DR. S.P.M. INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY - NAYA RAIPUR

MINOR PROJECT
IVTH SEMESTER

Accident Identification using Deep Learning

Author:
Marisetty Ashish
Addagatla Nishanth

Supervisor:
Dr.Muneendra Ojha

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Technology

in the

Department of Computer Science and Engineering

September 11, 2021

Declaration of Authorship

We Marisetty Ashish, Addagatla Nishanth declare that this thesis titled, "Accident Identification using Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
Date:

Signed:
Date:

DR. S.P.M. INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY - NAYA RAIPUR

Abstract

Dr.Muneendra Ojha
Department of Computer Science and Engineering

Bachelor of Technology

Accident Identification using Deep Learning

by Marisetty Ashish
Addagatla Nishanth

Road accidents are one of the serious threats to the lives of people within the age group 18-45 globally, according to the World Bank report. India has 1 per cent of the world's vehicles but accounts for 11 per cent of all road crash deaths. To put things in into perspective for every hour passed 53 road crashes occur i.e, 1 person is deceased every 4 minutes in a road accident.

Road accidents are the eighth leading cause of death globally. And India alone accounts for about 13 lakh people deceased and another 50 lakh injured people in the past decade. Traffic accidents not only take life, but also disrupt traffic operations, disrupt traffic flow and cause serious urban problems worldwide.

Major accidents can sometimes lead to long term damage, injuries and even deaths. After years of study, it has been broadly acknowledged that significant reductions in the impact of accidents can be accomplished through effective detection systems and appropriate reaction strategies.

The increasing availability of very large data-sets of small video footage (in the order of millions of labeled samples) enables rapid progress on difficult computer vision issues such as event, activity detection, common sense interpretation or prediction of future events, thereby facilitating timely assistance to the road accident victims. The ultimate aim of the research project is to deploy the system on edge devices capable of detecting an accident in real time and alerting emergency services.

Keywords - Deep Learning, Video Analytics, Real-time application

Acknowledgements

We are grateful to IIIT-NR and our advisor, Dr. Muneendra Ojha, for allowing us to work on the research problem and providing us with unconditional support.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Literature Review 3
2.1 Convolutional Neural Networks . . . 3
2.1.1 Convolutional layer . . . 3
2-D convolutions . . . 3
3-D convolutions . . . 3
2.1.2 Pooling layer . . . 5
Max pooling . . . 5
Average pooling . . . 5
Global Average Pooling . . . 5
2.1.3 Fully connected layers . . . 5
2.1.4 Review of papers using CNN architecture . . . 5
A Deep Event Network for Multimedia Event Detection and Evidence Recounting . . . 5
Large-scale Video Classification with Convolutional Neural Networks . . . 6
2.2 Sequence Models . . . 8
2.2.1 Recurrent Neural Network : . . . 8
2.2.2 Long short-term memory Network : . . . 9
2.2.3 Residual Networks . . . 9
Review of A New Video-Based Crash Detection Method: Balancing Speed and Accuracy Using a Feature Fusion Deep Learning Framework(Lu et al., 2020) . . . 10

3 Proposed Solution 13
3.1 Key terminology and concepts: . . . 13
3.1.1 Residual Networks . . . 13
3.1.2 Transfer Learning: . . . 14
3.1.3 Kinetics - 400 data-set . . . 14
3.2 Implemented models . . . 14
3.2.1 Transfer learning on 3D ConvNet's . . . 15
Convolutional residual blocks for video data . . . 15
Architecture of 3D-ResNet . . . 16
Implementation problems encountered and solutions . . . 17
3.2.2 2-D Resnets on temporal data . . . 17
Architecture and training pipeline . . . 17

Implementation problems encountered and solutions 18

4 Results **19**
4.1 Transfer learning on 3D ConvNet's 19
Data-set Used 19
4.1.1 Model summary and inference 19
4.2 2D Resnets on temporal data 20
Data-set Used 20
Model summary and inference 20
4.3 Comparison of the two models : 21

5 Conclusion **22**

Bibliography **23**

List of Figures

2.1 Visualization of 2-D convolution 4
2.2 Visualization of 3-D convolution 4
2.3 DevNet Architecture 6
2.4 Visualizing event recounting results 7
2.5 Different fusion types across time 8
2.6 RNN architecture 8
2.7 LSTM architecture 9
2.8 Model architecture 10
2.9 Visual Attention Module architecture 11
2.10 Convolutional block attention module architecture 12
2.11 ResNet + attention module 12

3.1 Architecture of Residual Networks. 13
3.2 Kinetic-400 data-set. 15
3.3 Block diagram of 3-D ResNet 16
3.4 Architecture of 3-D ResNet 16

4.1 Sample frames from the video data-set 19
4.2 Sample frames from the video data-set 20
4.3 Output of 2D ResNet model 21

List of Tables

4.1 Preformance metrics of 3D ResNet architecture. 20
4.2 Performance metrics of 2D ResNet architecture. 21

Chapter 1

Introduction

Intelligent Video Analytics is a field of research in which the temporal and spatial information of the scene is analyzed, interpreted and processed from a stream of live or captured video data. And these systems have become an invaluable method for researchers due to the abundance of zettabytes of data and improved training algorithms.Surveillance cameras have been extensively deployed in many cities and highways as a result of the growth of intelligent transportation systems (ITS). Vision-based crash detection strategies have gotten a lot of recognition in recent years because of their wide coverage and their ability of using existing architecture of network of surveillance cameras rather than building a new architecture. Their basic idea is to use machine vision tools to automatically distinguish collision scenes based on the characteristics of traffic images/videos. As a successful intelligent crash detection tool, such techniques are expected to dramatically reduce human labor and have a reasonably high detection accuracy. In general, there are two main types of features of interest: motion (temporal) and appearance (spatial) features. Appearance features include apparent damage to the vehicle, vehicle rollovers, etc. Motion features, including the intersection of vehicle trajectories and the gathering of pedestrians, need to be continuously identified. From this perspective, current crash detection methods can be divided into two groups: motion-based methods and fusion-based methods. Some motion features that are frequently used are intersection of vehicle trajectories, overlap of bounding box detectors, and changes in vehicle speed. Some of the methods used to extract the motion characteristics of vehicles (acceleration, direction and velocity) on the basis of which certain rules and thresholds were applied, to the identification of accidents [Gu et al. (2019), Ki and Lee (2007), Thomas, Gupta, and Subramanian (2017)]. In recent years, the performance of vehicle detection and tracking has significantly improved with the development of deep learning methods and big data. Vicente and Elian Arceda and Riveros (2018) used the YOLO model to detect motion features and used the support vector machine (SVM) to identify the crash. Lee and Shin Lee and Shin (2019) have used Faster R-CNN for vehicle detection and Simple Online and Real-Time Tracking (SORT) for vehicle tracking. The incident/crashes in tunnels were detected on the basis of these motion features. Paul Ijjina et al. (2019) applied Mask R-CNN (Mask Region-based CNN) to extract motion features and applied rules for crash detection.But motion-based models depend only on the motion of the vehicle. This requires a high level of object detection and tracking accuracy. The vehicle detentions and tracking performance could be reduced if the traffic environment is challenging, resulting in low performance for crash detection. Recently, the feature fusion-based (i.e., appearance and motion) crash detection methods have become increasingly popular. These fusion-based features can be again be further divided into 2 categories: i) Supervised learning methods. ii) Unsupervised learning methods. For instance, Dinesh Singh and Krishna Mohan Singh and Mohan (2018) developed a crash detection model based

on autoencoder methods in the unsupervised learning domain. And in supervised learning framework, Batanina et al. employed two-stream network to separately extract appearance features and motion features, which were then further combined to detect crashes. Although these feature fusion-based methods have achieved a better performance than motion feature-based methods, some improvements still can be made. Few of the drawbacks of these fusion modules are that these models necessitate a large amount of computing resources and a long computation time, preventing their use in a real-time traffic context.

Chapter 2

Literature Review

Traditional hand-crafted features combined with classical computer vision techniques were used for event detection prior to deep learning and big data. With the rise of big data, deep learning architectures have gained traction for their ability to address complex problems, automatic feature selection, and ability to learn complex non-linear patterns. In the next section, few of the relevant methods are discussed in the context of event detection and recognition (accident detection).

2.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a neural network that has one or more convolutional layers layers and is used primarily for tasks such as image processing, classification, segmentation etc. The layers in convolutional neural network can normally be categorized in to one of three of these categories:

i) Convolutional layers. ii) Pooling layers. iii) Fully connected layers.

2.1.1 Convolutional layer

In computer vision the input is often a 3 channel RGB image. Each convolutional layer contains a series of filters known as kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel.

The kernel moves through the input matrix of the numbers moving horizontally column by column, sliding/scanning over the first rows of the matrix containing the pixel values of the images. Each pixel in the grid (The area the kernel is able to view of input image at a time) is multiplied by the corresponding value in the kernel, then the result is summed up for a single value representing a grid cell in the output feature map. Then the kernel moves down to the next rows vertically.

2-D convolutions

In a simple 2-D convolution operation, an image of size $H \times W \times D_{in}$ is convolved with a $f_h \times f_w \times D_{in}$ channeled filter to output a feature map of size $H - f_h + 1 \times W - f_w + 1$ per filter. (As represented in Figure 2.1)

3-D convolutions

In a simple 3-D convolution operation, an image of size $H \times W \times D_{in}$ is convolved with a $f_h \times f_w \times f_d$ channeled filter to output a feature map of size $H - f_h + 1 \times W - f_w + 1 \times D_{in} - f_d + 1$ per filter. (**Note**: Here $f_d ! = fw$). (As represented in Figure 2.2)

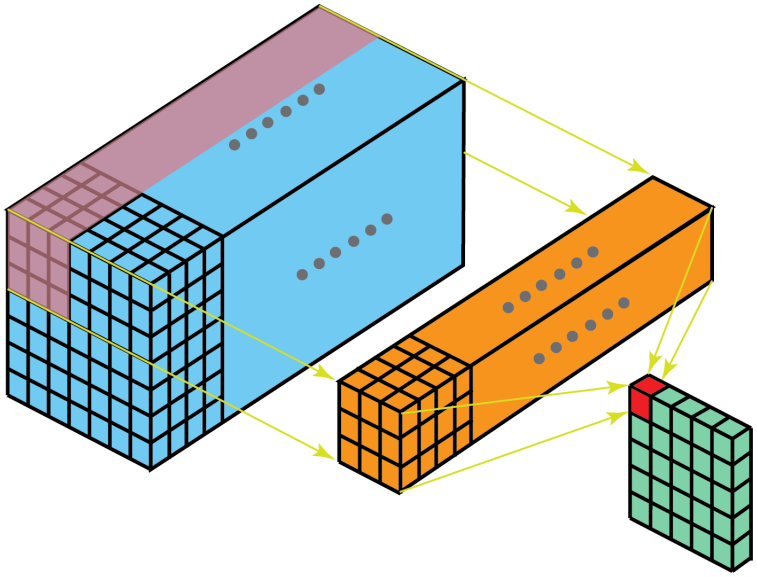

FIGURE 2.1: Visualization of 2-D convolutions

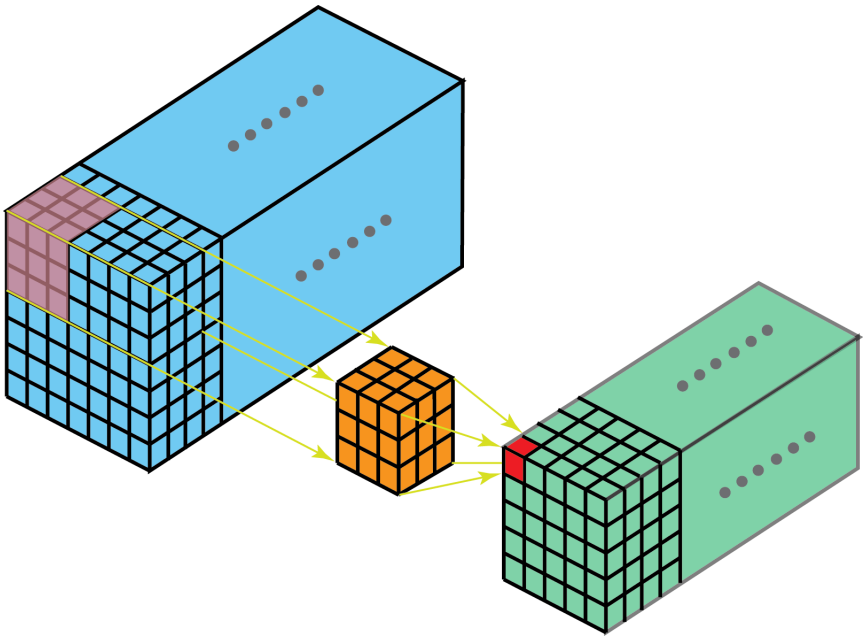

FIGURE 2.2: Visualization of 3-D convolutions

2.1.2 Pooling layer

The pooling layer typically facilitates a downsampling operation that reduces the dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions and to reduce the number of learning parameters.

Max pooling

The most popular form of pooling operation is max pooling, which extracts grids from the input feature maps and returns the maximum value in each patch, and discards all the other values.

Average pooling

Another form of pooling operation is average pooling, which extracts patches from the input feature maps and returns a singular float, the average values of all the pixels in each patch, to represent as a pixel in the output feature map.

Global Average Pooling

Another pooling operation worth noting is a global average pooling. A global average pooling performs an extreme type of downsampling, where a feature map with size of height X width is downsampled into a 1 X 1 array by simply taking the average of all the elements in each feature map, whereas the depth of feature maps is retained.

2.1.3 Fully connected layers

The output feature maps of the final convolution or pooling layer is typically flattened, i.e. transformed into a one-dimensional array, known as dense layers. A dense layer is typically found at the end of the network after a pooling layer. These layers are combined with each other to form a dense neural network known as fully connected layers. The final fully connected layer typically has the same number of output nodes as the number of classes.

2.1.4 Review of papers using CNN architecture

In this section we'll review few of the prominent papers using CNN architecture for event detection.

A Deep Event Network for Multimedia Event Detection and Evidence Recounting

The main focus of this paper [Gan et al. (2015)] is on the complex detection of events in videos. In addition to providing a single event label, the network also provides us with key evidence (key video frames) that lead to the decision to detect. The process is called Multimedia Event Recounting (MER). And it is addressed in the paper as an event recounting.In the proposed network, the CNN architecture includes nine convolutional layers and three fully-connected layers with a spatial pyramid pooling in between, very similar to the ImageNet winning network architecture [Krizhevsky, Sutskever, and Hinton (2012)].

The training of the paper can mainly be broken down in to two steps:

i) **Pre-training:** First, we use the large ImageNet dataset to pre-train CNN for initialization of the parameters. The goal of this pre-training stage is to learn the generic features at the image level.

ii) **Cross image max-pooling:** Then we remove the softmax classifier and the last fully connected layer of the pre-trained network because it is specific to the ImageNet classification task. Next, to aggregate the image-level features in the video-level representation, cross-image max-pooling is applied to fuse the outputs of the second fully-connected layer from all key frames in the same video (As in figure 2.3). Then a c-way independent logistic regression classifier is then used to generate the detection score for each event class. Here c denotes the number of classes of events.

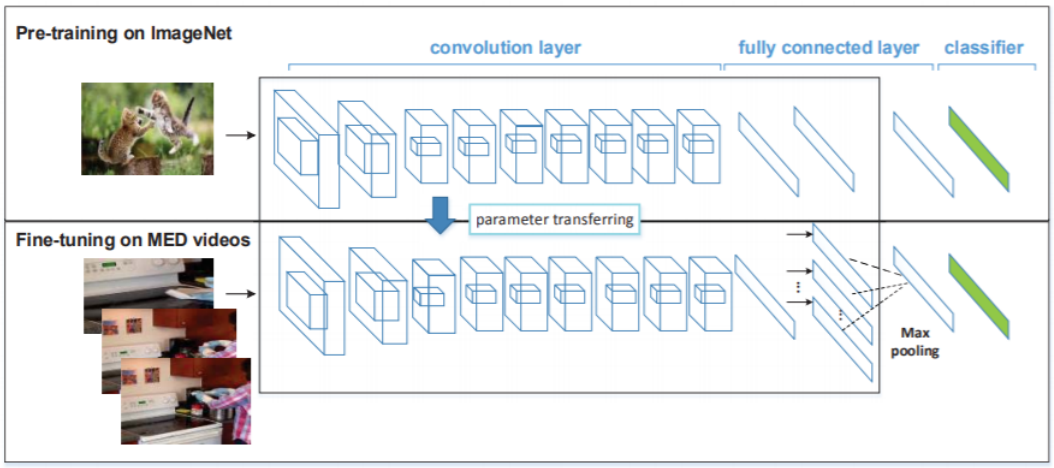

FIGURE 2.3: Representation of DevNet Architecture

Event Recounting: Coming to the paper's event-recounting approach, the main idea is that, given DevNet's learned detection and class of interest, we can trace back to the original input image by a backwards pass, with which we can find out how each pixel affects the final detection score for the specified event class. Thus for each event class, we can derive a single class-specific saliency score for each pixel in the video. After obtaining the spatial-temporal saliency map, we average the saliency scores of all pixels within the keyframe to obtain the keyframe level saliency score, and then we rank the keyframe level saliency scores to obtain the informative keyframe. For the top ranked key frames, we use the saliency scores as guidance and apply the graph-cut algorithm to the segment the active region in the salient maps to produce a output as in Figure 2.4.

Large-scale Video Classification with Convolutional Neural Networks

In this paper Karpathy et al. (2014) the authors suggest multiple approaches for extending the connectivity of a CNN in time domain to take advantage of local spatio-temporal information and also suggest a multiresolution, foveated architecture as a promising way of speeding up the training.

Since each clip contains several contiguous frames in time, authors extended the connectivity of the network in time dimension to learn spatio-temporal features. There are multiple options for the precise details of the extended connectivity but here they mainly focus on three broad connectivity pattern categories, namely Early Fusion, Late Fusion and Slow Fusion (Figure 2.5). Hence, they described a baseline

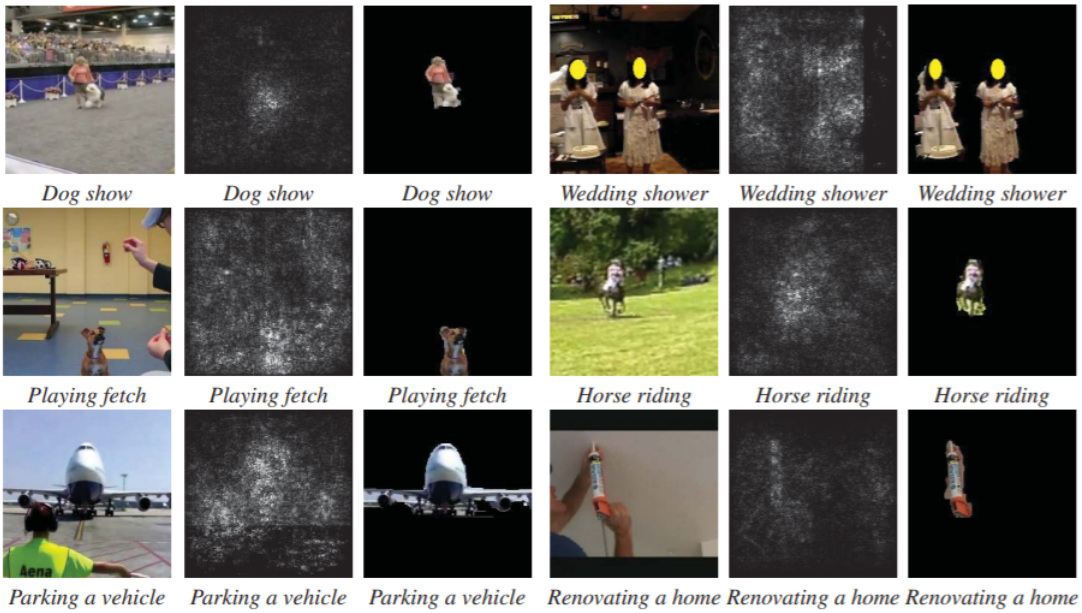

FIGURE 2.4: Event recounting results generated by DevNet. From left to right are top one temporal key evidence, spatial saliency map, and spatial key evidence

single-frame CNN and then discussed different types of fusion in time as described briefly below.

Single-frame: Authors used a single-frame baseline architecture similar to the ImageNet challenge winning model - 2012 [Krizhevsky, Sutskever, and Hinton (2012)], but accepts inputs of size 170 × 170 × 3 pixels instead of the original 224 × 224 × 3. Using shorthand notation, the full architecture is C(96, 11, 3)-N-P-C(256, 5, 1)-N-P-C(384, 3, 1)- C(384, 3, 1)-C(256, 3, 1)-P-F C(4096)-F C(4096), where C(d, f, s) indicates a convolutional layer with d filters of spatial size f ×f, applied to the input with stride s. F C(n) is a fully connected layer with n nodes. All pooling layers P pool spatially in non-overlapping 2 × 2 regions. The final layer is connected to a softmax classifier with dense connections.

Early Fusion : The Early Fusion extension combines information immediately at the pixel level across the entire time window. This is done by modifying the filters on the first convolution layer in the single frame model by extending them to a size of 11 × 11 × 3 × T pixels, where T is temporary (we use T = 10, or about a third of a second). Early and direct connectivity to pixel data enables the network to accurately detect local motion direction and speed.

Late Fusion : The Late Fusion model sets up two separate single-frame networks (as described above, up to the last convolutional layer C(256, 3, 1) with shared parameters at a distance of 15 frames and then merges the two streams into the first fully connected layer. Thus, neither single frame tower alone can detect any motion, but the first fully connected layer can calculate global motion characteristics by comparing the outputs of both towers.

Slow Fusion : The Slow Fusion model is a balanced mix between the two approaches that slowly fuse temporal information across the network in such a way that higher layers gain access to progressively more global information in both spatial and temporal dimensions. In the model we use, the first convolutional layer is

extended to apply every filter of temporal extent T = 4 on an input clip of 10 frames through valid convolution with stride 2 and produces 4 responses in time. The second and third layers above iterate this process with filters of temporal extent T = 2 and stride 2.

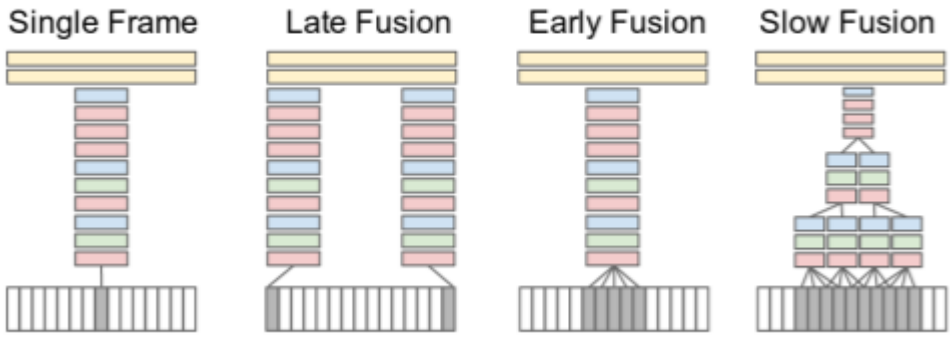

FIGURE 2.5: Various approaches to merging information over time across the network. Red, green and blue boxes indicate the convolution, normalisation and pooling layers, respectively.

Results of this paper indicate that while performance is not particularly sensitive to the architectural details of connectivity over time, the slow fusion model consistently performs better than the early and late fusion alternatives. Surprisingly, even the single frame model already shows very strong performance.

2.2 Sequence Models

2.2.1 Recurrent Neural Network :

A feed forward neural network with an intrinsic memory is known as a recurrent neural network(Figure 2.6). RNN is recursive in nature since it performs the same operation for each data entry, and the current input's output is dependent on the previous computation. The output is copied and sent back into the recurrent network after it is created. It respects the new input as well as the output it has acquired from the previous input when making a decision.

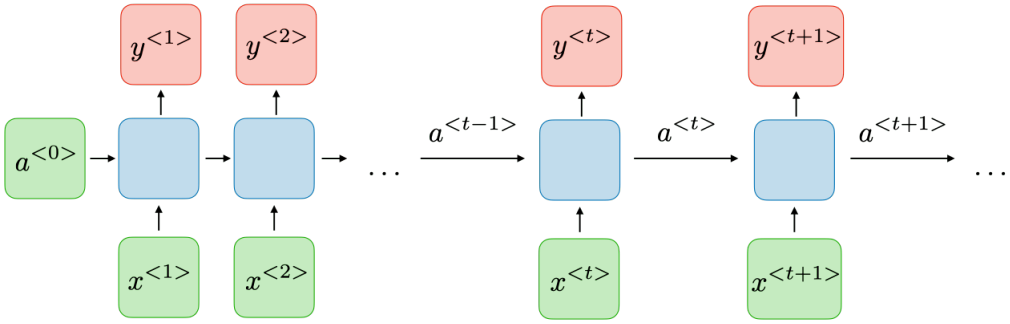

FIGURE 2.6: RNN architecture.

RNNs, unlike feed forward neural networks, can process sequences of inputs using their internal state (memory). As a result, activities like unsegmented, linked

handwriting recognition or speech recognition are possible. Many of the inputs in other neural networks are independent of one another. However, in an RNN, all of the inputs are connected to one another.

2.2.2 Long short-term memory Network :

Long Short-Term Memory networks (Figure 2.7) are a simplified variant of recurrent neural networks that make it easier to recall information from the past. Here, the RNN's vanishing gradient problem is solved. Provided time lags of uncertain length, LSTM is well-suited to characterise, process, and forecast time series. Backpropagation is used to train the algorithm. The three gates that constitute an LSTM network are:

i) **Input gate:** Input gate Discovers which value from the input can be used to change the memory with an input gate. The sigmoid function determines which values are allowed to pass through 0,1. and the *tanh* function assigns weight to the values transferred, determining their magnitude on a scale of -1 to 1.

ii) **Forget the gate:** Forget the gate figures out what information can be removed from the block. The sigmoid function determines this. For each number in the cell state C_t1, it looks at the previous state (ht_1) and the material input (X_t) and outputs a number between 0 (omit information) and 1 (keep information).

iii) **Output gate:** the output is determined by the block's input and memory. The Sigmoid function determines the values to allow by 0,1, and the *tanh* function assigns weightage to the values that are transferred, varying from -1 to 1, and is multiplied by the sigmoid function's output.

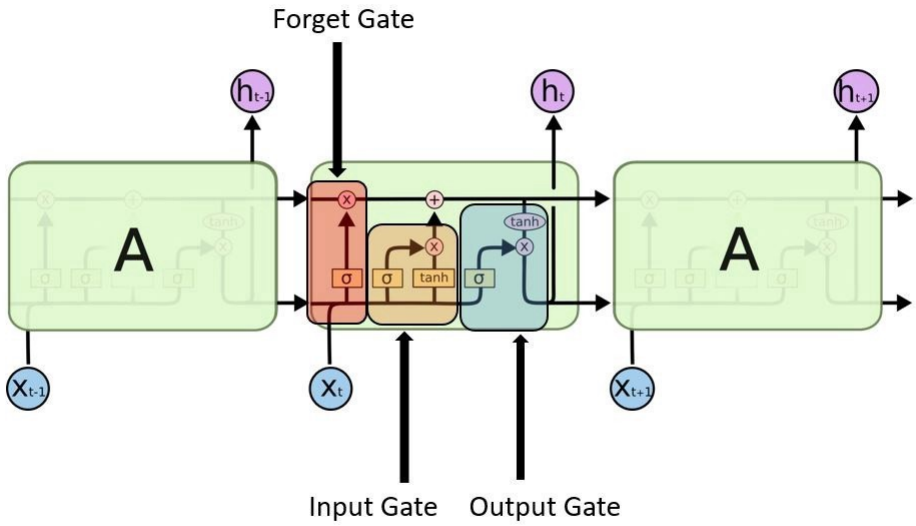

FIGURE 2.7: LSTM architecture.

2.2.3 Residual Networks

A Residual Network (ResNet)(He et al., 2016) is a Convolutional Neural Network (CNN) architecture that allows hundreds of convolutional layers to be used. ResNet

can incorporate a large number of layers with good results, whereas prior CNN architectures saw a drop off in the effectiveness of additional layers. ResNet was a ground-breaking solution to the problem of vanishing gradients. Backpropagation is a method of training neural networks that uses gradient descent to find the weights that minimise the loss function. If there are so many layers, repeated multiplication shrinks the gradient until it "disappears," allowing output to saturate or even degrade with each new layer.

The alternative suggested by ResNet to this issue is "Identity Bypass Connections". ResNet stacks identity mappings, which are layers that don't do much at first, and skips over them, reusing activations from previous layers. Skipping layers compresses the network into a few layers at first, allowing for quicker learning. As the network retrains, all layers are extended, and the network's "residual" bits are extinguished. As the network is retrained, all layers are extended, and the network's "residual" sections discover more and more of the source image's feature space.

Review of A New Video-Based Crash Detection Method: Balancing Speed and Accuracy Using a Feature Fusion Deep Learning Framework(Lu et al., 2020)

The model's overall structure is as depicted in Figure 2.8. To begin, the attention module was combined with ResNet to capture the crash images' appearance features. The ResNet will increase the speed of a traditional convolutional neural network, while the attention module allows the model to concentrate on localised appearance features rather than other unrelated data. The output function map is then reduced in dimension using a 1 X 1 convolutional layer before being chronologically fed into the Conv-LSTM network to retrieve further crash motion functions. Conv-LSTM outperforms traditional recurrent neural networks (e.g. LSTM) in terms of weight and spatial information retention. Finally, a completely linked layer and a global pooling layer were used to detect a crash (or no crash).

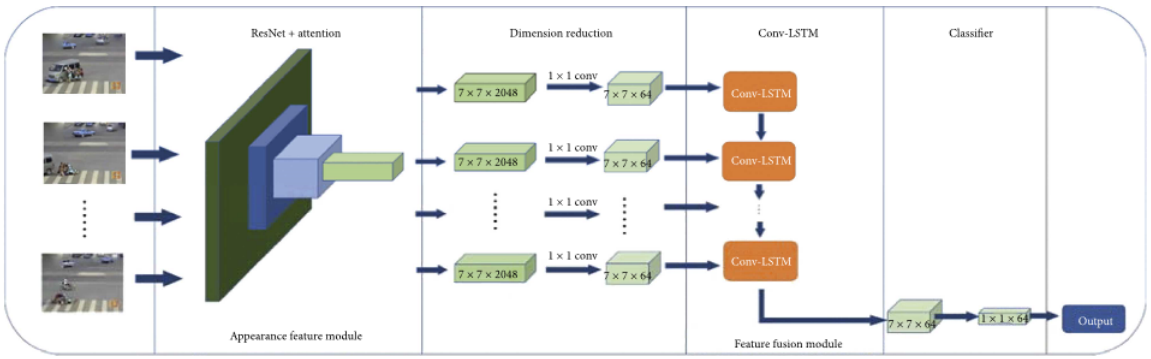

FIGURE 2.8: Model architecture.

The attention module and Conv-LSTM module used in the above mentioned network are described in detail below:

Feature Fusion Module (Conv-LSTM) : Conv-LSTM module(Shi et al., 2015) was first used in precipitation nowcasting (Problem of providing very short range (e.g., 0-6 hours) forecast of the rainfall intensity in a local region based on observation data). Data flattening is needed for traditional LSTM input, which frequently results in spatial information loss. The Conv-LSTM module uses a convolution neuron as a basic unit to preserve spatial information while inheriting the gating structure of standard LSTM.

Visual Attention Module : ResNet is further enhanced squeeze-and-excitation (SE) Block(Hu, Shen, and Sun, 2018). Since it is relatively simple and can increase the efficiency of many convolutional network models, this module has been widely used. SE Block is part of the channel attention mechanism, which assigns different weights to different channels of a feature map. Different channels in convolutional neural networks lead to different feature extractions. Different feature choices should be emphasised in different classification tasks. The definition is close to how we recognise things as humans. When judging cats and dogs, for example, people may concentrate on shape features, while when judging jaguars and leopards, they may focus on texture features. As a result, SE Block enhances the ability of convolutional neural networks to pick features.

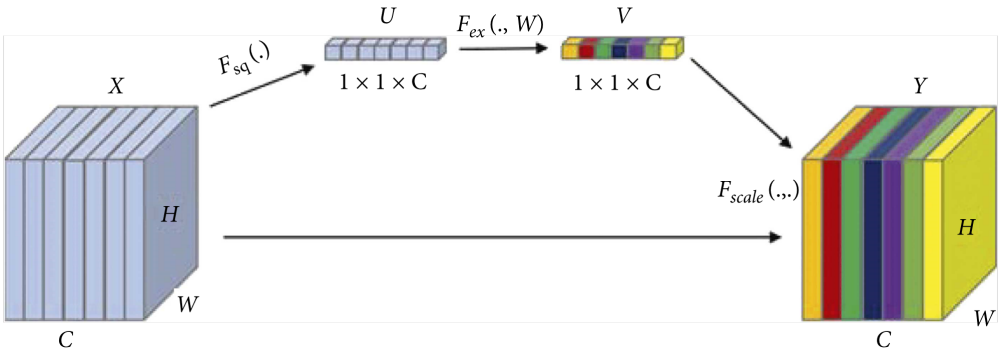

FIGURE 2.9: Visual Attention Module architecture.

First a global average pooling is performed after which, the output U is passed through a fully connected layer with a weight of W (W $\in R_{C\times C}$), that is $F_{ex}(, W)$ in Figure 2.9. And the result V is as depicted in the equation below , where " * " refers to matrix multiplication.

V = $F_{ex}(U, V) = \sigma(U * V)$

Given above is the activation function. The consequence V is also known as attention weight. To change the value of different channels of the signal, multiply the attention weight V and the input X by the channel weight as depicted by the below equation (Here " . " refers to the element-by-element multiplication.)

Y = F_{scale}(X, V) = X.V

Convolutional block attention module:

Convolutional block attention module(CBAM)(Woo et al., 2018)is an enhanced visual attention module over SE. CBAM presents the spatial visual attention module, which is based on the basic channel visual attention module (i.e., SE), in a groundbreaking way, as seen in Figure 2.10. The spatial visual attention module, unlike the basic module, performs maximum and average pooling operations $F_{s\_sq}(.)$ on the input X_S by channel before converting the two-layer feature map to a single-layer feature map using a 1×1 convolutional layer with a weight of W, as shown in $F_{s\_ex}(., W)$ in Figure 4. Finally, softmax is used to transform the initial distribution to a probability distribution and to adapt the model's value to various input spatial positions. This entire process can be expressed by the three following equations:

$U_S = F_{s\_sq}(X_S)$

$V_S = F_{s\_ex}(U_S, W) = softmax(U_S * W)$

$Y_S = F_{scale}(X_S, V_S) = X_S.V_S$

The CBAM module can be embedded into residual modules to improve its feature selection performance as in Figure 2.11.

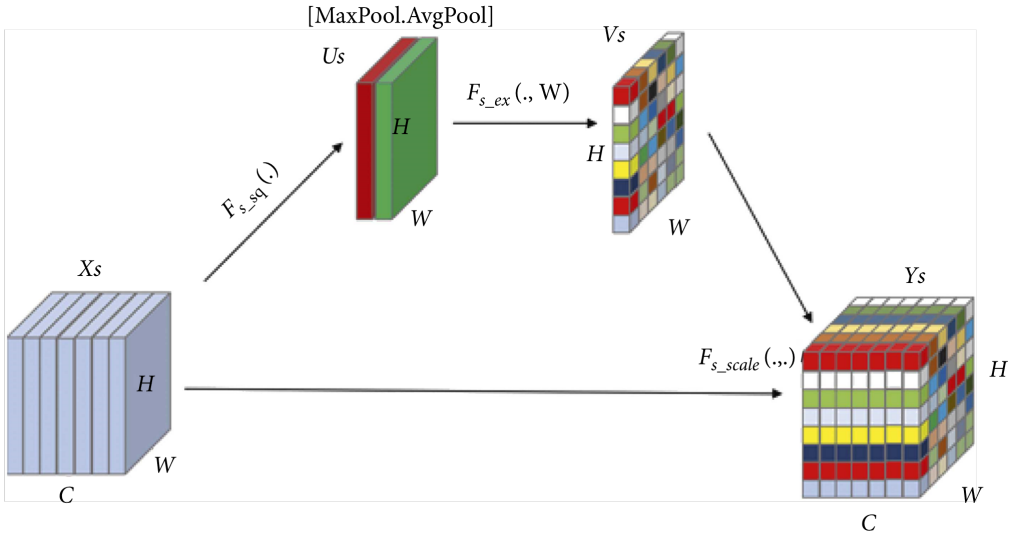

FIGURE 2.10: Convolutional block attention module architecture.

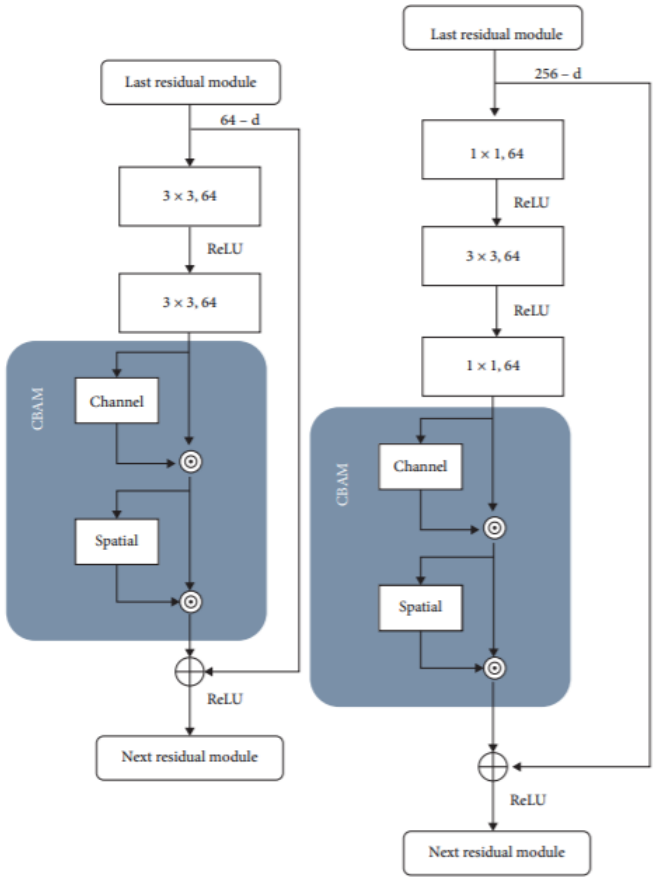

FIGURE 2.11: Positioning of CBAM module in ResNet model.

Chapter 3

Proposed Solution

As a solution to this research problem, we have adopted and tested 2 different models. The use of ResNet and more importantly residual blocks is a shared feature of both approaches. We shall address briefly the key terminology and concepts in the next section, before moving on to implemented models section:

3.1 Key terminology and concepts:

3.1.1 Residual Networks

A Residual Networks (ResNet)(He et al., 2016) are a form of neural network that uses identity mapping to solve problems. This ensures that the input to one layer is transferred to another layer directly or as a shortcut (i.e. through skip connection). ResNet solves vanishing gradients problem. Backpropagation is a neural network training technique that employs gradient descent to determine the weights that minimise the loss function. If there are so many layers, repeated multiplication shrinks the gradient until it “disappears,” allowing output to saturate or even degrade with each new layer. ResNet uses skipping layers. Skipping layers compresses the network into a few layers at first, allowing for quicker learning. As the network is retrained, all layers are extended, and the network’s "residual" sections discover more and more of the source image’s feature space.

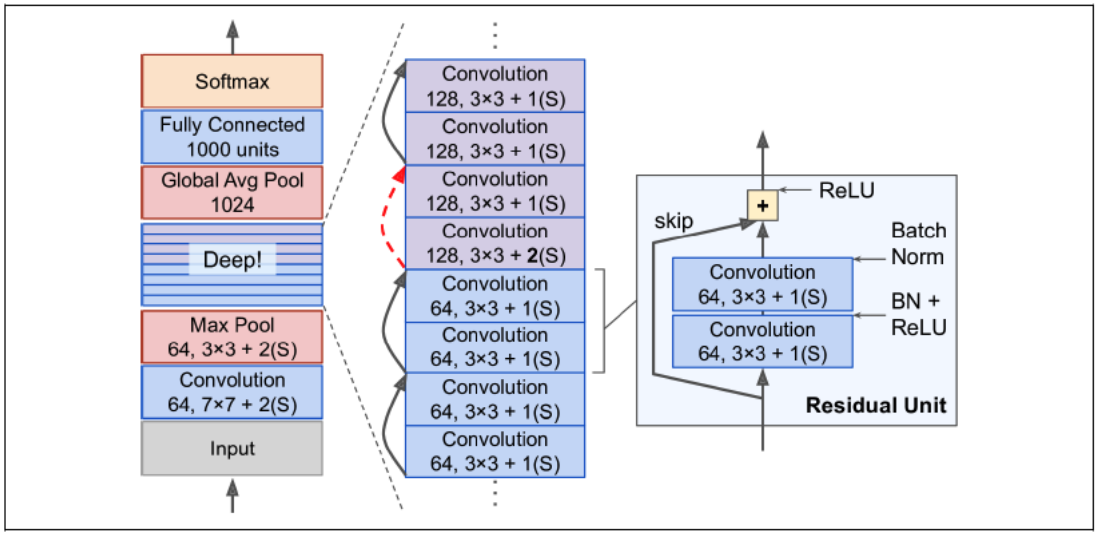

FIGURE 3.1: Block diagram of Residual Networks.

3.1.2 Transfer Learning:

Transfer learning is a machine learning research problem (ML), which focuses on the storage of information learned while solving a problem and applying it to a separate but associated problem. For example, when you try to recognise cat, information gained when learning to recognise vehicles can be used.Training a deep neural network from scratch is often not possible for a variety of reasons, including the need for a large enough data-set and the fact that achieving convergence can take too long for the tests to be worthwhile. Even if a large enough data-set is available and convergence is fast, it is always preferable to start with pre-trained weights rather than random initialised ones. One of the big transfer learning possibilities is fine-tuning the weights of a pre-trained network by continuing the training loop. Considering that the transferability of features reduces as the distance between the pre-trained and target tasks grows, moving features even from distant tasks may be preferable to random initialization. However, putting this transfer learning method into practise is not completely straight forward. On the one side, there are architectural requirements for using a pre-trained network. Nonetheless, since it is unusual to come up with a whole new architectural design, it's popular to reuse current network architectures (or components) in order to facilitate transfer learning. When fine-tuning rather than starting from scratch, however, the preparation phase varies somewhat. It's important to pick the right layers to fine-tune – typically the network's higher levels, since the lower one happens to have more generic characteristics – and choose an acceptable learning rate strategy, which is normally smaller since the pre-trained weights are supposed to be pretty good, so there is no reason to adjust them significantly. The size of per-pixel labelled segmentation data-sets is not as high as the size of classification data-sets like ImageNet due to the inherent complexity of collecting and constructing them. When working with RGB-D or 3D data-sets, which are much smaller, the problem becomes even worse. As a result, transfer learning, especially fine-tuning from pre-trained classification networks, has become a popular trend for many other problem paradigms, and has been successfully applied in the methods discussed in the following sections.

3.1.3 Kinetics - 400 data-set

Kinetics - 400 (Kay et al., 2017) is a collection of large-scale, high-quality datasets of URL links of up to 650,000 video clips that cover 400/600/700 human action classes, depending on the dataset version. The videos include human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. Each action class has at least 400/600/700 video clips. Each clip is human annotated with a single action class and lasts around 10 seconds.

3.2 Implemented models

We have used two different approaches - one is a research paper implementation of the paper *'A Closer Look at Spatiotemporal Convolutions for Action Recognition'* (Tran et al., 2018) and the other using a simple 2-D ConvNet which combines temporal information during the inference of the model. And in the upcoming sections, the model architectures and the training pipelines of these models are to be briefly discussed.

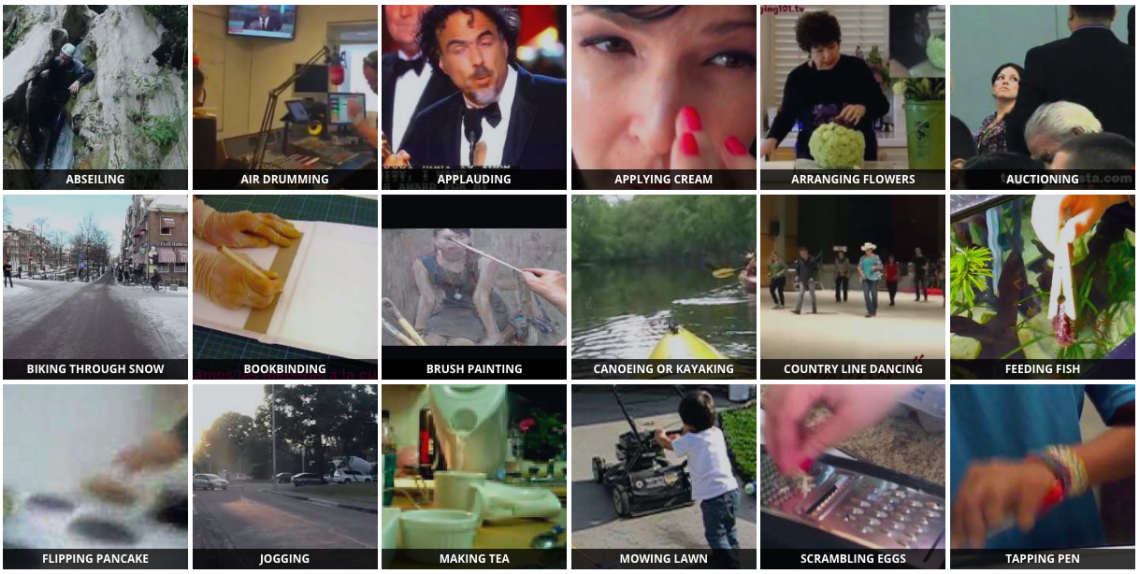

FIGURE 3.2: Kinetics-400 dataset

3.2.1 Transfer learning on 3D ConvNet's

In the research paper *A Closer Look at Spatiotemporal Convolutions for Action Recognition*, the authors explored various types of video analytical spatiotemporal convolutions and analysed their effect on the action recognition problem. They also validated that, in training and evaluating large scale,challenging actions such as Sports1M and Kinetics, 3D ResNets are substantially superior to 2-dimensions for the same given depth. We used this model to transfer learn the features to our accident detection problem with the available pre-trained weights.

Convolutional residual blocks for video data

Let x denote the input clip of size $3 \times L \times H \times W$, where L is the number of frames in the input video, H and W are the video frame height and width, and 3 refers to the RGB channels. In this model we consider each block to consist of two convolutional layers with a ReLU activation function after each layer, without the bottleneck layers. Let z_i, a_i be the tensors computed by the i^{th} convolutional block in the residual network and the activations obtained after applying ReLu function respectively The output of this i^{th} residual block can be represented as:
$a^{[i+2]} = g^{[i]}(a^{[i]} + z^{[i+2]})$

The tensor z_i in this case is 4D and has size $Ni \times L \times Hi \times Wi$, where N_i is the number of filters used in the i^{th} block. Each filter is 4-dimensional and it has size $Ni_1 \times t \times d \times d$ where t denotes the temporal extent of the filter and d denotes the spatial extent of the filter. The filters are then convolved in 3 dimensions i.e., over both time and space dimensions. The outputs from these convloutional layers are aggregated to the top layer where global average pooling takes place over the entire spatiotemporal volume and the final classification prediction is addressed by a fully connected layer as seen in the Figure 3.3

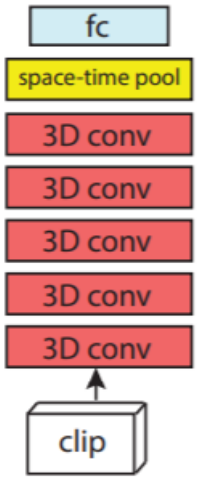

FIGURE 3.3: Block diagram of 3-D ResNet.

Architecture of 3D-ResNet

The figure 3.4 presents the architecture details of the experiments for the two R3D architectures (3D ResNets) considered by the authors. The first one consists of 18 layers and the second one of 34 layers. After feeding in the input we use one spatial down sampling at conv1 implemented by convolutional striding of $1 \times 2 \times 2$, and three spatiotemporal downsampling at conv3 1, conv4 1, and conv5 1 with convolutional striding of $2 \times 2 \times 2$. The output of these operations at last is average pooled over spatial and time dimensions before feeding into a fully connected layer to classify inputs and output the predictions. We have used the 18 layer variant of the ResNet 3D for training as see in figure 3.4. The pre-trained weights trained on the Kinetic-400 data-set , which contains raw, un pre-processed clips can be hard to achieve decent accuracy on, help the model learn better discriminatory features. And using transfer learning we have tried generalising the weights trained on action recognition problem to the accident detection paradigm.

layer name	output size	R3D-18	R3D-34
conv1	L×56×56	3×7×7, 64, stride 1 × 2× 2	
conv2_x	L×56×56	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix} \times 3$
conv3_x	$\frac{L}{2}\times28\times28$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix} \times 4$
conv4_x	$\frac{L}{4}\times14\times14$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix} \times 6$
conv5_x	$\frac{L}{8}\times7\times7$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix} \times 3$
	1×1×1	spatiotemporal pooling, fc layer with softmax	

FIGURE 3.4: Architecture of 3-D ResNet

Implementation problems encountered and solutions

All our extracted input frames are approximately of size $720 \times 1080 \times 3$ but the input size suggested by authors is of size $112 \times 112 \times 3$. Initially we have tried using the input data without resizing the inputs as the model doesn't depend on input image size, due to its penultimate global pooling layer. But due to limited RAM and GPU memory limitations, the script used to crash during training forcing us to resizing our input frames to $200 \times 350 \times 3$ and set the temporal extent of input be limited to 40 frames. Resizing the frames and limiting the the temporal extent of input led us to another minor problem - detail loss in frames which is a compromise we made to train the model with limited resources. Another practical obstacle we faced while implementing the solution is the variable temporal periods of the input video clips. To tackle this problem we have used a novel frame sampling approach of dividing the temporal dimension of clips into equal segments and sampling randomly from the segmented subsections.

3.2.2 2-D Resnets on temporal data

Detection of road accident based on CCTV surveillance videos is mainly done by means of typical steps, which are motion detection, feature extraction, feature analysis, and accident recognition. Being a video based detection system, model has to detect the anomaly behaviour (accident) even for short periods in the video input and report an accident. However, defining such a model on videos to detect an accident makes the model complex and at the same time resource intensive to train it. Since videos can be thought of as a set of individual images,we initially would apply object classification as if it were a sequence of image classifications performed N times in total, where N is the total number of frames in the video. These frames are trained on a 2-D Convolution Network and the model's outputs are then translated to make prediction on the entire video sent in as a series of frames by means of rolling prediction average method discussed in the next section. The main difference though, between convolutions extended to a series of frames in 2D and 3D (as done in the previous model) is that the 2D convolutions provide a single image by applying the same weights to the entire depth of a stack of frames (multiple channels), where as 3D convolutions employ 3D filters to generate a 3D volume as a result of the convolution, retaining the frame stack's temporal detail. A 2-D ConvNet model has to deal with less feature space as compared to 3-D convNets and this makes the model very simple yet powerful to predict the accident.

Architecture and training pipeline

In this model, we will be dividing the video data into contiguous sequence of frames which are then fed as a input to the model, trained using transfer learning on a ResNet model(He et al., 2016). This model consists pre-trained weights of ResNet trained on ImageNet data-set (Deng et al., 2009) and on top of that an extra sigmoid layer is added to give the binary output of 0 or 1. The model is then trained on a 1500 image data-set with labels "accident" and "non-accident". The model takes series of contiguous video frames and predicts accident (1) or no accident (0) as the output. We store the output of each frame in a list and predict the output accident (1) or no accident (0) of the original video. However, we can still face a problem with this method, which is the prediction flickering problem which is discussed in brief in the next section.

Implementation problems encountered and solutions

Prediction flickering: Prediction flickering is a problem occurs when we try to apply simple image classification to video classification.The output of some of the frames in the video differ from the rest even when the whole video is of a single type of activity. For e.g., when we input the accident video, the model takes each frame and outputs the prediction for each frame . Even though the whole video contains just the accident part, sometimes some of the outputs of the frames may display it as non accident due to some error in the prediction by the model, these few frames differ from the rest in their output. This flickering phenomenon in the output is termed as prediction flickering in deep learning. This problem can be solved using rolling prediction average.

Rolling prediction average: Rolling prediction average is a method for removing fine-grained variance between time phases from time series. It is done for the purpose of removing noise and exposing the data of the underlying causal processes. Rolling prediction average smoothens up our data avoiding the flickering in the data. Using Rolling prediction average results us with a far better accurate output. In this method we loop over all sampled frames in the video file and pass them through the model to obtain the predictions. It is necessary to maintain a list of the last K predictions to compute the average of those values and choose the label by clipping the calculated mean to 0 or 1. We then label the frame and write the output frame to disk accordingly.

Finally, we predict the output for video as non accident if rolling prediction average < threshold and accident if the rolling prediction average >= threshold. Since here, we are dealing with a binary classification we can take threshold as 0.5 but in this problem we need our model to have high recall i.e. very few false negatives and a reasonable precision, due to which the threshold can be set to a value less than 0.5.

Chapter 4

Results

4.1 Transfer learning on 3D ConvNet's

Data-set Used

This data-set consists of video clipping extracted from various Youtube videos as seen in figure 4.1. The video data contains two classes of videos (accident/ non accident).This data is pre-processed and reshaped to size 140 and fed in to the 3-D convNets.

FIGURE 4.1: Sample frames from the video data-set

4.1.1 Model summary and inference

The 3D ResNet model is fed in with a video input of size $3 \times L \times H \times W$ and after passing this input through a series of convolution and fully connected layers, the model predicts the binary output of 0 or 1 indicating a "Non Accident" and "Accident". The results of the very few epochs the model has trained on is summarized, in the table 4.1 :

Model- 1 Using 3D ConvNets			
Model	Accuracy	F1 score	Running time (sec)
R2plus1d_18	87.37	0.869	503
Mc3_18	85.35	0.856	495
R3d_18	84.11	0.847	502

TABLE 4.1: Preformance metrics of 3D ResNet architecture.

4.2 2D Resnets on temporal data

Data-set Used

This data-set contains frames captured from Youtube Videos involving accidents as seen in figure 4.2. Data-set is split into 3 folders - train, test and val. Each folder has Accident and Non Accident folders. Consecutive frames of an accident are included in the data-set so the model can learn to differentiate between an accident and non accident.This is the frames image data-set formed from the original video.

FIGURE 4.2: Sample frames from the video data-set

Model summary and inference

The image data i.e. frames of video is sent as an input and the pre-built ResNet model is fine-tuned over this data-set. The output of each frame is stored in a queue and we use rolling prediction average over k previous values to predict the actual output of the current video frame. To assure this model to be compatible with the real world video data, we have built a sampling framework to convert the real world video into sequence of frames and the contiguous frames are sent as input for this model to create a queue and the output resulted from this queue is the desired output for the entirety of the video. A sample output of this model is as seen in figure 4.3. Here we have considered rolling average number k as a hyper-parameter and

found k=5 gives best results on validation set. These performance metrics of the model recorded on test validation data is summarized as seen in Table 4.2 below.

FIGURE 4.3: Sample output of 2D ResNet model

Model- 2 using 2D ConvNets			
Rolling avg number	Accuracy	Precision	Recall
4	72.82	71.56	73.02
5	74.52	72.35	76.11
6	75.58	74.40	76.56
7	76.01	75.49	77.92

TABLE 4.2: Performance metrics of 2D ResNet architecture.

4.3 Comparison of the two models :

Performance: The preceding performance metrics tables clearly illustrate that the pre-trained 3D ResNet model outperforms the 2D ResNet model. The accuracy of 3D ResNets can also be improved by sampling more video frames. The performance difference can be attributed mostly to the 3D ResNet's skip connections, which allow temporal and spatial gradient information to readily back propagate through the 18 layer network. Furthermore, the 3D convolutions spanning space and time allow the network to efficiently model their relationship. 2D ResNets, on the other hand, are more faster and lighter in weight, but they fail to generalise to validation data points due to a poor strategy for fusing spatial and temporal information.
Run-time: While 3D CNNs aggregate temporal and spatial information in the convolution layer, which is very resource expensive to compute, 2D CNNs aggregate the information after outputting the targets using a rolling average across the input frames, making it lighter and faster.

Chapter 5

Conclusion

In this research, we use deep learning-based approaches to detect crashes and accidents. For identifying accidents, two distinct models (3-D convNets and 2-D ConvNets) were examined, assessed, and trained using transfer learning. The first technique, employing 3D ResNets, found to be a better solution due to its better feature aggregation and skip connections across image and time dimensions. This approach had a accuracy rate of 87.5 %. Though 3D ResNets are reasonably accurate, they are very resource expensive networks, and deploying and inferencing the model in real time can be challenging, in contrast to 2D ResNets, which are light weight and fast. The second proposed model was created on top of conventional ResNet utilising transfer learning. The original video data was supplied as a sequence of contiguous frames for the accident prediction model, making it simple and computationally efficient. The main issue encountered while developing this method is prediction flickering. It was solved using a rolling prediction average method. This approach had a hit rate of 76.01 % accuracy.

Interesting findings include: (1) the proposed 3D ResNet model outperformed naive machine learning and basic deep learning methods; (2) unique sampling strategy of video frames made the model resource and computational efficient; (3) the model doesn't perform well in low light, heavy traffic areas, lossy input frames and during bad weather (snow, rain).

Overall, the findings are positive, and the system seems to be viable. There are admittedly, a few issues that can be further addressed. To begin, the challenge of low-light detection can be handled utilising computer vision techniques that improve image quality in low-visibility settings such as night, cloudy, and rainy days. This problem can further be explored by improving the performance of 2D Convolutional networks by employing recurrent neural networks such as LSTMs and transformers instead of rolling prediction average for better temporal and spatial information fusion.

Bibliography

Arceda, V Machaca and E Laura Riveros (2018). "Fast car crash detection in video". In: *2018 XLIV Latin American Computer Conference (CLEI)*. IEEE, pp. 632–637.

Deng, Jia et al. (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Gan, Chuang et al. (2015). "Devnet: A deep event network for multimedia event detection and evidence recounting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2568–2577.

Gu, Xin et al. (2019). "Utilizing UAV video data for in-depth analysis of drivers' crash risk at interchange merging areas". In: *Accident Analysis & Prevention* 123, pp. 159–169.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hu, Jie, Li Shen, and Gang Sun (2018). "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141.

Ijjina, Earnest Paul et al. (2019). "Computer vision-based accident detection in traffic surveillance". In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, pp. 1–6.

Karpathy, Andrej et al. (2014). "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732.

Kay, Will et al. (2017). "The kinetics human action video dataset". In: *arXiv preprint arXiv:1705.06950*.

Ki, Yong-Kul and Dong-Young Lee (2007). "A traffic accident recording and reporting model at intersections". In: *IEEE Transactions on Intelligent Transportation Systems* 8.2, pp. 188–194.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25, pp. 1097–1105.

Lee, Kyu Beom and Hyu Soung Shin (2019). "An application of a deep learning algorithm for automatic detection of unexpected accidents under bad CCTV monitoring conditions in tunnels". In: *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*. IEEE, pp. 7–11.

Lu, Zhenbo et al. (2020). "A new video-based crash detection method: balancing speed and accuracy using a feature fusion deep learning framework". In: *Journal of advanced transportation* 2020.

Shi, Xingjian et al. (2015). "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *arXiv preprint arXiv:1506.04214*.

Singh, Dinesh and Chalavadi Krishna Mohan (2018). "Deep spatio-temporal representation for detection of road accidents using stacked autoencoder". In: *IEEE Transactions on Intelligent Transportation Systems* 20.3, pp. 879–887.

Thomas, Sinnu Susan, Sumana Gupta, and Venkatesh K Subramanian (2017). "Event detection on roads using perceptual video summarization". In: *IEEE Transactions on Intelligent Transportation Systems* 19.9, pp. 2944–2954.

Tran, Du et al. (2018). "A closer look at spatiotemporal convolutions for action recognition". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6450–6459.

Woo, Sanghyun et al. (2018). "Cbam: Convolutional block attention module". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19.