

**BACHELOR OF SCIENCE IN  
COMPUTER SCIENCE AND ENGINEERING**



Inspiring Excellence

**Blurring of Inappropriate Scenes in a  
Video Using Image Processing**

AUTHORS

**Rahat Shahriar Islam**

**Raisa Siddiqui**

**Dipta Roy**

SUPERVISOR

**Dr. Jia Uddin**

Associate Professor

Department of CSE

**A thesis submitted to the Department of CSE  
in partial fulfillment of the requirements for the degree of  
B.Sc. Engineering in CSE**

**Department of Computer Science and Engineering  
BRAC University, Dhaka - 1212, Bangladesh**

**October 2018**



I would like to dedicate this thesis to my loving parents ...



## Declaration

It is hereby declared that this thesis /project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

*Authors:*

---

Rahat Shahriar Islam  
Student ID: 15101055

---

Raisa Siddiqui  
Student ID: 15101066

---

Dipta Roy  
Student ID: 14101247

*Supervisor:*

---

Dr. Jia Uddin  
Associate Professor, Department of Computer Science and Engineering  
BRAC University

December 2018

---

The thesis titled Blurring of Inappropriate scenes in a video using image processing

Submitted by:

Rahat Shahriar Islam Student ID: 15101055

Raisa Siddiqui Student ID: 15101066

Dipta Roy Student ID: 14101247

of Academic Year Fall 2018 has been found as satisfactory and accepted as partial fulfillment of the requirement for the Degree of Computer Science and Engineering.

- |    |   |             |
|----|---|-------------|
| 1. | <hr/> <p>Md. Abdul Mottalib, PhD<br/>Professor and Chairperson<br/>CSE Department<br/>BRAC University</p> | Chairperson |
| 2. | <hr/> <p>Dr. Jia Uddin, PhD<br/>Associate Professor<br/>CSE Department<br/>BRAC University</p>            | Supervisor  |

## **Acknowledgements**

We wish to express our sincere gratitude to Dr. Jia Uddin, our thesis supervisor for providing us with all the necessary facilities that was required for our research. We would also like to thank Dr. Jia Uddin sir for his sincere and valuable guidance and encouragement. We will forever be indebted to Dr. Jia Uddin sir for providing all the necessary help throughout the journey. Moreover, we would also like to thank all the Faculty Staffs of BRAC University for guiding us throughout the study period in BRAC University.

Finally, we thank our loving parents, siblings, friends for being our support system. They were always helping us directly or indirectly into making this project a success.





## **Abstract**

Inappropriate scenes such as bloody scenes, nudity, gore, drugs, weapons etc. are considered inappropriate especially in a developing Muslim country like Bangladesh. In our country, such scenes are very discouraged for children, old people or heart patients. Thus, keeping these in mind we propose a model where these inappropriate scenes will be detected and blurred from any video stream. Moreover, the model also shows percentage of explicitly in an input video file. As a result, people playing the video will know beforehand whether they would want to watch it or not and parents will have a greater control on what their children are watching. For nudity detection, there will be fragmented human figures that will be extracted and then the fragments will be compared against a database to decide whether nudity is involved or not. If it is involved and exceeds a predetermined threshold, then the video will be considered as pornography that many people may not prefer to watch. Similarly, the extracted figures of objects or gore scenes will be compared against a database to know the percentage of inappropriate scene in a video. Bringing both the nudity and goriness under one roof, we named the term explicit and if a video is explicit, the user will have the option of knowing it from beforehand and blur out any portion from the video automatically by using our model. The accuracy of our model is 93% and the algorithm we have used in this paper is CNN.



# Table of contents

List of figures

List of tables

Nomenclature

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Research Contribution . . . . .	1
1.3	Thesis Outline . . . . .	2
<b>2</b>	<b>BACKGROUND STUDY</b>	<b>3</b>
2.1	Literature Review . . . . .	3
2.2	Fragmentation of video . . . . .	7
2.3	Nudity Detection . . . . .	11
2.4	Object and Scene Detection . . . . .	11
2.4.1	Convolutional Neural Network to detect objects . . . . .	11
2.4.2	The LeNet Architecture . . . . .	13
2.4.3	Convolutional Neural Network Algorithm . . . . .	14
2.5	Blurring of Inappropriate Scenes . . . . .	20
<b>3</b>	<b>PROPOSED MODEL AND EXPERIMENTAL SETUP AND RESULT ANALYSIS</b>	<b>23</b>
3.1	Flowchart . . . . .	23
3.1.1	Work-flow Description . . . . .	23
3.2	Experimental Setup . . . . .	25
3.2.1	Training . . . . .	25
3.2.2	Blurring . . . . .	25
3.2.3	Results and Comparisons . . . . .	36

**4 CONCLUSION 39**

4.1 Limitations . . . . . 39

4.2 Future Work . . . . . 39

**References 41**

# List of figures

2.1	The diagram shows the skin pixels from the training data used in this project	5
2.2	The diagram shows the output from a GMM trained with two components on the same skin pixel data. The RGB color space is used.	6
2.3	Video to frame fragmentation	9
2.4	sample simulation of video to frame fragmentation	10
2.5	Examples of Object Detection	12
2.6	Example of Convolutional Network	13
2.7	Example of a CNN method for detecting object from an input image	13
2.8	convolution step dividing itself into layers separating each color layer	14
2.9	(a) – Pixelated picture, (b) – Equivalent pixel value	14
2.10	Convolution Example	15
2.11	Max Pooling Example	16
2.12	Three output maps from three input maps	17
2.13	Effects of convolution of the input image with different filters	18
2.14	Artificial Neuron	19
2.15	Example of a neural network	19
2.16	Probabilistic output after fully connected layer	20
2.17	Backpropagation	21
2.18	The picture above is the original inappropriate image and the below image is the blurred output image	22
3.1	Flowchart of the proposed model	24
3.2	Extraction of frames	26
3.3	Extracting frames	27
3.4	training the model	28
3.5	Frames extracted	29
3.6	Gore dataset	30
3.7	Training curve of the algorithm	31

---

3.8	The first image shows the cross entropy curve and the second image shows the validation accuracy curve . . . . .	32
3.9	The above curve shows the overlapping curve of training and validation accuracy . . . . .	33
3.10	Classify.py . . . . .	34
3.11	Blurred frames After Classifying . . . . .	35
3.12	Explicit Percentage Curve of Input Videos . . . . .	36
3.13	Difference in accuracy of the three models. . . . .	38

## **List of tables**





# Nomenclature

## Roman Symbols

$F$  complex function

## Greek Symbols

$\gamma$  a simply closed curve on a complex plane

$i$  unit imaginary number  $\sqrt{-1}$

$\pi$   $\simeq 3.14\dots$

## Superscripts

$j$  superscript index

## Subscripts

0 subscript index

crit Critical state

## Other Symbols

$\oint_{\gamma}$  integration around a curve  $\gamma$

## Acronyms / Abbreviations

ALU Arithmetic Logic Unit

BEM Boundary Element Method

CD Contact Dynamics

CFD Computational Fluid Dynamics

<i>CIF</i>	Cauchy's Integral Formula
CK	Carman - Kozeny
CNN	Convolutional Neural Network
DEM	Discrete Element Method
DKT	Draft Kiss Tumble
DNS	Direct Numerical Simulation
EFG	Element-Free Galerkin
FEM	Finite Element Method
FLOP	Floating Point Operations
FPU	Floating Point Unit
FVM	Finite Volume Method
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
LBM	Lattice Boltzmann Method
LES	Large Eddy Simulation
MPM	Material Point Method
MRT	Multi-Relaxation Time
PCI	Peripheral Component Interconnect
PFEM	Particle Finite Element Method
PIC	Particle-in-cell
PPC	Particles per cell
RVE	Representative Elemental Volume
SH	Savage Hutter
SM	Streaming Multiprocessors

## Nomenclature

---

USF Update Stress First

USL Update Stress Last



# Chapter 1

## INTRODUCTION

### 1.1 Motivation

Development of this scheme is quite necessary for children or even teenagers as they have started using internet quite a lot. They may come across documentaries or movies on internet that they should not watch. All parents do not want children to see a fighting scene like a man killing another man with a knife. Yet, he or she might end up watching it in any movie as these type of scenes are a common phenomenon in the movies nowadays. Therefore, having this model will help parents have better control over the videos their children watch. Moreover, people may not prefer watching a nude scene, adult scene or a crime scene due to personal preferences or even religious reasons. Thus, it comes in handy to them as well. For the aforementioned reasons, we came up with a proposed model to tackle and give the viewers an idea of what to expect in a video, provided them with the power to act accordingly to their wants, making watching movies and videos better and as personalized as ever.

### 1.2 Research Contribution

In past, there had been research work on developing algorithms that would detect pornography in a video or an algorithm that would detect nudity in an image or even a video clip. Experimental work on detecting crime scenes, gore scenes or any certain inappropriate scene had been done as well. However, all of these algorithms focused on one particular aspect only. Inappropriate scene detection model is a system that defines inappropriate scenes on a wide range of spectrum. From pornography, to small duration adult scenes in any movie or video or documentary, to just a nude scene, the definition of inappropriate ranges upon criminal scenes, gore scenes, murder scenes, extreme violence, showing of dangerous weapons [2], etc.

The Convolutional Neural Network that had been used in detecting particular inappropriate objects or scenes can range and vary according to the preference of an individual user [11]. Thus, this system will provide better detection over a wide range of scenes, making it more personalized and useful.

Moreover, previously many work on nude detection had been done mostly on the basis of skin pixel detection [15] and then relative analysis of skin pixel with non-skin pixel to conclude whether the scene have nudity or not. However, our system ensures that human bodies are perfectly identified and separated from the original picture into a different image containing only the human figures before performing skin detection algorithm on it [16]. This enhances the performance of nudity detection as even a small nude portion in a large image can be identified and handled. Hence, the combination makes our nudity detection algorithm much more efficient and effective.

### **1.3 Thesis Outline**

Chapter 2 contains background studies that provide the literature review, overall work flow, stating the algorithms and techniques that have been used.

Chapter 3 shows the proposed model and experimental setup and result analysis.

Chapter 4 is conclusion that includes limitations and future works.

# Chapter 2

## BACKGROUND STUDY

### 2.1 Literature Review

There are number of works which are dealt with the same problem domain. We have noticed that there are number of different approaches to get the similar result. Some of the methods are Gaussian distribution, Gaussian mixture methods, artificial neural network, Bayesian networks, histograms, geometrical shape and thresholds. Our project is based mostly on convolutional neural network. In the paper automatic detection of image containing nudity using image processing tried to detect sexually explicit image by applying artificial neural network and two color models RGB and IHLS for detecting skin [2]. Their first step was to define an undesired image or nudity in an image. They did not take other objects into consideration like symbols, offensive images and so on. Their approach was to extract a number of features from the pictures and applying those features to the algorithm to get the desired result [2]. In order to come up with features, they started on a high level with fuzzy descriptions. The descriptions can then be divided into sub-features which can be found in images. The three high level features they came up with are:

1. Nudity: They had attempted to detect nudity by detecting skin. They had created a system that can detect different skin color in different lighting condition.
2. Shape: To determine shape, they had created some kind of template to match with the content. They created a program to use those templates on the image that need to be checked. In an image, the program tries to match by stepping through and scanning every possible location. If the program matches with the template in the image, then it is marked as detected. They had created various templates to detect several objects with variable size so that a program can scan any image of different perspective.

3. Object Classifier: A system that searches for specific object in an image. For example, an object can be a specific body part like a nose.

Methods: They had trained several different Gaussian Mixture Models (GMMs) by varying the number of components for both the RGB and IHLS color space. For each of the color spaces, they had trained 11 GMMs with 2, 3, 4, 5, 6, 7, 8, 10, 12, 14 and 16 components. Their target was to find the skin-pixels from the image. However, images usually have a high resolution nowadays. So, it is impossible to use GMMs if the computation takes more than a couple of seconds. They had overcome this problem by computing the skin probability for each pixels in the RGB color space and saved them in a look-up table with a size of 256 x 256 matrix.

They have used Artificial Neural Network (ANN) for skin segmentation. At first, they trained ANN for a single 3 x 3 pixels region. After getting success on small regions they considered the whole image. At first they had tested with the size of only 30 pixels and the output was same as the input size on the form of 0-1 where, bigger number means that it was more likely to skin. All their early testing with ANN was on segmented images and they got 90% of correct matching when ANN was trained with large set of data containing both sexually and non-sexually explicit images. During their process, they had extracted a lot of feature from each image that helped the system to recognize images containing nudity. ANN was also trained to recognize a number of faces and largest skin exposed in an image. They took into account the position of the largest skin-region as well as the skin percentage. If the largest skin-region is in the center of the image, then it might contain nudity. They also tried a few other methods to get the optimum result like face detection, feature network, evaluation, bootstrapping etc. They used ROC curve to plot the outputs obtained from different methods and to evaluate the output of each methods. In skin detection ANN, they had reached a true positive rate of 80% with 13% false positive rate. This method is very fast and can process every frame from web cam in real time. They also run the 9-pixel skin detection using ANN which used the same process as skin detection ANN where, inputs were 27 for each 9 pixels and got almost the same result. However, it requires more processing power and longer computation time than a single pixel network. In terms of multiple-pixel ANN, they did not get the satisfactory result. Here, the network scored only 65% true positive rate. They had stated that due to lack of training data, this multiple-pixel network did not work as they expected. In case of image based neural network, their system performed outstandingly. On the training set, it scored 90% of true positive rate with only 1.1% of false positive rate and the computation speed was also very promising. In terms of feature based network, they had used over 3000 training cases as this was fairly a large network. However, they got a very disappointing result. It had an error of only 0.1% in training data but performed very poorly



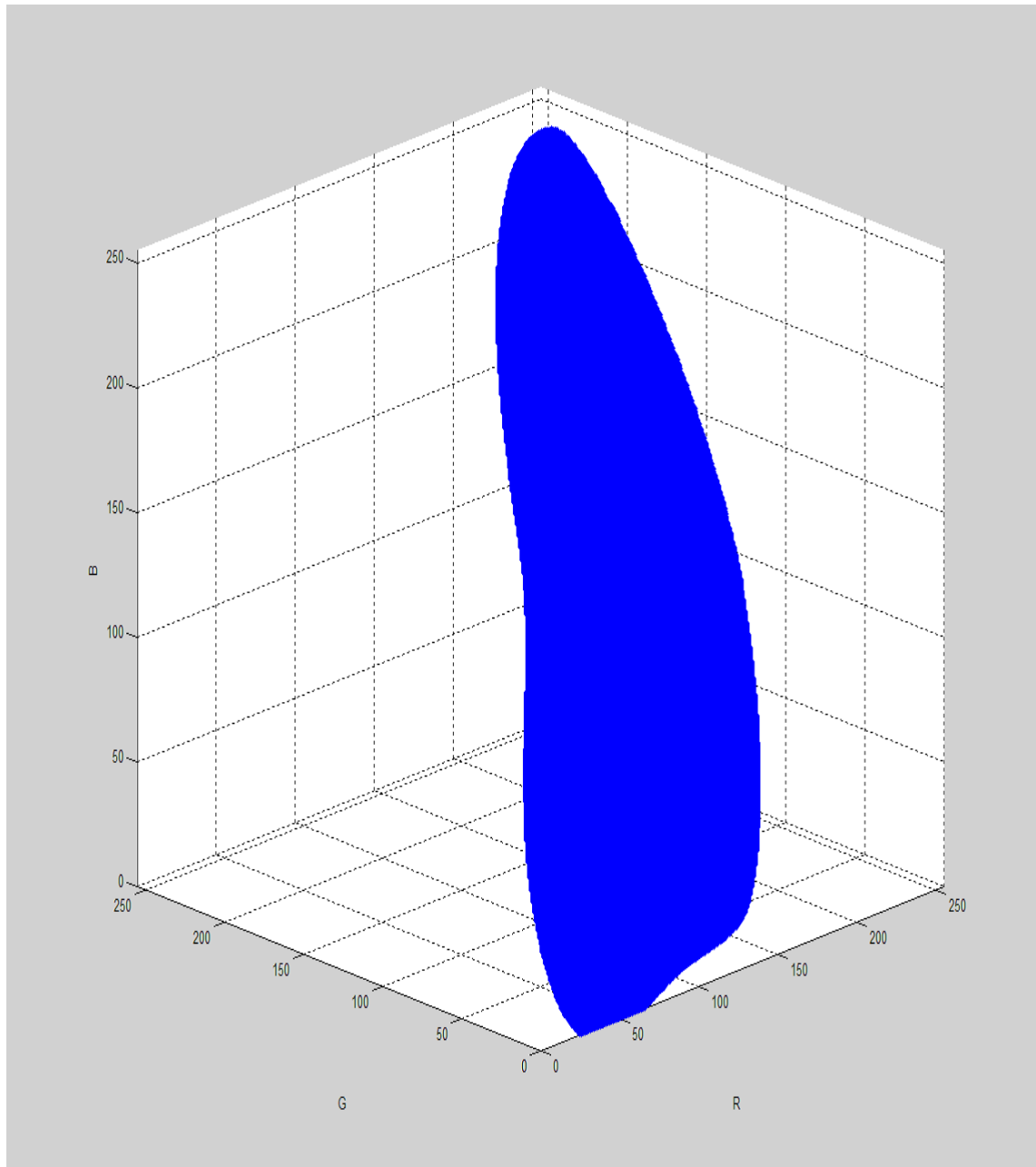


Fig. 2.1 The diagram shows the skin pixels from the training data used in this project

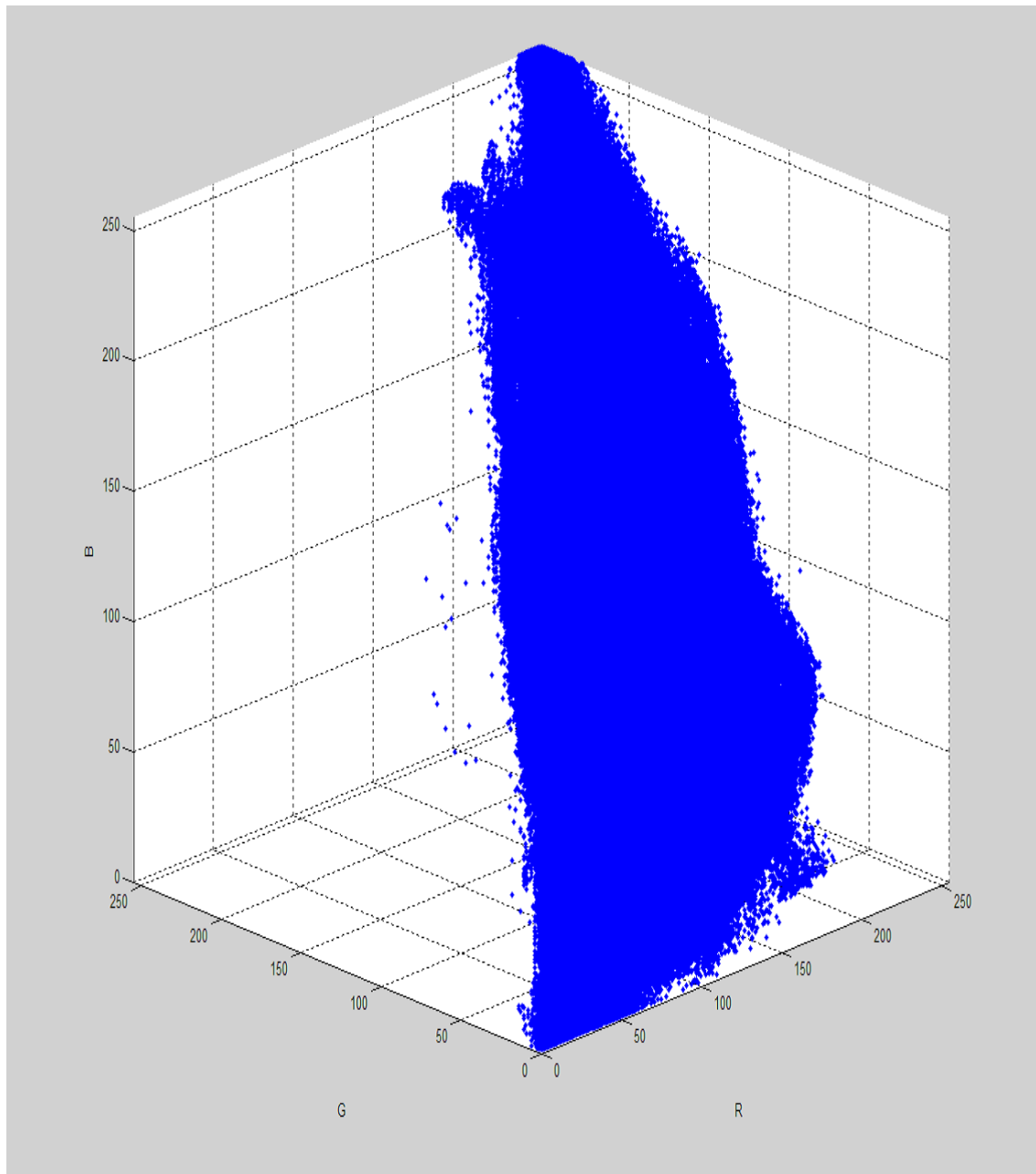


Fig. 2.2 The diagram shows the output from a GMM trained with two components on the same skin pixel data. The RGB color space is used.

in test data. Face detection was one of the vital features for their project. It also gave a very good output though the network encountered some problem in recognizing small faces and side of the faces. They had tried various approaches to detect nudity from image. When ANN was trained with whole image rather than small pixels, their system was successful in detecting nudity with a high true positive rate using image segmentation with RGB and IHLS as it itself trained the network with the feature extracted from the image.

There was another approach to detect skin. Their sole target was to build a Maximum Entropy Model for the skin distribution [20]. In this model, color gradients of adjacent pixels combined with Bethe Tree and Belief Propagation was considered as constraints. The output from the skin detection model map in gray scale and the brightness is equal to the probability of skin. The next target was to extract features and fit ellipses, global fit ellipse and local fit ellipse. Then, an artificial neural network consisting of one hidden layer works on pattern recognition. The author did not publish any performance numbers in his paper. However, from the ROC curve of the article we had seen 80% of true positive rate with 10% false positive and 90% true positive rate with 18% false positive rate.

In the paper "An elliptical boundary model for skin color detection", they tried to detect skin using elliptical boundary model which classify skin chrominance from non-skin chrominance [10]. The author tested six different chrominance spaces. Their comparison was between elliptical boundary model and Gaussian mixture models. They did not publish what kind of training data they considered. However, in their testing they had used 2000 skin images and 4000 non-skin images without any segmentation. This is why, it is not clear if the output is classified image or pixel-wise classification. However, from the ROC curve we get true positive rate of 95% where the false positive rate reached 35%.

## 2.2 Fragmentation of video

The very first step is to convert the video file into its individual frames [17]. In order to do that, we used Open CV(Open Source Computer Vision Library) to convert the video into frames. Open CV is an open source computer vision and machine learning software library. Open CV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, Open CV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects,

produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. Open CV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start ups such as Applied Minds, Video Surf, and Zeitera, that make extensive use of Open CV. Open CV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. Open CV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available [1]. A full-featured CUDA and Open CL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. Open CV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

At first, we take a video file as an input and then make a directory with the corresponding name of the video file. Then we extract the number of frames of the video using Open CV and store it to run a loop to extract each frame from the video. On extraction of each frame, we store it in the recently created directory. The file names of each of these extracted frames correspond to their frame number in the video. After reading all the frames of the video, we move on to our next step of the model. The above process is presented in the flow chart in Fig. 2.3. A sample simulation has been shown in Fig. 2.4.

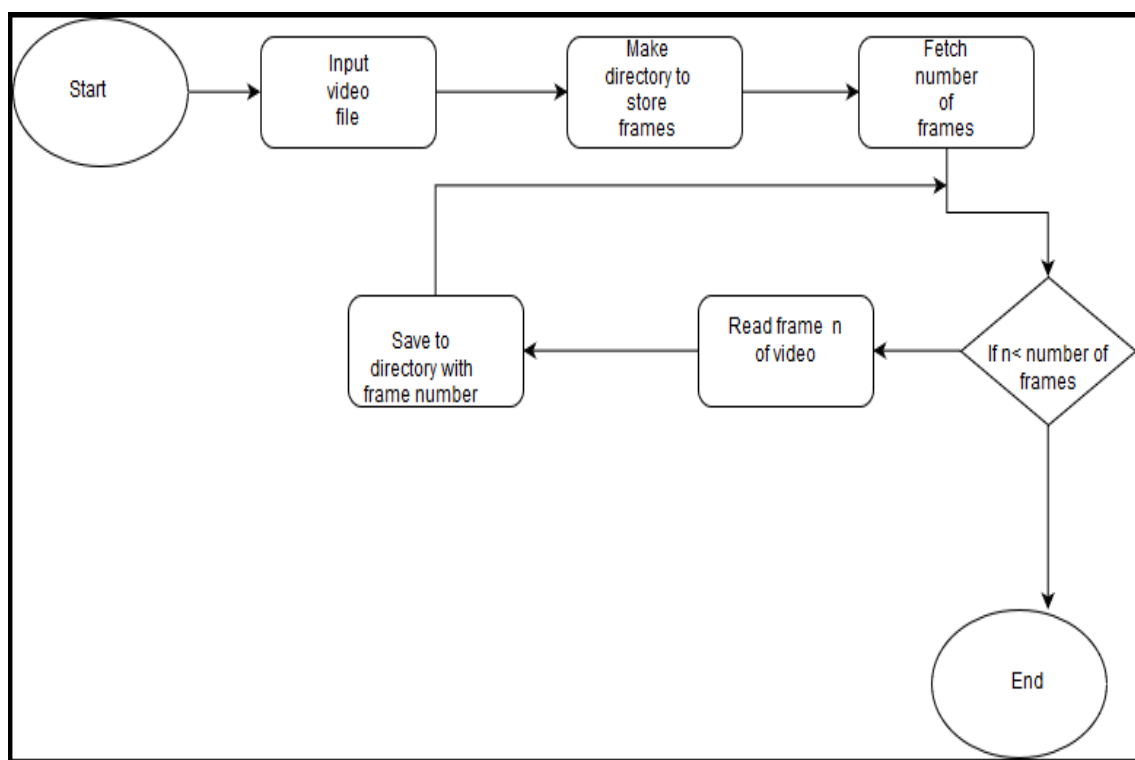


Fig. 2.3 Video to frame fragmentation

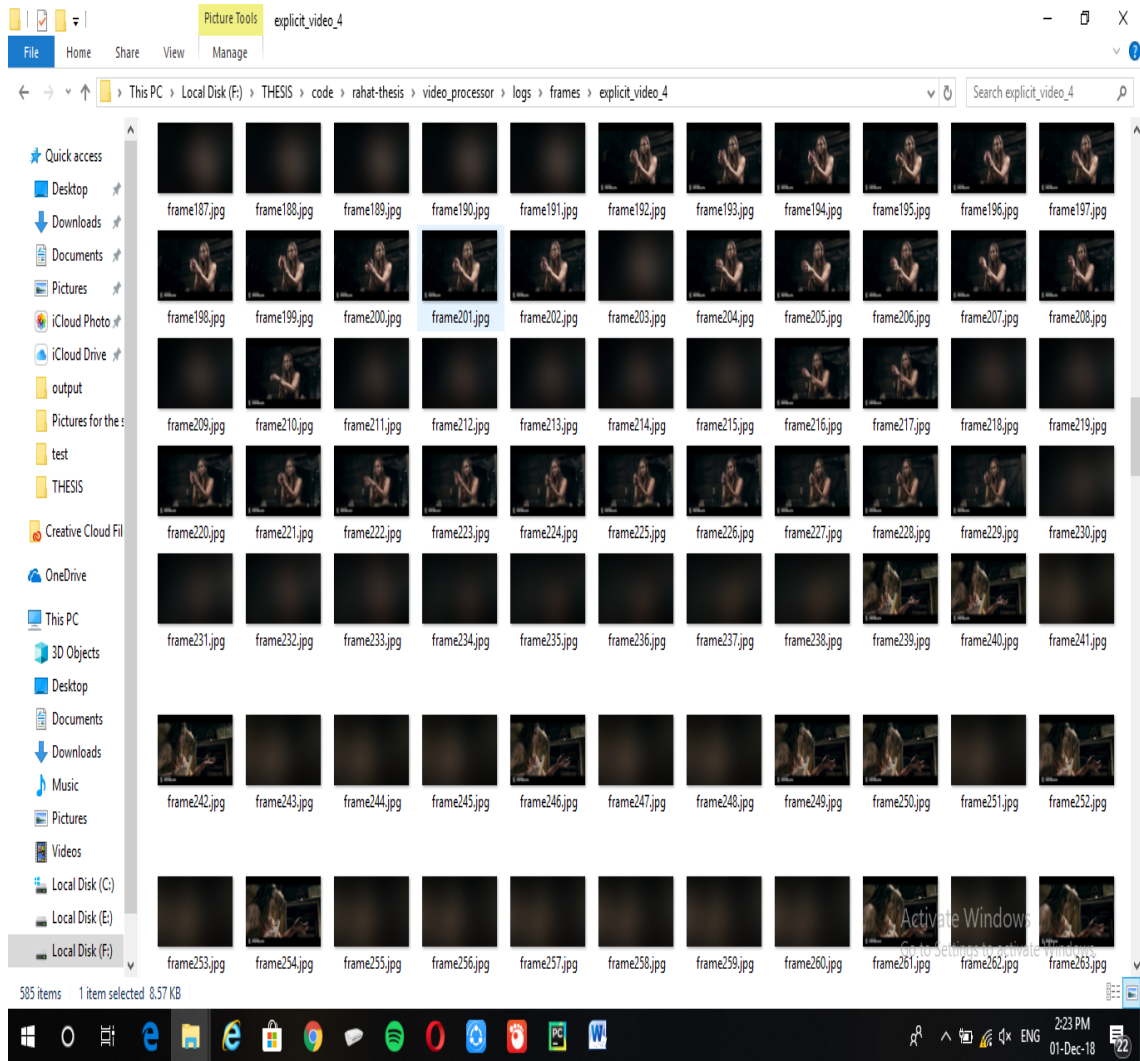


Fig. 2.4 sample simulation of video to frame fragmentation

## 2.3 Nudity Detection

The nudity detection part is implemented with the combined use of two libraries of the CNN algorithm. The first library i.e. Open CV, mainly detects human figures from the fragmented frames and crop them out in a separate image file. Then CNN is used on the new image files to determine if the images are nude or not. The threshold that has been currently set in the prototype made is 78% [9]. Any frame that has a nudity percentage of over 78% is considered as an inappropriate scene and is therefore eligible for blurring. The model has been trained with over 50,000 frames of images of nude scenes from videos all over the internet to make the process as accurate as possible.

## 2.4 Object and Scene Detection

The main motive behind object detection is to have a control on what the viewers want to watch according to their preferences. Some people might not want to watch dangerous weapons, drugs, gore scenes. Our model will detect these unwanted things and if it exceeds our predetermined threshold of 78%, then the image will be blurred. These scenes can be really harmful for people with mental disorder, children and older people. Thus, it is very essential to detect and blur such scenes prior to watching the video. To achieve this, we have used Convolutional Neural Network.

### 2.4.1 Convolutional Neural Network to detect objects

A Convolutional Neural Network, also known as CNN or ConvNet, is an artificial neural network that has so far been most popularly used for analyzing images [18]. CNN has some type of specialization for being able to pick out or detect patterns and this is what makes CNN so useful for image analysis[8]. It is known to work on a grid-like topology. CNN has been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars. CNN can be used to identify multiple objects at a high success rate. Examples of object detection results obtained by using CNN algorithm in a given image is shown in Fig. 2.5.

CNN consists of three steps to detect objects. These are convolution, pooling, getting fully connected network and finally, giving the output with probability matching [7]. It goes through two phases of convolution and pooling followed by fully connected phase and finally giving the required output. An example of convolutional network is shown in Fig. 2.6.

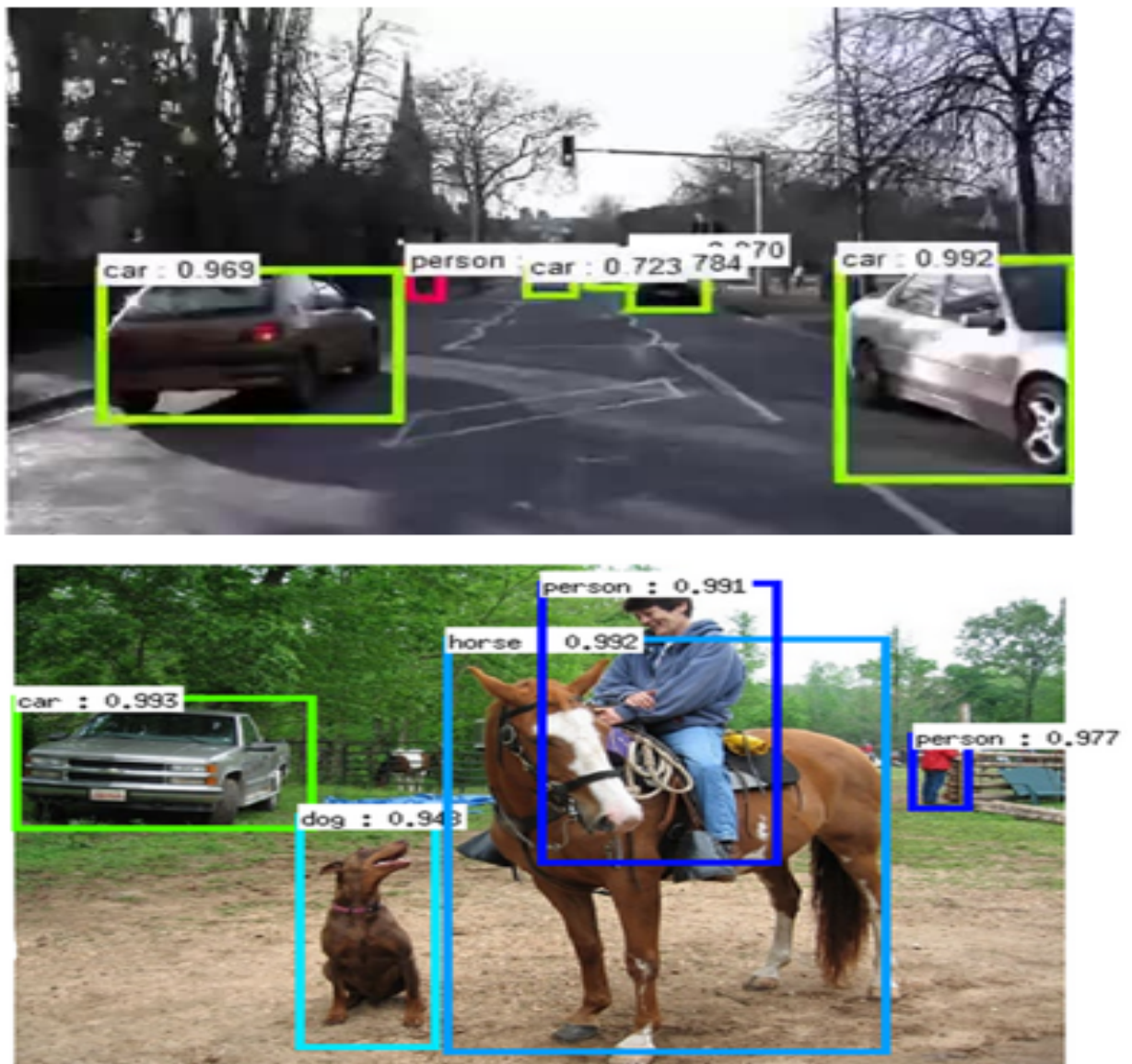


Fig. 2.5 Examples of Object Detection



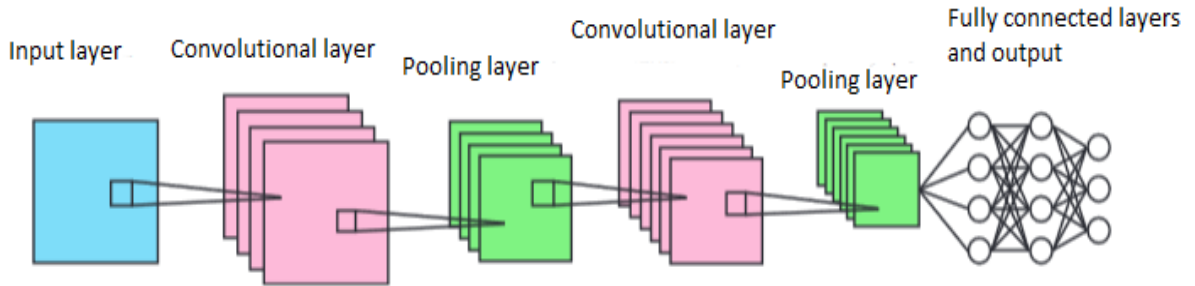


Fig. 2.6 Example of Convolutional Network

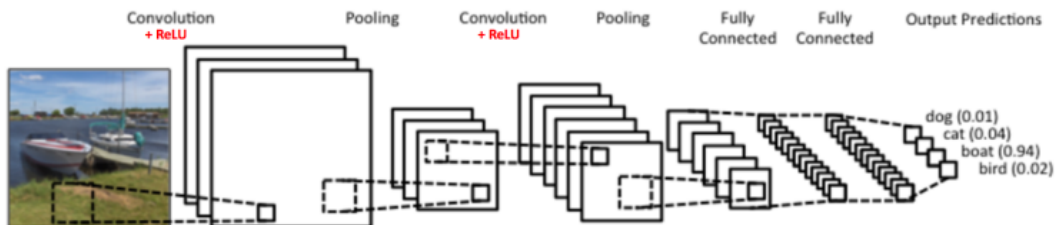


Fig. 2.7 Example of a CNN method for detecting object from an input image

### 2.4.2 The LeNet Architecture

LeNet was one of the very first convolutional neural networks. The LeNet architecture was first introduced by LeCun et al. in their 1998 paper. LeNet was used primarily for OCR and character recognition. Due to lack of significant computing power at that time, CNN started picking up after some years. The latter works are basically improvements done on LeNet in order to enhance its recognizing capabilities and increase its accuracy and efficiency. Convolution layer, Sub sampling or pooling, classification or fully connected layer are basic blocks of every CNN as mentioned earlier. The Convolutional Neural Network as shown in Fig. 2.7 is similar in architecture to the original LeNet and classifies an input image into four categories such as dog, cat, boat or bird whereas, the original LeNet was mainly used for character recognition tasks.

Before getting into how the architecture works, it is necessary to know that every image can be represented as a matrix of pixel values. A channel is a term used to refer to a component of an image. An image from a standard digital camera will have the channels red, green and blue (commonly known as RGB). These channels can be imagined as three 2D matrixes (one



Fig. 2.8 convolution step dividing itself into layers separating each color layer

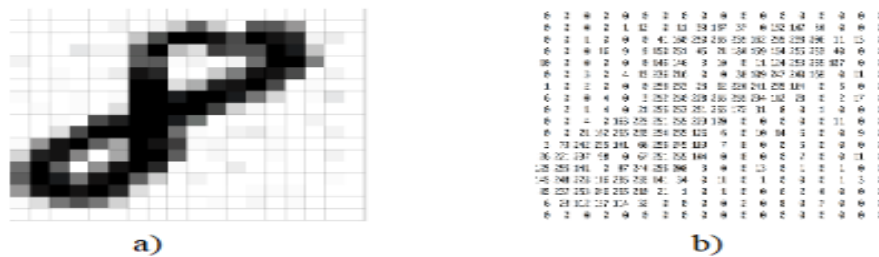


Fig. 2.9 (a) – Pixelated picture, (b) – Equivalent pixel value

for each channel) stacked over each other and each having pixels in the range 0 to 255 where, a value of 0 indicates black and 255 indicates white and the between values are arranged accordingly. A gray scale image has only one channel and the pixel range is also from 0 to 255 where, a value of 0 indicates black and 255 indicates white and the between values are arranged accordingly. In case of a colored image, the convolution step divides itself into layers separating each color layer (i.e. red, green blue) as shown in Fig. 2.8 and then performs its operations.

The pixel value is a single value that represents the brightness of a pixel. Every image can be represented as a matrix of pixel values as shown in Fig. 2.9.

### 2.4.3 Convolutional Neural Network Algorithm

#### Convolutional Operation

Convolution is a process of moving a filter mask over the image and computing the sum of products at each location except that the filter is first rotated by 180 degrees. Convolution is a mathematical operation on two functions [3]. The formula for convolution is given in equation (2.1).

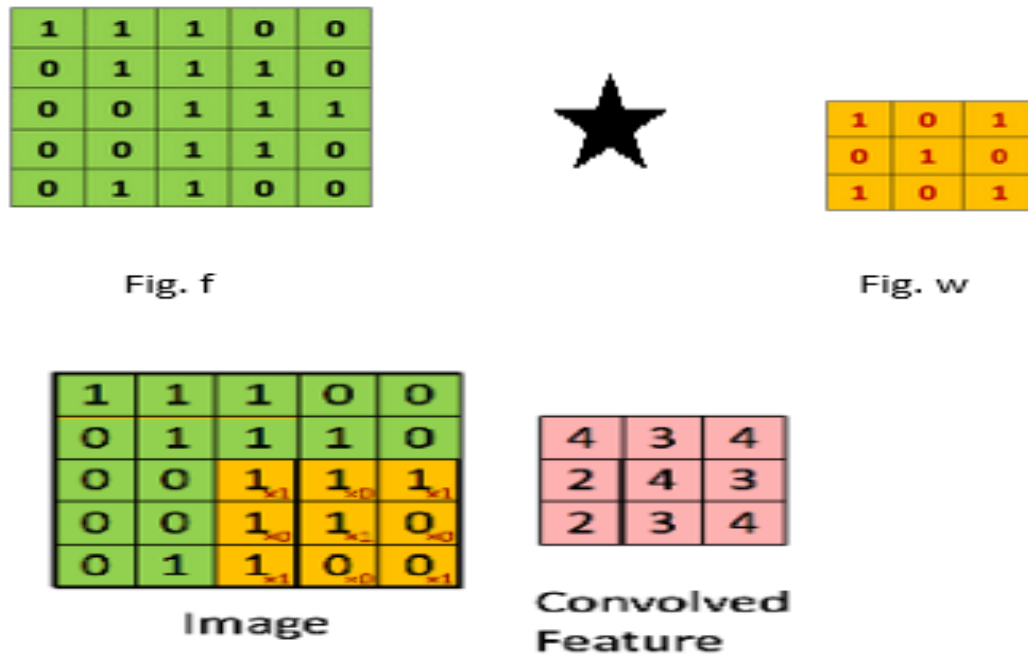


Fig. 2.10 Convolution Example

$$w(x,y) * f(x,y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s,t) f(x-s,y-t) \quad (2.1)$$

An example of the computation of convolution is shown in Fig. 2.10. In order to correctly identify or detect an object with maximum noise reduction, several images are taken and then the average operation is computed.

The computation is done by sliding the orange matrix over the original image (green) by 1 pixel (also called 'stride') and for every position, we compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix (pink). Top-left value of 4 is obtained by  $(1*1) + (1*1) + (1*1) + (0*1) + (1*1) + (1*0) + (0*1) + (0*0) + (1*0)$ . The matrix in Fig. w is called kernel, filter or feature detector or 'feature detector' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Activation Map' or the 'Feature Map'.

The primary purpose of convolution is to extract significant features from the input image. The convolution step preserves the spatial relationship between pixels by learning image features using small squares of input data.

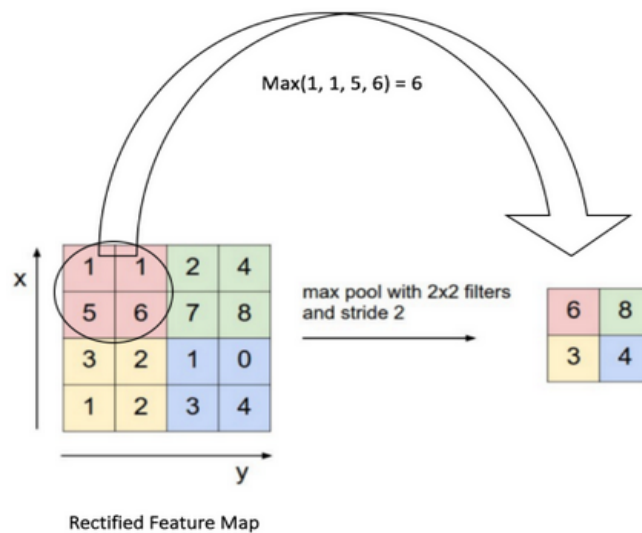


Fig. 2.11 Max Pooling Example

## Pooling

The pooling layer is usually placed after the Convolutional layer. Much like the convolution operation performed above, the pooling layer takes a sliding window or a certain region that is moved in stride across the input transforming the values into representative values. The transformation is either performed by taking the maximum value from the values observable in the window (called 'max pooling'), or by taking the average of the values. However, max pooling has been favored over others due to its better performance characteristics.

As discussed above, each pixel of an image gives the corresponding pixel value, thus forming a matrix of pixel values. The convolution step takes the image, selects an area of the image, and then performs max pooling, to collect the highest valued feature match from that particular section of the convoluting task. An example of max pooling is shown in Fig. 2.11. In the network shown in Fig. 2.12, pooling operation is applied separately to each feature map. As a result, we get three output maps from three input maps.

It is evident from the Fig. 2.10 shown above that different values of the filter matrix will produce different Feature Maps for the same input image. Thus, we can see the effects of convolution of the input image with different filters. As shown in Fig. 2.13, we can perform operations such as Edge Detection, Sharpen and Blur just by changing the numeric values of

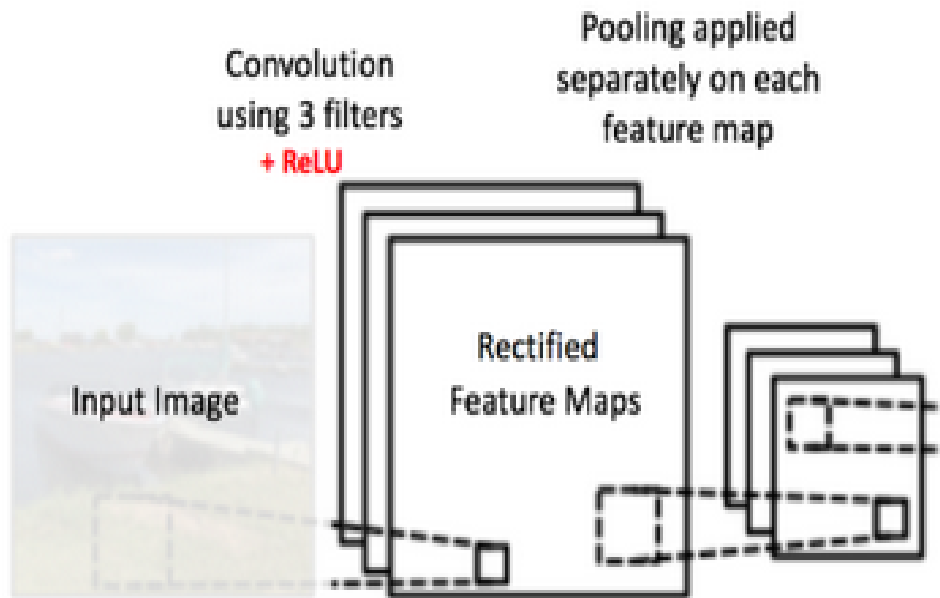


Fig. 2.12 Three output maps from three input maps

our filter matrix before the convolution operation. Hence, different filters can detect different features from an image, for example edges, curves etc.

It is important to understand that these layers are the basic building blocks of any CNN as mentioned earlier. As shown in the above Fig. 2.7, the second Convolution layer performs convolution on the output of the first Pooling Layer using six filters to produce a total of six feature maps. Then Max Pooling operation is performed separately on each of the six feature maps. Together these layers extract the useful features from the images. The output of the second Pooling Layer acts as an input to the Fully Connected Layer, which will be discussed in the next section.

### Fully Connected Layer

The final step of the CNN is to form the fully connected network and give an output of probability. The Fully Connected layer is a traditional Multi-Layer Perceptron. In a fully-connected feed-forward multi-layer network, each output of a layer of neurons is fed as input to each neuron of the next layer. The FCN uses the concept of an artificial neuron. It takes a series of inputs, which are weighted, from the previous step of the process as shown in Fig. 2.14.






Operation	Filter	Convolved Image
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Edge detection</b>	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Sharpen</b>	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Fig. 2.13 Effects of convolution of the input image with different filters

A Multi-Layer Perceptron (MLP) contains one more hidden layers in the process. The input layer is simply the input that it is fed from the previous step, and that is not altered by the FCN [5]. Next, in the hidden layer, it takes the weights of all input nodes and creates new hidden nodes. These are called hidden nodes because, it has no direct connection with the outside world. Furthermore, the required calculations and computations required to give outputs is all done in this layer. The values are then fed to the output layer where each node in the output layer is again calculated using the concept of neurons, shown above. After this, all the outputs are compared to see which has the best probabilistic value out of all in order to properly identify the given object that was fed to the CNN's algorithm.

A multi-layer network typically includes three types of layers: an input layer, one or more hidden layers and an output layer as shown in Fig. 2.15. The input layer usually merely passes data along without modifying it. Most of the computation happens in the hidden layers. The output layer converts the hidden layer activation to an output [6].

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training data set [9]. The network takes a training image as input, goes through the forward propagation step and finds the output

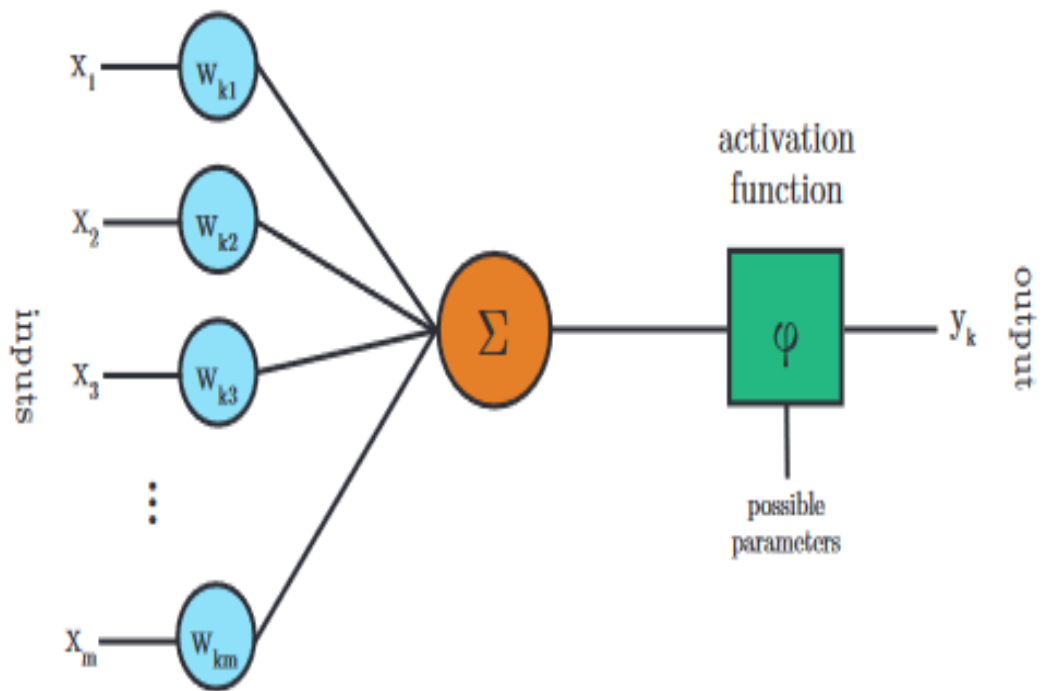


Fig. 2.14 Artificial Neuron

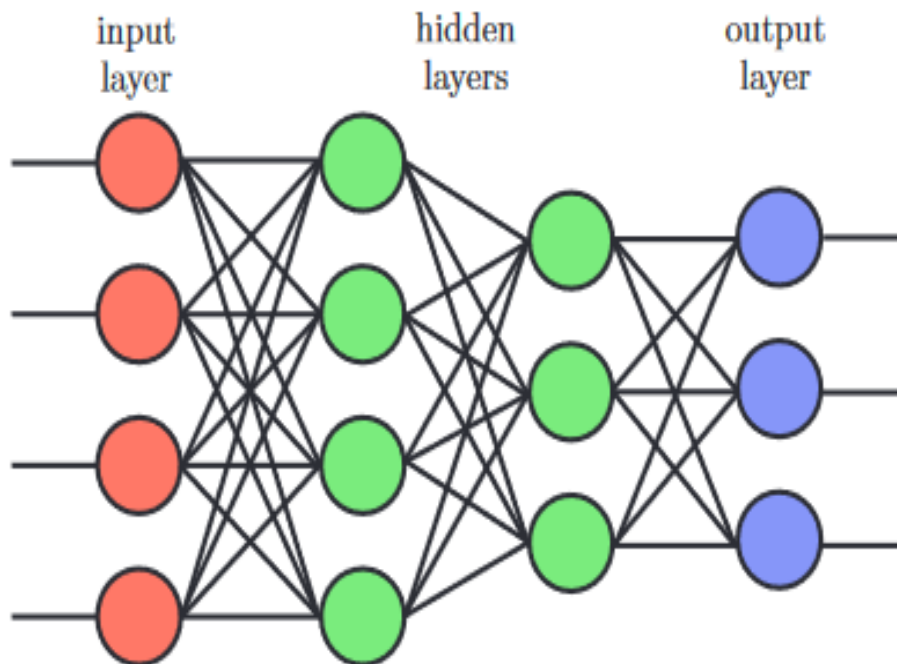


Fig. 2.15 Example of a neural network

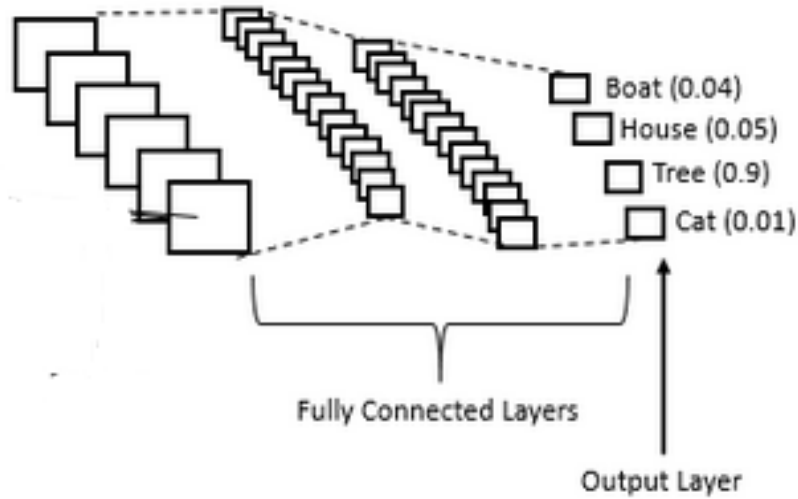


Fig. 2.16 Probabilistic output after fully connected layer

probabilities for each class as shown in Fig. 2.16.

Since weights are randomly assigned for the first training example, output probabilities are also random. The total error at the output layer is given by the formula shown in equation (2.2).

$$TotalError = \sum \frac{1}{2} (target\ probability - out\ put\ probability)^2 \quad (2.2)$$

Then, back propagation is used to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values/weights and parameter values to minimize the output error. In order to achieve higher accuracy, the received data in the output neurons is sent backwards to the nodes to recalculate the function with newly assigned weighted measurements [3] as shown in Fig. 2.17.

## 2.5 Blurring of Inappropriate Scenes

If any inappropriate scene is detected after nudity detection and object detection, then the next step is to crop out the inappropriate part and blur it. Pillow is a python library using which blurring is carried out. The video is re rendered including all the audio files and the frames that were detected as inappropriate are now blurred [14]. Pillow offers several standard procedures for image manipulation. These include:



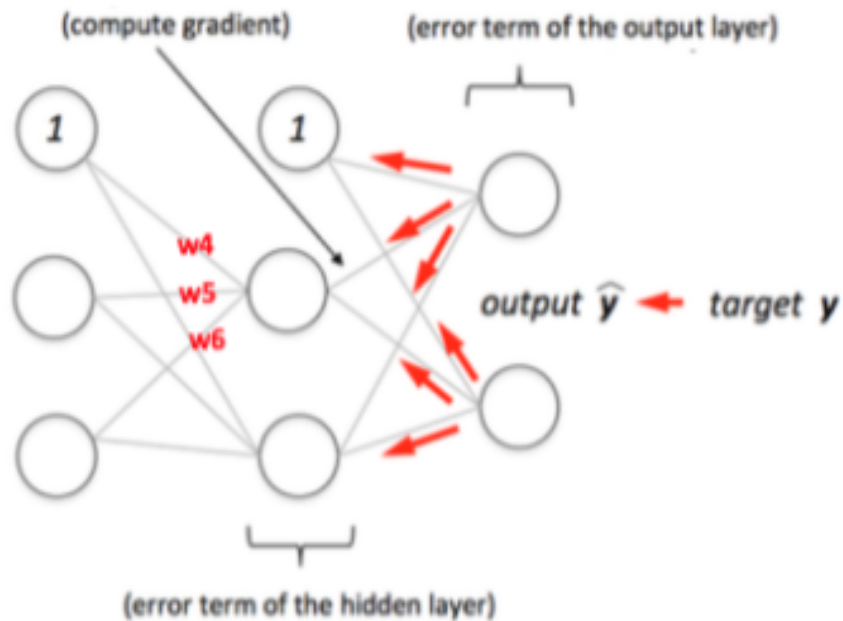


Fig. 2.17 Backpropagation

- per-pixel manipulations,
- masking and transparency handling,
- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- adding text to images and much more.

The system uses the third section stated above which helps in blurring out certain scenes from a video clip and thus making the video safer for any user that decides to use the model. An example of blurring an inappropriate scene is shown in Fig. 2.18. The accuracy of the scenes being blurred again depends on the training method of the code. Any such scene which have been recognized as a gore or nude is blurred out by Pillow.

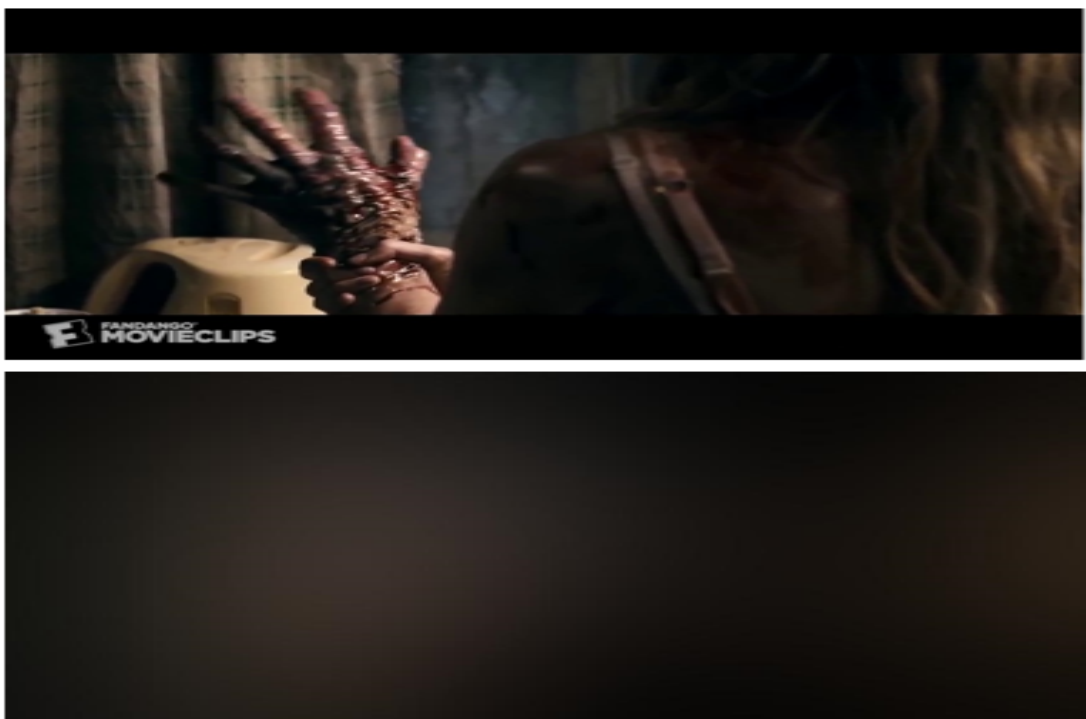


Fig. 2.18 The picture above is the original inappropriate image and the below image is the blurred output image

## **Chapter 3**

# **PROPOSED MODEL AND EXPERIMENTAL SETUP AND RESULT ANALYSIS**

In this chapter, we provided the flowchart of our proposed model, the experimental setup as well as the results and comparisons in the following sections.

### **3.1 Flowchart**

The flowchart of the proposed model as shown in Fig. 3.1 consists of three major steps. The very first step is to fragment the video into frames. The next two steps consist of nudity detection and object detection.

#### **3.1.1 Work-flow Description**

The work flow for this model has three major steps. In the very first step, the video input taken is fragmented into different frames. Then, the two detection processes nude detection and object detection are run simultaneously [17]. The process of nude detection revolves around two algorithms. The human detection algorithm crops out human portion and passes it to the nude detection algorithm and then based on the result from both of the algorithms, the percentage of nude images is calculated. If the percentage is more than 78%, then we can conclude that the video has pornography. Otherwise, presence of adult scenes is drawn as conclusion. Likewise, object detection algorithm is also performed to detect specific type of scenes. For instance, if we are detecting crime or gore scenes, then the object detection

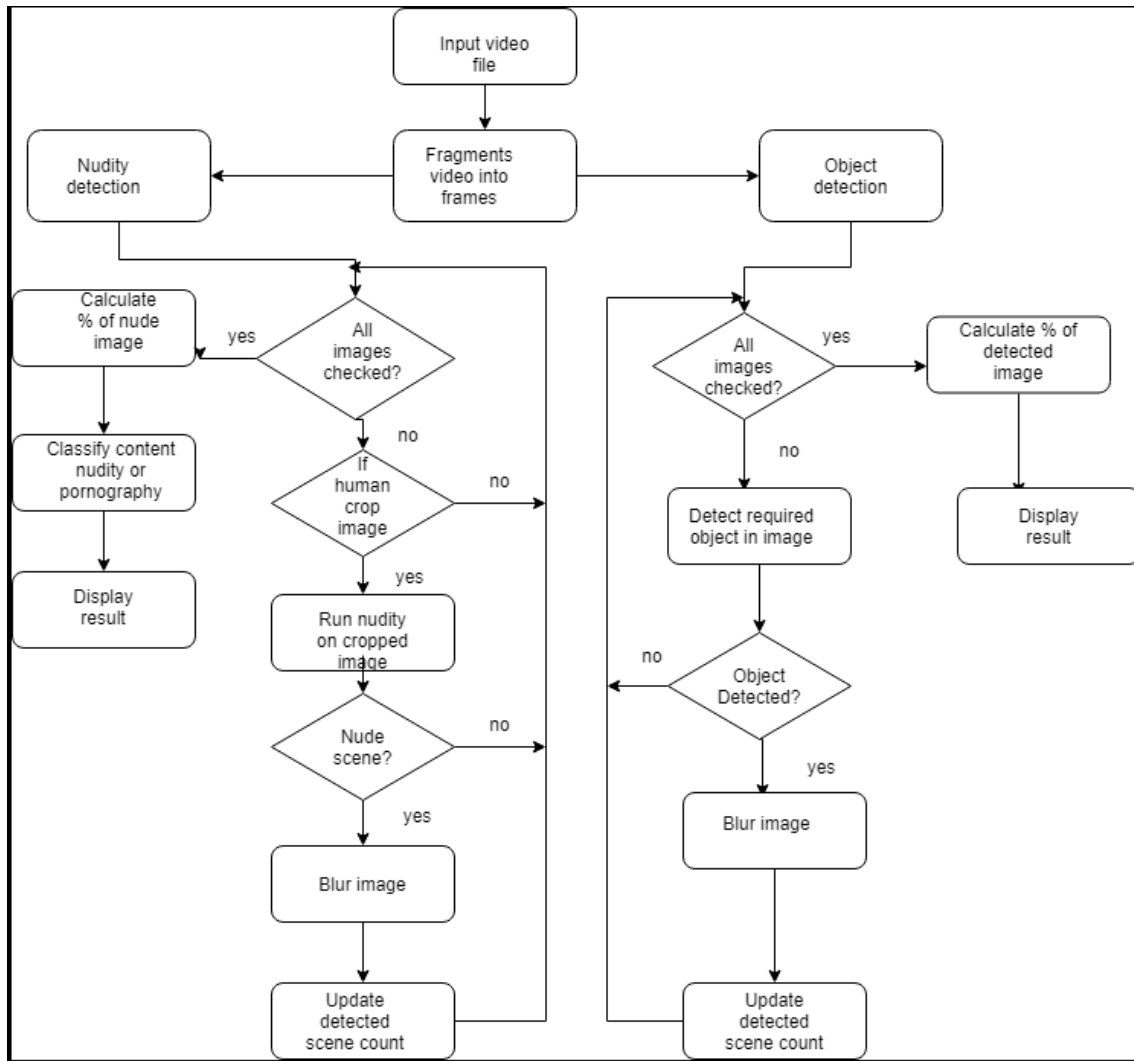


Fig. 3.1 Flowchart of the proposed model

algorithm would check for it in all of the frames and give out the percentage of inappropriate scenes in the total video.

## 3.2 Experimental Setup

The proposed model is divided into two parts as already discussed. One part deals with nudity detection which further determines adult scene like censored sex scene in any movie, nude scene or pornography. Another part deals with finding scenes like crime scene, gore scene, violent scene, etc.

### 3.2.1 Training

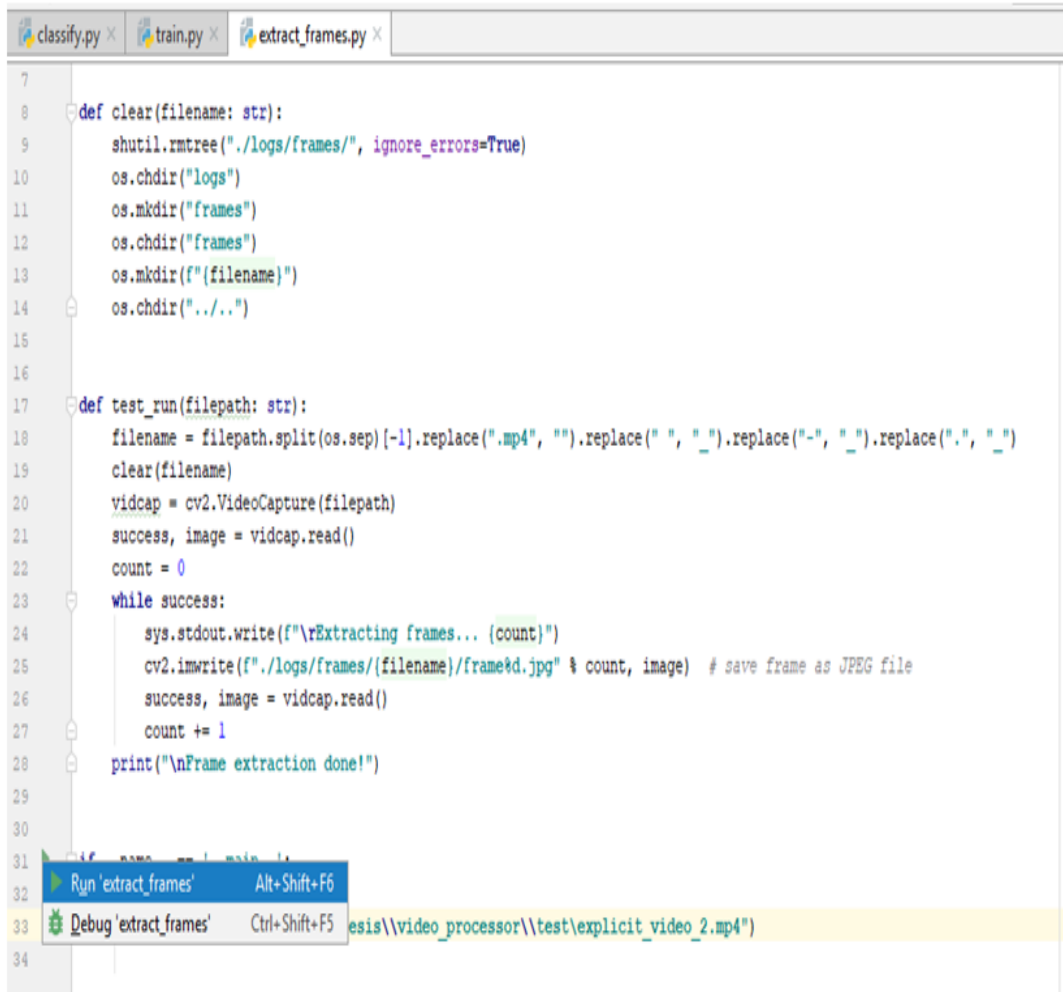
As mentioned before, we used CNN to detect any explicit content in any video. To do that, at first, we trained the model with over 50,000 frames of images (last count - 68,961 frames) that must be considered explicit [13]. We even trained the model with a considerable amount of non-explicit images (last count – 1209 frames) to know the difference between any explicit and non-explicit image. The number is pretty low in the non-explicit training procedure since the training mainly focuses on the explicit content and anything that is not considered to be explicit can be snipped into the non-explicit category easily. Most of the data has been taken from various online sources over the internet and some have been manually added to the model after extracting frames from an explicit video. Fig. 3.2, Fig. 3.3, Fig. 3.4 and Fig. 3.5 shows an exempt of how the model has been trained by the frames extracted from an explicit video. The model now recognizes all these frames from that video as inappropriate. Therefore, if there are any similar scenes in any other video the user wants to give as input and check, the model will identify the explicit content and pass the frames on to the next step of the iteration.

The training can also be done using internet images that are considered to be gory or inappropriate. Fig. 3.6 shows what the frames look like after they have been fed into the system as an example of inappropriate scenes.

The accuracy in the training is improved by the following methods shown in Fig. 3.7, Fig.3.8 and Fig. 3.9.

### 3.2.2 Blurring

After the training has been done, the system is ready to be used on user input videos. A path directory of a file to be analyzed is set in the "extractframes.py" and the code is run to extract all the frames from the input video [16]. After the frames have been extracted, these are



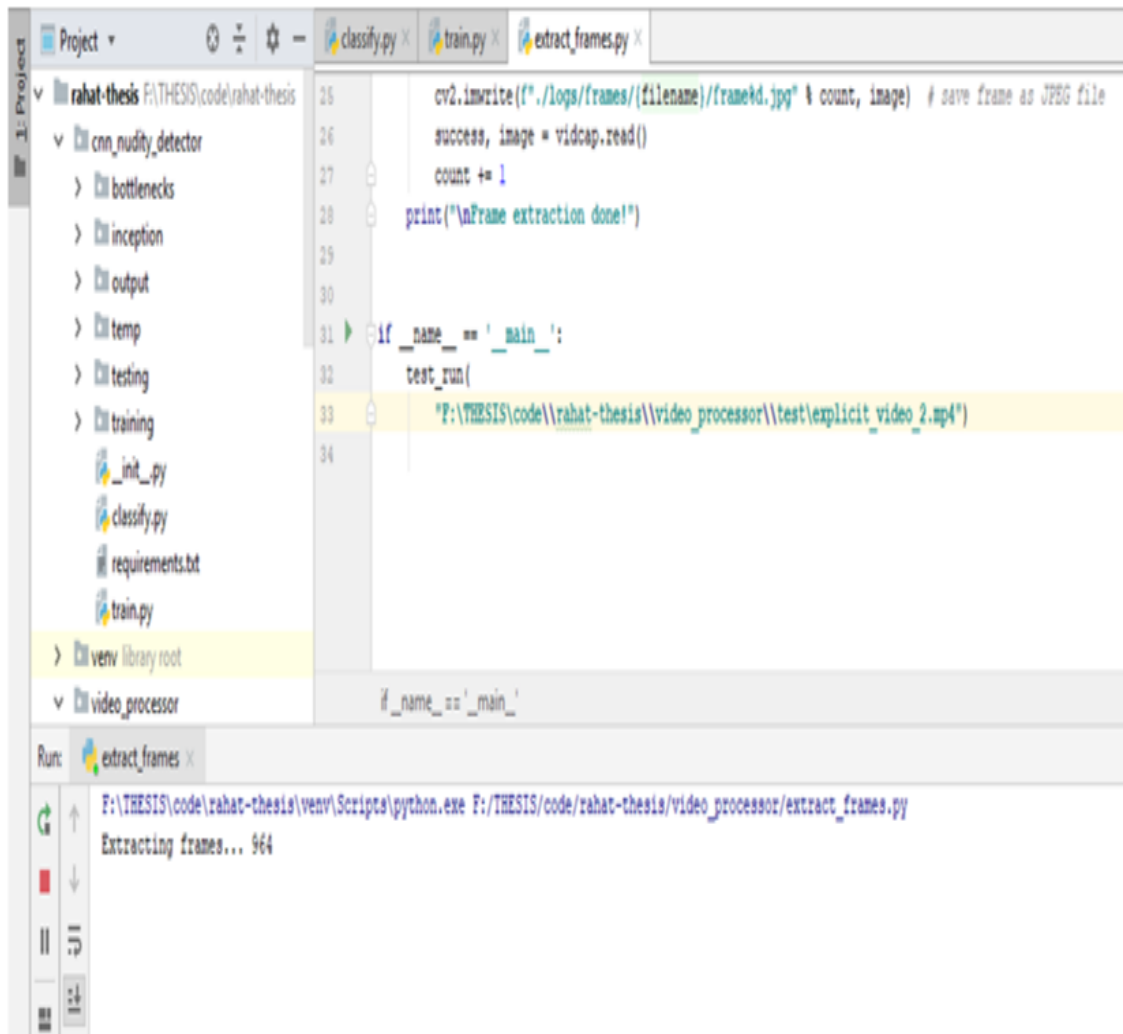
The image shows a Python IDE window with three tabs: 'classify.py', 'train.py', and 'extract\_frames.py'. The 'extract\_frames.py' tab is active, displaying the following code:

```
7
8 def clear(filename: str):
9     shutil.rmtree("./logs/frames/", ignore_errors=True)
10    os.chdir("logs")
11    os.mkdir("frames")
12    os.chdir("frames")
13    os.mkdir(f"{filename}")
14    os.chdir("../..")
15
16
17 def test_run(filepath: str):
18     filename = filepath.split(os.sep)[-1].replace(".mp4", "").replace(" ", "_").replace("-", "_").replace(".", "_")
19     clear(filename)
20     vidcap = cv2.VideoCapture(filepath)
21     success, image = vidcap.read()
22     count = 0
23     while success:
24         sys.stdout.write(f"\rExtracting frames... (count)")
25         cv2.imwrite(f"./logs/frames/{filename}/frame%d.jpg" % count, image) # save frame as JPEG file
26         success, image = vidcap.read()
27         count += 1
28     print("\nFrame extraction done!")
29
30
31
32
33
34
```

A context menu is open over the code, showing two options:

- Run 'extract\_frames' (Alt+Shift+F6)
- Debug 'extract\_frames' (Ctrl+Shift+F5) esis\\video\_processor\\test\\explicit\_video\_2.mp4)

Fig. 3.2 Extraction of frames

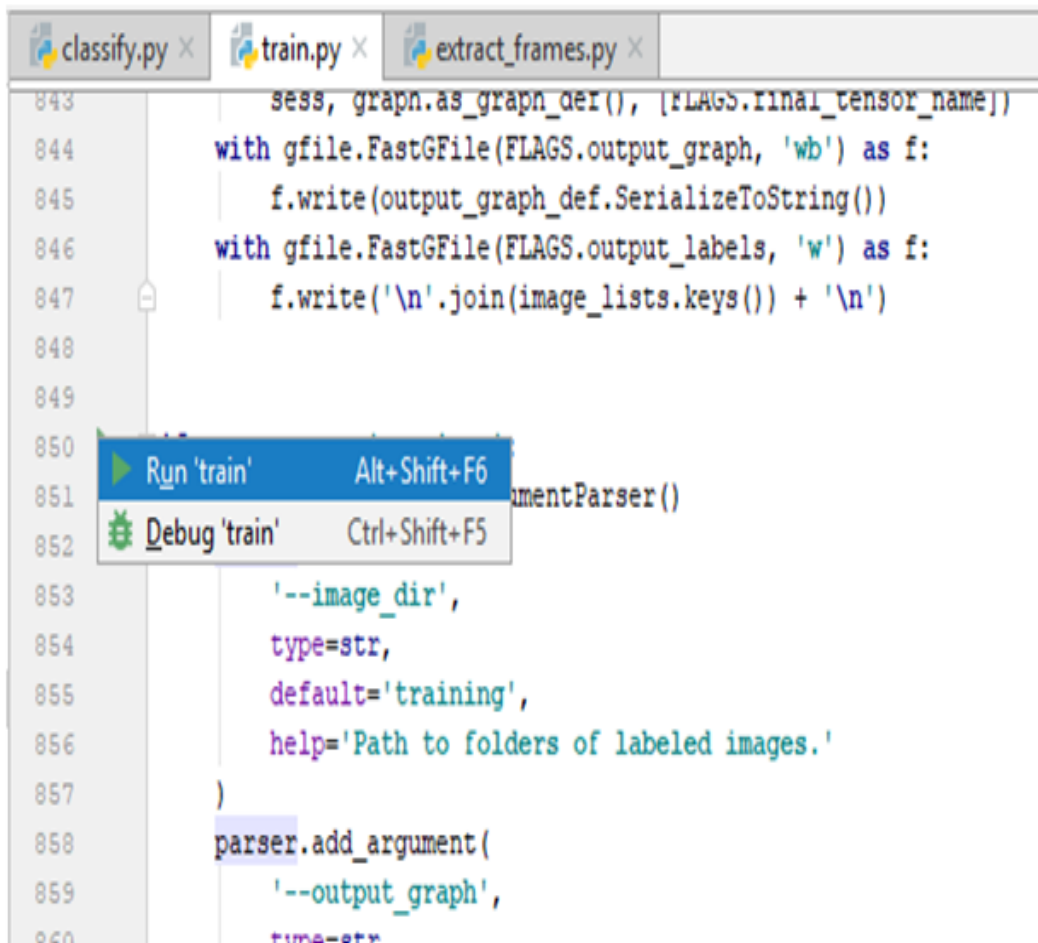


The image shows a screenshot of an IDE with a project named 'rahat-thesis'. The left sidebar shows a file tree with folders like 'cnn\_nudity\_detector', 'bottlenecks', 'inception', 'output', 'temp', 'testing', 'training', and 'video\_processor'. The main editor displays the code for 'extract\_frames.py' with the following content:

```
25 cv2.imwrite(f"./logs/frames/{filename}/frame%d.jpg" % count, image) # save frame as JPEG file
26 success, image = vidcap.read()
27 count += 1
28 print("\nFrame extraction done!")
29
30
31 if __name__ == '__main__':
32     test_run(
33         "F:\THESIS\code\rahat-thesis\video_processor\test\explicit_video_2.mp4")
34
```

The bottom of the IDE shows a terminal window with the command: `F:\THESIS\code\rahat-thesis\venv\Scripts\python.exe F:\THESIS\code\rahat-thesis\video_processor\extract_frames.py` and the output: `Extracting frames... 564`.

Fig. 3.3 Extracting frames



```
843     sess, graph.as_graph_def(), [FLAGS.final_tensor_name])
844     with gfile.FastGFile(FLAGS.output_graph, 'wb') as f:
845         f.write(output_graph_def.SerializeToString())
846     with gfile.FastGFile(FLAGS.output_labels, 'w') as f:
847         f.write('\n'.join(image_lists.keys()) + '\n')
848
849
850     ▶ Run 'train'      Alt+Shift+F6
851     🐛 Debug 'train'   Ctrl+Shift+F5
852
853     '--image_dir',
854     type=str,
855     default='training',
856     help='Path to folders of labeled images.'
857 )
858 parser.add_argument(
859     '--output_graph',
860     type=str
```

Fig. 3.4 training the model



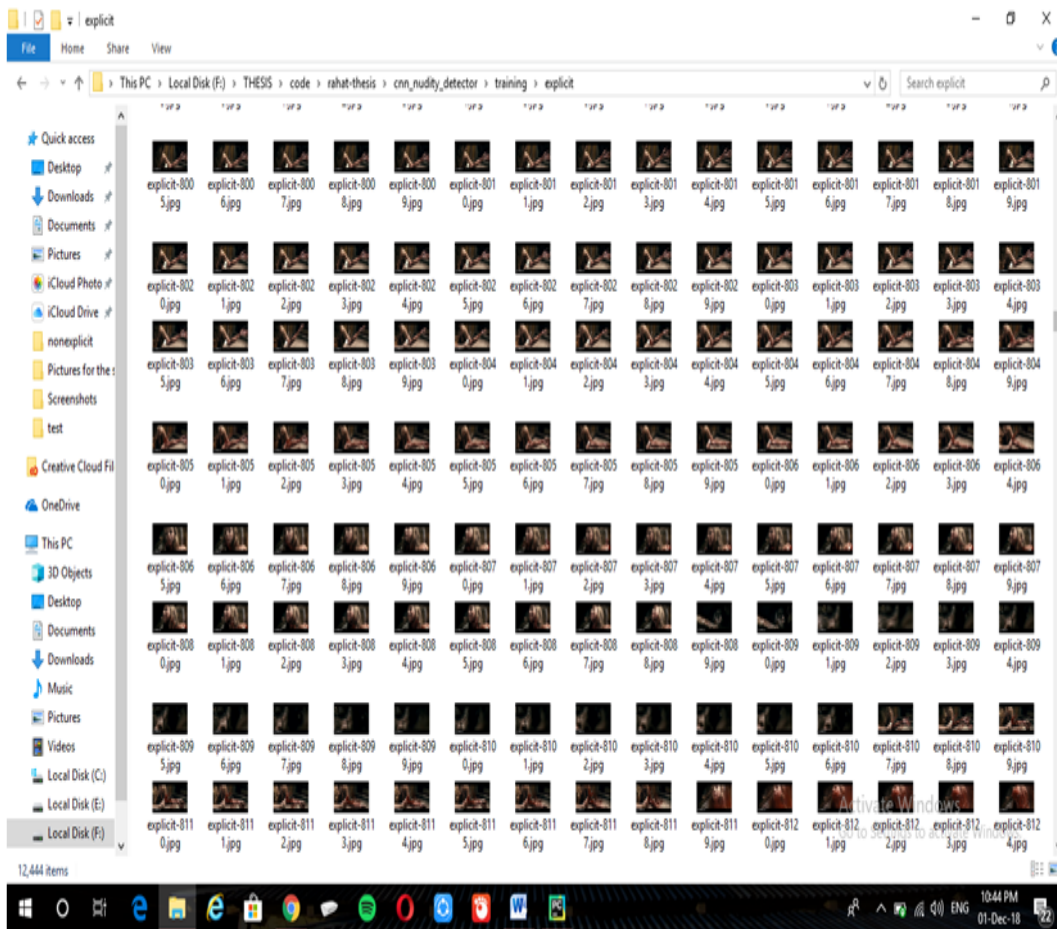


Fig. 3.5 Frames extracted

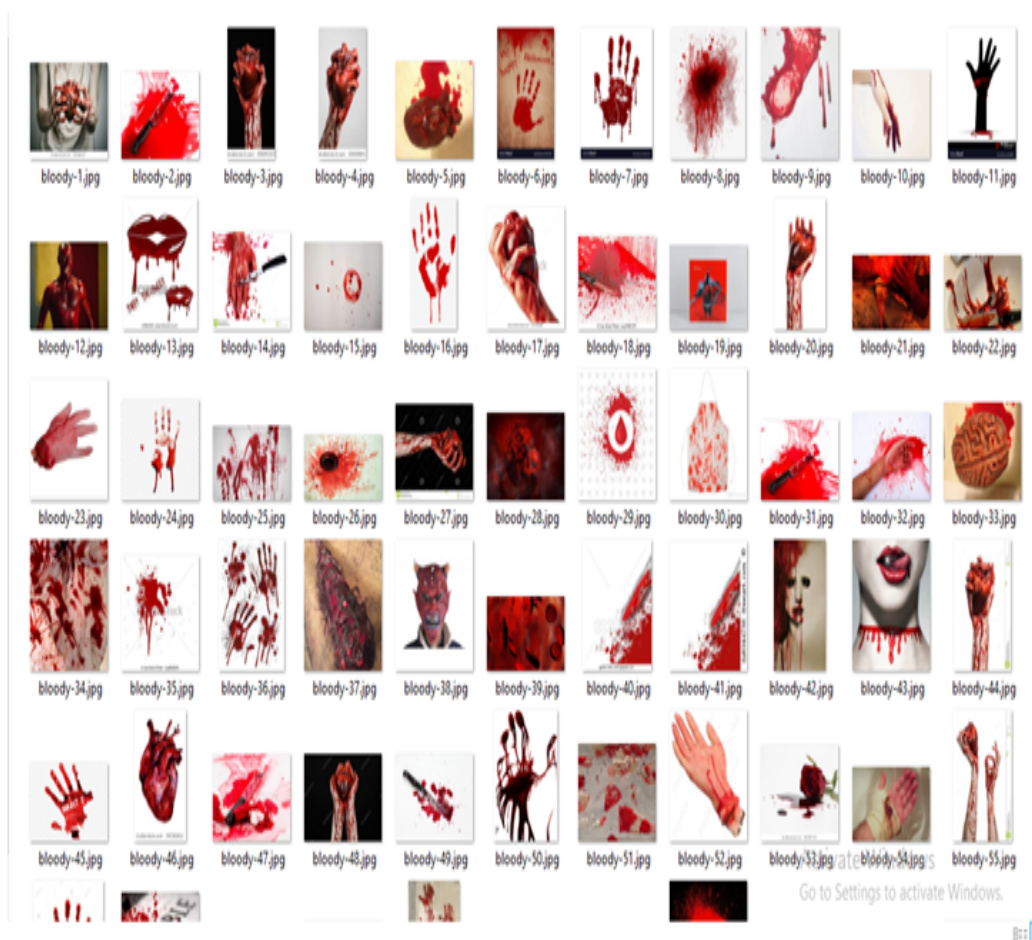


Fig. 3.6 Gore dataset

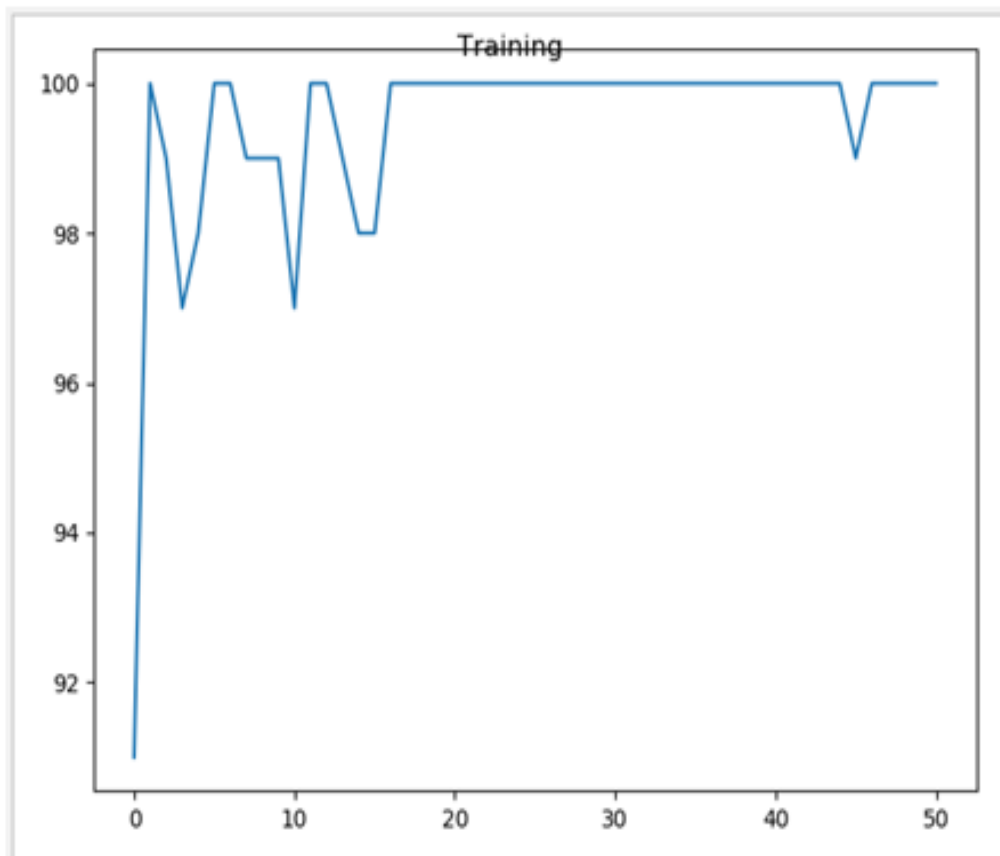


Fig. 3.7 Training curve of the algorithm

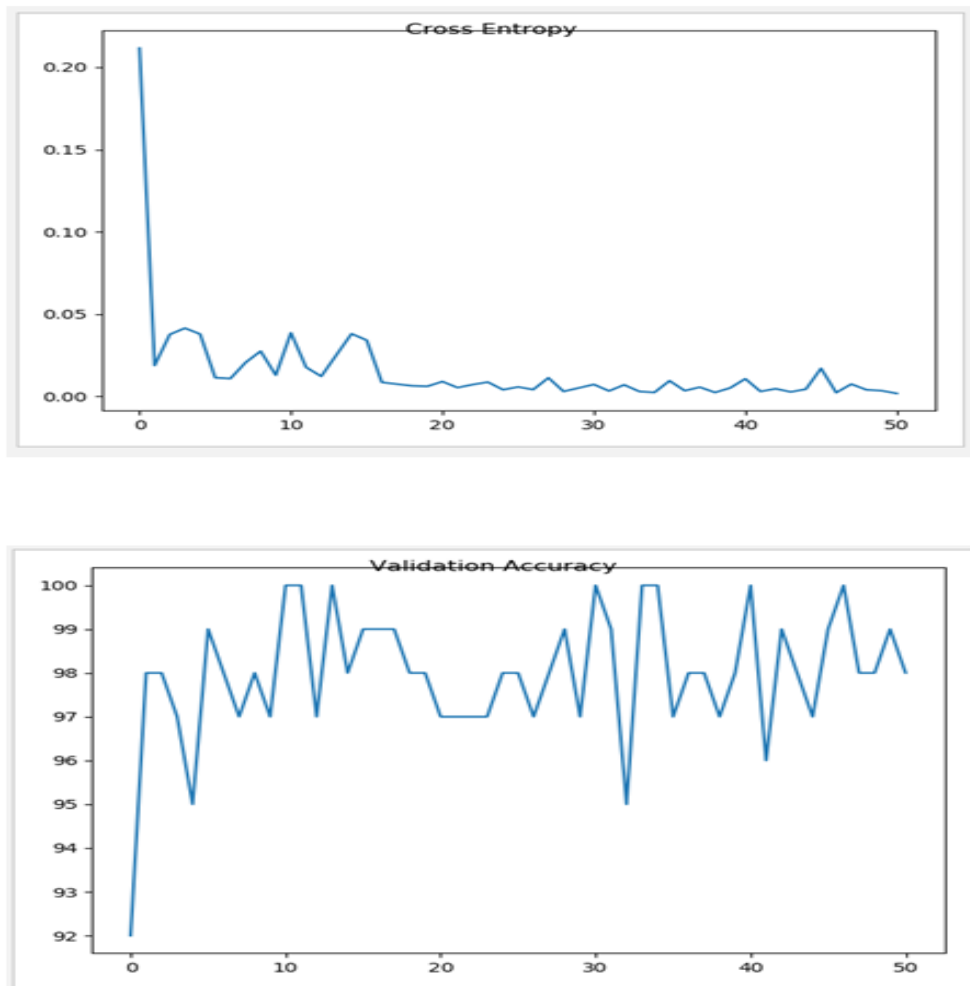
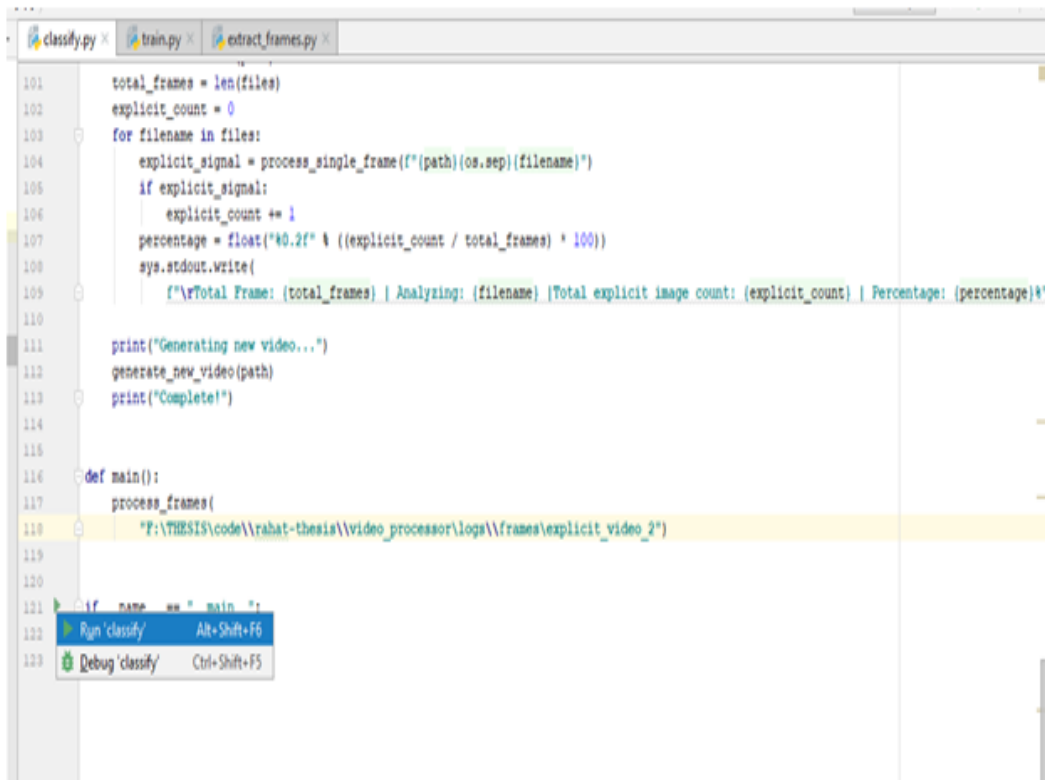


Fig. 3.8 The first image shows the cross entropy curve and the second image shows the validation accuracy curve



Fig. 3.9 The above curve shows the overlapping curve of training and validation accuracy



```

101 total_frames = len(files)
102 explicit_count = 0
103 for filename in files:
104     explicit_signal = process_single_frame(f"{path}{os.sep}{filename}")
105     if explicit_signal:
106         explicit_count += 1
107     percentage = float("%.2f" % ((explicit_count / total_frames) * 100))
108     sys.stdout.write(
109         f"\rTotal Frame: {total_frames} | Analyzing: {filename} | Total explicit image count: {explicit_count} | Percentage: {percentage}%"
110     )
111
112     print("Generating new video...")
113     generate_new_video(path)
114     print("Complete!")
115
116
117 def main():
118     process_frames(
119         "F:\THESIS\code\rahaf-thesis\video_processor\logs\frames\explicit_video_2")
120
121
122 if __name__ == "__main__":
123     Run 'classify' Alt+Shift+F6
124     Debug 'classify' Ctrl+Shift+F5

```

Fig. 3.10 Classify.py

saved in another file directory and those frames are now eligible to be classified as explicit and non-explicit. Therefore, the code "classify.py" is then run setting the file directory path as that of the frames that have already been extracted. The Fig. 3.10 shows how the transition looks on screen.

Every frame is analyzed whether it is explicit or not. The threshold set for the image to be considered as explicit is 78% for the model we built. Therefore, any image that crosses that threshold is considered as explicit and is blurred out using Pillow. Fig. 3.11 shows how the frames those were detected as inappropriate have been blurred.

All the frames are then rendered into a video and a new video file is then created which is completely explicit scene free. Thus, improving the video watching experience for any heart



Fig. 3.11 Blurred frames After Classifying

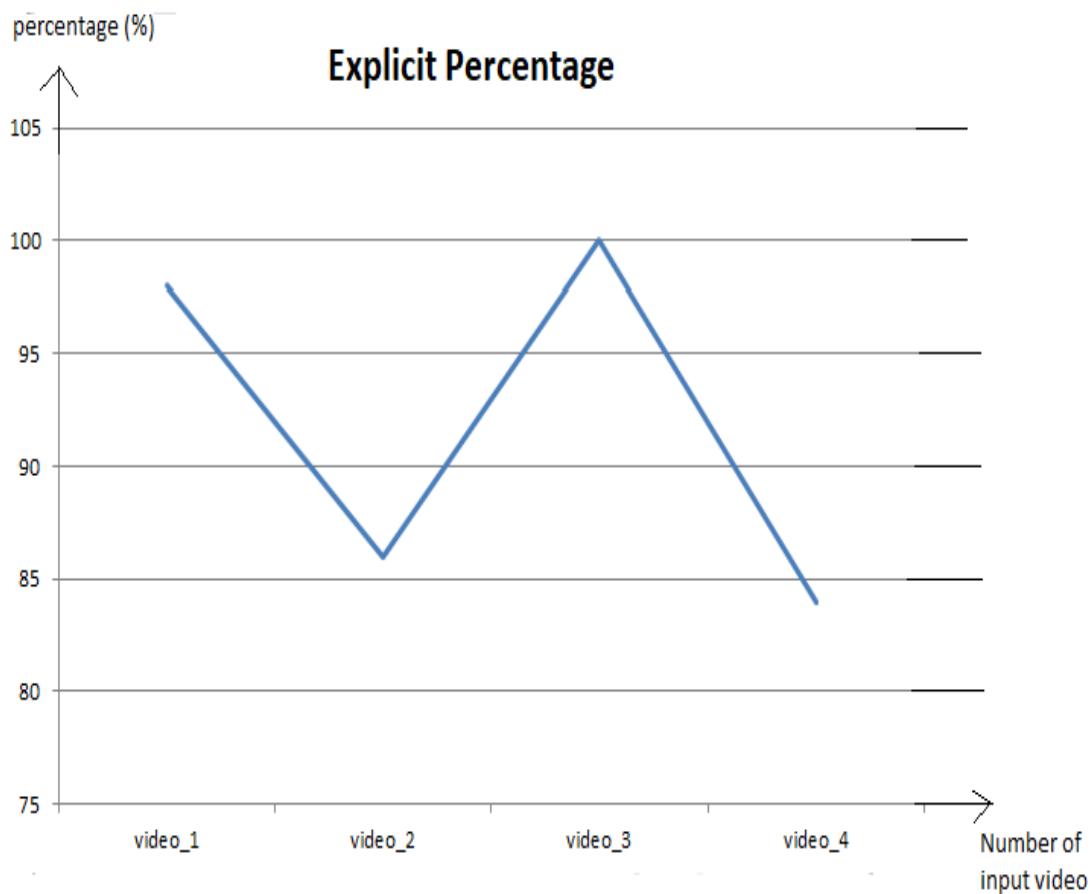


Fig. 3.12 Explicit Percentage Curve of Input Videos

patient who cannot watch any violent or gore scenes. It can also help parents to have better control over what their children are watching.

### 3.2.3 Results and Comparisons

The number of videos that are tested on the model are over 100. Four videos were taken as input. Some videos are 30 seconds, some are 45 seconds, some are 50 seconds. It can be concluded that the videos are approximately 1 minute long. The cumulative results of explicit percentages of each video have been shown in Fig. 3.12.

In Fig. 3.13, the results have been shown in a graphical representation to make easy comparison with other algorithms that are available for similar purposes.



Other methods to detect nudity and gore scenes from a video include HOG (Histogram of Oriented Gradients) and SIFT (Scalable Invariant Feature Transform) [10]. However, CNN is better and thus we have used CNN for our model.

The HOG feature is based on first order image gradients. The image gradients are pooled into overlapping orientation bins in a dense manner. HOG is:

- Based on first order image gradients pooled in orientation bins.
- Dense (evaluated all over the image).
- Hand engineered and no learning algorithm for it.

SIFT is similar to HOG only that SIFT is specifically a 128 dimensional vector that summarizes or describes a “16×16” window patch. The SIFT is obtained by dividing the “16×16” window into “4×4” bins. Each bin has 8 orientation bins or channels. So that makes the dimensionality of SIFT descriptor equal to “4×4×8 = 128”. SIFT is:

- Based on first order gradients.
- Course, that is, it is evaluated around scale invariant feature points obtained using the Difference of Gaussian key point detector. There is a dense variant known as the dense-SIFT.
- Hand engineered and thus does not learn the representation by itself, it is hard coded.

While CNN is a hierarchical deep learning architecture which is based on repeated convolutional operations and repeatedly filter the signal at each stage [4]. The filters are trainable, that is, they learn to adapt to the task at hand during learning. CNNs are:

- Trainable feature detectors which makes them highly adaptive. That is why they can achieve high accuracy levels in most applications such as image recognition. They can be trained end-to-end [5].
- Mainly supervised deep learning models motivated by the primary visual cortex with alternating layers of convolutions and pooling layers [19].
- They can learn low-level features similar to SIFT and HOG features from training examples alone, that is amazing [12]. Thus one can minimize feature engineering when it comes to using CNNs.

Thus SIFT and HOG features are low-level features which don't make use of hierarchical layer-wise representation learning while the CNN is a hierarchical deep learning model which is able to model data at more and more abstract representations.

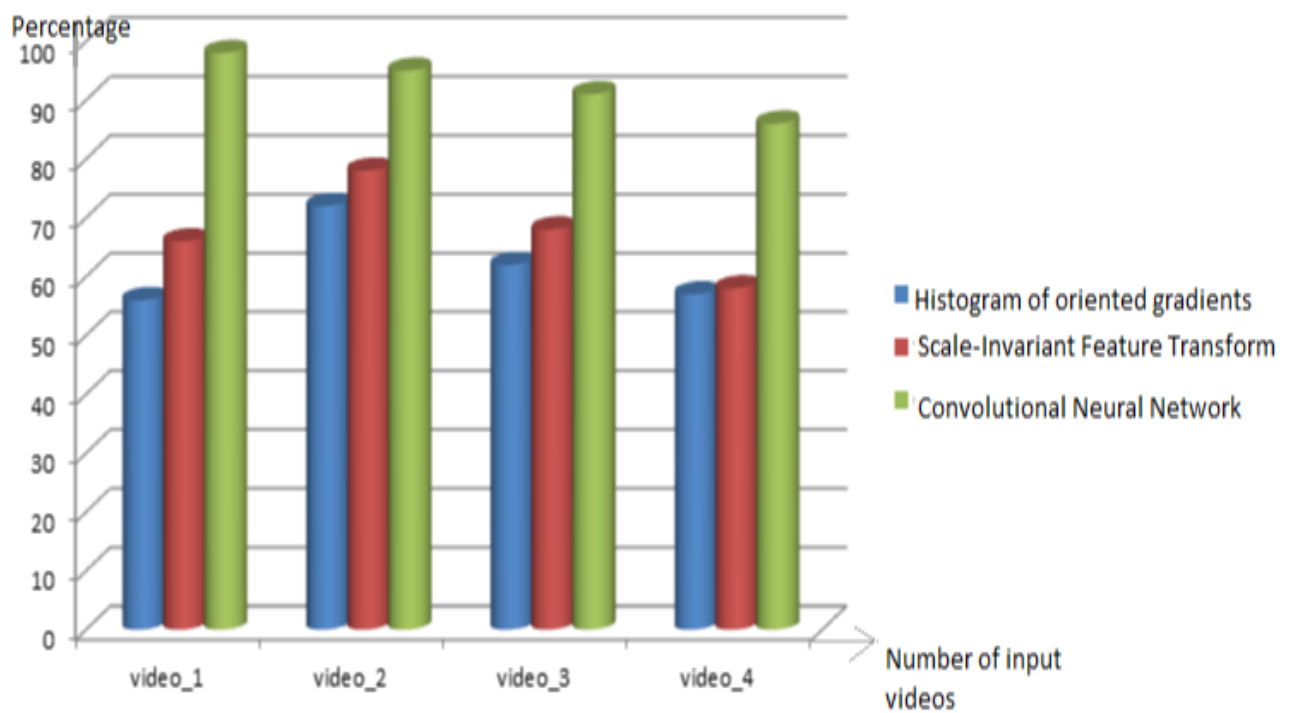


Fig. 3.13 Difference in accuracy of the three models.

# Chapter 4

## CONCLUSION

### 4.1 Limitations

Nudity detection and object detection only works for coloured images. These detections do not work for gray scale images. In this era, this limitation should not be a problem as all the images are coloured. For the inappropriate scenes to be fully detected, it is required that the scenes under detection are fully visible in the screen. If this is not the case, the results can be incorrect as the algorithms will fail to detect the inappropriate scenes. If people are wearing skin-colored clothes, then the results will lead to false positive conclusions.

### 4.2 Future Work

It would be very interesting to see if it is possible to detect various human poses and from those poses detect their action. Then it would be possible to scan human clothes and degree of nudity. It would be a new challenge for the system to identify explicit content by analyzing unlimited variations in clothing and human figure for different age group. We will also try to extract various features from our training data to train our system so that it can detect sexually explicit or gore content more accurately. If our system gives us satisfactory result using feature in neural network then we can also update our system for detecting violent action in a video.

Moreover, we plan to make an application for our model on an android platform using Android Studio. This will make life easier for people as they will be able to carry our platform and use it on any mobile phone device. The platform will also be improved by training the model with more data and in turn make the iterations and detections more accurate.



# References

- [1] Ap-Apid, R. (2005). An algorithm for nudity detection. In *5th Philippine Computing Science Congress*, pages 201–205.
- [2] Carlsson, A., Eriksson, A., and Isik, M. (2008). Automatic detection of images containing nudity: Image detection using artificial neural networks and statistical methods.
- [3] D’Haro, L. F., Banchs, R. E., Leong, C. K., Daven, L. G. M., and Yuan, N. T. (2017). Automatic labelling of touristic pictures using cnns and metadata information. In *Signal and Image Processing (ICSIP), 2017 IEEE 2nd International Conference on*, pages 292–296. IEEE.
- [4] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.
- [5] Elisseeff, A. and Paugam-Moisy, H. (1997). Size of multilayer networks for exact learning: analytic approach. In *Advances in Neural Information Processing Systems*, pages 162–168.
- [6] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- [7] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.
- [8] Karn, U. (2016). An intuitive explanation of convolutional neural networks. *The Data Science Blog*.
- [9] Lawrence, S., Giles, C. L., and Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. In *AAAI/IAAI*, pages 540–545. Citeseer.
- [10] Lee, J. Y. and Yoo, S. I. (2002). An elliptical boundary model for skin color detection. In *Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology*.
- [11] Liu, F., Lin, G., and Shen, C. (2015). Crf learning with cnn features for image segmentation. *Pattern Recognition*, 48(10):2983–2992.
- [12] Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H., and Yoshimoto, M. (2012). Architectural study of hog feature extraction processor for real-time object detection. In *Signal Processing Systems (SiPS), 2012 IEEE Workshop on*, pages 197–202. IEEE.

- 
- [13] Morris, R. J. and Rubin, L. D. (1991). Technique for object orientation detection using a feed-forward neural network. US Patent 5,060,276.
- [14] Nack, F., Schiphorst, T., Obrenovic, Z., KauwATjoe, M., De Bakker, S., Rosillio, A. P., and Aroyo, L. (2007). Pillows as adaptive interfaces in ambient environments. In *Proceedings of the international workshop on Human-centered multimedia*, pages 3–12. ACM.
- [15] Okada, H. and Rosenberg, J. D. (2002). Skin area detection for video image systems. US Patent 6,343,141.
- [16] Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International journal of computer vision*, 38(1):15–33.
- [17] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.
- [18] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [19] Tahmasebi, P. and Hezarkhani, A. (2011). Application of a modular feedforward neural network for grade estimation. *Natural resources research*, 20(1):25–32.
- [20] Zheng, H., Daoudi, M., Jedynek, B., LIFL-INT, M., and Lille, E.-T. (2003). Adult image detection using statistical model and neural network. *Electronic Letters on Computer Vision and Image Analysis*, 4(2):1–14.