Université de Carthage

Ecole Polytechnique de Tunisie

Engineer Internship Report

Flame detection in surveillance videos

Host organization :

Prepared by :

Siwar MHADHBI

2nd year Engineering Student - SISY

Supervised by :

Mrs. Amel BENAZZA

Professor at SUP'COM, Tunis

Academic Year : 2020 - 2021

Abstract

The threat to human lives and property posed by fires has become increasingly serious. In this respect, computer vision solutions have been developped to early detect fires. Pioneering methods have employed salient features such as color, texture, and motion to characterize the fire regions in the image. These features have been defined explicitly thanks to the expert knowledge. However, with the successul applications involving deep neural architectures, learning the feature directly from a training dataset is very appealing and has motivated several deep learning based methods for fire detection. In this work, we will focus on a very recent method relying on compact deep neural networks proposed by Samarth *et al.*. More precisely, our objective is to improve this method by exploiting the temporal variation of flames using optical flow fields and by combining spatial and temporal approaches.

Keywords : fire detection, flame localization, deep convolutional neural network, deep learning, motion estimation, late fusion.

Acknowledgement

I would like to express my gratefulness to my tutor Prof. Amel BENAZZA, whose guidance, efforts and support during the internship allowed me to push my limits through all the hard situations I have faced especially at the start point.
I would like also to thank all those who contributed to the completion of this internship project.

Contents

List of Figures

List of Tables

General Introduction

Fire accidents are the most commonly occurring disasters nowadays. In fact, these incidents typically result in lot of loss to life and property. In order to mitigate the number of fire accidents, several research attempts have been proposed to design effective fire detection systems with varying degrees of success, which is one way to reduce the damage caused by such accidents.

Current fire detection methods are typically based on physical sensors like thermal detectors, smoke detectors and, flame detectors. However, these sensor-based detection systems are not very reliable for fire location. Furthermore, in very wide areas, they are not appropriate since they should be placed near the fire ignition place. An alternative, which could lead to overcome such drawbacks is to develop visual fire detection approach based on an analysis of the video stream. The visual-based fire detection approach had many advantages in terms of cost, accuracy, robustness, and reliability [8]. Basically, there are two families of visual detector. The first one is concerned with features defined explicitly by the expert. The second category corresponds to features that are learnt through a deep learning of specific deep neural networks. Our internship falls within this context: flame detection by deep learning strategies. More precisely, we have considered the method recently proposed by Samarth *et al.* which employs the binary detection and localization of flames in CCTV footage using deep neural architectures essentially Convolutional Neural Networks (CNN). This method (which we will designate by the reference method) has been found as very competitive among the reported ones as it operates in real time in a frame by frame way with a good accuracy. Our objective is to study if it is possible to take into account the temporal information in order to improve the detection performances.

This report is organized as follows. Chapter 1 is devoted to a general overview of flame detection methods based on computer vision tools. The second chapter is dedicated to a focus on the reference method of Samarth. In Chapter 3, we presents our contribution concerning the flame motion patterns to be exploited. Chapter 4 deals with the analysis of the late fusion between spatial and temporal approaches. Finally, we sum up our work with a conclusion and some perspectives in Section 4.4.

Chapter 1

Brief review of flame detection

Most of the reported fire detectors consist of two stages. The first one aims at extracting salient features that are able to discriminate between fire pixels and the remaining ones. The second stage corresponds to a classifier that provides the label of the current pixel. Therefore, the detector intelligence lies on the feature extraction. It is expected that they are able to characterize the region of interest, robust to lighting variations. Two categories of features are often distinguished: features that are defined explicitly and features that are obtained thanks to an appropriate learning procedure. In what follows, we briefly review the fire detectors that make use of handcrafted features and learnt ones.

1.1 Handcrafted features

Most of the state-of-art Fire Detection methods (FD) involves color, texture and shape characteristics coupled with the temporal dynamics of flames. This is a very delicate and challenging task due to the random variability of color, shape and motion of flame patterns [8].

In [12], a thresholding of color is applied and it is also combined by taking into account the motion [23]. Later work based on vision [18] couples shape and motion by representing the fire region contour as Fourier coefficients whose the temporal change is used as the temporal signatures of fire patterns. A more recent work considers Markov models [27] in order to distinguish the flickering process of the flame from the motion of flame-colored moving objects. Moving object detection has become widely used in video fire detection. The most well-known methods are background subtraction methods introduced in [3, 7, 29, 13], temporal differencing treated in [16], and optical flow analysis mentioned in [14].

1.2 Learnt features

1.2.1 Principle of Deep Learning

Deep learning is a biologically inspired subfield of machine learning. It uses a layered structure of algorithms called *Artificial Neural Networks* and is based on *Feature Learning*. It is capable of automatically discovering the representations needed for feature detection or classification from raw data, which replaces manual feature extraction in handcrafted approaches.

To do so, deep neural networks compose computations performed by many layers. The computations are performed within simple computing units (or neurons) (cf. Appendix A). These latter are connected via weight connectors, calculate the weighted sum of the coming inputs and compute the output using an activation function (eg. Sigmoid, cf. Appendix B). The outputs of layers are interpreted as deep learnt features that reflect the local content of the underlying image.

There are two types of learning : unsupervised and supervised learning. The latter requires an annotated dataset containing a set of examples where each example is a pair consisting of an input object and a desired output value (in our study : the input is an image of a flame scene and the output is its corresponding labeled image). The training of the deep neural network is based on an optimization algorithm which minimizes a specific cost function that measures the error between the prediction and the target called also the ground truth.

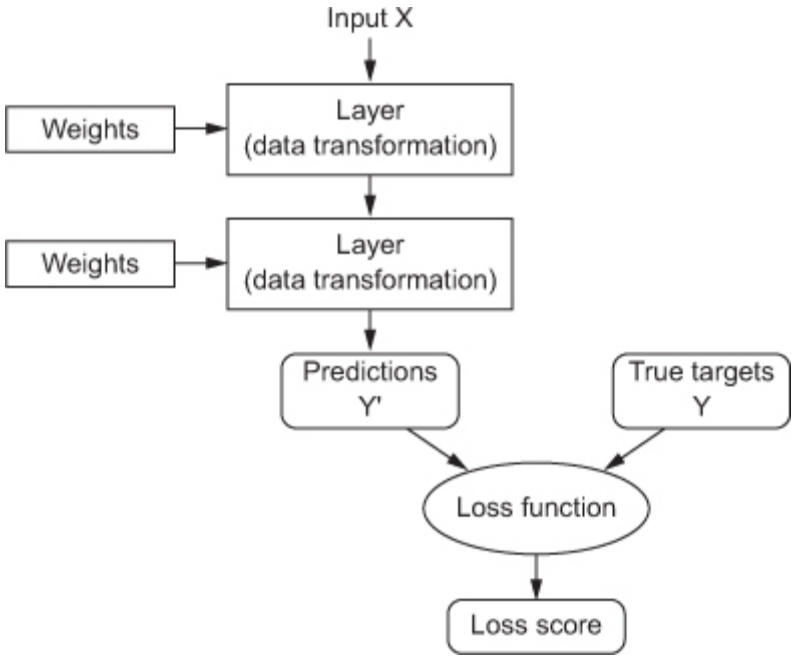

Figure 1.1: Principle of Deep Learning

1.2.2 Deep Learning based approach

Various deep learning approaches for fire detection have been proposed. Pu Li et al. Performed a CNN approach based on the advanced object detection models of Faster-RCNN, R–FCN, SSD, and YOLO v3 [17] which has a strong robustness of detection performance and satisfies the requirements of real-time detection.

Zhang et al. [30] utilized fire patches detection with a fine-tuned pretrained CNN, "AlexNet" [15] in their work on forest fire detection while Sharma et al. [24] proposed a CNN-based fire detection approach using Resnet50 [11] and VGG16 [25] as baseline architectures. Muhammad et al. fine-tuned different models of CNNs like AlexNet [19], SqueezeNet [20], GoogleNet [21], and MobileNetV2 [22].

1.3 Conclusion

The latter features, which are extracted from deep learning models, appeared to be more efficient in the previous works and in literature. Learnt features have generally an impressive performance that makes them generally more preferable than hand-crafted features.

In our work, we'll be basing on learnt features in both spatial and temporal approaches.

Chapter 2

Samarth method

This chapter is dedicated to the flame detection problem based on spatial aspects. Samarth et al. [6] investigated the automatic non-temporal binary fire detection and localization in video within real-time bounds. They explored different Convolutional Neural Network (CNN) architectures spanning FireNet and the Inception architectural concepts.

2.1 Overview of the method

In [6], Samarth et al. define convolutional neural network (CNN) architectures with low complexity such as **FireNet, InceptionV1, InceptionV3** and **InceptionV4** to consider *the first problem of binary fire detection* to determine if fire is present in a particular frame, and **SP-InceptionV1, SP-InceptionV3** and **SP-InceptionV4** to consider *the second problem of in-frame superpixel localization* to determine the precise location of fire within that frame. The SP (SuperPixel) architectures are the same as the first architectures but applied on the superpixels obtained from the semantic segmentation of frames.

2.1.1 Binary detection chain

The chain in Figure 2.1 takes as input a video and splits it into frames. Next, the model proceeds frame by frame; takes one as input to the pretrained architecture among FireNet, InceptionV1, InceptionV3 and InceptionV4 then a softmax layer is applied followed by a threshold equal to 0.5. The final output of this model will be a 2x1 vector of 0 and 1 corresponding to the binary fire detection.

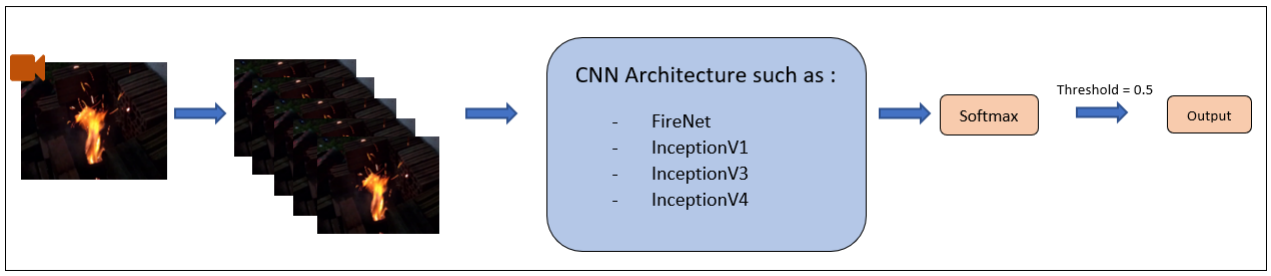

Figure 2.1: Fire binary detection chain

2.1.2 Localization chain

The chain in Figure 2.2 takes as input a video and splits it into frames. Each image frame is split into segments using SLIC superpixel segmentation. Next, a SP-InceptionVx convolutional architecture (for x = 1, 3, and 4 for InceptionV1, InceptionV3, and InceptionV4), trained to detect fire in a given superpixel segment, is used on each superpixel. This superpixel based approach was trained to perform localization within a given frame. The final output of the chain tells if the underlying superpixel belongs or not to the fire region.

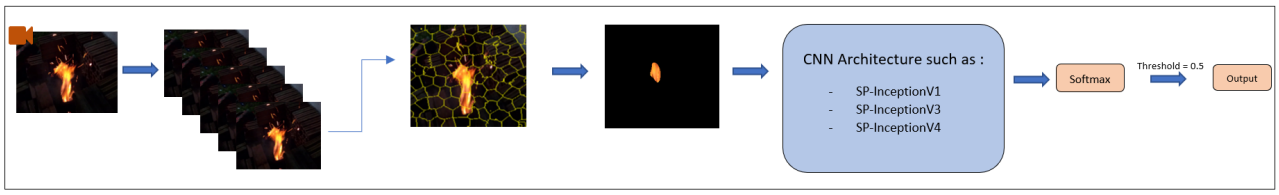

Figure 2.2: Fire localization chain

2.2 Employed CNN architectures

2.2.1 FireNet

The FireNet architecture contains only three convolutional layers of sizes 64, 128, and 256, with kernel filter sizes 5×5, 4×4, and 1×1 respectively. Each convolutional layer is followed by a max pooling layer with local normalization. This set of convolutional layers are followed by two dense layers with tanh() activation. Dropout of 0.5 is applied across these two dense layers to prevent overfitting. Finally we have a fully connected layer and a softmax activation output. The architecture is illustrated in Figure 2.3.

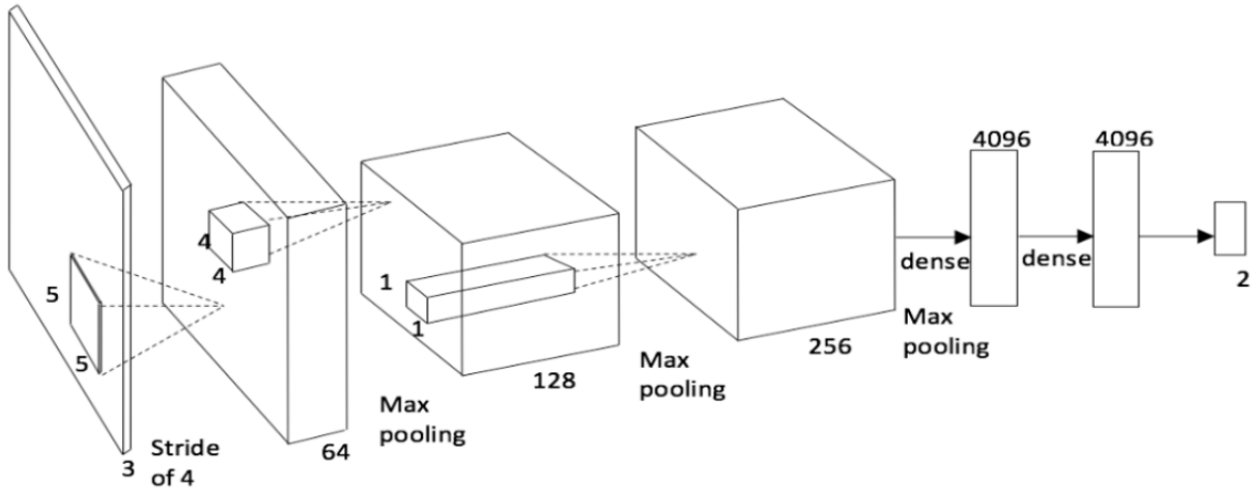

Figure 2.3: FireNet Architecture

2.2.2 InceptionV1

The architecture in Figure 2.4 is basically a convolutional neural network (CNN) with 27 layers. The first convolutions aim to extract the overall characteristics of the fire, followed by max pooling layers mainly to reduce dimensionality. Notice that there is a block called inception module. This is actually the main idea behind the Inception architectures. The inception module is the core concept of a sparsely connected architecture. It is a combination of 1×1 Convolutional layers, 3×3 Convolutional layers, and 5×5 Convolutional layers with their output filter banks concatenated into a single output vector forming the input of the next stage. The main purpose of the inception module is to allow the internal layers to pick and choose which filter size will be relevant to learn the required information. Finally, the last output is fed to a softmax function to detect the presence of fire.

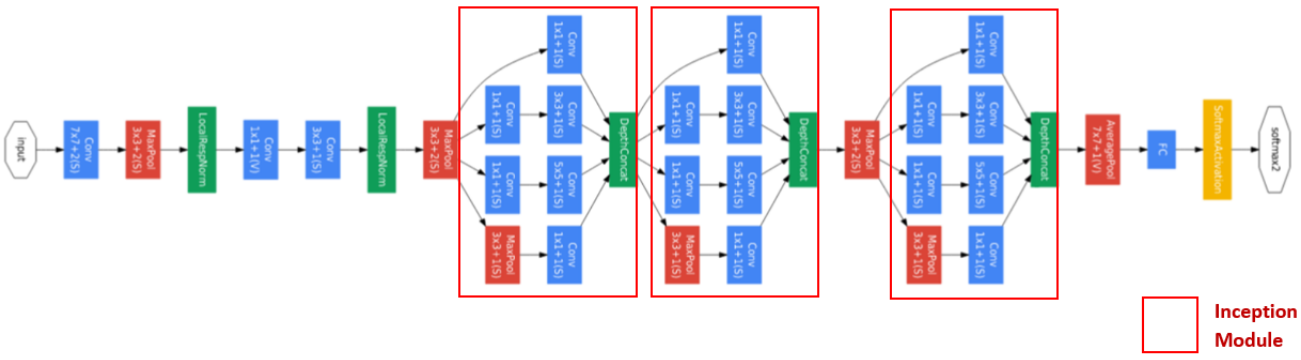

Figure 2.4: InceptionV1 Architecture

2.2.3 InceptionV3

InceptionV3 is a very similar architecture to InceptionV2 which is inspired from InceptionV1 but three different variants of the inception modules are defined (Modules A, B, and C illustrated in Appendix C). They are used mainly to make convolutions more efficient in terms of computational complexity.

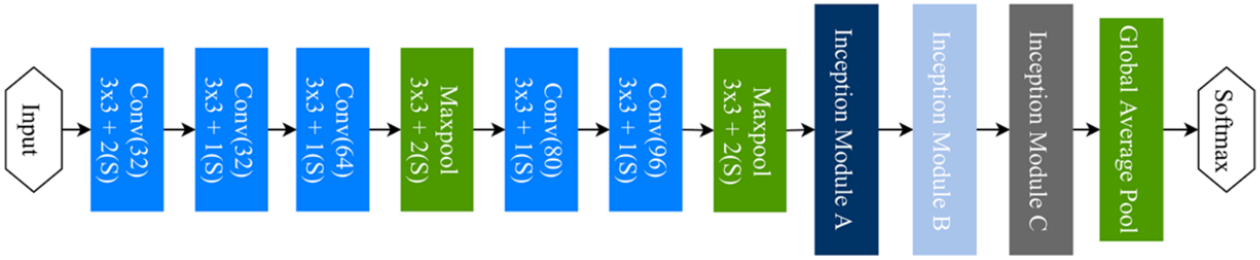

Figure 2.5: InceptionV3 Architecture

2.2.4 InceptionV4

InceptionV4 adds the idea of efficient grid size reduction (Appendix D) to the InceptionV3 architecture in order to reduce the dimensions of the image without losing any noteworthy information before being passed to the inception blocks (Module A, B and C).

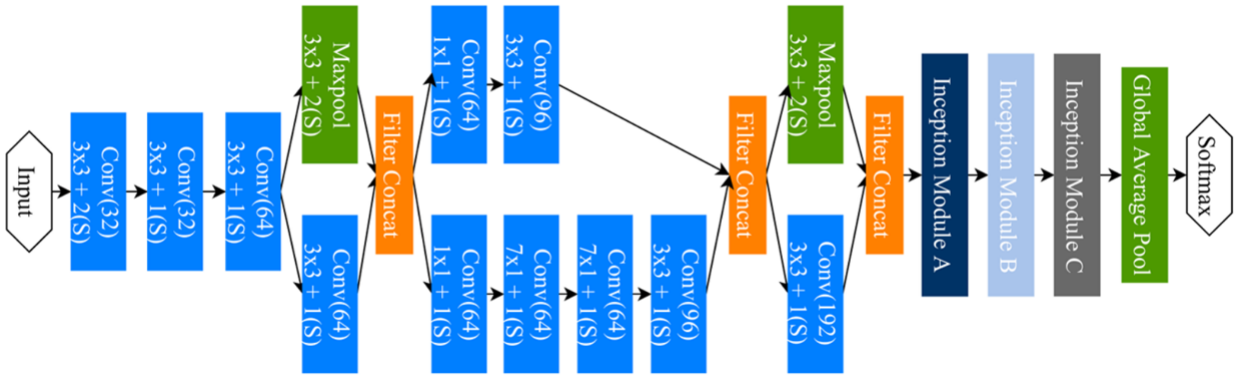

Figure 2.6: InceptionV4 Architecture

2.3 Evaluation of results

2.3.1 Dataset description

The work of [6] used a benchmark dataset to illustrate maximally robust real-time fire region detection. The entire dataset [10] is consisted of a custom dataset and a standard

one (furg fire dataset). The given dataset size is 10.5 GB ; it contains videos, images with and without flames and their corresponding annotations in the form of .txt and .xml files depending on binary classification or localization cases. It's divided into multiple folders in order to specify which images were used for either training or testing. Some examples of the dataset containing and not containing fire are shown in figure 2.7 and figure 2.8.

Figure 2.7: Example images containing flames

Figure 2.8: Example images not containing flames

As for the annotations, Full-Frame Binary Classification annotations are in the form of .txt files showing frame identity and 1 or 0 for each frame containing fire or not respectively. In-Frame Localization annotations are in the form of .xml files containing bounding box coordinates in case of existence of fire.

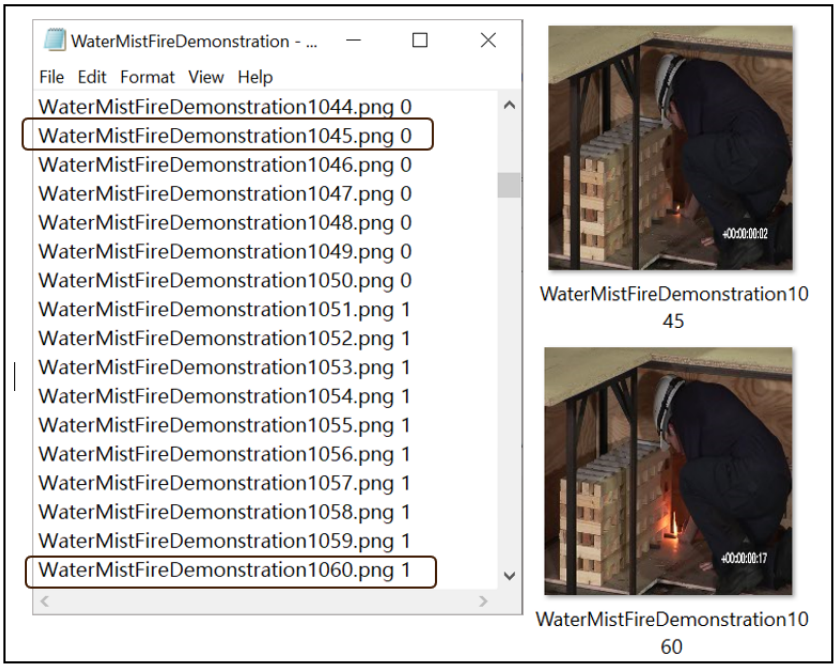

Figure 2.9: Annotations for Full-Frame Binary Classification

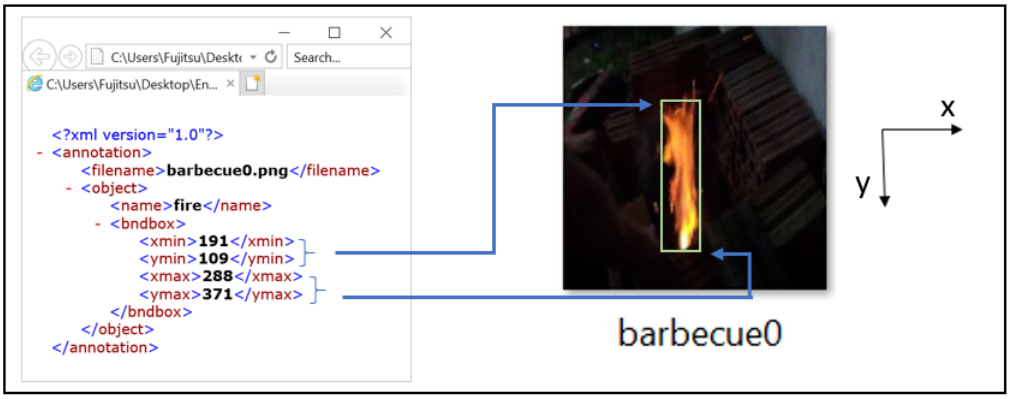

Figure 2.10: Annotations for In-Frame Localization – Fire case

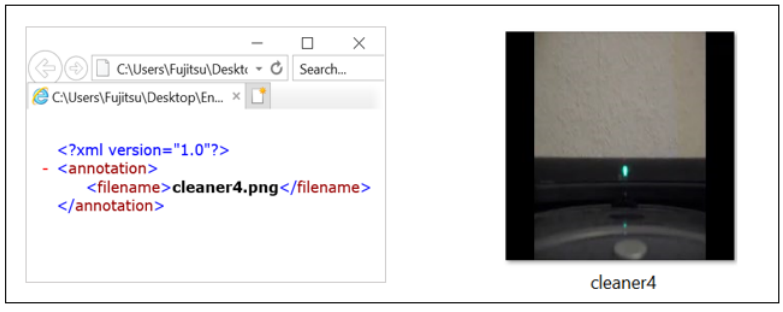

Figure 2.11: Annotations for In-Frame Localization – No Fire case

2.3.2 Inference details

We have installed the necessary framework and debugged the work of the mentioned paper in order to test it on our machine, a Fujitsu Laptop with Intel(R) Core(TM) i5-8350U CPU 1.7GHz/1.8GHz with 8GB of installed memory (RAM), using the command-line.

2.3.3 Results for full-frame binary classification

For the binary fire detection problem, network training and testing are performed on the dataset [10] consisting of 23,408 images. This dataset is split (80:20 split) into two portions for training and validation. We have tested these different algorithms on our machine, and we obtained the results shown in table 2.1. We can notice that InceptionV3 algorithm is the best in terms of compilation/processing time. While the inceptionV4 algorithm takes the longest processing time. On the other hand, it's clear from table 2.2 that inceptionV4 shows a high true positive rate (95 %) and a low false positive rate (4%) which means the best detection performance.

This is said, we can tell that there is a trade-off between performance and time delay related to compilation and processing of networked detection systems.

Architecture	FireNet	InceptionV1	InceptionV3	InceptionV4
Processing Time per frame	39.420ms	33.346 ms	21.347 ms	127.767 ms
Processing Time per video (length: 1min 45s)	2min 04s	1min45s	1min 07s	6min 42s

Table 2.1: Inference Results for Binary Classification

Architecture	TPR	FPR	F1	P	A
FireNet	0.92	0.09	0.93	0.93	0.92
InceptionV1	0.96	0.10	0.94	0.93	0.93
InceptionV3	**0.95**	0.07	0.95	0.95	0.94
InceptionV4	**0.95**	0.04	**0.96**	**0.97**	**0.96**

Table 2.2: Statistical results for Binary Classification [6]

Figure 2.12: Examples Full-Frame Binary Classification

2.3.4 Results for in-frame localization

As for the In-Frame Localization, training and testing are performed on the same dataset described above. The network architectures are trained on a total of 8,635 fire superpixel images, and 10,000 non-fire superpixel images with a test set of 3,000 images containing 1,500 fire and 1,500 non-fire examples. The method adopted consisted on the SLIC (Simple Linear Iterative Clustering) Algorithm which generates superpixels by clustering pixels based on their color similarity and proximity in the image plane, as shown in figure 2.13. Then, the previous algorithms, Inception-V1,-V3, and -V4, are performed on each generated superpixel of the frame in question.

Figure 2.13: Left : Original Image - Right : SLIC application

We have tested the mentioned algorithms on our machine and obtained the results shown in table 2.3. Just like Full-Frame Binary Classification, we can notice that SP-InceptionV3 algorithm is the best in terms of compilation/processing time. While the SP-InceptionV4 algorithm takes the longest processing time but shows the best detection performance like mentioned in table 2.4.

Architecture	SP-InceptionV1	SP-InceptionV3	SP-InceptionV4
Processing Time per frame	33.346 ms	21.347 ms	127.767 ms
Processing Time per superpixel	2s 401ms	1s 265ms	9s 990ms
Number of superpixels per frame	100 superpixels per frame		

Table 2.3: Inference Results for In-frame Localization

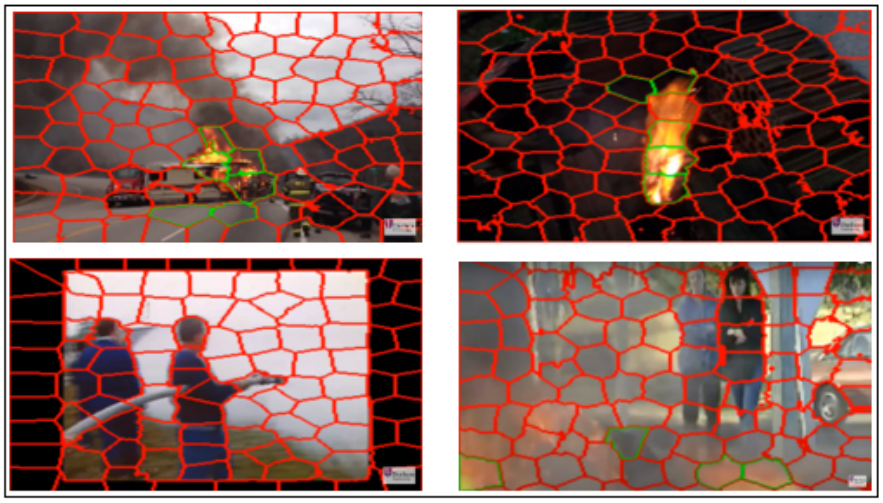

Figure 2.14: In-Frame Localization results using SLIC method [2]

Architecture	TPR	FPR	F	P	A
InceptionV1	0.92	0.17	0.9	0.88	0.89
InceptionV3	0.94	0.07	0.94	0.93	**0.94**
InceptionV4	**0.94**	**0.06**	**0.94**	**0.94**	**0.94**

Table 2.4: Statistical results for In-Frame Localization [6]

2.4 Selected architecture

Inception-V4 algorithm showed a high true positive rate and a low false positive rate in both Full-Frame Binary Detection and In-frame Localization.

As Inception-V4 shows a maximum overall accuracy of 96% and 94% for full-frame binary fire detection and superpixel localization respectively, which means the best detection performance in both cases, we select this algorithm to build on our later work.

2.5 Conclusion

As saw above, the Spatial Approach using InceptionV4 architecture witnessed a robust work on both binary classification and fire localization. However, the use of temporal scene information is missing yet seems interesting since fire flames can be characterized by specific motion patterns. In the following chapter, we aim to study the temporal information in videos that show flames motion looking forward to a better overall performance.

Chapter 3

Proposed temporal approach

In this chapter, we will be considering the temporal approach for fire detection. We will study the motion patterns of flames and analyze their movement and see whether it can be used to distinguish flames from other moving objects.

3.1 Motivation

Videos have much more potential to be mined ; exploring temporal information is important for better video understanding. It is one of the most important cues for machines to perceive the visual world better.

In the case of flame detection in CCTV videos, few efforts are put on the studies regarding to the Temporal Flame Detection [9]. Until now, video-based flame detection remains an open and challenging issue due to the fact that many natural objects have similar colors as those of flames and can often be mistakenly detected. The main challenge that researchers have to face is the complex nature of the fire phenomenon and the large variations over time of flame appearance in videos.

To fill this knowledge gap, we propose a novel framework that relies on temporal information; we implemented a convolutional neural network with reference to the work of [26] whose main idea is to segment all moving objects in videos based on the flow optical field.

3.2 Proposed method : Reference algorithm

The main role of the algorithm corresponding to the temporal approach based on optical flow field in the work of [26], is to capture the independent object motion and segment all moving objects in a given video.

We got inspired by this work in order to implement the architecture in figure 3.1 and build our own model which focuses on only learning flames temporal behavior.

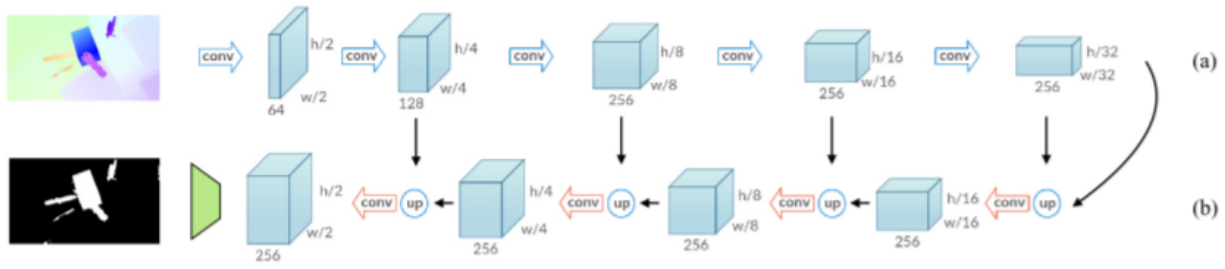

Fig. 3 Our motion pattern network: MP-Net. The blue arrows in the encoder part (a) denote convolutional layers, together with ReLU and max-pooling layers. The red arrows in the decoder part (b) are convolutional layers with ReLU, 'up' denotes 2×2 upsampling of the output of the previous unit. The unit shown in green represents bilinear interpolation of the output of the last decoder unit.

Figure 3.1: MP-Net (Motion Pattern Network) Architecture

3.2.1 Architecture description

The MP-Net architecture is based on optical flow field ; it takes as input optical flow images (upper image in figure 3.1) and learns to extract special temporal features through multiple blocs based on the corresponding labels (lower image in figure 3.1). As the desired output includes localization, the network is trained to predict the class label assigned to each pixel.

The network architecture is illustrated in Figure 3.1. It consists of a contracting path (upper side) and an expansive path (lower side).

The contracting path follows the typical architecture of a convolutional network. It consists of the application of multiple convolutional blocs. Each block consists of two unpadded convolutions, each followed by a rectified RELU and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels.

Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution that halves the number of feature channels which is followed by a concatenation with the correspondingly cropped feature map from the contracting path. In fact, high resolution features from the contracting path are combined with the upsampled output. So that a successive convolution layer can learn to assemble a more precise output based on this information which used for the localization task. As for the cropping, it is necessary due to the loss of border pixels in every convolution. Then, two convolutions layers come next with Batch normalization, each followed by a ReLU.

The final decoder step produces a motion label map at half the original resolution. We perform a bilinear interpolation, the green unit in figure 3.1, on this result to estimate labels at the original resolution.

As a consequence, the expansive path is more or less symmetric to the contracting path and yields a u-shaped architecture. So, the MP-NET in [26] is inspired from U-NET architecture which is a network utilized mainly for biomedicine.

3.2.2 Generate optical flow images

In order to track the motion of flames across video frames, we need the information given within the relationships between consecutive frames, this will be done based on Optical Flow Field.

3.2.2.1 Introduction to optical flow

Optical flow, or motion estimation, is a fundamental method of calculating the motion of image intensities, which may be ascribed to the motion of objects in two consecutive frames. It is defined as the apparent motion of pixels on the image plane as mentioned by [28].

Optical flow knows two main variants : Sparse and Dense. Sparse optical flow gives the flow vectors of some "interesting features" (say few pixels depicting the edges or corners of an object) within the frame. As for Dense optical flow, it gives the flow vectors of the entire frame (all pixels). In our project, we will consider Dense optical flow as it has higher accuracy, although at the cost of being a little more computationally expensive.

3.2.2.2 Theory behind Optical Flow estimation

There are many methods of estimating the optical flow between two frames. The most widely used method is the differential method, which is based on the assumption of image brightness constancy.

According to the brightness constancy assumption, the intensity of the voxel remains the same despite small changes of position and time period.

More specifically,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{3.1}$$

where :

(x,y,t): Voxel position in frame t
$(\delta x, \delta y, \delta z)$: Small change of the movement
I(x,y,t) : Intensity of a voxel at position (x,y) of the frame t

The optical flow equation is given by :

$$\frac{\partial I}{\partial x} Ux + \frac{\partial I}{\partial y} Uy + \frac{\partial I}{\partial t} \approx 0 \tag{3.2}$$

Where :

Ux= $\frac{\partial x}{\partial t}$: Horizontal composant of optical flow of voxel (x,y,t).
Uy= $\frac{\partial y}{\partial t}$: Vertical composant of optical flow of voxel (x,y,t).

3.2.2.3 Generate dense optical flow images

Dense Optical Flow shows arrows in each pixel of a frame that represents the direction and velocity of its movement. Using the Color Map shown in figure 3.2, we can represent optical flow as images that will be fed to our MP-Net in order to learn high-level temporal features for flame motion recognition, see examples of generated images in figure 3.3.

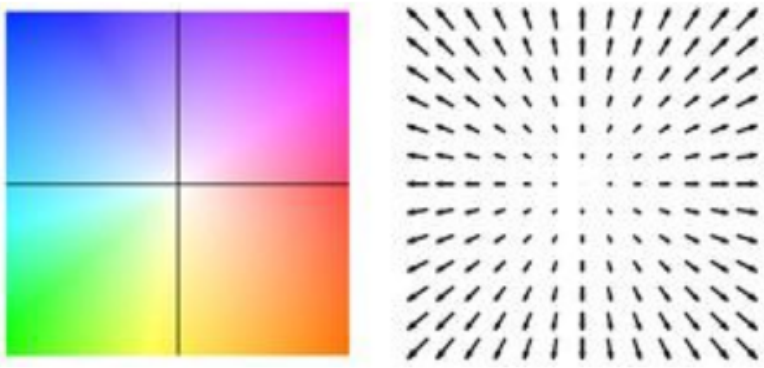

Figure 3.2: Color Map

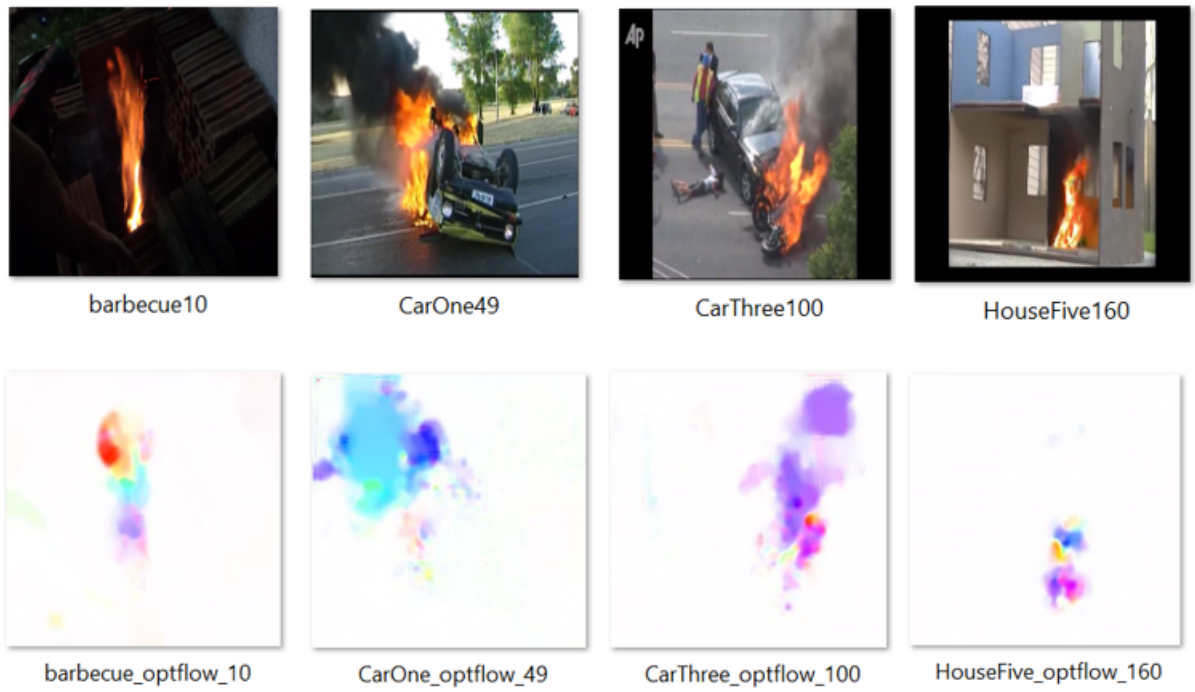

Figure 3.3: Examples Generated Optical Flow Images

3.2.3 Generate ground truth images

For our internship, we will be working on the dataset of the SOTA (state of the art) project. As our network goal is to localize flames position in images in case of existence, the ground truth for our algorithm will be a black and wight image where white pixels represent flame position and black pixels represent background and other objects. In other words, if a pixel corresponds to flame position, it is white. Otherwise, it is black. To generate these labels, we utilize the annotations provided by the SOTA dataset which are bounding box coordinates. Thus, given the coordinates of two points of the bounding box, we are able to localize the flames position as shown in figure 3.4.

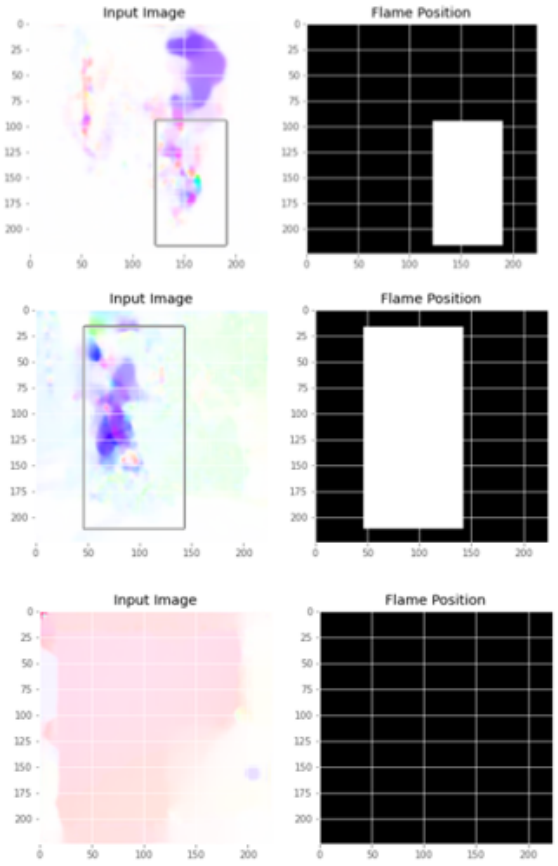

Figure 3.4: Input Images with the associated Ground Truth

3.2.4 Training

We implemented the MP-Net architecture from scratch on Google Colab using GPU runtime with 25.51 GB of RAM and 68.40 GB of Disk allocated. We trained our network on 12 videos of the SOTA dataset specified in table 3.1. We have about 10 343 frames and about 10 331 generated Optical Flow images where 6007 frames with flames and 4324 frames without flames.

We have generated the optical flow images using Matlab Code [1, 4, 5] . And it took 13 seconds per 2 consecutive frames, say 2 hours and a half for a 700-frame video on our machine. We used these optical flow images for training and validation tasks of our network MP-Net.

We use a batch size of 16, the maximum possible due to GPU memory constraints. The model is trained from scratch for 50 epochs with pixel-wise cross-entropy loss. Batch normalization is applied to all the convolutional layers of the network in order to speed up learning.

Videos	*Frames*
Barbecue.mp4	748
CarOne.mp4	580
CarTwo.mp4	750
CarThree.mp4	765
$Case2_house.mp4$	590
$Case2_car.mp4$	455
HouseSix.mp4	355
HouseOne.mp4	470
HouseFour.mp4	800
HouseFive.mp4	830
cleaner.mp4	2000
coolerbot.mp4	2000

Table 3.1: Training videos from SOTA Dataset

3.2.5 Performance evaluation

Performance evaluations must be fairly considered to evaluate the models we have suggested and created by utilizing standard evaluation form for each model. The performance of the models is measured using confusion matrix, accuracy, precision, recall, and F1-Score.

- Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The example of confusion matrix for a binary classifier is shown in figure 3.5.

	Predicted: NO	**Predicted: YES**
Actual: NO		
Actual: YES		

Figure 3.5: Confusion Matrix for binary classification

- Accuracy

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. It is given by the equation (3.3) :

$$Accuracy = \frac{Numberofcorrectpredictions}{Totalnumberofpredictions} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

- Precision

Precision is the fraction of relevant instances among the retrieved instances. It is given by the formula (3.4) :

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

- Recall

Recall is the fraction of the total amount of relevant instances that were actually retrieved. It is given by the formula (3.5) :

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

Both Precision and Recall are therefore based on an understanding and measure of relevance.

- F1-score

The F1-score is a measure of a test's accuracy. IT is the harmonic mean of the precision and recall. The highest possible value of F1 is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero.

It is given by the formula (3.6) :

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.6}$$

3.2.6 Experiments

3.2.6.1 Train results

The network performs reasonably on the training dataset, as it learns to correlate flame appearance with its motion. The results of the training dataset are shown in figure 3.6 and figure 3.7. Figure 3.6 represents the variation of loss function through epochs and figure 3.7 represents the variation of accuracy through epochs. It is very clear that loss values are very low whereas accuracy values are very high. The best model is reached in epoch 3 as mentioned by figure 3.6. The loss of best model is about 5% and its accuracy is about 97%. This means, our model learns very well the training dataset.

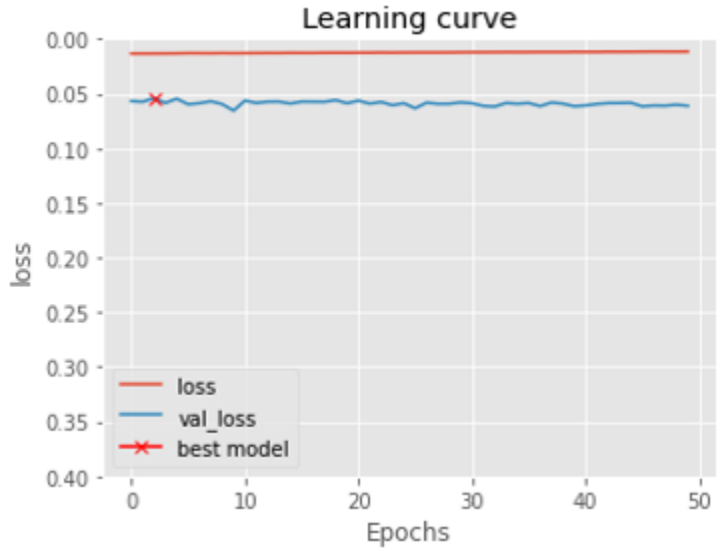

Figure 3.6: Loss variations

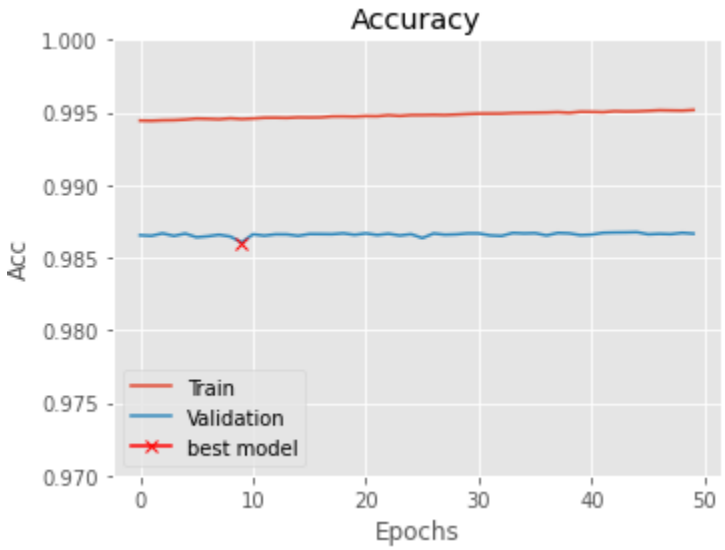

Figure 3.7: Accuracy variations

3.2.6.2 Test Results

For the test dataset, we generated the optical flow images of 6 videos from the SOTA data, say 3601 images with and without flames, i.e 30% of the data available.

The confusion matrix represented in figure 3.8, shows that 36% of the data are predicted correctly negative, and 26% are predicted correctly positive. Whereas, 12% are

missing cases and also 26% of the data are predicted positive but actually they are negative, which means that the false alarm rate is high with this model. Thus, its accuracy would not be high as training results.

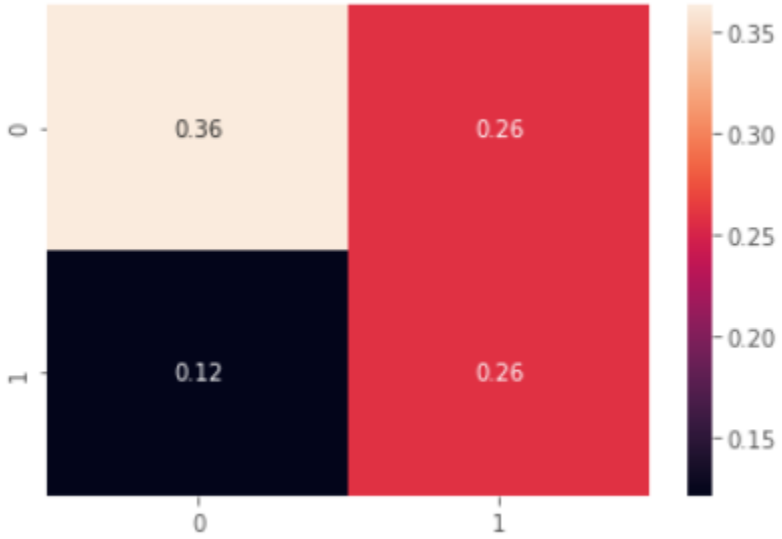

Figure 3.8: Test Confusion Matrix

The statistical results are given by the table 3.2. In order to obtain a well performed model, the recall and precision should be both high. However, in our case the recall is higher than precision which proves the high rate of false alarms.

Statistical Metrics	*Values*
Accuracy	62 %
Precision	50 %
Recall	68 %
F1 score	57 %

Table 3.2: Statistical results for test

3.2.7 Observations

Comparing train and test results, we can say that our algorithm models the training data too well. It learns the detail and noise in the training data as train accuracy is about 97% but this is to the extent that it negatively impacts the performance of the model on new data as test accuracy is around 62%. This refers to the problem of overfitting. The model now is not able to generalize on unseen datasets.

3.3 Modified architectures and results

Our model success was limited due to the size of the available training sets. In order to fix the problem of overfitting, we can either increase the dataset size and its quality or reduce the complexity of our model.

As the generation of the training dataset, i.e the generation of optical flow images takes a lot of computational time and as we are limited to the SOTA dataset on which we are working, we chose to modify the architecture of the MP-Net in order to reduce its complexity in such a way it works well with the available training data.

3.3.1 Proposed architectures

We propose two architectures. In fact, the initial one presents 4 blocs of convolutional layers referring to Downsumpling in the encoder part and other 4 blocs referring to Upsampling in the decoder part. The first proposed architecture removes one bloc from the encoder and the decoder parts as shown in figure 3.9. As for the second architecture, it removes two blocs from each part as shown in figure 3.10.

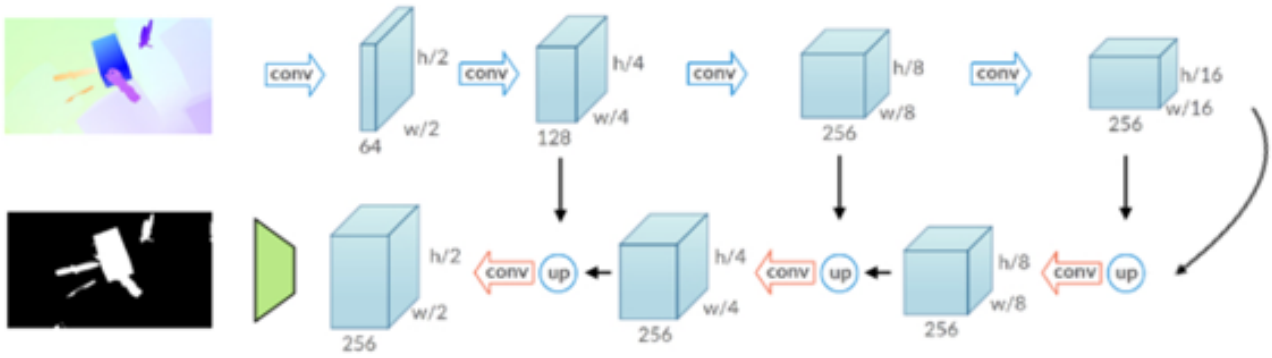

Figure 3.9: Proposed Architecture 1

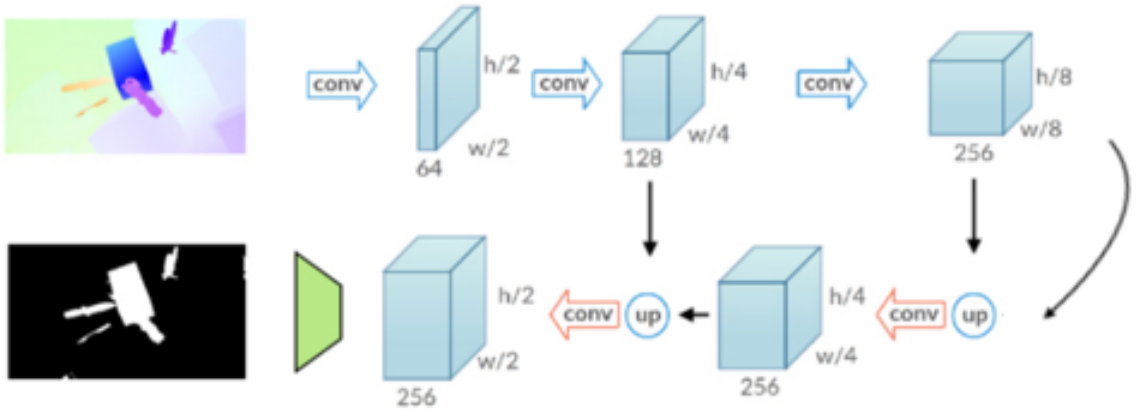

Figure 3.10: Proposed Architecture 2

3.3.2 Results

After training the above architectures on the given dataset and testing it on the test dataset, we obtained the results summarized in tables 3.5 and 3.6 which represents the number of layers, the number of total parameters and statistical results such as TP, FP, accuracy, recall and precision. Now, we will see in details the results obtained for both of Model1 and Model2.

3.3.2.1 Model 1

We implemented the MP-Net architecture in figure 3.9 from scratch on Google Colab using GPU runtime. We trained our network on the same dataset specified in table 3.1. We use a batch size of 16 and 50 epochs with pixel-wise cross-entropy loss.

The results of the training dataset are shown in figure 3.11, which represents the variation of loss function through epochs, and figure 3.12 which represents the variation of accuracy through epochs. It is very clear that loss values are very low whereas accuracy values are very high. The best model is reached at epoch 39 as mentioned by figure 3.11.

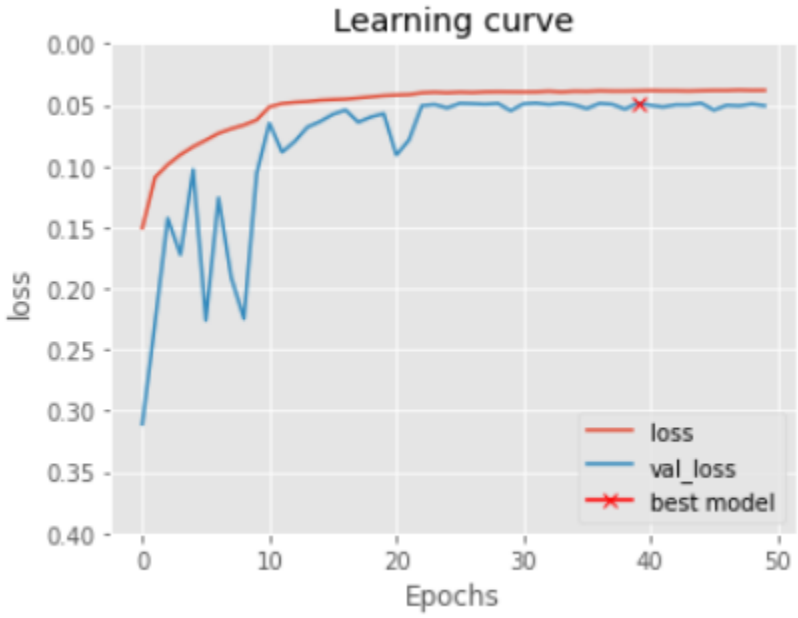

Figure 3.11: Loss Variations

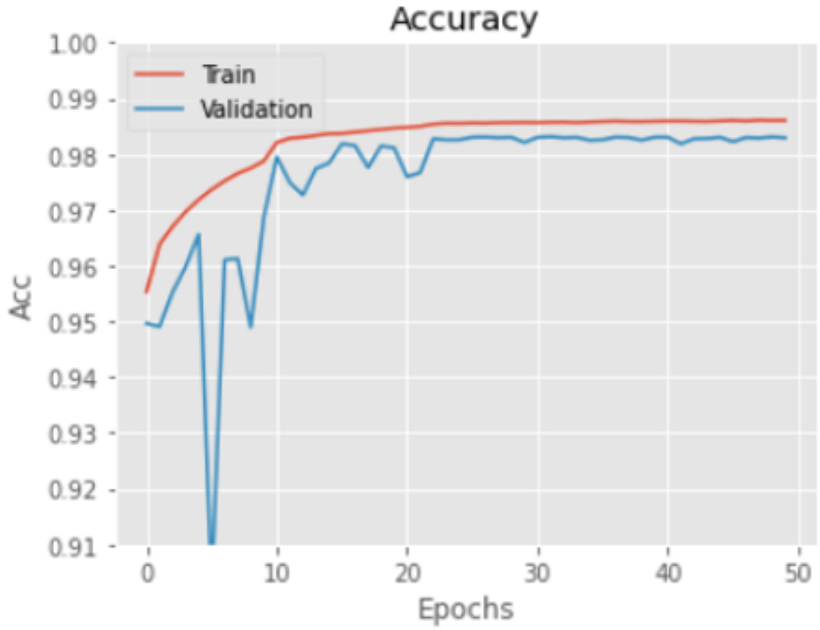

Figure 3.12: Accuracy Variations

The confusion matrix in figure 3.13 shows a high rate of false alarms (39%) but low missing rate(6%). And the rest of the data (55%) is predicted correctly.

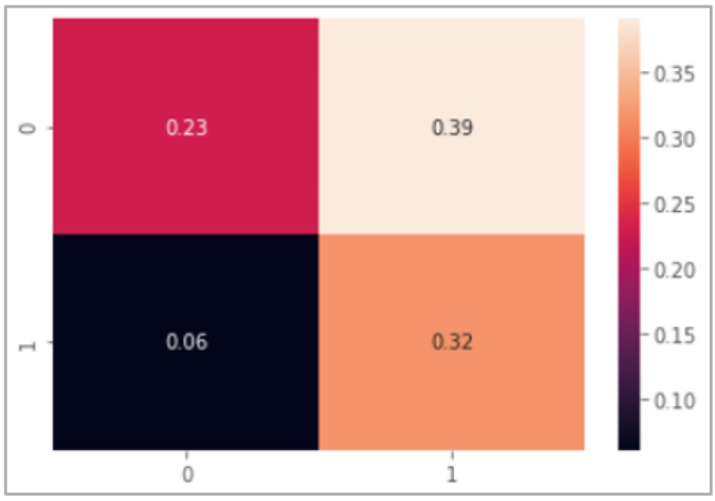

Figure 3.13: Confusion Matrix Proposed Architecture 1

The statistical results are given by the table 3.3. The test accuracy indicated (54%) is very low comparing to train accuracy (88%). Besides, Recall is very high, and Precision is very low which explains the high rate of false alarms. Thus, we can tell that this model does not work well on our dataset.

Statistical Metrics	*Values*
Accuracy	54 %
Precision	44 %
Recall	83 %
F1 score	58 %
ROC AUC score	60 %

Table 3.3: Statistical Results of Proposed Architecture 1

3.3.2.2 Model 2

We implemented the MP-Net architecture in figure 3.10 from scratch in the same framework. We trained our network on the same dataset and used a batch size of 16 and 10 epochs with pixel-wise cross-entropy loss.

The results of the training dataset are shown in figure 3.14 and Figure 3.15 which represent respectively the variation of loss function and the variation of accuracy through epochs. The best model is reached at epoch 3 as mentioned by figure 3.14 .

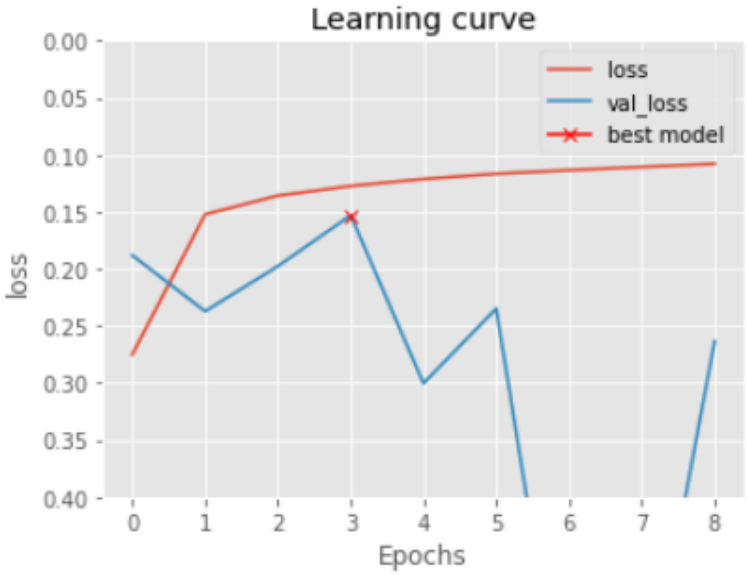

Figure 3.14: Loss Variations

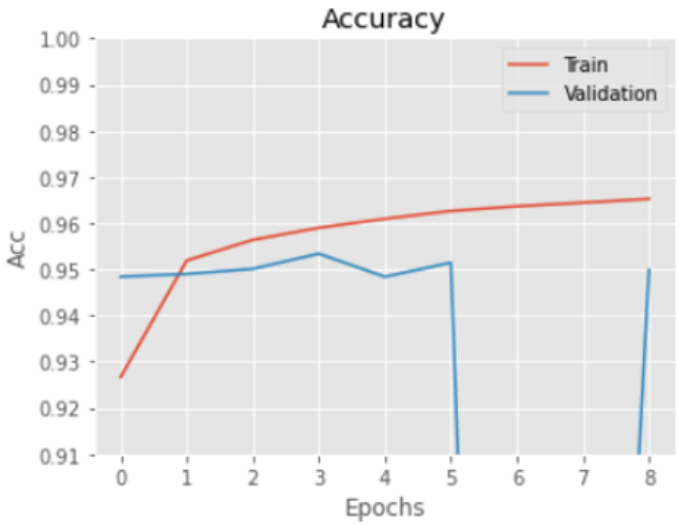

Figure 3.15: Accuracy Variations

The confusion matrix in figure 3.16 shows 28% of false alarms, a very low missing rate (4%) and 68% of our data are correctly predicted, either negative or positive cases.

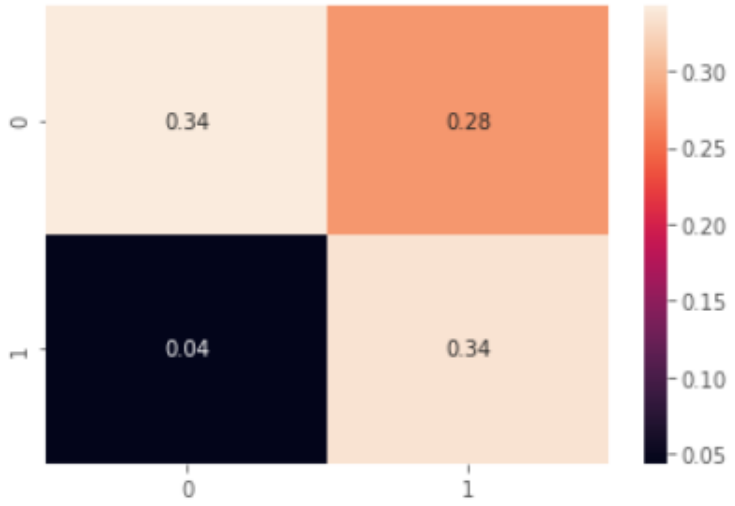

Figure 3.16: Confusion Matrix Proposed Architecture 2

The statistical results are given by the table 3.4. The accuracy indicated is close to train accuracy (74%). Besides, Recall is high. Precision is not very high, which explains the fact that the rate of false alarms isnât very low, but is the highest among the above models. Consequently, we can say that this last architecture can be a good solution for our problem.

Statistical Metrics	*Values*
Accuracy	67 %
Precision	54 %
Recall	88 %
F1 score	67 %
ROC AUC score	71 %

Table 3.4: Statistical Results of Proposed Architecture 2

3.3.3 Recap and Selected Architecture

To summarize, all the parameters and metrics of the three mentioned architectures are given by the table 3.5. As said, based on the available dataset, an architecture with reduced complexity should be more convenient for our problem. Model 2 represents 36 hidden layers and 796 593 as number of total parameters, i.e. with less complexity comparing to other models. It showed a good work vis-à-vis both the train and test datasets as it gave 74% as train accuracy and 67% as test accuracy. So, this model is the least one in terms of overfitting. Thus, it should be the one to work on for the next part of the study.

Figures 3.17, 3.18 and 3.19 illustrate the results of our selected model. We can visualize how much our model manages to localize flame positions. Good predictions, Missing cases and False alarm cases are shown respectively in Figure 3.17, Figure 3.18 and Figure 3.19.

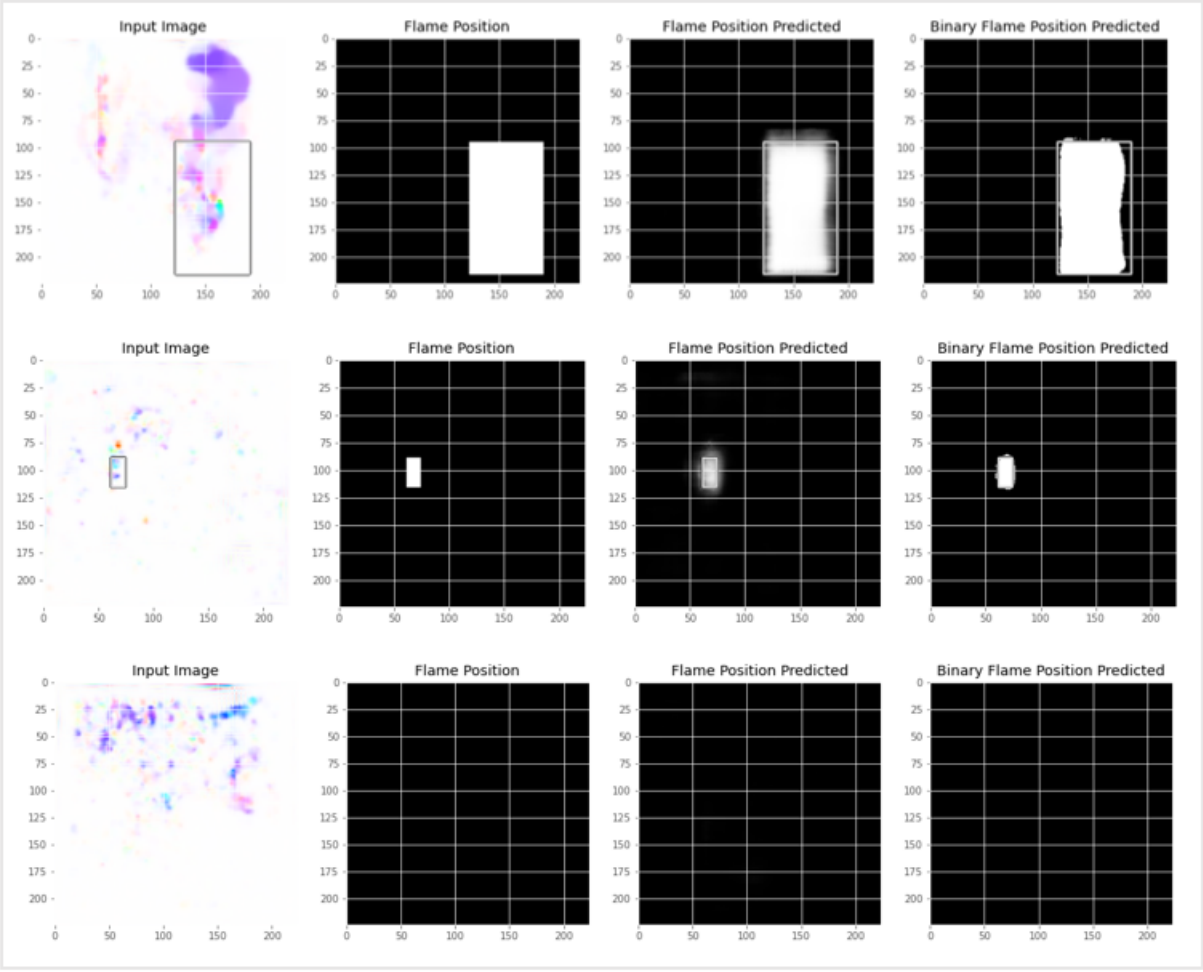

Figure 3.17: Examples of good predictions

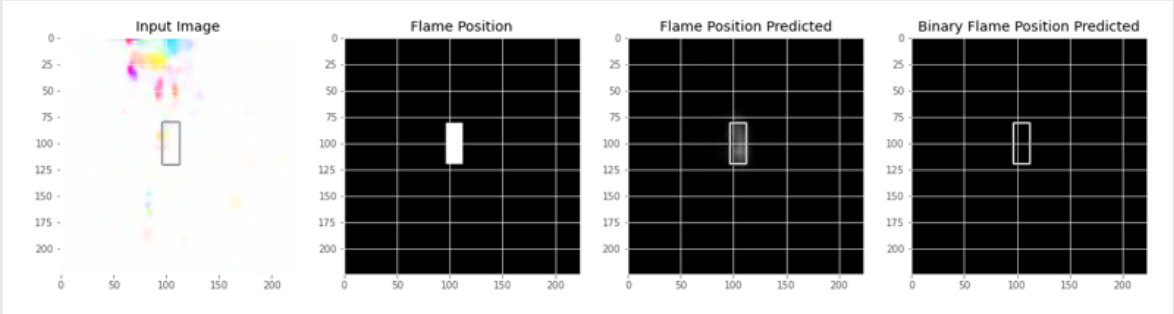

Figure 3.18: Example of a Missing case

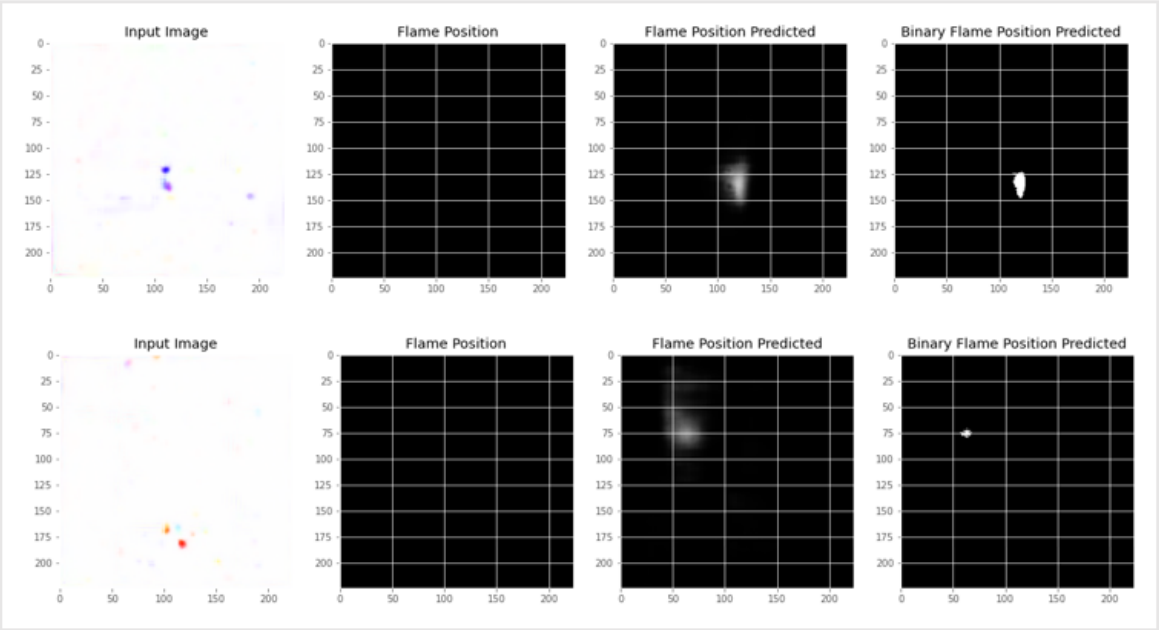

Figure 3.19: Examples of False alarms

	Layers number	Total parameters number	Train results				
			TP	FP	Accuracy	Recall	Precision
Model 0	1 input 66 hidden 1 output	2 164 593	57%	0.9%	98%	98%	98%
Model 1	1 input 51 hidden 1 output	1 129 713	56%	9%	88%	96%	85%
Model 2	1 input 36 hidden 1 output	796 593	46%	14%	74%	79%	76%

Table 3.5: Summary Results - Train

	Layers number	Total parameters number	Test results				
			TP	FP	Accuracy	Recall	Precision
Model 0	1 input 66 hidden 1 output	2 164 593	26%	26%	62%	68%	50%
Model 1	1 input 51 hidden 1 output	1 129 713	32%	39%	54%	83%	45%
Model 2	1 input 36 hidden 1 output	796 593	34%	28%	67%	88%	54%

Table 3.6: Summary Results - Test

3.4 Conclusion

To sum up, in this chapter we studied the temporal behavior of flames and analyzed their variations along time using Optical Flow field. After that, we implemented a CNN architecture that studies the optical flow images generated. According to the results we obtained, it is efficient to detect flames and their positions in footage videos based only on their temporal aspect. The question now is whether by combining spatial and temporal approaches we would improve the SOTA results or not. That is what we will analyze in the following chapter.

Chapter 4

Late fusion

4.1 Introduction

In the previous sections, we studied each of spatial and temporal aspects of flames apart. In this section, we aim to combine both of these approaches looking for a better overall performance in fire detection task. In fact, there are different fusion schemes that have done good work in several applications.

4.2 Models combination

Combining multimodal information is an important issue in pattern recognition. Indeed, the fusion of multimodal inputs can bring complementary information from various sources, useful for improving the quality of the final decision. In this work, we focus on multimodal fusion for fire detection. The different modalities correspond generally to a relevant set of features. Once these features have been extracted, another step consists in building voters, or classifiers, able to discriminate a given concept. In this context, two kinds of fusion schemes are generally considered : the early fusion and the late fusion. On the one hand, in the early fusion approach, all the available features are merged into one feature vector before the learning and classification steps. However, this kind of approach has to deal with many heterogeneous features which are sometimes hard to combine. On the other hand, the late fusion works at the decision level by combining the prediction scores available for each modality.

4.2.1 Late classifier fusion

In our work, we will focus on late classifier fusion which focuses on the individual strength of modalities. In fact, it indicates combining the different classifier scores into

one classification value. This process predicts the final output by considering soft level fusion, i.e. the individual scores of the involved classifiers.

As we have only two classifiers, a Convolutional Neural Network (Inception-V4) and an Encoder-Decoder Network (MP-Net), we used the Average decision rule (class label with the highest averaged confidence).

The scheme for late fusion is illustrated in Figure 4.1.

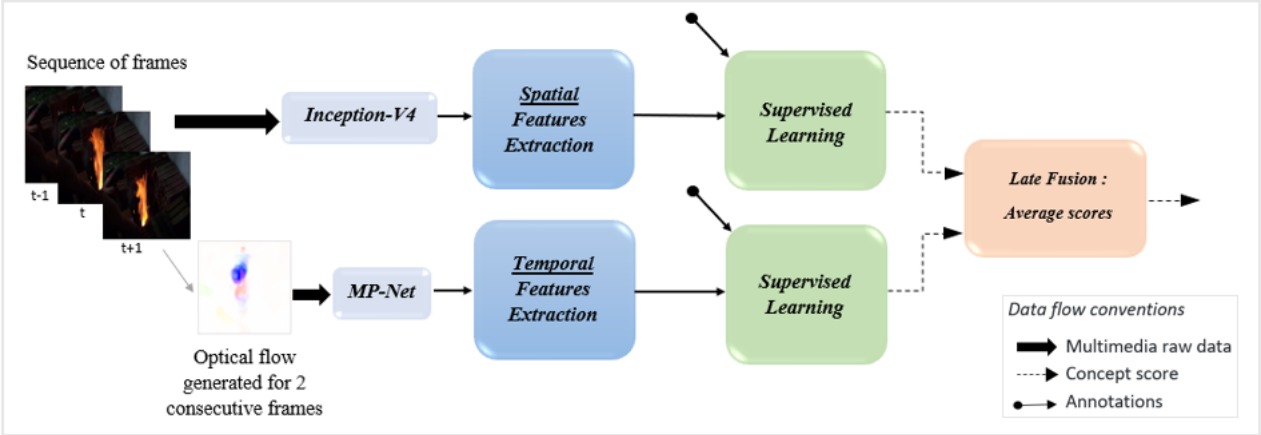

Figure 4.1: Scheme for late fusion

We perform feature extraction on the spatial and temporal modality. Then, we combine features into a multimodal representation using a late fusion. The spatial feature extraction is based on the method described in chapter 2 and the temporal feature extraction is based on the method described in chapter 3.

The late fusion function contains the function that combines the probabilistic output score after spatial analysis with the probabilistic score resulting from temporal analysis. The combination considers the Average of the given predictions. The equation of Average decision rule is given by equation (4.1).

$$Pred^c = \frac{Pred_{Incep} + Pred_{MP-Net}}{2} \tag{4.1}$$

Where :

$Pred^c$ is the prediction probability of late fusion classification
$Pred_{Incep}$ is the prediction probability of classification using Inception-V4 classifier
$Pred_{MP-Net}$is the prediction probability of classification using MP-Net classifier

4.3 Experiments and results

We experienced the fusion methods on frames of the same test dataset. Then, we applied the Average decision rule. This late fusion strategy gave us good results, similar to the Spatial approach results using Inception-V4. The confusion matrix resulted is shown in figure 4.2.

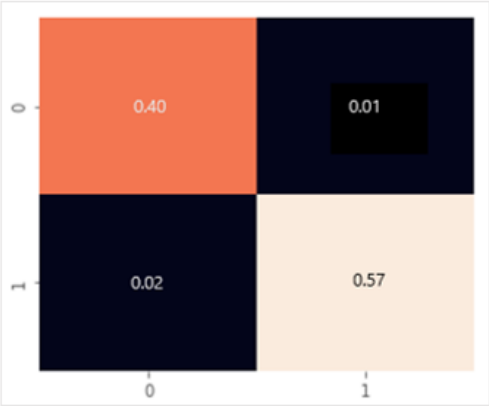

Figure 4.2: Confusion Matrix of late fusion

We can clearly notice that we had very good results as percentages of correctly predicted cases are high (57% and 40%) and we have low rates of false alarms (1%) and missing cases (2%). The accuracy is approximately 97% which is an improvement of almost 1% comparing to the state-of-the-art technique. This is said, we can tell that the supplementary information given by the temporal approach to take into account temporal variation of flames, even though it is slightly, but it did improve the overall performance of flames detection in videos.

4.4 Conclusion

In this chapter, we considered the two aspects of flames : spatial and temporal. As we have only two classifiers, our study is based on output probabilities and not labels. We implemented a soft voting decision rule which is Average rule and analyzed its results. The Average method shows slightly better results than both classifiers apart. Thus, Late Fusion approach has slightly outperformed the state-of-the-art technique.

Conclusion and Perspectives

During this internship, we implemented several algorithms in order to both detect and localize flames in footage videos. This report has first cited related works based on both machine learning and deep learning approaches. Afterwards, we implemented the state-of-the-art technique which showed robust work in analyzing the spatial aspects of flames. In another section, we analyzed the temporal behavior of flames by implementing several versions of MP-Net architectures and selected the best model that matches the available data. This model did not give very good results as the state-of-the-art technique. In fact, the lack of a sufficiently large labeled and high-quality data is one of the main reasons.

As an extension to prior work in the field, we thought about fusion between spatial and temporal aspects of flames. As we have implemented two classifiers separately, we combined both spatial and temporal aspects of flames using late fusion strategy and we got good results comparing to other works in the field. Well, there are some obvious obstacles that late fusion schemes suffer from. In fact, it increases the computational time required as every classifier requires a separate supervised learning stage, which leads to not considering correlations between models in the classifier outputs. There is another approach of fusion, which is early fusion. This strategy combines features beforehand into a multimodal representation. So that it considers the correlations between the inputs which are spatial and temporal features. Thus, early fusion can be evaluated in a future research. Therefore, further improvements could be achieved by using the appropriate features from the early fusion strategy.

This internship has made a potential effort to find a procedure that should be able to detect and localize flames relying on both spatial and temporal aspects of flames.

Bibliography

[1] *Matlab : Code to generate Dense Optical Flow*. https://www.cs.cmu.edu/~katef/LDOF.html.

[2] *Results of Convolutional Neural Networks for Real-time Fire Detection*. https://www.youtube.com/watch?v=RcNj8aMDer4&feature=youtu.be.

[3] Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49–58, 2006.

[4] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[5] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010.

[6] Ganesh Samarth CA, Neelanjan Bhowmik, and Toby P Breckon. Experimental exploration of compact convolutional neural network architectures for non-temporal real-time fire detection. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 653–658. IEEE, 2019.

[7] Simone Calderara, Paolo Piccinini, and Rita Cucchiara. Smoke detection in video surveillance: a mog model in the wavelet domain. In *International conference on computer vision systems*, pages 119–128. Springer, 2008.

[8] A. Enis Cetin, Kosmas Dimitropoulos, Benedict Gouverneur, Nikos Grammalidis, Osman Gunay, Y. Hakan Habiboglu, B. Ugur Toeyin, and Steven Verstockt. Video fire detection review. *Digital Signal Processing*, 23:1827 – 1843, 2013.

[9] Kosmas Dimitropoulos, Filareti Tsalakanidou, and Nikos Grammalidis. Flame detection for video-based early fire warning systems and 3d visualization of fire propagation. In *13th IASTED International Conference on Computer Graphics and Imaging (CGIM 2012), Crete, Greece*, 2012.

[10] Andrew J Dunnings and Toby P Breckon. Experimentally defined convolutional neural network architecture variants for non-temporal real-time fire detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1558–1562. IEEE, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] G. Healey, D. Slater, T. Lin, B. Drda, and A. Goedeke. A system for real-time fire detection. In *Proc. Int. Conf. Comp. Vis. and Pat. Rec.*, pages 605–606, 1993.

[13] ByoungChul Ko, Kwang-Ho Cheong, and Jae-Yeal Nam. Early fire detection algorithm based on irregular patterns of flames and hierarchical bayesian networks. *Fire safety journal*, 45(4):262–270, 2010.

[14] Ivan Kolesov, Peter Karasev, Allen Tannenbaum, and Eldad Haber. Fire and smoke detection in video with optimal mass transport based optical flow and neural networks. In *2010 IEEE International Conference on Image Processing*, pages 761–764. IEEE, 2010.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[16] Byoungmoo Lee and Dongil Han. Real-time fire detection using camera sequence image in tunnel environment. In *International Conference on Intelligent Computing*, pages 1209–1220. Springer, 2007.

[17] Pu Li and Wangda Zhao. Image fire detection algorithms based on convolutional neural networks. *Case Studies in Thermal Engineering*, page 100625, 2020.

[18] Che-Bin Liu and Narendra Ahuja. Vision based fire detection. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 4, pages 134–137. IEEE, 2004.

[19] Khan Muhammad, Jamil Ahmad, and Sung Wook Baik. Early fire detection using convolutional neural networks during surveillance for effective disaster management. *Neurocomputing*, 288:30–42, 2018.

[20] Khan Muhammad, Jamil Ahmad, Zhihan Lv, Paolo Bellavista, Po Yang, and Sung Wook Baik. Efficient deep cnn-based fire detection and localization in video surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(7):1419–1434, 2018.

[21] Khan Muhammad, Jamil Ahmad, Irfan Mehmood, Seungmin Rho, and Sung Wook Baik. Convolutional neural networks based fire detection in surveillance videos. *IEEE Access*, 6:18174–18183, 2018.

[22] Khan Muhammad, Salman Khan, Mohamed Elhoseny, Syed Hassan Ahmed, and Sung Wook Baik. Efficient fire detection for uncertain surveillance environment. *IEEE Transactions on Industrial Informatics*, 15(5):3113–3122, 2019.

[23] W. Phillips, M. Shah, and N. da Vitoria Lobo. Flame recognition in video. *Pattern Recognition Letters*, 23(1-3):319327, 2002.

[24] Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin, and Jahn Thomas Fidje. Deep convolutional neural networks for fire detection in images. In *International Conference on Engineering Applications of Neural Networks*, pages 183–193. Springer, 2017.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Pavel Tokmakov, Cordelia Schmid, and Karteek Alahari. Learning to segment moving objects. *International Journal of Computer Vision*, 127(3):282–301, 2019.

[27] B Ugur Toreyin, Yigithan Dedeoglu, and A Enis Cetin. Flame detection in video using hidden markov models. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–1230. IEEE, 2005.

[28] Pavan Turaga, Rama Chellappa, and Ashok Veeraraghavan. Advances in video-based human activity analysis: challenges and approaches. In *Advances in Computers*, volume 80, pages 237–290. Elsevier, 2010.

[29] Feiniu Yuan. A fast accumulative motion orientation model based on integral image for video smoke detection. *Pattern Recognition Letters*, 29(7):925–932, 2008.

[30] Qingjie Zhang, Jiaolong Xu, Liang Xu, and Haifeng Guo. Deep convolutional neural networks for forest fire detection. In *2016 International Forum on Management, Education and Information Technology Application*. Atlantis Press, 2016.

A A simple computing unit (neuron)

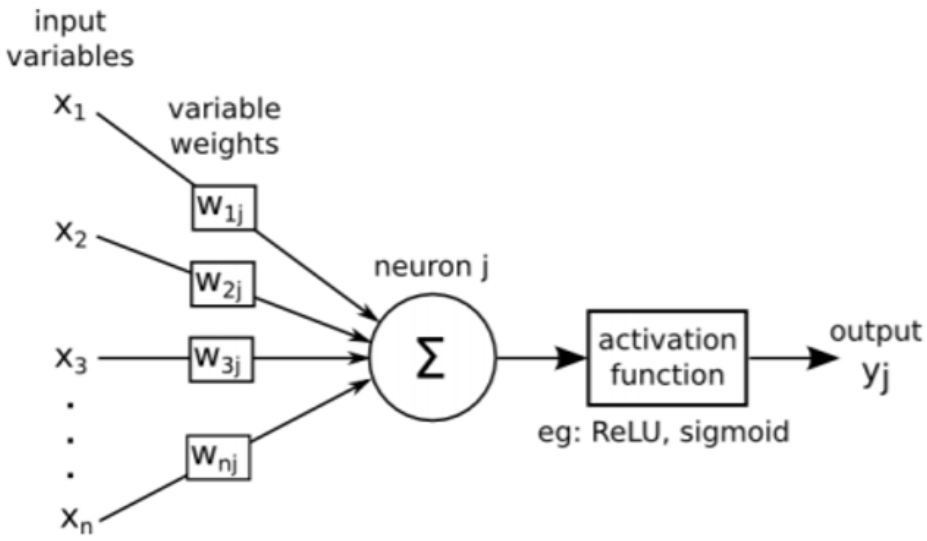

Figure 3: Functioning of a single neuron j

B Most common activation functions

The activation functions are mathematical equations that determine the output of a neural network. The function is attached to each unit to determine if it should be activated or not.

The Nonlinear Activation Functions are the most used since they make it easy for the model to generalize and differentiate between the output. They allow the model to create complex mappings essential for learning for units to solve complex problems.

The most commonly used activation functions are :

- **Sigmoid** : gives an output between 0 and 1. Figure 4 (a).
- **Tanh** : is symmetric around the origin and gives values between 1 and 1. Figure 4 (b).
- **ReLU** : as Rectified Linear Unit, gives values when x>0. Figure 4 (c).
- **LeakyReLU** : is an improved version of the ReLU function adressing the problem of ReLU consisting in obtaining null gradient for x<0 which would deactivate the units in that region. In fact, for x<0, Leaky ReLU is an extremely small linear component. Figure 4 (d).

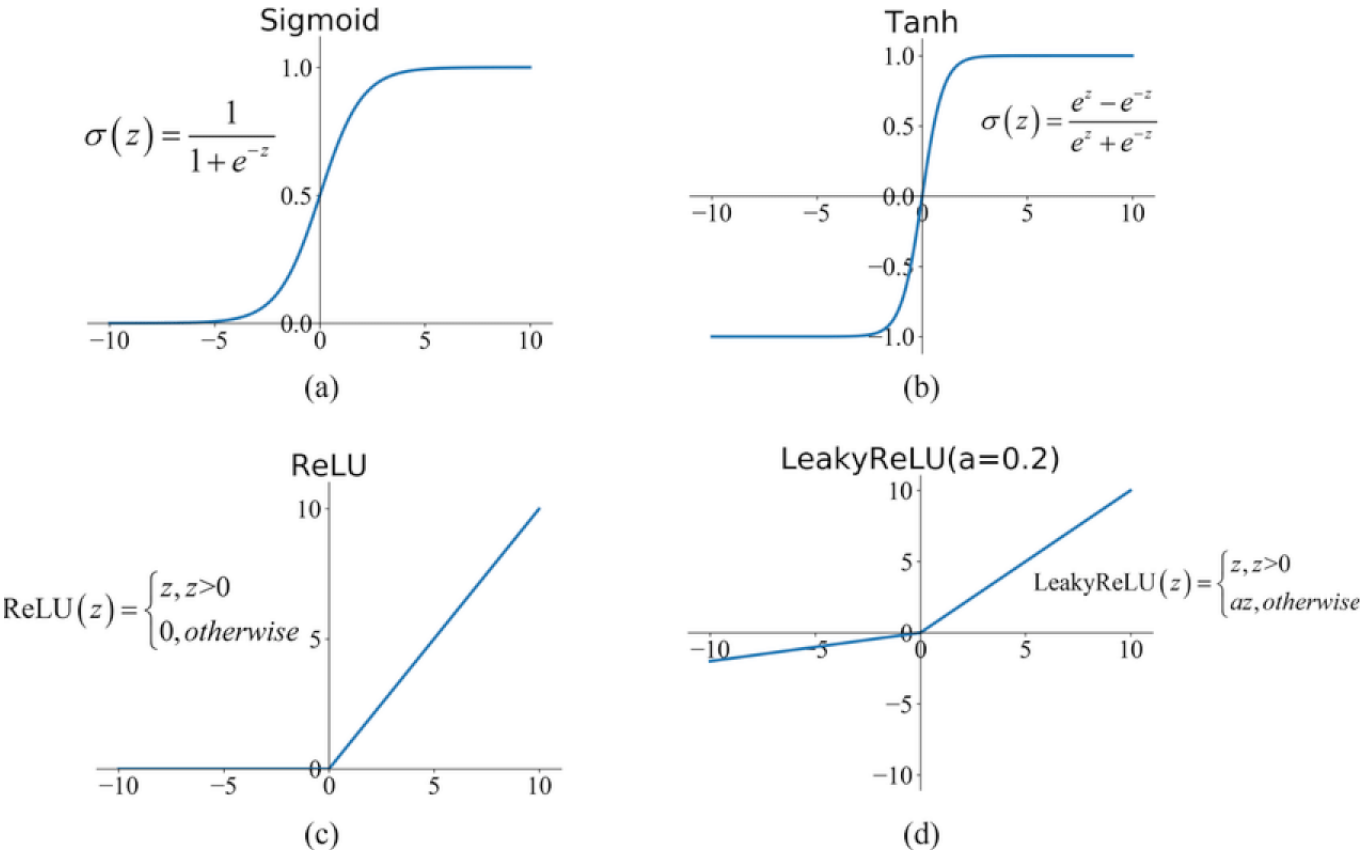

Figure 4: Most common activation functions

C Different Inception modules

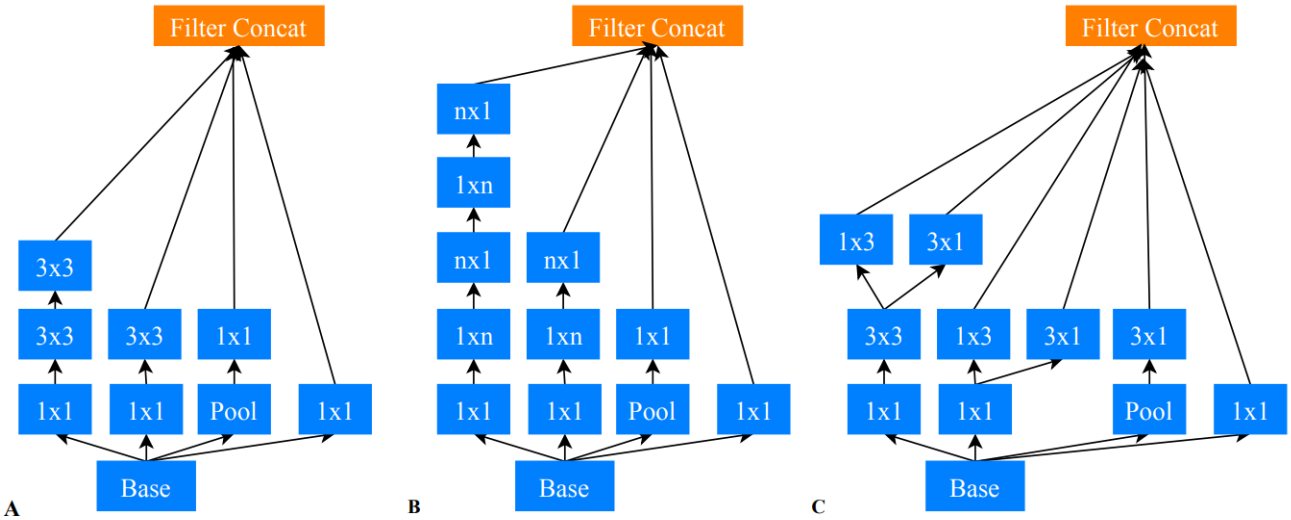

Figure 5: Inception Modules

D Grid size reduction

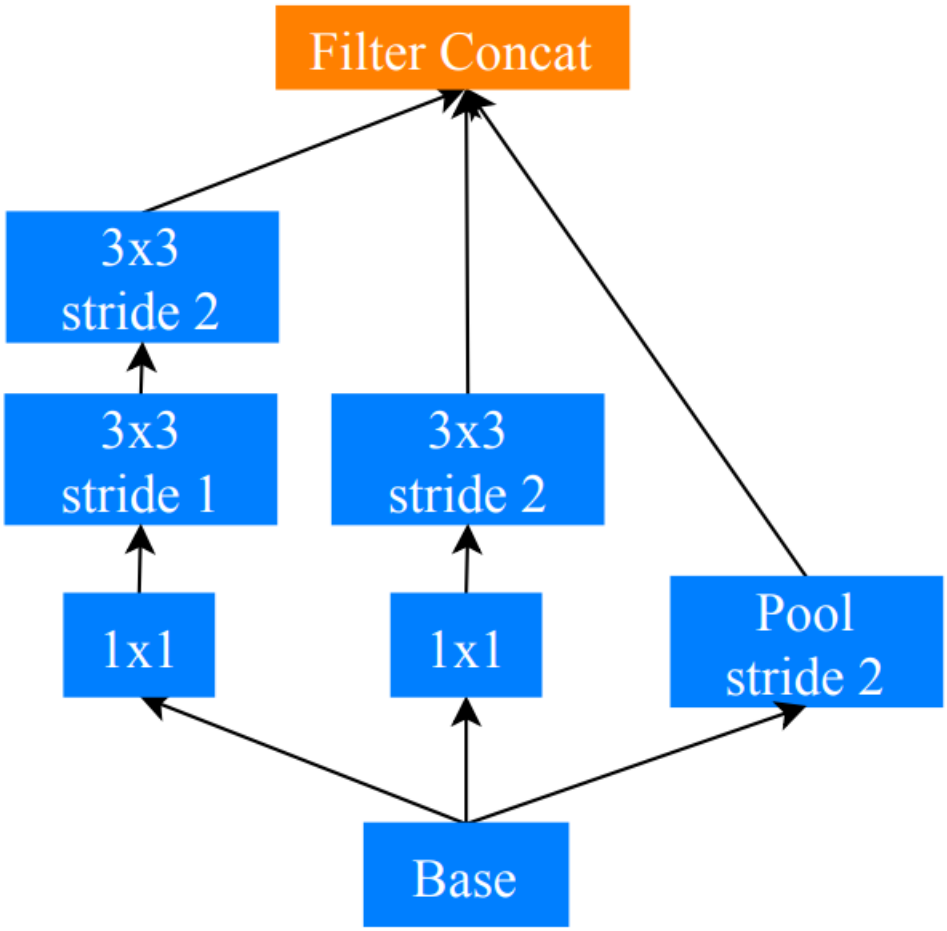

Figure 6: Grid size reduction scheme