Accident and Violation Detection

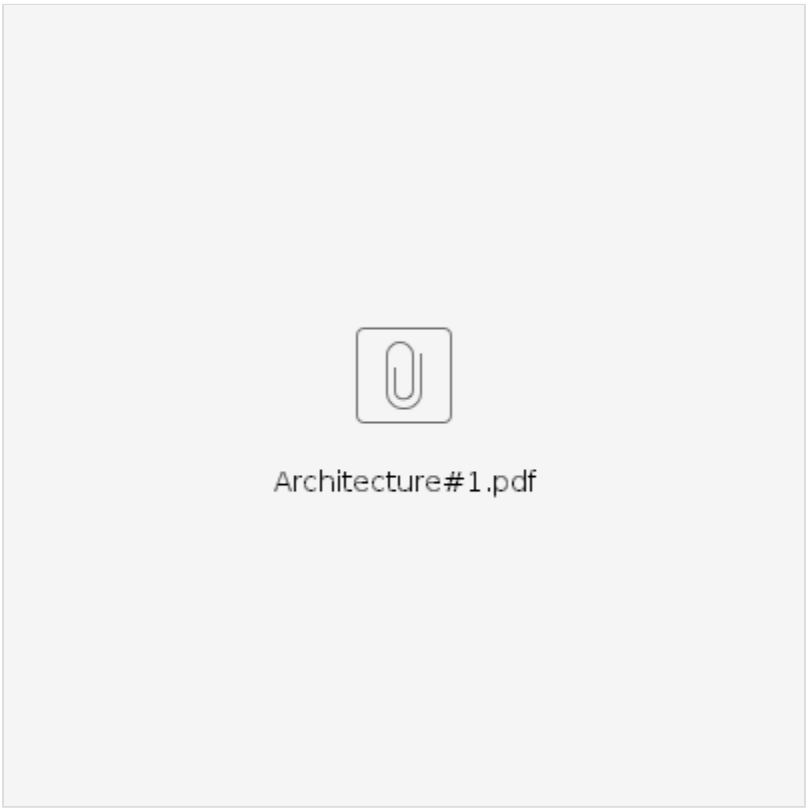
I. Architectures Looked at and possible Methods:

1. 3D CNN's - The 3D CNN will be used to learn temporal features across time using fixed chunks of video frames as inputs. These chunks must be classified as accident or non accident.

2. Long term recurrent convolutional networks - The LRCN model could be used as a substitute to the 3D CNN as it is more recent and claims better performance that the 3D CNN on the same task.

3. Neural Turing Machines - The NTMs could be used to classify the input videos as accidents or non accidents using very little training data as they are built for such purposes.

Method to be tried out first:



1. LRCN + NTM + Siamese network :
The NTM and LRCN will be first trained to classify accident and non accident videos. The outputs of these two networks are used as inputs to the Siamese network. A Siamese network in general is used to check the similarity between two inputs; the idea of using a siamese network is to check the difference between two inputs, assuming one input to be a standard non-accident video chunk and the other to be our testing sample, if they are very dissimilar (or similar) - the siamese network will tell us the the same. By doing this over a large number of samples of non accident videos paired with accident videos, we could have a total of NxM training samples where N would be the number of non accident samples and M would be the number of training samples.
The LRCN is trained with a mixture of accident and non accident data. Naturally, it would be biased towards non accidents as they are more in number.
The NTM will also be trained with a mixture of accident and non accident data but there will be as much non-accident data as there is accident data to prevent it from being biased to either of the datasets.
Since the NTM does not achieve state of the art performance but classifies to a reasonable extent, and also, in our case the LRCN cannot properly classify due to too many false positives, there must be some way of utilizing the strength of both to get a good classifier. This improvement in classification is brought about by the siamese network.
Each of the two networks in the siamese network are connected to the FC layers of the LRCN and NTM.
Once the LRCN and NTM have been trained, the descriptors that we get from their FC layers are fed into the siamese network inputs. For an LRCN input as non accident and NTM input as accident, the siamese network must show them to be very dissimilar, and if both are given non accident, the siamese network should indicate high similarity => this would be the training paradigm. Since the training is being done end to end, the siamese network will force the LRCN and NTM to output desriptors that are similar or dissimilar based on the inputs. This way, the individual strengths of the networks are being exploited.

Alternative methods to be looked at:

1.Generation of Data: Use Reinforcement learning using openAI's universe to generate data (accidents) and refine those accidents using simGANs to incorporate realism.
2.Use the pixel wise segmentation to detect intersections of vehicles which chandan suggested. Need to think about how cars that have been joined can be detected and thus confirm accidents.

3.Think about how unsupervised LSTM descriptors can be used to apply NTM's to videos.

Comments:

- Add a detailed architecture of LRCN + NTM + Siamese Network
- The following traffic violations need to be considered:
    - Red signal jumping

      Lane indiscipline

      Pedestrian crossing violation (overstepping)

      Riding without helmet

      Wrong parking

      Taking "U"-turn when prohibited

      Driving against One-way traffic

      Lane indiscipline while taking a turn

      Driving on footpath/sidewalk

      Standing/parking in prohibited areas

      Over-speeding
- Will the same architecture identified for accident detection work for most scenarios of traffic violation? If not, what alternative networks can be considered? Create an architecture diagram for this.

ACCIDENT DETECTION AND LOCALIZATION

Soumith Kumar Dachepalli
Work done from 13th may to 12th july, 2019.

CADP dataset

The dataset had 1415 videos. All videos are positive cases for accidents. Many videos are of low quality and resolution which badly affects object detection. To achieve good accuracy for object detection in their low-resolution images, context mining is necessary. This adds overheads during both training and inference. As per the requirements, the quality of the videos that's expected doesn't require this. So, to skip this context mining step, videos of better quality and resolution are to be extracted for training.

Also, the original dataset doesn't provide annotations for the video clips. It provided only temporal annotations for the raw YouTube videos. Since original videos from YouTube cannot be shares, they've shared the URLs of these videos. This causes a lot of issues. Some of these videos are no longer available on YouTube. All of these are uncleaned videos. Many of them are compilations which will include clips from different scenes. So these annotations are unusable.

After filtering through all the 1415 videos, I've been able to extract videos of good quality and resolution. I've annotated these videos (temporal annotations).

Annotating guideline – An accident starts on the first contact of the objects involved in the accident and ends when the objects are separated and when all the objects are either stationary or moving naturally.

Extracted Dataset stats

Number of videos	-	186
Avg. Number of frames per video	-	340

Note that all of these are positive instances. Need to explicitly mine negative instances for train and test.

Accident Detection:

Approach used is the one mentioned in CADP paper (https://arxiv.org/abs/1809.05782). This builds on the approach in paper titled 'Anticipating Accidents in Dashcam Videos' available here. (https://yuxng.github.io/chan_accv16.pdf).

Abstract of the approach:

A classic method has been extracting features from an image and passing them onto an LSTM to detect an anomality. In the case of accidents, the paper suggests augmenting these full image features with the features of the objects in the image. The paper proposes an attention technique called Dynamic Spatial Attention for objects in a frame. In the case of accident detection, tracking objects in the image rather than only using whole scene related features will give better results.

Object Features:

In this case, these are extracted from last fully connected layer fc7 (1024D) of Faster RCNN (https://arxiv.org/abs/1506.01497) are used as object features.

Full Image Features:

Features from the base network of Faster RCNN (ResNet50 in this case) are extracted and used for this purpose. Features are extracted from avg pool (2048D) layer of ResNet50 pretrained on ImageNet.

For these features, a pretrained model of Detectron on coco dataset has been used. Detectron has an implementation of Faster RCNN. Implementation available here:

https://github.com/facebookresearch/Detectron

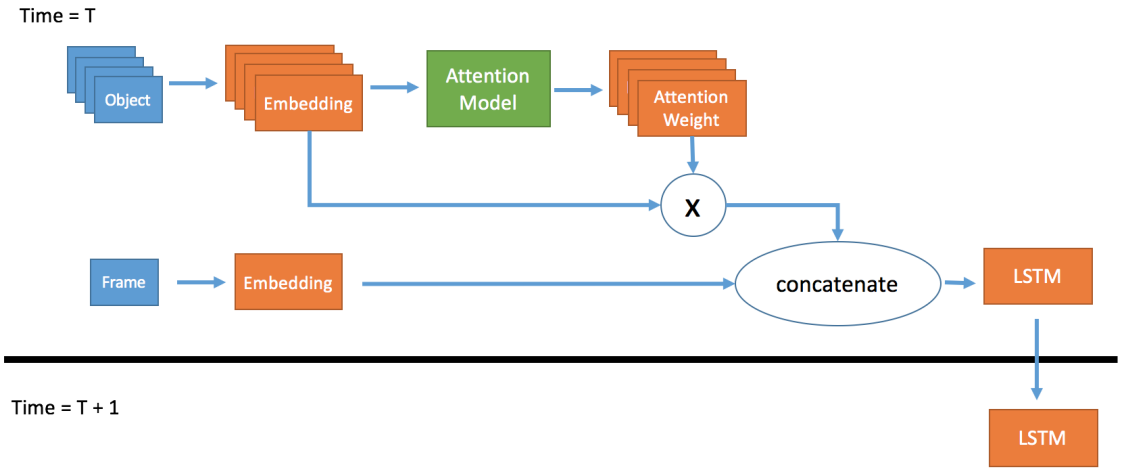
The implementation available here has been modified to extract required features from appropriate layers during inference. Features have been extracted for all the frames of the videos in extracted videos.

After extracting the features as mentioned in the previous step, the data is processed to contain clips of 100 frames each. The DSA LSTM mentioned earlier takes features of 100 frames and predicts probability of accident at each frame.

The aim is to detect if the accident had happened at 90^{th} frame i.e. the model looks at 90 frames before the accident and 10 frames after the accident to detect if the accident had happened. The implementation present in the below link has been used as a base and changes have been made to include image related features from ResNet.

https://github.com/smallcorgi/Anticipating-Accidents

It should be noted that CADP contains only positive videos.



Training :

The model has been trained with a sample of 130 clips each of which has 100 frames. In a positive clip, the accident starts at 90th frame.

Train Sample Size:

330 (Positive – 150, Negative – 180). The negative samples have been extensively mined from the parts of CADP videos before the accident starts. Also few samples from itm have been used.

Test Sample Size :

60 (Positive – 40, Negative – 20) . The negative samples are explicitly from the ctm videos.

Test average precision (AP) - 0.6735

The authors of the CADP paper report an average precision of 0.47. Also they were able to predict the accident 1.3 seconds before it happens on average. In this case the model was able to predict the accident as early as 3 seconds on average.

During visualization, it has been noticed that the probability of accident is spiking pretty early in the video, in most cases, even before the scene starts changing or the vehicles involved in the accident appear in the frame. This behavior is not expected since at each frame, the model only looks only at the previous frames to predict the accident and not at the future frames. This indicates that the model is overfitting with respect to scene related features. It should be noted that overall quality of the videos in CADP dataset is lower than itm videos and model might have been using this as a cue that the video is negative since all the positive cases come from CADP dataset.

So, a second round of training has been done which included only samples from CADP dataset. Negatives mined from videos before accidents took place have been used.

Train Sample Size :

270 (Positive – 140, Negative – 130).

Test Sample Size :

80 (Positive – 40, Negative – 40).

Note that all these samples are from CADP dataset. The test average precision has decreased to 0.48 which is more like the score reported in the paper.

Other references used:

Learning Deep Representations of Appearance and Motion for Anomalous Event Detection : https://arxiv.org/abs/1510.01553

Maxout Networks : https://arxiv.org/pdf/1302.4389

More Investigation Into Other Approaches

The model is overfitting and the performance is not satisfactory. So more models should be explored.

It is difficult to obtain accident videos. Discriminative models working with deep features tend to overfit in such conditions. Also, these can't be used for localizing the accident. Most of the data available is from other countries which doesn't reflect the Indian traffic conditions. Methods that are localized are necessary to filter through the dense traffic conditions of India. Also, the model should be robust to overfitting.

One possible method is to track the vehicles to detect collisions by tracing their trajectories. Here, there is a problem of collision vs occlusion. Below paper, uses object tracking using deepsort, to detect accidents.

https://arxiv.org/abs/1901.01138 (Intelligent Intersection: Two-Stream Convolutional Networks for Real-time Near Accident Detection in Traffic Video)

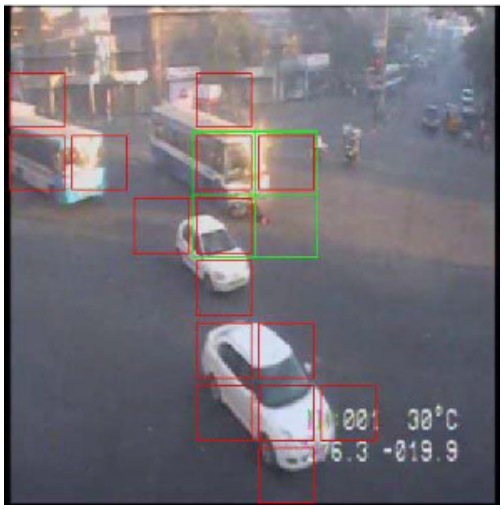
Here, videos from fisheye view are used. This essentially eliminated trajectory intersection due to occlusion. But, in the case of CCTV videos, differentiation between these two cases is needed.

Below paper addresses this problem.

https://ieeexplore.ieee.org/document/8367975 (Deep Spatio-Temporal Representation for Detection of Road Accidents Using Stacked Autoencoder)

This paper treats the video frame as a grid. At each grid point, if there's movement, it encodes a spatio – temporal sequence of frame and does anomality detection to check for accident.

Some samples images from the paper.

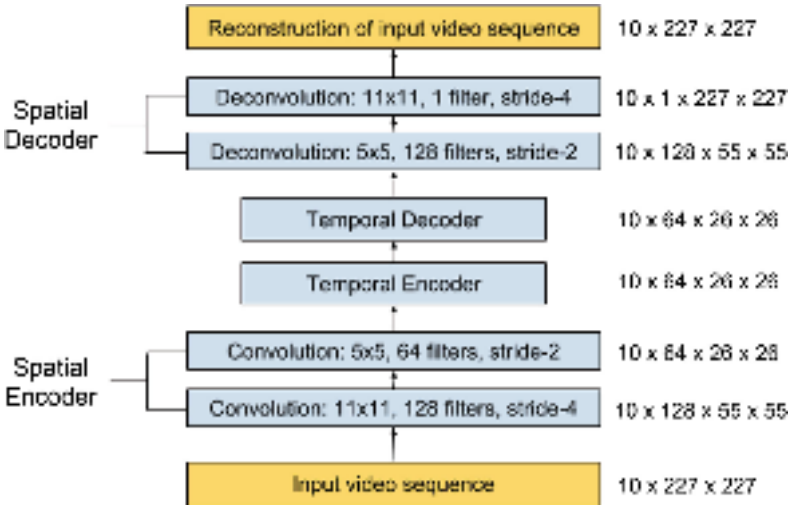


Autoencoder Approach

https://code.siemens.com/adm-vp/itm/accident-detection/tree/beta

Uses spatio-temporal sequence of frames and sequence of optical flow. Calculates anomality score by using reconstruction error and outlier detection. This addresses the problem of trajectory intersection due to occlusion by looking at pre-accident and post-accident scene features. This allows us to localize the accidents and also is robust to overfitting since it works with autoencoders.

The paper uses a denoising autoencoder proposed in the paper Learning Deep Representations of Appearance and Motion for Anomalous Event Detection . These autoencoders treat the image as a flat vector. For better utilization of spatial features of the image, conv-lstm has been used. The encoder architecture proposed in the paper Abnormal Event Detection in Videos Using Spatiotemporal Autoencoder.



The architecture has been tweaked to suit the input shape in this case. Following is the summary of the architecture used.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 15, 50, 50, 3)	0
time_distributed_1 (TimeDist	(None, 15, 17, 17, 128)	18944
time_distributed_2 (TimeDist	(None, 15, 17, 17, 128)	512
time_distributed_3 (TimeDist	(None, 15, 17, 17, 128)	0
time_distributed_4 (TimeDist	(None, 15, 9, 9, 64)	204864
time_distributed_5 (TimeDist	(None, 15, 9, 9, 64)	256
time_distributed_6 (TimeDist	(None, 15, 9, 9, 64)	0
convlstm1 (ConvLSTM2D)	(None, 15, 9, 9, 64)	295168
convlstm2 (ConvLSTM2D)	(None, 15, 9, 9, 32)	110720
convlstm3 (ConvLSTM2D)	(None, 15, 9, 9, 64)	221440
time_distributed_7 (TimeDist	(None, 15, 17, 17, 128)	204928
time_distributed_8 (TimeDist	(None, 15, 17, 17, 128)	512
time_distributed_9 (TimeDist	(None, 15, 17, 17, 128)	0
time_distributed_10 (TimeDis	(None, 15, 50, 50, 3)	18819

Total params: 1,076,163
Trainable params: 1,075,523
Non-trainable params: 640

The input to the model is a Spatio Temporal Video Volume (STVV) of shape (frames x width x height x volume). For experiments, shapes (15, 50, 50, 3) and (15, 75, 75, 3) have been considered.

Experiments and Results:

The average loss between volumes with accidents and volumes without accidents are considered. A validation set of 80 volumes has been used for this purpose.

	Average loss Volumes with accident	Average loss Volumes without accident
50 x 50	1130	896
75 x 75	1017	788

For further training, 50 X 50 size has been chosen. Larger dataset has been used. Dataset contained 500 batches of STVVs from CADP dataset (frames that are mined before accident begins) and 250 batches of STVVs from itm dataset. Batch size is 32. Then total number of STTVs is around 24000 and training has been done for 15 epochs. (Note that these are STTVs with movement in them)

Loss Function - MSE

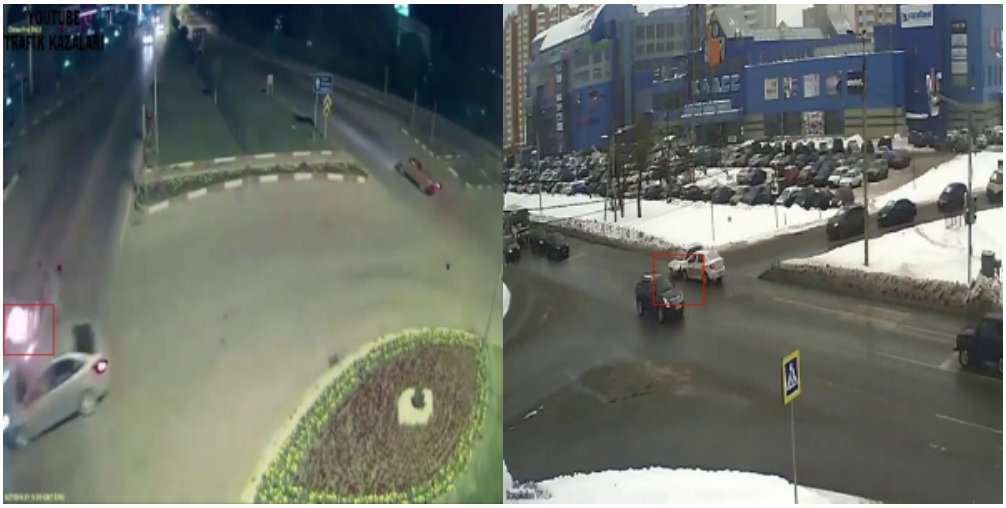
Optimizer - RMSProp with learning rate 0.001

loss: 376.9718 - val_loss: 507.6585

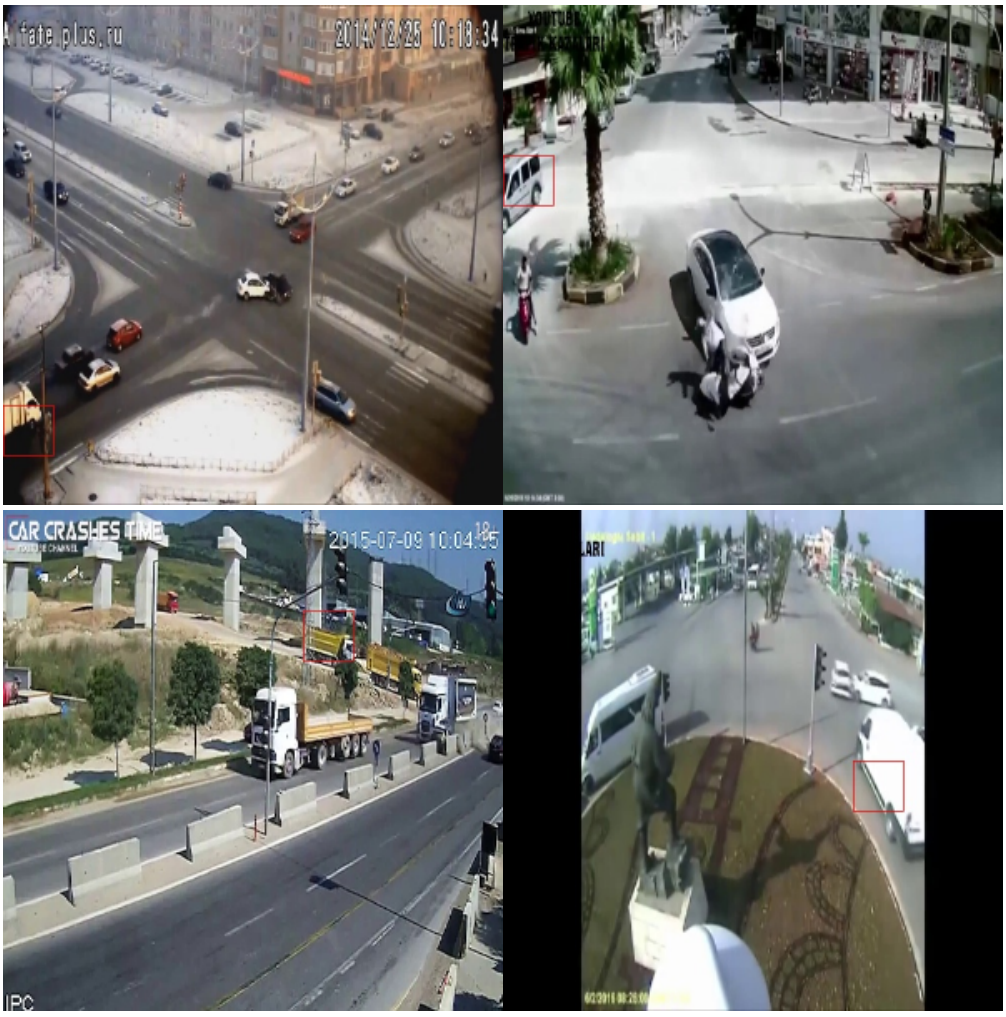
For visualization, part of the accident clip where the reconstruction error is highest is plotted.

Successfully Localized


Channel: Russian car crash compilation
2017-05-11 07:02:32
WWW.OOGNET.PL
ONLY



Failed to Localize



It is clear that the model is able to flag abnormal looking events in the frames. For better detection of accidents, motion needs to be considered too since this is failing in cases where the deformation due to accident is less and when new kinds of vehicles appear in the scene. Another failure case is when the scale of the vehicles in the video is very small. Introducing motion should solve the first two problems to an extent.

Detecting Anomaly in Vehicle Movement - Optical Flow

Detecting anomalies in vehicle movement can help detect accidents better. Dense optical flow has been used for this purpose.

Optical flow using OpenCV (can be found here) has been tried out.

FlowNet

Papers:

FlowNet: Learning Optical Flow with Convolutional Networks : https://arxiv.org/pdf/1504.06852

FlowNet2.0 : https://lmb.informatik.uni-freiburg.de/Publications/2017/IMKDB17/paper-FlowNet_2_0__CVPR.pdf

Optical Flow Estimation using a Spatial Pyramid Network link : https://arxiv.org/pdf/1611.00850.pdf

FlowNet takes two frames as input and predicts the optical flow.

The implementation present here (https://github.com/sniklaus/pytorch-liteflownet) has been used. Also for writing various functions including reading, writing and visualizing the flow, code has been taken from the following repos :

https://github.com/lmb-freiburg/flownet2/blob/master/scripts/run-flownet.py

https://github.com/opencv/opencv/blob/master/samples/python/opt_flow.py

The output of FlowNet is converted to HSV which is in turn converted to RGB using OpenCV. In it's RGB form, color of a patch represents it's directions and the thickness of the color represents it's speed.

An encoder architecture similar to the one that has been used before has been used. For this case, the number of filters at each layers has been reduced (learning optical flow is easier than learning visual representation). For experiments, half and one fourth the number of filters at each layer given in the previous architecture have been tested out. Since there's one flow between every two frames, using (50, 50, 15) as our visual slice gives us (50, 50, 14) shaped optical slice. The encoder has been adjusted to fit this. Rest of the training is similar as before.

Here's the final model

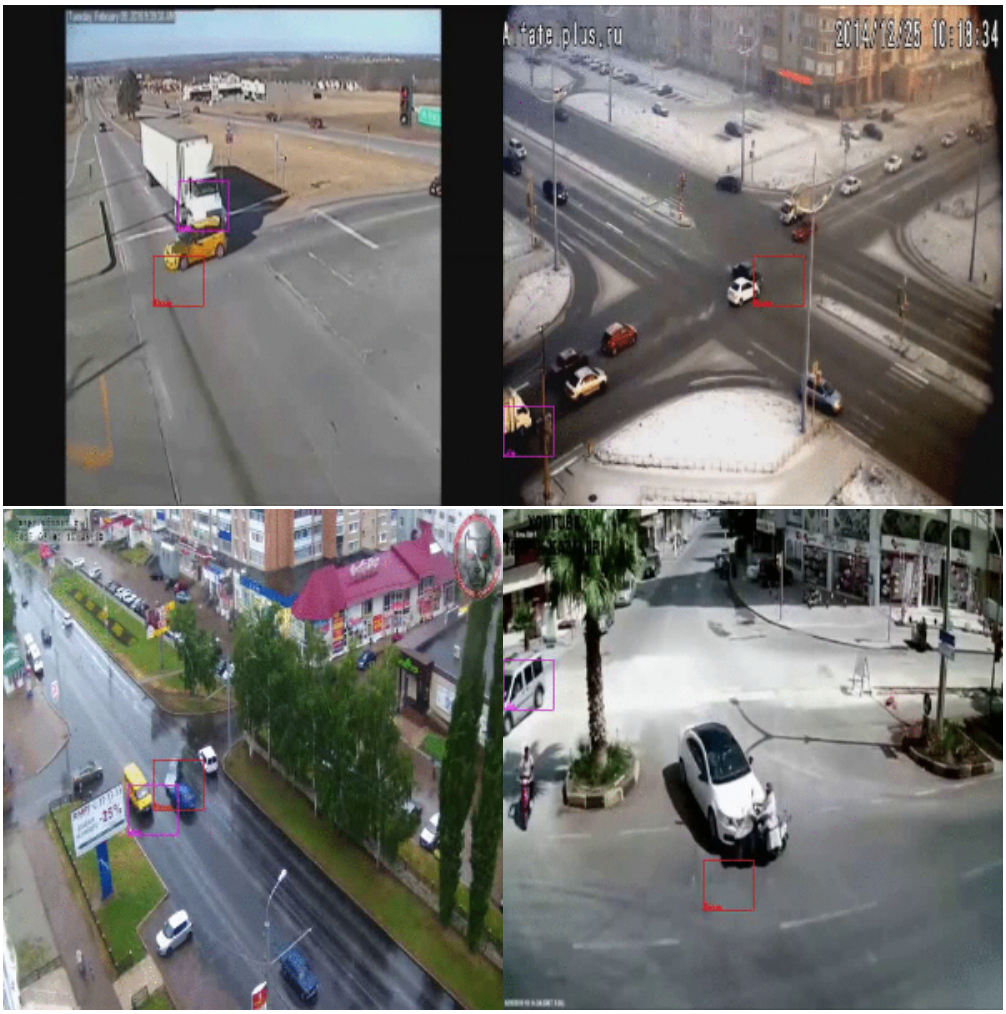
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 15, 50, 50, 3)     0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 15, 17, 17, 32)    4736
_________________________________________________________________
time_distributed_2 (TimeDist (None, 15, 17, 17, 32)    128
_________________________________________________________________
time_distributed_3 (TimeDist (None, 15, 17, 17, 32)    0
_________________________________________________________________
time_distributed_4 (TimeDist (None, 15, 9, 9, 16)      12816
_________________________________________________________________
time_distributed_5 (TimeDist (None, 15, 9, 9, 16)      64
_________________________________________________________________
time_distributed_6 (TimeDist (None, 15, 9, 9, 16)      0
_________________________________________________________________
convlstm1 (ConvLSTM2D)       (None, 15, 9, 9, 16)      18496
_________________________________________________________________
convlstm2 (ConvLSTM2D)       (None, 15, 9, 9, 8)       6944
_________________________________________________________________
convlstm3 (ConvLSTM2D)       (None, 15, 9, 9, 16)      13888
_________________________________________________________________
time_distributed_7 (TimeDist (None, 15, 17, 17, 32)    12832
_________________________________________________________________
time_distributed_8 (TimeDist (None, 15, 17, 17, 32)    128
_________________________________________________________________
time_distributed_9 (TimeDist (None, 15, 17, 17, 32)    0
_________________________________________________________________
time_distributed_10 (TimeDis (None, 15, 50, 50, 3)     4707
=================================================================
Total params: 74,739
Trainable params: 74,579
Non-trainable params: 160
_________________________________________________________________
```

Results

Model with validation loss of 532 has been chosen for experiments. It has been observed that optical flow helps detect accidents better in the cases when there is not much change visually. In many accidents, deformation is less which is hard to detect through visual reconstruction error like lateral movements of vehicles, sudden jerks. Optical flow solves this problem and helps detect such cases. Following are some results of such cases where visual detection failed but the optical flow encoder gives higher reconstruction error.

Note 1 - The below plots are the part of the clips with highest reconstruction error. The accident detection pipeline has not been completed.

Note 2 - In the below clips, pink rectangle shows the areas with highest visual reconstruction error and red rectangle shows the clips with highest optical flow reconstruction error.



Combined Representation :

Using an encoder working on combined representation of optical flow and the frames to improve results of the pipeline. In this case, STTV has 6 channels (3 rgb from image, and 3 channels from flow). Same encoder used for the visual representation is used. But, in this case the shape of the STTV will be 14 x 50 x 50 x 6. (14 since optical flow has only 14 frames. Last visual frame is ignored).

Different scenes and lighting conditions lead to different range of reconstruction errors. Fixing one threshold for all the conditions might not work well unless all the videos are similar (This effects visual and combined encoders more. Optical flow encoder is more robust to such changes). To work with different scenarios, to detect if a part of the frame is an anomaly, one way is to compare the reconstruction there with the reconstruction error of the rest of the frame. In this case, for testing, STTVs of a clip whose reconstruction error is higher that the sum of mean and twice the standard deviation of reconstruction errors of all the STTVs in the clip are flagged as animalities. This leads to a few false positives. Combining all three reconstruction errors might bring false positive rate down.

Testing has been done on the dataset from IIT Hyderabad available here. Visual encoder is failing due to low quality of the videos. This indicates the problems with visual representation. Out of the five accident videos available, optical flow is able to identify and localize two accidents.

Successes

Failures

Implementation Specifics For Training

Data used for training available at server-166, /disk1/intern1/soumith/datasets/auto-encoder-data

Normal frames from CADP videos (frames before accident begins) have been used.

- All the frames from ~1400 videos of CADP - /disk1/intern1/soumith/datasets/cadp_data/extracted_frames
- Good quality video frames from CADP (~180) - /disk1/intern1/soumith/datasets/cadp_data/cleaned_frames

- Normal frames extracted from CADP - /disk1/intern1/soumith/datasets/cadp_data/normal_clips

Data Format

The DataGenerator object expects data to be in the following format.

1. Each video should be converted to frames and all the frames of one video should be placed in a directory.
2. All these directories should be placed in another directory.
3. Make sure that there is nothing else in these directories.
4. In the video directory, frames should be named numerically. (eg 0.jpg, 1.jpg, ..)

> *Eg -*
>
> */data/*
>
> */data/train/*
>
> */data/train/video1/*
>
> *0.jpg*
>
> *1.jpg*
>
> *...*
>
> */data/val/*

Extracting STTVs

Aim is to extract the STTVs which have movement in them. This is done using background subtraction.

1. Each video is broken down into clips of 15 frames each.
2. All the frames in the clip are resized to 500 X 500.
3. Background subtraction is done on the 15 frames to detect movement. (Used smoothened image - Gaussian blur function in OpenCV to reduce noise)
4. This is then thresholded using binary threshold function in OpenCV.
5. A kernel of shape 50 x 50 x 15 with stride 25 has been used to extract STTVs. These STTVs are then thresholded in the process mentioned above.
6. Only the STTVs with movement in them are used for training. For detecting movement, from the thresholded STTV, the ratio of pixels that are moving is taken if this is more than a threshold, the STTV is used for training else it is discarded.
7. Same method is used during inference to see if an STTV has some movement.

Reading Optical Flow

1. Optical flow of each frame is precomputed (Not during training). This helps in faster training. Currently, this is the bottle neck in the pipeline.
2. A slightly changed implementation available here has been used (Can be found in repo here).
3. For each frame, if the name of the frame is FRAME_NAME.jpg, corresponding flow file is stored with it in the same directory as FRAME_NAME.flo
4. Note that flow is calculated between two frames. Flow between current and next frame is treated as flow image at the current frame.
5. This flow is converted into HSV and then into RGB using the functions available here and here. These RGB images are used as the input for encoder.
6. While extracting STTVs using background subtraction, the start (top left corner points) points of the STTVs which have been extracted are stored. These starts are used to extract corresponding flow STTVs.
7. Note that for 15 frames, there will be only 14 flow frames. In our case, the shape of an flow STTV is (14, 50, 50, 3).

Combined Representation

1. Visual and flow representations are stacked together. Since there are only 14 flow frames for 15 visual frames, last frame from visual representation is neglected.
2. The shape of the combined representation STTV is (14, 50, 50, 6).

Steps_per_epoch and val_steps

1. It is not apparent what these values are given a dataset because, the elements of batches are STTVs and not frames. And there may be any number of STTVs that pass the threshold given a video.
2. To get this number for a dataset, one way to go is to use DataGenerator object.
3. Create a DataGenerator object with the path and batch size. Keep calling it until it returns false (The first return value will be false if data is exhausted).
4. The number of times 'next' can be called before the object runs out of the memory gives the number of batches of STTVs in the dataset.

Other references used:

Simple Online and Realtime Tracking with a Deep Association Metric : https://arxiv.org/abs/1703.07402

Real-world Anomaly Detection in Surveillance Videos : https://arxiv.org/abs/1801.04264

Action Recognition with Improved Trajectories : http://lear.inrialpes.fr/~wang/download/iccv13_poster_final.pdf