Human Abnormal Behavior Detection Using Convolution Neural Network

Md. Ashiqussalehin
Department of Computer Science and Engineering (CSE)
Green University of *Bangladesh,*
Dhaka, Bangladesh
Dr. Khan NasrinJahan
Department of Community Medicine
Popular Medical College, Dhaka, Bangladesh
Md. Atikqur Rahaman
Faculty of Science Technology Engineering and Mathematics
International University of Malaya-Wales
Kuala Lumpur, Malaysia
Md. Shahidul Salim
Department of Computer Science and Engineering (CSE)
Khulna University of Engineering & Technology(KUET)
Khulna, Bangladesh

Abstract-
When a behavior is regarded as unusual in particular circumstances, it is termed abnormal. The term "abnormal behavior" has many definitions depending on the situation. People sprinting in a field, for example, is regarded natural, but it appears weird if it occurs in a mall. Similarly, loitering in alleyways, arguing, or pushing each other in public places are all regarded weird in certain situations. Deep learning has been widely used in the computer vision sector in recent years, with considerable success in human activity recognition. The proposed technique helps in the detection of abnormal behavior under a variety of contexts, including background changes, the number of subjects (one, two people, or a crowd), and a variety of unusual human behaviors.
Keywords— Foreground Extraction; Shutil Library; Image Net; Action Recognition.

Introduction

In a smart surveillance system, the capacity to rapidly detect unwanted activities in video surveillance systems is crucial. First, the sort of anomalous activity is determined by the monitoring scenario's requirements. The Fourth Industrial Revolution, which was announced on January 20, 2016, at the World Economic Forum in Davos, Switzerland, anticipated that the combination of artificial intelligence (AI) technology and many industrial domains would result in profound changes in a variety of fields. These changes are the result of contemporary AI-related technologies such as computer vision, robotics, and machine learning becoming exponentially more advanced. Next-generation technologies based on this trend include omnipresent, mobile super-

computing, intelligent robotics, and self-driving cars, all of which have the potential to drastically alter people's lives. At the current state of computer vision technology, many forms of picture data can be gathered using a variety of vision sensors, such as a stationary camera or a stereo camera, making the acquired scene more convenient.

Surveillance recordings from the cameras are first pre-processed to separate the foreground and background. The Mixture of Gaussian method [10] is used to extract foreground pixels. Following that, morphological opening operators [9] eliminate isolated foreground and noise pixels. The intense optical flow field from the video is used to extract features of object movements. The optical flow method is used because predicting motion only requires two consecutive frames, allowing new information to be recovered from each frame. With the use of an image pyramid, larger flow vectors may be approximated more precisely. Figure 2 shows the flow vectors extracted from the videos.

One of the most important aspects of video surveillance is target detection. Extracting the target's properties, target recognition, and target positioning are all steps in the classic target detection method. It is applied to different Pedestrian and mission detection tasks in the background, such as the SIFT technique. Deep learning-based target recognition and detection algorithms have become the standard in recent years [11]. The deep neural network-based target identification framework is separated into two kinds [12]. The two-stage target detection framework is one form, whereas the single-stage target detection framework is another, combining the classification and regression tasks into one phase.

Related Works

Deep learning. Recently, machine learning studies have developed various algorithms based on deep learning that exhibit remarkable capabilities in computer vision tasks, as for example convolutional neural networks (CNNs), which achieved the best accuracy in the object recognition and detection task with large-scale image databases in the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2012 (Everingham et al., 2015) and the PASCAL VOC (Visual Object Classes) challenge (Everingham et al., 2015). The original CNN architecture proposed by Krizhevsky et al. is called AlexNet (Krizhevsky et al., 2012), and its top-5 error rate was 15.3%, which exceeded the result of the second-place team by more than 40%. The algorithm outperformed previous approaches such as support vector machine (SVM) and traditional pattern classification algorithms. Furthermore, the development of support hardware, such as a graphics processing unit (GPU), along with the emergence of the concept of big data, makes it possible to learn and evaluate a large-scale deep learning architecture in a short time, and various attempts have focused on this. High-performance GPU processing (such as with CUDA) and large public image repositories (such as ImageNet (Deng et al., 2009)) have enabled CNNs to become the most popular method in the computer vision field. Several attempts have focused on enhancing the performance of AlexNet to achieve improved accuracy in image- and video-based object recognition tasks. The best-

performing submissions to the ILSVRC were ZF-Net (Zeiler and Fergus, 2014) in 2013, GoogLeNet (Szegedy et al., 2015) in 2014, and ResNet (He et al., 2016) in 2015.

Dataset Description:

We employed the UT-Interaction dataset in our study, which contains videos of continuous executions of six different types of human-human interactions: shake-hands, point, embrace, push, kick, and punch. Time intervals and bounding boxes are provided as ground truth labels for these interactions. There are a total of 20 video clips, each lasting around one minute. Each video comprises at least one execution per interaction, resulting in an average of 8 human activity executions per video. The videos feature a variety of participants dressed in over 15 distinct outfits. The videos were shot at a resolution of 720*480 at 30 frames per second, with a person's height in the video being around 200 pixels.

We divide the videos into two groups. The first series consists of ten video clips shot in a parking lot. Set 1's videos were shot at a slightly different zoom rate, with primarily static surroundings and minimal camera jitter. Set 2 (the last 10 sequences) was shot on a stormy day on a lawn. They have more camera jitters, and the background moves slightly (for example, a tree moves). Only two interacting people appear in the scene from sequences 1 to 4 and 11 to 13. Both interacting people and pedestrians are featured in the scenario from sequences 5 to 8 and 14 to 17. Several pairs of interacting people conduct the exercises simultaneously in sets 9, 10, 18, 19, and 20. The background, scale, and illumination are all different in each set.

Figure : Types of Activities in the Interaction Challenge

Dataset Preprocessing:

First, we divided the video into different segments. Each segment represented one activity. Here, we used python language to find the appropriate frame from the videos. Each video had several activities. We needed to find the proper frame duration to retain the right frame, which defines an activity.

First, we needed to read the excel file where the frame starting point and ending point were located.

```
import pandas as pd
df = pd.read_csv (r'C:\Users\Inception\Desktop\data\ut-interaction_labels_110912_2.csv')
sequence='seq20'
```

We import the pandas' library. We used pandas as a CSV reader. ‘pd.read_csv’ file reads the file where it was located.

We took a video from the dataset and read every video, and every video was divided by frame.

We read every frame and classify every frame to its corresponding activities.

```
import cv2
vidcap = cv2.VideoCapture(r'C:\Users\Inception\Downloads\Compressed\ut-interaction_set2/'+sequence +'.avi')
success,image = vidcap.read()
count = 0
while success:
    val=int(call1(count))
    #print(val)
    if int(val)!=-1:
        st="C:\\Users\\Inception\\Desktop\\data\\" + str(val) +"\\"+sequence+"_frame"+str(count)+".jpg"
        print(st)
        cv2.imwrite(st, image)
    success,image = vidcap.read()
    print('Read a new frame: ', success,count)
    count += 1
```

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it integrated with various libraries, such as NumPy, python can process the OpenCV array structure for analysis. To Identify image patterns and its various features we use vector space and perform mathematical operations on these features.

```
def call(frame):
    frame=int(frame)
    for x in range(len(df['sequence #'])):
        print(frame," ",df['Starting Frame #'][x]," ",df['Ending Frame #'][x])
        if frame>=int(df['Starting Frame #'][x]) and frame<=int(df['Ending Frame #'][x]) and str(df['sequence #'][x])==sequence:
            return df['Activity Class ID'][x]
    return -1
```

We import cv2, the OpenCV library. 'cv2.videocapture' reads the frame from the video. We iterated all the frames, and then we selected the appropriate frame from the video using the below code.

Methodology

Then we needed to divide the data into two folders. One is a train folder, and the other is a test folder. Here, we used the supervised learning method. So, we need to level the data according to its activities.

We created a folder where we put all the files according to their activities.

```
import os
import numpy as np
import shutil
import random
root_dir = 'C:\\Users\\Inception\\Desktop\\dest\\'
classes_dir = ['0', '1','2','3','4','5']

test_ratio = 0.20

for cls in classes_dir:
    os.makedirs(root_dir +'train/' + cls)
    os.makedirs(root_dir +'test/' + cls)
```

Here we used the “shutil” library to copy the data from the folder. Shutil module in Python provides many functions of high-level operations on files and collections of files, and it comes under Python’s standard utility modules. This module helps automate the process of copying and removing files and directories. We randomize the file so that it holds all the combination data. Then, we placed the file according to its activity.

```
def call2(filen):
    src = 'C:\\Users\\Inception\\Desktop\\source\\'+filen

    allFileNames = os.listdir(src)
    np.random.shuffle(allFileNames)
    train_FileNames, test_FileNames = np.split(np.array(allFileNames),
                                                           [int(len(allFileNames)* (1 - test_ratio))])


    train_FileNames = [src+'/'+ name for name in train_FileNames.tolist()]
    test_FileNames = [src+'/' + name for name in test_FileNames.tolist()]

    print("****************************")
    print('Total images: ', len(allFileNames))
    print('Training: ', len(train_FileNames))
    print('Testing: ', len(test_FileNames))
    print("****************************")



    for name in train_FileNames:
        shutil.copy(name, root_dir +'train/'+filen+'/' )

    for name in test_FileNames:
        shutil.copy(name, root_dir +'test/'+filen+'/')
    print("Copying Done!")
```

```
for i in range(3,6):
    call2(str(i))
```

```
****************************
Total images:  2401
Training:  1920
Testing:  481
****************************
Copying Done!
****************************
Total images:  1392
Training:  1113
Testing:  279
****************************
Copying Done!
****************************
Total images:  1985
Training:  1588
Testing:  397
****************************
Copying Done!
```

Training

We need to preprocess the data for training and testing purpose. Here, we convert the image into target size (224*224).

```
trdata = ImageDataGenerator()
traindata = trdata.flow_from_directory(directory='../dataFile/train',target_size=(224,224))
```

```
Found 9760 images belonging to 6 classes.
```

```
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory='../dataFile/test', target_size=(224,224))
```

```
Found 2444 images belonging to 6 classes.
```

Here, we used transfer learning for better performance. We imported image net weights.ImageNet is a standard for images classification. A yearly contest is run with millions of training images in 1000 categories. The models used in the ImageNet classification competitions are measured against each other for performance. Therefore, it provides a "standard" measure for how good a model is for image classification. So many often-used transfer learning model models use the ImageNet weights. Transfer learning can be customized for your application by adding additional layers to the model. ImageNet weights is beneficial as it helps the model converge in less epochs.

```
from keras.applications.vgg16 import VGG16
vggmodel = VGG16(weights='imagenet', include_top=True)
vggmodel.summary()
```

In neural network , last layer defines the output of the model. As we had 6 classes. So we needed to set lastDense layer was 6.

```
predictions = Dense(6, activation="softmax")(X)
```

The model is created and compiled as follow.

```
model_final = Model(input = vggmodel.input, output = predictions)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: UserWarning: Update your `Model` call to the Keras 2 API: `Model(inputs=Tensor("in..., outputs=Tensor("de...)`
  """Entry point for launching an IPython kernel.
```

```
model_final.compile(loss = "categorical_crossentropy", optimizer = optimizers.SGD(lr=0.0001, momentum=0.9),
```

The model summery is presented bellow.

```
model_final.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 24582
=================================================================
Total params: 134,285,126
Trainable params: 134,285,126
Non-trainable params: 0
_________________________________________________________________
```

Result and Discussion

Here ,we used early stopping function so that if value accuracy did not improve 40 epochs. We will stop the training. We plotted the “Accuracy”, “Validation Accuracy", "loss" and "Validation Loss” and it looked as follow.

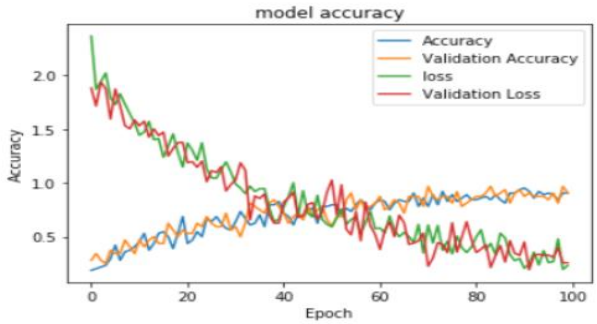

Figure: Accuracy, Validation and Loss function of the model.

Conclusion

This system will help to detect abnormal behavior more accurately and precisely in video surveillance systems than other proposed systems. As a result, any unusual public activities can detect and take proper steps to avoid tragedies. This system used a deep convolution framework to detect abnormal human behavior from a standard RGB image. A typical dataset has been used to train the model. This proposed model will predict abnormal activities and behavior from different contexts.

REFERENCES

1. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2015. The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. 111 (1), 98–136.
2. Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105.
3. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009.IEEE Conference on. IEEE, pp. 248–255.
4. Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. Springer, pp. 818–833
5. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015.Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9
6. Simonyan, K., Zisserman, A., 2014b. Very deep convolutional networks for large-scale image recognition, 2014b, aArXiv preprint arXiv:1409.1556
7. Ni, B., Wang, G., Moulin P RGBD, H., 2011. A colour-depth video database for human daily activity recognition. In: Proceedings of IEEE International Conference on ComputerVision Workshops, ICCV Workshops, November, pp. 6–13
8. Schmidt, A., 2000. Implicit human computer interaction through context. Pers. Technol. 4 (2), 191–199.

9. Shuiwang, J., Wei, X., Ming, Y., Kai, Y., 2013. 3D convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35 (1), 221–231. http://dx.doi.org/10.1109/TPAMI.2012.59.
10. Simonyan, K., Zisserman, A., 2014a. Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems. pp. 568–576.
11. Han, Y., Zhang, P., Zhuo, T., Huang, W., Zhang, Y., 2017. Going deeper with two-stream convnets for action recognition in video surveillance. Pattern Recognit.Lett.. http://dx.doi.org/10.1016/j.patrec.2017.08.015. URL http://www.sciencedirect.com/science/article/pii/S0167865517302702.
12. Girshick, R., 2015. Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448.
13. Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99.