Kirin Patel
AE 457
October 15th 2019

Analyze Multiple Solutions, Pros & Cons

Currently, I have a proposed solution and fallback solution. The proposed solution is to train a machine learning model to detect meteor features within spectrographs, then, after detecting them, provide region information about the detected meteors. The aim of this model is to reduce the time required by people at BRAMS, Zooniverse, and other facilities when processing data. Using a machine learning model would allow for continual training on new data, thereby increasing the accuracy of the model, it would also allow for real-time data processing to occur after collection, further reducing time spent processing data. However, even with these monetary and time incentives, a machine learning model requires thousands upon thousands of data points for accurate and effective training. These data points all must have the meteors mapped out within the spectrograph. Even if data points are obtained with this information provided it still is likely that the size of the data set is not large enough. Alongside the requirement for a large to massive data set, time and monetary constraints revolving around model training are also a concern. Training a model can take hours to days depending on the hardware being used to train the model. Should this hardware be cloud based, it can result in thousands of dollars of bills. Tradeoffs between speed and money must be accounted for with this approach. A final concern with this approach revolves around my lack of knowledge with Tensorflow/Keras and machine learning in general. I would have to spend much of my time learning the tools surrounding implementation and how they should be used rather than implementing a model itself.

Alternatively, the fallback solution is to use OpenCV to create a feature detection algorithm can circumvent the major issues presented by creating a machine learning model. Using OpenCV will not require the same sized data set, although more data is always ideal, nor will it require the same training time as there will be no requirement for model training. This will allow for a rapid development cycle for the OpenCV algorithm approach which are not possible when creating machine learning models. The OpenCV algorithm has more flexibility than a machine learning model, if there are restrictions with OpenCV or other known or currently unknown elements, then other languages and libraries can be used to circumvent those restrictions. For instance, OpenCV has C++ and Python libraries, but it is more beneficial for efficiency based applications to use C++ as it can be better optimized due to the fact that C++ is a low level language that can be compiled down to system code, rather than a high level language that runs in an interpreter like Python. Unfortunately, just like the machine learning approach, the OpenCV algorithm approach has flaws. Within this use case, the first major concern revolves around the development of the algorithm. I have limited knowledge of C++ and OpenCV. Again, This would likely result in much of the development time being spent learning a language and its interaction with OpenCV rather than actually developing the algorithm itself. Alongside this concern, there is a smaller concern, the coolness of an algorithm based approach. There are two factors reducing the coolness of this approach. The first is that machine learning is a popular buzzword. The second is that there is an existing approach for this specific problem with a feature detection algorithm (Calders, 2019). While this does not reduce the impact the solution can have, it does decrease the incentive to take this approach.

Work Cited

Calders, S. (2019, January 29). calders/RadioMeteorZoo. Retrieved from https://github.com/calders/RadioMeteorZoo.