**Experiment 1 (base model)**

Batch size = 10

frames = [15,16,17,18,19,20,21,22]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 42,383,554

loss: 13.3917 - categorical\_accuracy: 0.1692 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: The batch size which we have used in this experiment is not an optimal value as the machine doesn’t give any error. So, we are not utilizing the GPU to full capacity. Let’s increase the batch size to 25.

**Experiment 2**

Batch size = 25

frames = [15,16,17,18,19,20,21,22]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 42,383,554

loss: 12.9955 - categorical\_accuracy: 0.1937 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: Even after increasing the batch size to 25, we didn’t receive any error. Let’s increase the batch size to 50.

**Experiment 3**

Batch size = 50

frames = [15,16,17,18,19,20,21,22]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 42,383,554

loss: 12.5757 - categorical\_accuracy: 0.2198 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: Even after increasing the batch size to 50, we didn’t receive any error. Let’s increase the batch size to 100.

**Experiment 4**

Batch size = 100

frames = [15,16,17,18,19,20,21,22]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 42,383,554

loss: 13.0102 - categorical\_accuracy: 0.1928 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: I would like to fix the batch size to 100 as the data is easily fitting into the memory. Even though it is not an optimal value we can fit the entire validation data at a time and 100 is not a small number either.

On observing around 15 videos, I found out that the actual action is happening from 6th or 7th frame to 19th or 20th frame. So, let’s try with alternative images between 6 to 20 images.

**Experiment 5**

Batch size = 100

frames = [6, 8, 10, 12, 14, 16, 18, 20]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 42,383,554

loss: 13.2596 - categorical\_accuracy: 0.1773 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: There is no significant improvement. Let’s try with alternate images from 6 to 25 frames

**Experiment 6**

Batch size = 100

frames = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 54,492,994

loss: 13.1265 - categorical\_accuracy: 0.1856 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: There is no significant improvement. Let’s try with all images from 6 to 25 frames

**Experiment 7**

Batch size = 100

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 115,040,194

loss: 13.1265 - categorical\_accuracy: 0.1856 - val\_loss: 12.4109 - val\_categorical\_accuracy: 0.2300

**Observation**: There is no significant improvement. Let’s add a max pooling layer to reduce parameters.

**Experiment 8**

Batch size = 100

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 13,424,194

**Observation**: Trainable parameters were reduced to 13 million, but while training the model received memory error. So lets reduce the batch size to 75.

**Experiment 9**

Batch size = 75

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 13,424,194

loss: 12.8875 - categorical\_accuracy: 0.2004 - val\_loss: 12.8945 - val\_categorical\_accuracy: 0.2000

**Observation**: We have poor accuracy. Let’s add one more Conv3D layer and Max pooling 3D layer.

**Experiment 10**

Batch size = 75

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 10 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 2,896,284

loss: 1.3919 - categorical\_accuracy: 0.3568 - val\_loss: 1.3197 - val\_categorical\_accuracy: 0.4600

**Observation**: We can see significant improvement, but the accuracy is not up to the mark. Let’s add one more Conv3D layer and Max pooling 3D layer. Let’s also try with batch size of 85.

**Experiment 11**

Batch size = 85

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 3

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 10 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 20 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 343,024

loss: 1.3506 - categorical\_accuracy: 0.4709 - val\_loss: 1.4485 - val\_categorical\_accuracy: 0.4000

**Observation**: Accuracy at this point is not good enough. Let’s increase the number of epoch to 10 and let’s also try with batch size of 90.

**Experiment 12**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 10

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 5 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 10 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 20 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 1 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 343,024

loss: 1.3040 - categorical\_accuracy: 0.3333 - val\_loss: 1.2708 - val\_categorical\_accuracy: 0.4000

**Observation**: Accuracy at this point is not good enough. Let’s increase the filters to 16, 32 and 64 and add 1 more layer with 64 neurons.

**Experiment 13**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 10

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 2 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 2,483,637

loss: 2.4452 - categorical\_accuracy: 0.0000e+00 - val\_loss: 3.5275 - val\_categorical\_accuracy: 0.2500

**Observation**: We have an accuracy of 25 which is very poor. Let’s add one more Conv3D layer with 128 filter and increase number of epochs to 3 0 for better training.

**Experiment 14**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 30

Used Adam optimizer with 0.001 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 2 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 665,141

**Observation**: The best val\_categorical\_accuracy is 45, lets increase the number of epochs to 45 for more training to happen and increase the learning rate for faster training.

**Experiment 15**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by dividing with 255

epochs = 45

Used Adam optimizer with 0.01 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 2 dense layer with 64 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 665,141

**Observation**: The best val\_categorical\_accuracy is 50, lets increase the number of neurons n dense layers to 128 and lets use min-max normalization technique to see if we can get better accuracy

**Experiment 16**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by subtracting minimum pixel value and divide the same with range.

epochs = 45

Used Adam optimizer with 0.01 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

Flatten the activation and feed to next layer which has 2 dense layer with 128 neurons and Relu activation function

SoftMax layer with 5 neurons

Trainable params: 1,251,317

**Observation**: The best val\_categorical\_accuracy is 50, lets try adding bacth normalization and Dropouts for better accuracy.

**Experiment 17**

Batch size = 90

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by subtracting minimum pixel value and divide the same with range.

epochs = 45

Used Adam optimizer with 0.01 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

Flatten the activation and feed to 2 dense layer with 128 neurons and Relu activation function each with Batch Normalization and Dropout(0.5)

SoftMax layer with 5 neurons

Trainable params: 1,251,797

**Observation**: Got ResourceExhaustedError, lets reduce the batch size to 50.

**Experiment 18**

Batch size = 50

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by subtracting minimum pixel value and divide the same with range.

epochs = 45

Used Adam optimizer with 0.01 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

Flatten the activation and feed to 2 dense layer with 128 neurons and Relu activation function each with Batch Normalization and Dropout(0.5)

SoftMax layer with 5 neurons

Trainable params: 1,251,797

**Observation**: The best val\_categorical\_accuracy is 64, let’s try with SGD optimizer weather we can get better accuracy.

**Experiment 19**

Batch size = 50

frames = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]

y, z = (120, 160)

normalized the image by subtracting minimum pixel value and divide the same with range.

epochs = 45

Used SGD optimizer with 0.01 learning rate

**Model as below**

1 conv3D layer with 16 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 32 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 64 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

1 conv3D layer with 128 filters of size (2, 2, 2), input shape and Relu activation

1 max pooling 3d layer of size (2,2,2)

1 Batch Normalization layer

Flatten the activation and feed to 2 dense layers with 128 neurons and Relu activation function each with Batch Normalization and Dropout (0.5)

SoftMax layer with 5 neurons

Trainable params: 1,251,797

**Observation**: The best val\_categorical\_accuracy is 72