Re-entry Analysis of Serendipitous Radar data (RASR)

Yash Sarda

Computer Science
Computational Astronautics, Science and Technology
The University of Texas at Austin

ABSTRACT

As the frequency of satellite launches into Low Earth Orbit has increased, so has there been an increase in the reentry of these objects into, and sometimes through, the atmosphere. As seen in some recent reporting, the objects that do not completely incinerate during atmospheric entry represent a growing threat not only for falling onto the Earth, but also any aircraft or spacecraft in transit. RASR will create a reliable, automated detection and warning system for anthropogenic objects in space using an existing nationwide network of NOAA doppler radars. Existing work done by the Astro-materials Research and Exploration Science (ARES) department of NASA JSC explored the feasibility of using this radar network to bolster the number of meteorite re-entry detections made, which currently rely on eyewitness reports.

The RASR project expands on the proof-of-concept work of ARES and leverages its small database of detections to extend an automated detection system into the real-time domain. It sifts through real-time atmospheric data and detect phenomena, outputting a confidence value and detection, based on fall velocity, reflectivity, and spectrum width signatures, and the detection neural network architecture. With success in test cases but lacking in data diversity, it is an important first step forward in the understanding of this domain and sets a baseline for future iterations and work.

1. INTRODUCTION

As our shared space environment becomes increasingly cluttered with satellites and other space objects, the risk of a catastrophic collision with unknown consequences increases. Quite often, the lack of data sharing and fragmented sensor models force an incomplete picture, even among government agencies. Our understanding of this space is limited by the measurements we make and requires multiple sources of high-level data fusion. RASR is an attempt to create one such source to better understand entering meteors, and eventually, reentering anthropogenic space objects.

2. RESEARCH BACKGROUND

Public weather radar provided by the National Oceanic and Atmospheric Administration (NOAA) provides a way to automate detection, characterization, and impact site estimation of re-entering bodies. Meteorite falls are a well-documented phenomena and are analyzed here to identify a methodology applicable to tracked and untracked satellites.

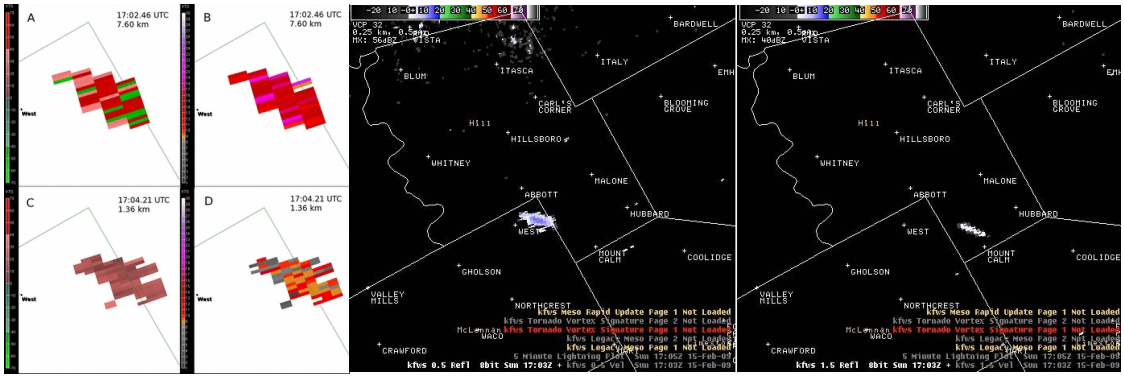

Figure 1: Example manual detections of a fireball in West, TX

Current meteorite reentry detections rely on eyewitness input and may cover only 0.3% of total reentries [1]. This work covers the radar data analysis of the Astro-materials Research and Exploration Science (ARES) department of NASA JSC, automating it for real-time identification without eyewitness dependency.

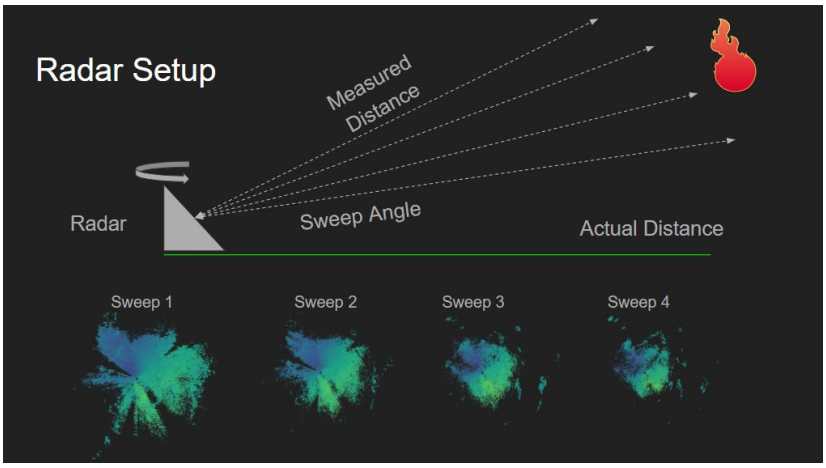

Figure 2: Radar setup diagram with example scans

The NOAA operates a series of 159 Weather Surveillance Radar 1988 (NEXRAD) Doppler radar stations across the continental United States as shown in Figure 3. Every 5-10 minutes, every station uploads a set of local scans across several altitudes to a public archive, each covering a 100x100 km horizontal area up to 8.4 km in altitude, with significant overlap. The "Level II" data products provided by each station include reflectivity, radial velocity, and spectrum width (or standard deviation of local velocity) measurements. While all three products can provide useful information, only radial velocity was considered here, as it measures the uniquely strong atmospheric shear from high-speed reentries which can persist throughout the long-period scans.

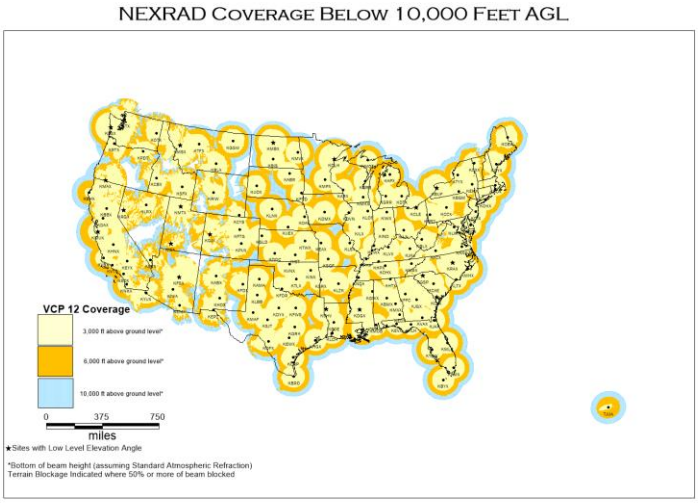

Figure 3: NEXRAD Coverage

3. METHODS

RASR is divided into two main programs, RASR-Get and RASR-Detect. RASR-Get parses the NEXRAD data sites for Level II data from the last 24 hours and downloads it to a local folder. RASR-Detect then unpacks said data and analyzes it for detections.

RASR-Get in essence is a data scraping tool that autonomously downloads the publicly available Level-II Base data from each one of 159 Weather Surveillance Radars (WSR-88D) across the contiguous United States. It utilizes the ”Beautiful Soup” Python library for HTML parsing of the NOAA website. This program is necessary for streamlining data scraping from the NEXRAD inventory to be processed daily through RASR-Detect. Due to complications with Amazon Web Services (AWS) of which the full historical archive of the NEXRAD Level-II data is stored, RASR-Get will pull data directly through an HTTP get request from the NEXRAD Data Archive Website rather than accessing AWS directly. The Level-II base data is saved by the radar sites in increments of four, five, six, or ten minutes which is dependent on the volume coverage pattern of the radar sites. The data is saved as a GZ archive that can be opened with the Python ARM Radar Toolkit (Py-ART) [2]. Other files such as those with an MDM summary file extension or tarball files are not useful for this analysis and not downloaded. The structure of the data URL links is leveraged such that the domain name is constant with additional parameters for the date, radar site, and file name as shown in Figure 4. The links for the data are saved to a text file and then downloaded using Python’s Requests library. Figure 5 shows the general process of how this was implemented in software.

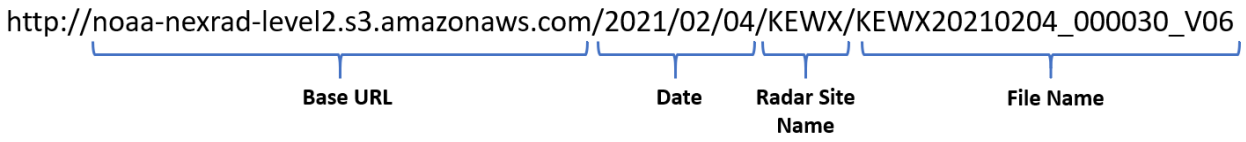

Figure 4: Example of URL link of Level-II Base Data

https://noaa-nexrad-level2.s3.amazonaws.com/{year}/{month}/{day}/{site_id}/{site_id}{year}{month}{day}_{time}

Figure 5: Structure of unique URL links of Level-II Base Data

Some radar sites will undergo maintenance or will be retired over time so data will subsequently not be downloaded for those sites. Despite plans by the NOAA to consolidate current domain names (https://www.ncdc.noaa.gov/) to the National Centers for Environment Information (https://www.ncei.noaa.gov/) domain, it is assumed that the appropriate redirects will be implemented, and the operation and running of RASR-Get should not be affected. If not, the program can be modified to include this domain change. It can also be adapted to download specific dates from manually inputted radar sites which was useful when retrieving data for training the neural network. RASR-Get will nominally be called once per day, and since real-time data from the radars is not publicly available, the program will download a full 24- hours' worth of data from each NEXRAD radar, for processing in RASR-Detect.

Initial attempts to detect these 'fall' signatures were carried out using the Python OpenCV computer vision library. Firstly, radial velocity was filtered based on a minimum speed, and radial acceleration gradients were then calculated as the spatial derivative of these velocities. A basic set of area, eccentricity, and spatial density filters were applied, and contour features were defined. If successful, contours were then calculated for the spectrum width data by applying the latter filter set.

However, this empirical method quickly proved to be too rigid for general use, as it was adapted from certain test cases. Adaptability was required, which prompted the transition to a machine-learning approach. For this application, the PyTorch machine learning library was selected for object detection over Tensorflow. PyTorch allows for a more flexible input array when initially training, which allowed seamless integration with an already developed unpacking algorithm. The Detecto package interface was used on top of PyTorch for ease of use. In order to detect, locate, and classify the fall within the image, a simple convolutional neural network (CNN) would not suffice. Instead, the Faster R-CNN ResNet-50 FPN [3] was chosen, which had been partially trained on the COCO 2017 dataset and was set up for transfer learning. The Faster R-CNN is a combination of a region proposal network and the Fast R-CNN, a deep convolutional neural network with feature extraction and ROI pooling.

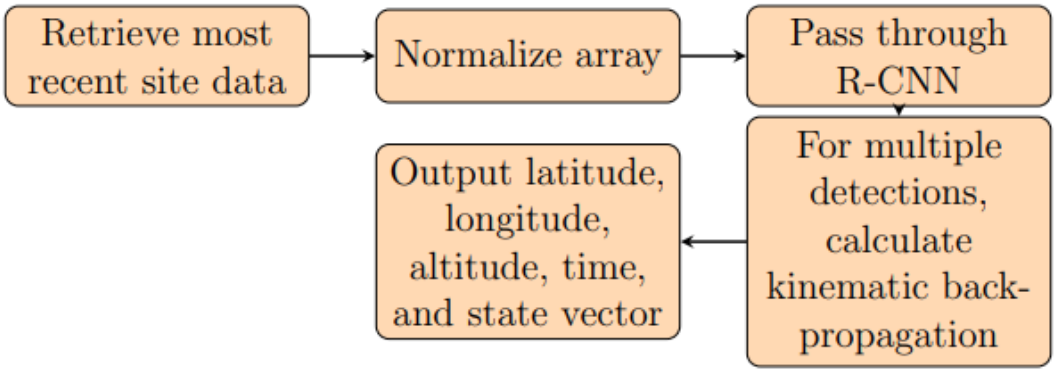

Figure 6: Re-entry identification process

To train the R-CNN properly, the ARES library of radar-captured meteorite falls was manually parsed and labeled. As such, the efficacy and accuracy of the algorithm depends solely on the training data and ensures that as more radar captured meteorite falls are manually detected, RASR will become more and more robust. The automated detection process follows a multi-stage process as shown above. After data unpacking, velocity arrays are normalized and passed through the R-CNN. Following a minimum of two detections, altitude-specific processing was performed iteratively on the velocity data using a 2*iteration pixel dilation filter. The Cartesian positions relative to the radar station could then be converted to latitude, longitude, and altitude for Cesium-based plotting, permitting integration with the ASTRIAGraph satellite visualization interface. Following the individual geodetic positional analysis, the data is also converted to the ECI frame for dynamic backpropagation using the following state equation:

$$\sum \vec{F} = m_o \ddot{\vec{x}} = \frac{-Gm_o m_e}{\vec{x}^3}|x| + C_D l \frac{1}{2}\rho \dot{\vec{x}}^2, \text{ where } \rho = \rho(|x|)$$

$$\ddot{\vec{x}} = \frac{-Gm_e}{\vec{x}^3}|x| + \frac{C_D l}{2m_o}\rho \dot{\vec{x}}^2$$

$$\text{assume } C_D \approx 0.9 \text{ [4]}, \frac{l}{m_0} \approx 10, \text{ thus } k \equiv \frac{C_D l}{2m_o} \approx 5$$

$$\ddot{\vec{x}} = \frac{-Gm_e}{\vec{x}^3}|x| + k\rho \dot{\vec{x}}^2$$

The state equation assumes a Keplerian orbit decay with a basic drag model, where G is the universal gravitational constant, me is the mass of the earth, m_o is the mass of the object, C_D is the coefficient of Drag, ρ is atmospheric density, l is the characteristic length of the object, and

$\vec{x}$, $\dot{\vec{x}}$, and $\ddot{\vec{x}}$ are position, velocity, and acceleration, respectively. As stated above, the ratio of the meteor's characteristic length to mass is assumed to be 10. All measurements are in metric. The timescale of propagation is user-defined, however, as a rough model, the propagation should be kept on the order of 2 minutes to avoid large data discrepancies.

4. DATA

Training data for all fields labelled and built out in the /training/all_fields repository. Raw detection data also tracked down and stored for future use in /training/2500/all_raw_detections. All raw null used stored and sorted into test files (raw_null and test_null). Additionally, time series data for training CRNN is placed in the repository as well in "video format" for video dataset loader. All data was collected through 24 hour downloads from NOAA.

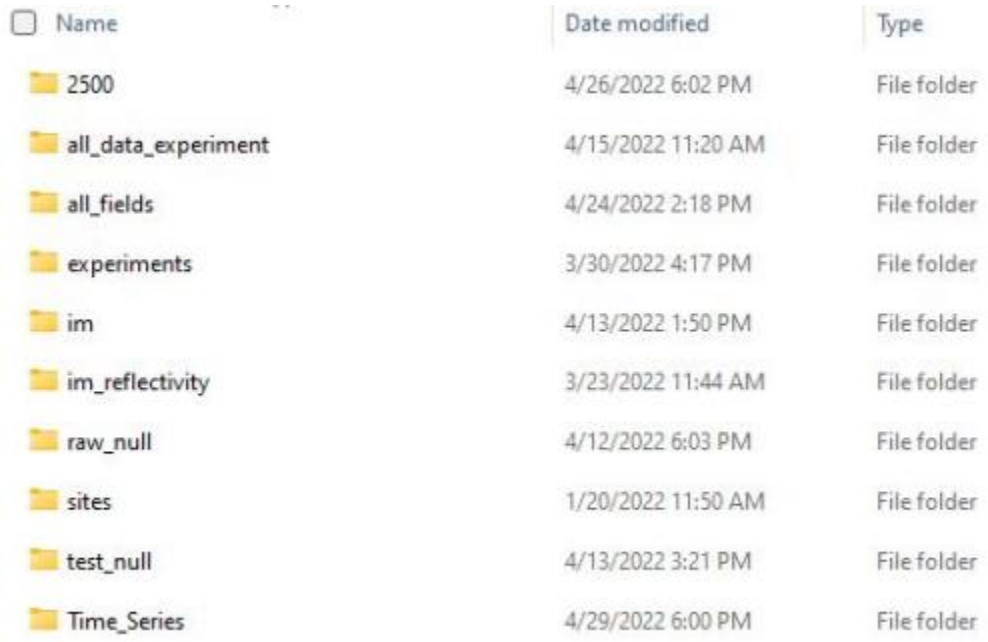

Figure 7: Data architecture

5. RESULTS/CONCLUSION/DISCUSSION

Single iterations of the algorithm took an average of 30 seconds to complete with non-optimized programming on a Lake Michigan, WI test case locally, with a single process over 8 CPUs. The employed OD-CNN successfully detected all 3 case study events with 1 false positive, suggesting that the model requires more training data to better understand the characteristics of a re-entry.

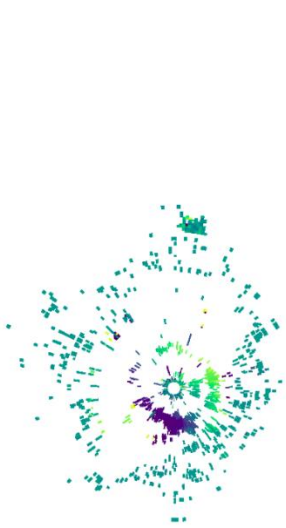

Figure 8: Example detection, visualized with a bounding box and confidence

Local case studies were insufficient to characterize whether the methodology was robust enough to weather and noise effects to permit continental-scale real-time operation, which demanded testing on TACC resources. In processing the 24 hours of continental-scale radar data, no detections were made, which aligns with the average frequency of manual radar meteor detections.

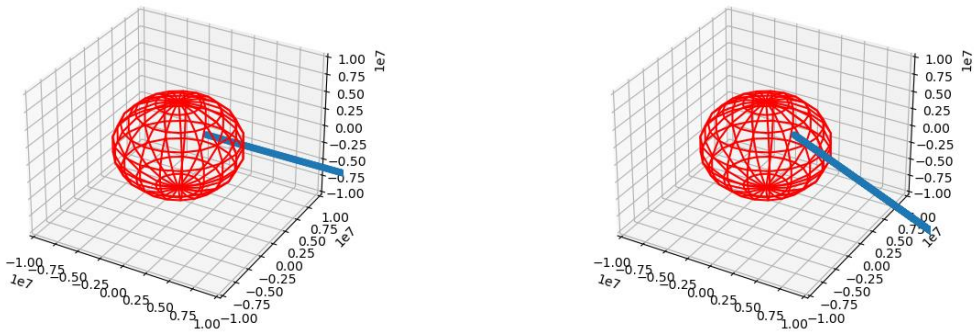

Figure 9: Separate kinematic backpropagations of Lake Michigan, WI falls. Note the misalignment of the paths due to the false positive.

The biggest faults of the OD-CNN model are easily explained by the dataset, as it is approximately 90% null data and 10% actual detections. Expanding on this concept, a confusion matrix (with a confidence interval of 0.70), a receiver operating characteristic (ROC) curve, and a precision recall (PR) curve were plotted for the current model.

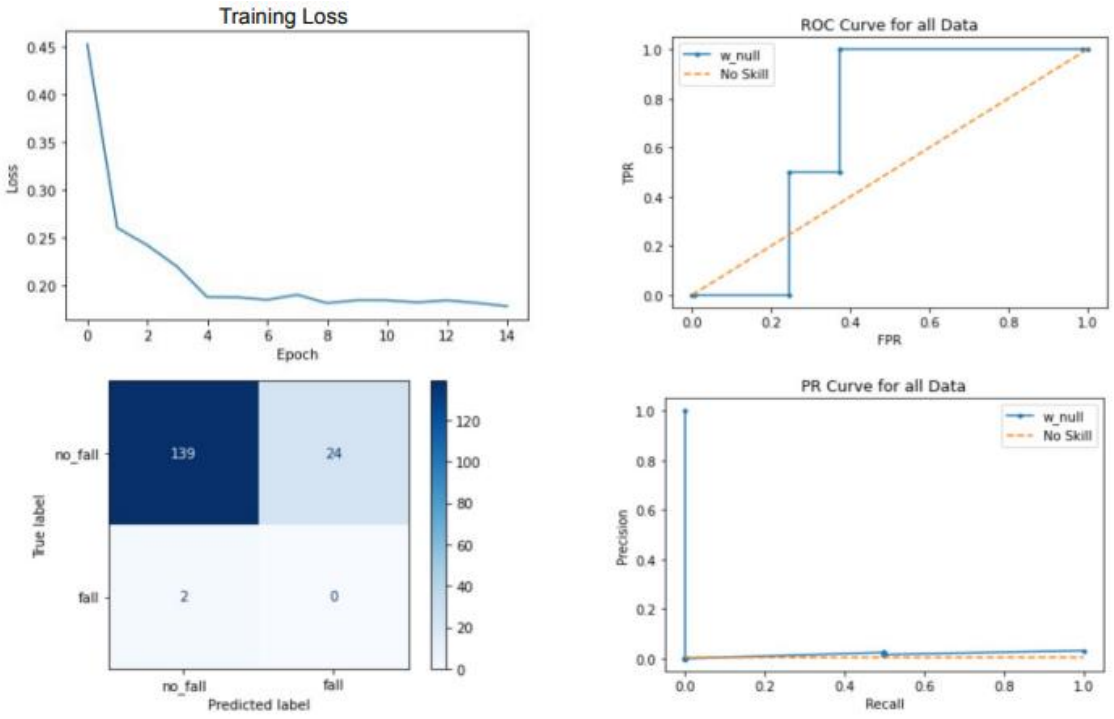

Figure 10: Data exploration of OD-CNN architecture

With a precision of 2% and a recall of 69%, the model shows a high propensity for false positives. Switching to a simple classifier that trains on the null data, the results were far more appealing.

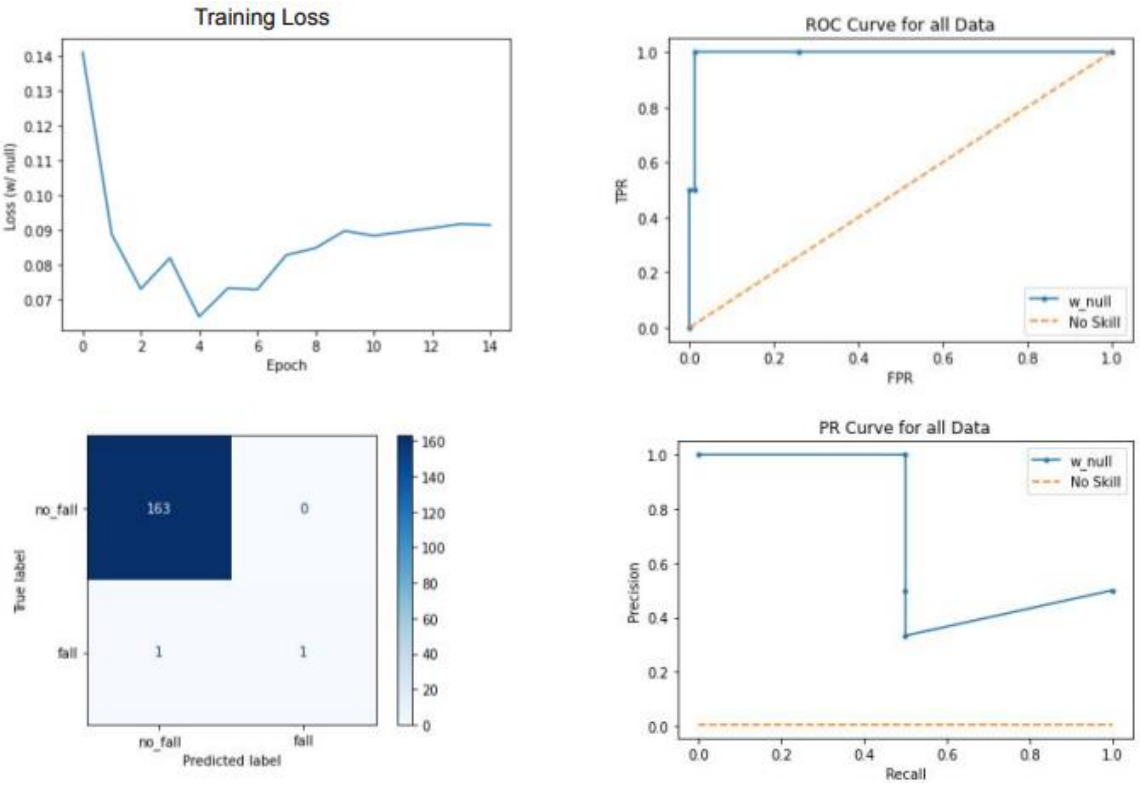

Figure 11: Data exploration of classifier, including null data

With a precision of 71% and a recall of 99%, the "null classifier" showed far more promise in the domain of proper classification, but obviously had no concept of object detection or location, an important step in gaining useful metrics out of the weather data. Thus, it became apparent that a two-pronged approach was necessary. A single classifier incorporating null data, and a follow-up object detector for any image classified as "falls". This type of bootstrapping could prove to only propagate error forward, so work was done to include other forms of data beyond velocity (like reflectivity and spectrum width) into the initial classifier and following object detector.

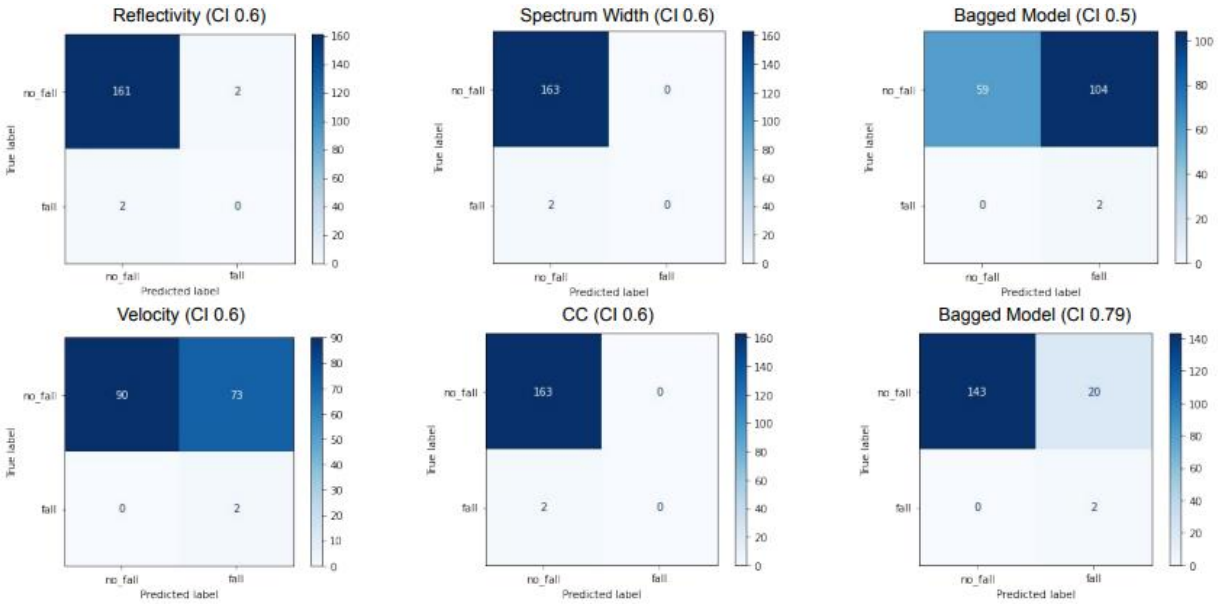

Figure 12: Confusion matrices of various data input types

Models were trained on Reflectivity, Velocity, Spectrum Width, and correlation coefficient (CC) data as well as a bagged model that is a linear combination of all data. Their PR curves showed significant improvement as well:

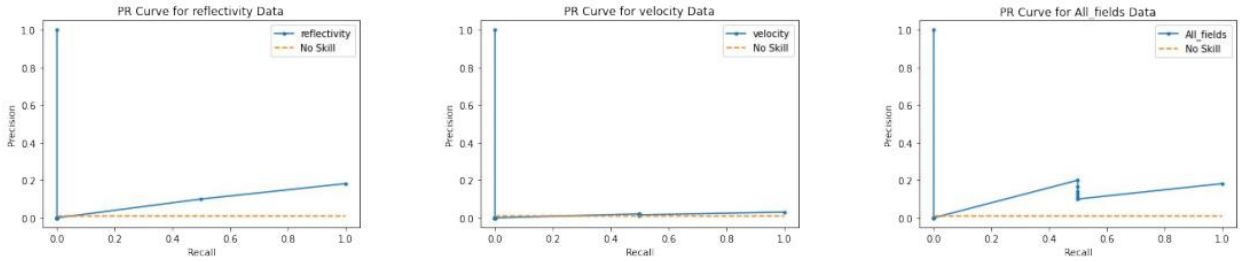

Figure 13: PR curves for reflectivity, velocity, and bagged models

Most of the bias/error can be removed by the reflectivity data (at least for current labelled train/test set), as shown in the confusion matrices in Figure 11. The precision for the current RASR model based on velocity (2%) was improved on by the bagged model (12%) and proved to be a significant step towards an unbiased object detector. Thus, a full development and next iteration of the model would look like Figure 14. Along with the existing experimental model that uses 1 weak learner and RNN to classify detections, this architecture expands on that, integrating other data fields through other weak learners and utilizes ensemble models for

classification head and detection. The LSTM could also be constrained by physics for increased precision.

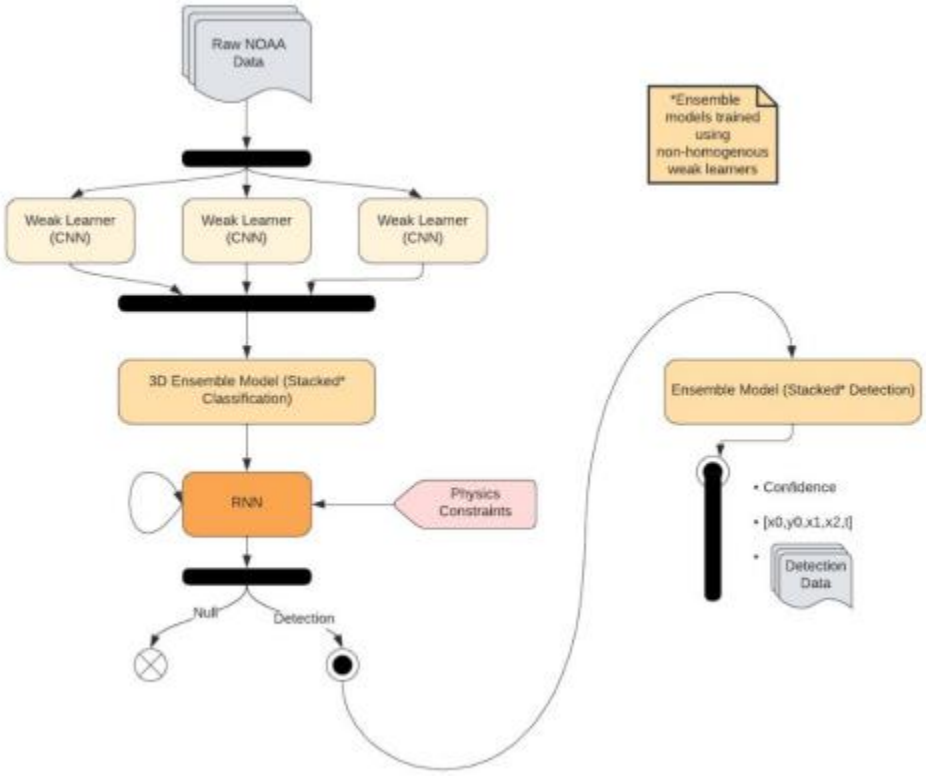

Figure 14: Proposed overall detection architecture

6. FUTURE WORK

As a topic for future research, it may be possible to measure persistent turbulence and thereby characterize velocity. Similarly, it may be possible to exploit the shape of the turbulence regions in order to infer mass and establish initial boundary conditions for more accurate trajectory analysis. Another major path of work is the development of an algorithm to establish kinematic paths from solely ECI positional data, as seen in Figure 13. A hypothesis-based filter is currently proposed, but untested.

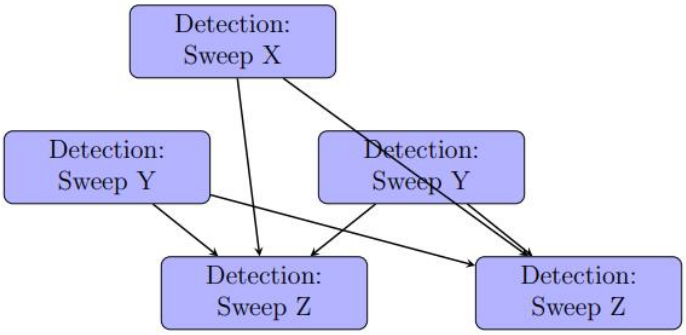

Figure 15: With multiple detections and sweeps, understanding the state and strewn field is difficult

It is also possible, based on the magnitude of the velocity data, to distinguish between meteors and true re-entering orbital debris. Most meteors entering the Earth's atmosphere travel at approximately 10 km/s, while re-entering objects from orbit travel on the order of 1 km/s. This difference in the velocity signature has yet to be examined through the R-CNN or numerically in post-processing.

7. REFERENCES

[1] FRIES, M. and FRIES, J. (2010), Doppler weather radar as a meteorite recovery tool. Meteoritics \& Planetary Science, 45: 1476-1487. https://doi.org/10.1111/j.1945-5100.2010.01115.x

[2] Helmus, J.J. \& Collis, S.M., (2016). The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. Journal of Open Research Software. 4(1), p.e25. DOI: http://doi.org/10.5334/jors.119

[3] Ren, S., He, K., Girshick, R. B. \& Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[4] Carter, R., Jandir, P., \& Kress, M. (2009). ESTIMATING THE DRAG COEFFICIENTS OF METEORITES FOR ALL MACH NUMBER REGIMES.