CV Project Report
Final evaluation

Team: Research >> Intern

Members
Anchit Gupta - 20171041
Vaibhav Garg - 20171005
Neel Trivedi - 20171015

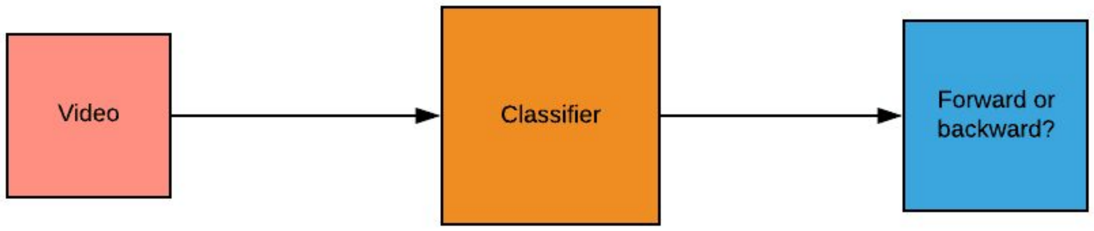
Github link: click here

Summary

Abstract from the paper

We explore whether we can observe Time’s Arrow in a temporal sequence–is it possible to tell whether a video is running forwards or backwards? We investigate this somewhat philosophical question using computer vision and machine learning techniques. We explore three methods by which we might detect Time’s Arrow in video sequences, based on distinct ways in which motion in video sequences might be asymmetric in time. We demonstrate good video forwards/backwards classification results on a selection of YouTube video clips, and on natively-captured sequences (with no temporally-dependent video compression), and examine what motions the models have learned that help discriminate forwards from backwards time.

The basic aim of the paper is to be given a video, predict whether the video is in forward or backward direction.



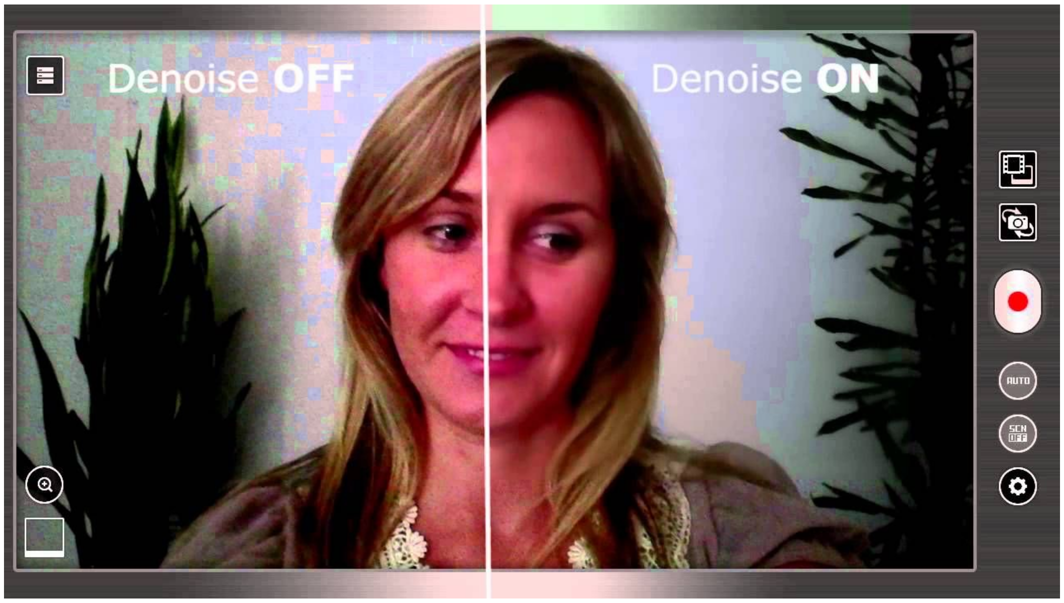
However, the paper does not aim to learn things like a car moves forward in a particular direction or in which direction a door opens, rather it aims to focus on underlying physics, phenomena like *increase in entropy.*

The aim of the paper is to work on videos as described above. To learn temporal semantics and properties that convey collision and destruction always occur forward in time.

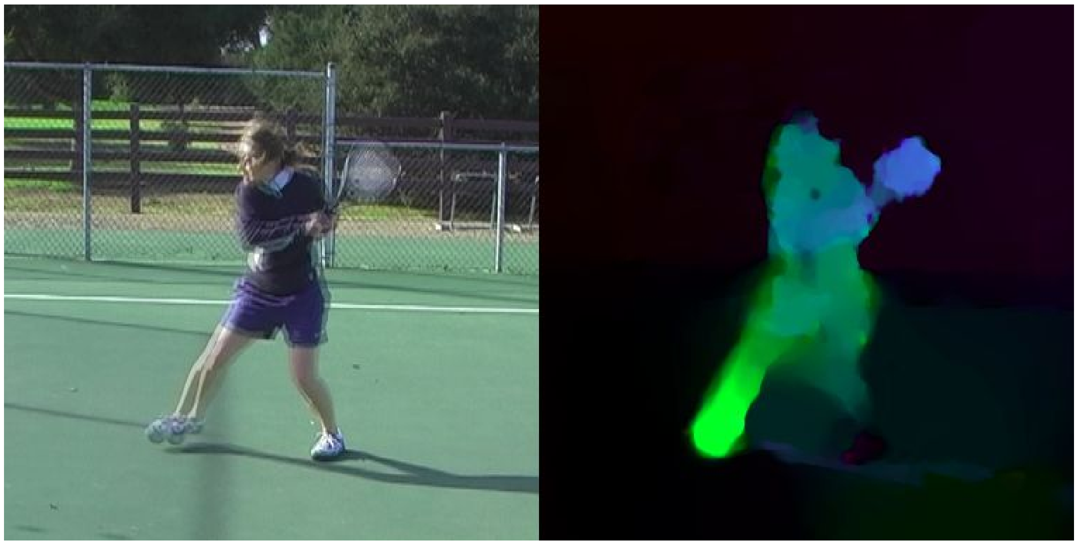
Also, the image shown on the left is obviously in reverse direction since entropy is decreasing. This analysis will be really helpful in later stages.

Applications

- Video Denoising



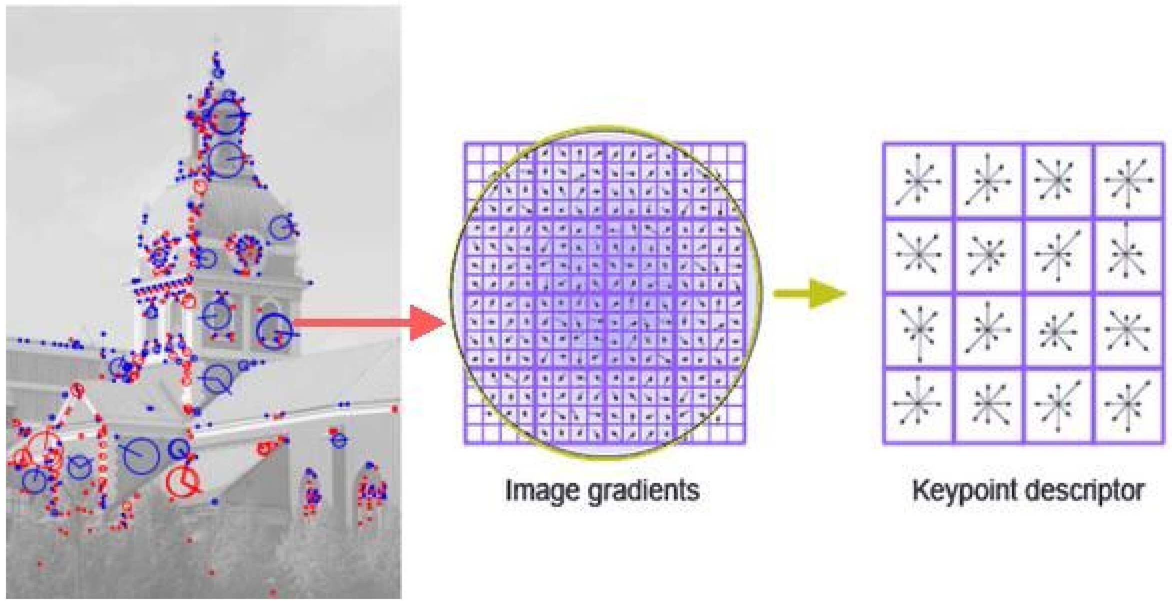
- Optical flow estimation



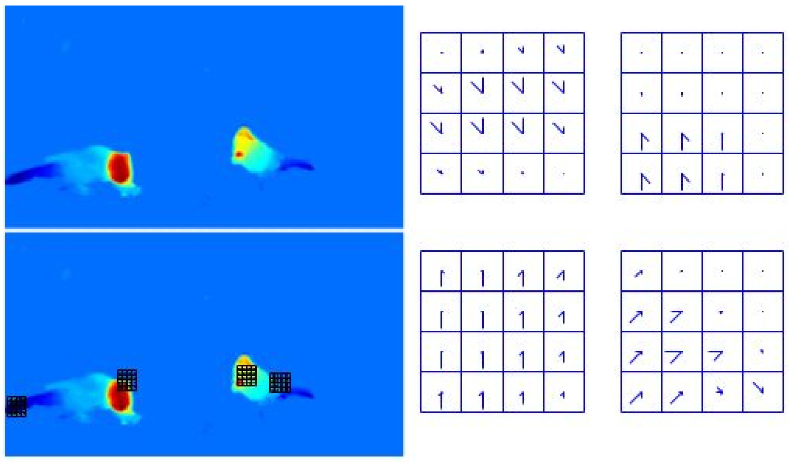
- Causal Inference and many more….

Flow Word

SIFT features

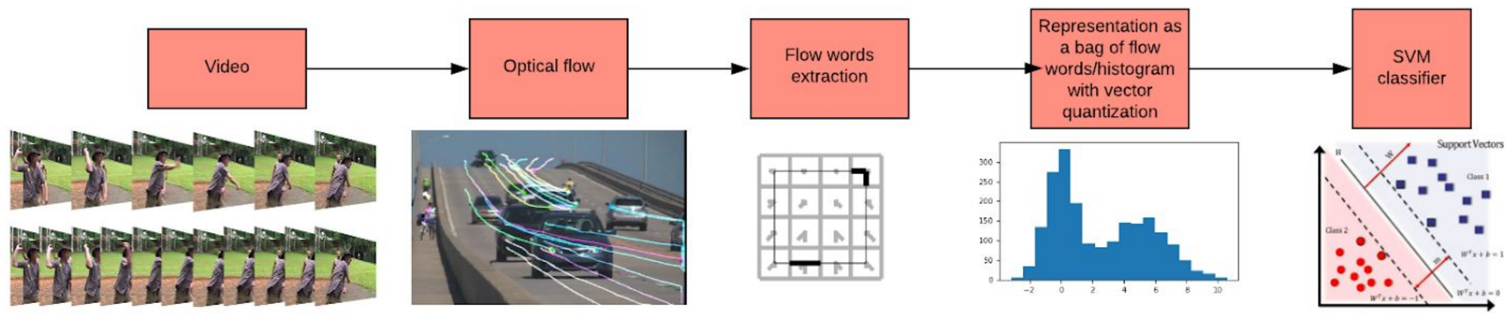


The inspiration for a flow word came from the immense success of SIFT as a feature vector which was able to capture the textures and gradients of a local region in image. ***Could there be a similar vector which could capture local temporal information?*** The paper believes the basis for such information can be captured using optical flow between consecutive frames.



A flow word is nothing but a matrix representing the optical flow between 2 consecutive frames. This vector could be the basis of temporal information.

Main Pipeline



1. We first split the video into frames.

2. For a given frame, we register its next and previous frame to account for camera motion and panning.

3. Optical flow is computed between these registered frames.

4. Using optical flow, flow-words are sampled along with some kind of thresholding.

5. Steps 2 to 3 are done for every frame, giving quite an extensive list of flow words per video.

6. We apply K-Means clustering in order to vector-quantize flow words from all the videos.

7. Each video is then represented as a bag-of-flow-words(normalized histogram) and used as a feature vector for SVM training.

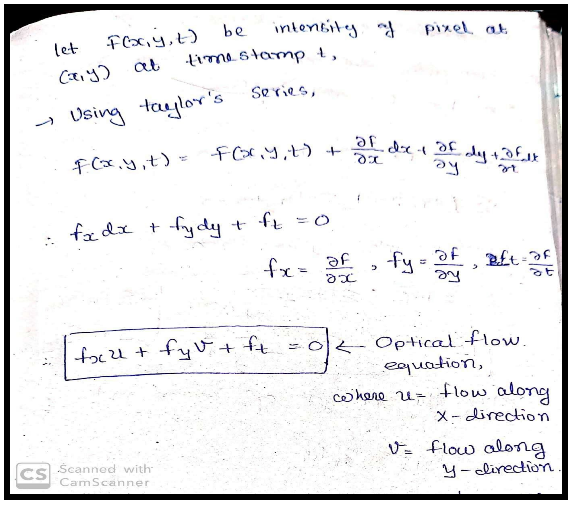
Recap

Following is the summary of what we had accomplished till the last evaluation.

- We started working on the **flow word** based methods to obtained key feature for the given input video
- We were successfully able to convert the video dataset into flow word histogram dataset using optical flow of frame patches and clustering of these flows to obtain histograms (flow words)
- Now that we had the flow word data for the videos we started doing different experiments to train a robust classifier on these flow words to identify whether the given video is being played forward or backward.

Lucas Kanade Optical Flow implementation

- One of the major components of the pipeline of this project is to find the optical flow in the given video.
- We implemented the Lucas Kanade method to find the optical flow between frames
- Lucas Kanade method takes the intensity difference at a given pixel across the frame to find the optical flow in the given frame sequence.



let $F(x,y,t)$ be intensity of pixel at (x,y) at timestamp t,

→ Using taylor's series,

$$F(x,y,t) = F(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

$$\therefore f_x dx + f_y dy + f_t = 0$$

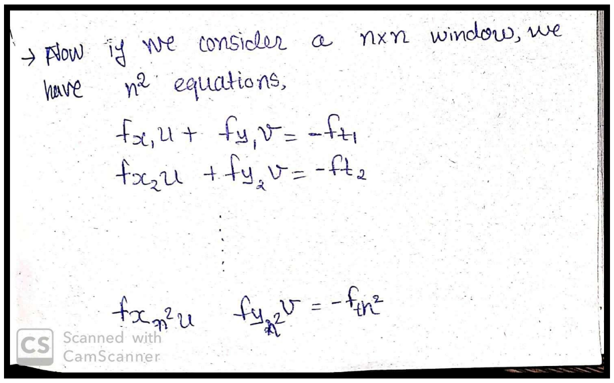
$$f_x = \frac{\partial f}{\partial x},\ f_y = \frac{\partial f}{\partial y},\ f_t = \frac{\partial f}{\partial t}$$

$$\therefore \boxed{f_x u + f_y v + f_t = 0} \leftarrow \text{Optical flow equation,}$$

where u = flow along x-direction

v = flow along y-direction.

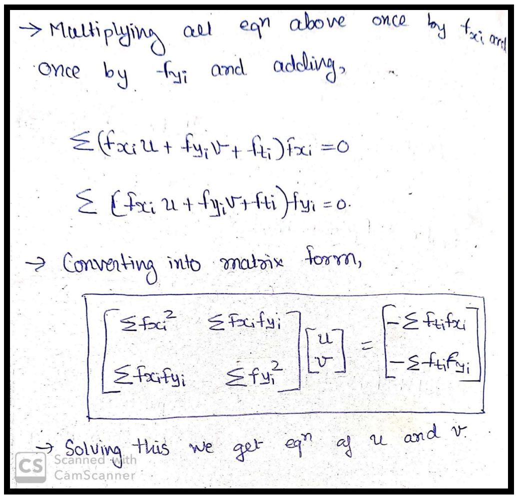
Scanned with CamScanner



→ Now if we consider a $n \times n$ window, we have n^2 equations,

$$f_{x_1} u + f_{y_1} v = -f_{t_1}$$
$$f_{x_2} u + f_{y_2} v = -f_{t_2}$$
$$\vdots$$
$$f_{x_{n^2}} u \quad f_{y_{n^2}} v = -f_{t_{n^2}}$$

Scanned with CamScanner



→ Multiplying all eq^n above once by f_{x_i} and once by f_{y_i} and adding,

$$\Sigma(f_{x_i}u + f_{y_i}v + f_{t_i})f_{x_i} = 0$$

$$\Sigma(f_{x_i}u + f_{y_i}v + f_{t_i})f_{y_i} = 0.$$

→ Converting into matrix form,

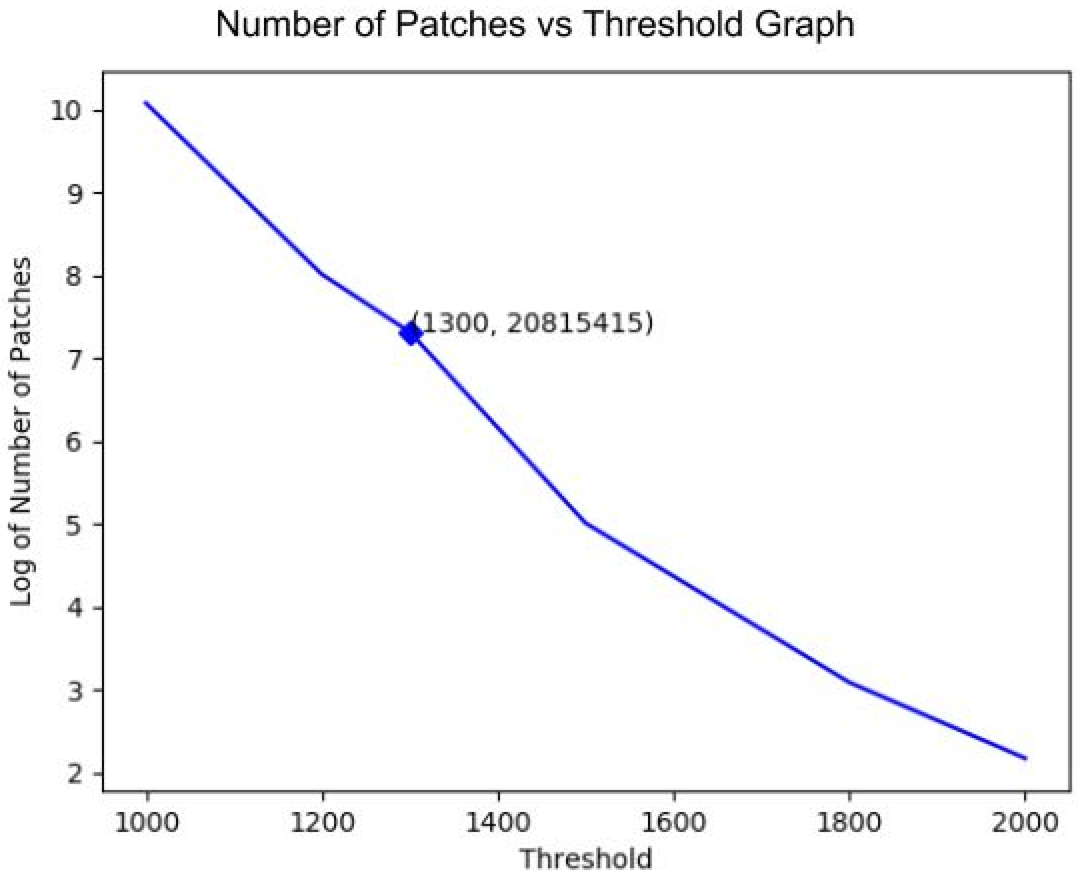
$$\boxed{\begin{bmatrix} \Sigma f_{x_i}^2 & \Sigma f_{x_i}f_{y_i} \\ \Sigma f_{x_i}f_{y_i} & \Sigma f_{y_i}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\Sigma f_{t_i}f_{x_i} \\ -\Sigma f_{t_i}f_{y_i} \end{bmatrix}}$$

→ Solving this we get eq^n of u and v.

Our Experiments(with final results/accuracies)

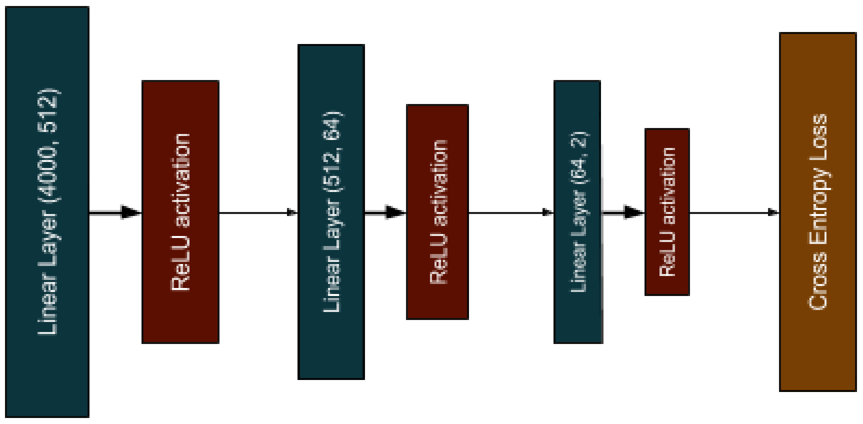
<u>Different Fixed Thresholding</u>

- We tried different values of threshold for L2 norm of patches ranging from 1000 to 2000.
- Problem with the **lower** threshold was that it gave **too many** patches which was difficult to store and process.
- Problem with the **higher** threshold was that it gave **too few** patches. And 0 patches for many videos.
- So the best value for the same was selected as **1300** as it gave the optimal number of patches (**O(10^7)**) as mentioned in the paper.
- Accuracy achieved by this method: **65%**

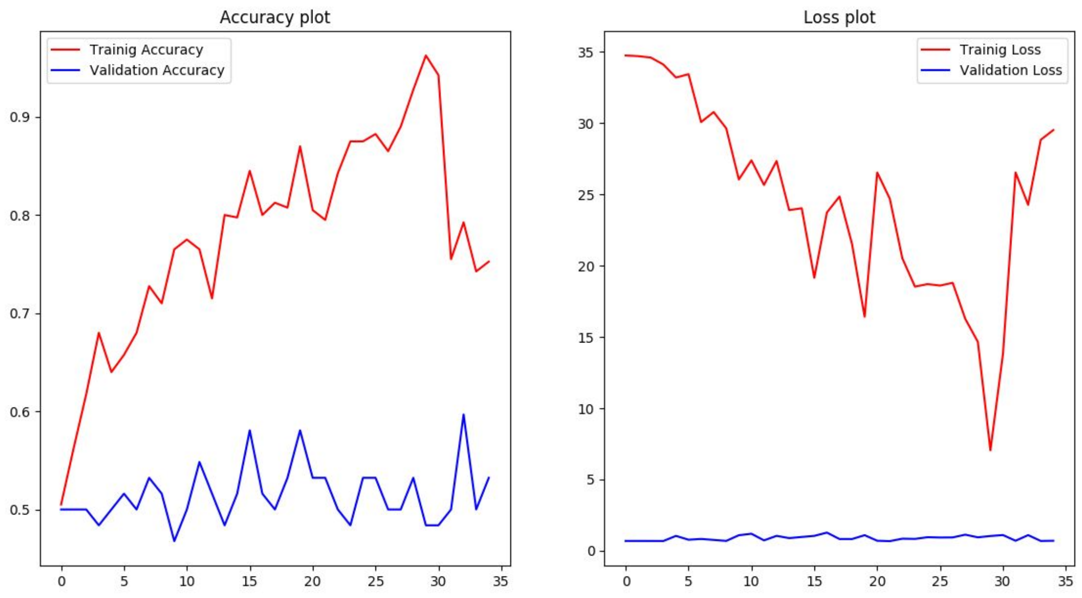


MLP

- We tried to train a Multilayer perceptron classifier for our data as well.
- The architecture of the MLP was simple as shown below,



- We used **SGD** as our optimizer and a batch size of **8**
- We divided the training data into train and validation split with training data having **400** samples and validation data having **62** samples
- Our testing data already had **236** samples.
- The results we achieved for this classifier after training the MLP**:**
    - ***Training Accuracy : 85.5%***
    - ***Validation Accuracy : 53.22%***
    - ***Testing Accuracy : 55.08%***
- The reason for low testing accuracy is the small amount of data and overfitting of the model on the training set.



<u>KNN</u>

- For the baseline experiments we also tried using the K-nearest neighbour algorithm (with k = 3) to classify the videos.
- Initial results were **74%** accuracy on training data and **1.6%** on testing data. The reason for such low accuracy is the very high dimensions of input data (4000 length histograms)
- Hence we tried to reduce the dimensions of the data using PCA and reduced it to 10 and we were able to get accuracy of **17%** on testing data. The reason for such low accuracy even after the dimensionality

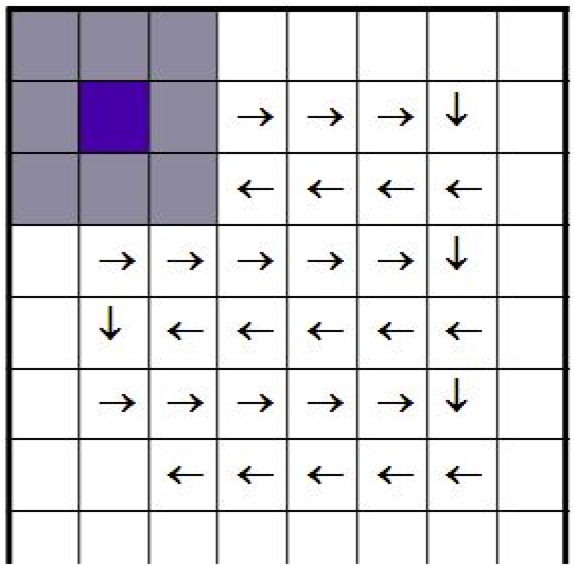
reduction is the inherent complexity of data and hence simple models like KNN can not easily classify the data.

Adaptive Thresholding

- To eliminate the dependency of thresholding we tried Adaptive thresholding.
- To have an order of **10^7 patches**, 100 patches with the greatest norm were selected from each frame of every video for 4 different orientations of videos mentioned in the paper.
- **Accuracy achieved:** 55%

Different window size and stride

One of the reasons we believed we couldn't get accuracies as good as the paper was that our window size was too small. So we also ran an iteration with a window size=24.



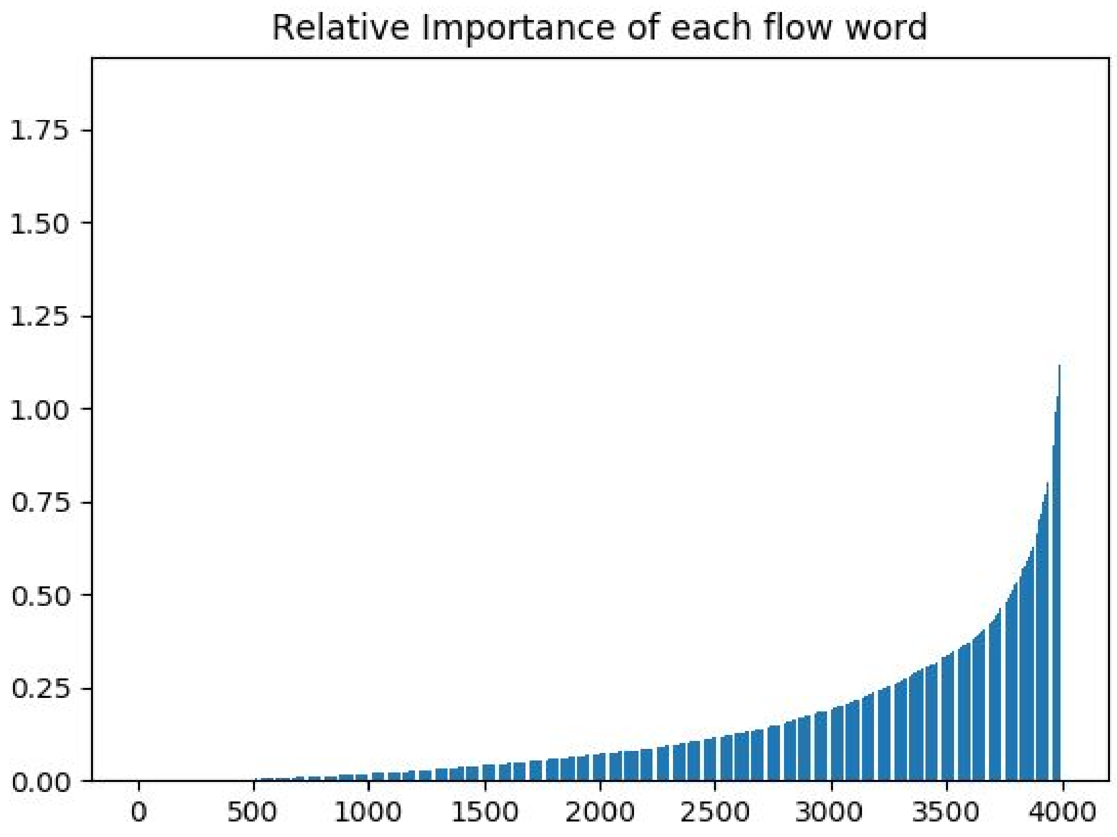
But since the flow-word size now increased significantly, we also had to increase the stride length significantly to 16.

Accuracy achieved: 52%

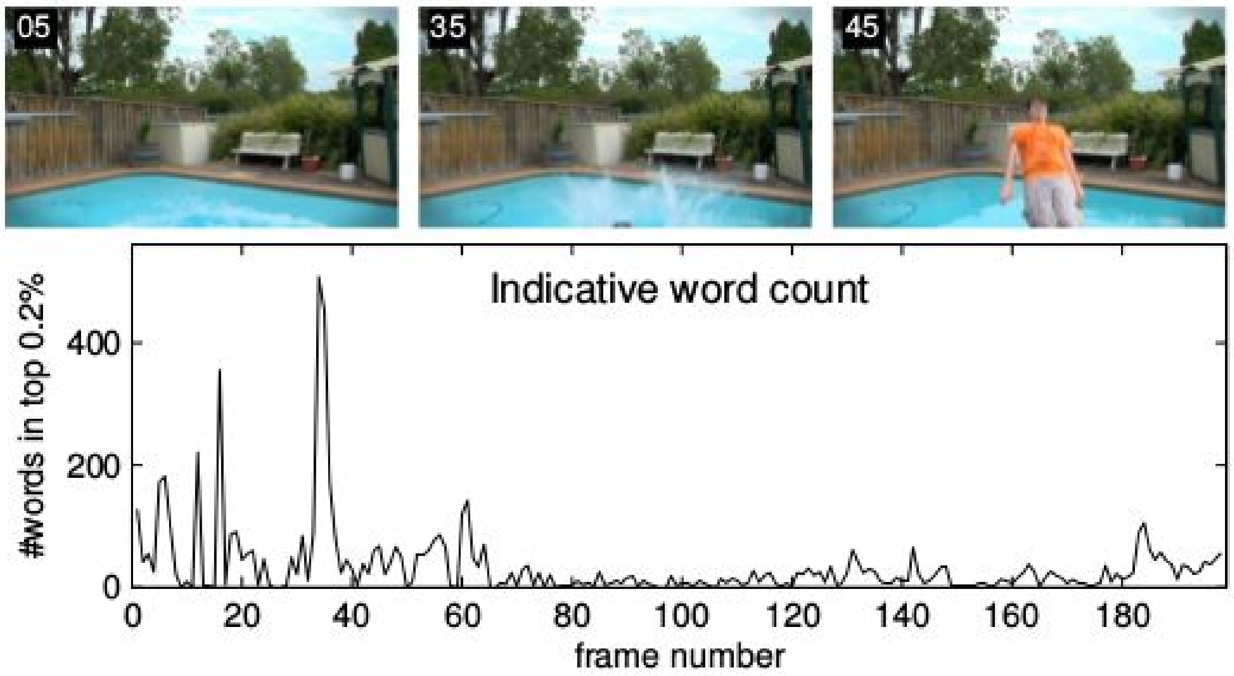
We believe the reason for such a bad accuracy could be due to very less number of total-flow words sampled, since only O(10^5) flow-words were achieved in total, a very low number of flow-words per video are not good enough to represent the video.

Deeper Analysis

One of the great things about SVM is that it enables us to see which feature has played the most important role in the classification process.



As you can see from the graph, there is a very clear spectrum showing how some flow-words are way more important than any other. In the same way, how some words are more important than others in a text document. **But what are these important flow-words?**



As mentioned earlier, the important flow-words in fact turned out to be the ones corresponding to motion like a *splash of water.* Such motion is the one which is most indicative in the classification process.