Approach 1: Extracting features using a pretrained CNN model(inception) followed by GRUs for classification:

Approach 2: A custom Conv3D network:

Here, we feed the images of the videos to a custom conv3d network. After playing around with various architectures, the following network seems to be the best performer:

In the following two approaches we compute optical flow on each frame of the video and feed that to a 3D Convolution network. But before we jump in to the actual implementation, let's look at what OF is and how flow vectors are obtained mathematically.

Optical Flow:

When you are looking at motion in videos, optical flow should definitely be considered. It is defined as the apparent motion of individual pixels on the image plane. It often serves as a good approximation of the true physical motion projected onto the image plane. Most methods that compute optical flow assume that the color/intensity of a pixel is invariant under the displacement from one video frame to the next. Optical flow provides a concise description of both the regions of the image undergoing motion and the velocity of motion.

So, we are mainly interested in looking at the intensity variation between two frames. A voxel at location (x,y,t) with intensity I(x,y,t) has moved to Δx , Δy & Δt . Following the brightness constancy constraint:

$$I(x, y, t) = I(x + \Delta x,\ y + \Delta y,\ t + \Delta t)$$

Assuming the movement to be small, the image constraint at I(x,y,t)} with Taylor series can be developed to get:

$$I(x + \Delta x,\ y + \Delta y,\ t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\delta I}{\delta y}\Delta y + \frac{\delta I}{\delta t}\Delta t$$

We only keep the first order terms and ignore everything else.

Dividing by Δt , we get:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0$$

or:

$\frac{\partial I}{\partial x} V_x + \frac{\partial I}{\partial y} V_y + \frac{\partial I}{\partial t} = 0$, where Vx, Vy are the components of the velocity or optical flow of $I(x, y, t)$, while $\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}$ & $\frac{\delta I}{\delta t}$ are the derivatives of the image at (x, y, t) in the corresponding directions. Writing it succinctly as follows:

$$I_x V_x + I_y V_y = - I_t$$

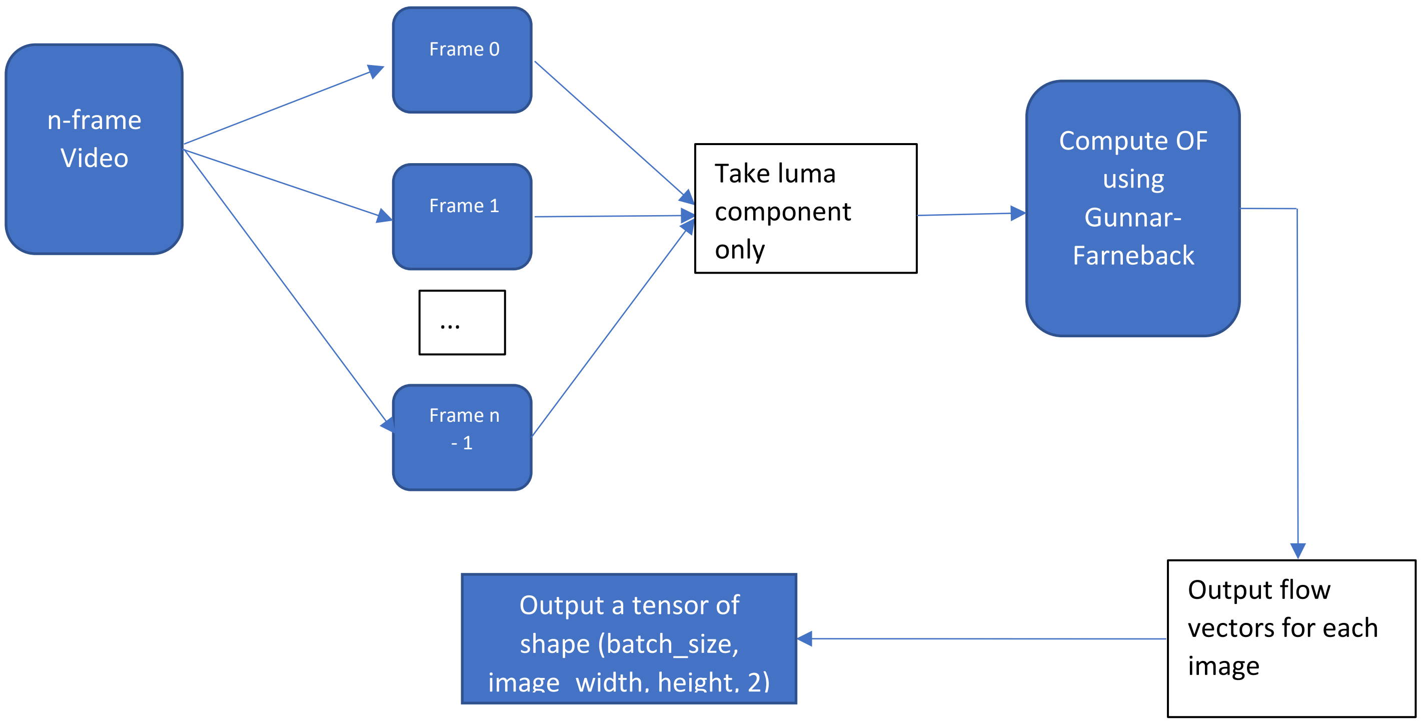
Or $\nabla I . V = - I_t$

There are various ways to solve the above problem. OpenCV has two ways implemented: Lucas-Kanade, which computes sparse OF and the Gunnar-Farneback OF which computes dense OF i.e., a flow vector is obtained for every pixel of the image. So, for a (224, 224) image, the output will be (224, 224, 2).

We went with dense OF in our implementation. There is also a new improved method of computing optical flow viz. the Semi-global matching OF proposed by Heiko Hirschmüller. Though we have not tried it, we have reserved this as a future scope.

Approach 3: Optical flow followed by Conv3D network

The flow of the generator code for this approach is as follows:


n-frame Video
Frame 0
Frame 1
...
Frame n - 1
Take luma component only
Compute OF using Gunnar-Farneback
Output flow vectors for each image
Output a tensor of shape (batch_size, image_width, height, 2)

With this approach alone, we were able to obtain an accuracy of 93% on the dataset given by Upgrad.

Approach 4: Quo Vadis, Action Recognition:

In 2017, a seminal paper was released by Joao Carreira & Andrew Zisserman which talks about how to tackle action classification while looking at some of the popular datasets like UCF-101, HMDB-51 & the Kinetics Human Action Video dataset. They introduced a new Two-stream Inflated 3D ConvNet(I3D) that is based on 2D ConvNet inflation: filters and pooling kernels of very deep image classification ConvNets are expanded into 3D, making it possible to learn seamless spatio-temporal feature extractors from video while leveraging successful ImageNet architecture designs and even their parameters.

I3D builds upon state-of-the-art image classification architectures, but inflates their filters and pooling kernels (and optionally their parameters) into 3D, leading to very deep, naturally spatiotemporal classifiers.

Here are the various architectures considered in the paper:

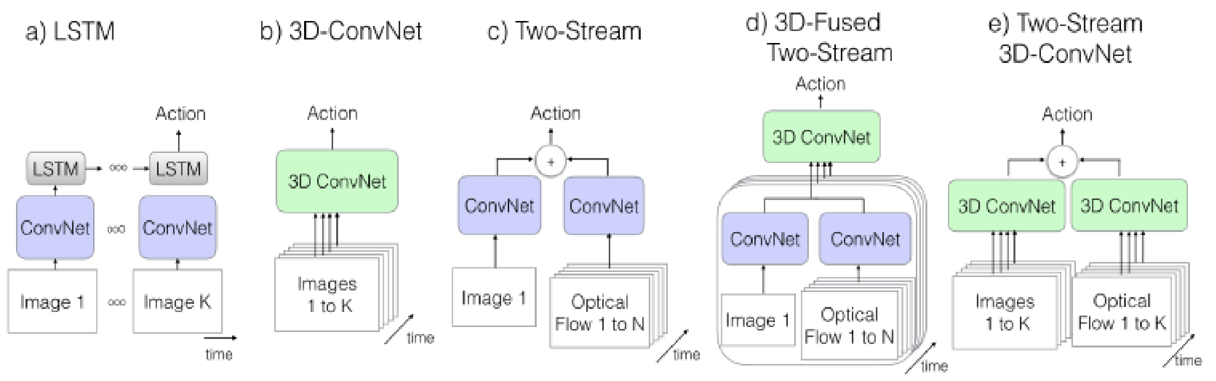
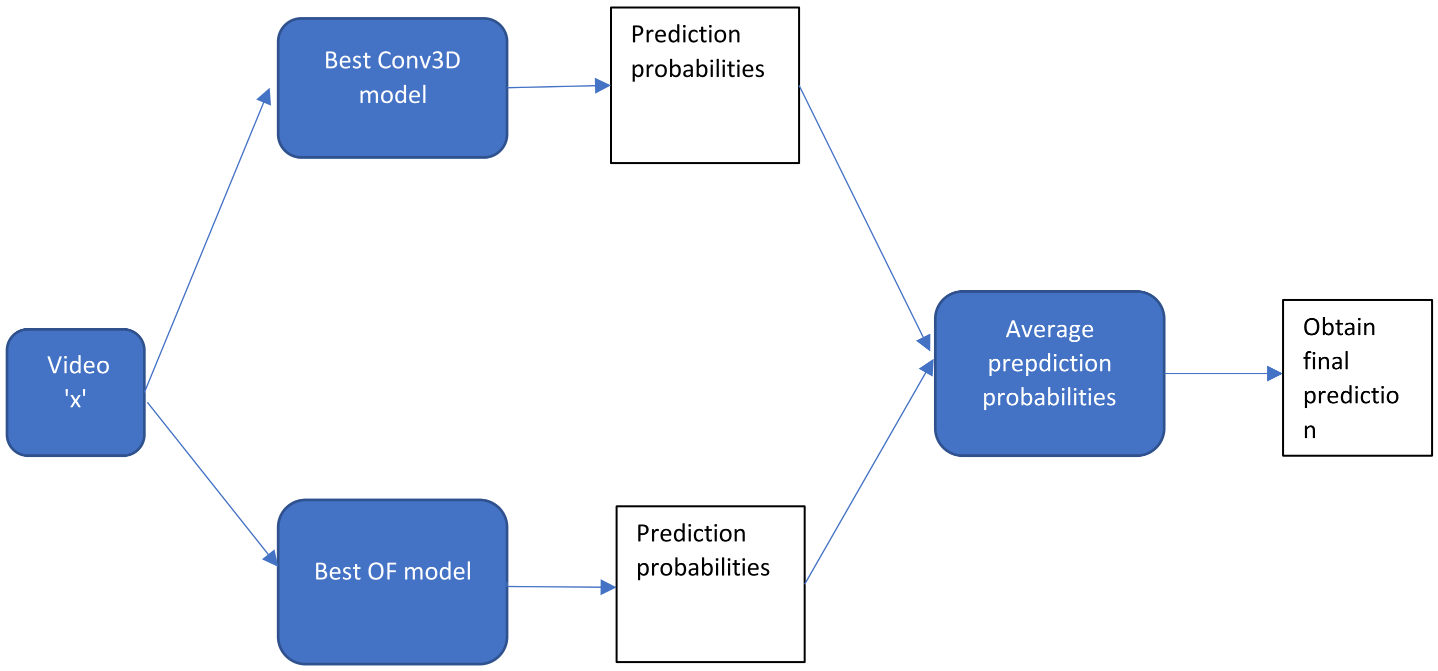


Figure 2. Video architectures considered in this paper. K stands for the total number of frames in a video, whereas N stands for a subset of neighboring frames of the video.

Of the 5 shown above, the last one is their novelty i.e., expand the 2d conv from the existing two-stream architecture shown in c) to a 3D Conv. While a 3D ConvNet should be able to learn motion features from RGB inputs directly, it still performs pure feedforward computation, whereas optical flow algorithms are in some sense recurrent (e.g., they perform iterative optimization for the flow fields)

In our implementation, we trained the two models separately and cached their respective best models. Finally, on the validation data, we averaged the prediction probabilities from both the models and obtained the final result. It can be summarized with a diagram as follows:



With the added complexity, we only got a minor improvement of 1% and the final accuracy obtained was 94%. According to us with the current implementation, the added complexity does not seem to be worth the hassle. Perhaps, if we could tweak our approach slightly and build upon this method, the results could be different. This will be bookmarked as another area of improvement for the future.