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

In this approach we can use a pre-trained network like Inceptionv3 or VGG to extract meaningful features from each frame of the video. These feature for each frame from the video are feed into RNN as sequence of the features.

For this approach we have written a custom generator which will give a batch of videos containing features for each frame using InceptionV3 pretrained model.

These batch are input to the RNN model containing the following layers

1. GRU layer with 16 features
2. Dense layer with 32 features and relu activation
3. Dense layer with 5 features and softmax activation

Below are the mentioned experiments done the above model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Conv3D** | **Result** | **Decision + Explanation** |
| **1** | **RCNN** | **Throws Generator error** | **Started with initial one GRU Layer with 16 features and 5 features softmax dense layer** |
| **2** | **RCNN** | **Throws Input shape error** | **Fixed issues with generator** |
| **3** | **RCNN** | **Accuracy: 0.70** | **Fixed issue with generator to get the desiered input shape for the model** |
| **4** | **RCNN** | **Accuracy: 0.60** | **Added another GRU layer with double the feature from previous layer** |
| **5** | **RCNN** | **Accuracy: 0.72** | **Removed the GRU as it did not improve the accuracy** |
| **6** | **RCNN** | **Accuracy: 0.65** | **Added another GRU layer with half the feature from previous layer** |
| **7** | **RCNN** | **Accuracy: 0.70** | **Removed the GRU as it did not improve the accuracy** |
| **8** | **RCNN** | **Accuracy: 0.71** | **Add 30% dropout after dense layer which is after Conv3D layers** |
| **9** | **RCNN** | **Accuracy: 0.82** | **Added a Dense Layer with 32 features** |
| **10** | **RCNN** | **Accuracy: 0.72** | **Added Cropping to all frames of a video in generator** |
| **11** | **RCNN** | **Accuracy: 0.81** | **Removed Croping as it did not seem to improve accuracy** |
| **12** | **RCNN** | **Accuracy: 0.68** | **Added masking to video which did not have 1:1 acpect ratio** |
| **13** | **RCNN** | **Accuracy: 0.82** | **Removed the masking as it had no effect** |

# **Approach 2: A custom Conv3D network:**

In this approach we can have used a custom Conv3D network with the features increasing with layers with relu activation and 3x3x3 filter size.

For this approach we have written a custom generator which will give a batch of videos containing frames which have been resized to (244,244,3) and normalized by dividing each pixel by 255.

After playing around with various architectures, the following network seems to be the best performer

1. Conv3D layer with 16 features
2. Max Pooling layer
3. Conv3D layer with double the features from the previous layer
4. Max Pooling layer
5. Conv3D layer with double the features from the previous layer
6. Max Pooling layer
7. Conv3D layer with double the features from the previous layer
8. Max Pooling layer
9. Flatten layer
10. Dense layer with 216 features and relu activation
11. Dense layer with 5 features and softmax activation

Below are the mentioned experiments done the above model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Conv3D** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | **Throws Generator error** | **Started with initial one Conv3D layer with a 3x3x3 filter 16 features and 5 features softmax dense layer** |
| **2** | **Conv3D** | **Throws Input shape error** | **Fixed issues with generator** |
| **3** | **Conv3D** | **Accuracy: 0.42** | **Fixed issue with generator to get the desiered input shape for the model** |
| **4** | **Conv3D** | **Accuracy: 0.57** | **Added another Conv3D layer with double the feature from previous layer and MaxPolling3D** |
| **5** | **Conv3D** | **Accuracy: 0.61** | **Added Dense layer with double the features from the previous layer** |
| **6** | **Conv3D** | **Accuracy: 0.65** | **Added another Conv3D layer with double the feature from previous layer and MaxPolling3D** |
| **7** | **Conv3D** | **Accuracy: 0.70** | **Added another Conv3D layer with double the feature from previous layer and MaxPolling3D** |
| **8** | **Conv3D** | **Accuracy: 0.75** | **Added another Conv3D layer with double the feature from previous layer and MaxPolling3D** |
| **9** | **Conv3D** | **Accuracy: 0.79** | **Added another Conv3D layer with double the feature from previous layer** |
| **10** | **Conv3D** | **Accuracy: 0.78** | **Added another Conv3D layer with double the feature from previous layer** |
| **11** | **Conv3D** | **Accuracy: 0.74** | **Added another Conv3D layer with double the feature from previous layer** |
| **12** | **Conv3D** | **Accuracy: 0.70** | **Added another Conv3D layer with double the feature from previous layer** |
| **13** | **Conv3D** | **Accuracy: 0.80** | **Removed the last 3 Conv3D layers as it did not seem to improve the accuracy and lead to overfitting the model** |
| **14** | **Conv3D** | **Accuracy: 0.84** | **Added a Flattern layer after the Conv3D Layers** |
| **15** | **Conv3D** | **Accuracy: 0.84** | **Add 20% dropout after dense layer which is after Conv3D layers** |
| **16** | **Conv3D** | **Accuracy: 0.83** | **Incresed the dropout to 40%** |
| **17** | **Conv3D** | **Accuracy: 0.75** | **Added Cropping to all frames of a video in generator** |
| **18** | **Conv3D** | **Accuracy: 0.85** | **Removed Croping as it did not seem to improve accuracy** |
| **19** | **Conv3D** | **Accuracy: 0.70** | **Added masking to video which did not have 1:1 acpect ratio** |
| **20** | **Conv3D** | **Accuracy: 0.84** | **Removed the masking as it had no effect** |

Now we have two approaches to go and in both we compute optical flow on each frame of the video and feed that to a 3D Convolution network. But before we jump into 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 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 , & . Following the brightness constancy constraint:

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

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

Dividing by , we get:

or:

, where Vx, Vy are the components of the velocity or optical flow of , while are the derivatives of the image at (x, y, t) in the corresponding directions. Writing it succinctly as follows:

Or

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:

Feed to Conv3d network

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)

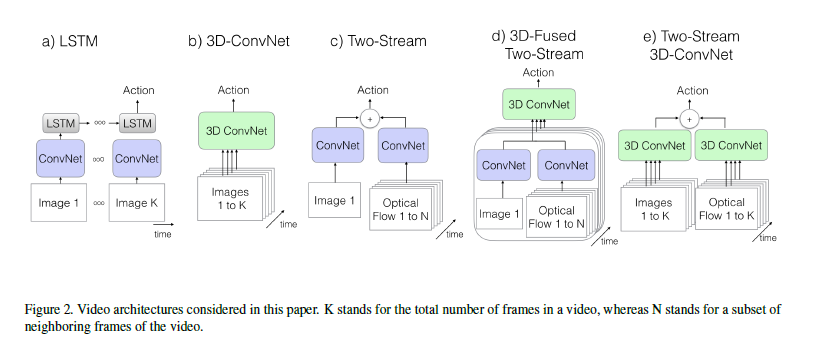
With this approach alone, the best accuracy we were able to obtain was 94% on the dataset given by Upgrad.

# **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:



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 result. It can be summarized with a diagram as follows:

Best Conv3D model

Video 'x'

Prediction probabilities

Best OF model

Prediction probabilities

Average prepdiction probabilities

Obtain final prediction

With the added complexity, the final accuracy obtained was 93%. We feel that 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 in the future.