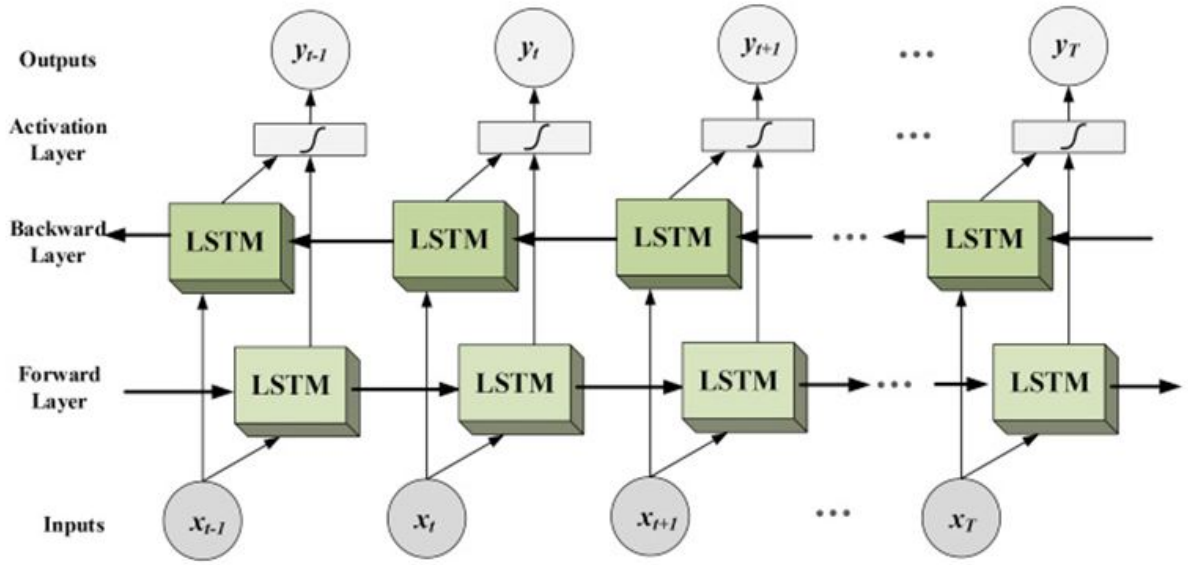
Report on Implementation of Violence Detection with CNN+BiLSTM model

Description of the method:

BI-LSTM (Bi-directional long short term memory):

Bidirectional long-short term memory (bi-lstm) is the technique of allowing any neural network to store sequence information in both ways, either backwards (future to past) or forwards (past to future). Our input runs in two ways in bidirectional, distinguishing a bi-lstm from a conventional LSTM. We can make input flow in only one way using a conventional LSTM, either backwards or forwards. However, using bi-directional, we can make the input flow in both ways, preserving both the future and the past.



Implementation:

To begin, I have divided a video into numerous frames. To extract the information/features in the current frame, I ran each frame through a convolutional neural network. Then, to recognize any sequential flow of events, as given in the paper I have utilized a Bidirectional LSTM layer to compare the information of the current frame once with the prior frames and once with the forthcoming frames. Lastly, the classifier determines whether or not an action is violent. As a result, for prediction analysis, this design employs both spatial and temporal data in both directions.

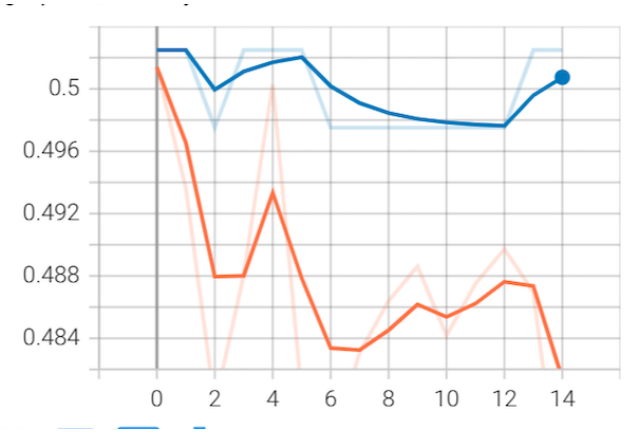
The model must be able to anticipate sequences in successive frames, such as a pattern in the movement of the individuals or the degree of their motion, in order to classify violent or non-violent activities. This is not feasible if just the spatial properties (features that relate to a

certain frame) of the frames are considered. When recognizing sequences in frames, temporal or time-related characteristics must also be taken into account. The temporal characteristics can be handled in either the forward or backward order. In addition to geographical features, the model analyzes temporal features in both directions as well.

For training and testing, the datasets are split in an 8:2 ratio. The complete model was built and trained from scratch over 25 epochs. The model was fed a set of 20 consecutive frames with 200 x 200 dimensions to extract the spatial and temporal properties. I have also used stochastic gradient descent as an optimizer with lr = 0.0001. In this paper, the loss function is "sparse categorical crossentropy," however I used categorical crossentropy.
Preprocess:

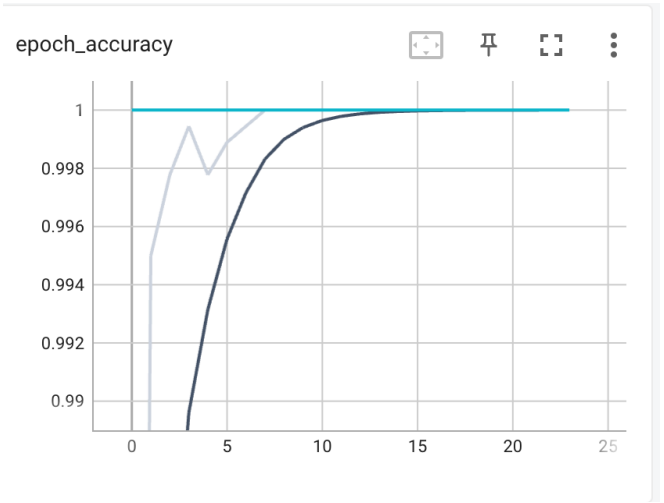
The videos' frames have been retrieved. The retrieved frames are reshaped to a size of 200 x 200 pixels. The training data consists of a Numpy3 array, with each row representing a video sequence or pattern, to extract the temporal characteristics, as suggested by the paper, I have utilized 20 consecutive frames.
Important adjustments made to the ones that are found in the paper:

1. The image dimensions that I have chosen are 200 x 200 x 3 where in the paper its 100 x 100 x 3
2. During training a group of 20 consecutive frames were passed where in the paper only a group of 10 were forwarded.
3. Extra convolution layer better extraction of features.
4. 3d max pooling instead of 2d that has been done in the paper.



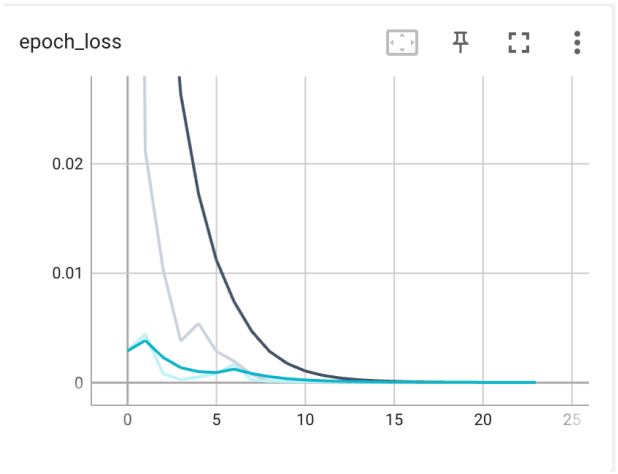
fig(1): These results are obtained by using the traditional CNN+LSTM network (Trained for 15 epochs). Blue color Indicates the validation accuracy and orange color indicates training accuracy.

These later are compared with the prepared model. Which are attached below:



fig(2): Blue indicates validation and Black indicates training.

This is the accuracy I have obtained using my model implementation. Given the changes that are present from the paper.



fig(3) : This is the loss plot. Blue indicates the testing loss and black is for training

Comparison of obtained results:

The network has been trained over 40,000 Images and the results of the implemented paper when compared with the existing methods on the same dataset:

Methods	**Hockey Fights**
MoSIFT+HIK	90.9%
ViF	82.9%
MoSIFT+KDE+Sparse Coding	95%
SELayer-3D CNN	99%

CNN+BiLSTM (Reimplemented)	100%

Code of Implementation:
My own implementation is represented in bold character apart from the refereed:

#implementation of CNN BiLSTM Model for violence detection:

```
import os
import cv2
import time
import numpy as np
from tqdm import tqdm
from random import shuffle
import tensorflow as tf
from tensorflow import keras
from keras import backend as K
from tensorflow.keras import Model, layers
from tensorflow.keras.callbacks import TensorBoard
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Bidirectional,
TimeDistributed, Reshape
from tensorflow.keras.layers import  Conv2D, Dense, Flatten,
MaxPooling2D
from tensorflow.keras.layers import Dense,Dropout, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Input

#Directories of the raw dataset (videos corresponding to the
data)
hviolence = 'HockeyFightsvideos/violence/'
hnoviolence = 'HockeyFightsvideos/noviolence/'

frames = 20 #NUmber of frames given in single pass to the
network
img_shape = 200

#---------Network---------------
#CNN
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu',
input_shape=(20, img_shape, img_shape, 3), padding="same"))
```

```
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling3D((1, 2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling3D((1, 2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling3D((1, 2,2)))
model.add(Reshape((20, 10816)))
#BiLSTM
lstmF = LSTM(units=32)
lstmB = LSTM(units=32, go_backwards = True)
model.add(Bidirectional(lstmF, backward_layer = lstmB))
#Dense
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
#model.summary()
#------------------------------
#this function will join the extracted frames into the data
frame directory
def classifytodfs(violence, noviolence, path):
    count = 0
    for files in os.listdir(violence):
        cap = cv2.VideoCapture(os.path.join(violence, files))
        sucess, image = cap.read()
        sucess = True
        while sucess:
            sucess, image = cap.read()
            if not sucess:
                break
            cv2.imwrite(path+str(count)+".jpg",image)
            if cv2.waitKey(10) == 27:
                break
            count += 1
    for files in os.listdir(noviolence):
        cap = cv2.VideoCapture(os.path.join(noviolence,
files))
        sucess, image = cap.read()
        sucess = True
        while sucess:
            sucess, image = cap.read()
            if not sucess:
                break
            cv2.imwrite(path+str(count)+".jpg",image)
            if cv2.waitKey(10) == 27:
```

```
                break
            count += 1

classifytodfs(hviolence, hnoviolence, path =
"Data/Hfights/DF/")

def dataframe(folder):
    dataset = []
    image = []
    limit = 0
    c = 0

    for file in tqdm(os.listdir(folder)):
        path = os.path.join(folder, file)
        img = cv2.resize(cv2.imread(folder), (200, 200))
        image.append(np.array(img))
        limit += 1
        c += 1
        if c == frames:
            c = 0
            if limit < 20056:
                dataset.append([image, np.array([1, 0])])
            elif limit >= 20056:
                dataset.append([image, np.array([0, 1])])
            image = []

    shuffle(dataset)
    np.save('dataset.npy', dataset)
    print(dataset)
    return dataset


df = dataframe(folder='Data/Hfights/DF')

data = np.load('dataset.npy', allow_pickle=True)
train, test = train_test_split(data, train_size = 0.9)
X = np.array([i[0] for i in train]).reshape(-1, 20, img_shape,
img_shape, 3)
Y = np.array([i[1] for i in train])
x_valid = np.array([i[0] for i in test]).reshape(-1, 20,
img_shape, img_shape, 3)
y_valid = np.array([i[1] for i in test])
X = X.astype('float32')/255
x_valid = x_valid.astype('float32')/255
```

```
model.compile(loss='categorical_crossentropy',
optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
metrics = ['accuracy'])
Tensorboard = TensorBoard(log_dir='logs/{}'.format('Model
{}'.format(int(time.time()))))
model.fit( X,Y, epochs=15, validation_data=(x_valid,
y_valid),batch_size=16, verbose=1, callbacks=[Tensorboard])
model.save('CNN-BiLSTM200.h5', overwrite=True,
include_optimizer=True)
#Testing:
model = load_model('CNN-BiLSTM200.h5')
data = dataframe(folder='rawdata/dataframes')
data = np.load('dataset.npy',allow_pickle=True)
X = np.array(i[0] for i in data).reshape(-1, 20, img_shape,
image_shape,3)
Y = np.array(i[1] for i in data)
X = X.astype('float32')/255
model.evaluate(np.array(X),np.array(Y), batch_size=16,
verbose=1)
```

Bibliography:

CNN-BiLSTM Model for Violence Detection in Smart Surveillance
Rohit Halder & Rajdeep Chatterjee

A CNN-LSTM Approach to Human Activity Recognition Ronald Mutegeki School of Computer Science and Engineering Kyungpook National University Daegu, by Dong Seog Han
* School of Electronics Engineering Kyungpook National University Daegu, Korea dshan@knu.ac.kr

Violence Detection in Surveillance Videos with Deep Network Using Transfer Learning
December 2018 DOI:10.1109/EECS.2018.00109 Conference: 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS)