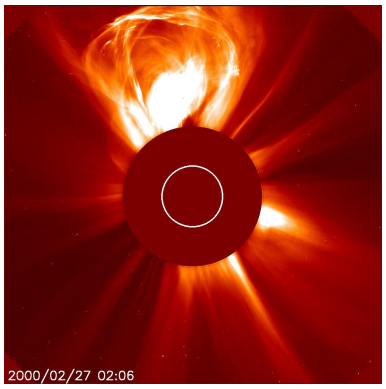
Final report-Development of Optical Flow technique to track coronal transients in inner corona

N.Arutkeerthi-P49, Harini Suresh Maruthi-P25

September 2023

1 Introduction

The study of space weather fuels both curiosity and applications like predicting geomagnetic storms and protecting satellites.
Coronal Mass Ejections (CMEs) are a well known phenomenon of the sun that have an appreciable magnitude of effect on us.[1] The magnetic fields generated on the surface of the sun due to the movement of plasma in the outer layers form loops in seemingly random directions. The tangling of these loops in the corona gradually builds up potential, which, beyond a certain threshold, result in phenomena like jets, solar flares and mass ejections. [2][3]
CMEs cause geomagnetic storms on collision with the Earth. The geomagnetic storms can cause damage to satellites and result in effects like electrical power outages.
Thus, predicting their velocities and trajectory would benefit in forecasting space weather and taking necessary actions.
In this project, a computer vision technique called 'optical flow' is being applied to track coronal transients as close to the sun as possible. The initial phase of our project involves work on LASCO C2 data, which enables us to track transients from as close as 2 solar radii. With the Aditya L1 mission launched and data released soon, detection of CME close to 1.05 solar radii seems possible.



2 Optical Flow - The algorithm

Optical flow is a computer vision algorithm to find the velocity of pixels, the units that form the objects, moving in subsequent frames of a video.[4][5]
It works on an assumption that the brightness intensity in a small given patch is constant.
Therefore, the intensity between consequent frames is taken to be constant

$$I(x, y, t) = I(x + \partial x, y + \partial y, t + \partial t) \tag{1}$$

The Taylor series approximation is used for the right hand side (RHS) of the equation and the common terms are eliminated.

$$I(x, y, t) = I(x, y, t) + (\partial I/\partial x)\delta x + (\partial I/\partial y)\delta y + (\partial I/\partial t)\delta t \tag{2}$$

$$(\partial I/\partial x)\delta x + (\partial I/\partial y)\delta y + (\partial I/\partial t)\delta t = 0 \tag{3}$$

Dividing by δt on both sides,

$$(\partial I/\partial x)u + (\partial I/\partial y)v + (\partial I/\partial t) = 0 \tag{4}$$

$$I_x.u + I_y.v + I_t = 0 - - - > EQUATION! \tag{5}$$

This requires simple matrix algebra henceforth to get the velocity in the x and y directions (u and v respectively in this case) from the I_x, I_y and I_t matrices.

3 Corner detection algorithm: Shi-Tomasi

Corners are given importance and have been focused upon because in the case of an edge or other dimensions' analysis, the velocity we would calculate would not be the actual velocity, rather, would be a component. On scanning an image with a window as that with a kernel, if there is an area with a major change or gradient irrespective of direction of scan, there is a high probability that the area has a corner. [6]

For a window W located at (X, Y) with pixel intensity I(X, Y), formula for Shi-Tomasi Corner Detection is –

$$f(X, Y) = \Sigma[I(X_k, Y_k) - I(X_k + \Delta x, Y_k + \Delta y)]^2 \tag{6}$$

Calculation of f(X,Y) would be time consuming. Hence, Taylor Expansion is used to simplify the scoring function R

$$R = min(\lambda 1, \lambda 2)$$

where $\lambda 1, \lambda 2$ are eigenvalues of the resultant matrix M, where M is:

$$M = w(x, y) \begin{pmatrix} \Sigma I_x^2 & \Sigma I_x.I_y \\ \Sigma I_x.I_y & \Sigma I_y^2 \end{pmatrix} \tag{7}$$

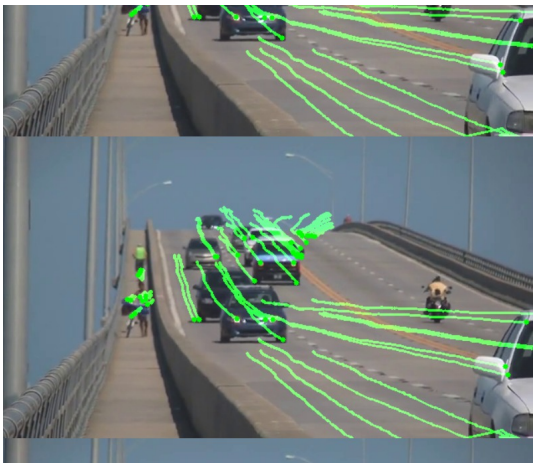
where w(x,y) is an arbitrary window equation containing information about the size of the kernel analysed The value of R can be between 0 and 1. The quality parameter can be defined, above which the algorithm would consider a given moving patch as a corner.
Typically, since this algorithm is used in simple videos like car traffic, birds movements and analysing water streams, it is suggested to keep the quality parameter between 0.4-0.6.
The project involves coronagraphs with less sharper features. Hence, lower parameters, i.e., between 0.1-0.2 could be used to track the CMEs.

4 Application of algorithm

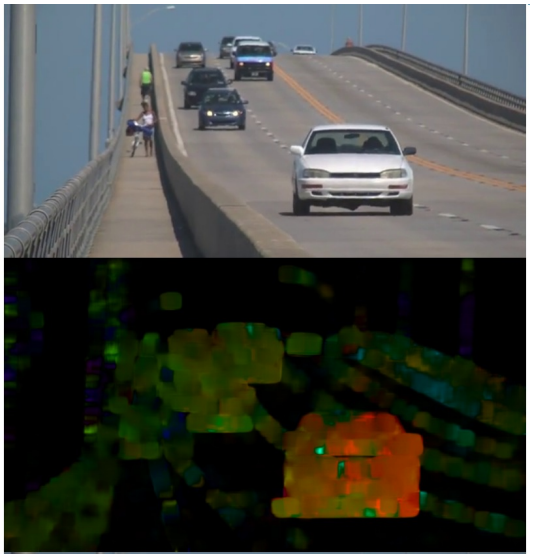
4.1 On simple videos like traffic

Optical flow was first applied on simple slow traffic videos. Two types of optical flow techniques were considered: "Sparse" optical flow and "Dense" optical flow.
Sparse optical flow gives the flow vectors of "interesting features" i.e., corners, whereas dense optical flow gives the flow vectors of the entire frame i,e., all pixels, up to one flow vector per pixel.

In the case of large streams or enhanced videos, dense optical flow seems to be a better alternative. Since our initial work would be with LASCO C2 coronagraph, which doesn't have the CME flow on the entire coronagraph, sparse flow would be the better algorithm to apply.
Below are the results of the initial attempts of tracking objects and their flow, without magnitudes, using both sparse and dense flow.



(a) Sparse flow tracking



(b) Dense flow tracking

The above was the initial attempt by us in understanding the working of the "opencv" library for optical flow.
From the above, only the "flow" was obtained, and the magnitudes of the velocity of corners at each frame remained unknown.
Upon further search, an extensive library- "OpyFlow" was found. [7] OpyFlow is a python package applying optical flow algorithm with Shi-Tomasi corner detection. Applying it resulted in visual and access to both the flow vectors and their magnitudes stored in .csv format respectively, thus making data analysis much more efficient.
From OpyFlow, the obtained velocities of the cars were about 10-20 km/hr which seems to be an appropriate result for slow traffic.

(a) Using opyflow library depicting magnitudes

	A	B	C	D	E	F
1	Time_A	Time_B	X	Y	Vx	Vy
2	0.5	1.5	308.1624908	207.8433456	0.3249817	-0.31330872
3	0.5	1.5	177.926918	189.5297241	-0.14616394	-0.94055176
4	0.5	1.5	301.902832	108.7465553	-0.19433594	-0.50688934
5	0.5	1.5	304.8906555	98.74398422	-0.21868896	-0.51203156
6	0.5	1.5	304.8752899	103.7544365	-0.24942017	-0.491127
7	0.5	1.5	176.8641968	176.5275574	-0.27160645	-0.94488525
8	0.5	1.5	241.9809341	155.8336868	-0.038131714	-0.33262634
9	0.5	1.5	232.9363403	154.821785	-0.12731934	-0.35643005
10	0.5	1.5	232.9302063	148.8126907	-0.1395874	-0.37461853
11	0.5	1.5	266.9558105	155.8518143	-0.088378906	-0.29637146
12	0.5	1.5	240.9829407	145.8017807	-0.034118652	-0.3964386
13	0.5	1.5	267.0638275	103.6517639	0.12765503	-0.69647217
14	0.5	1.5	268.9617157	145.7832489	-0.0765686	-0.4335022
15	0.5	1.5	167.8935928	188.6317368	-0.21281433	-0.7365265
16	0.5	1.5	380.355896	206.9067688	0.711792	-0.1864624
17	0.5	1.5	323.5796051	100.659214	-0.8407898	-0.68157196
18	0.5	1.5	313.1057739	199.7590179	0.21154785	-0.4819641
19	0.5	1.5	315.6402283	99.72716904	-0.71954346	-0.5456619
20	0.5	1.5	306.0373077	246.8880615	0.07461548	-0.22387695
21	0.5	1.5	172.8015137	167.5180969	-0.39697266	-0.96380615
22	0.5	1.5	305.0941162	254.8943863	0.18823242	-0.21122742
23	0.5	1.5	258.9196472	118.8031807	-0.16070557	-0.3936386
24	0.5	1.5	306.0527802	239.9067764	0.1055603	-0.18644714
25	0.5	1.5	165.8429871	206.6036453	-0.31402588	-0.79270935
26	0.5	1.5	393.8647614	96.7110939	-0.2704773	-0.5778122
27	0.5	1.5	307.9249725	91.6986351	-0.15005493	-0.6027298
28	0.5	1.5	265.9722137	112.6221619	-0.05557251	-0.75567627
29	0.5	1.5	158.8188095	193.6749573	-0.36238098	-0.65008545
30	0.5	1.5	375.3479156	199.8695035	0.6958313	-0.22099304

(b) The magnitudes are stored in .csv format which is pre-processed before analysis

4.2 On simulated CMEs

The optical flow algorithm was then applied on simulated CMEs.[8]. The simulation was a coronagraph with 1 pixel equal to 2.5 arcsecond, which with respect to Earth, 1 arcsecond being equal to 700 kilometre. Multiple inconsistencies and issues were faced since the simulations were significantly noisy. Making appropriate changes to the quality level parameter resulted in better outcomes. A parameter of 0.1-0.3 was the most appropriate and resulted in better velocity vectors.

Value obtained from one of the trials:

Minimum velocity:146.79 km/s

Mean:298.83 km/s

Maximum:439.04 km/s

which is appropriate enough, given the simulated CMEs had velocities between 150-400 km/s, with the mean being around 250 km/s.

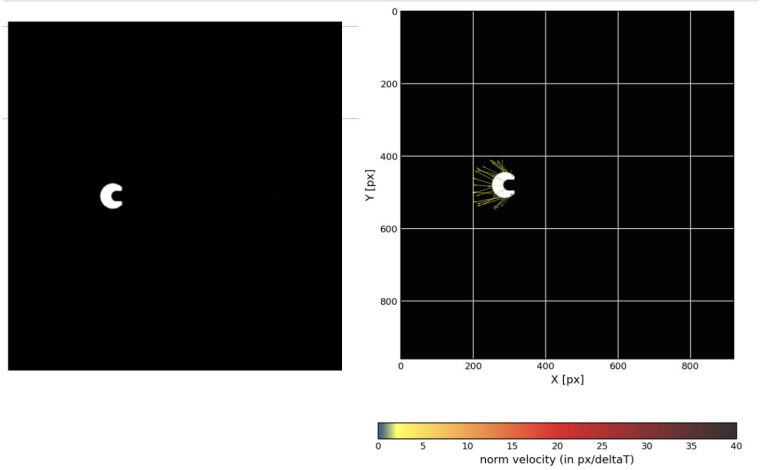


Figure 3: Simulated CME and optical flow on it

4.3 On Observational Data

When applied on actual SOHO LASCO C2 data a lot of challenges were faced.

In our initial attempts, the CMEs went almost completely undetected, the reason being the corners of the CME were much less sharper than those of other features in the observation, which resulted in other features being detected over the CMEs, and also the rapidly changing brightness of the CMEs. Since the algorithm works on the assumption that the brightness of a local patch remains constant in a given short period of time, this fluctuating brightness was not ideal.

The next section would be a discussion on our ideas and approach to resolve the above.

5 Challenges faces and proposed solutions

5.1 Back propagating vectors

While working with the simulated data, some frames resulted in strange observations. Back propagation of vectors was observed. While this might be a possibility in actual observational data, the CMEs in the simulations were purely forward in motion. Therefore, such vectors pointing backwards were concluded as an error by the algorithm.

Probing further into the algorithm, it was found that the reason was a large window kernel.This gave rise to relativistic (not Lorentzian, but Newtonian) velocities where two corners are considered as a single same

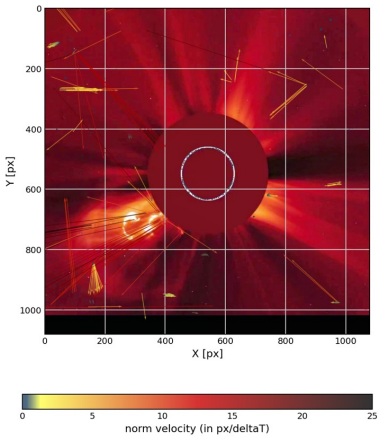
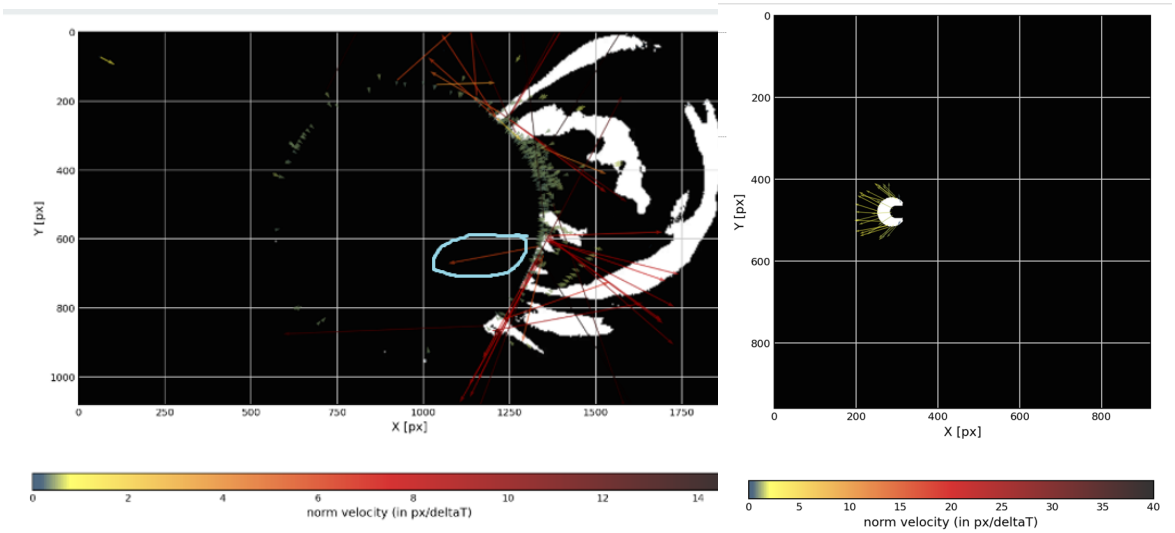


Figure 4: Optical flow on original data with brightness fluctuation

corner by the algorithm, and thus although the corner moves forward, some other corner that enters the window is considered as the same one, and relatively, the merged corner seems to have moved backwards.

This problem was solved by a simple parameter change - reduction in the window kernel for analysis. Despite this, there were still rare instances of this error, but the frequency reduces significantly. The window size had been reduced from 20x20 to 9x9. Further reduction is possible, however, that would result in an upper limit to the magnitude of the velocity vectors that would be obtained, unless better enhanced coronagraphs with real time data with the least possible cadence are worked on. Hence, an optimal window size is preferred.



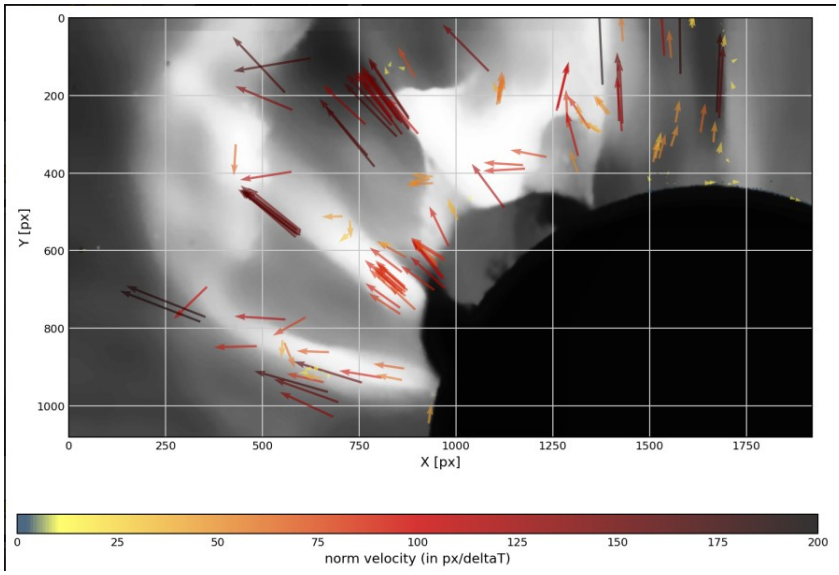
(a) Backward vectors on simulated data (one such vector highlighted)

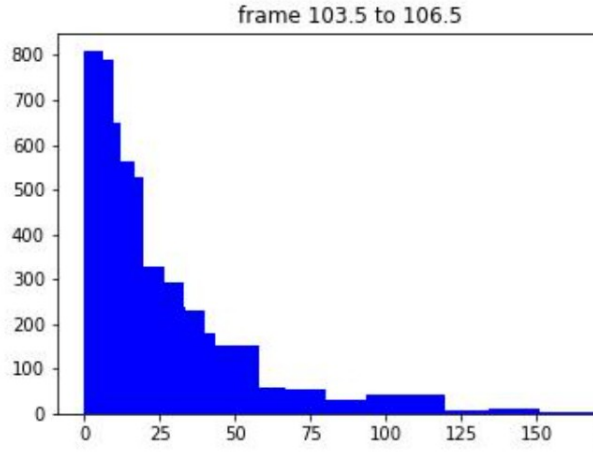
(b) Most backward-vector errors were eliminated when window kernel size for analysis made smaller. The outliers could be eliminated during pre-processing

5.2 Highly fluctuating brightness in real data

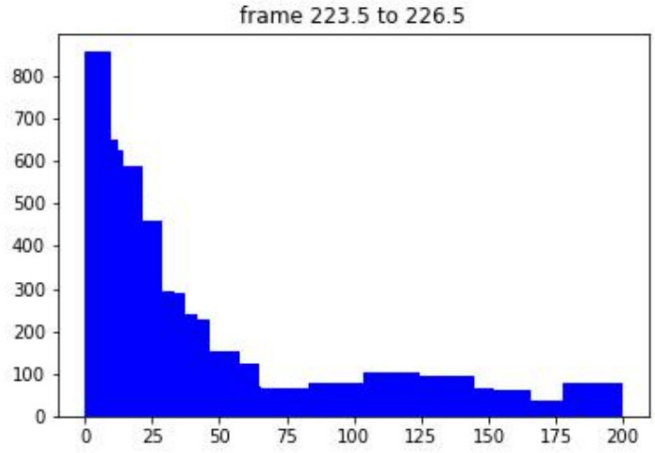
It is known that the algorithm works on an assumption that the intensity or brightness remains constant over a small patch for a small amount of time. However, the cadence of coronagraph is high and the brightness fluctuates very rapidly due to the nature of a CME. Thus, the algorithm failed to even recognize a few good CMEs during the 2014-2015 solar maximum.

Hence we have to preprocess the data a bit. Base differencing was applied to the data. The data had to be filtered to enhance the detection of the CMEs over all the remaining noise and other features. The filtering had to be done such that the noise and other features get masked without compromising on the sharpness of the CME much. The Median filter did remove the noise, but the CME was softened out beyond detection. An appropriate choice of filter was the Bilateral Filter. Bilateral filter does the same denoising as the median filter but with some Gaussian weights defined. This effectively removes the salt and pepper noise to some extent while also not smoothening out the CME features much. The CME remains detectable when the quality level is kept low while applying the optical flow algorithm. A bilateral filter on a kernel 100*100 with weights 0.75 worked well for our detection. More about the filter could be looked upon here: [9]





(a) Velocity distribution pre-CME - solar winds and other features are present along with the CMEs



(b) Velocity distribution post-CME - there is a visible increase in incidence of higher velocities after the CME erupts

Figure 6: The Velocity Distribution histogram - Velocities are in pixels/frame

The analysis of the filtered CME video is available here -https://tinyurl.com/5abpckcm
The histogram showing distribution of velocities with time (pre and post CME) is available here-https://tinyurl.com/2uthhhb7

5.3 Vortices and possibility of rotational motion

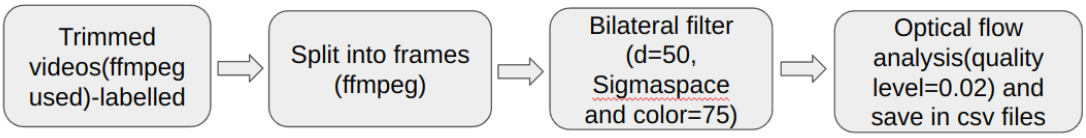
Running the algorithm on the video after pre-processing resulted in some frames showing relatively low magnitude velocities having a vortex-like motion with some curl on the velocity vector field near the region of eruption of the CME. Most of the detected velocities fell in this category. Manipulating the quality level reduced the incidence of such vortices but they never could be completely omitted. These velocities were not observed with as much prominence in the simulated CMEs. It is to be noted that at the quality level considered, these could have been false velocities that were reported. These were hence not considered to be of much importance for further analysis.
On further testing for these velocities with multiple videos, if they are prominent in most of them, one of the possible directions this project can be taken in is to develop a dynamical model of the CME near the region of eruption, which could probably justify the incidence of such velocities according to possible regions of stability in the phase space as might be described by the dynamical model.

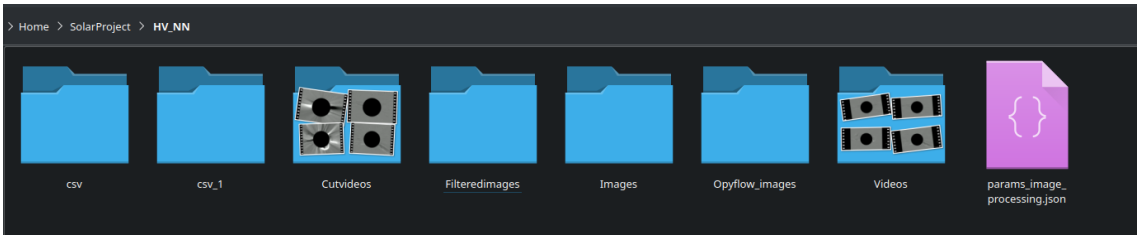
6 CME warning system using neural networks

Now that the algorithm is slightly perfected to detect CMEs, we try to build a model that could warn about CME events using the velocimetric inputs it gets from optical flow analysis.

6.1 Preprocessing and Optical flow analysis

We take 50 videos - each representing coronagraph LASCO C2 of 24 hours. 25 of them are those with partial halo events and other 25 with very poor events. We give binary labels - 1 representing a partial halo or good event has occurred while 0 represents null or poor events. The videos, their frames, filtered frames and the optical flow analysis on each video have been linked here- https://rb.gy/7e50h.
The pipeline includes first downloading events from helioviewer, trimming the video to avoid some excess parts from the internet on video, split into frames and bilateral filter them (kernel size=50 now since its full size coronagraph, sigma Space and color remains 75).
And finally optical flow is applied with quality level 0.02 and other parameters which you could find in our script (like maximum deviation within radius and Goodflags to be checked at each region, etc). The .csv files are saved on which the model will be created.





6.2 Architecture of Neural network and Results[10]

We consider only time and the magnitudes of velocities we get for every frame (or time) and correlate with the label the particular sequence(or .csv file) has. In a nutshell by seeing how velocity distributions change with time we want to predict the occurrence of a CME event.

We use recurrent neural networks with LSTM(long short term memory) to learn the patterns in a given time sequence and give binary classification of that event. The architecture in brief is as below:

-The neural network first takes in variables TIME and its corresponding velocities V , stacks it in a Sequential model(using keras)- a linear stack of layers with one layer added at a time.
-Passed through two LSTM layers of 64 units each and sequence length 100(it goes back and forth in time steps of 100 to capture the event no matter when it happens)
-Flatten this 3D output into 2D tensor, connect it to two full connected Dense layers with 32 units each with RELU activation
-A final output layer with sigmoid activation function(1 unit) to output 0 or 1.

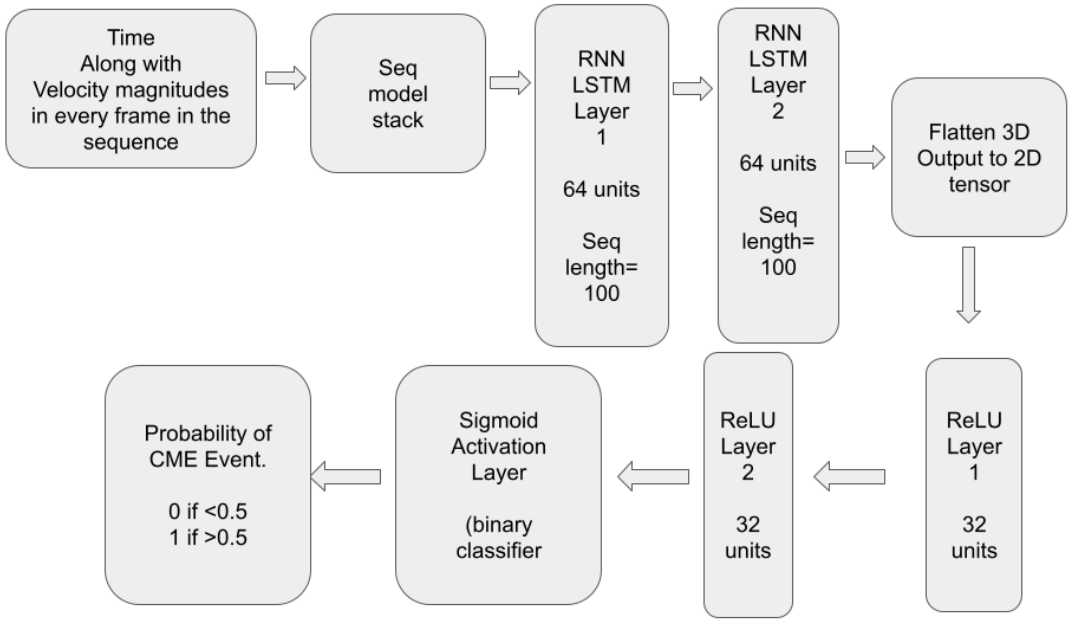


Figure 7: RNN with LSTM network architecture

This was the best model possible with 100 epochs (we also had to break down the .csv file sequences into batches since every file had around 400,000 points and tensorflow couldn't handle that many datapoints at one go without enough memory allocated)

```
Best Model Statistics:
Training Loss: 0.1982, Training Accuracy: 97.50%
Test Loss: 1.3009, Test Accuracy: 80.00%
```

The variance as you could see is pretty huge but this model is good considering we ran the model only on 50 samples and that too for just 100 epochs(though even this took a lot of memory space and all our computers resources). We would plan to tune this model more by augmenting data, regularization and other hyperparameter tuning. Another alternative would be to run on more data as well. One could download this model from our Github repository, load it and run it after preprocessing the .csv file they get from our optical flow analysis on a filtered CME video.

7 Conclusion and Future possibilities

The algorithm has been understood and the challenges it poses have been identified. A model that could act as a CME warning system is developed using RNNs with LTSM. The model needs to be further tuned by reducing the variance, better data. Another thing to note is we have taken only the magnitudes of the velocity into consideration(although data the Vx and Vy with directions are available too). With better computation resources we could make a more complex network that would be able to precisely predict the location of CME eruptions as well taking X,Y,Vx and Vy as inputs.

It is also important to note we have used helioviewer LASCO C2 data. The original fits files have a lot of processing to do(This is parallely being done, it is not yet complete due to a lot of issues encountered with the data). Aditya-L1 gives us a promising future to better coronagraphs, cleaner data and given they already have automatic CME detection in Aditya-L1 a velocimetric model, using optical flow would enable us to completely model CMEs into the future.[11]
Future possibilities include integration with other models, fine tuning hyperparameters and work to make the algorithm faster. Usual CNNs are slow compared to immediate velocimetric analysis, so an integration of optical flow with exisiting models would be able to predict the eruption of a CME much before.

8 Acknowledgement

We extend our sincere gratitude to our project mentor Prof.Dipankar Banerjee and our co-mentor Dr.Vaibhav Pant, who provided their insights and expertise and assisted us in every step so far into the project.
We are grateful to Dr. Ritesh Patel, Satabdwa Majumdar, Arpit K Shrivastav and other reasearch scholars at ARIES for the resources, comments and the guidance throughout.
We also thank Abhas Pradhan, an NIUS senior, for the assistance and support throughout.
And a special mention to Dr. Gauthier Rousseau - author of the OpyFlow library, for constantly helping us with software related issues, keeping in touch and guiding us throughout.

All our codes and works have been uploaded in the github repository given below:
$https://github.com/Arut123/Opticalflowcme$
Mail at arutawesome@gmail.com. Suggestions, comments or feedback is appreciated. Thank You.

References

[1] M. Stix, *The Sun: an introduction*. Springer Science & Business Media, 2004.

[2] T. Howard, *Coronal Mass Ejections-An Introduction*. Springer Science & Business Media, 2011.

[3] D. F. Webb and T. A. Howard, "Coronal mass ejections: Observations," *Living Reviews in Solar Physics*, vol. 9, no. 1, pp. 1–83, 2012.

[4] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, "Learning optical flow," in *European Conference on Computer Vision*, pp. 83–97, Springer, 2008.

[5] R. C. Colaninno and A. Vourlidas, "Analysis of the velocity field of CMEs using optical flow methods," *The Astrophysical Journal*, vol. 652, no. 2, p. 1747, 2006.

[6] M. Bansal, M. Kumar, M. Kumar, and K. Kumar, "An efficient technique for object recognition using Shi-Tomasi corner detection algorithm," *Soft Computing*, vol. 25, no. 6, pp. 4423–4432, 2021.

[7] groussea, "OpyFlow:Python Package for Optical Flow measurements,"

[8] R. Patel, V. Pant, D. Banerjee, A. Kumar, *et al.*, "Onboard automated CME detection algorithm for the visible emission line coronagraph on ADITYA-L1," *Solar Physics*, vol. 293, no. 7, pp. 1–25, 2018.

[9] S. Paris, P. Kornprobst, J. Tumblin, F. Durand, *et al.*, "Bilateral filtering: Theory and applications," *Foundations and Trends® in Computer Graphics and Vision*, vol. 4, no. 1, pp. 1–73, 2009.

[10] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[11] H. Morgan and M. B. Korsós, "Tracing the magnetic field topology of the quiet corona using propagating disturbances," *The Astrophysical Journal Letters*, vol. 933, no. 2, p. L27, 2022.