International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Surveillance Automation through DeepLearning

Kada srisai, Pyla Vamsi, Maddhukuri Monika

[1]BTech, Department of CSE, Sanketika Vidya Parishad Engineering College
[2]BTech, Department of CSE, Sanketika Vidya Parishad Engineering College
[3]BTech, Department of CSE, Sanketika Vidya Parishad Engineering College
[4]Guided by Mrs G.Vijaya Lakshmi, M.TECH, Assistant professor,Department of CSE.

Abstract - *Surveillance security is a very tedious and time-consuming job. In this paper, we will build a system to automate the task of analyzing video surveillance. We will analyze the video feed in real-time and identify any abnormal activities like violence or theft.There is a lot of research going on in the industry about video surveillance among them; the role of CCTV videos has overgrown. CCTV cameras are placed all over the places for surveillance and security*

***KeyWords*: 3D-convolutionnetwork, Spatiotemporal autoencoder, Motion patterns, Deep neural networks, Frame Extraction, Thresholdvalue.**

1.INTRODUCTION

The role of video streams from CCTV cameras is equally important as other sources like social media data, sensor data, agriculture data, medical data and data evolved from space research. With the rapid growth of video data, there is an increasing need not only for recognition of objects and their behaviour, but in particular for detecting the rare, interesting occurrences of unusual objects or suspicious behaviour in the large body of ordinary data. Finding such abnormalities in videos is crucial for applications ranging from automatic quality control to visual surveillance. Meaningful events that are of interest in long video sequences, such as surveil-lance footage, often have an extremely low probability of occurring. As such,manually detecting such events, or anomalies, is a very meticulous job that of-ten requires more manpower than is generally available. This has prompted the need for automated detection and segmentation of sequences of interest. How-ever, present technology requires an enormous amount of configuration efforts on each video stream prior to the deployment of the video analysis process, even with that, those events are based on some predefined heuristics, which makes the detection model difficult to generalize to different surveillance scenes. Video data is challenging to represent and model due to its high dimension- ality, noise, and a huge variety of events and interactions. Anomalies are also highly contextual, for example, running in a restaurant wouldbe an anomaly, but running at a park would be normal. Moreover, the definition of anomaly can be ambiguous and often vaguely defined. A person may think walking around on a subway platform is normal, but some may think it should be flagged as an anomaly since it could be suspicious. These challenges have made it difficult for machine learning bmethods to identify video patterns that produce anomalies in real-world applications Surveillance videos have a major contribution in unstructured big data. CCTV cameras are implemented in all places where security having much importance. Manual surveillance seems tedious and time consuming. Security can be defined in different terms in different contexts like theft identification, violence detection, chances of explosion etc. In crowded public places the term security covers almost all type of abnormal events. Among them violence detection is difficult to handle since it involves group activity. The anomalous or abnormal activity analysis in a crowd video scene is very difficult due to several real world constraints.

Artificial intelligence paves the way for computers to think like human. Machine learning makes the way more even by adding training and learning components. The availability of huge dataset and high performance computers lead the light to deep learning concept, which extract automatically features or the factors of variation that distinguishes objects from one another. Among the various data sources which contribute to terabytes of big data, video surveillance data is having much social relevance in today's world. The widespread availability of surveillance data from cameras installed in residential areas, industrial plants, educational institutions and commercial firms contribute towards private data while the cameras placed centers, public conveyances and religious places contribute to public data. in public places such as city

2. Existing System

A. TRADITIONAL CAMERAS AND MANUAL SURVEILLANCE

Surveillance systems may be used as smart watchdogs, which are used to observe a restricted area, identify trespassers, unauthorized entries and therefore identify authorization. They are often connected to a recording device (Surveillance camera) and IP network watched by a security guard or law enforcement officer. Manual operation is needed for this traditional cameras and a person want to monitor the video footage continuously. In normal cases what we are doing is, if any abnormal

activity took place we will check for the video footage and find the culprit. It is the manual way of video surveillance. This method requires human personnel to monitor the camera footage. Also real-time alert is not possible in this method. We have to check the video after a particular unauthorized event has occurred and we want to find the unauthorized one. Also, the video data generated is in huge amount, hence monitoring the video footage is becoming a difficult task and time-consuming. By using this method we didn't get a real-time status of the security places also the lack of appropriate alert systems.

B. VIDEO SURVEILLANCE USING THE RASPBERRY PI ARCHITECTURE

In this method, Raspberry PI architecture is used as a surveillance system. Raspberry PI B+ model can be used for this purpose; it will have the features of the basic computer. The hardware component included in the project was Raspberry Pi processor, stepper motor, webcam, PIR sensors and GSM module. The video footage is captured using the webcam and PIR sensors that are interfaced using a Raspberry Pi. The footage can be locally and remotely visualized. A 360° Surveillance can be achieved using a stepper motor. Video is processed in the Raspberry Pi and data is send to the officials. PIR sensor is used for motion detection. If the PIR sensor detects any motion the raspberry pi will send notification using the GSM module. LCD and alarm are used to give the surveillance message locally. Data can send anywhere at any time because of the wireless medium. This method can be used in hospitals, city buses or WIFI enabled train. But it does not detect intruder person by identifying authorized or unauthorized personal. It just detects the motion using PIR sensors and sending notification. It requires hardware components and its installation. Also, require power consumption.

3. PROPOSED SYSTEM

In this proposed system, we will introduce a spatio temporal autoencoder, which is based on a 3D convolution network. The encoder part extracts the spatial and temporal information, and then the decoder reconstructs the frames. The abnormal events are identified by computing the reconstruction loss using Euclidean distance between original and reconstructed batch.

(i).Spatio Temporal Auto Encoder:

which utilizes deep neural networks to learn video representation automatically and extracts features from both spatial and temporal dimensions by performing 3-dimensional convolutions. Autoencoders, as the name suggests, consist of two stages: encoding and decoding. It was first used to reduce dimensionality by setting the number of encoder output units less than the input

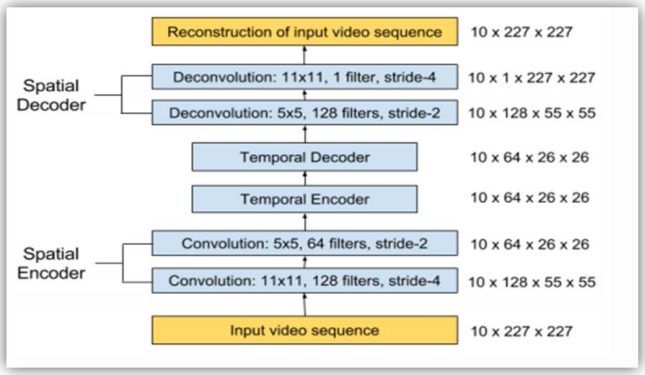

Figure1: Structural expansion for Spatio-Temporal Auto-Encoder

The model is usually trained using back-propagation in an unsupervised manner, by minimizing the reconstruction error of the decoding results from the original inputs. With the activation function chosen to be nonlinear, an autoencoder can extract more useful features than some common linear transformation methods such as PCA.

(ii),Network Architecture:

Encoder: The encoder part extracts the spatial and temporal information by extracting frames from the given input video

Decoder:The decoder reconstruct the frames again into the normal form.and reconstruct the video.

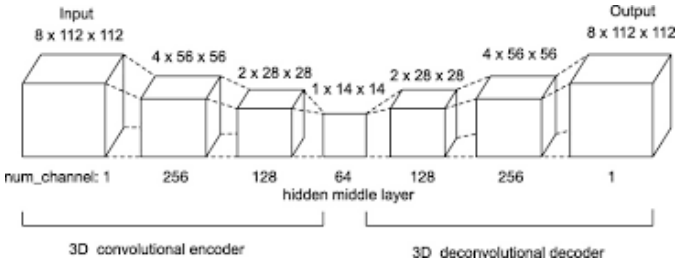

Figure2: 3D-convolution Auto encoder

In this deep learning project, we train an autoencoder for abnormal event detection. We train the autoencoder on normal videos. We identify the abnormal events based on the euclidean distance of the custom video feed and the frames predicted by the autoencoder. The primary purpose of convolution in case of a convolutional network is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of inputdata. Mathematically, convolution operation performs dot products between theFigure 2: Our proposed network architecture. It takes a sequence of length T as input, and output a reconstruction of

the input sequence. The numbers at the rightmost denote the output size of each layer. The spatial encoder takes one frame at a time as input, after which T = 10 frames have been processed, the encoded features of 10 frames are concatenated and fed into temporal encoder for motion encoding. The decoders mirror the encoders to reconstruct the video volume.

Code snippet for encoder creation :

```
stae_model=Sequential()

stae_model.add(Conv3D(filters=128,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',input_shape=(227,227,10,1),activation='tanh'))

stae_model.add(Conv3D(filters=64,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'))

stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,padding='same',dropout=0.4,recurrent_dropout=0.3,return_sequences=True))

stae_model.add(ConvLSTM2D(filters=32,kernel_size=(3,3),strides=1,padding='same',dropout=0.3,return_sequences=True))

stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,return_sequences=True,
padding='same',dropout=0.5))

stae_model.add(Conv3DTranspose(filters=128,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'))

stae_model.add(Conv3DTranspose(filters=1,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',activation='tanh'))

stae_model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
```

4. METHODOLOGY

The Methodology of our proposed system "Survelliance Automation through deep learning " involves following steps:

Step 1: Extraction of Frames

Initialize directory path variable and describe a function to process and store video frames. Extract frames from video and call store function.it is used to change the input image to numeric data arrays are used to process on computer.each of these frame is extracted and processed separately.store the image list in a numpy file.

Step 2: Create spatial autoencoder architecture:

The spatial Auto encoder was desingned and analyzed the encoder model with the stored frames and created a saved model.which is further used for analysis of the abnormal events.

Figure-3:Creation of Autoencoder

Step 3:Trained model: We train our model on five most commonly used benchmarking datasets: Avenue , UCSD Ped1 and Ped2, Subway entrance and exit datasets.All videos are taken from a fixed position for each dataset. All training videos contain only normal events. Testing videos have both normal and abnormal events.In Avenue dataset, there are total 16 training and 21 testing video clips.Each clips duration vary between less than a minute to two minutes long. The normal scenes consist of people walking between staircase and subway entrance, whereas the abnormal events are people running, walking in opposite direction, loitering and etc. The challenges of this dataset include camera shakes and a few outliers in the training data. Also, some normal pattern seldom appears in the training data. UCSD Ped1 dataset has 34 training and 36 testing video clips, where each clip contains 200 frames. The videos consist of groups of people walking towards and away from the camera. UCSD Ped2 dataset has 16 training and 12 testing video clips, where the number of frames of each clip varies. The videos consist of walking pedestrians parallel to the camera plane. Anomalies of the two datasets include bikers, skaters, carts, wheelchairs and people walking in the grass area. Subway entrance dataset is 1 hour 36 minutes long with 66 unusual events of five different types: walking in the wrong direction (WD), no payment (NP), loitering (LT), irregular interactions between people (II), and miscellaneous (e.g.sudden stop, running fast). First 20 minutes of the video is used for training. Subway exit dataset is 43 minutes long with 19 unusual events of three types: walking in the wrong direction (WD), loitering (LT), and miscellaneous (e.g. sudden stop, looking around, a janitor cleaning the wall, gets off the train and gets on the train again quickly. First 5 minutes of the video is used for training.

Step 4:Testing phase: Now make another python test file and observe the results of abnormal event detection on any custom video. We identify the abnormal events based on the euclidean distance of the custom video feed and the frames predicted by the autoencoder.

THRESHOLDING:It is straightforward to determine whether a video frame is normal or anomalous.The reconstruction error of each frame determines whether the frame is

classifiedas anomalous. The threshold determines how sensitive we wish the detection system to behave — for example, setting a low threshold makes the system become sensitive to the happenings in the scene, where more alarms would be triggered. We obtain the true positive and false positive rate by setting at different error threshold in order to calculate the area under the receiver operating characteristic (ROC) curve (AUC). The equal error rate (EER) is obtained when false positive rate equals to the false negation.

We set a threshold value for abnormal events. In this project, it is 0.0068.As shown in figure-3, if it is less than maintained threshold.it considered as an Abnormal event.

```
    if loss>0.00068:
        print('Abnormal Event Detected')
        cv2.putText(image,"Abnormal Event",
(100,80),cv2.FONT_HERSHEY_SIMPLEX,2,(0,0,255),4)
```

Figure-4-code of threshold value

5.RESULT and DISCUSSION :

The result in our proposed system are implemented by using the OpenCv tool.We use keras.layers to import Conv3D,ConvLSTM2D,Conv3DTranspose.in which the 3D convolution model works on it.At first we have to train our encoder based on the given input type .As shown in figure a,b.where in some places waking on a wrong way is considered as a Abnormal event.Where as some parks carrying vehicles like cycles is Treated as Anamoly.where im places like museums runnings are considered as Anamoly.

Fig-a:Abnormal object

Fig-b:Wrong Direction

Here we already trained our model with the given data sets of Abnormal events.which will be stored in the form of frames As shown in the above figures.they are converted into gray scale.where we do geometroic transformation to improve mage quality.and after completion it will go for prediction on the image dump.by using the threshold value it will detect the bnormality in between both the cases.and display it through cv2.imshow.

6. EXECUTION

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from keras.preprocessing.image import img_to_array,load_img
import numpy as np
import glob
import os
import cv2
from keras.layers import Conv3D,ConvLSTM2D,Conv3DTranspose
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping
import imutils
```

```
store_image=[]
train_path='/content/drive/MyDrive/Datasets/train/training_videos'
fps=5
train_videos=os.listdir( train_path )
train_images_path=train_path+'/frames'
os.makedirs(train_images_path,exist_ok=True)
def store_inarray(image_path):
```

```
def store_inarray(image_path):
    image=load_img(image_path)
    image=img_to_array(image)
    image=cv2.resize(image,(227,227),interpolation = cv2.INTER_AREA)
    gray=0.2989*image[:,:,0]+0.5870*image[:,:,1]+0.1140*image[:,:,2]
    store_image.append(gray)
```

```
for video in train_videos:
    os.system( 'ffmpeg -i {}/{} -r 1/{}  {}/frames/%03d.jpg'.format(train_path,video,fps,train_path))
images=os.listdir(train_images_path)
for image in images:
    image_path=train_images_path + '/' + image
    store_inarray(image_path)
```

```
store_image=np.array(store_image)
a,b,c=store_image.shape
store_image.resize(b,c,a)
store_image=(store_image-store_image.mean())/(store_image.std())
store_image=np.clip(store_image,0,1)
np.save('training.npy',store_image)
```

```
callback_save = ModelCheckpoint("saved_model.h5", monitor="mean_squared_error", save_best_only=True)

callback_early_stopping = EarlyStopping(monitor='val_loss', patience=3)

stae_model.fit(training_data,target_data, batch_size=batch_size, epochs=epochs, callbacks = [callback_save,callback_early_stopping])
stae_model.save("saved_model.h5")
```

```
Epoch 1/5
23/23 [==============================] - 18s 350ms/step - loss: 0.2533 - accuracy: 0.5287
WARNING:tensorflow:Can save best model only with mean_squared_error available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 2/5
23/23 [==============================] - 8s 344ms/step - loss: 0.2049 - accuracy: 0.5446
WARNING:tensorflow:Can save best model only with mean_squared_error available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 3/5
23/23 [==============================] - 8s 346ms/step - loss: 0.2000 - accuracy: 0.5477
WARNING:tensorflow:Can save best model only with mean_squared_error available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 4/5
23/23 [==============================] - 8s 345ms/step - loss: 0.1392 - accuracy: 0.6417
WARNING:tensorflow:Can save best model only with mean_squared_error available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 5/5
23/23 [==============================] - 8s 348ms/step - loss: 0.1024 - accuracy: 0.6961
WARNING:tensorflow:Can save best model only with mean_squared_error available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```

```
stae_model=Sequential()

stae_model.add(Conv3D(filters=128,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',input_shape=(227,227,10,1),activation='tanh'))
stae_model.add(Conv3D(filters=64,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'))
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,padding='same',dropout=0.4,recurrent_dropout=0.3,return_sequences=True))
stae_model.add(ConvLSTM2D(filters=32,kernel_size=(3,3),strides=1,padding='same',dropout=0.3,return_sequences=True))
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,return_sequences=True, padding='same',dropout=0.5))
stae_model.add(Conv3DTranspose(filters=128,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'))
stae_model.add(Conv3DTranspose(filters=1,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',activation='tanh'))

stae_model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
```

```
training_data=np.load('training.npy')
frames=training_data.shape[2]
frames=frames-frames%10

training_data=training_data[:,:,:frames]
training_data=training_data.reshape(-1,227,227,10)
training_data=np.expand_dims(training_data,axis=4)
target_data=training_data.copy()

epochs=5
batch_size=1

callback_save = ModelCheckpoint("saved_model.h5", monitor="mean_squared_error", save_best_only=True)
```

```
import cv2
import numpy as np
from keras.models import load_model
import argparse
from PIL import Image
import imutils


def mean_squared_loss(x1,x2):
    difference=x1-x2
    a,b,c,d,e=difference.shape
    n_samples=a*b*c*d*e
    sq_difference=difference**2
    Sum=sq_difference.sum()
    distance=np.sqrt(Sum)
    mean_distance=distance/n_samples

    return mean_distance


model=load_model("saved_model.h5")
```

```
model=load_model("saved_model.h5")

cap = cv2.VideoCapture("/content/drive/MyDrive/testing_videos/19.avi")
print(cap.isOpened())

while cap.isOpened():
    imagedump=[]
    ret,frame=cap.read()

    for i in range(10):
        ret,frame=cap.read()
        image = imutils.resize(frame,width=700,height=600)

        frame=cv2.resize(frame, (227,227), interpolation = cv2.INTER_AREA)
        gray=0.2989*frame[:,:,0]+0.5870*frame[:,:,1]+0.1140*frame[:,:,2]
        gray=(gray-gray.mean())/gray.std()
        gray=np.clip(gray,0,1)
        imagedump.append(gray)

    imagedump=np.array(imagedump)

    imagedump.resize(227,227,10)
    imagedump=np.expand_dims(imagedump,axis=0)
    imagedump=np.expand_dims(imagedump,axis=4)

    output=model.predict(imagedump)

    loss=mean_squared_loss(imagedump,output)
```

```
        gray=np.clip(gray,0,1)
        imagedump.append(gray)

    imagedump=np.array(imagedump)

    imagedump.resize(227,227,10)
    imagedump=np.expand_dims(imagedump,axis=0)
    imagedump=np.expand_dims(imagedump,axis=4)

    output=model.predict(imagedump)

    loss=mean_squared_loss(imagedump,output)

    if frame.any()==None:
        print("none")

    if cv2.waitKey(10) & 0xFF==ord('q'):
        break
    if loss>0.00068:
        print('Abnormal Event Detected')
        cv2.putText(image,"Abnormal Event",(100,80),cv2.FONT_HERSHEY_SIMPLEX,2,(0,0,255),4)

    cv2.imshow("video",image)

cap.release()
cv2.destroyAllWindows()
```

Now, by running this script and observe the results of video surveillance, it will highlight the abnormal events.

7.OUTPUT

In normal it is not showing any message like in first case.in second case it will show an error message or warning like

abnormal event which helps to prevent the abnormal events easily.

8. CONCLUSIONS:

In this deep learning project, we train an autoencoder for abnormal event detection. We train the autoencoder on normal videos. We identify the abnormal events based on the euclidean distance of the custom video feed and the frames predicted by the autoencoder.We set a threshold value for abnormal events. In this project, it is 0.0068; we can vary this threshold to experiment getting better results. By incorporating convolutional feature extractor in both spatial and temporal space into the encoding-decoding structure, we build an end-to-end trainable model for video anomaly detection. The advantage of our model is that it is semi-supervised – the only ingredient required is a long video segment containing only normal events in a fixed view. Despite the models ability todetect abnormal events and its robustness to noise, depending on the activitycomplexity in the scene, it may produce more false alarms compared to other methods. For future work, we will investigate how to improve the result of video anomaly detection by active learning – having human feedback to update the learned model for better detection and reduced false alarms.

ACKNOWLEDGEMENT:

We authors wants to send sincere thanks to Principal, HOD and all CSE staff members Department, & our guide for them high support, motivation, and encouragement at all respect for publishing this paper in connect to our B.Tech Project work

REFERENCES

[1] Medel, J.R.: Anomaly Detection Using Predictive Convolutional Long Short-Term Memory Units. Master's thesis, Rochester Institute of Technology (2016), accessed from http://scholarworks.rit.edu/theses/9319

[2] Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009. pp. 935–942 (2009)

[3] Mo, X., Monga, V., Bala, R., Fan, Z.: Adaptive sparse representations for video anomaly detection. IEEE Transactions on Circuits and Systems for Video Technology 24(4), 631–645 (2014)

[4] Oneata, D., Verbeek, J., Schmid, C.: Action and Event Recognition with Fisher Vectors on a Compact Feature Set. 2013 IEEE International Conference on Computer Vision pp. 1817–1824 (2013), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751336

[5] Patraucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. International Conference On Learning Representations (2015), 1–10 (2016), http://arxiv.org/abs/1511.06309

[6] Armitage, R., G. Smyth, and K. Pease (1999). "Burnley CCTV Evaluation." In N. Tilley (ed.), Surveillance of Public Space: CCTV, Street Lighting and Crime Prevention, Vol. 10. Monsey, NY: Criminal Justice Press. [Full text]

[7] Bennett, T., and L. Gelsthorpe (1996). "Public Attitudes Towards CCTV in Public Places." Studies on Crime and Crime Prevention 5(1):72-90.

[8] Cavoukian, A. (2001). Guidelines for Using Video Surveillance Cameras in Public Places. Toronto, Canada: Information and Privacy Commissioner.[Full Text]

[9] Davies, S. G. (1996). "The Case Against: CCTV Should Not be Introduced." International Journal of Risk, Security and Crime Prevention 1(4):327-331

[10] Adam, A., Rivlin, E., Shimshoni, I., Reinitz, D.: Robust real-time unusualevent detection using multiple fixed-location monitors. IEEE Transactionson Pattern Analysis and Machine Intelligence 30(3), 555–560 (2008)

[11] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems. pp. 568–576 (2014)

[12] Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. ImageNet Challenge pp. 1–10 (2014), http://arxiv.org/abs/1409.1556

[13] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 4489–4497 (Dec 2015)

[14] Vu, T.H., Osokin, A., Laptev, I.: Context-aware cnns for person head detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2893–2901 (2015).

BIOGRAPHIES :

Kada Srisai: (317132910011) Final year B.Tech, Department of Computer Science & Engineering, Sanketika vidya parishad of engineering college, affiliated to Andhra University, Visakhapatnam, India.

Pyla Vamsi: (317132910021) Final year B.Tech, Department of Computer Science & Engineering, Sanketika vidya parishad of engineering college, affiliated to Andhra University, Visakhapatnam, India

MaddukuriMonika: (316177110087) Final year B.Tech, Department of Computer Science & Engineering, Sanketika vidya parishad of engineering college, affiliated to Andhra University, Visakhapatnam, India.

GUIDED BY:

Mrs.G.Vijayalakshmi: Associate Professor in Department of Computer Science & Engineering, Sanketika vidya parishad of engineering college, affiliated to Andhra University, Visakhapatnam, India.