.

Fault 1: Training data quality

One suspected fault in our method was the quality of our training data. As previously discussed, 76% of dataset environments were 'flawless' (containing towers where perspective does not affect stability prediction). Thus, although we had a large dataset in volume, the data points which exposed the effect of perspective selection on stability prediction were relatively scarce. Our model may have perceived these rich data points as noise rather than signal. We believe this contributed to our model's lack of convergence.

Fault 2: Definition of "good" perspective

Another suspected fault in our approach is our method for assessing the quality of perspectives when producing our target vectors. As previously discussed, we constructed our target vectors according to the following formula:

$$target\ vector = 1 - \ |ground\ truth\ vector - ShapeStacks\ model\ prediction|$$

Under this construction, a value of '0' for a perspective in our target vector means that the ShapeStacks model was confidently wrong. The flaw in our approach is that we assume that this error is entirely attributable to the perspective being bad. This assumption neglects to consider that the ShapeStacks stability prediction model is not perfectly accurate, so this error is possibly due to the model's inaccuracy. We believe that this conflation of 'bad perspective' with ShapeStacks model error may have misdirected our model during training, preventing convergence.

Fault 3: Loss functions

Mean Squared Error (MSE)

MSE is not very robust to outliers and is thus most effective when applied to normally distributed data (*Mean squared error loss function | Peltarion Platform n.d.).* Figure 10. indicates that our dataset is far from normally distributed, with a significant negative skew towards perspectives that result in a confident correct prediction. This mismatch between the data and the loss function likely contributed to our initial model's lack of convergence.

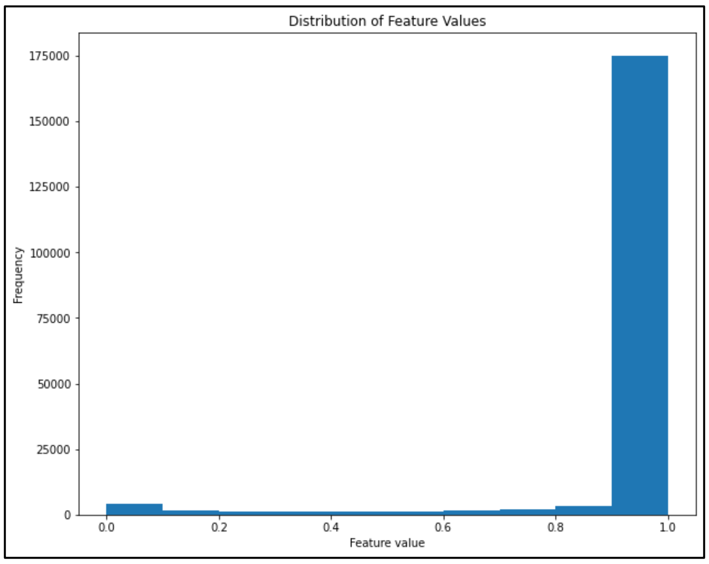

Figure 10: Frequency of target output for all 16 perspectives of every tower

Cross-Entropy Loss (CEL)

CEL is most suitable for binary and multi-class classification problems (Cross Entropy Loss n.d.). As we did not design our problem as a traditional classification task, the CEL loss function was likely unsuitable for our model and contributed to its lack of convergence.

Euclidean Distance

Euclidean Distance is primarily used for regression problems (Yalćiner 2019) and therefore appears to suit our model better. However, this loss function did not resolve the model's lack of convergence.

We believe that identifying a loss function that better fits our model and data would significantly increase the chances of our model converging.

Fault 4: Invalid premise

We also considered the possibility that the premise of our project relies on the assumption that the problem is solvable and that it is even possible to determine the quality of all other perspectives from such limited information. This assumption is questionable and could be an area of further research.

Inconclusive Conclusion

Under the assumption it is, we concluded that the aggregated effect of the flaws discussed above most likely causes our model's lack of convergence. Our research has invalidated these approaches and paves the way for more sophisticated future work.

Question 7 (3 marks)

What would be the next steps, either to answer your research question, or if answered, to extend it? (Max 500 words)

As discussed, the results of our research were inconclusive due to not being able to produce a converging model. We propose several ideas for future work that aim to solve this lack of convergence and answer our research question:

PROBLEM	DESCRIPTION	FUTURE WORK
Scarcity of useful training data	As 76% of dataset environments are 'flawless', the data points which expose the effect of perspective selection on stability prediction are relatively scarce.	One could produce a richer dataset of tower environments where perspective consistently impacts stability prediction (potentially using the underlying ShapeStacks engine). Using a dataset like this, a model like ours may have a better chance of converging.
Contrived definition of “bad perspective”	Our method for constructing target vectors conflates 'bad perspectives' with ShapeStacks model error, potentially misleading our model during training.	One could take a neuro-symbolic approach to perspective selection, considering heuristics based on factors like occlusion. This approach could provide a more well-defined structure for learning what separates a good perspective from a not-so-good perspective during training.
Inappropriate loss functions	The loss functions we explored seemingly do not fit our problem. We attempted MSE, CEL and Euclidean Distance, which all failed.	One could further explore related literature to identify an established loss function that better suits our model. Alternatively, one could develop a new loss function tailored to our model and dataset.
Input data provides no depth information	The input images we trained our model on did not contain depth information. This information may be necessary for a model to interpret a 2D image of a 3D environment to predict the utility of other potential perspectives.	One could use the provided RGBD images from the ShapeStacks dataset as input to the model. This idea yields another worthwhile research question: how important is an understanding of depth to training a model for these sorts of problems?

PROBLEM	DESCRIPTION	FUTURE WORK
Model is not ‘sophisticated’ enough	Our model is possibly just not sophisticated enough to identify patterns in the input information.	One could try train a more complex model capable of learning the required perspective-selection function. Like (Cheng et al. 2022), they could then distil this model into a simpler model like the model we proposed.

Table 5: Displays identified problems in our method, descriptions of those problems and possible future work to address these problems.

Beyond these flaws there also exists a more philosophical question of whether the problem of perspective selection in this context can be represented by a learnable function. Take the image in Figure 11, as humans, trying to estimate which perspectives are good seems beyond our IP ability, which leads us to raise this question.

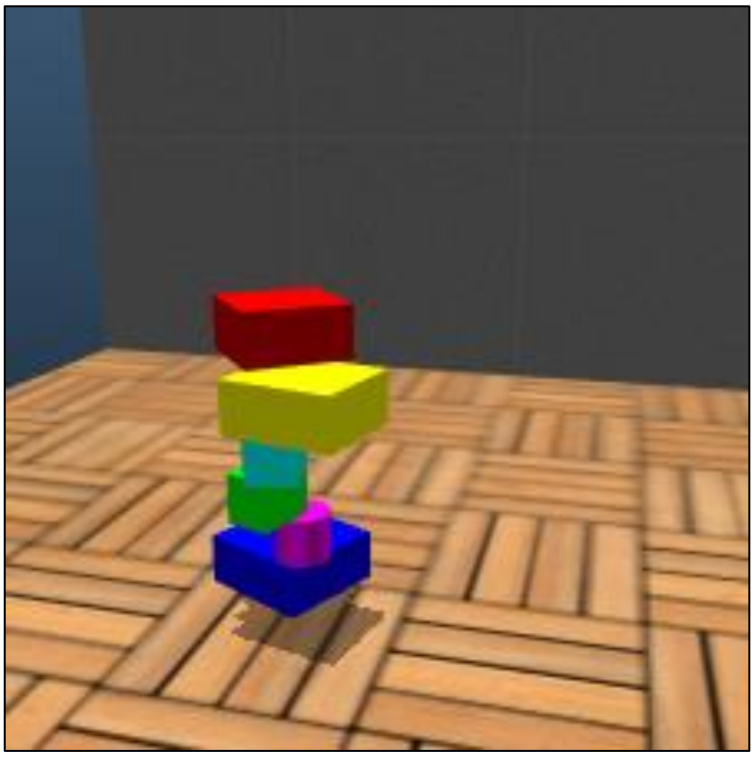

Figure 11: An example of a random image from the ShapeStacks dataset, how good are humans at predicting the validity of all other perspectives?

Our research also heralds the more philosophical question of whether a good perspective-selection function is even learnable from a single arbitrary image? And do humans have an intuition for this type of physical reasoning task? Looking at Figure 11, we are not sure that we can reliably estimate where the 'best perspective' would be.

One final idea for future research could be to investigate these questions through a study of human subjects. One could ask the subjects of this study to perform a similar task and predict how good relative perspectives are when looking at images of towers. The study results would provide insight into how feasible this task is and what additional information (such as depth) would make the perspective selection easier.

Question 8 (3 marks)

What have you learned from this research project? What would you do differently in hindsight? (Max 500 words)

This project required us to upskill in many aspects. Some gained skills were technical, some theoretical and some research based. At the start of the course, we all had little understanding of AI/ML techniques, let alone the application of these in the field of intuitive physics.

Through analysing an array of papers, we gained knowledge about the current state of the IP field, the story of the field's progression, state-of-the-art benchmarks, and models, and some of the many theoretical techniques which are most popular and emerging. We also learnt how to find papers, read them efficiently and collaborate to solidify our understanding.

Following this, we undertook our literature review and converged on a question. This process of idea, investigation, evaluation and repeat when identifying our research question was also a valuable skill gained in and of itself.

Our next step was to implement the ShapeStacks pre-trained model and construct our own. We first discovered how collaborative and open research is. We contacted the author of ShapeStacks for information regarding broken links and received a tip to investigate the Next-Best View Estimation field. We were also lucky to secure a meeting with an AI researcher, Jack Miller, to get up to speed with applied-ML fundamentals. This opportunity allowed us to rapid-fire questions, to which we received high-quality answers. These questions included:

- Which technologies should we use?
- Where can we find compute?
- What are loss functions?
- How does one identify an appropriate loss function?
- What is gradient descent?

- How do convolutional neural networks work?
- What are dense layers?

It suffices to say that we learnt a lot from this conversation.

We then went headfirst into the tech stacks and used online articles and documentation to learn technical skills such as:

- Using Google Colab and Google Drive to store bigger than memory data
- Implementing our data feeder and running a pre-trained model on TensorFlow
- Using Cuda to accelerate compute
- Manipulating, storing, and loading tensors
- Using PyTorch to develop and train a model:
  - Importing pre-existing models
  - Setting which parameters are trainable and not
  - Replacing layers of pre-trained models
  - Creating test-train splits and a data-feeder to fit the purpose
  - Running and evaluating the success of a training phase

On top of the earlier mentioned technical skills, we also gained knowledge about theoretical concepts such as:

- Basics of neural networks
- An array of loss functions and their use cases and caveats
  - Mean Squared Error
  - Cross Entropy Loss
  - Euclidean Distance
- Convolutional networks, dense layers
- Normalisation (inc. Softmax function) and standardisation of training data for different purposes

In hindsight, we could have saved time by reaching out to researchers at ANU when stuck with our loss functions. Our brute force approach meant we expended time implementing and testing loss functions instead of taking a more direct and considered approach. A more considered approach may have led to a conclusive result.

It also would have been beneficial to conduct an even more extensive literature review from an earlier stage. Doing so could have led us to arrive at a better approach or assisted in better assessing the scope of the work before diving straight in.

Nonetheless, learning how to work collaboratively, digest academic research and make progress towards a research question are skills that will benefit us for the rest of our careers, both academic and otherwise.

References

1. Bonaventura, X, Sbert, M, Feixas, M, Chuang, L & Wallraven, C 2018, *A Survey of Viewpoint Selection Methods for Polygonal Models*, 16 May.
2. Cheng, J, Ma, C, Gu, J, Tan, P, Wang, H, Deng, X, Zhang, Y, Wan, Y & Zuo, D 2022, 'Efficient Virtual View Selection for 3D Hand Pose Estimation', in *AAAI Conference on Artificial Intelligence (AAAI)*.
3. Christian, S, Alemi, A, Vanhoucke, V & Loffe, S 2016, 'Inception-ResNet and the Impact of Residual Connections on Learning', paper presented at AAAI Conference on Artificial Intelligence.
4. *Cross Entropy Loss* n.d., ml-cheatsheet.readthedocs.io.
5. Groth, O, Fuchs, FB, Posner, I & Vedald, A 2018, 'ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking', paper presented at The European Conference on Computer Vision (ECCV), viewed 30 March 2022, <https://openaccess.thecvf.com/content_ECCV_2018/papers/Oliver_Groth_ShapeStacks_Learning_Vision-Based_ECCV_2018_paper.pdf>.
6. He, K, Zhang, X, Ren, S & Sun, J 2015, *Deep Residual Learning for Image Recognition*.
7. Krizhevsky, A, Sutskever, I & E. Hinton, G 2012, 'ImageNet Classification with Deep Convolutional Neural Networks', in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, vol. 1, Curran Associates Inc., Red Hook, NY, USA, pp. 1097–1105, viewed 22 April 2022, <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
8. Lerer, A, Gross, S & Fergus, R 2016, *Learning Physical Intuition of Block Towers by Example*, 3 March.
9. Massiosy, NA & Fisherz, RB 1998, *A Best Next View Selection Algorithm incorporating a Quality Criterion*.
10. *Mean squared error loss function | Peltarion Platform* n.d., Peltarion.
11. V´azquez, P-P, Feixas, M, Sbert, M & Heidrich, W 2001, *Viewpoint Selection using Viewpoint Entropy*, November.
12. Vasquez-Gomez, JI, Troncoso, D, Becerra, I, Sucar, E & Murrieta-Cid, R 2021, 'Next-best-view regression using a 3D convolutional neural network', *Machine Vision and Applications*, vol. 32, no. 2.
13. Wu, J, Lu, E, Kohli, P, Freeman, WT & Tenenbaum, JB 2017, 'Learning to See Physics via Visual De-animation', paper presented at 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA.
14. Yalçıner, A 2019, *Euclidean Distance Loss Function*, Medium, viewed 30 May 2022, <https://medium.com/@aybukeyalcinerr/loss-functions-f324d1f57964#:~:text=Euclidean%20Distance%3A>.
15. Zeng, R, Wen, Y, Zhao, W & Liu, Y-J 2020, *View planning in robot active vision: A survey of systems, algorithms, and applications*, September.

Appendix

Our Code

https://github.com/pstefa1707/perspective-selection-model

Email Chain with Oliver Groth

“Hi Dr Groth,

My name is Paras Stefanopoulos, I am a second year Computer Science student at the ANU. I am conducting some research in a similar area to ShapeStacks.

Essentially we are looking into training a model which given multiple perspectives of the same 3D environment, selects the perspective which conveys the most information about the tower’s stability.

I believe we can use the code you have developed for ShapeStacks to speed up our process. Ideally we’d use one of your pre-trained models as our evaluation function for determining how good a perspective is. Note, I am very new to ML, and as such may get common terminology wrong/expose my little understanding.

However, I noticed the links to download your pre-trained models, from the research paper are no longer working.
I.e. on https://github.com/ogroth/shapestacks/tree/master/shapestacks_stacker.

If its not too much of a hassle, would you know of another place to source these pre-trained models.

I appreciate your time in reading this email. Loved the paper!

Kind Regards,
Paras Stefanopoulos.”
Paras to Oliver Groth 22/04/2022

“Hi Paras,

Thank you for your interest in ShapeStacks. :) Apologies, the website was moved at some point but the README links on GitHub have not been updated to reflect that. Please find the working download links here: https://ogroth.github.io/shapestacks/#source-code

(Best use a Firefox browser for the download links, recent Chrome blocks the download because it’s not HTTPS.)
Good luck with your project – that sounds like a very interesting spin on next best view estimation.

Kind regards,
Oliver”
Oliver Groth to Paras 22/04/22