# Lawrence Livermore National Laboratory Data Science Challenge

Wright, Kyle
kwright11@ucmerced.edu
Graduate Team Lead

Marquard, Dillon
dmarquard4@gmail.com

Ashokkumar, Kirankumar
kirankashok@gmail.com

Engel, Chloe
chloeengel01@gmail.com

Sanders, Ahmaree
asanders5@ucmerced.edu

Summer 2021

**Abstract**

There are many things that are unknown to the world about what lies outside of Earth, outside of our solar system, outside of our galaxy. Much research is conducted to understand what lives in our solar system that may impact life on Earth. New technologies are developed to adapt, learn, and optimize itself from data that has been and is being collected.

The main focus of this research is to detect Near Earth Objects (NEOs) in order to determine if there will be any potential threats to Earth. Near Earth Objects have the potential to cause great damage to planets and life in general. In order to help prevent future catastrophes, research on what is out in space must be conducted. In this work, a model is developed to detect, locate, and classify near earth asteroids that may potentially pose as a threat. In order to do this, the model searches through difference images (exposures from space) to locate candidates that may be asteroids. It will also preprocess the difference images so that the model will train with the data to better detect the asteroids in new images. This process of training the model is called machine learning. Machine learning is often preferred since machines can frequently detect things that are not obvious to humans.

Our model detects asteroids by dividing the image into smaller segments using a footprint algorithm (used to detect bright spots in the image) and an Gaussian filter (used to smooth noise from the image) to select potential asteroid candidates from within an image. This method of detecting and selecting asteroid candidates converts the problem into one of correctly classifying the asteroid candidates. Once the candidates are detected, classification is the next step. Our model implements two convolutional neural networks (CNNs), a type of machine learning.

The first CNN is used to classify the initial group of candidates, eliminating many non asteroid candidates while retaining as many true asteroid candidates as possible. The candidates classified as asteroids by the first CNN are then classified again by the second CNN (a form of boosting). This second CNN specializes in removing the candidates incorrectly classified as asteroids. The end result is a list of candidates very likely to be asteroids.

The overall result of the process from selecting candidate asteroids from the difference images to offering a prediction of asteroids is a precision of over $0.97$ and a recall of $\sim 92\%$. This means that the model presented here correctly detects over $90\%$ of asteroids in an image with only $3\%$ of the candidates selected by our model being "false" asteroids. We believe that this work offers a proof of concept that a model consisting of an algorithm to select candidates from a difference image along with using a combination of 2 CNNs to classify candidates as asteroids and non asteroids is a feasible way to detect asteroids in a difference image. In addition, if offers a foundation than can be expanded upon to further our ability to detect asteroids that may pose a threat.

## 1 Introduction

Asteroids can cause tremendous damage to city populations which will result in the loss of lives and millions of dollars worth in damages. Early detection and intervention is a critical factor to stop asteroids from harming society. Current

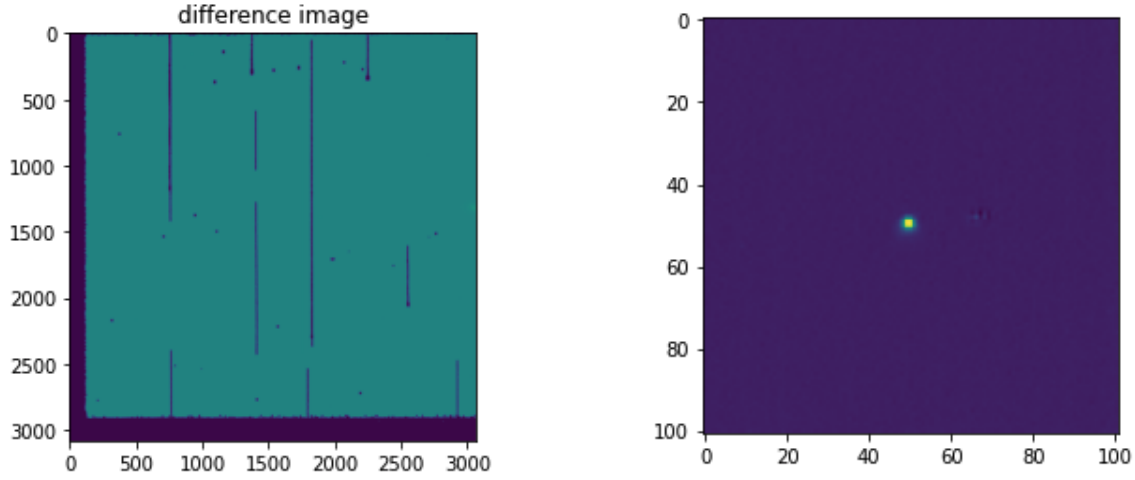
methods of asteroid detection are using high spec cameras and monitoring specific asteroids that have been logged by NASA. Computation about the trajectory of the asteroid if it could possibly hit major cities is also noted. The program currently monitors asteroids that are well known. However, these methods don't alleviate the risk of undetected asteroids in the far reaches of space. Using machine learning, we can detect asteroids that are otherwise difficult for humans to distinguish from the background of space. These techniques can sift through thousands of images and potentially produce a high accuracy of classification of these space objects. Using such methods, our knowledge and identification of asteroids can be increased in a more streamlined and efficient manner.

The images that have been provided to us are difference images taken straight from Zwicky Transient Facility (ZTF). These images have synthetic asteroids and are injected into real science images from the ZTF telescope in which we can train our models on. There are 20 injected asteroids in each image. Currently there are 10 files with 100 images inside each file. The size of the asteroids are extremely difficult to tell and this will reflect as a real world implication of how they are naturally seen in space.

Thus, the main goal of this work is to provide a proof of concept for a method to detect asteroids from a difference image. To accomplish this, we will identify candidates as potential asteroids from the difference images, cut out sub-images centered around our candidates, and use these image cutouts to train machine learning models (primarily convolutional neural networks) to differentiate between asteroids and non-asteroid candidates. We hope that our work grants further insight into improved methods of detecting asteroids.

## 2 Image Data

The dataset is comprised of artificial asteroids injected into difference images from ZWT. Each difference image is formed by comparing images of the same region of space taken at different times and showing the differences between the images. One difference image from file 0 is shown in Figure 1(a). There are 10 npz files that each contain 100 difference images of size 3080×3072px. The injected asteroids in each image are approximately 5×5px. This means that are 20000 asteroids available to use for training and testing our model. There is also information data for each image, providing the magnitude, length, angle, and location of each asteroid. The location of each asteroid is provided as a pair of coordinates in the image which create an approximately $100 \times 100$px box with the asteroid located at the center. An example of the cutout using the location coordinates is shown in Figure 1(b).




*(a) Difference image from npz file 0.*

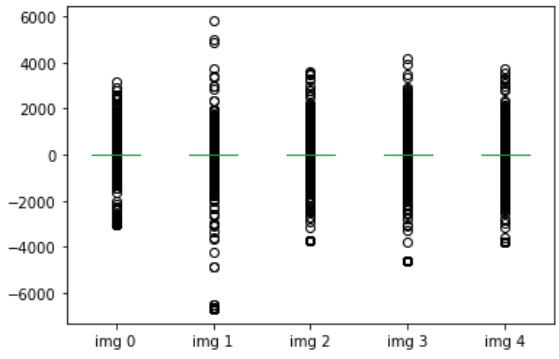
*(b) A single asteroid from npz file 0*

*Figure 1*

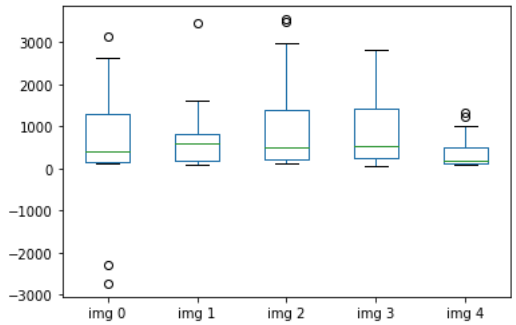
Figure 2 shows boxplots formed using pixels from the first 5 images in file 0. The boxplots shown in Figure 2(a) are

created using the values at every pixel from each of the 5 images. These boxplots have no apparent skews to the positive or negative directions and appear roughly symmetric. In addition, the vast majority of the data points in are also centered around zero. However, there are still many outliers that deviate from the point 0 which can be noted by circles.

Figure 2(b) is a boxplot formed of the maximum values of the nine central pixels from each of the 20 asteroid sub-images corresponding to the previous 5 images. It can be noted that the boxplots show a skew towards the positive values and are centered around 500, with relatively few outliers. Comparing the two plots supports the hypothesis that asteroids could perhaps be distinguished by pixel value since the majority of the maximum value asteroid pixels are likely to be greater than most other pixels in the image.




*(a) Boxplots of the pixel values in 5 images.*




*(b) Boxplots of the maximum pixel values of asteroids in 5 images.*

*Figure 2*

# 3 Model Data

## 3.1 Footprint Algorithm

Our initial approach was breaking the image into 50x50px non-overlapping cutouts and classifying each sub-image. However, this approach made it hard to label images based on the coordinate data due to some asteroids being split into separate images, and those images being falsely classified as non asteroids. This is despite the fact that there actually was part of an asteroid in the image. The training data generated was of a poor quality and demanded another approach.

The footprint algorithm iterates over the image in $50 \times 50$px with a stride of 50px. Then, for each cutout, it selects the coordinate of the pixel with the highest value. These coordinates are then used to make $21 \times 21$px cutouts centered around those highest value coordinates. These $21 \times 21$px cutouts are used as the inputs for our classifying models.

Considering the errors from the initial process, we opted to use the footprint algorithm to generate better training data. The benefits from using the cutouts were immediately apparent during model training and testing. The main benefit came from being able to center asteroids, which would only happen by coincidence with our first approach. This also ended up having the effect of improving the predictive power of our models down the line.

Finding that our algorithm located the majority of the asteroids, we investigated the size of the kernel that iterates over each image, and found that 50px was a good choice when considering time and space complexity along with the percentage of asteroids successfully located within an image. In addition, we found that a kernel size of less than 30px both took too long to be practical and negatively impacted our model due to the overwhelming number of false positive candidates generated. However, it was untested whether our boosted model, described in a later section to specialize in removing false positives, could have eliminated enough of the false positives to justify the small percentage increase in asteroids successfully located.

In practice, we found, on average, 95% of the asteroids to be within the candidate set determined by our footprint algorithm. We can increase the accuracy by decreasing the size of the kernel–shown in Figure 3–or stride at the cost of time and increased numbers of false positives. As a general rule of thumb, if you decrease the size of the kernel by a half, then you will quadruple both the number of false positive candidates and the time to generate the coordinate list. In that regard, a better method for finding candidates should be investigated further with this rudimentary approach as a baseline.

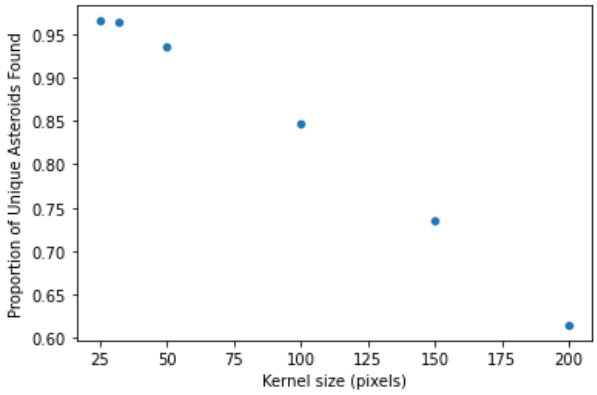



*Figure 3: Proportion of asteroids located vs kernel size.*

The footprint algorithm performs fairly well, but investigation into the missed classes lets us better understand what characteristics of the asteroids are related to our algorithm not identifying them as candidates. As seen in Figure 4, asteroids with a lower magnitude are more likely to be identified as candidates by the footprint algorithm. This is expected because our algorithm uses the brightness values of each pixel to select candidates and higher magnitude asteroids have lower brightness values. That means our algorithm will inherently have a harder time finding dim asteroids.

To understand Figure 4, the total height of the bars shows the proportion of asteroids that are within different ranges of magnitudes. The portion of the bar colored orange represents the proportion of asteroids located by the footprint algorithm within that range. The portion of the bar colored blue similarly shows the proportion of asteroids not located by the footprint algorithm within that magnitude range.

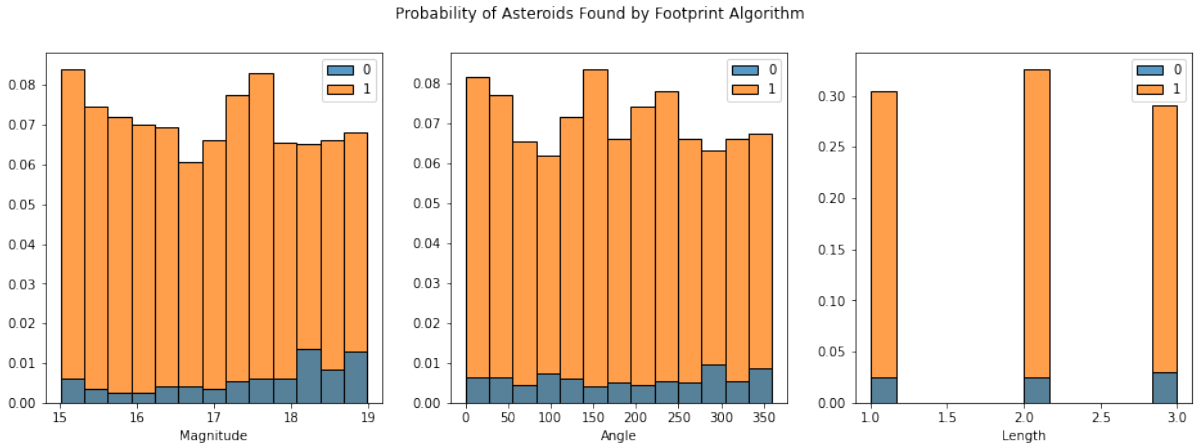
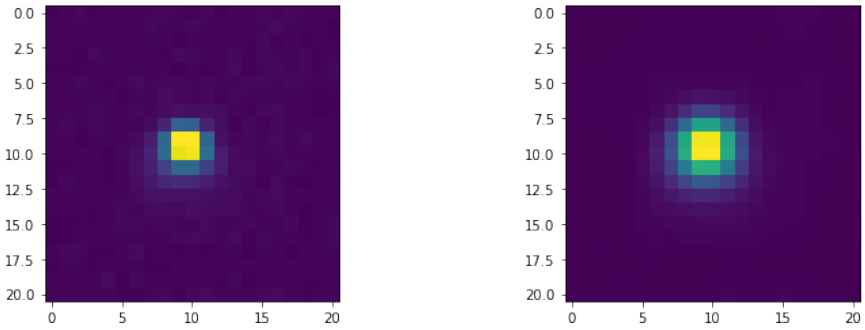



*Figure 4: Footprint Algorithm Performance*

## 3.2 Filters

One issue we encountered with raw data were artifacts. We utilized a Gaussian Filter to smooth the image (Figure 5) and found that the filter overall improved our method of classification both by improving the performance of our neural network in classifying image cutouts and by improving our footprint algorithm's ability to choose asteroids as candidates. It would be interesting to explore other methods of filtering, but this approach was based on James Buchanan's galaxy blending filter.



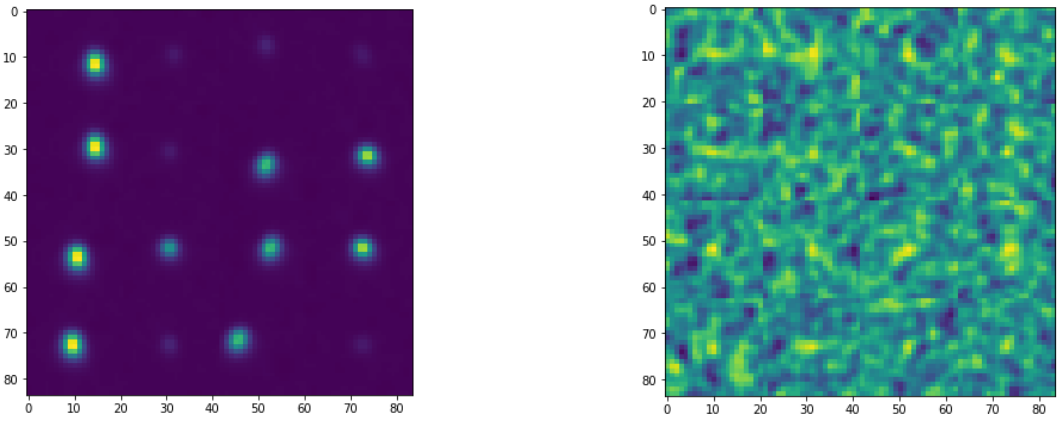

*(a) Raw image* *(b) Filtered image*

*Figure 5*

However, one issue we had with filtering was smoothing over high magnitude asteroids. In that regard, the sigma we chose for our filter was a balance between smoothing noise, but not smoothing so much that the higher magnitude asteroids wouldn't be found by our footprint algorithm. In addition, it was discovered that our attempts at normalizing the data similarly resulted in our footprint algorithm not detecting these asteroids as candidates. Finally, we attempted to manipulate the image by computing the density of brightness both locally and globally. This was also unsuccessful in improving our ability to detect asteroids overall. We suspect that these failed attempts are due to the distribution of the pixel values combined with the fact that some high magnitude asteroids are included among the negative outliers. However, it seems likely that there is a technique, perhaps involving a more sophisticated way of choosing image segments and applying a method to each segment, that could result in better performance.

## 3.3 Data Visualization

We also wanted to visualize the centered $21 \times 21$px cutouts of candidate asteroids to determine whether it is plausible that our algorithm is producing images where the asteroids can be differentiated from non asteroids. Figure 6(a) shows the positive class grid containing asteroids of varying magnitudes. We noticed that the dimmer asteroids are harder to discern from background noise, but it is promising that our algorithm is also selecting these asteroids as candidates. We can see from the negative class grid–Figure 6(b)–that the noise is fairly different from that of the positive class grid. With that in mind, it seems justified to attempt to differentiate these classes using our model.




*(a) Positive Class Grid* *(b) Negative Class Grid*

*Figure 6*

### 3.4 TSNE (3-components)

We can visualize a 3D embedding of the data to see that the positive class has a bias towards one side and the negative class tends to be further from that side; however, there is no definitive boundary between the classes, so we should expect some error with support vector machines or decision tree models. It would be interesting to see other methods of reducing the dimension of the data and how far the boundary between the classes could get. We can see how well the embedding separates the two classes by using k-Means in section 3.5.

The means' of the two groups are relatively far from one another and just predicting on the distance from the two means would prove fairly effective. We can also quantify this with k-Means. In particular, the region where you see a blend is why a support vector machine would prove effective by maximizing the boundary region.

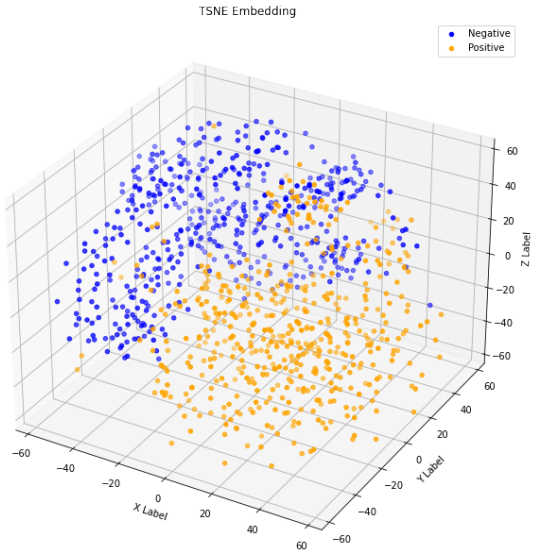


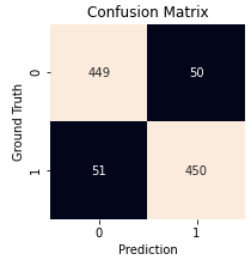

*Figure 7: 3D Embedding*

### 3.5 k-Means (2-clusters)

We want to understand how well the two classes are spatially separated using the embedding from TSNE. We can use k-Means to cluster the data into two classes based on the 3D embedding to see how well TSNE performs.

TSNE performed with around 75% accuracy justifying that the data can be spatially separated and the use of more sophisticated clustering techniques would be worth investigating.

In particular, an support vector machine should produce even better results with the use of non-linear kernel to increase the boundary distance between both classes.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.898 | 0.900 | 0.899 | 499 |
| 1 | 0.900 | 0.898 | 0.899 | 501 |
| accuracy | | | 0.899 | 1000 |
| macro avg | 0.899 | 0.899 | 0.899 | 1000 |
| weighted avg | 0.899 | 0.899 | 0.899 | 1000 |

*Figure 8: k-Means Performance on embedding*

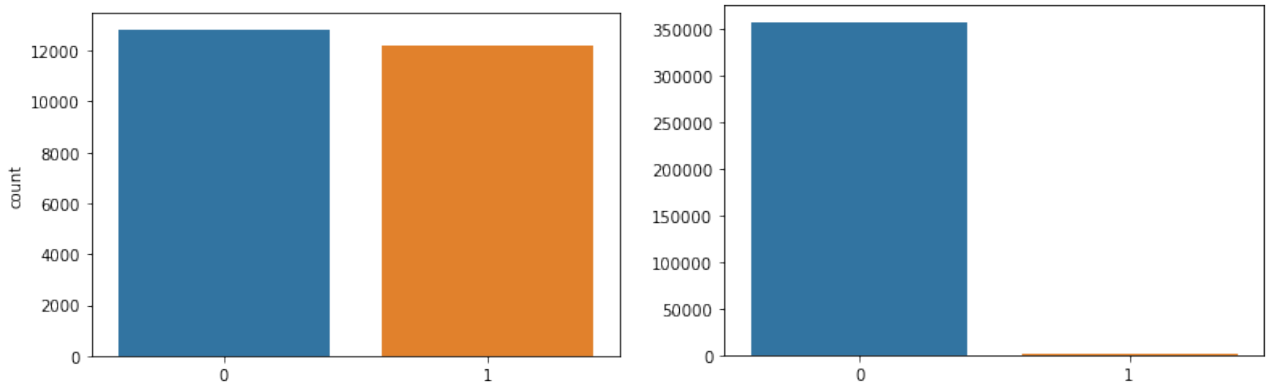
### 3.6 Training and Testing Data

We chose to validate our model using a train/test split of the data. Supposing that different images within the same file may be related in some way, we decided to split the training and testing data based on the original image dataset file numbers. Image datasets with file numbers 0-6 were chosen as our training set and image datasets with file numbers 7 and 8 were used when testing our models.

After implementing our footprint algorithm on the training data, the result is approximately 12000 positive class (asteroid) image cutouts and 2,230,000 negative class (non-asteroid) image cutouts. Keep in mind that due to using the

footprint algorithm, only $\sim 95\%$ of asteroids are found for use in the positive class. Also, note that the ratio between negative to positive image cutouts is almost 200:1. Simply shuffling our training data cutouts before dividing it into batches to feed into our network would result in an overwhelming number of negative class inputs into our model per batch.

To account for this, we created a new training set for each epoch run when training our neural network models. The new training set was created by including all positive examples and a random sample of the negative examples of a similar size, seen in Figure 9(a). Both this distribution of the classes and introducing the model to an increased variety of negative class examples resulted in better trained models. In hindsight, it would have been more efficient to iterate through a shuffled false negative set instead of sampling the entire negative set for each epoch. The test set was unmodified and its distribution is shown in Figure 9(b). In these figures, 0 represents the negative class and 1 represents the positive class.




*(a) Train Set Distribution*

*(b) Application Test Set Distribution*

*Figure 9*

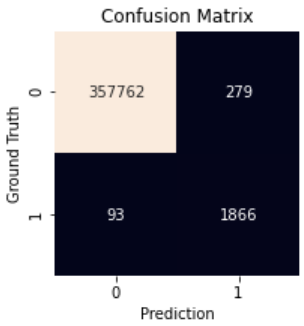
To provide labels for our image cutouts, we chose a label of 1–asteroid–if the image cutout was centered within 1 pixel of the actual location of the center of the asteroid and a label of 0–non asteroid–otherwise. These labels are used when training our models to define which class they belong to.

# 4 Modelling

## 4.1 Random Forest

We performed some preliminary testing using a random forest. With spatially relevant data a decision tree would be worth investigating. We did not have enough time to explore principal component analysis (PCA), so we flattened the input for 441 features (corresponding to the $21 \times 21$px image cutouts) and max depth of 12 decision nodes. Despite the lack of PCA, the random forest model still performed fairly well with an f1-score of 91%, Figure 10. In particular, the model seemed to struggle with false positives even after using a CNN to boost the model. The convolutional neural network used to boost the model is the baseline CNN, as seen in Figure 11.

It was interesting that the model could even perform well with the flattened input as we had initially suspected that we would need to use PCA to get palatable results; however, with a $21 \times 21$px cutout size that does not seem to be the case. Note that while we were somewhat surprised by how well the random forest model performed, it was outperformed by the CNN model.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 0.999 | 0.999 | 358041 |
| 1.0 | 0.870 | 0.953 | 0.909 | 1959 |
| accuracy | | | 0.999 | 360000 |
| macro avg | 0.935 | 0.976 | 0.954 | 360000 |
| weighted avg | 0.999 | 0.999 | 0.999 | 360000 |

*Figure 10: 3D Embedding - Boosted by CNN*

## 4.2 Convolutional Neural-Network(s)

### 4.2.1 CNN Overview

Convolutional Neural Networks (CNNs) are composed of three layers: convolutional, pooling, and non linear activation layer. Each of the layers play a role in the classification of images. The convolutional layer traverses each pixel, summing the surrounding 9 pixels multiplied by the corresponding kernel entry to output a new image that specifies the features of the original image. The pooling layer ideally keeps the important information and eliminates the noise. The linear layers come after the mixed combination of convolution and pooling layers. The linear layers are fully connected layers. The convolutional and pooling layers are used in the generation of features from the image, whereas the linear layers are used to classify the image based off of the detected features.

For our CNN model we prioritized using Pytorch as our main library as it provided many useful functionalities that we wanted to implement. Adam seemed to be the better option for optimization and produced better results than stochastic gradient descent (SGD). A learning rate of 0.0001 was chosen as it descended the gradient in a more efficient way without over stepping the minimum point as a learning rate of 0.001 did. Note that more epochs were necessary using the smaller learning rate. The smaller learning rate is worth the trade off in time taken. As a side effect of how we select the training data for each epoch, an increased number of epochs also allowed our model to ”see” a greater variety of negative class examples. A weight decay of 0.00001 was chosen for our model. Since the problem is a classification problem, where the objective is to correctly classify images into those of asteroids or non asteroids, we chose the cross-entropy loss function to train our CNNs.

Figure 11 is a visual representation of our baseline CNN architecture. The input to network is represented by the $21 \times 21$ square, while the output is the final column with a height of 2. Our baseline network architecture consists of 3 convolutional layers to generate features and 3 linear layers to classify based on the generated features. Each of the 3 convolutional layers, appearing as boxes, uses a $3 \times 3$ kernel with a stride of 1. A padding of 1 is chosen for the first convolution. The linear layers, appearing as columns, are fully connected layers.

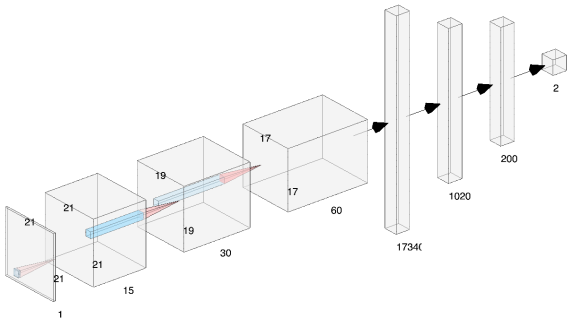
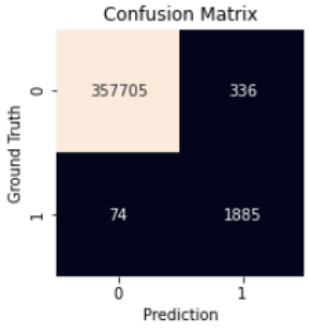



*Figure 11: Baseline CNN architecture.*

### 4.2.2 Single CNN

At the beginning of the Data Science Challenge (DSC) with Lawrence Livermore National Laboratory, we gained experience using a CNN to classify images into two classes, stars and galaxies, where the images were centered on the respective star or galaxy. This gave us confidence that, using the centered image cutouts as our training data, we could potentially successfully train a CNN to distinguish between the positive and negative classes. We began by training a CNN and analyzing our test results. We will compare using different network architectures to our benchmark architecture, seen in Figure 11, to gain insight into how changing network architectures can improve model performance.

Due to the over representation of negative examples found as candidates by our footprint algorithm, seen as the upper left number in the confusion matrix in Figure 12, we have chosen to focus on the precision and recall statistics associated with an asteroid being present to represent how well our model is performing. The precision statistic describes the percentage of the candidates our model predicts to be asteroids are actually asteroids, and the recall statistic represents the percentage of asteroids our model labels as asteroids. The f1-score is a combined score to interpret both recall and precision simultaneously.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 0.999 | 0.999 | 358041 |
| 1.0 | 0.849 | 0.962 | 0.902 | 1959 |
| accuracy | | | 0.999 | 360000 |
| macro avg | 0.924 | 0.981 | 0.951 | 360000 |
| weighted avg | 0.999 | 0.999 | 0.999 | 360000 |

*Figure 12: Baseline CNN architecture performance.*

The pipeline begins with loading the raw images into memory. Then we pass each image through our Gaussian filter to smooth the image. We then use our footprint algorithm to generate a coordinate list of candidates. Those candidates are then used to make cutouts to then be classified by our model(s). The results are mapped to the coordinates and classification by index, so they can be easily parsed for a list of positive (asteroid) classifications using our CNN. After generating our predictions, we can compare our results to the known truth for the locations of the injected asteroids to determine how well our model is performing. A visualization of this pipeline is shown in Figure 13.

Based on the recall and precision statistic in Figure 12, our model is overall doing a decent job of classifying both positive and negative examples. However, due to the sheer number of negative cases, the result is a large number of false

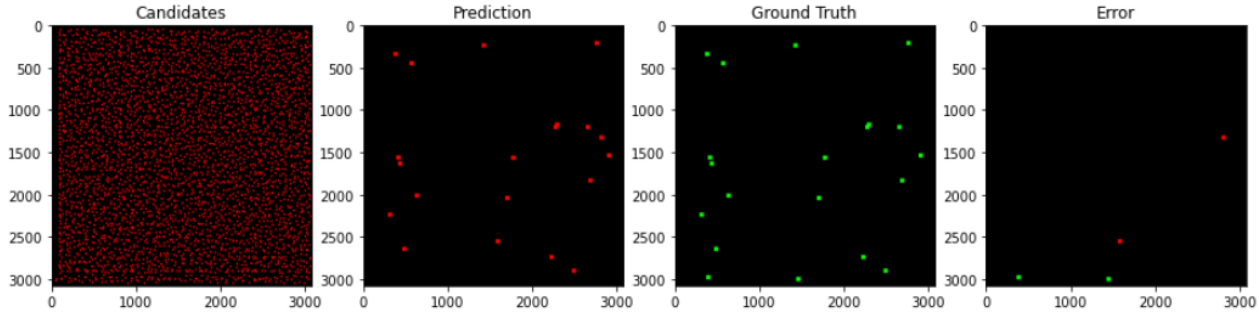



*Figure 13: Single CNN data pipeline visualization.*

positives. Examples of the false positives we're seeking to reduce are shown in Figure 14.

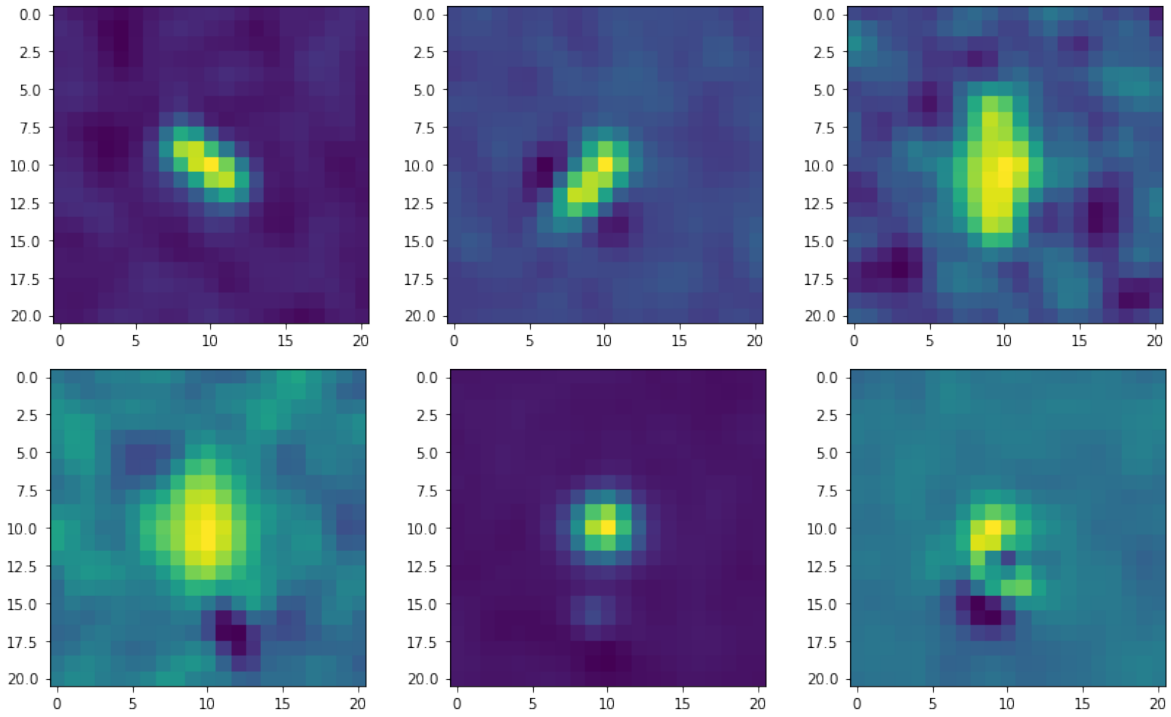
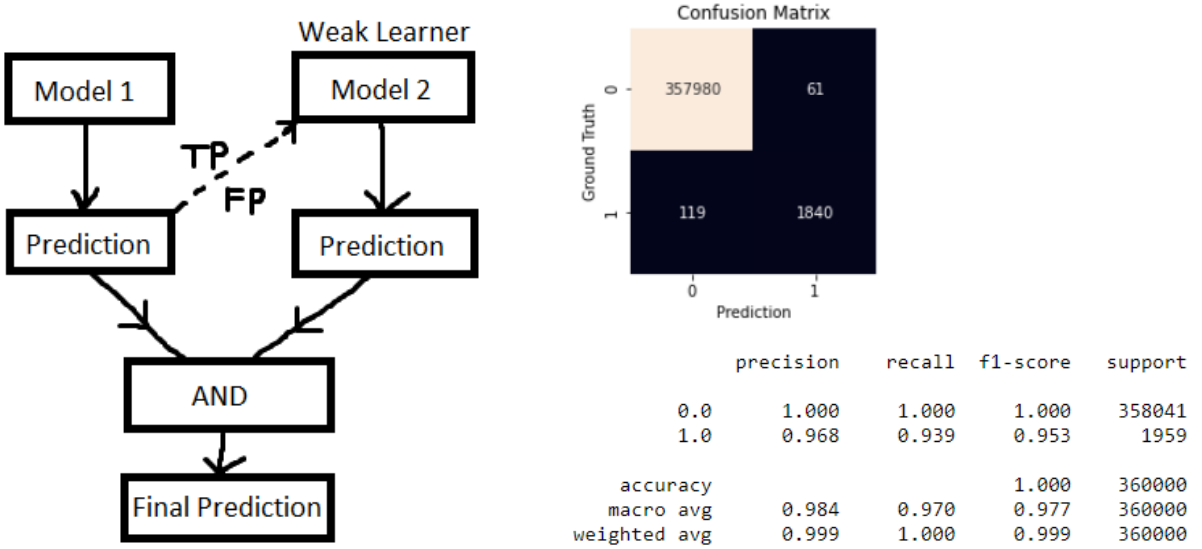


*Figure 14: Sample of false positive examples using benchmark CNN model.*

#### 4.2.3 Boosting with CNN

One issue we found with our first model was a high proportion of false positives. To help combat the number of false positives, we trained a second neural network on the first CNN's false positive and true positive outputs in an effort to eliminate more false positives. By training a model to differentiate between asteroids and potential artifacts, our model performed far better. This approach greatly reduced the proportion of false positives in the testing sets while only marginally increasing the number of false negatives, Figure 15(b). The vast majority of false positives similar to those shown in Figure 14 were successfully removed using our boosted model. Note that while the same network architecture was used for the boosting CNN during all of our training and testing, this is unlikely to be optimal and it is likely that further experimentation and study could be done to produce a network architecture for the boosting CNN that further

improves performance.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 1.000 | 1.000 | 358041 |
| 1.0 | 0.968 | 0.939 | 0.953 | 1959 |
| accuracy | | | 1.000 | 360000 |
| macro avg | 0.984 | 0.970 | 0.977 | 360000 |
| weighted avg | 0.999 | 1.000 | 0.999 | 360000 |

*(a) Model Architecture*

*(b) Test statistics using baseline model for first and boosting CNN.*

*Figure 15*

Using the boosted CNN model, candidates are still chosen using the footprint algorithm. This is followed by our first CNN classifying candidates as either asteroids or non asteroids. All candidates our first model classifies as asteroids are fed into our second model, which then makes its own prediction. If both models agree that a candidate is an asteroid, then our full model predicts that this candidate is an asteroid. Otherwise, the candidate is labelled as a non asteroid. This can be visualized in Figure 15(a). An updated pipeline for this process can be seen in Figure 16.

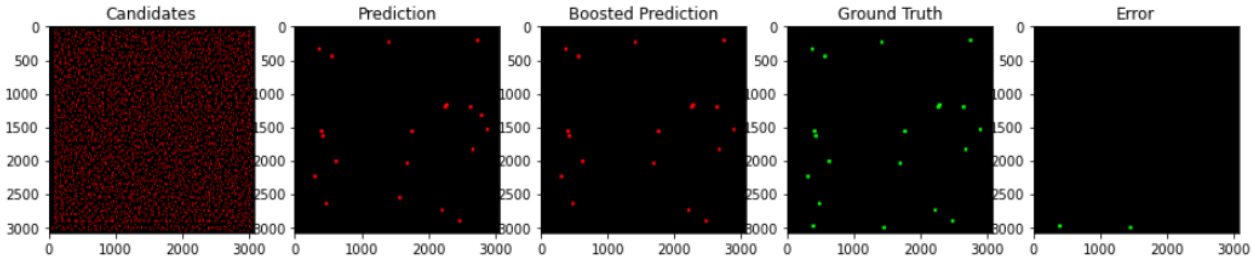


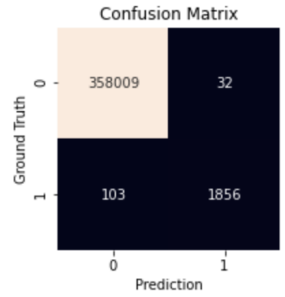

*Figure 16: Boosted CNN data pipeline visualization.*

### 4.2.4 Different CNN architectures

Multiple models, using differing network architectures, were run through our data pipeline. A few of these model architectures are shown below in Figures 17 - 19, alongside with their corresponding test statistics. By analyzing how changing network architectures changes model performance, we can gain insight into how we might improve upon our baseline network architecture.

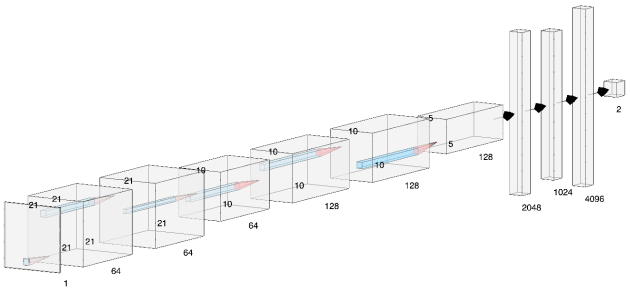
After much experimentation, some of which is shown from the models in Figures 17-19, we gained insight into portions of the architecture that seem to improve performance. Some form of pooling in middle or later layers seemed to improve performance, particularly average pooling. It remained unclear whether max pooling resulted in any performance in-

crease. Although increasing the number of connections within a network naturally improves performance, some specific places resulted in greater impact.



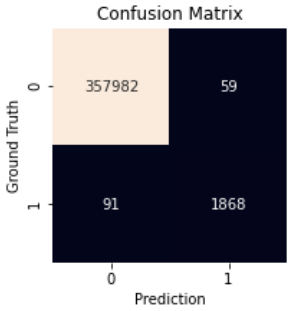

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 1.000 | 1.000 | 358041 |
| 1.0 | 0.983 | 0.947 | 0.965 | 1959 |
| accuracy | | | 1.000 | 360000 |
| macro avg | 0.991 | 0.974 | 0.982 | 360000 |
| weighted avg | 1.000 | 1.000 | 1.000 | 360000 |

*(a) Test statistics using altered VGG16, Figure 17(b), for first and second CNN.*



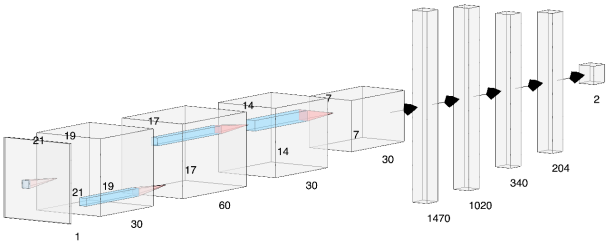
*(b) Version of VGG16 model by Kiran.*

*Figure 17*




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 1.000 | 1.000 | 358041 |
| 1.0 | 0.969 | 0.954 | 0.961 | 1959 |
| accuracy | | | 1.000 | 360000 |
| macro avg | 0.985 | 0.977 | 0.981 | 360000 |
| weighted avg | 1.000 | 1.000 | 1.000 | 360000 |

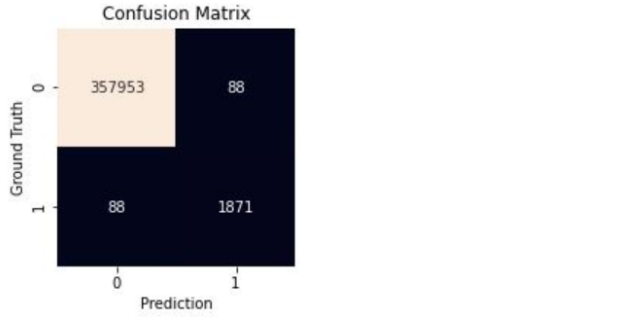
*(a) Test statistics using Figure 18(b) for first and second CNN.*



*(b) Model Architecture by Dillon.*

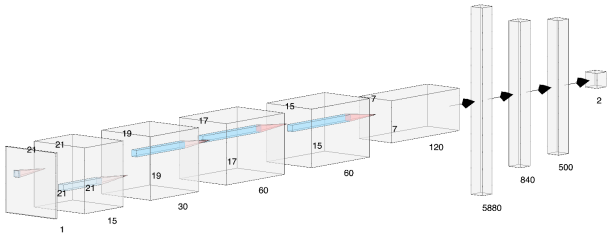
*Figure 18*

Our neural networks particularly liked having a large number of out channels after the first convolution. This seems to indicate that the network needs the ability to generate features directly from the image and that many of the most important final features generated by the convolutional layers are coming from different channels of the first convolutional layer. We also noticed steep diminishing returns on using more than 4 convolutional layers (not including pooling). However, based on the results from Figure 19, we also suspect that the number of critical features is relatively low and that with more time and experimentation, we could significantly reduce the number of final features produced by our




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 1.000 | 1.000 | 358041 |
| 1.0 | 0.955 | 0.955 | 0.955 | 1959 |
| accuracy | | | 1.000 | 360000 |
| macro avg | 0.977 | 0.977 | 0.977 | 360000 |
| weighted avg | 1.000 | 1.000 | 1.000 | 360000 |

*(a) Test statistics using Figure 19(b) for first and second CNN.*




*(b) CNN model architecture by Chloe.*

*Figure 19*

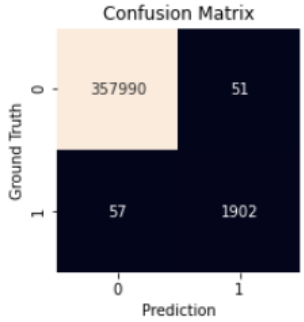
convolutional layers in our final model with similar results. That said, without the time we can simply use more final output channels to ensure we grab those more important features. Finally, we discovered that features smaller than size $3 \times 3$ seemed to perform less well.

We also found that increasing the number of neurons in the final linear layer steadily improved the model up until $\sim 500$ neurons. After that, further increases resulted in diminishing returns. This leads us to believe that, given the features our convolutional layers are generating, our model prefers around 500 degrees of freedom when making its final decision on which class to categorize an image cutout. This seems to indicate that breaking into the black box to infer how the model is reaching its decision is difficult. That said, we were able to make some observations to use in the creation of our final model.

# 5 Final Model and Results

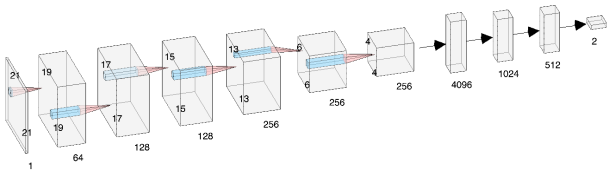
For our final model, we incorporated a variety of modifications that performed well. We saw the best results with 4 convolutional layers in addition to 2 average pooling layers. The average pooling was applied after each of the middle two convolutions. Each convolution and the first pooling uses a kernel size of 3, a stride of 1, and no padding. Our second pooling layer used a kernel size of 2, a stride of 2, and no padding. The network architecture, including the number of input and output channels used, can be observed in Figure 20(b). The number of neurons in the linear layers can similarly be observed, and we did choose to use 512 neurons in the final linear layer. This is in line with our experience testing other network architectures and discussed at the end of the previous section.

We can see from Figure 20(a) that our final model performs markedly better than our baseline model, with a recall and precision of over 0.97. Although this performance is one reason this network architecture was chosen for our final model, another consideration was that the recall of this model for the first CNN–the one sorting the candidates–produced a recall of over $99\%$ on our test data. This was the highest recall that we observed when filtering candidates and ensured that as few false negatives as possible would be present before using a boosting CNN. It should be noted that the recall presented throughout this document only considers asteroids located by the footprint algorithm. If we take into account that only $\sim 95\%$ of asteroids are located by the algorithm and the small portion of asteroids that are located multiple times, our true recall is expected to be closer to $90 - 92\%$, as opposed to the $97\%$ observed in Figure 20(a).




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.000 | 1.000 | 1.000 | 358041 |
| 1.0 | 0.974 | 0.971 | 0.972 | 1959 |
| accuracy | | | 1.000 | 360000 |
| macro avg | 0.987 | 0.985 | 0.986 | 360000 |
| weighted avg | 1.000 | 1.000 | 1.000 | 360000 |

*(a) Test statistics using final model for first and boosting CNN.*



*(b) Final model CNN architecture.*

*Figure 20*

We were also curious to see whether the magnitude of an asteroid had an impact on our ability to correctly classify it, assuming that our footprint algorithm located the asteroid. To analyze this, we created image cutouts for every asteroid in our test set and ran it through our final model. The bar chart in Figure 21 shows the percentage of asteroids misclassified as non asteroids for the shown magnitude ranges. It can be observed that higher magnitude asteroids are more likely to be misclassified, with the highest magnitude asteroids being roughly twice as likely to be misclassified as the lowest magnitude asteroids, at $\sim 8\%$ and $\sim 4\%$ respectively.

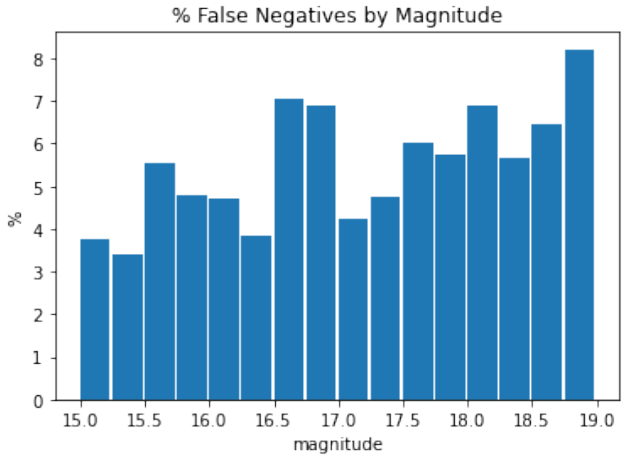



*Figure 21: False negatives by magnitude.*

Investigation into false positive, false negative, and true positive candidates was also done with the hope to gain insight as to how our model may be classifying the images. Samples of candidates from each category are found in Figures 22-24. By inspection of the image cutouts, we hypothesize that our model tends to classify the image cutouts as asteroids if the object in the center of the image has the correct shape (as opposed to the false positives shown in Figure 14), has a single or $4$ pixel point source, is brighter than its immediate neighboring pixels, and its neighboring pixels are roughly the same magnitude relative to the object.

In particular, we observed the fourth condition from comparing the false negative images to the true and false positive image cutouts. In the false negative images, when as asteroid is near a boundary with a sharp gradient in pixel brightness, our model appears to have difficulty correctly classifying the image cutout. Although frequently observed in the false negatives, the pattern is rarely observed in the true and false positive image cutouts.

It is also important to note that some of the false positive images could in fact be asteroids. Although the difference

images were injected with synthetic asteroids, there could also be real asteroids in the images. Since we determined the class of an image by whether it was near–within 1 pixel–of the coordinates of a simulated asteroid and the images are large relative to a 3 pixel grid, it is reasonable to conclude that any real asteroid is unlikely to be labeled as an asteroid when generating our training and testing data. Although we are not experts at identifying asteroids from difference images, it was found to be difficult to discern why many of the false positives are not, in fact, asteroids.

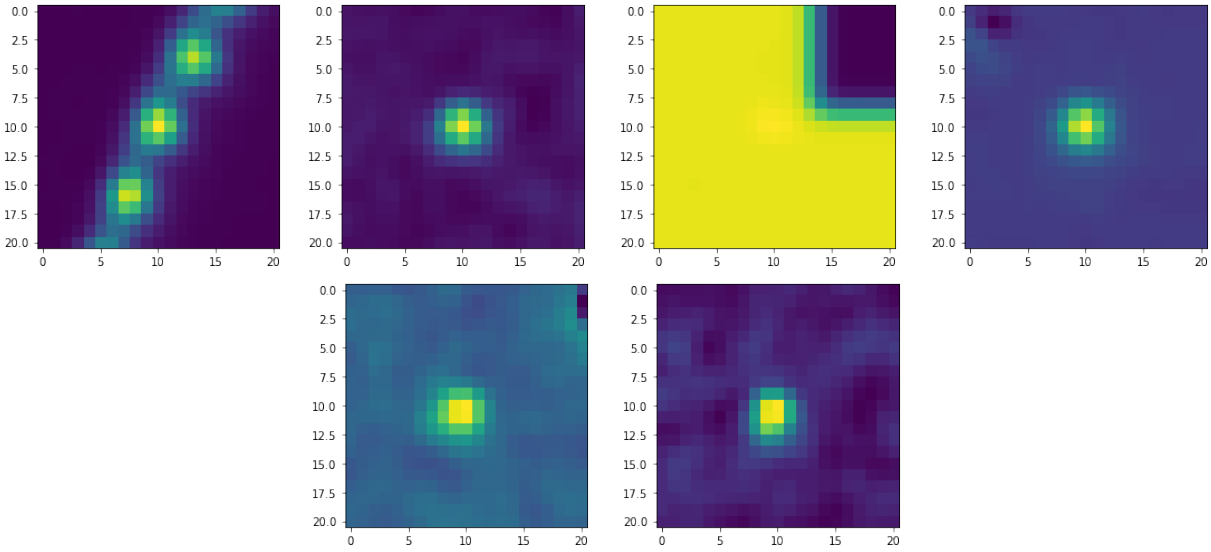
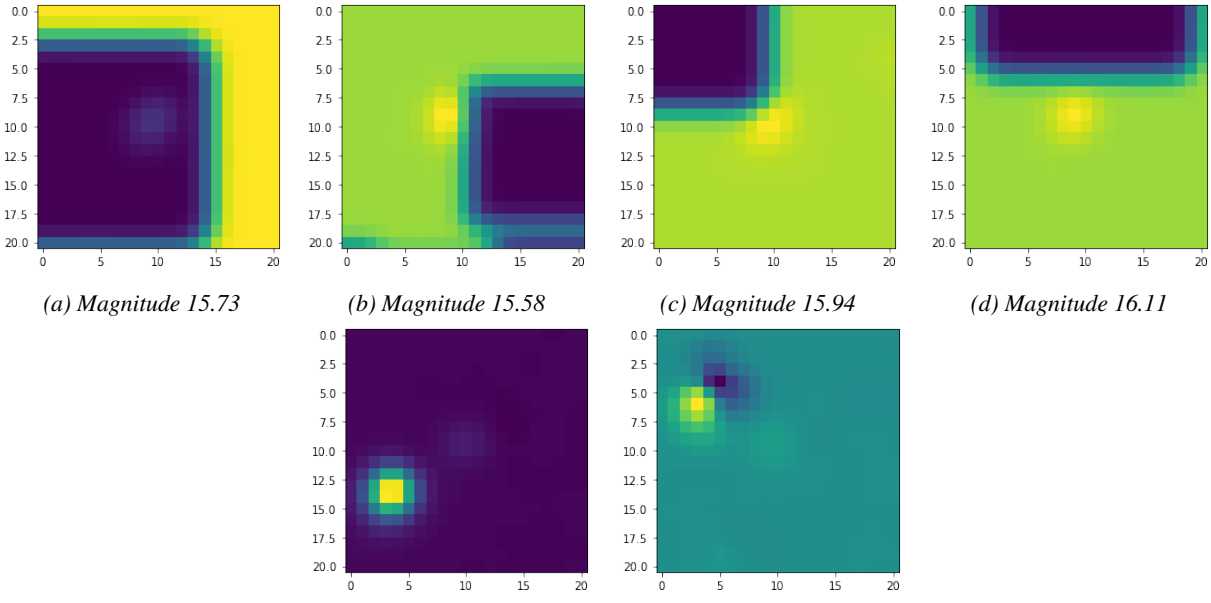


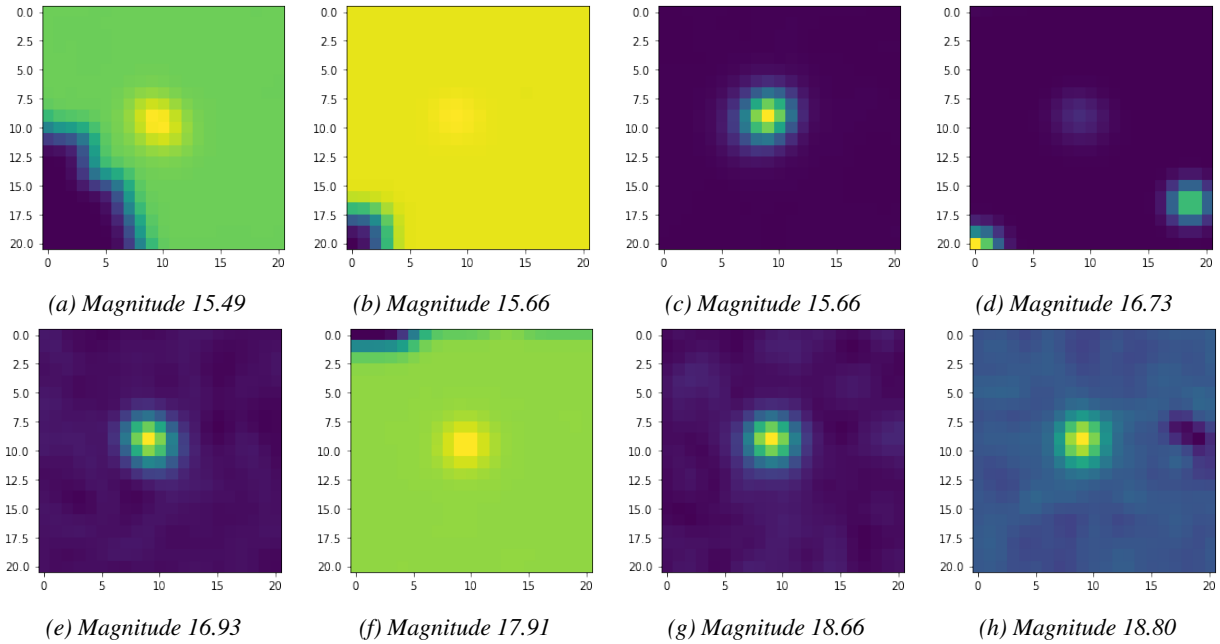
*Figure 22: Sample of false positive examples using Boosted-CNN final model.*



*(a) Magnitude 15.73* *(b) Magnitude 15.58* *(c) Magnitude 15.94* *(d) Magnitude 16.11*

*(e) Magnitude 18.75* *(f) Magnitude 18.95*

*Figure 23: Sample of false negative examples using Boosted-CNN final model.*



*(a) Magnitude 15.49* *(b) Magnitude 15.66* *(c) Magnitude 15.66* *(d) Magnitude 16.73*

*(e) Magnitude 16.93* *(f) Magnitude 17.91* *(g) Magnitude 18.66* *(h) Magnitude 18.80*

*Figure 24: Sample of true positive examples using Boosted-CNN final model.*

## 5.1 Results on Unmodified Data

After training and testing our model on images containing synthetically injected asteroids, we were also able to use our model on unmodified images. Figure 25 shows our model selecting candidates using the footprint algorithm and classifying the candidates using first CNN and boosting CNN. The end result was the identification of one potential asteroid. This potential asteroid is shown as the central image in Figure 26, alongside two other potential asteroids from other images. Although we are uncertain as to whether these objects are actually asteroids, they do resemble the images of synthetic asteroids observed during training and testing our model.

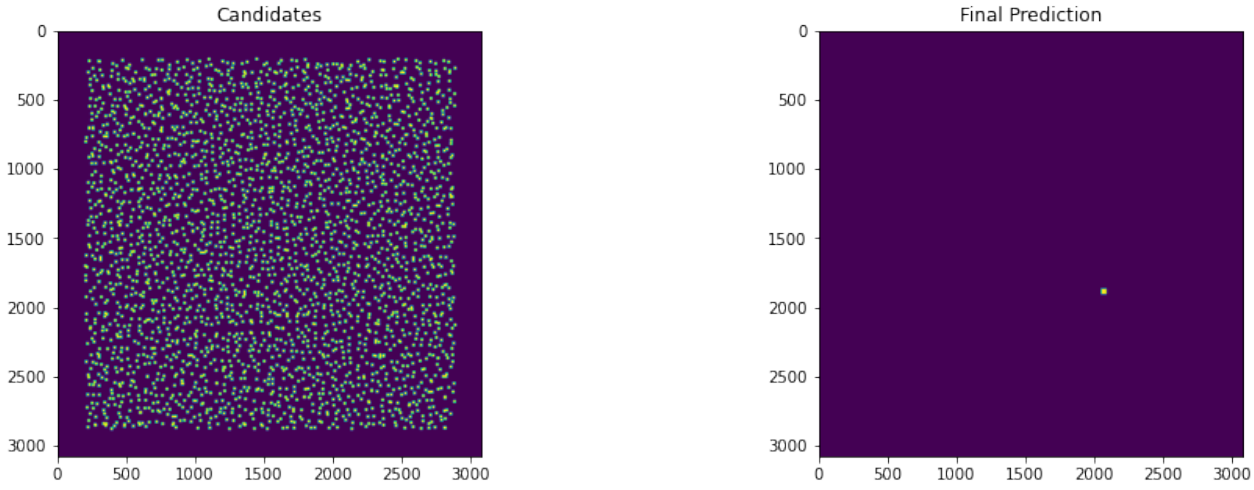



*Figure 25: Selecting asteroids from unmodified image.*

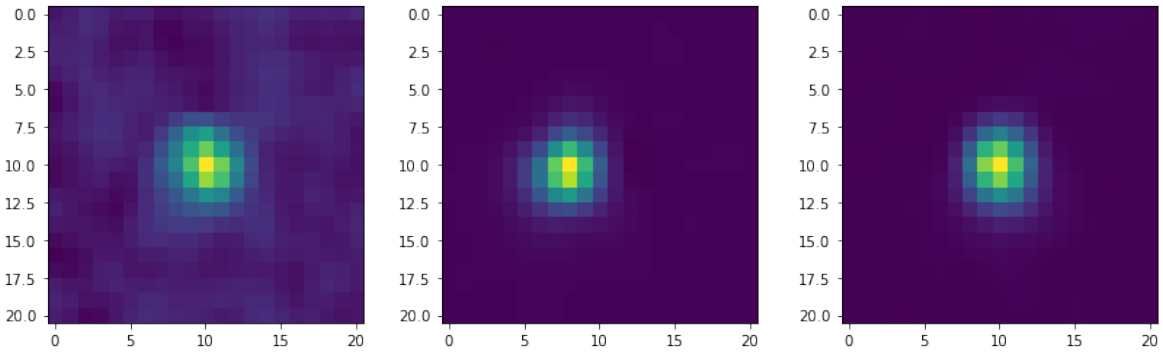


*Figure 26: Sample of asteroids located from unmodified images.*

# 6 Conclusions

In this work, a model for the detection of asteroids in difference images is proposed along with a method of preprocessing difference images to train our model. Our model was evaluated on a dataset that was constructed using a footprint algorithm and a Gaussian filter. The footprint algorithm divides the image into segments and selects a candidate from each segment as a potential asteroid. Having the difference image being broken down into smaller pictures with various artifacts streamlines the process in a efficient manner for us to detect asteroids. This gave us a method of detecting candidates as potential asteroids and reduced the problem into one of object classification.

For classification, our model uses two CNNs. The first CNN sorts through the candidates generated by the footprint algorithm and eliminates as many candidates as possible while simultaneously incorrectly classifying a few true asteroids as possible. This tends to label a large number of non asteroid candidates as asteroids. To account for this, we then classify all candidates the first CNN classified as an asteroid using our boosted CNN. This boosted CNN correctly reclassifies a large portion of these non asteroid candidates while only incorrectly reclassifying a small portion of true asteroid candidates. The overall result of the entire process is a precision of over $0.97$ and a recall of $\sim 92\%$. We believe that this work offers a proof of concept that a model consisting of an algorithm to select candidates from a difference image along with using a combination of 2 CNNs to classify candidates as asteroids and non asteroids is a feasible way to detect asteroids in a difference image.

There is a great deal of future work that could be done on this problem. The area that seems most likely to improve results is a better algorithm to select candidates. Another potential method would be defining candidates as the brightest pixel among some arbitrary number of neighboring pixels. This method is far more computationally intensive; however, it should prove more effective due to the more locally relevant pixels, than a fixed kernel size. It would also be superior in that it is independent of asteroid density. For our method, as the density of asteroids increases you would expect to see a decrease in performance due to multiple asteroids being located in the same segment.

Another area of consideration for future work involves finding an improved method to determine candidates. The footprint algorithm proved to vary in effectiveness and efforts to better detect high magnitude asteroids should be investigated. One potential method is source extraction and utilizing the parameters to minimize the false positive candidates as well as the inclusion of more high magnitude asteroids. We believe that source extraction could be a fair middle ground between performance and speed.

Finally, future work could also include the optimization of the CNNs used in our model. Although we used the same architecture for both neural networks, investigating different architectures presents a plausible path to improving model performance, particularly for the boosting CNN. One could also draw inspiration from or use other well established architectures such as AlexNet or GoogLeNet.

In this study, we showed the potential of our method for the task of detecting asteroids on difference images. We also emphasize that larger and more diverse datasets are needed in order to gauge the methodology in a more realistic

manner. As a future research path, building a dataset that involves true asteroids in images, as opposed to synthetic asteroids injected into images could benefit in validating our methods.