University of California
Santa Barbara

Advances in Deep Space Exploration via Simulators and Machine Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

James Craig Bird

Committee in charge:

Professor Linda Petzold, Chair
Professor Chandra Krintz
Professor Philip Lubin

June 2021

The Dissertation of James Craig Bird is approved.

Professor Chandra Krintz

Professor Philip Lubin

Professor Linda Petzold, Committee Chair

May 2021

Advances in Deep Space Exploration via Simulators and Machine Learning

To my parents, Barbara and James Bird. If not for your endless support, love, and effort, I would be nowhere.

Acknowledgements

I would like to acknowledge my advisors, Dr. Linda Petzold and Dr. Philip Lubin, who took me in and inspired me to follow the path that I truly wanted to take. Their unending support along the way was crucial to my success as a student, a researcher, and an educator-in-training.

I would also like to acknowledge Dr. Chandra Krintz and her wonderful research presentations. One of which, during my first year, especially encouraged me to pursue deep learning concepts throughout my research.

To my paper collaborators - thank you for all of your hard work!

Curriculum Vitæ
James Craig Bird

Education

2021	Ph.D. in Computer Science (Expected), University of California, Santa Barbara.
2015	M.S. in Statistics, Rutgers University.
2012	M.A. in Applied Math & Applied Physics, SUNY Buffalo.
2009	B.A. in Mathematics, SUNY Binghamton.

Publications

J. Bird, K. Colburn, L. Petzold and P. Lubin, *Deep Neural Network Categorizer vs. Binary Decision Tree for Deep Space Object Detection* (Pending submission)(2021)

J. Bird, K. Colburn, L. Petzold and P. Lubin, *Novelty detection in deep space: The plausibility of template sphere training in place of simulator-based approaches* (Pending submission)(2021)

J. Bird, K. Colburn, L. Petzold and P. Lubin, *Model Optimization for Deep Space Exploration via Simulators and Deep Learning)*(Submitted to New Astronomy)(2021)
https://arxiv.org/abs/2012.14092

J. Inglies, J. Bird and G. Ashby, *A Multidimensional Thurstone-Coombs Model of Simultaneous Sensory and Liking Ratings*(Submitted to Applied Psychological Measurement)(2021)

J. Bird, L. Petzold, P. Lubin, and J. Deacon, *Advances in Deep Space Exploration via Simulators & Deep Learning* ,New Astronomy, Vol 84, 101517(2020)
https://doi.org/10.1016/j.newast.2020.101517

Abstract

Advances in Deep Space Exploration via Simulators and Machine Learning

by

James Craig Bird

The NASA Starlight and Breakthrough Starshot programs conceptualize fast interstellar travel via small relativistic spacecraft that are propelled by directed energy. This process is radically different from traditional space travel and trades large and slow spacecraft for small, fast, inexpensive, and fragile ones. The main goal of these wafer satellites is to gather useful images during their deep space journey. We introduce and solve some of the main problems that accompany this concept. First, we need an object detection system that can detect planets that we have never seen before, some containing features that we may not even know exist in the universe. Second, once we have images of exoplanets, we need a way to take these images and rank them by importance. Equipment fails and data rates are slow, thus we need a method to ensure that the most important images to humankind are the ones that are prioritized for data transfer. Finally, the energy on board is minimal and must be conserved and used sparingly. No exoplanet images should be missed, but using energy erroneously would be detrimental.

Contents

Chapter 1

Introduction

The NASA Starlight and Breakthrough Starshot programs conceptualize fast interstellar travel via small relativistic spacecraft that are propelled by directed energy. This process is radically different from traditional space travel and trades large and slow spacecraft for small, fast, inexpensive, and fragile ones. The main goal of these wafer satellites (wafersats) is to gather useful images during their deep space journey. We introduce and solve some of the main problems that accompany this concept. First, we need an object detection system that can detect planets that we have never seen before, some containing features that we may not even know exist in the universe. Second, once we have images of exoplanets, we need a way to take these images and rank them by importance. Equipment fails and data rates are slow, thus we need a method to ensure that the most important images to humankind are the ones that are prioritized for data transfer. Finally, the energy on board is minimal and must be conserved and used sparingly. No exoplanet images should be missed, but using energy erroneously would be detrimental.

I begin by introducing the novel concept of simulators and their plausibility. Chapter 2 will introduce a broad concept overview of the problem and some beginning solutions. Chapter 3 will delve into the ability to analyze knowledge from human experiments and implant it into a machine learning model for use in deep space journeys. Chapter 4 will optimize the modeling process for highest accuracy given the special circumstances in our astrophysical application. Chapter 5 will solidify the use of advanced modeling processes by comparing our models to a more approachable and low-energy solution. Finally, Chapter 6 will upgrade the problem to a multi-object classification problem and analyze whether state-of-the-art categorizers are the best solution for our unique application.

Chapter 2

Advances in Deep Space Exploration via Simulators & Deep Learning

2.1 Introduction

Space travel, up until recently, was constrained by chemical propulsion, large spacecraft, and therefore, relatively slow speeds. Since the main objective has been exploration of our solar system, these methods were sufficient. In contrast, the recent Starlight program (Kulkarni et al., 2017) has introduced methods for deep space travel that utilize small discs, which travel at approximately one-fourth of the speed of light via directed energy. Alongside the prospect of fast deep space travel comes many new challenges. The normal model for space travel includes spacecraft capable of housing instruments, propulsion and navigational equipment, telescopes, energy banks, and much more. Since the Starlight program will be utilizing small wafersats that are approximately the size of a coffee can lid, all of these features need to be reworked or discarded. Besides physical constraints, this new model of space travel introduces feasibility constraints as well. The star of interest is beyond four light-years away, meaning that transmission of data and response command transmissions are a combined eight years or more. Thus, the wafersats need to be able to make decisions without human intervention, and for that, artificial intelligence (AI) is paramount. The major hurdles that we will discuss are those concerning computer vision via planetary detection, data and storage blockages via novelty detection and ranking, and energy management via combining simulator features with subtraction-algorithm-fed computer vision. For all of these issues, taking advantage of a universe simulator will introduce solutions that were otherwise ineffective or impossible to find.

2.1.1 Previous Work

The effectiveness of machine learning, specifically deep learning via TensorFlow and cuDNN, has been indisputably demonstrated in the last decade (Abadi et al. (2016), Chetlur et al. (2014), Canziani et al. (2016)). The fight over the best model and the most accurate results, especially between the most popular models like ResNets, DenseNets (Huang et al. (2018)), Inception(Szegedy et al. (2015)), Masks (He et al. (2018)), and models that combine some of these together(Szegedy et al. (2016)), is one that has produced a plethora of potent options to choose from. Models that are more accurate than human beings at doing extremely difficult tasks are still being discovered (Rajpurkar et al. (2017)). The areas of deep learning and astronomy have come together in recent years (Ruffio et al. (2018), Morad et al. (2018), Schaefer et al. (2018), Pearson et al. (2017)), mostly in the form of light curves (Shallue & Vanderburg (2017), Zucker et al. (2018), Carrasco-Davis et al. (2018)). The results and general concepts promote a healthy symbiosis between deep learning and the problems that arise in astronomy. Yet, the processes are carried out from Earth, not space, and do not address real images, two big issues that create a gap in comparability. Outside of astronomy, simulators have been used to train data in specific instances where the benefits outweigh the drawbacks. Smyth et al. (2018) outlines some major drawbacks, namely that the process takes a lot of time and knowledge, as well as a note that simulator-based training may not generalize well to real images. Alongside those concerns, McDuff et al. (2018) begins with a common issue in machine learning models, which is that training data sets are often biased. This bias arises when there are minorities in the training set, which in turn produces poor results when the model is asked to evaluate a similar entity in the population. These issues are handled throughout this paper and are shown to not be an issue with the specific problem at hand. Simulators also introduce a lot of benefits. One large one, also seen in Connor & Leeuwen (2018), is that "the small catalogue of real events is probably not yet a representative sample of the underlying .. population, nor is it big enough to build a meaningful training set for machine learning, deep or otherwise." An important theme throughout this paper, and an extremely useful aspect of simulators, is that they provide an untold amount of training data, assuming that one can create realistic simulations.

2.1.2 Unsupervised Learning for Planetary Detection

The intuition behind object detection, in particular planetary detection, might point toward an unsupervised learning technique. After all, one might reasonably think that detecting a nearby planet after months of traveling through deep space would be easy. We test this idea using an unsupervised technique called a *Grow When Required (GWR) Network* (Marsland et al., 2002).

GWR Setup

Using the worldwide telescope, we generated a 9,000 frame series of solar system images. It begins with Neptune, then it explores Mercury, the Sun, and finally Mars.

The majority of images contain only background stars.

The images were down-scaled to a 320x180 resolution in order to improve computational speed. For learning, they were decomposed into red, green, and blue channels and vectors were constructed of length $320 \times 180 \times 3 = 172,800$.

Our challenge is to label each image as novel or regular. That is, we wish to generate a classification n s.t. for each input x, $n(x) \in \{0, 1\}$, where a 0 indicates regularity and a 1 indicates novelty. Since the video is composed of 9,000 images, large objects like planets or the Sun will be in view for a few hundred or thousand consecutive frames. During these large bins when a planet or the Sun is clearly in view, the algorithm should hopefully yield a large number of 1's and should yield very few 1's when the image is mostly distant stars.

GWR Algorithm

Define A as the set of nodes in our network and C as the set of edges between these nodes. We denote our inputs as $\boldsymbol{\xi}$ and the weight vector for any node n as $\boldsymbol{w}_{\Gamma}30062$. Each node n has a habituation h_n which represents how familiar that node is to the system.

1. Initialize two nodes that represent two random samples from the dataset. We set their habituations each to 1. The set of edges between nodes begins empty.

2. Iterate through the dataset. For each data sample $\boldsymbol{\xi}$:

   (a) For each node i in the network, calculate its distance from $\boldsymbol{\xi}$, which is $||\boldsymbol{\xi} - \boldsymbol{w}_{\Gamma}30057||$.

   (b) Find the node s with the smallest distance and the node t with the second-smallest distance.

   (c) Add an edge between s and t if it does not already exist.

(d) Calculate the activity $a = \exp(-\sum_j (\boldsymbol{\xi}[j] - \boldsymbol{w}_s[j])^2/C)$, where $C = 29{,}203{,}200$ was chosen to prevent an integer overflow. There are 172,800 fields in each data vector, and since the average of the quantities in each vector is close to 13, and $13^2 = 169$, we divide by $172,800 \times 169 = 29,203,200$.

(e) If $a < a_T$ and s's habituation $h_s < h_T$ (where a_T is some insertion threshold and h_T is some habituation threshold), then add a new node r. If a new node is added, data point is considered novel. Set $\boldsymbol{w}_r = \frac{\boldsymbol{w}_s + \boldsymbol{\xi}}{2}$. Insert edges between s and r and r and t and remove the edge between s and t.

(f) Otherwise, update the weight and habituation of s as follows: $\Delta \boldsymbol{w}_s = \epsilon_b \times (\boldsymbol{\xi} - \boldsymbol{w}_s)$ and $\Delta h_s = \tau_b \times 1.05 \times (1 - h_s) - \tau_b$, where ϵ_b and τ_b are parameters. Next, update the weight and habituation of s's neighbors i as follows: $\Delta \boldsymbol{w}_i = \epsilon_n \times (\boldsymbol{\xi} - \boldsymbol{w}_i)$ and $\Delta h_i = \tau_n \times 1.05 \times (1 - h_i) - \tau_n$, where ϵ_n and τ_n are other parameters.

(g) Remove any nodes without any neighbors.

Our chosen values are: $a_T = 0.7, h_T = 0.1, \tau_b = 0.3, \tau_n = 0.1, \epsilon_b = 0.1$, and $\epsilon_n = 0.01$.

Results

Figure 2.1 is a scatter plot that was generated to visualize the novelty detected from the data. The x-axis is the id of each picture, and the y-axis is the number of novel images that were detected in each bin of 100 images. Figure 2.2 is a continuous representation of the same concept.

Moving along the Image ID axis, we see that novelty was detected in clumps around 0-500, 900, 4000-4500, 6000-6500, 7000-7500, 7700-8000, and 8200-8500. We observed that novelty was detected first on Neptune, then again on some particularly bright stars.

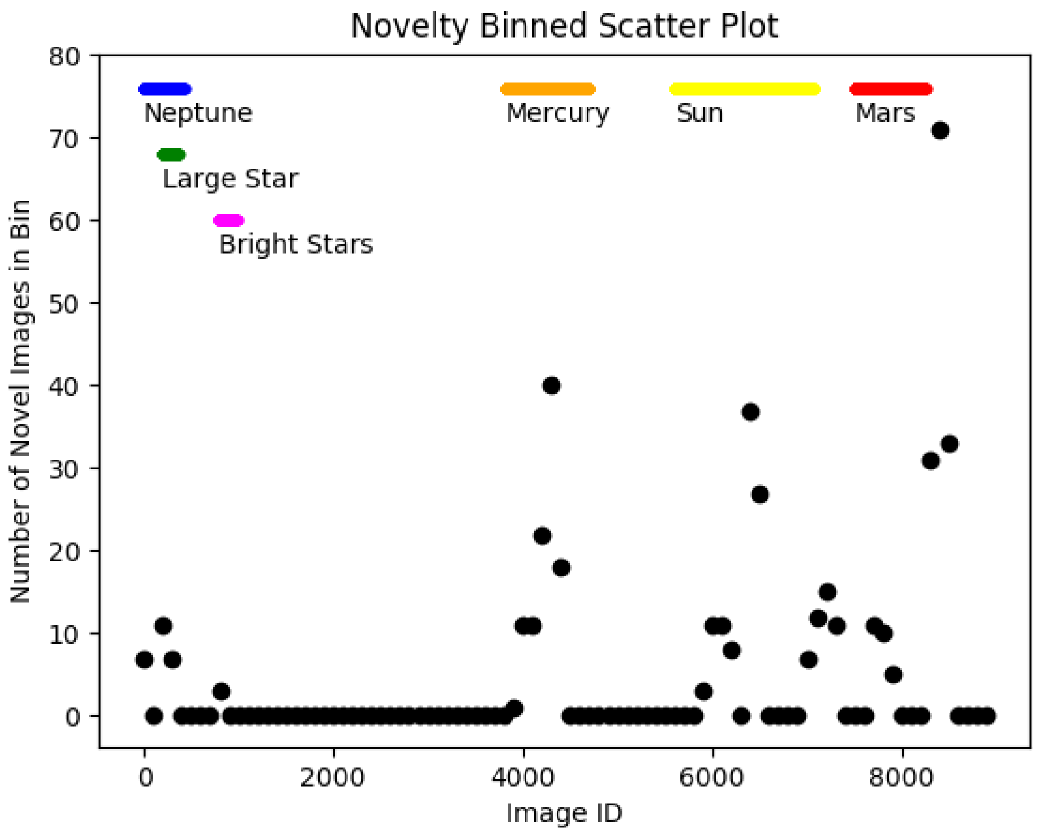

Figure 2.1: A scatter plot of the detected novelty of the data.

No novelty is detected during the long period of only stars. Next we see increased novelty detection when Mercury is plainly in view, and then when the sun appears, and finally when we zoom into Mars.

We notice that Neptune's collection of novelty is roughly one quarter the size of the other three celestial objects that come into view. We also notice a huge spike around image 8300. This is very interesting because there are no large celestial objects in view at this time.

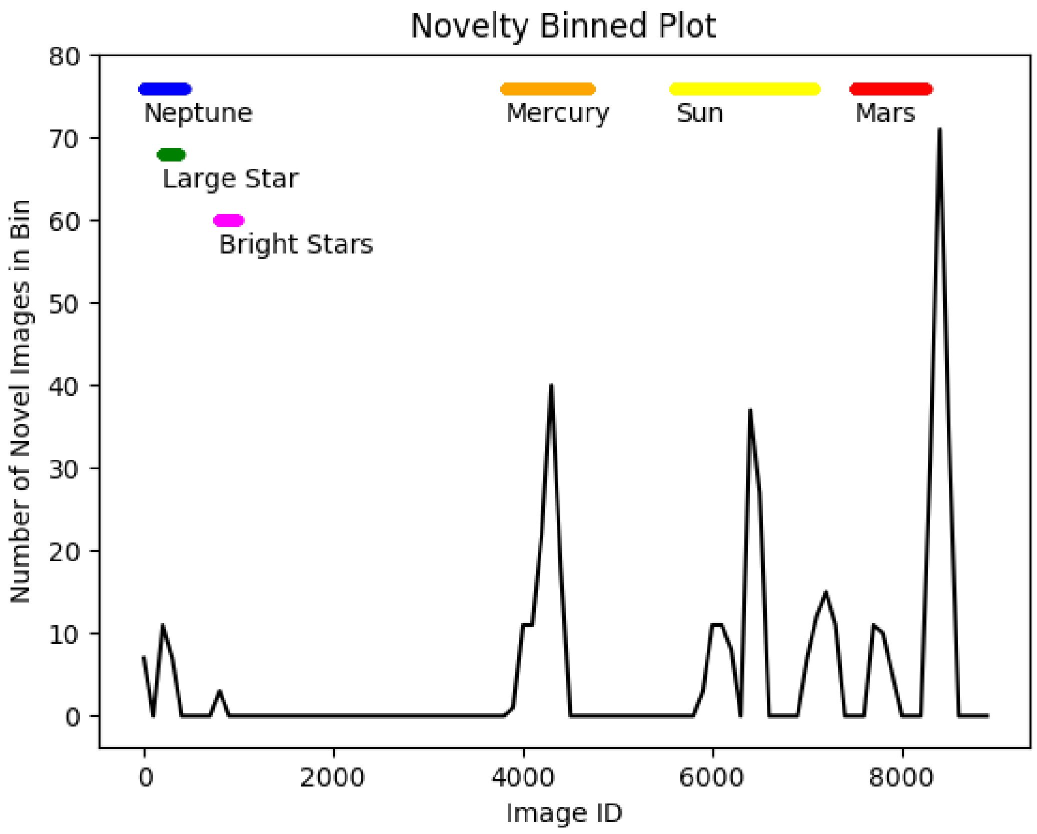

Figure 2.2: A binned scatter plot of the novelty of the data. Image ranges that are salient to the human eye are labeled on the plot.

GWR Discussion

A deep space exploration mission would come with many challenging objectives. A small but connected subset of those would involve detecting objects, deciding whether they are important, extracting key features that we would want to study or observe, and prioritizing their information retrieval.

GWR wouldn't be able to decide importance, extract features, or prioritize information retrieval, yet if it could detect novel objects in deep space, this would be useful. We can see from Figure 2.1 and Figure 2.2 that the detection is inconsistent and unreliable.

Neptune is almost completely missed and the three smaller peaks at the Sun, as seen in Figure 2.2, are larger than Neptune. The largest peak of all happens while Mars is minuscule and essentially not in view.

Although *GWR* had high novelty detection peaks while passing by Mercury and the Sun, it failed to correctly activate at Mars or Neptune. These observations, paired with its inability to do anything further with the data, introduce a need for a more advanced model that can achieve all of the above objectives.

2.1.3 Object Detection vs. Novelty Detection

Throughout this paper, our main goals will constantly be alluding to object detection and novelty detection. In a general computer science setting, object detection is used to identify something in an image that has already been trained via some algorithm. For example, we may feed thousands of images of human beings into a YOLO algorithm, and then once it is trained, we can walk the streets of New York and see if our algorithm can identify human beings. In this setting, identifying a human being is a success, and not identifying a car or stop sign as a human being would also be a success. Yet, identifying anything non-human as a human being would be a failure. The accuracy of a model, which is mathematically computed per identification, can be used as a measure of how sure the algorithm is that the object being identified is the correct type. In this paper, we will delve into why this is difficult for our specific scenario, and we will test whether this can benefit severely from the use of simulators.

On the other hand, novelty detection is used to attempt to identify something that has never been seen before. One powerful example is self-driving cars being able to see traffic signs that are unique to a certain country, and therefore have never been seen or used during the training process (Kim et al., 2017). In this example, the self-driving car algorithm has never seen this specific sign before, and so identifying it without any training data is very difficult. In our paper, unseen planetary features are analogous to the unseen traffic sign in the example, and we delve into methods of solving this via simulators.

2.1.4 Simulator Options

Graphical options in the simulator are abundant, which allow for complete control of the simulated universe. In general, the feature set should be optimally set for realism while traveling through space, but the ability to tweak these options speaks to greater breadth for learning and adapting to unique situations that may arise in space. For example, the image of an exoplanet while traveling at one-fourth of the speed of light with a nebula in the background is a completely new concept. Yet, two features in the simulator may be able to deal with that combination. First, the ability to toggle lens flares will provide the AI with training images that both contain and lack lens flares. Second, a feature called overbright can drastically adjust how bright the background stars and nebula appear. Training on images that embrace the entire spectrum of overbright will allow this machine to deal with novelty detection in a very advanced manner. Having a plethora of options to enable and tweak can introduce a much larger set of training images and will let the AI absorb more information before embarking into deep space.

Some other important options, besides those that deal with graphics and rendering, are diffraction spike intricacy and size, lens effect on stars, and planetary shine. Altering all of these settings and training on the resulting images enables the capture of more information.

For these experiments, settings were chosen that were not too extreme in any direction. Stars were kept as they randomly formed in order to have natural deviation

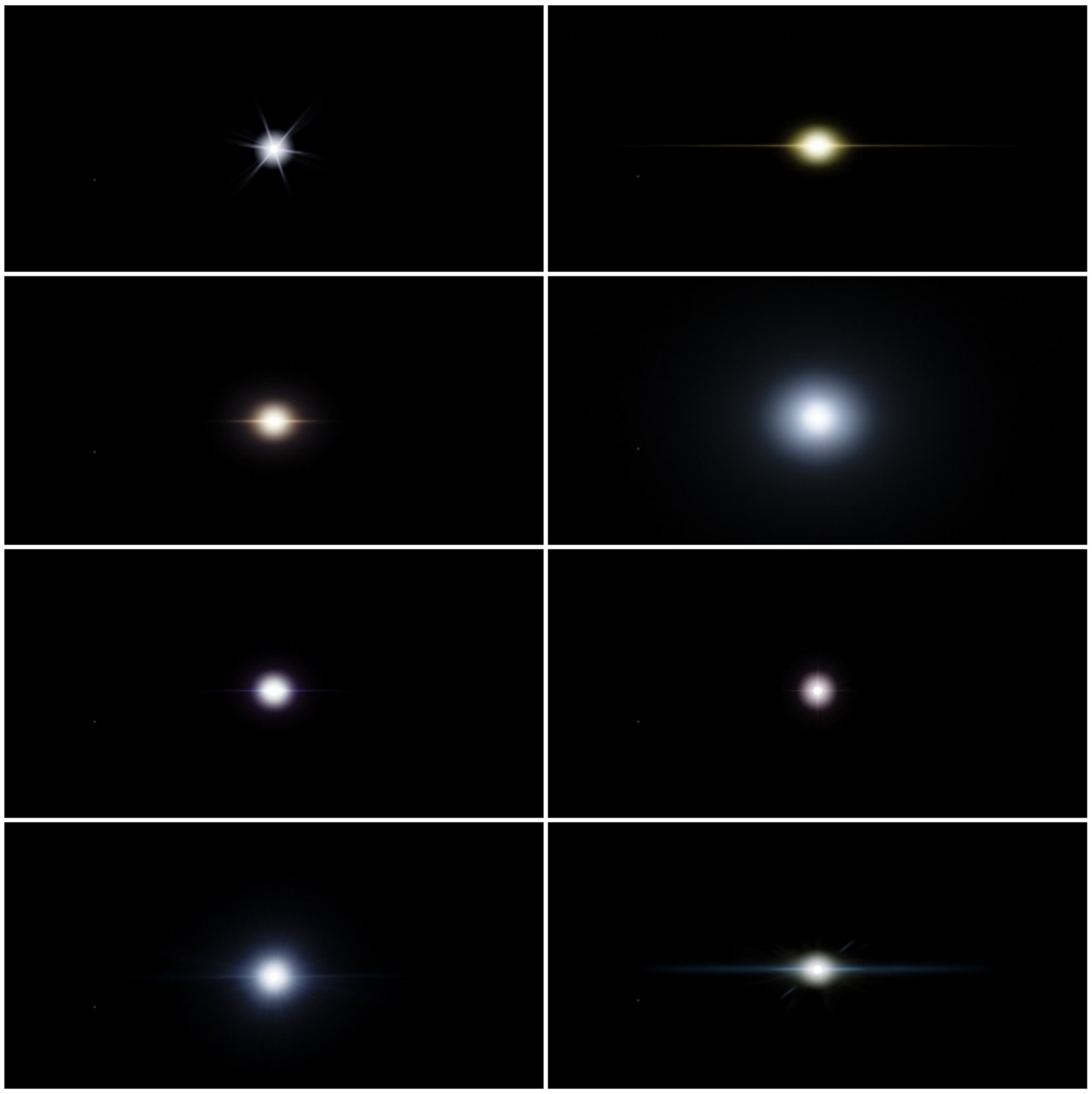

Figure 2.3: Eight lens options applied to the same star

in the image background. All quality parameters were turned to the highest setting to provide realism and pixel definition. The following table outlines all display and graphics settings.

Settings	Value
Resolution	3840x2160
Projection	Perspective
Auto Exposure	Simple
Bloom	0.600
Aurora Quality	High
Black Hole Quality	High
Ship Warp Quality	High
Skybox/Impostors/FB	Enabled
Diffraction Spikes	Normal
Diffraction Spikes Size	0.1
Lens	SE 0.95 Single
Point Sources	Sprites
Scale	0.900
Overbright	4.467
Desaturate Dim Stars	0.1995
Landscape LOD	-0.6
Planet Shine	Super
Planet Shine Bright	2.512
Thermal Emission Shift	0.05
Real Sun Brightness	Disabled
Real Planet Brightness	Disabled

Table 2.1: Table of simulator settings used throughout the training process.

2.1.5 Overall Simulator Importance

In this paper, we consider three main areas of deep space travel that can be drastically improved with the use of a simulator.

First, computer vision is an extremely useful tool for detecting objects and making decisions based on what is seen. The training process consists of tagging images and providing a label for each tag, feeding those images and tags into a model, and having that model learn the associations. The model can then be given images, and based on how successfully it was trained, it may be able to identify parts of the image.

Since we have never photographed exoplanets in detail, training a model using real images is not feasible. Therefore, we rely on training using images of planets that we have photographed, which would be those in our own solar system. Yet, detailed photographs of planets are not very abundant and would only teach the model to look for those specific features. In realizing that this would not be sufficient, we may move toward novelty detection, a branch of computer vision that tries to classify data that deviates from the data used during training. Co-domain embedding (Kim et al., 2017) has proven useful in some situations, such as those where a template design would resemble a real image almost exactly, but planetary features do not translate well to use in novelty detection. This is because planetary features, such as atmospheric patterns, are extremely unique.

Simulators can provide very detailed and randomly generated images of planets that obey universal physical laws. Therefore, we will be able to generate countless images of planets that resemble real images of possible exoplanets. Training on these images and features, the model will learn an exorbitant amount of information. While traveling through space and faced with an image of a real exoplanet, the model will now have a much broader knowledge base.

Second, we introduce the notion of novelty ranking. A major hurdle in deep space wafersat travel is data storage and transmission. On-board memory is limited by physical constraints of the wafersat and astrophysical exposure, while transmitting data from a wafersat to a communications hub would be slow and dependent on energy reserves.

A system that can deal with this issue is one that prioritizes the most important on-board data and sends that first. This not only ensures that the critical images are sent in descending order of importance in case of some malfunction, but that the most relevant data is quickly known for the next wafersats in line.

With the overarching goal being the identification and transmission of the most important data, novelty ranking will quantify the on-board images based on importance. Simulators will provide the breadth of planetary features that are needed to find out what *importance* means, as perceived by humans, and then this information can be applied to software.

Third and last, sending a small disc into deep space means that on-board energy reserves will be very small. Yet, the objective of detecting and imaging astronomical bodies while traveling must still be met.

2.2 Simulator for Planetary Detection

Our main objective here is to identify novel planets while traveling through deep space. In order to do so, and for subsequent sections, we will require a basic conceptual understanding of object detection in order to logically progress. We should point out that the main backbone of object detection, through a few core processes, is the same as that used by humans when they naturally process information and identify objects. We will discuss these fundamental core processes.

First, the object that we will try to have the model identify should be seen beforehand in order to train the model. Unsupervised techniques have their uses and do not require this training process, but we will only deal with supervised learning models from here on out. Mainly, this is done because planetary detection is the simplest task, so we need a model that can adapt afterwards in order to successfully identify planetary features and rank novelty.

Second, the model will become more robust with more images. Of course there are exceptions here, such as feeding poor images or images that do not match the objects category. We will test this concept thoroughly while we also test the importance of simulated images.

2.2.1 Setup

Here we will discuss the details of our model, our hypothesis, and how we will go about testing the importance of simulator images. Our main goals when choosing a model are finding one that has high accuracy, low to medium computation time, and has been tested to be a reliable model. Because of this, no new models that haven't had time to be tested thoroughly throughout the computer vision community will be used. Also, the ideal model will sacrifice computation time for accuracy, if needed.

Our hypothesis stems from our second core process and states that simulator images will not decrease accuracy for planetary detection and planetary features. These two processes, the detection of planets while traveling through space, as well as the detection and recognition of features on those planets, are the inspiration for the two main experiments that are set up. Currently, our collection of useful astronomical images is very limited. Therefore, using only real images of planets would limit us to those found in our solar system. Also, planetary features would suffer since our solar system contains very few features out of the set of total planetary feature combinations.

The first experiment will test the validity of simulator images in general. It is set up in three different stages using the same object detection framework and always testing on the set of real images of Jupiter. First, we will train on real images of every planet in our solar system except for Jupiter. These images will be collected from NASA image repositories and will not include composite images, artist renditions, or any other variations except for true unaltered images. Second, we will train on only simulator images with the goal of solidifying whether simulator images alone are useful in detecting real planets. Third, we will combine the first and second training sets, comprised of simulator images and all real images (excluding Jupiter),to determine whether simulator images and real images together provide the best of both worlds.

The second experiment will introduce and test an extremely important feature of using simulator images - the ability to detect novel planetary features, i.e. those which have never been seen in any real images. Since simulators can be programmed to emulate real physics, the outcome can give us an extremely large number of realistic looking planets with features that have never been observed. In order to proceed, we need to use a planetary feature that exists in our solar system so that we can train using simulator images and test using real images. Planetary rings have a solid theoretical foundation and would easily appear in any physics-based simulator, while also being present around Saturn. Rings are also fairly complex, as they contain extremely unique striation patterns, can look wildly different depending on the viewing angle, and can even co-exist with other rings around the same host planet.

2.2.2 The Model

When using deep learning models for a specific purpose, it is imperative that a model is chosen that optimizes what it can while prioritizing what it must. For example, an object detection model that may be implemented on a smart phone for real-time detection of human faces might prioritize speed and give up a small amount of accuracy.

For our purposes, a model should be able to identify planets in deep space and features on that planet. As seen in Section 1.2.3, this task is not trivial. Experiment #1 will utilize the model for identifying the existence of a planet *with a strong accuracy.* Experiment #2 will utilize the model for identifying novel features, either on planets or near planets, such as rings. This model does not incorporate bodies other than planets during these experiments.

For our purposes, accuracy is of the utmost importance, while operation count is also of some importance. In deep space, we have plenty of time to do calculations, but we also have very little energy. Therefore, reaching maximum accuracy with a small amount of operations is the ideal scenario.

As we can see from Figure 2.4, originally presented in Canziani et al. (2016) whereas the authors compared many models for practical applications like this one, there are few models that fit into the optimal space of accuracy and operations. The main choice was ResNet architecture vs. Inception architecture. The accuracy and operations for both models are almost identical, yet the residual neural network (ResNet) architecture provides a shortcut in case the training phase introduces the vanishing gradient problem (He et al., 2015). Along with this feature, ResNet is a very established model in many domains, and for these reasons, will be our model of choice going forward.

In terms of ResNet choices, Resnet-34 and ResNet-50 do not provide enough accuracy for this setting. The best options are ResNet-101 and ResNet-152, and while both provide approximately the same level of accuracy, ResNet-152 is much more computationally taxing. Therefore, ResNet-101 is the optimal blend of accuracy and operations, utilizing 101 layers.

Improvements to base ResNet-101 have been achieved via Fast R-CNN (Girshick, 2015) and then Faster R-CNN (Ren et al., 2016). The final model that we use is a hybrid

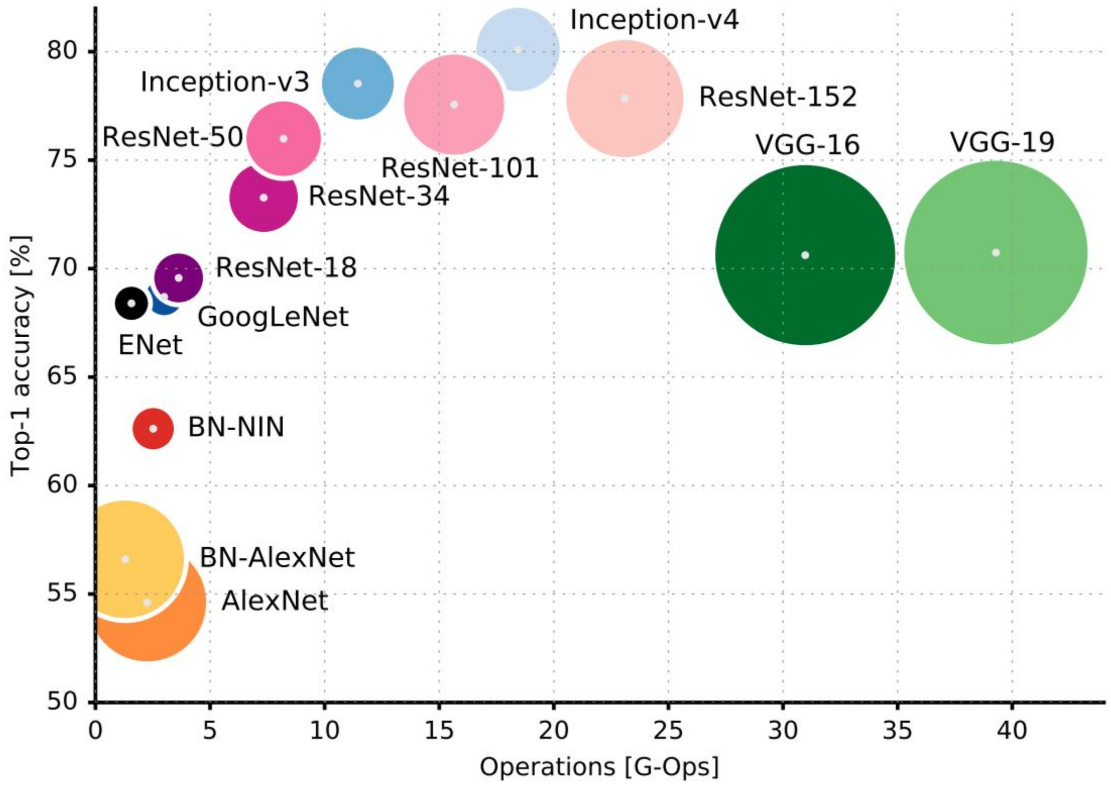

Figure 2.4: Results shown in Canziani et al.(2016) that compare model accuracy vs. operation count during the ImageNet challenge.

that incorporates the advanced methods of Faster R-CNN with the strong foundations of ResNet-101, aptly named *Faster R-CNN with ResNet-101*.

During our experiments, this model was run on a GTX 1080 Ti. In order to achieve 60,000 iterations, the runtime was approximately 24-30 hours. During the testing phase, each image was evaluated in approximately 1-2 seconds. Again, this was using a GTX 1080 Ti, which is presumably much more powerful than any technology that will be used to evaluate images on board the wafersat. *Faster R-CNN with ResNet-101* is used throughout the paper, no model changes occur.

2.2.3 Experiment #1 - Planetary Detection

In this experiment, we tested the theory that simulator images could be used to train a model that could then detect real objects, and in particular, planets. Our hypothesis is that simulator images are at least as good as real images in terms of information gain during training. Although simulator images can be produced in bulk, the idea was to test the theory using similar image count in order to avoid any bias. The table below shows the number of images used for each model and for their testing phase.

Real	Sim	Real+Sim
120	122	242

Table 2.2: Experiment #1 training image count for real images, simulated images, and the combination of both real and simulated images.

Concerning the testing images, the images were broken down into 2 sections. The first section was comprised of independent images taken at differing angles. The second section was the exact same frame of reference, including angle and distance, but included a time-lapsed series of images.

The results of the testing phase produced a detection score. This score is a built-in function of TensorFlow and represents the mathematical certainty that the model has identified the correct object. The table below shows the final detection score results.

The table shows quite a few compelling results right off the bat. The most direct

Section	Real	Sim	Real+Sim
1	**99.875**	99.375	**99.875**
2	98.889	**100**	**100**
Total	99.353	99.706	**99.941**

Table 2.3: Experiment #1 detection Score for real images, simulated images, and the combination of both real and simulated images.

one being that *Real+Sim* has achieved equal or better results than *Real* or *Sim* alone did in all categories. Besides *Real+Sim*, we can also say something about *Real* vs. *Sim*. Although *Real* achieved slightly better scores in Section 1, *Sim* not only achieved better scores in Section 2, the total score of *Sim* was also higher and *Sim* contained at least one section that had perfect scores.

Our initial hypothesis was that using simulator images would be as good as real images. Different models and training image sets will always produce different results, but considering the total scores with our two sections, *Sim* produced equal or better results when compared to *Real*.

2.2.4 Experiment #2 - Novel Features

In order to be able to show the importance of using a simulator for novel planetary features, we use the results from Experiment #1 as proof of concept. Those show that *Sim* images do not decrease detection scores on real testing images, with the added benefit of being able to mass produce them and customize feature information in each image.

With this in mind, Experiment #2 will gather 65 simulator images of ringed planets and train a new model with the same framework as Experiment #1. We will then test novel feature detection on real images of Saturn. The machine will have never seen any real image and will have never been exposed to prior knowledge of Saturn or our solar system at all. This experiment is, in theory, identical to training a model with simulator images on Earth and sending it out into deep space in order to identify novel, never-before-seen features found on real exoplanets and in real images.

One of the main benefits of simulator images can be observed here. Even a planetary feature that we can observe will be found once, or perhaps a few times at best. Therefore,

Type	Accuracy	Detection Errors
Planet	99.22	None
Rings	99.11	None

Table 2.4: Experiment #2 detection score results.

we have limited variability to work with in terms of ring structure, width, pattern, count, etc.. Yet, if this experiment produces promising results, we can simply build a physics-based simulator that generates planets, filter by the presence of rings, capture an image, and repeat the process any amount of times. From Experiment #1, we know that training on these simulator images will provide approximately-equivalent information gain when compared to real images of ringed planets. Since our simulator is physics-based, it should produce many features that we have not even seen before, transforming this problem from novelty detection into object detection.

The experiment was set up in parallel to what would hypothetically happen during deep space exploration. The training was done on a small batch of simulated images of ringed planets. The idea in the experiment is that, in theory, we have never seen a ringed planet before. Yet, our physics-based models of planet formation dictate that they would naturally occur. So we collect simulated images, train on that, and then send it deep into space. Upon finding a ringed planet for the first time, it would need to recognize those planets. Normally, we wouldn't be able to do this since we have no ringed planets to train on (in this hypothetical experiment), but since we used simulated images, we now have a model to deal with this. The experiment goes through this entire process, and even tests the model on real images of Saturn. Again, the machine has only seen a small batch of randomly generated simulated images of hypothetical ringed planets, never a real image of a planet. The results of the experiment are extremely positive and can be seen in the table below.

As we can see, the model is dependably accurate based on solely simulated images. This experiment shows a key point of using simulators - by combining planet generation theory and realistic rendering, we have turned a novelty detection problem into an object detection problem, which is significantly easier to deal with. Now, instead of having to detect unknown features, we can simply construct planets randomly based on physical laws and train a model using those simulator images. This would introduce a more complex model with multiple classifications, increasing its complexity, size, and run-time.

2.3 The Simulator

Although there are quite a few universe simulators available today. Here, we utilized SpaceEngine (SpaceEngine.org) for its realism, expansive set of options and customizations, and unique informational tools.

2.3.1 Simulator Features

One of the best features of the simulator is its extremely realistic rendering capabilities. In combination with a 3840x2160 4K monitor and GTX 1080 Ti, the simulator produces extremely detailed and realistic images.

The simulator also includes the ability to edit any planet, so that instantly rendering an exoplanet with a very particular feature set is simple. Alongside image features are astronomical features, which are tracked and shown for every body in the universe. Some of these features are type, class, orbital period and mass, but most importantly, distance.

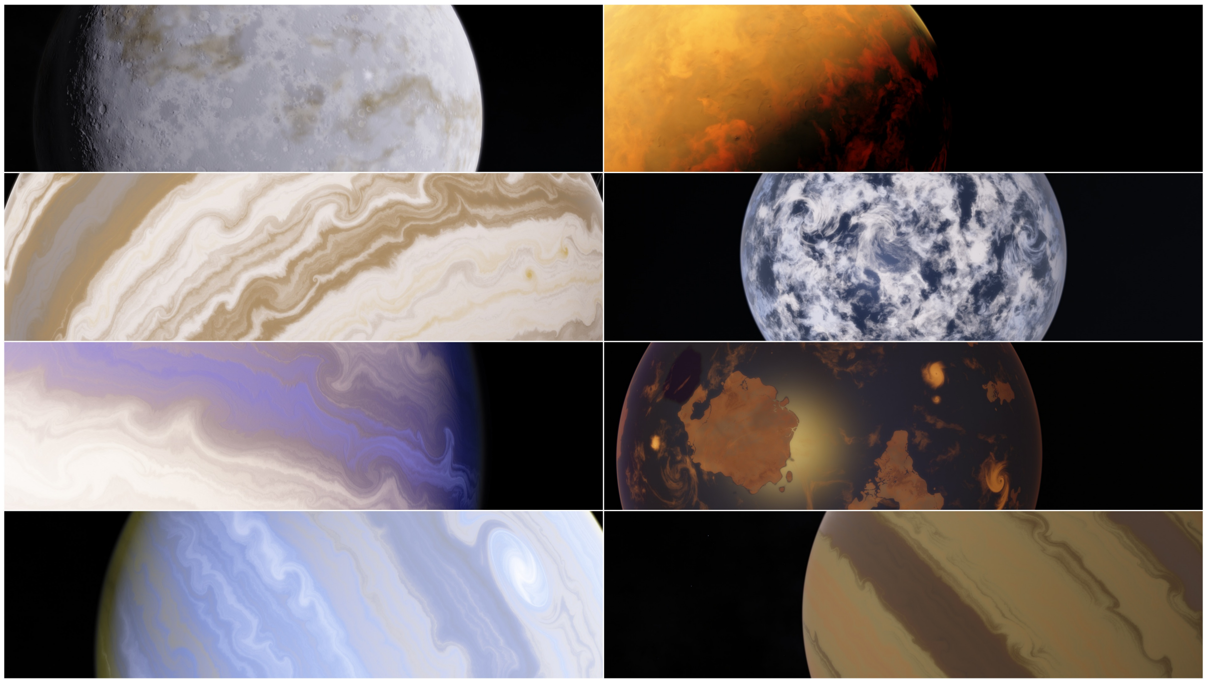

Figure 2.5: Examples of 3D-rendered randomly generated exoplanets

2.4 Foundations for Future Work

2.4.1 Simulator for Novelty Ranking

We have shown previously that simulator images can be used with astounding accuracy, and with mass production, can make training via real images unnecessary. Therefore, we can train using hundreds or thousands of simulated images and when we encounter a planet, we can detect it, image it, and send those images back to Earth.

Say, for instance, that the wafer passes and images the five planets in a hypothetical solar system. Soon after that, it may be on an inevitable course toward that solar system's star, which will destroy the wafer and all of the images. One downside to small wafers is that they are easily destroyed or corrupted. This makes a priority system vitally important, as it would allow the wafer to possibly send back one or two images from the five that it collected before it is destroyed. This section is dedicated to figuring out which images should be sent back, and discussing the approach in doing so.

The Concept

We will assume that we have a small storage of images that we need to send back to Earth in an order that is based on importance. Wafers could be destroyed relatively easily and data transmission rates in space are very slow, so sending data based on a notion of importance is paramount.

Figure 2.6 helps show us the extremely abstract definition of importance that we, as humans, may place on new planets. The top planet is colorful and full of land and different bodies of liquid, while the bottom planet has a unique double ring, an atmosphere, and a single large ocean, one that may be assumed to be water by visual inspection alone. The main question we want to ask here is: If you could only send one of these images back to humanity, which would you send?

An astrobiologist might choose the top planet since the presence of land and many differently-composed bodies of liquid exist, giving multiple opportunities for life to possibly flourish. Yet, someone interested in another planet that may be able to accommodate human existence might choose the bottom planet since it seems to offer two important features for us, water and an atmosphere. The question of importance to humans is very subjective, yet we need a solution that would be able to rank these two planets, and many more, in order of importance.

The Human Experiment

The implausibility of teaching vast conceptual knowledge to a machine in hopes of it gaining context made us seek out a different approach:

1. Generate simulated images of planets that range in features. This will remove the

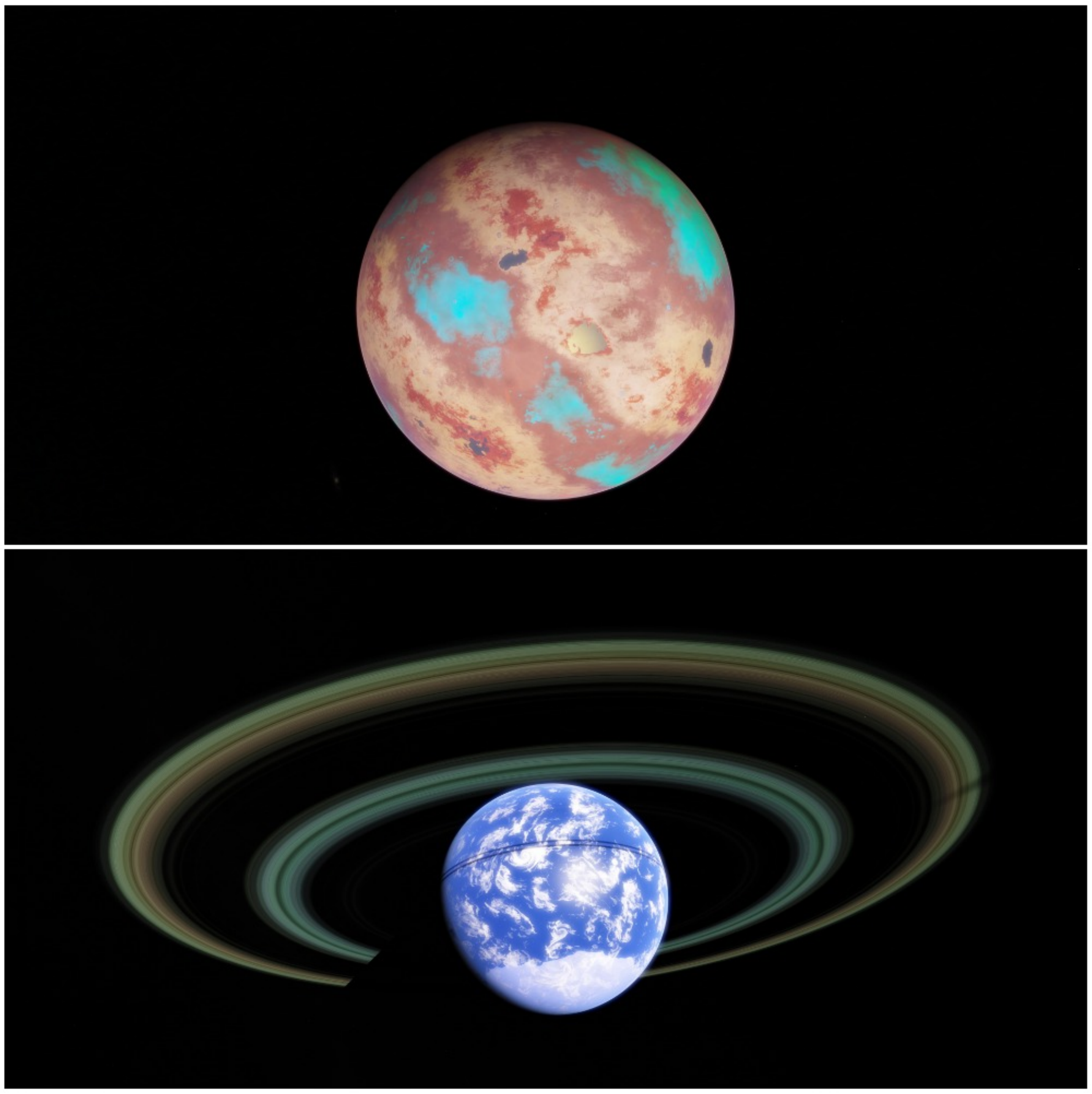
Figure 2.6: Hypothetical storage of two images that need to be ranked based on importance.

bias that some people may have about our own solar system, since we are not using any real images of our own planets. It will also allow data to be gathered about features that do not currently exist in our solar system, but based on astrophysical theory, could exist in the universe.

2. Ask experimental subjects to rate each planet by *Importance*. This is posed via the question: "On a scale from 1-7, how important would it be for humankind to see this image if it were gathered by a spacecraft during deep space exploration?"

3. Ask experimental subjects to rate each planetary feature. This will comprise our total planetary feature set. For instance: On a scale from 1-7, how much does this planet exhibit the presence.. of rings? ..of an atmosphere? ..of moons? ..of a livable environment for humans?

4. Using the data gathered from human thought processes and individual analysis of importance and interestingness of a planet, train a model to predict importance given a feature set.

5. Rank all planets in storage based on importance and send them back to Earth via this priority system.

Essentially, the process would generate a plethora of novel planets via the simulator. Subjects would participate in the experiment, and in doing so, would give us vital information about each planets perceived features and the planets overall importance. This information would then be fed into a machine learning algorithm, with the output being importance and the input being feature strength. The wafer would then see a planet, extract information about each feature via a model like that seen in Experiment #1, plug it into the machine learning model, and from there, it would have an accurate prediction of importance.

This process takes in human definitions and thought processes in order to break down the concept of what we find interesting in planets that we have not even seen before. Using this method, we can bypass a problem of novelty detection, which is difficult, and machine contextual learning, which is extremely difficult, and turn it into a problem of human information gain via experiments and subject participation, which is easy, and object detection, which is also easy.

Future work will show experimental results beyond this proof of concept solution.

2.4.2 Simulator for Energy Management

During a deep space voyage, the wafersat will need to be supplied with enough energy to perform necessary functions, such as imaging, analyzing the images, and transmitting data. We don't assume that the system is perfect, nor do we need certain restrictions on the amount of energy available. We have one goal: minimizing the amount of energy needed while ensuring planetary detection. At one end of the spectrum is full energy conservation, which would mean that the camera never turns on and therefore we never collect any data. On the other end of the spectrum is full energy use, meaning that the camera never turns off until the energy runs out, which would yield us many images but most likely none with important findings. Somewhere in between is optimal, but how do we find it?

The Two Phases

Simulators open a whole new universe that can be utilized in order to make a virtual interstellar journey to Alpha Centauri hundreds of times in the span of a day. By doing this process, we can train our models to identify stars, predict distances, swap between the two possible phases, and in doing so, save energy while capturing meaningful images.

Phase One is essentially comprised of time spent in open-space travel. This would mean that the probe is beyond a 'fair' distance away from any nearby star and that planetary detection would be a fruitless endeavor. Yet, during this phase, the main objectives would be nearby star detection and star distance predictions.

Phase Two would be a rare occurrence whereas the probe has traveled within a 'fair' distance of a star and we no longer need to deal with nearby stars until we have left that star's system. Instead, this phase would prioritize planetary detection, imaging, and ranking.

The Process

The trip from Earth to Alpha Centauri can be done in approximately 20 years. But, in the simulator, one can travel at any speed and cut out the majority of the time spent in an uneventful space. This makes it possible to simulate a 20 year journey in a few hours, or many journeys in just a single day.

Once these are done, we can train a machine learning model using star type, the section of the image containing the star, and the distance from the probe to the star (a simulator feature). Combined with a subtraction algorithm, and only using enough energy to take two images, the machine will be able to identify stars and predict their distance from the probe.

Using this information, the probe will know the approximate distance to the nearest star in its forward path. A simple calculation can tell it a safe amount of time to wait until it should take two more images, confident that the time it has waited has been uneventful.

Repeating this process is extremely energy efficient, and should eventually lead to coming within a reasonably 'fair' distance from a star. When this occurs, we would change into Phase Two.

Phase Two would use the same intuition except that instead of stars, we substitute in planets. Once identified, instead of being interested in distances, we would prioritize imaging. Details on planetary detection, imaging, feature extraction, and ranking have been detailed in earlier sections.

An example of Phase One

One extremely difficult concept in this entire process is making sure that the probe can successfully understand what is close to it versus what is very far away. The concept used is straight-forward: bodies that are closer will tend to shift more while the probe travels in a straight path. As an extreme example, a body that is 1 AU away from the probe will shift from center screen to completely off screen in approximately 33 seconds. Yet, a very distant star could go without changing position for months or years.

In order to deal with this, a subtraction algorithm is implemented. The probe will take a photo, wait a certain amount of time, and take another photo. Then, the first will be subtracted from the second and the resulting image will show any pixels that have shifted state during the elapsed time. If enough of these pixels shift, we will get a clear image of something that is relatively close.

The main problem here, again, is that nobody has any concept of the "wait a certain amount of time" part of the process. How much time is the right amount of time? If you do not wait long enough, nothing will move and your subtracted image will be all black. If you wait too long, even things that are very far away will begin to shift and you will be left with a large amount of stars, still unsure about which of those are actually close. This difficult part becomes approachable with the use of simulators.

The example deals with a simulated star that exists 0.08 light-years away from the probe. We travel at 500c and perform a subtraction algorithm in 10 second intervals, resetting after each one. This equates to traveling at 0.25c and performing a subtraction algorithm every 20,000 seconds, or approximately every 5.55 hours. So, the first image is 5.55 hours in real-time, the second image is 11.11 hours in real-time, then 16.66 hours, and so on. The goal is to see if simulators can be useful, and if so, at what point we would want to optimally take images in order to ensure we capture bodies that are nearby while also saving energy.

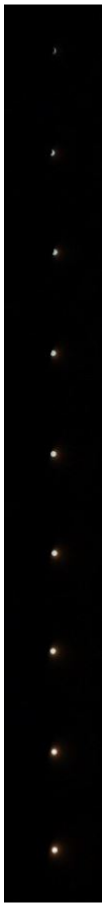

Figure 2.7: Subtraction algorithm performed in 20,000 second intervals

As we can see in Figure 2.7, the first few bins produce a hazy image of the target star. At the fifth image, which would have the probe waiting approximately 28 hours between images, we can see a full image of the star. By the last image, which is represented by approximately 50 hours of real time, other nearby stars were showing very hazy signs of recognition from the subtraction algorithm.

This proof of concept is extremely vital to star recognition and energy management. Depending how far away from a star we want the probe to be when it is able to recognize it, this process can be altered and honed easily.

2.5 Citations

References

Abadi, M., Agarwal, A., Barham, P., et al. 2016, TensorFlow:large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467v2 [cs.DC].

Canziani, A., Paszke, A., & Culurciello, E. 2016, An analysis of deep neural network models for practical applications, arXiv:1605.07678v4 [cs.CV].

Carrasco-Davis, R., Cabrera-Vives, G., Förster, F., et al. 2018, Deep learning for image sequence classification of astronomical events, arXiv:1807.03869v3 [astro-ph.IM].

Chetlur, S., Woolley, C., Vandermersch, P., et al. 2014, cuDNN: efficient primitives for deep learning, arXiv:1410.0759v3 [cs.NE].

Connor, L. & Leeuwen, J. van 2018, Applying deep learning to fast radio burst classification, AJ, 156, 256.

Girshick, R. 2015, Fast R-CNN, arXiv:1504.08083v2 [cs.CV].

He, K., Gkioxari, G., Dollár,P., Girshick, R. 2018, Mask R-CNN, arXiv:1703.06870v3 [cs.CV].

He, K., Zhang, X., Ren, S., Sun, J. 2015, Deep Residual Learning for Image Recognition, arXiv:1512.03385v1 [cs.CV].

Huang, G., Liu, Z., Maaten, L.van der, Weinberger, K.Q. 2018, Densely connected convolutional networks, arXiv:1608.06993v5 [cs.CV].

Kim, J., Lee, S., Oh, T.-H., & Kweon, I.S. 2017, Co-domain embedding using deep quadruplet networks for unseen traffic sign recognition, arXiv:1712.01907 [cs.CV].

Kulkarni, N., Lubin, P., & Zhang, Q. 2017, Relativistic spacecraft propelled by directed energy, arXiv:1710.10732 [astro-ph.IM].

Marsland, S., Shapiro, J. & Nehmzow, U. 2002, A self-organising network that grows when required, Neural Networks, 15, Issue 8-9, 1041-1058.

McDuff, D., Cheng, R., & Kapoor, A. 2018, Identifying bias in AI using simulation, arXiv:1810.004171 [cs.LG].

Morad, S., Nallapu, R.T., Kalita, H., et al. 2018, On-Orbit smart camera system to observe illuminated and unilluminated space objects, arXiv: 1809.02042 [cs.CV].

Pearson, K.A., Palafox, L., & Griffith, C.A. 2017, Searching for exoplanets using artificial intelligence, arXiv:1706.04319v2 [astro-ph.IM].

Rajpurkar, R., Irvin, J., Zhu, K., et al. 2017, CheXNet: radiologist-level pneumonia detection on chest x-rays with deep learning, arXiv:1711.05225v3 [cs.CV].

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A 2016, You only look once: Unifided, Real-Time Object Detection, arXiv:1506.02640 [cs.CV].

Ren, S., He, K., Girshick, R., Sun, J. 2016, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, arXiv:1506.01497 [cs.CV].

Ruffio, J.-B., Mawet, D., Czekala, I., et al. 2018, A bayesian framework for exoplanet direct detection and non-detection, AJ, 156, 196.

Schaefer, C., Geiger, M., Kuntzer, T., & Kneib, J.-P. 2018, Deep convolutional neural networks as strong gravitational lens detectors, A&A, 611, A2 (2018).

Shallue, C.J., & Vanderburg, A. 2017, Identifying exoplanets with deep learning: a five planet resonant chain around Kepler-80 and an eight planet around Kepler-90, AJ, 155, 94.

Smyth, D.L., Glavin, F.G., & Madden, M.G. 2018, Using a game engine to simulate critical incidents and data collection by autonomous drones, IEEE Games, Entertainment, Media Conference (GEM), Galway, 2018, pp. 1-9.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. 2016, Inception-v4, Inception-ResNet and the impact of residual connections on learning, arXiv:1602.07261v2 [cs.CV].

Szegedy, C., Vanhoucke, V., Ioffe, S., et al. 2015, Rethinking the inception architecture for computer vision, arXiv:1512.00567v3 [cs.CV].

Zucker, S., Giryes, R. 2018, Shallow transits - deep learning I: feasibility study of deep learning to detect periodic transits of exoplanets, AJ, 155, 147.

Chapter 3

A Multidimensional Thurstone-Coombs Model of Simultaneous Sensory and Liking Ratings

3.1 Introduction

Hedonic responses about a novel object are often based on the sensory characteristics of that object. Is the color pleasing or not? Does the curry have the right amount of heat? A popular model of such responses, called the unfolding model, was proposed more than 50 years ago by Coombs (1964). The unfolding model assumes that when judging one's hedonic responses to a set of stimuli, the observer imagines his or her ideal stimulus and then compares each stimulus in the set to this imagined ideal. The stimuli are then ordered by preference according to their similarity to the ideal. The unfolding model has been generalized in a variety of different ways (e.g., Borg (2018); DeSarbo & Rao

(1984); De Soete et al. (1986); Schönemann & Wang (1972)), and applied successfully in a wide variety of different domains (e.g., Andrich (1989); Davison (1979); DeSarbo et al. (1997)).

The unfolding model provides an accurate account of preference orderings, but it is less successful at identifying the sensory characteristics associated with the ideal stimulus. Some multidimensional versions of the model produce a multidimensional scaling (MDS) solution that situates each of the stimuli and the hypothetical ideal as a single point in a multidimensional space (e.g., De Soete et al. (1986)). However, as in traditional MDS, no information is provided about the nature of these dimensions. Sometimes, by noting which stimuli are situated at one extreme on a dimension and which stimuli are situated at the other extreme, it is possible to speculate about the nature of one or more dimensions. But this process is not always successful, and whatever inferences are made are impossible to test.

One popular experimental method for estimating the sensory characteristics of a stimulus is the concurrent ratings task, in which participants rate each stimulus simultaneously on a number of sensory dimensions Hirsch et al. (1982); Olzak (1986). The observed ratings are then used to estimate the participant's sensory, perceptual, or cognitive impressions of the stimulus. This method is a multivariate generalization of the rating experiment that is a popular signal-detection theory method for estimating an ROC curve Green & Swets (1966); Macmillan & Creelman (2005). As in signal-detection theory, the resulting data often can be modeled accurately by assuming that (1) the unobservable perceived values have a trial-by-trial (or participant-by-participant) univariate normal distribution across the relevant sensory dimension, (2) the participant establishes a set of criteria or cut-points on the dimension that partitions the dimension into intervals, and (3) a different numerical rating is assigned to each interval Ashby (1988); Wickens (1992). This model assumes that on each trial, the participant determines in which interval the percept is in and then selects the associated rating.

Ashby & Ennis (2002) combined the unfolding model and the signal-detection theory model of the rating experiment to account for simultaneous sensory and liking ratings. This model used the participants sensory ratings to estimate the sensory representation of the ideal. However, the model was only developed and applied to situations in which the various stimuli all varied on a single sensory dimension. This article extends the model of Ashby & Ennis (2002) to more complex real-world stimuli that vary on many sensory dimensions. The resulting model estimates the distribution of imagined ideals (i.e., across trials and participants) by identifying the ideal mean on each rated sensory dimension and estimating the variance-covariance matrix of the ideal distribution across all rated dimensions.

The new model, which we call the multivariate Thurstone-Coombs ratings model (MTCRM), is described in the next section. Section three describes general methods for applying the model to data from an experiment that collects ratings on multiple sensory dimensions or attributes and on some hedonic dimension, such as liking. Section four describes an empirical test of the MTCRM against data from a new experiment. Section five discusses implications of our results and closes with some brief conclusions.

3.2 The Multivariate Thurstone-Coombs Ratings Model (MTCRM)

This section develops the MTCRM. Consider an experiment in which participants are presented with N stimuli (one per trial) and each stimulus varies on D sensory dimensions. The goal is to collect ratings from 1 to r on each stimulus on the sensory strength on all D dimensions and on liking or some other hedonic response (with r

representing maximum strength or maximum liking).

The model is illustrated in Figure 3.1 for a trial in which stimuli vary on two sensory dimensions and $r = 4$. The figure depicts events on trials in which the participant is asked to rate the stimulus on dimension 1 and on liking. In this hypothetical example, the participant responds with a rating of 3 on sensory magnitude and 2 on liking. The MTCRM makes the following assumptions.

1) The sensory value on a trial when stimulus i is presented is represented by a $D \times 1$ random vector $\underline{\boldsymbol{x}}_i$ in which $\underline{\boldsymbol{x}}_i' = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_D]$, where $\boldsymbol{x}_d$ represents the sensory magnitude on stimulus dimension d. Because of stimulus and perceptual noise and individual difference, $\underline{\boldsymbol{x}}_i$ varies randomly over trials and participants. We assume $\underline{\boldsymbol{x}}_i$ has a multivariate normal distribution with mean vector $\underline{\boldsymbol{\mu}}_i$ and variance-covariance matrix Σ_i.

Note that the variance-covariance matrix Σ_i contains $D(D-1)/2$ covariances and D variances. For example, in the next section we consider an application of the MTCRM to an experiment in which participants rate the stimuli on 6 sensory dimensions. In this case, each Σ_i includes 15 covariances and 6 variances. If these are all free parameters then the model would include 27 parameters for each stimulus (15 covariances, 6 variances, and 6 means). These would require an enormous amount of data for accurate estimation. Furthermore, estimation of the covariances would require simultaneous ratings on all possible pairs of dimensions, plus the assumption that all of these ratings are based on the same sensory sample of the stimulus. Unfortunately, this assumption seems untenable. For example, if a participant is asked to rate a stimulus on 6 different dimensions then it seems likely that the participant would re-examine the stimulus one or more times before responding with all 6 ratings. According to the model, the sensory representation of the stimulus after each examination is represented by a new random sample $\underline{\boldsymbol{x}}_i$. If ratings on two dimensions are based on different $\underline{\boldsymbol{x}}_i$ samples then the correlation (e.g., across trials)

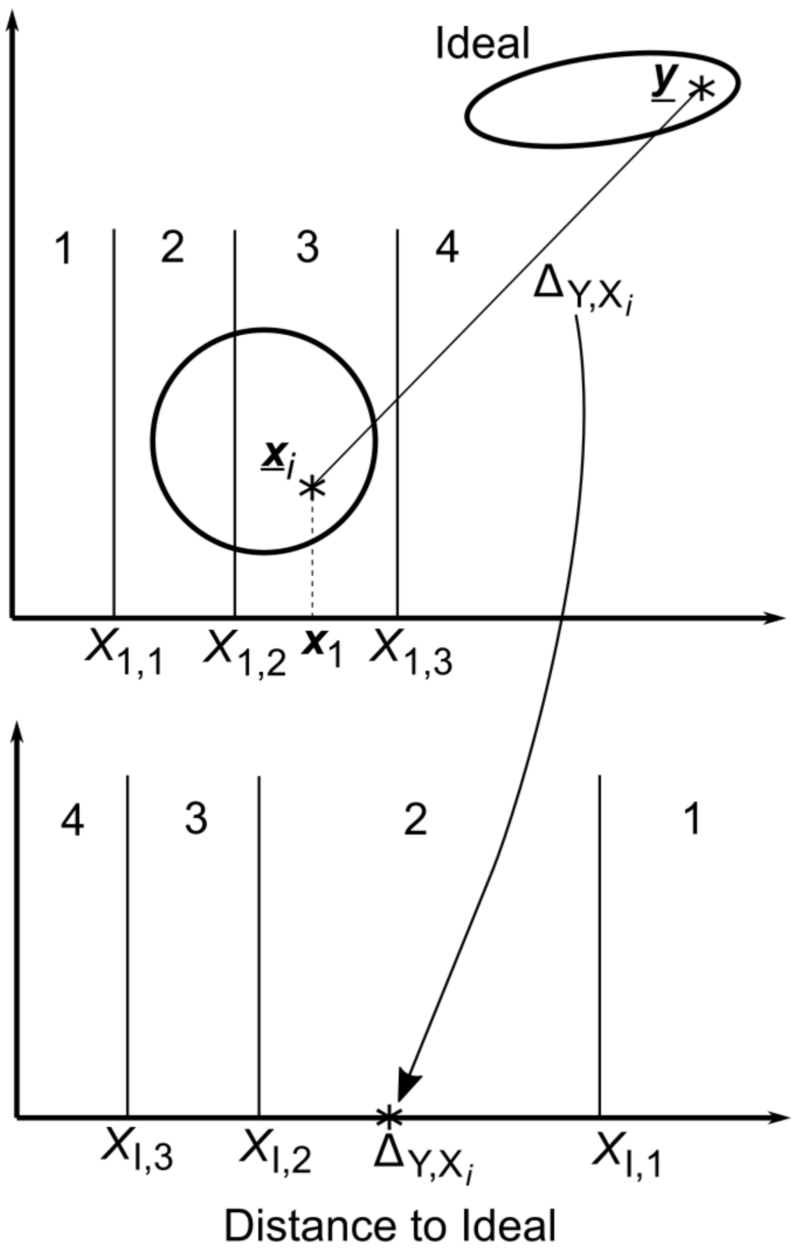

Figure 3.1: A schematic illustrating the multivariate Thurstone-Coombs ratings (MTCR) model for a trial when the participant is presented with stimulus i and asked to provide a rating (from 1 – 4) on the first of two sensory dimensions and on liking. The participant's responses on this trial are "3" on sensory magnitude and "2" on liking.

between the ratings will not reflect the correlation between sensory dimensions.

For these reasons, we only consider applications of the model to experimental paradigms in which a single one of the $D + 1$ ratings are requested on each trial, and each stimulus is presented to every participant on at least $D + 1$ different trials to ensure that all the necessary ratings are collected. In this case, no information about covariances is available, and as a result, we assume that all covariances equal 0 and therefore that Σ_i is diagonal. Furthermore, we also assume, without loss of generality, that all variances equal 1. This just serves to set the arbitrary unit of measurement on each dimension. Collectively, these assumption mean that, for all stimuli, $\Sigma_i = \mathrm{I}$, where I is the identity matrix.

2) When asked to rate the sensory magnitude of the stimulus on dimension d, the participant constructs $r - 1$ response criteria, denoted $X_{d,1}, X_{d,2}, ...X_{d,r-1}$, and responds with rating j if and only if $X_{d,j-1} < \boldsymbol{x}_d \leq X_{d,j}$, where $X_{d,0} = -\infty$ and $X_{d,r} = \infty$. Note that in the Figure 3.1 example, the perceived value of stimulus i on dimension 1 of this hypothetical trial (i.e., x_1) lies between $X_{1,2}$ and $X_{1,3}$ and therefore the participant rates the sensory magnitude of this stimulus on dimension 1 as 3.

3) To generate a liking rating, the participant first imagines an ideal stimulus, which is represented by the $D \times 1$ random vector $\underline{\boldsymbol{y}}$. Because of variability in the imagining process (e.g., due to variability in memory and affective state) and individual difference, $\underline{\boldsymbol{y}}$ varies randomly over trials and participants. We assume $\underline{\boldsymbol{y}}$ has a multivariate normal distribution with mean vector $\underline{\boldsymbol{\mu}}_Y$ and variance-covariance matrix Σ_Y.

In the Figure 3.1 example, note that the imagined ideal distribution has greater variance on dimension 1 than dimension 2, and that the values on these two dimensions have a slight positive correlation. The greater dimension 1 variance indicates that dimension 1 is less critical to liking than dimension 2 because when participants imagine their ideal they are more consistent in their imagined value on dimension 2 than on dimension 1.

4) The participant computes the Mahalanobis distance Δ_{Y,X_i} between the imagined ideal $\underline{\boldsymbol{y}}$ and the sensory value $\underline{\boldsymbol{x}}_i$ (from step 1).

5) The participant constructs $r-1$ response criteria, denoted $X_{\mathrm{I},1}, X_{\mathrm{I},2}, ...X_{\mathrm{I},r-1}$, and responds with rating j if and only if $X_{\mathrm{I},j} < \Delta_{Y,X_i} \leq X_{\mathrm{I},j-1}$, where $X_{I,0} = \infty$ and $X_{I,r} = 0$. Note that in the Figure 3.1 example, the distance between the imagined ideal and the perceived stimulus (i.e., Δ_{Y,X_i}) lies between $X_{\mathrm{I},2}$ and $X_{\mathrm{I},1}$ and therefore the participant responds with a liking rating of 2.

3.3 Fitting the Model to Data

For each stimulus, the data can be collected as a $(D+1)\times r$ matrix in which the entry in row d and column j is the frequency that participants assigned rating j to the stimulus on dimension d, where row $D+1$ is liking. Note that each matrix has $(D+1)\times(r-1)$ degrees of freedom, since there is one constraint per row (i.e., each row sum equals the number of trials that participants rated the stimulus on the attribute associated with that row). There is one such matrix for each of the N stimuli, so overall, the data include $N\times(D+1)\times(r-1)$ degrees of freedom.

The model predicts that the probability that rating j is assigned to stimulus i on sensory dimension d equals the area under the dimension d marginal pdf of $\underline{\boldsymbol{x}}_i$ between $X_{d,j-1}$ and $X_{d,j}$. Because these marginal distributions are all normal, each of these probabilities can be computed via straightforward z transformations and appeal to the cumulative z distribution function.

Computing the predicted probabilities of various liking ratings is considerably more difficult. The predicted probability that participants assign stimulus i a liking rating of j equals

$$P_{\mathrm{L}}(j|S_i) = P(X_{\mathrm{I},j} < \Delta_{Y,X_i} \leq X_{\mathrm{I},j-1}), \tag{3.1}$$

where, as before, Δ_{Y,X_i} is the Mahalanobis distance between the imagined ideal $\underline{\boldsymbol{y}}$ and the sensory value $\underline{\boldsymbol{x}}_i$. Since Δ_{Y,X_i} is nonnegative, note that

$$\begin{aligned} P_{\mathrm{L}}(j|S_i) &= P(X_{\mathrm{I},j} < \Delta_{Y,X_i} \leq X_{\mathrm{I},j-1}) && (3.2)\\ &= P(X^2_{\mathrm{I},j} < \Delta^2_{Y,X_i} \leq X^2_{\mathrm{I},j-1}). && (3.3)\end{aligned}$$

Now

$$\Delta^2_{Y,X_i} = (\underline{\boldsymbol{y}} - \underline{\boldsymbol{x}}_i)'\Sigma_Y^{-1}(\underline{\boldsymbol{y}} - \underline{\boldsymbol{x}}_i), \tag{3.4}$$

which has the distribution of a weighted sum of D non-central χ^2 random variables, each with one degree of freedom (e.g., Paolella (2018)). In the application described in the next section, $D = 6$, which is large enough so that this weighted sum is approximately normally distributed. Therefore, we need only to compute the mean and variance of Δ^2_{Y,X_i}.

Define the new random vector $\underline{\boldsymbol{w}} = \underline{\boldsymbol{y}} - \underline{\boldsymbol{x}}_i$. Then Eq. 3.4 becomes

$$\Delta^2_{Y,X_i} = \underline{\boldsymbol{w}}'\Sigma_Y^{-1}\underline{\boldsymbol{w}}. \tag{3.5}$$

Note that $\underline{\boldsymbol{w}}$ is multivariate normally distributed with mean vector $\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i$ and variance-covariance matrix $\Sigma_Y + \mathrm{I}$.

The mean of the Eq. 3.5 random variable is (e.g., Khatri (1980))

$$\begin{aligned} \mu_{\Delta^2} &= \mathrm{trace}(\Sigma_Y^{-1}\Sigma_{\underline{\mathrm{w}}}) + \underline{\boldsymbol{\mu}}'_{\underline{\mathrm{w}}}\Sigma_Y^{-1}\underline{\boldsymbol{\mu}}_{\underline{\mathrm{w}}}\\ &= \mathrm{trace}[\Sigma_Y^{-1}(\Sigma_Y + \mathrm{I})] + (\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i)'\Sigma_Y^{-1}(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i)\\ &= D + \mathrm{trace}(\Sigma_Y^{-1}) + (\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i)'\Sigma_Y^{-1}(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i). && (3.6)\end{aligned}$$

Note that the last term is just the squared Mahalanobis distance between the means of the two distributions. The variance of the Eq. 3.5 random variable is (e.g., Khatri (1980))

$$\begin{aligned}\sigma^2_{\Delta^2} &= 2[\text{trace}(\Sigma_Y^{-1}\Sigma_{\underline{\text{w}}})]^2 + (2\Sigma_Y^{-1}\underline{\boldsymbol{\mu}}_{\underline{\text{w}}})'\Sigma_{\underline{\text{w}}}(2\Sigma_Y^{-1}\underline{\boldsymbol{\mu}}_{\underline{\text{w}}}) \\ &= 2[D + \text{trace}(\Sigma_Y^{-1})]^2 + 4(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i)'\Sigma_Y^{-1}(\Sigma_Y + \text{I})\Sigma_Y^{-1}(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i) \\ &= 2[D + \text{trace}(\Sigma_Y^{-1})]^2 + 4(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i)'\Sigma_Y^{-1}(\text{I} + \Sigma_Y^{-1})(\underline{\boldsymbol{\mu}}_Y - \underline{\boldsymbol{\mu}}_i). \qquad (3.7)\end{aligned}$$

Therefore, we can approximate the predicted probability that rating j is assigned to stimulus i on the liking dimension by computing the area between $X^2_{\text{I},j}$ and $X^2_{\text{I},j-1}$ under the pdf of a normal distribution with mean and variance specified by Eqs. 3.6 and 3.7, respectively.

3.4 An Empirical Application

As an empirical test of the model, we ran an experiment in which people rated the 20 images of hypothetical planets shown in Figure 3.2 on six sensory dimensions and on liking. Specifically, participants were told to imagine that they were in a spaceship traveling through deep space, and that their mission was to rate planets they encountered (from 1 to 7) on the prominence of a number of sensory dimensions (water, clouds, rings, moons, blue-green, red-yellow) and on how important it was to retain a photograph of the planet and send it back to earth.

3.4.1 Stimuli

All images were gathered using SpaceEngine (SpaceEngine.org), a universe simulator that randomly generates a plethora of astronomical objects. The procedural generation process creates 3-dimensional rendered planets, which are captured with extreme detail

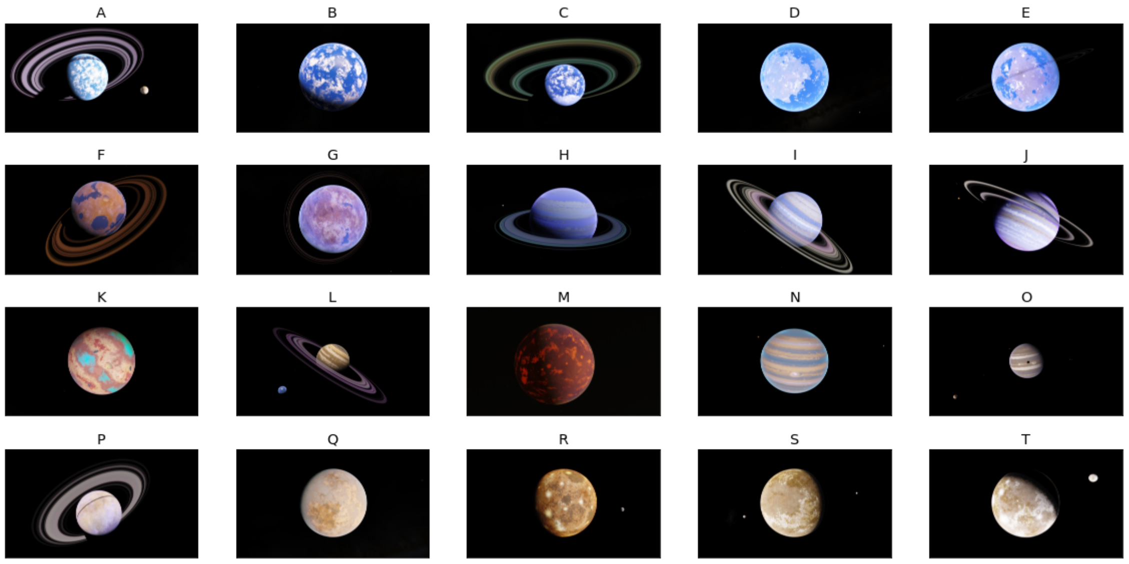

Figure 3.2: The planets shown to each participant ordered by distance to the ideal (with planet A closest to the ideal and planet T furthest from the ideal).

using a 3840×2160 $4K$ resolution and resulting in over 8 million pixels per image. Due to the stochastic nature of each planet, the options for planetary features and combinations are nearly limitless. The stimuli used in this experiment are displayed in Figure 3.2.

3.4.2 Participants

Twenty-nine students at the University of California, Santa Barbara participated in an (approximately) one-hour experiment in exchange for course credit. All participants had normal color vision. All relevant ethical regulations were followed and the study protocol was approved by the Human Subjects Committee at UCSB. Informed consent was obtained from all participants, and every participant was allowed to quit the experiment at any time for any reason and still receive credit.

3.4.3 Procedure

Participants were told to imagine that they were in a spaceship traveling through deep space and that the ship automatically takes photos of planets that it encounters. They were also told that their mission was to rate each planet on a number of physical attributes and on how important they thought it was to send the image back to earth so that the rest of humanity would know of that planet's existence. Participants were presented the images in 5 phases. During each phase, the 20 images were displayed one-at-a-time in a random order. In phase 1, participants passively observed the images. In phases 2-5, each image was displayed with a ratings bar that ranged from 1 to 7 and participants were instructed to move the mouse and click the integer on the ratings bar that agreed with their rating. During phases 2 and 4, the image and ratings bar were accompanied by a word cue that specified the physical attribute to be rated, such as "water". Participants rated 6 different attributes (or dimensions) and each attribute/image combination was presented once per phase, resulting in a total of 240 sensory judgments during phases 2 and 4 (2 sensory judgments per planet per dimension). During phases 3 and 5, the image and ratings bar were accompanied by the word cue "importance". Prior to each phase the participants were reminded that their job is to use the mouse to click the value on the scale that best reflects how prominent the feature is, with 1 being least prominent and 7 being most prominent and that this feature is indicated by the word presented above the scale. Each image was presented once during phases 3 and 5, resulting in 2 liking judgments per planet.

A few participants were not sufficiently engaged in some phases of the experiment. These participants tended to repeat the same rating, over and over. Therefore, any liking phase (3 and 5) in which the participant emitted 3 or fewer unique ratings was excluded from analysis. Six liking phases were excluded, leaving 52 liking phases for analysis.

Additionally, any sensory phase (2 and 4) in which the participant gave the same rating on any dimension to all images was excluded from analysis. One sensory phase was excluded resulting in 57 sensory phases for analysis.

3.4.4 Results

The data from this experiment were aggregated across participants and then recorded in a 20 (planets) $\times$ 7 (dimensions) $\times$ 7 (ratings) frequency array, where liking was included as one of the 7 dimensions. The liking ratings for each planet are shown in Figure 3.3. For each planet and dimension, the frequency sum across the 7 ratings equals the number of trials participants were asked to rate that planet on that dimension. Therefore, the data include 6 degrees of freedom for each planet and dimension, and so the entire data set includes 840 degrees of freedom (i.e., 20 $\times$ 7 $\times$ 6).

The MTCRM model was fit to these data. The model included a total of 183 free parameters. Without loss of generality, we fixed the mean vector for planet N, $\underline{\boldsymbol{\mu}}_N$, to the zero vector. Additionally, to limit the number of free parameters, we fixed the variance-covariance matrices of all sensory distributions to $\Sigma_i = \mathrm{I}$. The following parameters were all free to vary:

1) The remaining 19 mean vectors, $\underline{\boldsymbol{\mu}}_i$ for all $i \neq N$. Each $\underline{\boldsymbol{\mu}}_i$ is 6×1, so there were a total of 114 free mean parameters (i.e., 19 $\times$ 6).

2) Six criteria, $X_{d,j}$, on each of the 6 sensory dimensions, resulting in an additional 36 parameters.

3) Six means for the ideal distribution, $\underline{\boldsymbol{\mu}}_Y$,

4) The 6 $\times$ 6 ideal variance-covariance matrix, Σ_Y (21 free parameters).

5) Six criteria, $X_{\mathrm{I},j}$, on the squared-distance-to-ideal dimension.

All parameters were estimated via constrained optimization by linear approximation

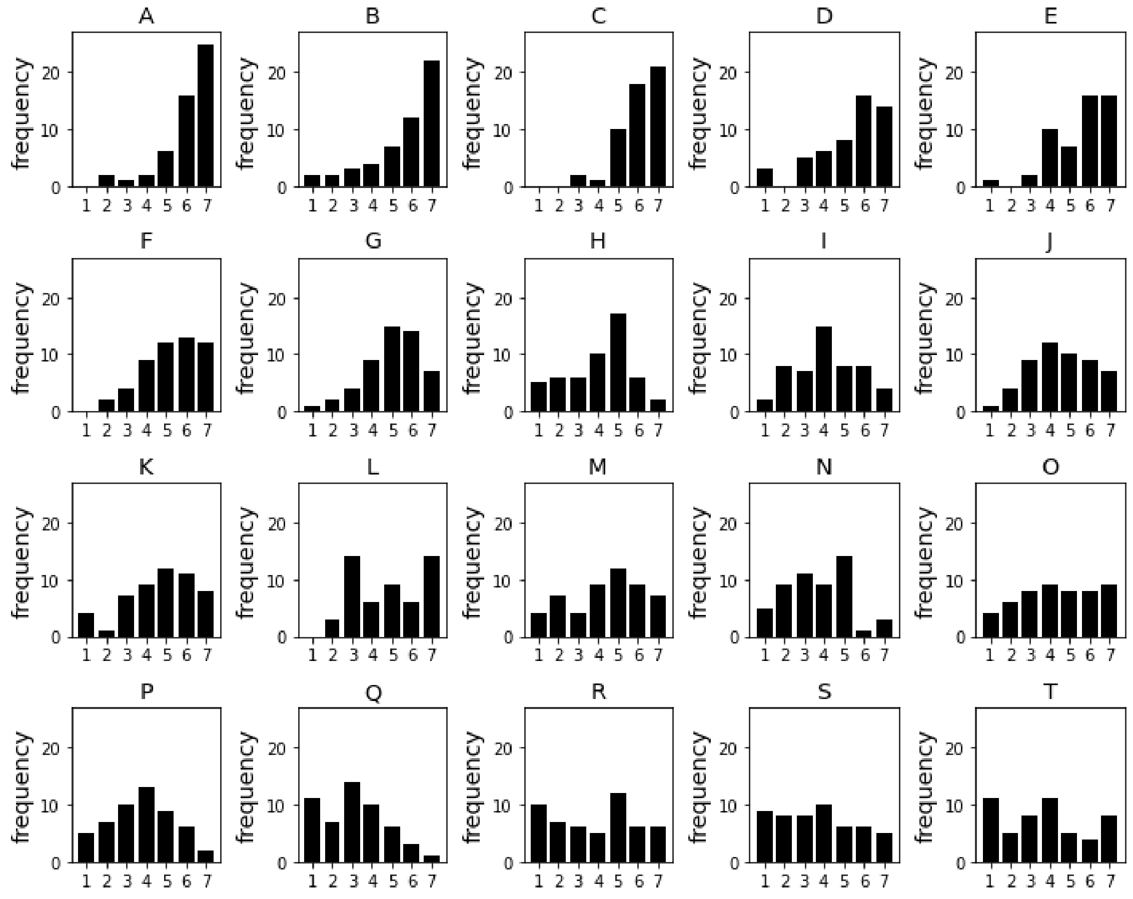

Figure 3.3: Liking ratings for each of the 20 planets.

(COBYLA; Powell (1994)) using SciPy Virtanen et al. (2020) in Python Van Rossum & Drake Jr (1995) by minimizing the sum of squared errors between the predicted and observed response frequencies.

Overall, the MTCRM accounted for 95.19% of the variance in the data (r^2). Although the model included 183 free parameters, because the data had 840 degrees of freedom, after parameter estimation, there were still 657 degrees of freedom left to test the model (i.e., $840 - 183$). So accounting for 95% of the variance in these 657 proportions seems impressive. Not surprisingly, however, the model was more successful at accounting for the sensory ratings than the liking ratings. Specifically, the MTCRM accounted for 96.09% of the variance in the sensory ratings data and 71.89% of the variance in the liking ratings.

Figure 3.4 shows estimated sensory distributions for each planet on each dimension as well as the estimated criteria. Note that, except for rings, the planets vary fairly continuously on all sensory dimensions. Not surprisingly, the perceived prominence of rings is approximately bimodal with some planets displaying prominent rings (e.g., planets A, F, and C) and other planets showing a prominent absence of rings (e.g., planets B, D, and S).

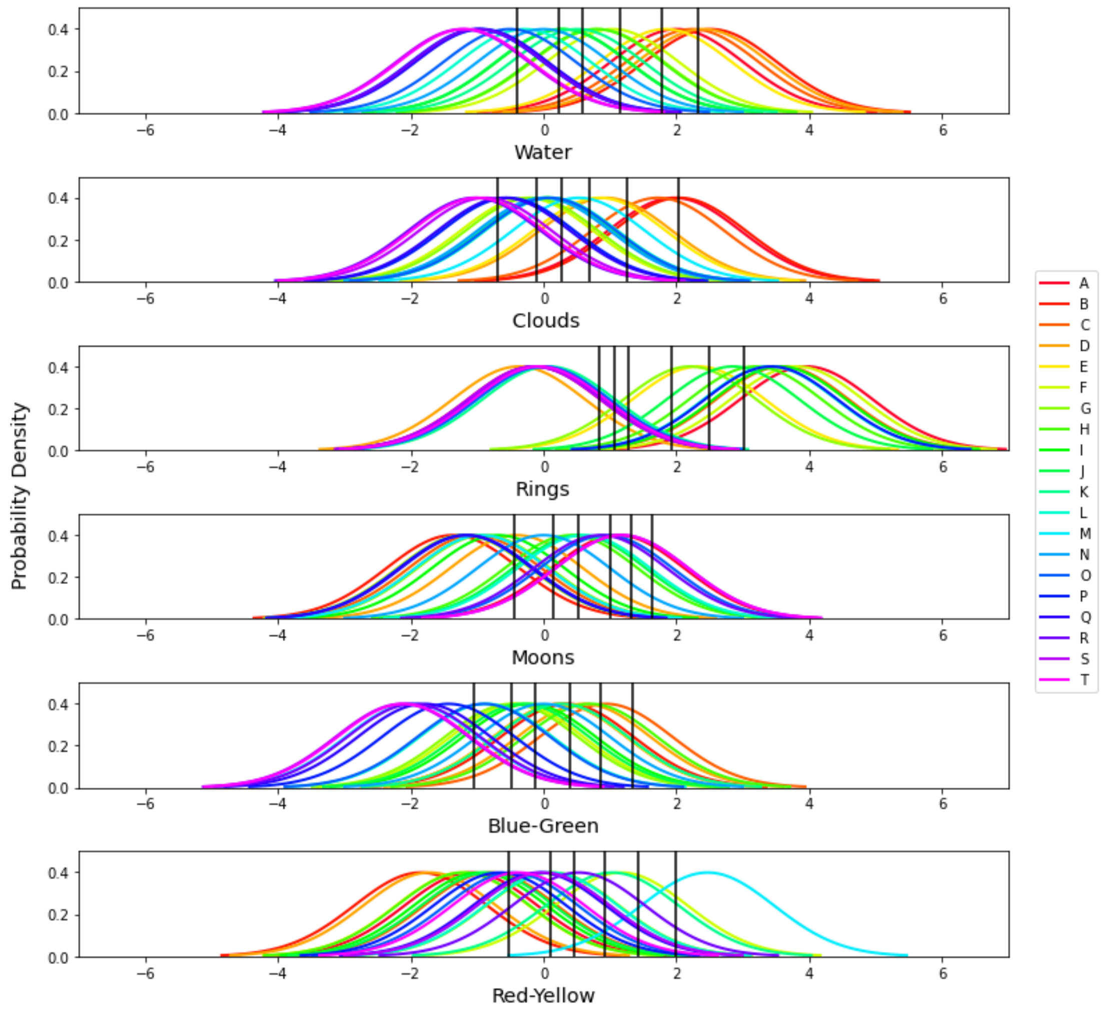

Figure 3.4: Estimated sensory distributions from the best-fitting version of the MTCRM, along with the estimated criteria on each dimension that participants used to assign ratings.

Table 1 shows the variance-covariance matrix of the estimated ideal distribution. The variances provide an inverse measure of how important each dimension is to the ideal. Note that the smallest variance is on the clouds dimension and the next smallest is on water. The small variances suggest that when ideal planets are imagined on different trials, participants always tend to imagine a planet with similar values on the water and cloud dimensions. In contrast, the variances on the red-yellow and moons dimensions are large, suggesting that the different imagined ideals vary widely on the red-yellow and moons dimensions. Therefore, for example, if the imagined ideal sometimes has a moon and sometimes does not, then the presence or absence of a moon is not an important attribute of the ideal planet.

Table 3.1: Variance-covariance matrix of the ideal distribution from the best-fitting version of the MTCRM.

	Water	Clouds	Rings	Moons	Blue-Green	Red-Yellow
Water	17.82	-11.33	-3.92	1.70	13.69	18.93
Clouds		14.11	4.04	-0.59	0.74	-12.60
Rings			203.89	-1.57	61.24	55.46
Moons				418.06	63.21	207.57
Blue-Green					61.36	76.60
Red-Yellow						706.73

Figure 3.5 shows the ideal distribution and the mean of each planet distribution projected onto the plane defined by the two most important dimensions – namely, water and clouds. The ellipses denote the contours of equal likelihood of the ideal distribution, so the ideal mean lies at the center of these ellipses. Note from Table 1 that water and clouds are negatively correlated in the ideal distribution, which is the reason that the ellipses in Figure 3.5 have a negative orientation. This makes sense because as cloud cover increases there is less available surface to display water. Note that planet B is closest to the ideal mean, closely followed by planets A and C, and that planets R, S,

and T are furthest (i.e., see Figure 3.2). Therefore, these data suggest that the ideal planet would have about the same cloud cover as planet C but more water than any of the planets that were shown to participants.

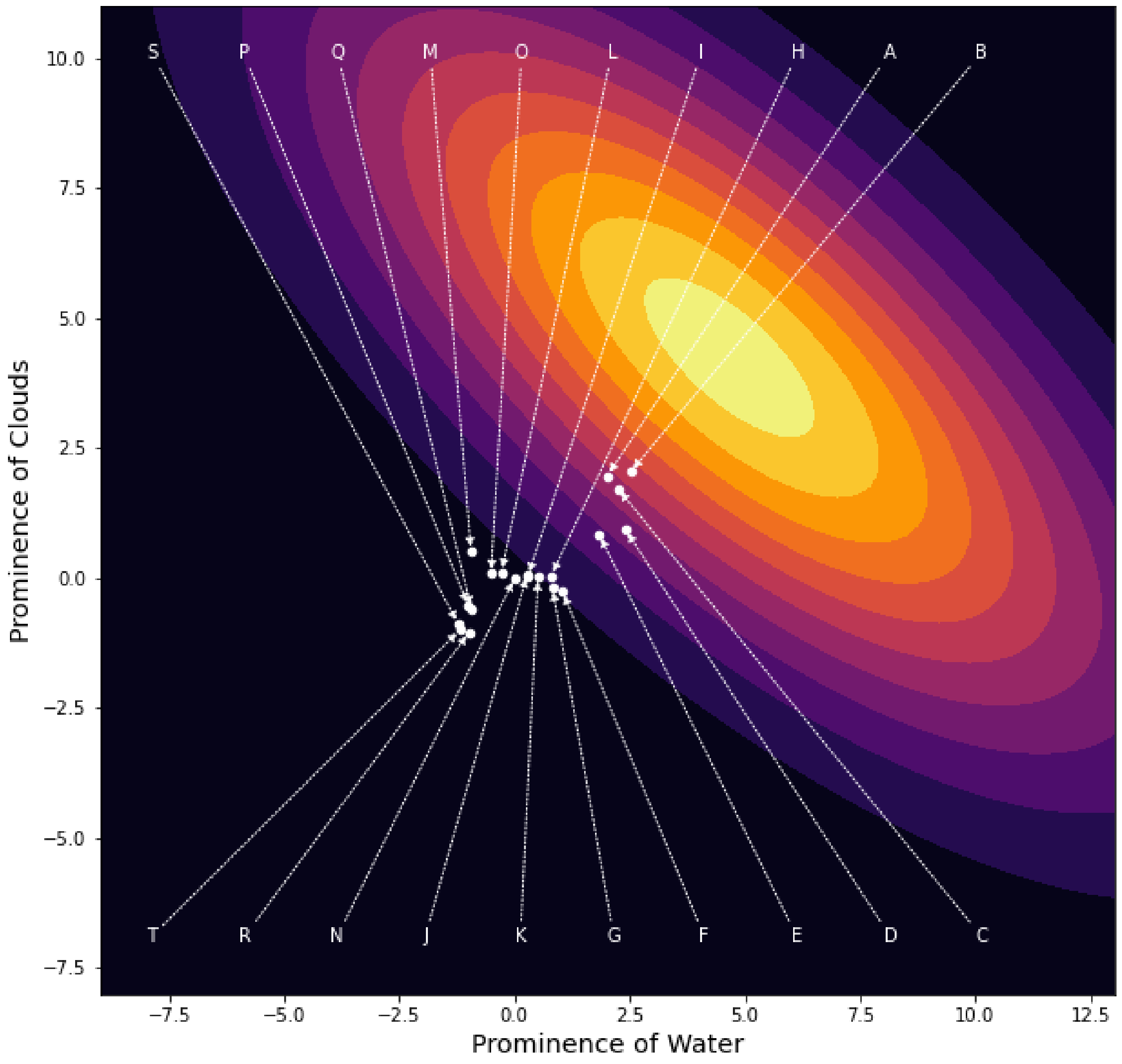

Figure 3.5: Contours of equal likelihood of the ideal distribution from the best-fitting version of the MTCRM (i.e., the ellipses) and the sensory means of each planet projected onto the plane defined by the water and cloud dimensions.

3.5 Discussion

The MTCRM uses sensory ratings to build a probabilistic, multidimensional representation of the sensory experiences elicited by exposure to each stimulus. If participants rate the stimuli on D sensory dimensions then the sensory representations built by the model will be D dimensional. And the model will also build a representation of the ideal stimulus in this same space. It then attempts to account for liking ratings by measuring differences between the presented stimulus and the imagined ideal on each of these D sensory dimensions. This approach can only hope to account for the hedonic responses of participants if the rated sensory dimensions include all stimulus attributes that significantly affect liking. To take an extreme example, consider an experiment in which participants rate a set of stimuli on D sensory dimensions but that the participants' hedonic responses to those stimuli depend exclusively on some other, unrated sensory dimension. In this case, their hedonic responses will be independent of the stimulus value on any of the rated dimensions, and therefore a comparison of the stimulus to the ideal values on the D rated dimensions will not predict the participant's hedonic response. So the efficacy of the MTCRM depends strongly on the ability of the experimenter to identify all sensory dimensions that could significantly affect the hedonic responses of participants to the selected stimuli.

Given this, the default expectation should be that the model will account for sensory ratings better than it accounts for hedonic ratings. In the experiment described here, the MTCRM accounted for 96% of the variance in the sensory ratings and 72% of the variance in the hedonic ratings. Therefore, we believe that one plausible account for this difference is that participants based their hedonic responses, at least in part, on some unrated dimension or attribute of the planets. Traditional multidimensional unfolding models that lack any sensory data could just add more unspecified dimensions to the model until goodness-of-fit is maximized (e.g., exactly as in MDS). Even so, note that a better fit by such a model would provide only vague information about the sensory qualities of the ideal. Given that the MTCRM provides precise estimates of the sensory qualities of the ideal on all rated sensory dimensions, we believe that accounting for 72% of the variance in the hedonic ratings is impressive, especially since the model was provided no information about how participants might make these judgments.

As an empirical test of the MTCRM, we chose the planets shown in Figure 3.2 because they are interesting, real-world objects. However, the MTCRM could be applied to any stimuli. Future research could improve on the experimental design by applying the model to domains that rely on more well defined dimensions selected by experts with domain specific knowledge. For example, the model could be applied to find the ideal Merlot and sommeliers could be consulted to choose dimensions, such as sweetness and acidity, etc. It seems reasonable to suspect that this may ameliorate the problem of participants basing their hedonic responses on some unrated dimension. Furthermore, this experiment could investigate the effect of the development of expertise on the location and variance-covariance matrix of the ideal. For example, the model could be fit separately to experiments conducted on the general population and Master Sommeliers. The development of expertise may cause the location of the ideal to shift and the variances of the ideal to shrink. Lastly, because the sensation elicited by a stimulus is related to an underlying physical quantity (e.g. the sweetness of Merlot is related to residual sugar content, among other factors), locating the sample Merlots and the ideal Merlot in sensory space can inform product development. Using this information, one could attempt to physically instantiate the ideal Merlot discovered by the model.

3.6 Citations

References

Andrich, D. 1989, Applied Psychological Measurement, 13, 193

Ashby, F. G. 1988, Perception & Psychophysics, 44, 195

Ashby, F. G., & Ennis, D. M. 2002, Journal of Sensory Studies, 17, 43

Borg, I. 2018, Applied multidimensional scaling and unfolding (Amsterdam: Springer)

Coombs, C. H. 1964, A theory of data (New York: Wiley)

Davison, M. L. 1979, Psychometrika, 44, 179

De Soete, G., Carroll, J. D., & DeSarbo, W. S. 1986, Journal of Mathematical Psychology, 30, 28

DeSarbo, W. S., & Rao, V. R. 1984, Journal of Classification, 1, 147

DeSarbo, W. S., Young, M. R., & Rangaswamy, A. 1997, Journal of Marketing Research, 34, 499

Green, D. M., & Swets, J. A. 1966, Signal detection theory and psychophysics (New York: Wiley)

Hirsch, J., Hylton, R., & Graham, N. 1982, Vision Research, 22, 365

Khatri, C. G. 1980, in Handbook of Statistics, Volume 1, ed. P. R. Krishnaiah (Amsterdam, North-Holland), 443–469

Macmillan, N. A., & Creelman, C. D. 2005, Detection theory: A user's guide (Mahwah, NJ: Lawrence Erlbaum Associates)

Olzak, L. A. 1986, Vision Research, 26, 1143

Paolella, M. S. 2018, Linear Models and Time-Series Analysis: Regression, ANOVA, ARMA and GARCH (Hoboken, NJ: John Wiley & Sons)

Powell, M. J. 1994, in Advances in optimization and numerical analysis (Springer), 51–67

Schönemann, P. H., & Wang, M. M. 1972, Psychometrika, 37, 275

Van Rossum, G., & Drake Jr, F. L. 1995, Python reference manual (Centrum voor Wiskunde en Informatica Amsterdam)

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2

Wickens, T. D. 1992, Journal of Mathematical Psychology, 36, 213

Chapter 4

Model Optimization for Deep Space Exploration via Simulators and Deep Learning

4.1 Introduction

This paper will address some of the challenges and possibilities of exoplanet detection and classification for future exosolar system missions. Future missions may allow for travel far outside of our solar system, as well as deep into our own solar system, where return bandwidth will be severely limited; thus, choices of which data (images in particular) are important to ”send back” (Lubin, P. (2016), Lubin & Hettel (2020), Sheerin et al. (2020)). The basis for exoplanetary detection via fast interstellar travel is a combination of The Starlight Program (Kulkarni et al., 2017) and recent results that show how exoplanets can be detected, and distinguished from other objects, via AI-based modeling that utilizes simulated data (Bird et al., 2020). The groundwork has been laid for an AI-based small spacecraft that can travel long distances in a short amount of time, gather information on its surroundings with minimal energy requirements, and detect exoplanets and other targets of interest with excellent accuracy. The same core technology we discuss here can be applied to a wide range of astrophysics and cosmology where subtle and often transient phenomenon are critical to retrieve in low SNR situations. In future papers we will discuss using our techniques in these other application spaces.

The major points that we will discuss and examine here are related to the accuracy of exoplanetary detection. In our foundational paper (Bird et al., 2020), we used a robust model and detection score for proof of concept. Going forward, this paper will compare a wide array of models using accuracy as our main metric to determine model strength and reliability.

4.2 Previous Work

The basis for much of our work lies in deep learning via TensorFlow (Abadi et al., 2016), as well as the expected additions, such as cuDNN (Chetlur et al., 2014) and CUDA, which allows for faster deep neural network processing via a graphics processing unit (GPU). Although the idea of direct exoplanetary detection and imaging via interstellar travel is new, astronomy has been attempting the general feat via light curves for years, and even more recently with deep learning (Shallue & Vanderburg (2017), Zucker et al. (2018), Carrasco-Davis et al. (2018)).

For direct imaging purposes, we test a variety of robust models, including variants of each model, and analyze factors such as accuracy and computational complexity. Since deep space is uncharted territory, an extremely large training data set is not possible. Therefore, we include some simpler models to offset the possibility of having models that are too advanced for the data. The overall goal of these models is to be able to identify when a planet is present in an image, while also being capable of not mistaking other astronomical objects for planets.

For the simpler models, we will compare MobileNet (Howard et al., 2017), MobileNet V2 (Sandler et al., 2018), DenseNet 121, 169, and 201 (Huang et al., 2018), and NASNet-Mobile (Zoph et al., 2017). These provide solid baseline accuracy and low computational complexity, which may prove to be beneficial for our specific needs. For the intricate models, we will compare NASNet-Large (Zoph et al., 2017), Xception (Chollet, F., 2017), VGG 16 and VGG19 (Simonyan & Zisserman, 2015), Inception V3 (Szegedy et al., 2015), Inception-ResNet V2 (Szegedy et al., 2016), and ResNet 50, 50 v2, 101, 101 v2, 152, and 152 v2 (He et al., 2015). In contrast to the simpler models listed above, the training time and complexity will increase with these. However, that process is done beforehand while the wafer satellite (wafersat) is still on Earth, so these concerns are negligible when compared to the possible gains in accuracy from the more robust models.

These models have been tested against each other in the past to some degree. ResNet has been shown to out-perform VGG (He et al. (2015),Canziani et al. (2016)) and even advanced Inception models (Le et al., 2020), while other results show all of these models being out-performed by the DenseNet and InceptionResNet architectures (Zhen et al., 2018).

The structure of these models and their performance is dependent on the data that is being processed. In this case, we are training on simulated images of planets and testing on real images of planets. This concept was shown to be viable in Bird et al. (2020); however, optimizing this process will require an in-depth look at advanced deep learning techniques and models.

4.3 The Process

4.3.1 Deep Neural Network Architecture

Deep neural networks, including those used for object detection, begin by deconstructing images into pixel-based groupings that constitute an input layer. This layer, along with the hidden layer(s) and output layer, is comprised of smaller entities called neurons. Each layer of neurons is connected to the next via weights, which are learned through a training process. Gradient descent is a powerful and widely used method that allows us to minimize the cost function in order to get the most effective learning process. By taking the negative gradient, we minimize the cost function. After all is done, we are left with a network that can take an input image and output something of interest based on the training and model parameters.

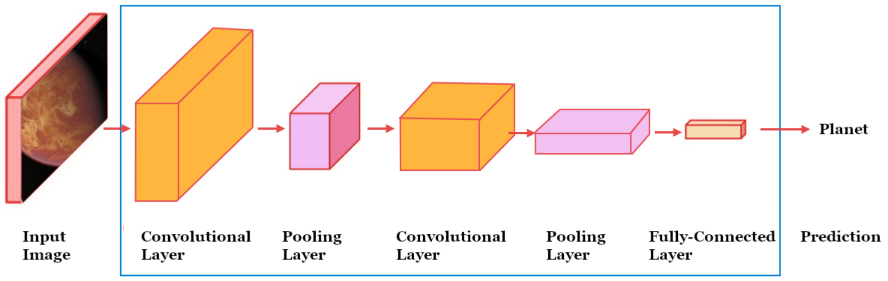

Figure 4.1: A convolutional neural network decomposed into a simplified diagram.

A more illuminating analogy would be to treat the initial inputs (pixels, or in the case of a convolutional neural network (CNN), groupings of pixels) as an input tensor. This input tensor is then essentially acted upon by a function (the neural network), which outputs a tensor corresponding to the the categorization of the input. This function initializes with random values, and is then optimized via the methods described above such that the output tensor has the highest accuracy when identifying inputted data.

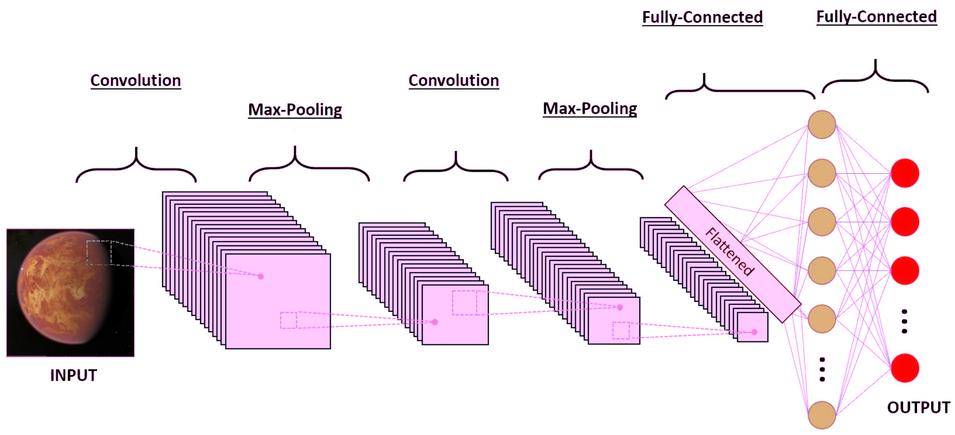

Figure 4.2: A convolutional neural network decomposed into a more descriptive diagram.

In Figure 4.2, the general process that occurs within a CNN is shown. First, an image is inputted. A small filter is placed and moved throughout the image, known as a convolution, and features are extracted depending on what filter is used. Next, pooling occurs, which down-samples the data. Many *convolution* + *pooling* cycles can occur, followed by a flattening of the finalized data. This flat data then enters a full-connected neural network, followed by an output layer, resulting in a prediction. Note that the output layer in Figure 4.2 has multiple categories, while our networks are binary.

4.3.2 The Setup

As discussed in detail in Bird et al. (2020), the simulator (SpaceEngine.org) provides us with easy access to 4K, 3-D rendered images of exoplanets. Although they are randomly generated, one could create a specific planet, or filter planets by a set of conditions in order to achieve a subset of planets that have certain traits. All models were pre-trained on ImageNet (Deng et al., 2009) and fine-tuned on simulated images of exoplanets. This allowed for a robust learning experience for features, and a more specific learning experience for our data set. All models were evaluated using an AMD Ryzen Threadripper 3970X 32-Core Processor @ 3.70 GHz, 128 GB of RAM and an NVIDIA Titan RTX graphics card.

In most deep learning applications for image analysis, both training sets and testing sets contain images of the same object. In our deep learning application, the training set is taken from a universe simulator (SpaceEngine.org), and the testing images are real images of planets. Without a simulator, we would not have enough images of planets, and those planets would not constitute a large enough sub-sample of possible exoplanets. By using a physics-based simulator, we can produce an abundance of realistic novel exoplanets to train on. Then, we use real planets to test the model's accuracy. This translates directly to the wafersats process during an actual interstellar journey. Image counts for all three sets is shown in Table 1.

Training	Validation	Testing
915	200	284

Table 4.1: Image count for the training, validation, and testing sets.

The process being performed here is unique for two major reasons. First, the entire training set is simulated images, while the entire testing set is real images. This presents a particular challenge for neural networks, as they learn in a template space and are then tested in a real space. Second, deep space provides an enormously large variety of objects. For example, gas giants vary wildly in many ways, such as size, feature differences, surface gas formations, colors, temperature, and more. In our solar system alone, we witness two quite unique gas giants: Saturn with its rings and famous hexigon, and Jupiter with its eye and dolphin formations. Training on extremely unique objects can cause neural networks to lose their generality and under-perform. The images in Figure 4.3 compare real images taken by NASA against simulated images.

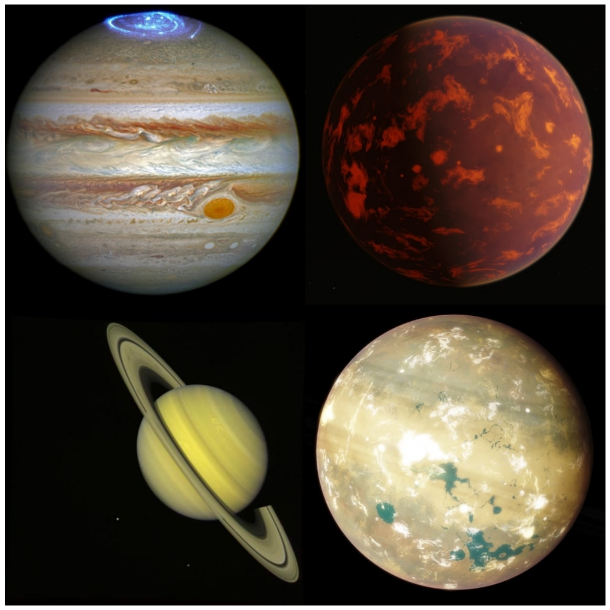
Figure 4.3: Two examples of real testing images are Jupiter and Saturn, seen on the left, while the right side shows examples of simulated training images.

4.4 The Results

For each model previously mentioned, we trained, validated, and tested the neural network in batches of five epochs each. An epoch is when the entire data set is run through the neural network once.

Model	Maximum Accuracy Achieved	Respective Epoch
VGG19	0.9964788556	5
VGG16	0.9929577708	5 & 25
ResNet50	0.9894366264	85 & 120
ResNet101	0.9894366264	115
ResNet152	0.9894366264	10 **
MobileNet	0.985915482	5
MobileNet v2	0.9788732529	5**
DenseNet121	0.9788732529	5
DenseNet169	0.9788732529	5 - 15
ResNet152 v2	0.9753521085	25 *
DenseNet201	0.9753521085	5-15
Inception v3	0.9683098793	5 & 15
ResNet101 v2	0.9647887349	5
ResNet50 v2	0.9612675905	10 - 20
NasNet-Mobile	0.9542253613	10
Inception-ResNet v2	0.950704217	5 & 10
Xception	0.950704217	10
NasNet-Large	0.9154929519	5

Table 4.2: Maximum epoch-based accuracy achieved for each model, ordered from highest to lowest. ** denotes continued accuracy for all remaining epoch counts. ** denotes that maximum accuracy was sporadically achieved again after first occurrence.*

From Table 2, as well as Figures 4.4 and 4.5, it can be seen that VGG19 reached the highest maximum accuracy at five epochs, while VGG16 reached the second-highest maximum accuracy at both five and 25 epochs. This result is extremely interesting, as VGG variants are typically under-performing models when compared to Inception or ResNet variants. MobileNet performed extremely well, not only in maximum accuracy achieved, but also in terms of consistency. The remaining models were simply out-performed and provided no concrete reason why they should be chosen as a viable model for this specific task.

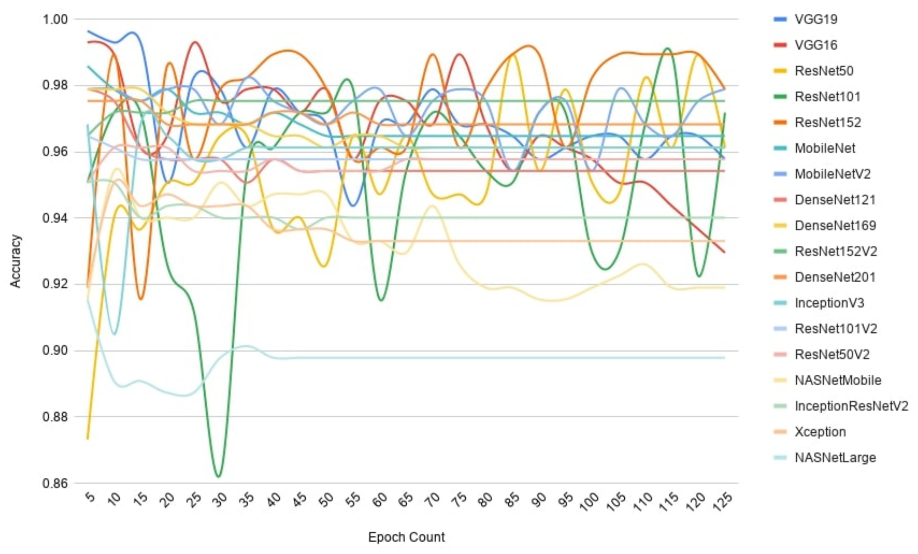

Figure 4.4: Accuracy of all models based on epoch count.

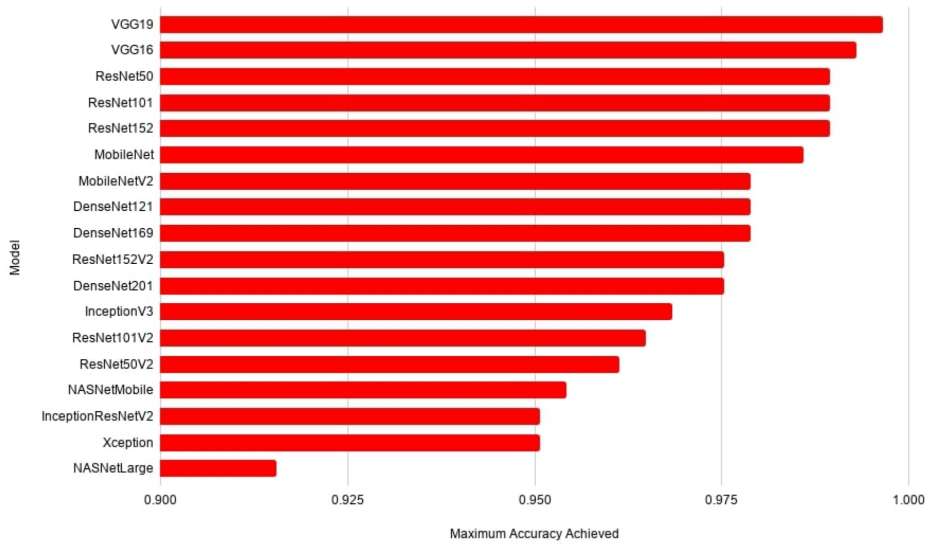

Figure 4.5: Accuracy of all models based on epoch count.

Based on Figure 4.6, ResNet50 and ResNet101 both dipped below 88% accuracy, and often times bounced from low to high accuracy, showing clear signs of inconsistency. Despite overall good performance, ResNet152 has large dips in the 5-20 epoch count, the main area where most models performed at their best. For these reasons, the ResNet variants were ultimately rejected as reasonable choices, as their epoch-based accuracy fluctuated too wildly.

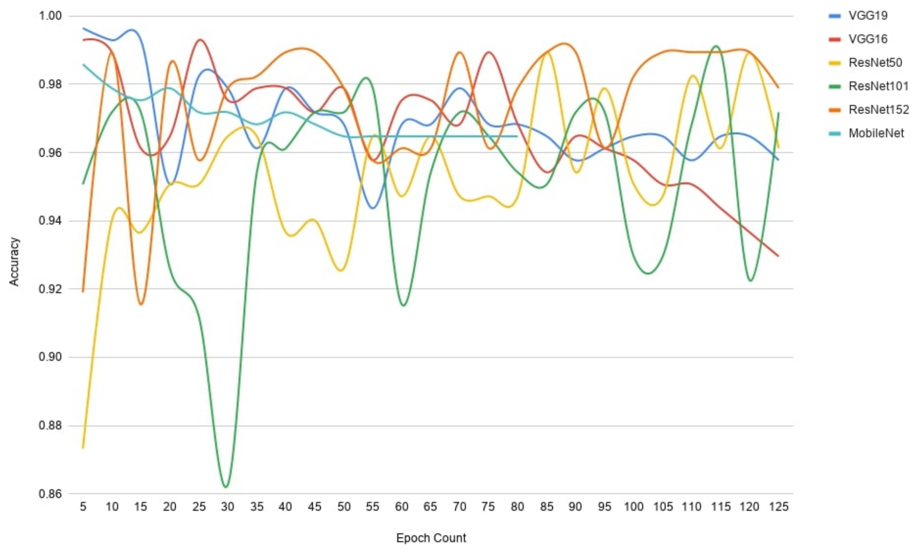

Figure 4.6: Accuracy of models that reached at least 98% maximum accuracy based on epoch count.

One very interesting result was concerning the remaining 98% or better maximum accuracy models, namely VGG19, VGG16, and MobileNet. Pertaining to Figure 4.7, we can see that VGG19 and VGG16 fluctuate to some degree, while MobileNet acts as a hedge. While the dependability of VGG19 and VGG16 in certain epoch ranges is vastly superior, MobileNet grants you a consistently strong choice across all epoch ranges.

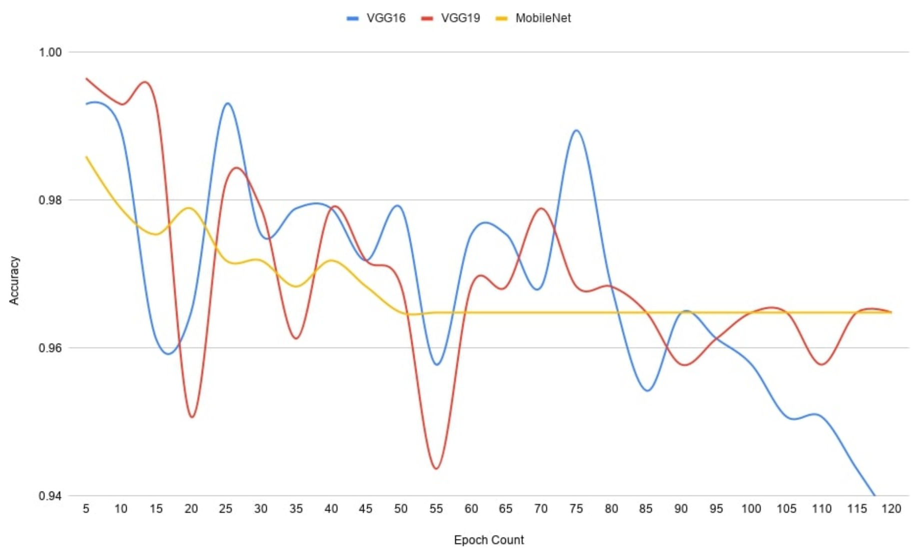

Figure 4.7: Accuracy of dependable models that reached at least 98% maximum accuracy based on epoch count. These do not include the ResNet variants.

We note that once ResNet variants are removed for their instability, every model reaches its peak accuracy at the 5-25 epoch range. Conditional on this range, the strongest models remain VGG19 and VGG16, but we can clearly see that their strength can fluctuate with small changes to epoch. Yet, MobileNet, MobileNet v2, DenseNet169, and DenseNet201 respectively perform the most consistently, while preserving most of the accuracy that we see in VGG19 and VGG16.

Some insight can be obtained by breaking down the accuracy into false negatives, which occur when a planet is present but the model does not identify it, and false positives, which occur when a planet is not present but the model identifies one. For our objective, false negatives are a much more severe error, as finding a planet and missing it is the worst possible situation. Alternatively, false positives would simply send back a picture of empty space to Earth, which would result in something mildly interesting, but nothing lost. Linking this information to our previous findings, ResNet variants continue to show instability, with some models having as many as 13 false negatives. Both VGG variants had no false negatives, meaning that they exhibit both extreme accuracy and reliability in detecting planets when they are actually present in the image. Lastly, we noted that DenseNet variants were very reliable in the 5-25 epoch range. In terms of false negatives, all DenseNet variants continue this stability with no false negatives.

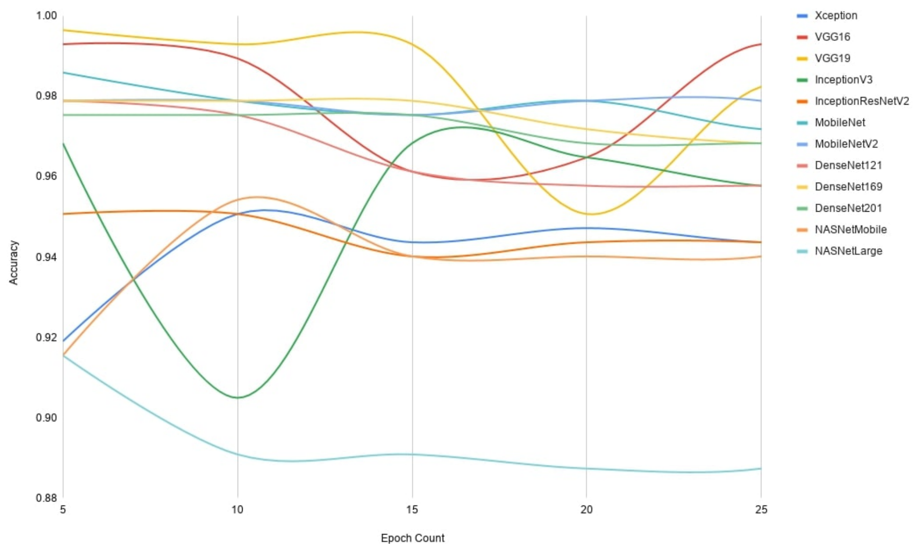

Figure 4.8: Accuracy of dependable models based on the 5-25 epoch range.

These results connect well to the previous section, where we outlined the unique circumstances that surround this particular problem. We note that we are training models on extremely specific objects in space, with unique features, patterns, colors, etc. Advanced models such as ResNet and Inception learn too well and proceed to analyze the extremely minute details in the training set. Then, when asked about other images that are unique in different ways, they struggle to find the connection. Meanwhile, less advanced models, such as VGG and MobileNet variants, do not lose that generality while learning.

4.4.1 Implications for future simulator training

Results show quite a few viable models, depending on whether we want extreme accuracy at the cost of variance (VGG variants), or dependability at the slight cost of accuracy (MobileNet variants).

Moreover, we note that this approach yields high accuracy with relatively few training images. With only 900 training images, we have achieved 98% accuracy with multiple models, giving us a wide variety of options depending on the situation. Thus, for future work, even small samples of simulated images can successfully train a neural network to detect real objects in space with extremely high accuracy.

4.5 Foundations for Future Work

We have expanded upon Bird et al. (2020) and have shown that a small subset of simulated images can produce extremely accurate predictions of real-world planets during an interstellar journey and beyond. This paper solidifies the overall outlook on optimization methods for exoplanet detection, while introducing many ideas that will open new and exciting problems in deep space exploration. The same methodology can be used in a wide variety of astrophysical (and other) applications, where subtle issues in both the temporal and spatial domain are critical to access, and make decisions for, low bandwidth return applications.

An upcoming paper will address categorization, which will expand the ideas of simulator-based detection to objects beyond exoplanets. In particular, we will further explore whether simulators can help train neural networks to distinguish between specific types of planets. What about specific types of stars, comets, asteroids, and even more interestingly, signs of life?

4.6 Conclusion

In our previous work, we showed how simulator images could be used to successfully train a neural network to identify real images of planets. In this paper, we delve into specific model optimization and obtain some fascinating results. First, multiple models are proven to have above 99% accuracy when trained only on simulator images and tested on real images of planets. This result completely supports a simulator-based training model for deep space journeys, allowing us to train large neural networks pre-flight on Earth. Second, we have shown that extremely high accuracy does not depend on large data sets in this niche problem. With under 1,000 training images, we have achieved over 98% maximum accuracy with six different models. Finally, we demonstrate that there exists both high accuracy and high stability models that can perform well with no false negatives.

References

Abadi, M., Agarwal, A., Barham, P., et al. 2016, TensorFlow:large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467v2 [cs.DC].

Bird, J., Petzold, L., Lubin, P., Deacon, J. 2020, Advances in Deep Space Exploration via Simulators & Deep Learning, New Astronomy, Vol. 84.

Canziani, A., Paszke, A., & Culurciello, E. 2016, An analysis of deep neural network models for practical applications, arXiv:1605.07678v4 [cs.CV].

Carrasco-Davis, R., Cabrera-Vives, G., Förster, F., et al. 2018, Deep learning for image sequence classification of astronomical events, arXiv:1807.03869v3 [astro-ph.IM].

Chetlur, S., Woolley, C., Vandermersch, P., et al. 2014, cuDNN: efficient primitives for deep learning, arXiv:1410.0759v3 [cs.NE].

Chollet, F. 2017, Xception: Deep Learning with Depthwise Separable Convolutions, arXiv:1610.02357 [cs.CV].

Deng, J., Dong, W., Socher, L., Li, K.L. & Fei-Fei, L. 2009, ImageNet: A large-scale hierarchical image database, 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

Greaves, J.S., Richards, A.M.S., Bains, W. et al. 2020, Phosphine gas in the cloud decks of Venus, Nat Astron (2020), https://doi.org/10.1038/s41550-020-1174-4

He, K., Zhang, X., Ren, S., & Sun, J. 2015, Deep Residual Learning for Image Recognition, arXiv:1512.03385 [cs.CV].

Honniball, C.I., Lucey, P.G., Li, S. et al. 2020 Molecular water detected on the sunlit Moon by SOFIA, Nat Astron (2020), https://doi.org/10.1038/s41550-020-01222-x

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv:1704.04861 [cs.CV].

Huang, G., Zhuang, L., vander Maaten, L., Weinberger, K.Q. 2018, Densely Connected Convolutional Networks, arXiv:1608.06993 [cs.CV].

Kulkarni, N., Lubin, P., & Zhang, Q. 2017, Relativistic spacecraft propelled by directed energy, arXiv:1710.10732 [astro-ph.IM].

Le, H., Gupta, R., Hou, L., Abousamra, S., et al. 2020, Utilizing Automated Breast Cancer Detection to Identify Spatial Distributions of Tumor Infiltrating Lymphocytes in Invasive Breast Cancer, arXiv:1905.10841[eess.IV].

Lubin, P. 2016, A Roadmap to Interstellar Flight, Journal of the British Interplanetary Society - JBIS, vol. 69, pp.40-72.

Lubin, P. & Hettel, W. 2020, The Path to Interstellar Flight, Acta Futura 12, 9-45.

Sandler, M., Howard, A., Zhu, M., Chmoginov, A., Chen, L-C. 2018, MobileNetV2: Inverted Residuals and Linear Bottlenecks, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp.4510-4520

Shallue, C.J., & Vanderburg, A. 2017, Identifying exoplanets with deep learning: a five planet resonant chain around Kepler-80 and an eight planet around Kepler-90, AJ, 155, 94.

Sheerin, T., Petro, E., Winters, K., Lozano, P, & Lubin, P. 2020, Fast Solar System Transportation with Electric Propulsion Powered by Directed Energy, Acta Astronautica 179, 78, Feb 2021.

Simonyan, K., & Zisserman, A. 2015, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs.CV].

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. 2016 Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv:1602.07261 [cs.CV].

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. 2015, Rethinking the Inception Architecture for Computer Vision, arXiv:1512.00567 [cs.CV].

Towns, J., Cockerill, T., Dahan, M., Foster, I., et al. 2014, XSEDE: Accelerating Scientific Discovery, Computing in Science & Engineering, Vol. 16, No. 5, pp. 62-74, Sept.-Oct. 2014, doi:10.1109/MCSE.2014.80.

Zhen, Y., Wang, L., Lui, H., Zhang, J., & Pu, J. 2018, Performance assessment of the deep learning technologies in grading glaucoma severity, arXiv:1810.13376 [cs.CV].

Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V. 2017 Learning Transferable Architectures for Scalable Image Recognition, arXiv:1707.07012 [cs.CV].

Zucker, S., Giryes, R. 2018, Shallow transits - deep learning I: feasibility study of deep learning to detect periodic transits of exoplanets, AJ, 155, 147.

Chapter 5

Novelty detection in deep space: The plausibility of template sphere training in place of simulator-based approaches

5.1 Introduction

Advancements in novelty detection and object detection have led to simpler means to advanced problems. In the case of space exploration, the main problem to address is a unique variant of object detection wherein training images do not exist but the difference between negatives and positives in testing images is seemingly obvious.

Bird et al. (2020) addresses attacking the problem from a very simply direction via a Grow When Required (GWR) (Marsland et al., 2002) network. Results show that such a simple technique is not only unrealistic for other processes that are needed while in space, but the accuracy is low and unreliable. In order to get cutting-edge results, Bird et al. (2021) uses a plethora of advanced modeling to determine how accurate and reliable current methods are. Results show that extremely high accuracy can be achieved with zero real images used, while providing multiple models that sacrifice small accuracy gains for more dependable predictions.

We return briefly to the original problem, which presents us with deep space exploration and very few useful real images to use in order to train a model for object detection. Simulators have proven to be an invaluable resource for solving this problem, and have led to amazing results for future missions. But, simulators can be expensive to make and keep up to date, and collecting simulator images is currently a slow process. In this paper, the need for simulators is tested by suggesting a possible alternative.

5.2 The Process

We test using the same advanced modeling scheme that was used in Bird et al. (2021), except that we not only test on simulator images but also on a much simpler and more easily produced template called *Spheres* or *Sphere Templates*. Note that we expanded the epoch range from the original in Bird et al. (2021) to allow for more comparison opportunities throughout testing. Both training image sets contained the same number of images at 915.

The background of these Sphere images is generated to imitate empty space as realistically as possible. We create a scattered star background, each star being randomly generated in terms of clustering density and size.

The foreground of these Sphere images is generated by randomly choosing a concept planet size and overlaying a white circle on the background. After this white circle is produced, an overlapping black circle is produced and placed randomly on the white circle to further produce a stochastic light angle from the nearby star.

The overarching intuition here is that although the very simple techniques that were tested early on produced poor detection, and advanced techniques produced excellent model prediction at the cost of computational energy, perhaps there is a middle-ground with Sphere Templates that can provide great accuracy while being easily reproducible and computationally inexpensive.

The main goal with this testing is to ensure that if Sphere Templates work, they provide accuracy and dependability that matches Simulators across models. Since model accuracy varies significantly depending on the image set, a single great result might not be reproduced in further experiments or in actual use. Therefore, in analyzing Sphere Template results, we must consider both accuracy and dependability across all models.

5.3 Results

The results are shown in both Table 5.1 and Figure 1.

Many interesting findings can be deduced from the results of Simulator and Sphere testing. First, the simulator accuracy has actually increased from including a broader set of possible epochs. In Bird et al. (2021), VGG16 & VGG19 were the top models, and that continues here as well, with the additional epoch ranges providing us with even more accuracy. In particular, VGG16's architecture fit the problem so well that 100% accuracy occurred with both training image setups. Since accuracy varies highly depending on different image sets, we must explore all models for dependability.

As we also found previously, MobileNet provided high accuracy, which held true for both Simulator and Sphere training sets. The lower accuracy for Spheres using MobileNetv2 implies that the model is not extremely strong, and we must investigate other options. DenseNet variants have proven to be quite dependable with Simulator images, which holds true here again with expanded epochs, but does not hold true for

Model Accuracy Comparison				
Model	Simulator Epoch	Simulator Accuracy	Sphere Accuracy	Sphere Epoch
Xception	4	93.33	82.5	1
VGG 16	1	100	100	1
VGG 19	1	99.63	85.92	1
ResNet50	2	93.70	62.96	1
ResNet101	4	90.37	70.37	1
ResNet152	1	82.96	63.33	1
ResNet50V2	4	94.81	92.59	1
ResNet101V2	1	96.66	92.59	1
ResNet152V2	4	97.77	91.85	1
InceptionV3	4	97.77	93.7	1
InceptionResNetV2	4	93.33	87.40	1
MobileNet	2	97.04	98.14	1
MobileNetV2	3	98.88	95.92	1
DenseNet121	4	98.14	97.40	1
DenseNet169	3	98.14	99.62	4
DenseNet201	4	97.77	93.33	1
NASNetMobile	3	92.59	93.33	2
NASNetLarge	1	92.59	93.70	1

Table 5.1: For each model, maximum epoch-based accuracy is shown alongside the corresponding epoch, which is done for both simulator image training and template sphere image training.

Spheres.

Overall, Simulators provide a plethora of great options that are dependable across architectures, with VGG variants having 99% or above, DenseNet variants having 97.7% or above, and MobileNet variants having 97% or above. Yet, Sphere testing proves that dependability and accuracy are both lacking. Although some ResNet variants have decent accuracy around 92%, some fall short at about 63%, not much better than a random guess. The poor performance of VGG19 makes the impeccable performance of VGG16 seem questionable for future experiments depending on the image set.

Table 5.2 shows a comparison between Simulator and Sphere accuracy. This difference table shows a stark difference in Simulator and Sphere accuracy, and points to

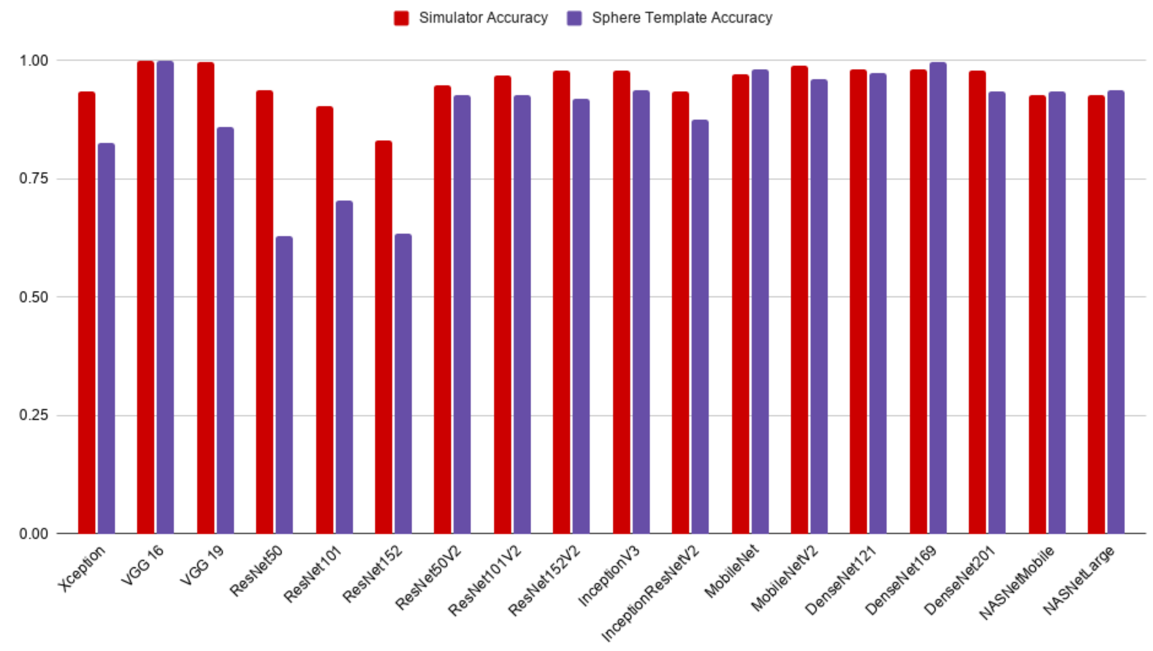

Figure 5.1: This graph shows the data from Table 1 in a more approachable and intuitive setting.

Sphere dependability being at unacceptable levels. For MobileNet, DenseNet169, NASNetMobile, and NASNetLarge, Spheres had better accuracy but only by a very small amount at approximately 1% difference. If this was consistently the case, an argument for Spheres being used would be completely viable. Yet, most models not only show a difference that favors Simulators, but the differences themselves are quite large. With an average difference of about 9.6% in favor of Simulators, with some models as high as 30.74%, Spheres are simply too inconsistent to defend.

Model Accuracy Difference	
Model	Simulator Accuracy - Sphere Accuracy
Xception	10.83
VGG 16	0
VGG 19	13.7
ResNet50	30.74
ResNet101	20
ResNet152	19.63
ResNet50V2	2.22
ResNet101V2	4.07
ResNet152V2	5.93
InceptionV3	4.08
InceptionResNetV2	5.93
MobileNet	-1.11
MobileNetV2	2.96
DenseNet121	0.74
DenseNet169	-1.48
DenseNet201	4.44
NASNetMobile	-0.74
NASNetLarge	-1.11

Table 5.2: For each model, Simulator Accuracy - Sphere Accuracy is shown. Positive numbers show that Simulator accuracy is better, which negative numbers show that Template accuracy is better. This is based on max-epoch accuracy.

5.4 Conclusion

We set out to find a middle-ground between our simple GWR network, which provided poor accuracy and limited functionality, and our optimized models, which provided stellar accuracy at the cost of computational complexity. A Sphere Template concept was introduced, which would make image collection easier and provide simpler template training images for the neural networks to learn.

Results show that Sphere Templates did not provide enough dependability or accuracy amongst the set of all models to be realistic. Some Sphere Template models produced an accuracy that barely exceeded a random guess. Despite some Sphere-based models minimally out-performing Simulator-based models, the overall results for Sphere Templates were inconsistent and significantly worse than their Simulator counterpart.

As discussed in Bird et al. (2020), even a small loss in accuracy can be extremely detrimental to a deep space mission. We therefore must conclude that Sphere Templates are not a suitable alternative to Simulators, and the slight increase in computational complexity must be dealt with in a different way in order to provide a system that maximizes accuracy.

References

Bird, J., Petzold, L., Lubin, P., Deacon, J. 2020, Advances in Deep Space Exploration via Simulators & Deep Learning, New Astronomy, Vol. 84.

Bird, J., Petzold, L., Lubin, P., Deacon, J. 2021, Model Optimization for Deep Space Exploration via Simulators and Deep Learning, arXiv:2012.14092.

Csurka, G. 2017, Domain Adaptation for Visual Applications: A Comprehensive Survey, arXiv:1702.05374

Kim, J., Lee, S., Oh, T.-H., & Kweon, I.S. 2017, Co-domain embedding using deep quadruplet networks for unseen traffic sign recognition, arXiv:1712.01907 [cs.CV].

Kulkarni, N., Lubin, P., & Zhang, Q. 2017, Relativistic spacecraft propelled by directed energy, arXiv:1710.10732 [astro-ph.IM].

Marsland, S., Shapiro, J. & Nehmzow, U. 2002, A self-organising network that grows when required, Neural Networks, 15, Issue 8-9, 1041-1058.

Chapter 6

Deep Neural Network Categorizer vs. Binary Decision Tree for Deep Space Object Detection

6.1 Introduction

In this paper, we will delve into a more complicated version of the concepts introduced in Bird et al. (2020) and Bird et al. (2021), namely the categorization of objects that might be seen during the NASA Starlight and Breakthrough Starshot program interstellar journeys. Previously, simulator images were shown to provide high accuracy and ease of training in a binary setting. Although the particular binary situation tested was the most important of all (planets vs. negatives), the remaining problem was to combine all astronomical objects during training and testing phases.

We introduce other objects besides planets, namely asteroids, comets, and stars. We also include three large planet subcategories, which are gas giants, rocky planets without water, and planets with water or liquid that resembles water, respectively referred to throughout this paper as gas giants, rocky planet, and water planet.

The goal moving forward is to provide a system of models that can take in an image and output a categorical label with the highest accuracy across all astronomical object types.

6.2 The Process

For the binary decision tree, we began by breaking the images down into two main chunks, *planets* and *not planets*. *Planets* are comprised of *gas giants*, *rocky planets*, and *water planets*, while *asteroids*, *comets*, *stars*, and *negatives* are considered *not planets*.

If the model determined the astronomical object to be a *planet*, we then test the same image for classification as a *gas giant*. If it is not a *gas giant*, the last step is determining if it is a *rocky planet* or *water planet*.

If the model determined the astronomical object to be a *not planet*, we then test to see if it is an *asteroid*. If it is not, we then continue the binary decision tree to test if it is a *comet*. If it is not a *comet*, it will be run through the final stage, which will determine if it is a *star* or a *negative*.

For the categorizer, we ran three different setups. First, the *complete categorizer* simply took in all images at once through a single model. Second, the *planets only categorizer* took in only planetary images, which consisted of all gas giants, rocky planets, and water planets. Third, the *not planets only categorizer* took in only images of objects that are not planets, which consisted of asteroids, comets, stars, and negatives, and evaluated them with a single model.

Each model was evaluated at epochs 1-5, as well as every multiple of 5 up until 100.

6.3 Results

The main results are shown in table format via Table 6.1 and in decision tree format via Figure 1. The split between *Planets vs. Not Planets* gave us similar accuracy to that found in Bird et al. (2021) while testing *Planets* against *Negatives*. In this case, our displayed testing accuracy is a more complicated problem of *Planets* vs. *Not Planets*.

Understandably, the most difficult part of this problem is distinguishing between planet types, which is confirmed by the accuracy of *Gas vs. Rocky+Water* and *Rocky vs. Water*. Conditional on making it to the *Planet* section, the *Categorizer: Planets Only* had an accuracy of 67.3611%, while making it to a *gas giant* (or possibly-habitable planet type) can be done via a binary decision tree with an accuracy of 84.7222%, and making it to a specific *rocky* or *water* planet can be done via a binary decision tree with an accuracy of 72.1020%. Essentially, no matter how deep into the *Planet* section of the decision tree you go, binary decision tree accuracy beats a categorizer.

Binary Decision Tree Accuracy		
Image Segmentation	Accuracy	Model
Planets vs. Not Planets	98.0198	DenseNet169
Gas vs. Rocky+Water	84.7222	DenseNet169
Rocky vs. Water	85.1063	VGG 19
Asteroid vs. Comet+Star+Negative	99.4219	ResNet152-v2
Comet vs. Star+Negative	91.4728	InceptionResNet-v2
Star vs. Negative	97.7011	InceptionResNet-v2
Categorizer: Complete	76.3406	InceptionResNet-v2
Categorizer: Planets Only	67.3611	NASNetLarge
Categorizer: Not Planets Only	87.2832	ResNet152-v2

Table 6.1: Results are shown for each image segmentation section. Accuracy column describes the highest accuracy achieved across all models, while Model column describes the respective model that achieved highest accuracy.

When dealing with *Not Planets*, the accuracy is significantly higher than we saw with *Planets*. In general, investigation further into the top tree section in Figure 1 deals with a low probability of finding something of interest, and investing energy into a *Not Planet* is not an efficient or optimal process. Assuming a *Not Planet* is detected, the *Categorizer: Not Planet Only* achieved an accuracy of 87.2832%, while the binary decision tree had an accuracy of 99.4219% to identify an *asteroid*, 90.9441% to identify a *comet* correctly, and 88.8534% to make it all the way to the end with a correct *star* vs. *negative* prediction. This shows that binary decision tree is the optimal setup for the *Not Planets* section as well.

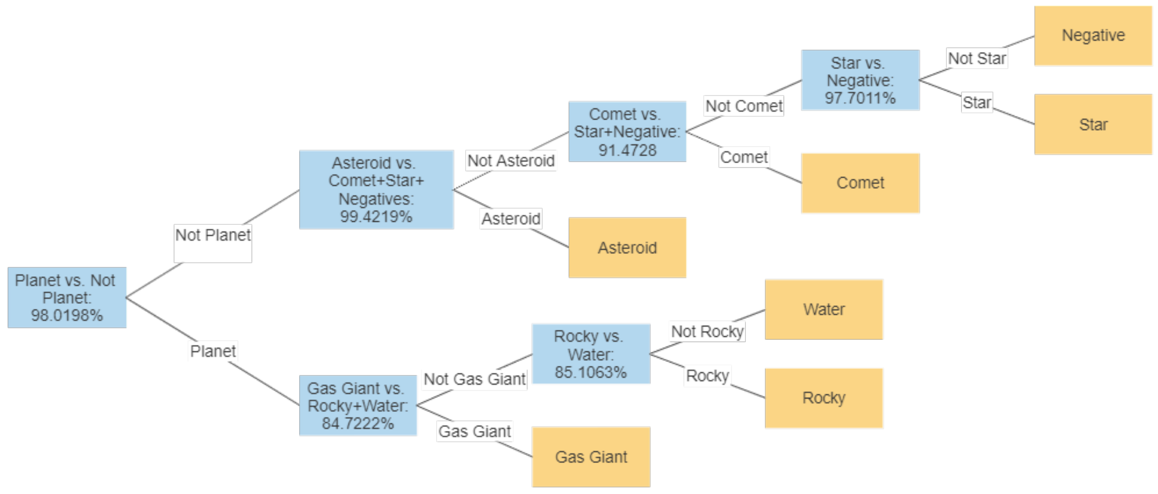
Planet vs. Not Planet: 98.0198%
Not Planet
Planet
Asteroid vs. Comet+Star+ Negatives: 99.4219%
Not Asteroid
Asteroid
Comet vs. Star+Negative: 91.4728
Not Comet
Comet
Star vs. Negative: 97.7011%
Not Star
Star
Negative
Gas Giant vs. Rocky+Water: 84.7222%
Not Gas Giant
Gas Giant
Rocky vs. Water: 85.1063%
Not Rocky
Rocky
Water

Figure 6.1: Results are shown in a binary decision tree diagram. Each binary model provides a small description and its respective accuracy.

As a finalization, we see if the *Categorizer: Complete* is worth using despite the *Categorizer: Planets Only* and *Categorizer: Not Planets Only* having worse accuracy in all situations when compared to a binary decision tree. The last part of each tree segment will have the most reduced accuracy, so if the *Categorizer: Complete* is worse than both of those, it is automatically the worse option. The non-conditional accuracy of *star* vs. *negative* is 87.0939%, while the non-conditional accuracy of *rocky* vs. *water* is 70.6762%.

Since both of these are greater than the 67.3611% achieved by *Categorizer: Complete*, we have exhausted all combinations and find that a custom binary decision tree is the optimal choice is all situations for this application.

6.4 Conclusion

We set out to test the general acceptance that deep learning has evolved to a point where a general categorizer is an acceptable solution to any object categorization problem. In our application, the astronomical objects were either very distance or very close to one another in likeness. For example, *asteroids* and *rocky* are extremely similar for detail-oriented filters, while *gas giants* and *comets* differ is almost every way imaginable. The separation of *planet* and *not planet* categories also served other functions, such as a *not planet* categorization automatically stopping the process to save energy.

Our results show that a custom binary decision tree gave us access to specific commands along the tree process, as well as the best accuracy for any and all astronomical objects that may be encountered during the journey.

Chapter 7

Conclusion

We began with a set of new challenges that arose from the Starlight program and the ability to perform fast interstellar travel. These included identifying stars, identifying new planets, extracting never-before-seen features, conceptually ranking these new planets against each other in terms of importance, understanding what importance means in the context of planets, and conserving energy while performing necessary tasks.

We started off by showing that a simple classification model would not suffice. Not only does it perform poorly, but it does not come with the range of tools that are needed for further processes down the line. We showed that while training on simulated images, accuracy on real images does not suffer, which is an amazing concept for such an application. Along with this, we provided results and justifications as to why simulators enable us to identify features that we have yet to observe in actual images. We also demonstrated how simulators can be utilized to save energy while ensuring that all necessary functions are completed.

Once the foundation for simulators had been laid, we introduced the concept that human knowledge gain can instruct machine learning algorithms to prioritize certain planetary features over others. This lead to the construct of planetary importance, which raised questions about which planetary images should be prioritized. In chapter 3, we showed that feature space can be a strong predictor of planetary importance, solidifying the idea that human experiments can be used to gain knowledge, which can in turn fuel an accurate model that sends the more "important" images back first.

Once the idea of using simulators had been verified and we knew that contextual knowledge could be used to inform a machine, model optimization was done to show that accuracy can get extremely high. Multiple models reached above 98% accuracy for *Planet* vs. *Negative* comparisons.

The remaining two chapters pertain to the uniqueness of the application and challenge current concepts in deep learning and object detection. Chapter 4 tested the theory that simple classification, yet something more advanced than that seen via GWR in chapter 1, can be used to replace simulators during classification. Our modeling tests confirmed that simulator training almost always achieved higher accuracy, but more importantly, it got high accuracy in a more dependable and robust fashion. Chapter 5 challenged the current deep learning method of categorization as the default method for multi-object models. As we showed, a categorizer was underwhelming compared to a custom binary decision tree for our application. Higher accuracy was achieved for all astronomical objects via a binary decision tree method.

Next steps for this research path include star detection phases and navigation, energy gain and management systems, and optical flyby correction software. Future research needs to ensure that the wafersats have the ability to check their positions, navigate via small course corrections, optimize energy use and gain energy while traveling, and correct possible image distortion due to their extremely high velocities.