PROJECT ON VIDEO SHOT BOUNDARY DETECTION

REPORT OF MAJOR PROJECT SUBMITTED OF

PARTIAL FULFILLMENT FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATION

SANTOSH CHANDRA MANDAL

Registration No.: 151170510039 OF 2015-2016

University Roll No.: 11701015038

UNDER THE SUPERVISION OF

Mr. Biswanath Chakraborty

Assistant Professor, Department of CA

RCC Institute of Information Technology

अमम् विना न किमपि साध्यम्

AT

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Affiliated to Maulana Abul Kalam Azad University of Technology

Canal South Road, Beliaghata, Kolkata – 700015

May, 2018

RCC INSTITUTE OF INFORMATION TECHNOLOGY
KOLKATA-700015, INDIA

CERTIFICATE

The report of the project titled Video Shot Boundary Detection submitted by **Mr. Santosh Chandra Mandal** (Roll No.: 11701015038 of MCA 6th Semester 2018) has been prepared under my supervision for the partial fulfilment of the requirements for **MCA** degree in **Maulana Abul Kalam Azad University of Technology**. The report is hereby forwarded.

Countersigned by

Arup Kumar Bhattacharjee
HOD and Asst. Professor, Dept. of CA
RCC Institute of Information Technology,
Kolkata-700015,India

Biswanath Chakraborty
Asst. Professor, Dept. of CA
(INTERNAL SUPERVISION)
RCC Institute of Information Technology,
Kolkata-700015,India

ACKNOWLEDGEMENT

I express my sincere gratitude to Mr, Biswanath Chakraborty, Asst. Prof. Department of Computer Application, RCCIIT, whose role as project guide was invaluable for the project. I am extremely thankful for the keen interest he took in advising me, for the books and reference materials provided for the moral support extended to us.

This project was initially started as minor project but we extended it in this phase with more contribution.

This entire project is jointly carried out with Mr. Biswajit Bakshi (Roll no.:11711416003, M. Tech in Information Technology, RCCIIT) under supervision of our Project Guide Mr. Biswanath Chakraborty Sir, .

I am also indebted to my Head of the Department (CA), Asst. Prof. Arup Kumar Bhattacharjee sir for his unconditional help and inspiration.

Date: _______________

Santosh Chandra Mandal
Reg. No.:151170510039 of 2015-2016
Roll No.: 11701015038
MCA 6th Semester, RCCIIT

RCC INSTITUTE OF INFORMATION TECHNOLOGY
KOLKATA-700015, INDIA

CERTIFICATE OF ACCEPTANCE

The report of the Project titled **Video Shot Boundary Detection** submitted by **Santosh Chandra Mandal** (Roll No.: 11701015038 of MCA 6thSemester 2018) is hereby recommended to be accepted for the partial fulfilment of the requirements for MCA degree in Maulana Abul Kalam Azad University of Technology.

Name of the Examiner(s)	**Signature with Date**
1. ______________________	______________________
2. ______________________	______________________

PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, worksor inventions of another as if they were one’s own. I know that plagiarismnot only includes verbatim copying, but also the extensive use of anotherperson’s ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use of material found in textual sources and from the Internet.

2. I acknowledge and understand that plagiarism is wrong.

3. I understand that my project must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and theuse of quotations as set out in the Departmental Guide.

4. This assignment is my own work, or my group’s own unique group assignment. I acknowledge that copying someone else’s assignment, or partof it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.

5. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Date: _______________

____

Santosh Chandra Mandal
Reg. No.:151170510039 of 2015-2016
Roll No.: 11701015038
MCA 6th Semester, RCCIIT

CONTENT

Video Shot Boundary Detection

1. Abstract

Multimedia streams usage increases nowadays and that creates the scope of development of efficient and effective methodologies for manipulating different image databases storing this type of information. Any content-based access to video data always requires parsing of each video stream into its building blocks. Any video stream consists of a number of shots, each one is a sequence of frames pictured using a single camera. Transition from a shot to the next one means switching from one camera to another. The detection of these transitions, known as scene change or shot boundary detection, is the very first step in any video stream-analysis system. There are numbers of proposed techniques are available for solving the problem of shot boundary detection, but the major limitation to them are their inefficiency, lack of reliability and less trustworthy. The performance has a direct impact on the performance of all other stages as the reliability of the scene change detection stage is a very significant requirement. Here, proposes to learn shot boundary detection end-to-end, from pixels to final shot boundaries. For training such a model, we created our own dataset and automatically generated transitions such as cuts, dissolves and fades. Here we propose a Convolutional Neural Network (CNN) which is fully convolutional in time and efficiently analyse hours of videos. Also, we propose to use Euclidean Distance algorithm and Change Point Analysis algorithm to make the system more efficient and accurate in nature. With this architecture my method will obtain state-of-the-art results while running at an unprecedented speed. I outperform dissolve gradual detection, generate competitive performance for sharp detections and produce significant improvement in wipes. In a short, the experimental results achieve the high efficiency of the proposed system in detecting shot boundaries within different video shots.

2. Introduction

All digital video information consists of a series of many frames or images. Over the years image processing technology has developed comprehensive and complete measures and techniques to index, store, edit, retrieve, sequence and present video material. To develop any content-based manipulations on digital video stream information, this information must first be structured and broken down into different components. The basic structural building blocks are called shots and the boundaries between shots need to be determined automatically.

A shot in video stream information may be defined as continuous images (i.e. frames) from a single camera at a time. A shot boundary is defined the gap between two shots. A cut is a type of shot boundary where one shot abruptly changes to another shot. An example of a shot cut is where the last frame in one shot is followed by the first frame in the next. Examples of other different types of shot boundary are fades (where the frames of the shot gradually change from or to black), dissolves (where the frames of the first shot are gradually morphed into the frames of the second) or wipes (where the frames of the first shot are moved gradually in a horizontal or vertical direction into the frames of the second).

The main reason why automatic shot boundary detection is difficult is the fact that any kind of shot transition can be easily confused with camera and object motion which occurs in video anyway. A shot with much object motion throughout the frame such as a sports or action shot or a clip from a music video, can cause the false recognition of a shot boundary.

Conventionally, if there exist frames that are merged by the adjacent shots but belong to neither of them, the transition is called a gradual one; otherwise, it is called a cut.[1]

Figure-1: hard cut effect[1]

Figure-3: Wipe effect[1]

Figure-2: fade effect[1]

Figure-4: Dissolve effect[1]

3. Literature Survey

This paper is mostly concentrate on the work till done in respect of video shot boundary detection in different areas. **John S. Boreczky et al [1996]** proposed Comparison of video shot boundary detection techniques and present a comparative analysis of various shot boundary detection techniques and their variations including histograms, discrete cosine transform, motion vector, and block matching methods**. Patrick Bouthemy et al [1999]** proposed Unified Approach to Shot Change Detection and Camera Motion haracterization which describes an approach to partition a video document into shots by using image motion information, which is generally more intrinsic to the video structure itself. **A. Miene et al [2001]** presented Advanced and Adaptive Shot Boundary Detection techniques which are based on–feature extraction and shot boundary detection. First, three different features for the measurement of shot boundaries within the video are extracted. Second, detection of the shot boundaries based on the previously extracted features. **H. Y: Mark Liaoff et al [2002]** proposed a novel dissolve detection algorithm which could avoid the mis-detection of motions by using binomial distribution model to systematically determine the threshold needed for discriminating a real dissolve from global or local motions. **Jesús Bescós [2004],** proposed a detection of change of video shot(i.e. cut) in real time on MPEG22 online video there he describes a software module for video temporal segmentation that is enough capable of detecting abrupt transitions and all kinds of gradual transitions in real time.

Guillermo Cisneros et al [2005] proposed a document on A Unified Model for Video-Shot Transition Detection Techniques. The approach presented here focuses on the mapping of the space of distances between frames in a new decision space more suitable to achieve an independent thresholding of the sequence. **Liuhong Liang et al. [2005],** presented an Improved Trigger Limit Detection using video text information, in which various edge-based techniques have been proposed to detect abrupt firing limits to avoid the influence of common flashlights in many types of video, such as sports, news, entertainment and interview videos. **Daniel DeMenthon et al [2006]** proposed a document on trigger limit detection based on the correlation functions of video images. This document is based on the correlation functions of images in the videos. The cut detection is based on the so-called 2max ratio criterion in a sequential image buffer. Dissolution detection is based on the difference of image jump and linearity error in a sequential image buffer. **Kota Iwamoto and others [2007],** the detection of wipes and digital video effects was proposed based on an independent model of image boundary line characteristics pattern that is based on a new independent model of patterns. These models are based on the characteristics of the image boundary lines that divide the two image regions in the transition frames. **Jinhui Yuan et al [2008]** proposed a document on a trigger limit detection method for news video based on the segmentation and tracking of objects. It combines three main techniques: the method of comparison of partitioned histograms, segmentation of video objects and tracking based on wavelet analysis. The comparison of the partitioned histogram is used as the first filter to effectively reduce the number of video frames that need segmentation and object tracking. **Yufeng Li et al [2008]** proposed an article on Algorithm of detection of new shots based on the theory of the information. First, the characteristics of the color and texture are extracted by wavelet transform, then the difference between two successive frames that collide the mutual information of the color characteristic and the mutual information of matching of the texture characteristic is defined. The threshold is adjusted adaptively depending on the entropy of the Continuous frames and does not depend on the type of video and the type of shot. **Vasileios T. Chasanis et al [2009]** presented the detection of scenes in videos using clustering of shots and alignment of sequences. First the keyframes were extracted using a spectral clustering method using the fast global k-means algorithm in the clustering phase and also providing an estimate of the number of the keyframes. Then, the shots are grouped into groups using only the visual similarity as a function and are labeled according to the group assigned to them. **Jinchang Ren et al [2009]** proposed a document on detection of trigger limits in MPEG videos using local and global indicators that operate directly in the compressed domain. Several local indicators are extracted from the MPEG macroblocks, and Ada Boost is used for the selection and merging of features. The selected characteristics are then used to classify the candidate cuts in five subspaces by pre-filtering and rules-based decision making, then the global indicators of frame similarity are examined among cut-off frames of cut candidates using the phase correlation. of CC images. **Priyadarshinee Adhikari et al [2009]** proposed a document on Video Shot Boundary Detection. This document presents the recovery of video using detection of limit of shot. **LihongXu et al [2010]** proposed a paper on a new shot detection algorithm based on grouping. This article presents a novel trigger limit detection algorithm based on the K-means grouping. The extraction of the color feature is done first and then the difference of the video frames is defined. The video frames are divided into several different sub-clusters by performing K-means clusters. **Wenzhu Xu and others [2010]** proposed an article on a new shot detection algorithm based on graph theory. This article presents a trigger limit detection algorithm based on graph theory. Video frames are divided into several different groups through the realization of a theoretical graphics algorithm. Arturo Donate et al [2010] presented Detection of shooting limits in videos using a robust three-dimensional tracking. The proposal is to extract the highlighted features of a video sequence and track them over time to estimate the limits of the shots within the video. **Min-Ho Park et al [2010]** proposed a paper on the detection of efficient trigger limits using characteristics based on block movement. It is a measure of discontinuity in camera and an object / background movement for SBD is proposed based on the combination of two movement characteristics: the

modified displaced frame difference (DFD) and the block wise movement similarity. **Goran J. Zajić et al [2011]** proposed a document on detection of video trigger limits based on multifractal analysis. Low-level features (color and texture characteristics) are extracted from each frame in video sequence, then concatenated into feature vectors (FV) and stored in the feature matrix. The rows of matrix correspond to FV of frames of the video sequence, while the columns are time series of a particular FV component. **Partha Pratim Mohanta et al [2012],** proposed an article on a model-based trigger limit detection technique that uses frame transition parameters that is based on a formulated frame estimation scheme that uses the previous frame and the next frame. **Pablo Toharia et al [2012]** proposed an article on Shot Boundary detection using Zernike moments in multi-CPU multi-GPU architectures along with the different possible hybrid combinations based on Zernike moments. **Sandip T et al. [2012]** proposed a document on the video summary based on keyframes using the automatic threshold and the speed of correspondence of the edges. First, the Histogram difference of each frame is calculated, and then the edges of the candidate keyframes are extracted by the Prewitt operator.
Zhe Ming Lu et al [2013] present a fast video trigger limit detection based on SVD and pattern matching. It is based on the selection of segments and decomposition of singular values (SVD). Initially, the positions of the firing limits and the lengths of the gradual transitions are predicted using adaptation thresholds and most non-contour frames are discarded at the same time. **Sowmya R et al [2013]** proposed a document on Analysis and verification of summary video using Shot Boundary Detection. The analysis is based on the difference of the block-based histogram and the euclidean distance difference based on blocks for various block sizes. **Ravi Mishra et al [2014]** proposed an article on a "Comparative study of the block matching algorithm and the complex transformation of two trees for the detection of shots in videos". This article presents a comparison between the two detection methods in terms of several parameters, such as false rate, hit rate, failure rate tested in a set of different video sequences. **Wenjing Tong et al [2015]** proposed a document on trigger limit detection based on CNN and video annotation. This analysis is based on TAG frames generated by a CNN model. **Ahmed Hassanien et al [2017]** proposed a document on detection of large-scale, rapid and precise trigger limits through spatial-temporal convolutional neural networks. This analysis is based on exploiting Big Data to optimize both the accuracy and speed of two large data sets.

4. Few Keywords and Definitions

4.1 What is Video?

Digital video is audio-visual stream in a binary format. Information is presented as a sequence of digital data block, rather than in a continuous signal as analog information provides.

Digital video shows up on our screens but conceptually the same as the simpler to understand motion pictures, invented over couple of decades ago. Just like physical film and analog video, a digital video stream is made up of individual frames, each one representing a time slice/block of the scene. Film displays 24 frames/second, and an American video presents 30 frames/second, it is known as the frame rate. To get smoother video need to increase the number of frames in any given second. Digital video clips use frame rates from 12-30 frames per second, whereas 24 frames per second commonly used.

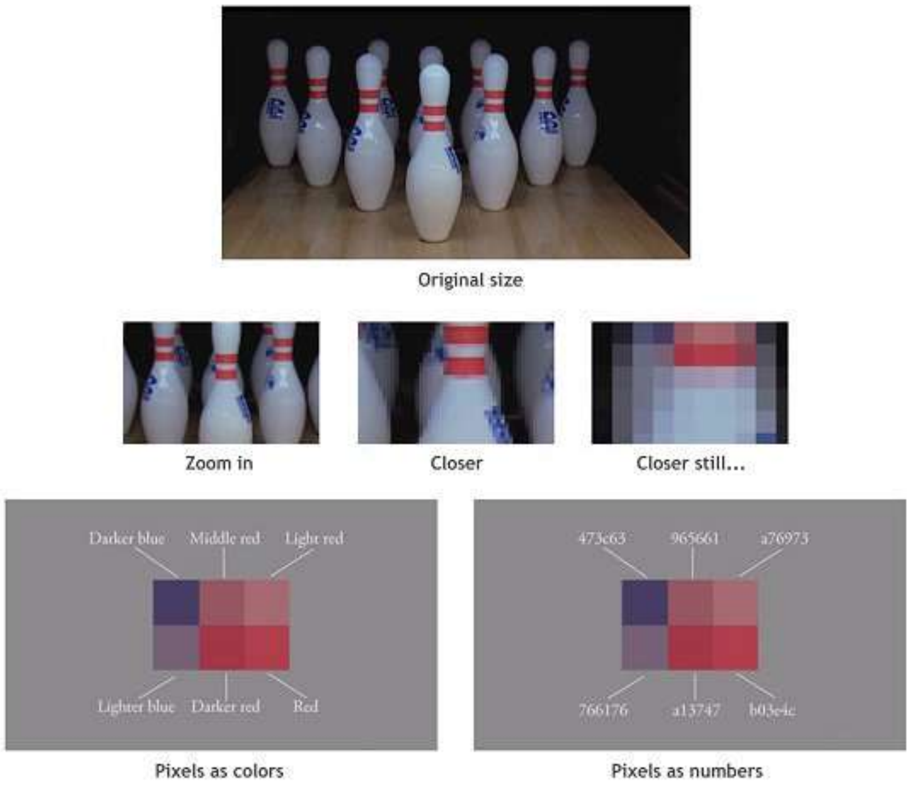


Figure-5 [19]

4.1.1. Hierarchical Structure of Video

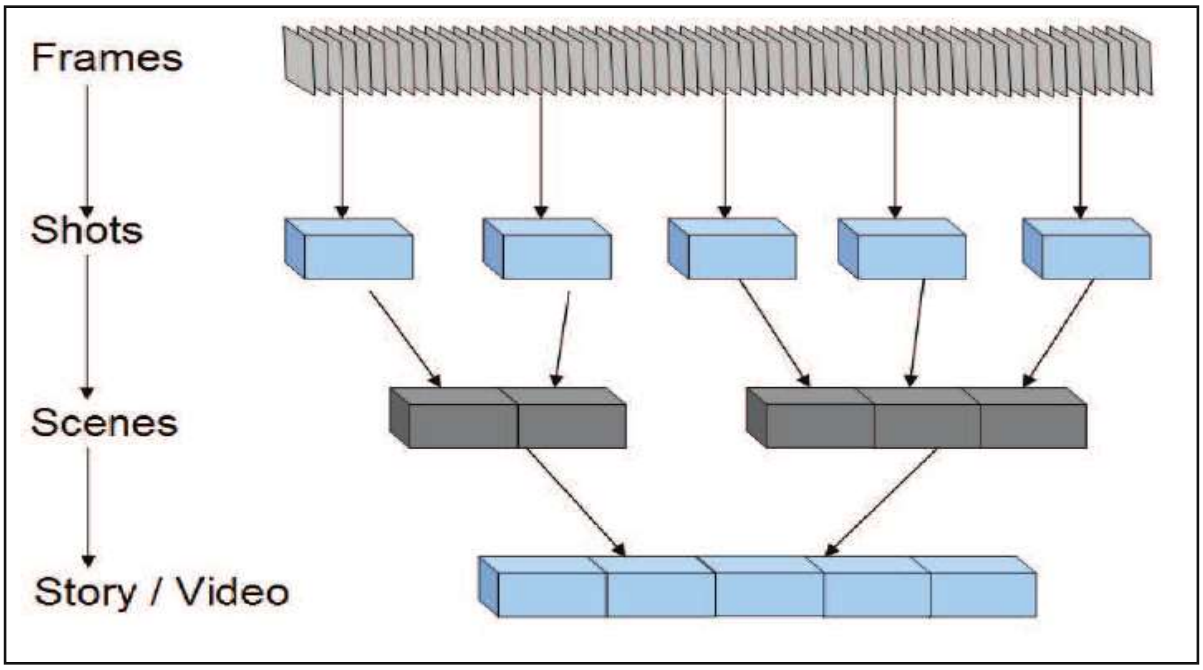


Figure-6 [22]

- **Scene:** A number of shots that form a semantic unit.

- **Shot:** All frames with in single camera action.
- **Frame:** One Static image from a series of static images constituting a video.

4.1.1.1. Types of Shot

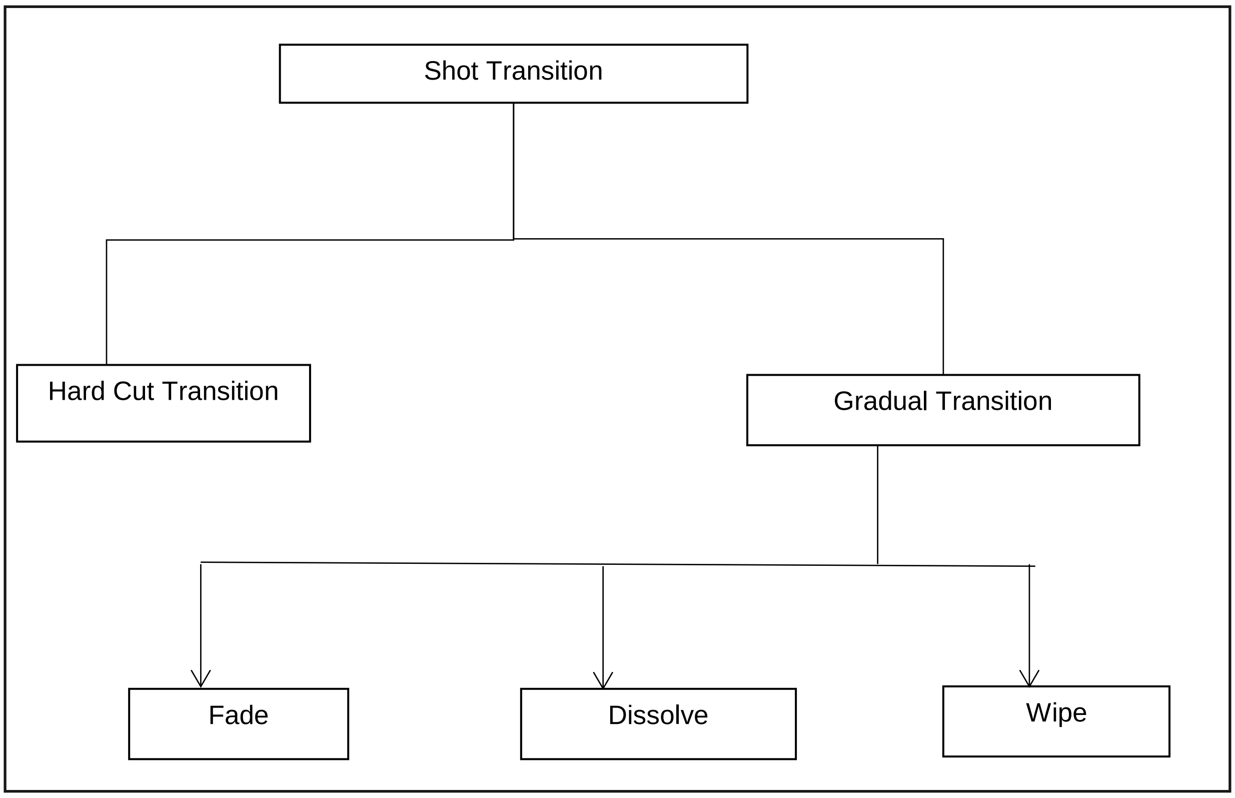


Figure-7

4.1.1.1.1. Hard Cut

Hard cut is the basic cutaway. The filmmaker is moves from the action to other things and then comes back to the action. Cutaways are used to edit out boring or add action to a sequence by changing the pace of the footage.

4.1.1.1.2. Fade

Two keywords, fade-in and fade-out usually signal the beginning or end of a scene, especially if the filmmaker fades to/from black. This is the most common scenario, of course, but fading to white has become trendy, too.

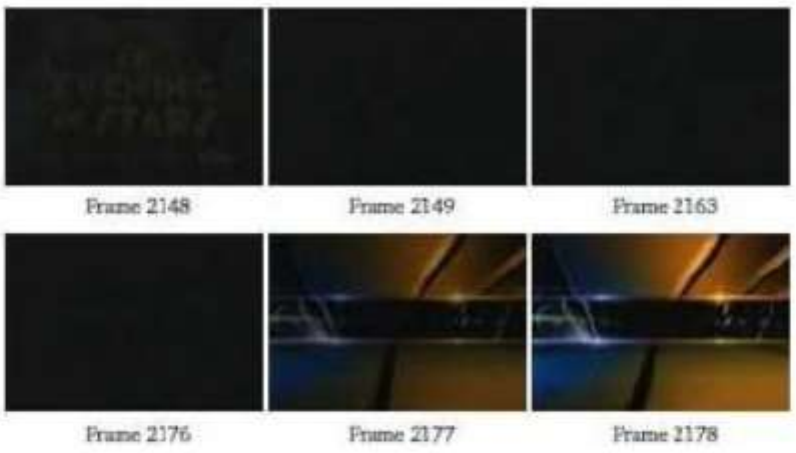


Figure-8: Fade – out gradual transition [18]

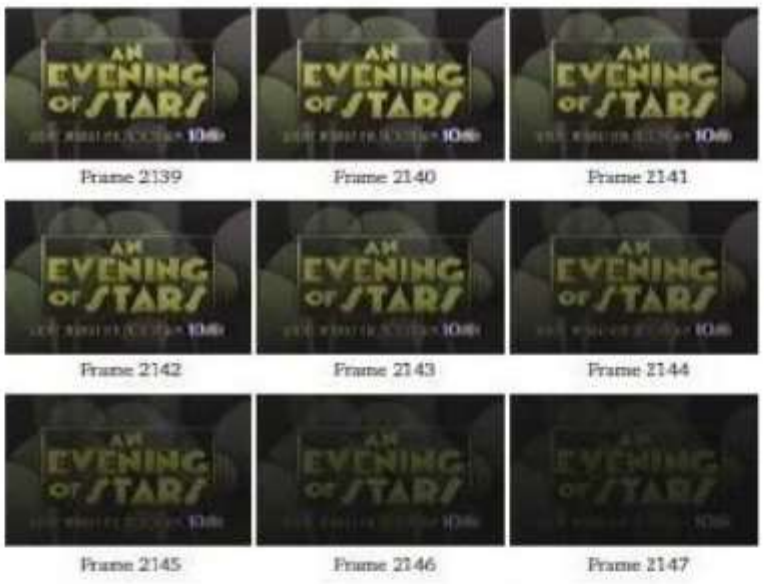


Figure-9: Fade – in gradual transition [18]

4.1.1.1.3. Dissolve

This is an editing technique where one clip seems to dissolve, or fade-in to the next. When the first clip is fading out, getting lighter and lighter, the second clip starts fading in, becoming more and more prominent/visible. The viewer is not aware of the transition as the process usually happens so subtly and so quickly.

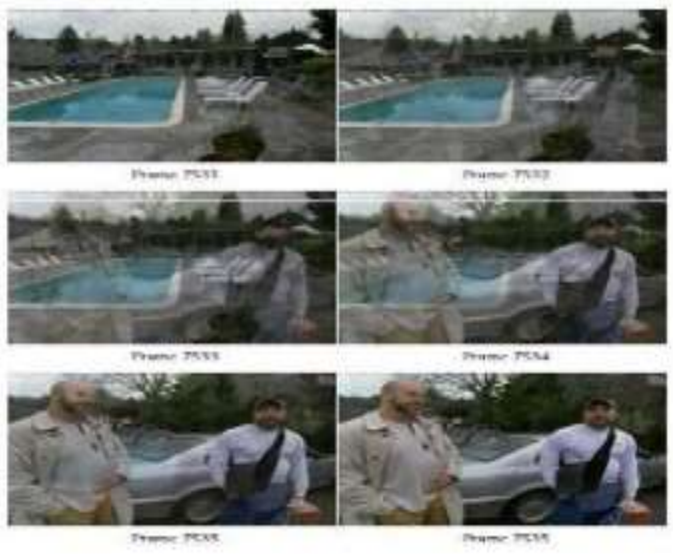


Figure-10: Dissolve transition [18]

4.1.1.1.4. Wipe

Figure-11: Wipe transition [18]

This wipe transition is the just opposite of the dissolve transition in that it draws attention to itself. The best example of the wipe is what's known as the Iris Wipe, which you usually find in silent films. Other commonly used wipe shapes includes stars, diamonds, and the old turning clock.

4.2. Boundary Detection

- For any video indexing, browsing, retrieval, representation and other video analysis technologies video shot boundary detection is the first and fundamental step.
- To identify the transition between every two adjacent shots, video shot boundary detection is the process.

4.2.1. Earlier Various Approaches to Shot Detection

4.2.1.1 Pixel Comparison

In Pixel Comparison, if there two frames are significantly different and to count the number of pixels that change in value more than any threshold. This method is sensitive to camera motion. We note that manually adjusting the threshold is unlikely to be practical. This commonly used matching process duplicates the process used to extract motion vectors from an image pair. Then the pixel differences for each region were sorted, and then the weighted sum of the sorted region differences. The Gradual transitions were detected by generating a cumulative difference measures from consecutive values of the image differences. During dissolves and fades, this chromatic image assumes a reasonably constant value.

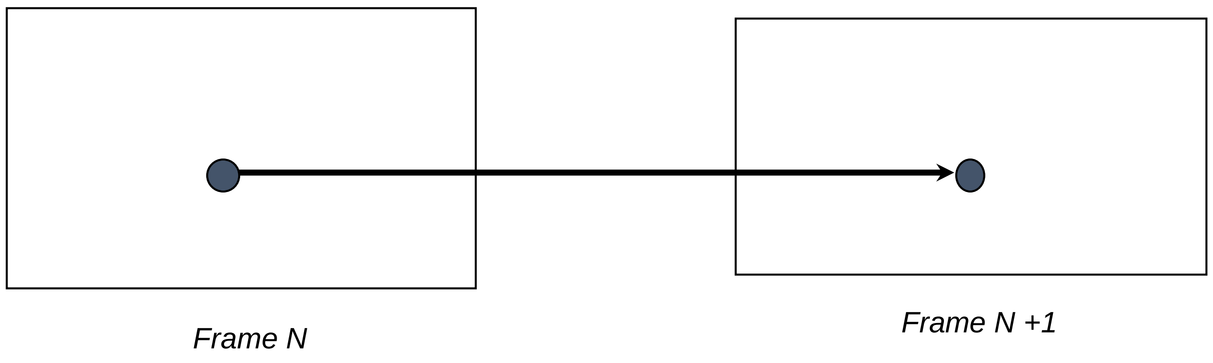


Figure-12

4.2.1.2. Histogram Comparison

The histogram comparison methods are the most common method used to detect shot boundaries. The simplest histogram method computes, two types, gray level or color level histograms of any of the two images. If the bin-wise difference between the two histograms is above a threshold, a shot boundary is assumed.

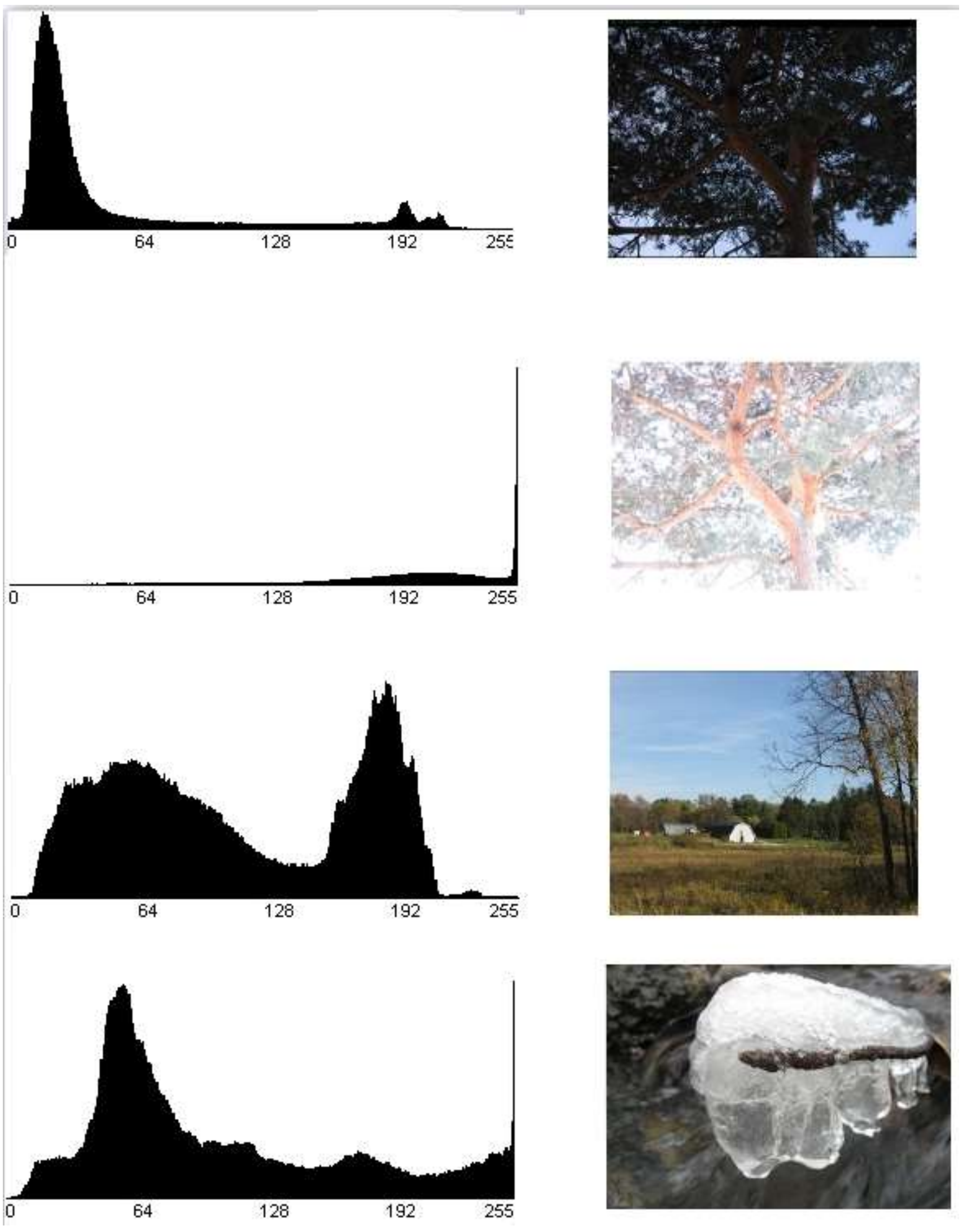


Figure-13: Histogram Comparison [20]

4.2.1.3. Statistical Differences

The statistical method is nothing but the idea of pixel differences by breaking the images into regions/blocks and comparing statistical measures of the pixels in those regions. It divides the frames into small regions. Then compares some of the few properties of every pixel in those regions between successive frames using the measurable statistical computation parameters.

4.2.1.4. Motion Vectors

MPEG compressed video sequences can also contains Motion vector information. The block matching performed as part of MPEG encoding based on compression efficiency and thus often selects inappropriate vectors for image processing purposes.

4.2.1.5. Edge Change Ratio

To detect, if any new edges have entered the image or if some old edges have disappeared, commonly uses the edges of successive aligned frames are detected first and then the edge pixels are paired with nearby edge pixels in the other image.

The main reason why automatic shot boundary detection is difficult is the fact that any kind of shot transition can be easily confused with camera and object motion which occurs in video anyway. A shot with much object motion throughout the frame such as a sports or action shot or a clip from a music video, can cause the false recognition of a shot boundary.

In this project we report on shot boundary detection by using Convolutional Neural Network(CNN).

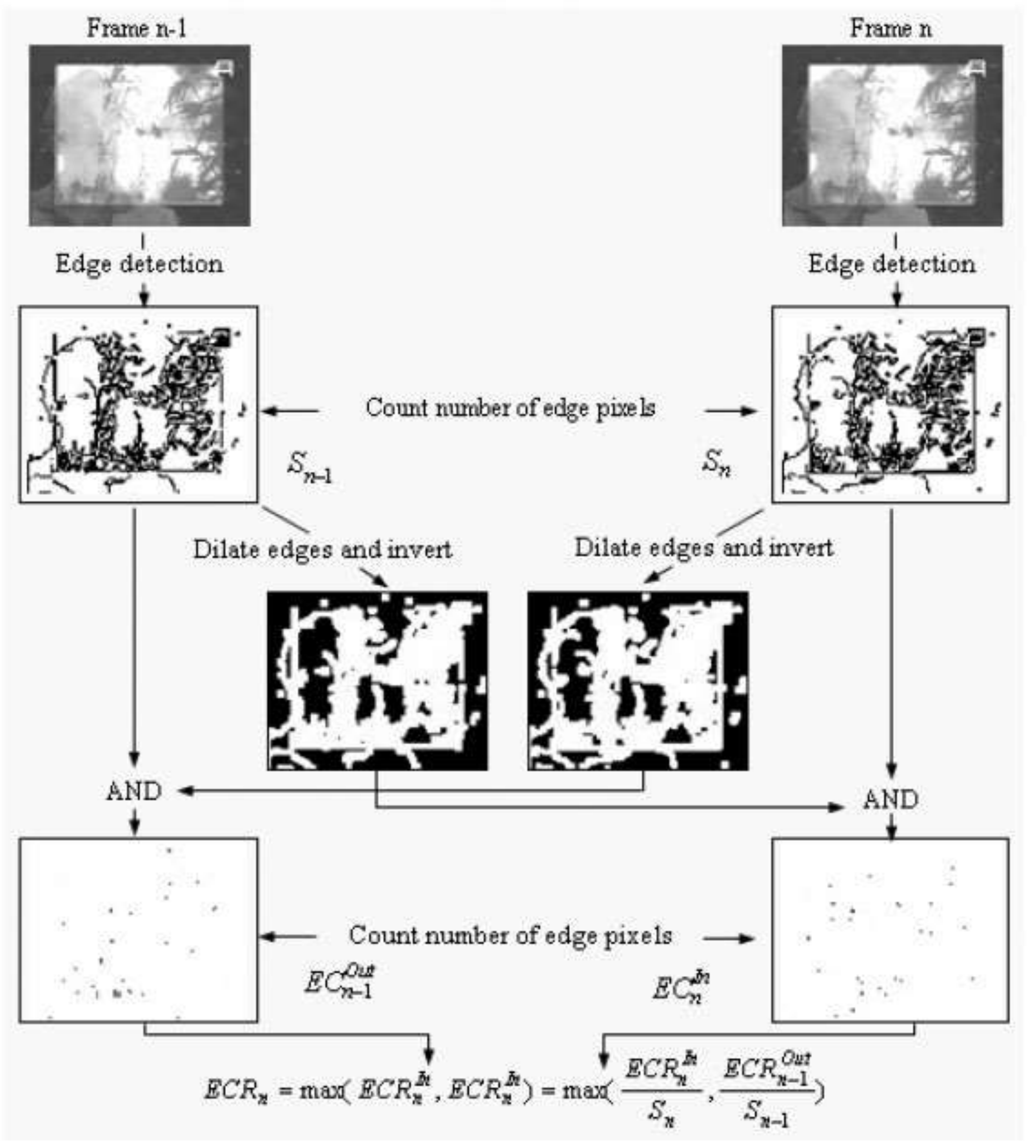

Frame n-1
Frame n
Edge detection
Edge detection
Count number of edge pixels
S_{n-1}
S_n
Dilate edges and invert
Dilate edges and invert
AND
AND
Count number of edge pixels
EC_{n-1}^{Out}
EC_n^{In}
$ECR_n = \max(ECR_n^{In}, ECR_n^{In}) = \max(\frac{ECR_n^{In}}{S_n}, \frac{ECR_{n-1}^{Out}}{S_{n-1}})$

Figure-14: Edge Change Ratio [21]

4.3. Deep Learning

As a part of Artificial Intelligence (AI) technology world, deep learning really stands behind numerous innovations: both voice and image recognition, self-car driving, security surveillance system etc. Nowadays, this technology has occupied multiple aspects of human lives. Generates such a huge interest in both machine and deep learning technologies is based on their advantages.

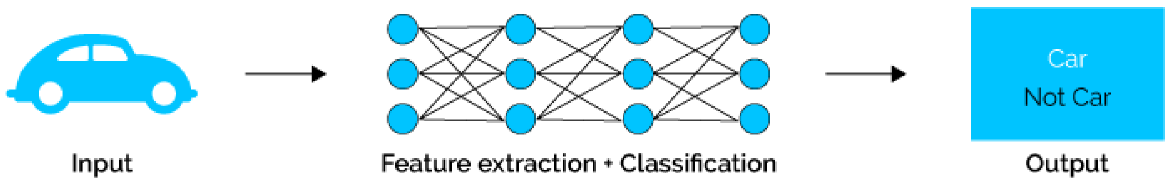


Figure-15: Deep Learning [23]

4.3.1. The Definition of Deep Learning

A set of machine learning algorithms that model high-level views in data using architectures called Deep learning. However, a deep learning technology is based on Artificial Neural Networks(ANNs). These ANNs helps continuously growing amounts of data and constantly receive learning algorithms to increase the efficiency of training processes. The larger data volumes are make the process more efficient. With the time passing, a neural network covers a growing number of levels, the training process is called «deep». The higher its productivity is the «deeper» this network penetrates.

4.3.2. How Deep Learning Works

Two main phases are there in a deep machine learning process: training and inferring. Labeling of large amounts of data and determine their matching characteristics called training process. The system compares these characteristics and memorizes(learns) them to make correct conclusions when it faces similar data stream next time.
The following stages are there is a deep learning training process:

1. ANNs ask a set of binary false/true questions or.
2. Extracting numerical values from data blocks.
3. Classifying data according to the answers received.
4. Labeling Data.

During the inferring phase, the deep learning AI makes conclusions and label new unexposed data using their previous knowledge.

4.3.3. Deep Learning and Machine Learning?

Deep learning is a kind of traditional machine learning. However, classical machine learning is the extraction of new knowledge from a large data array loaded into the machine. Users first formulate the machine training rules and then correct errors made by a machine. This approach eliminates a negative overtraining effect frequently appearing in deep learning.
In machine learning, users provide both examples and training data to a machine to help the system make correct decision is called supervised learning.
Diversity between machine learning and deep learning:

- **Deep Learning** uses a lot of un-labelled training data to make appropriate conclusions whereas **Machine Learning** can use small data stream provided by users.
- Unlike **Machine Learning**, **Deep Learning** needs high-performance hardware.

- **Machine Learning** requires features to be accurately identified by users while Deep Learning creates new features by itself.
- **Machine Learning** divides tasks into small pieces and then combine received results into one conclusion while **Deep Learning** solves the problem on the end-to-end basis.
- In comparison with **Machine Learning**, **Deep Learning** needs much more time to train.
- Unlike **Deep Learning**, **Machine Learning** can provide enough transparency for its decisions.

The machine creates its functionality by itself as long as it is possible - the concept of deep learning explains. Deep learning applications use a hierarchical approach involving determining the most important characteristics to compare to infer.

4.3.4. Creating New Features

Ability to generate new features from limited series of features located in the training dataset is one of the main benefit of deep learning over various machine learning algorithms. Therefore, deep learning algorithms can create new tasks to solve current ones.
Data scientists can save much time on working with big data and relying on this technology as deep learning can create features without a human intervention. It allows the data scientists to use more complex sets of features in comparison with traditional machine learning software.

4.3.5. Advanced Analysis

Deep learning generates actionable results when solving data science tasks due to its improved data processing models. While machine learning works only with labeled data, deep learning supports unsupervised learning techniques that allow the system become smarter on its own. Due to the capacity to determine the most important features allows deep learning to efficiently provide data scientists with reliable and concise analysis results.

4.3.6. Deep Learning Challenges

Models of human abstract thinking used by deep learning rather than using it. Despite all its benefits, this technology has a set of significant disadvantages also.

4.3.7. Continuous Input Data Management

A training process is based on analyzing large amounts of data is there in deep learning. However, fast-moving and streaming input data provides little time for ensuring an efficient training process. That is why data scientists adapt their deep learning algorithms in a way neural networks can handle large amounts of continuous input data.

4.3.8. Ensuring Conclusion Transparency

Another important disadvantage of deep learning software is that it is not explicitly says the reason why it has reached a certain conclusion. Unlike in case of traditional machine learning, you cannot follow an algorithm to find out why your system has decided that it is a cat on a picture, not a dog. Have to revise the whole DL algorithm to correct that.

4.3.9. Resource-Demanding Technology

Deep learning technology is a quite high resource demanding technology as it requires more powerful GPU(Graphics processing unit)s, with large amounts of storage to train the models, etc. Apart from this technology needs more time to train in comparison with traditional machine learning process.
Though there are lots of disadvantages, deep learning discovers new improved methods of unstructured big data analytics for those with the intention to use it. The businesses can gain significant benefits from using deep learning within their tasks of data processing.

4.3.10. What is an Artificial Neural Network?

An ANN(artificial neural network) is based on the neural structure of the brain. It is also able to learn and perform tasks like the followings as classification, prediction, decision-making, visualization

An ANN consists of artificial neurons or processing elements and is organized in three consecutive interconnected layers: input, hidden that may include more than one layer, and output.

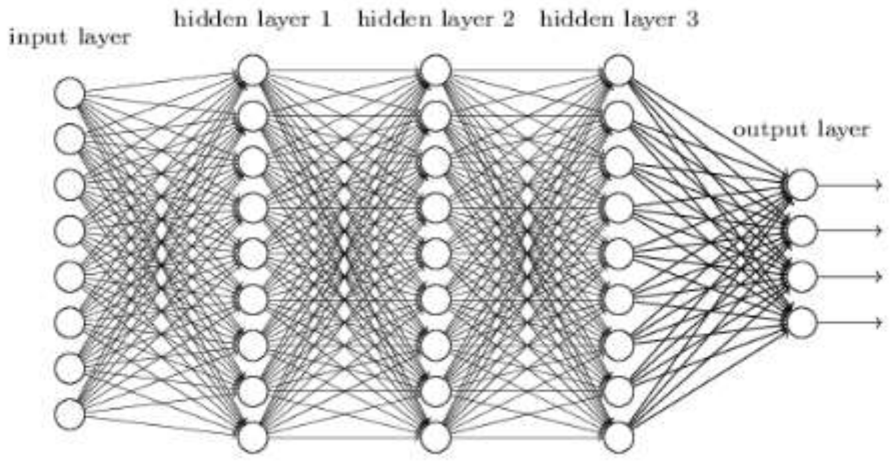


Figure-16: Artificial Neural Network[13]

The input layer contains input neurons those send information to the hidden layer. Then the hidden layer sends data to the output layer. Every neuron has weighted inputs (synapses), an activation function (defines the output given an input), and one output. Here synapses are the adjustable parameters that helps to convert a neural network to a parameterized system.

Why Neural Network

- A neural network can perform tasks that a linear program cannot does.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- No reprogrammed is required in neural network learning.
- It can be implemented in any application and can be implemented without any problem

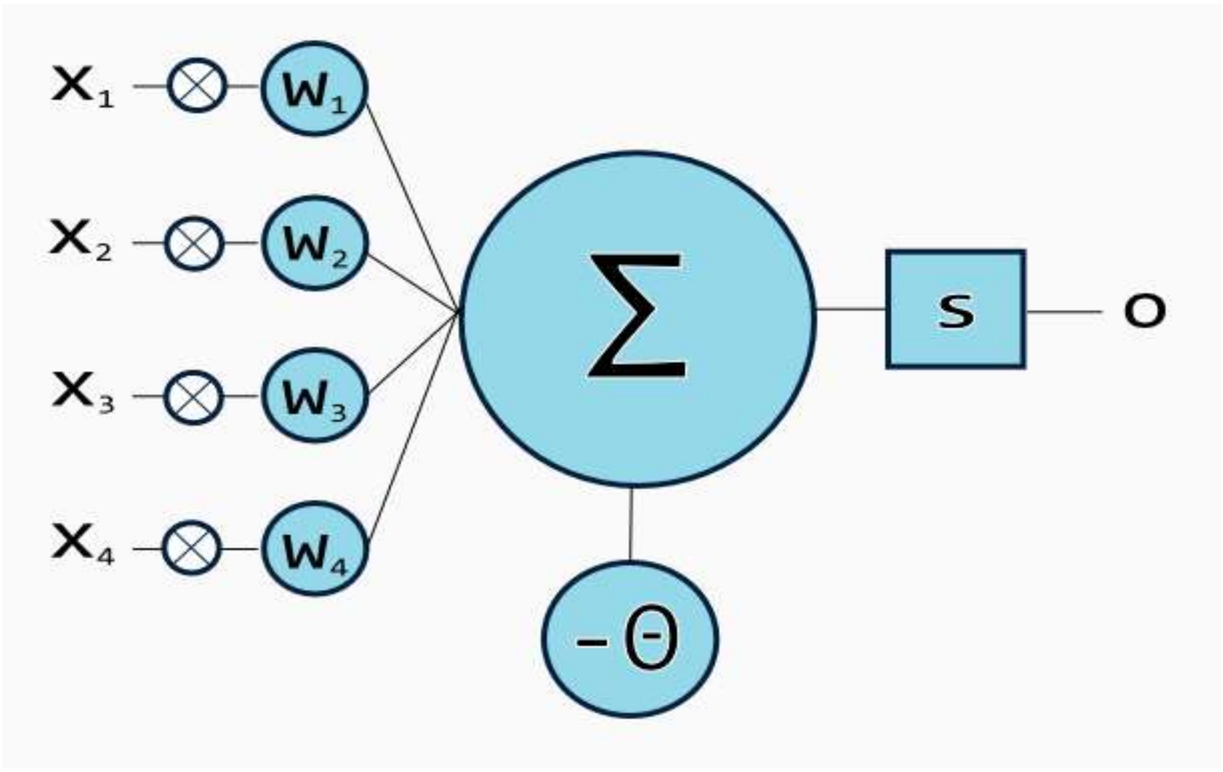


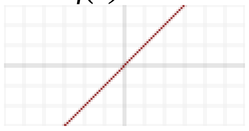
Figure-17: Artificial Neuron with four input[15]

Artificial neuron with four input

To get one output from the neuron, the weighted sum of the inputs produces the activation signal that is passed to the activation function. The commonly used activation functions are linear, step, sigmoid, tanh, and rectified linear unit (ReLu) functions.

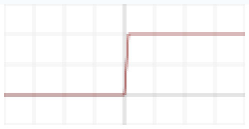
Linear function

f(x)=ax



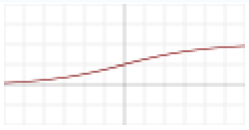
Step function

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$



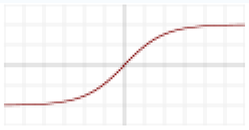
Logistic (Sigmoid) Function

$$f(x) = \frac{1}{1 + e^{-x}}$$



Tanh Function

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



Rectified linear unit (ReLu) function

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

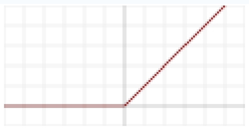


Figure-18[16]

The error of predictions is minimized and the network reaches a specified level of accuracy during training. The method mostly used to determine the error contribution of each neuron is called backpropagation that calculates the gradient of the loss function.
It is possible to make the system more flexible and more powerful by adding additional hidden layers. Artificial neural networks with multiple hidden layers between the input and output layers are called deep neural networks (DNNs), and they can model complex nonlinear relationships.

4.3.10.1. Feedforward Neural Network

First and simplest type of artificial neural network devised was the feed-forward neural network. It contains multiple neurons/nodes arranged in layers. Neurons/nodes from adjacent layers have connections between them. All these connections have weights associated with them.
An example of a feed-forward neural network is shown below.

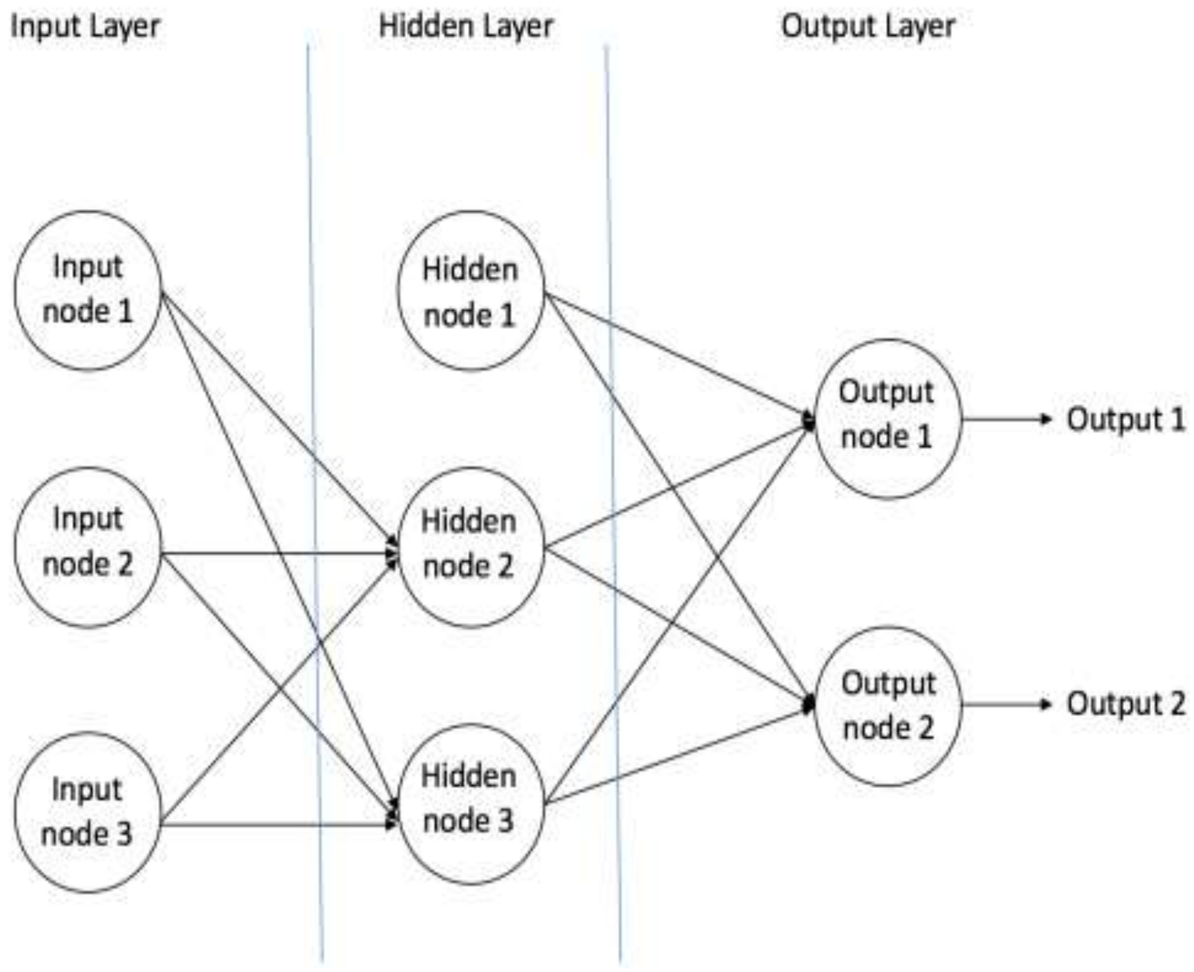


Figure-19: An example of feedforward neural network[16]

A feed-forward neural network can consist of three types of nodes:

1. **Input Nodes** – This node provides information from the outside world to the network. Together referred to as the "Input Layer". In any of the Input nodes, no computation is performed – they just pass on the information to the hidden nodes.
2. **Hidden Nodes** – The Hidden nodes have no direct connection with the outside world (hence the name "hidden"). They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer". While a feedforward network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers.
3. **Output Nodes** – These nodes are collectively referred to as the "Output Layer". Those are responsible for computations and transferring information from the network to the outside world.

In a feedforward network, the information moves in only forward direction – from the input nodes, through the hidden nodes and to the output nodes. There are no cycles or loops in the network.

Two examples of feedforward networks are given below:

1. **Single Layer Perceptron** – This is the simplest feedforward neural network and does not contain any hidden layer.
2. **Multi Layer Perceptron** – A Multi-Layer Perceptron has one or more hidden layers.

Figure-20 shows a multi-layer perceptron with a single hidden layer. Note that all connections have weights associated with them, but only three weights (w0, w1, w2) are shown in the figure.

Input Layer: The Input layer has three nodes. The Bias node has a value of 1. The other two nodes take X1 and X2 as external inputs (which are numerical values depending upon the input dataset). As discussed above, no computation is performed in the Input layer, so the outputs from nodes in the Input layer are 1, X1 and X2 respectively, which are fed into the Hidden Layer.

Hidden Layer: The Hidden layer also has three nodes with the Bias node having an output of 1. The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges).

Figure-20 shows the output calculation for one of the hidden nodes (highlighted). Similarly, the output from other hidden node can be calculated.

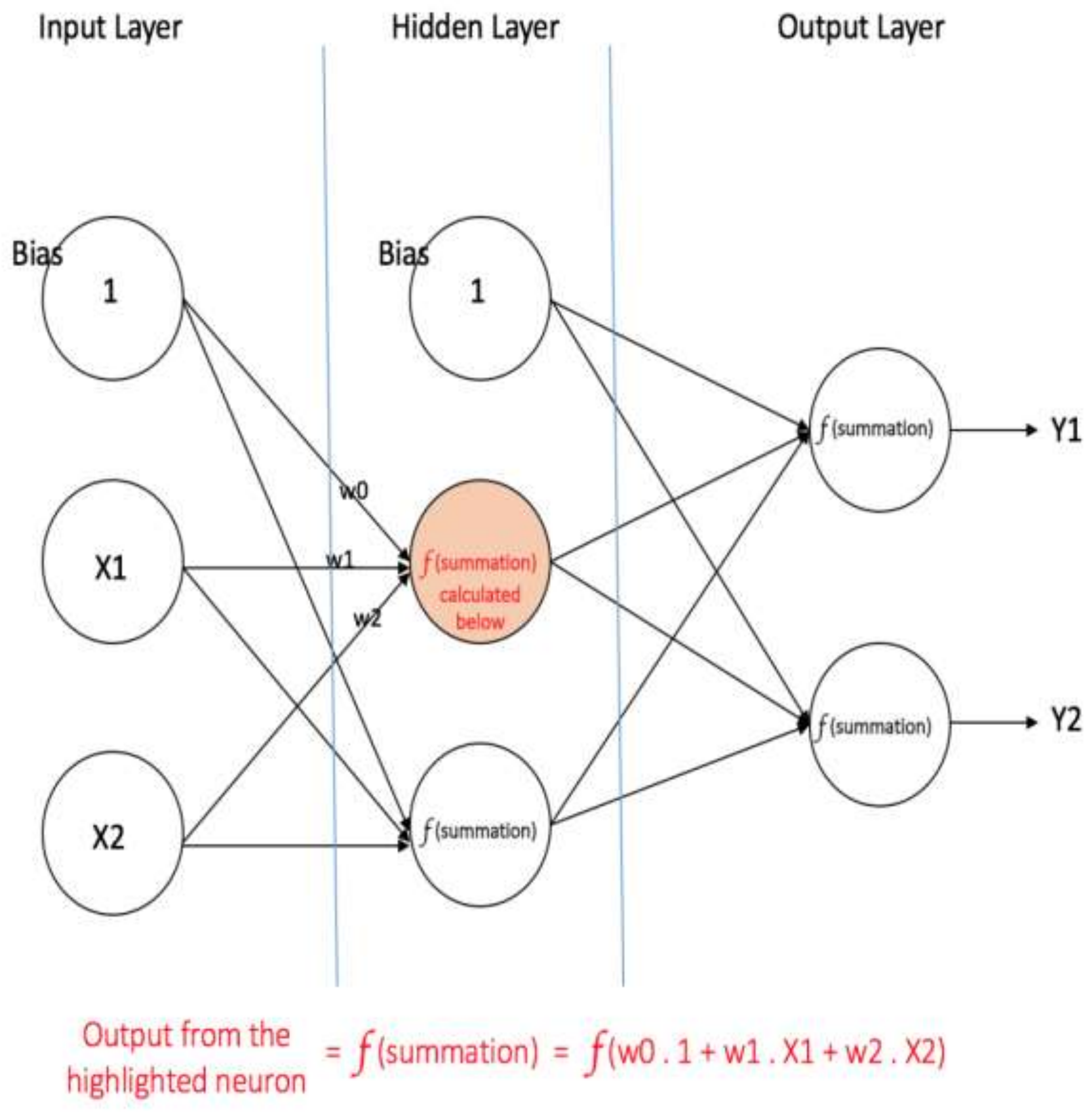

Input Layer
Hidden Layer
Output Layer
Bias
1
Bias
1
X1
X2
w0
w1
w2
f(summation) calculated below
f(summation)
f(summation)
f(summation)
Y1
Y2
Output from the highlighted neuron = f(summation) = f(w0 . 1 + w1 . X1 + w2 . X2)

Figure-20: A multi-layer perceptron having one hidden layer[16]

Output Layer: The two nodes of Output layer which take inputs from the Hidden layer and perform similar computations as shown for the highlighted hidden node. The values calculated (Y1 and Y2) as a result of these computations act as outputs of the Multi Layer Perceptron.
Given a set of features X = (x1, x2, …) and a target y, a Multi Layer Perceptron can learn the relationship between the features and the target, for either classification or regression.

4.3.10.2. Backward Propagation of Errors

This is often abbreviated as BackProp is one of several ways in which an artificial neural network (ANN) can be trained. It is under a supervised training scheme, which means that it learns from the tagged training data.
To put it in simple terms, BackProp is like "learning from mistakes". The supervisor corrects the ANN every time he makes a mistake.
An artificial neural network consists of nodes in the following layers: input layer, intermediate hidden layer, and output layer. The "weights" associated with the connected nodes of adjacent layers. The goal of learning is to assign correct weights for these edges. In supervised learning, the training set is labeled. This means that, for some given inputs, we know the desired / expected output.

BackProp Algorithm:

Initially, all edge weights are assigned randomly. For each entry in the training data set, the ANN is activated and its output is observed. This result is compared to the desired result that we already know, and the error "propagates" back to the previous layer. This error observes and the weights are "adjusted" accordingly. This process is repeated until the output error is below a predetermined threshold.
Once the previous algorithm ends, we have a "learned" ANN that, we believe, is ready to work with "new" entries. It is said that this ANN has learned from several examples (tagged data) and its errors (propagation of errors).
Now that we have an idea of how Backpropagation works, let's go back to our student brand data set shown above.
The multilayer perceptron shown in Figure-20, has two nodes in the input layer (apart from the Bias node) that take the 'Hours Studied' and 'Mid Term Marks' entries ' It also has a hidden layer with two nodes (apart from the Bias node). The output layer also has two nodes: the upper node generates the probability of 'Pass', while the lower node generates the probability of 'Fail'.
In classification tasks, we generally use a Softmax function as the activation function in the Multilayer Perceptron output layer to ensure that the outputs are probabilities and add 1. The Softmax function takes a vector of arbitrary real value scores and crushes it to a vector of values between zero and one that adds up to one. So, in this case,

Probability (Pass) + Probability (Fail) = 1

Step 1: Forward Propagation
All weights in the network are randomly assigned. Lets consider the hidden layer node marked V in Figure-20 below. Suppose that the weights of the connections of the inputs to that node are w1, w2 and w3 (as shown).
Then, the network takes the first example of training as input (we know that for entries 35 and 67, the probability of passing is 1).

- Network input = [35, 67]
- Desired network output (destination) = [1, 0]

Then, the V output of the node under consideration can be calculated in the following way (f is an activation function as a sigmoid):

V = f (1 * w1 + 35 * w2 + 67 * w3)

Similarly, the outputs of the other node in the hidden layer are also calculated. The inputs to the two nodes in the output layer come from the outputs of the two nodes in the hidden layer. This allows us to calculate the exit probabilities of the two nodes in the output layer.
Suppose, the output probabilities of the two nodes in the output layer are 0.4 and 0.6 respectively. We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively), hence the network in Figure-20 is said to have an 'Incorrect Output'.

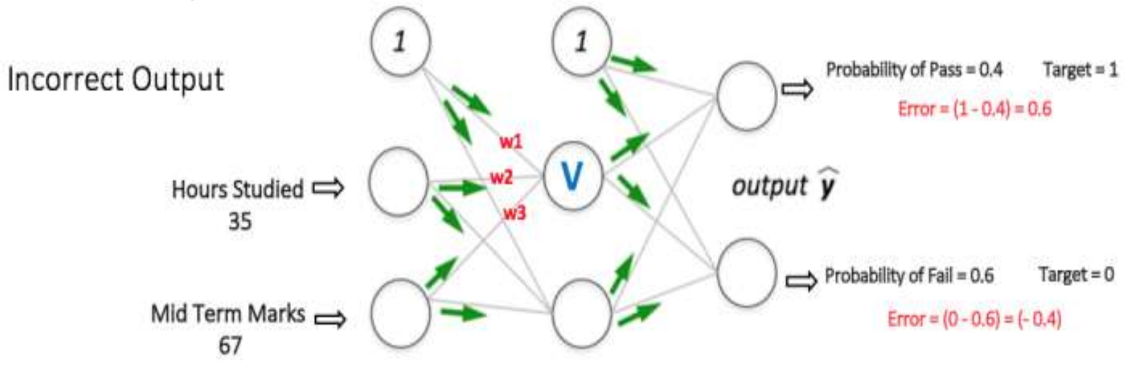
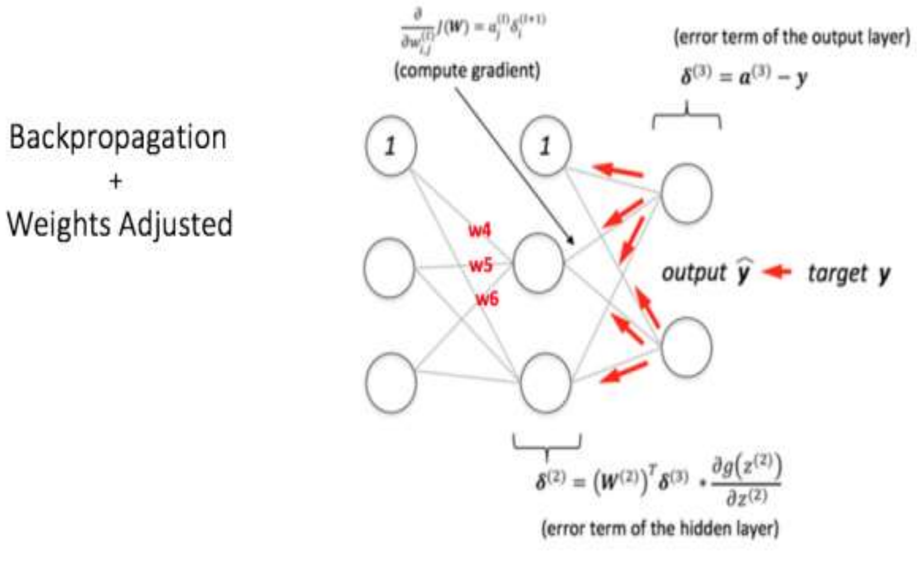


Figure-21: Forward propagation step in a multi-layer perceptron[16]

Step 2: Back Propagation and Weight Updation
We calculate the total error in the output nodes and propagate these errors through the network using Backpropagation to calculate the gradients. Then we use an optimization method like Gradient Descent to 'adjust' all the weights in the network in order to reduce the error in the output layer. This is shown in Figure 6 below (ignore the mathematical equations in the figure for now). Suppose that the new weights associated with the considered node are w4, w5 and w6 (after the inverse propagation and weight adjustment).



$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
Backpropagation + Weights Adjusted
1
1
w4
w5
w6
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} \cdot \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

Figure-22: Backward propagation and weight updation step in a multi-layer perceptron[16]

If we now enter the same example into the network again, the network should work better than before since the weights have now been adjusted to minimize the prediction error. As shown in Figure 7, errors at the output nodes are now reduced to [0.2, -0.2] compared to [0.6, -0.4] above. This means that our network has learned to correctly classify our first example of training.

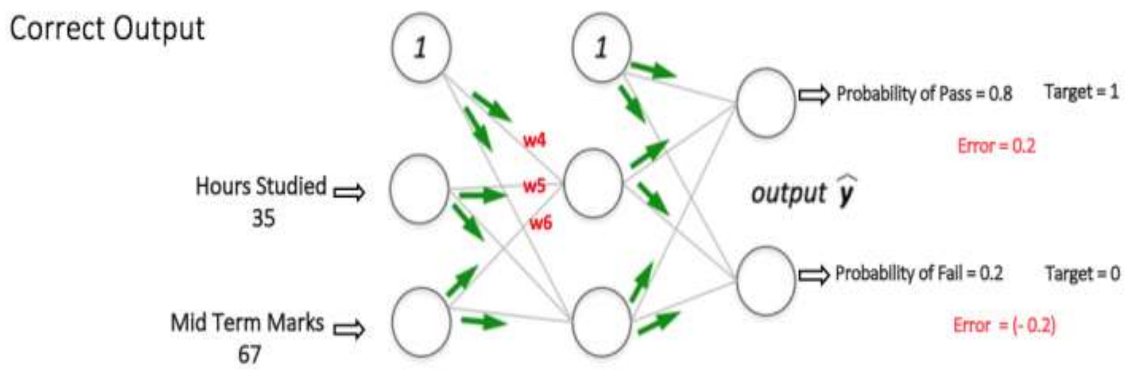


Figure-23: The MLP network now performs better on the same input[16]

We repeat this process with all the other examples of training in our data set. Then, it is said that our network learned those examples.
If we now want to predict if a student who studies 25 hours and has 70 medium-term qualifications will pass the final term, we pass through the forward propagation step and find the exit probabilities for Pass and Fail.
I have avoided the mathematical equations and the explanation of concepts like 'Gradient Descent' here and I have tried to develop an intuition for the algorithm. For a more mathematically involved discussion of the Backpropagation algorithm, see this link.

5. Different Neural Networks

5.1. Convolutional Neural Network (CNN)

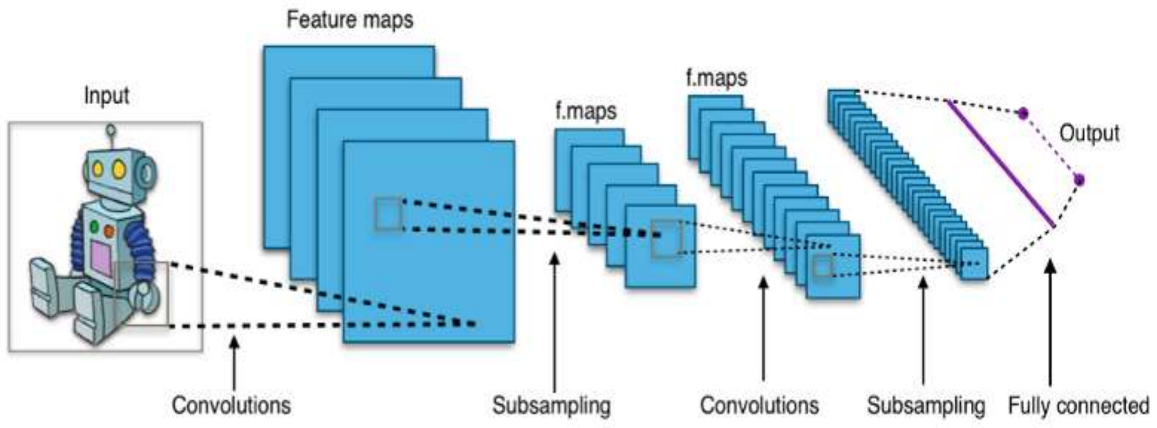


Figure-24: Typical CNN architecture[16]

A convolutional neuronal network (CNN) contains one or more convolutional layers, grouped or fully connected, and uses a variation of multilayer perceptions. The convolutional layers use a convolution operation at the input that passes the result to the next layer.
Yoon Kim in convolutional neural networks for classifying sentences describes the process and results of text classification tasks using CNN. He presents a model built on word2vec, carries out a series of experiments with it and compares it with several reference points, which shows that the model has an excellent performance.

5.2. Recursive Neural Network (RNN)

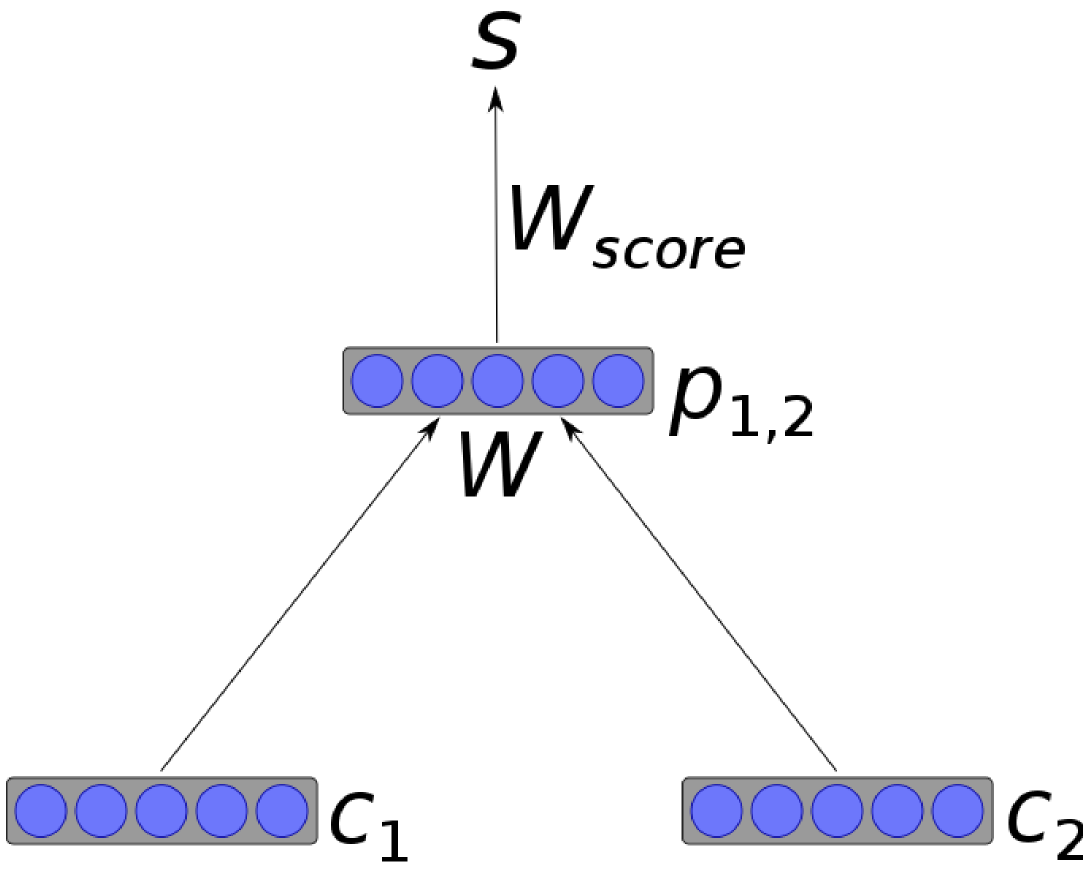


Figure-25: A simple recursive neural network architecture[16]

A recursive neural network (RNN) is a type of deep neural network formed by applying on the same original set of weights in a recursive manner on a structure to make a structured prediction on input structures of varying size, traversing a given structure in topological order. In the simplest architecture, a non-linearity such as tanh and a weighting matrix that is shared throughout the network are used to combine nodes in parents.

5.3. Recurrent Neural Network (RNN)

A recurrent neural network (RNN), unlike a feedforward neural network, is a variant of a recursive artificial neural network in which the connections between the neurons make a directed cycle. It means that the output depends not only on the current inputs but also on the neuronal state of the previous step. This memory allows users to solve NLP problems such as handwriting recognition or voice recognition. In a document, Generation of natural language, paraphrasing and summary of revisions of users with recurring neural networks, the authors demonstrate a recurrent neural network model (RNN) that can generate novel sentences and document summaries.
Siwei Lai, Liheng Xu, Kang Liu and Jun Zhao created a recurrent convolutional neural network for text classification without human-designed functions and described it in recurrent convolutional neural networks for text classification. His model was compared with existing text classification methods such as Bag of Words, Bigrams + LR, SVM, LDA, Tree Kernels, Recursive Neural Network and CNN. It was shown that their model exceeds the traditional methods for all the data sets used.

5.4. Long Short-Term Memory (LSTM)

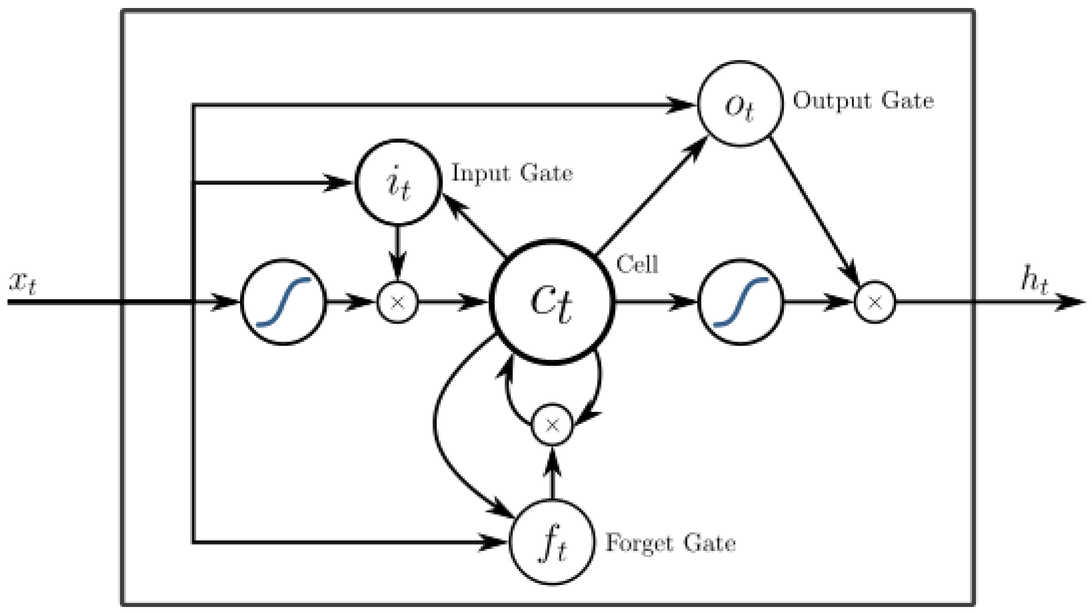


Figure-26: A peephole LSTM block with input, output, and forget gates[16]

A specific recurrent neural network (RNN) architecture is the LSTM(Long Short-Term Memory) that was designed to model a temporal sequences with their long-range dependencies what is bit more accurately than a conventional RNNs. LSTM does not use the activation function within its recurring components, the stored values are not modified and the gradient does not tend to disappear during training. In general, LSTM units are implemented in "blocks" with several units. These blocks have three or four "doors" (for example, entrance door, forgetting door, exit door) that control the flow of information that is based on the logistics function.

However, Apple, Amazon, Google, Microsoft and other companies incorporated LSTM as a fundamental element in their products.

5.5. Sequence-To-Sequence models

In general, a sequence-to-sequence model consists of two recurrent neural networks: an encoder that processes the input and a decoder that produces the output. The encoder and the decoder may use the same or different sets of parameters.

Sequence to sequence models are mainly used in answering systems for questions, chatbots and machine translation. Such multilayer cells have been used successfully in sequence-to-sequence models for translation in the Sequence to Sequence Learning with Neural Networks study.

In a Paraphrase Detection method Using Recursive Autoencoder, a well established recursive autoencoder architecture is presented. Representations are vectors in a n-dimensional semantic space where sentences with similar meanings are close to each other.

5.6. Shallow Neural Networks

Shallow models are also popular and useful tools. For example, a word2vec is a group of two-layer surface models that are used to produce word inlays. Presented in Efficient

Estimation of Representations of Words in Vector Space, word2vec takes a large corpus of text as input and produces a vector space. Each word in the corpus obtains the corresponding vector in this space. The distinctive feature is that the words from common contexts in the corpus are located close to each other in the vector space.

Why CNN is smarter way to train a data set

Not to feed the entire images in our neural network as a grid of numbers, instead of doing that we are going to do something much better way that takes advantage of the idea that an object is the same no matter where it appears in an image.

This is how it will work, step by step:

Step 1: Divide the image into overlapping tiles, over the entire original image let's move a sliding window and save each result as a small and separate image mosaik:

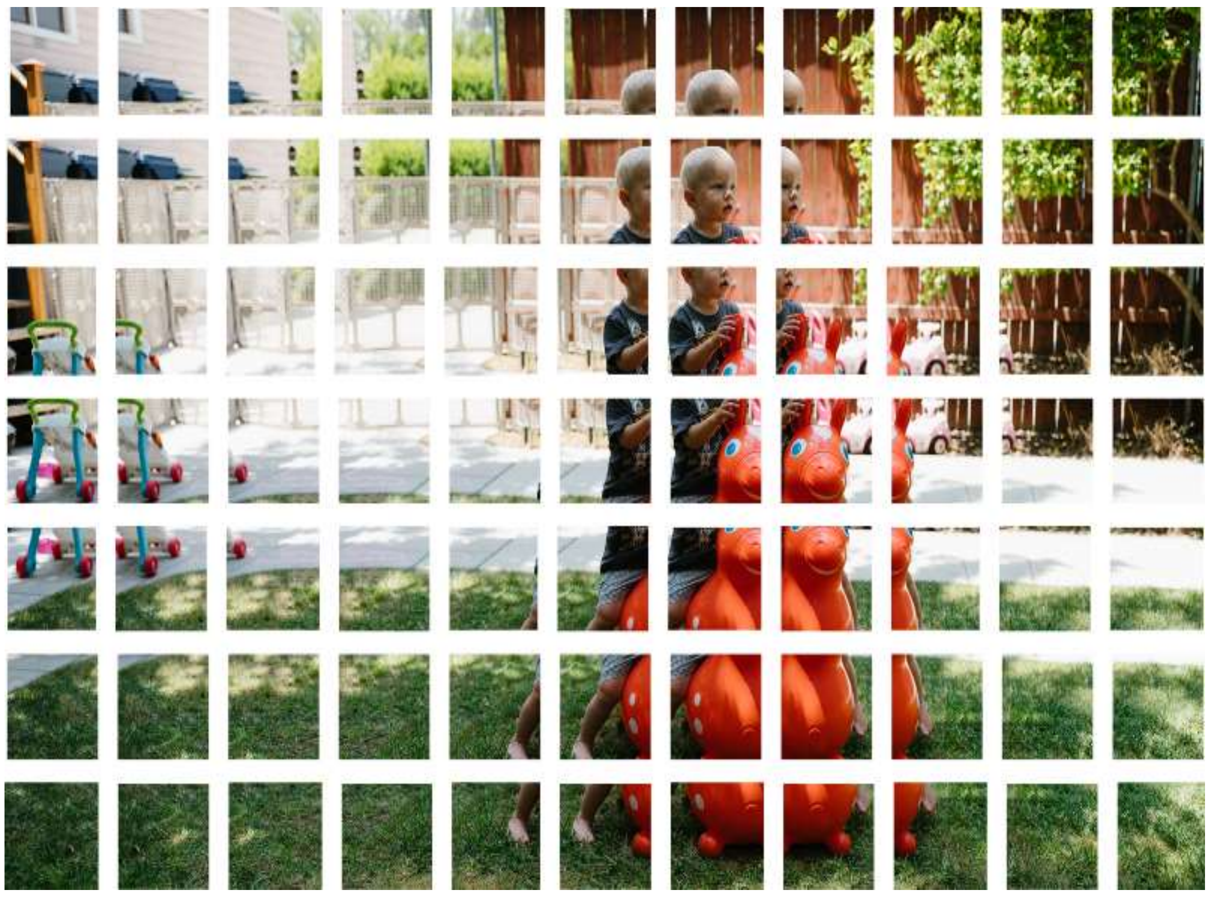


Figure-27[15]

By doing this, we converted our original image into 77 small mosaics of images of equal size.

Step 2: Feed each image mosaic into a small neural network

Previously, we introduced a single image in a neural network to see if it was an "8". We will do exactly the same here, but we will do it for each individual image mosaic:

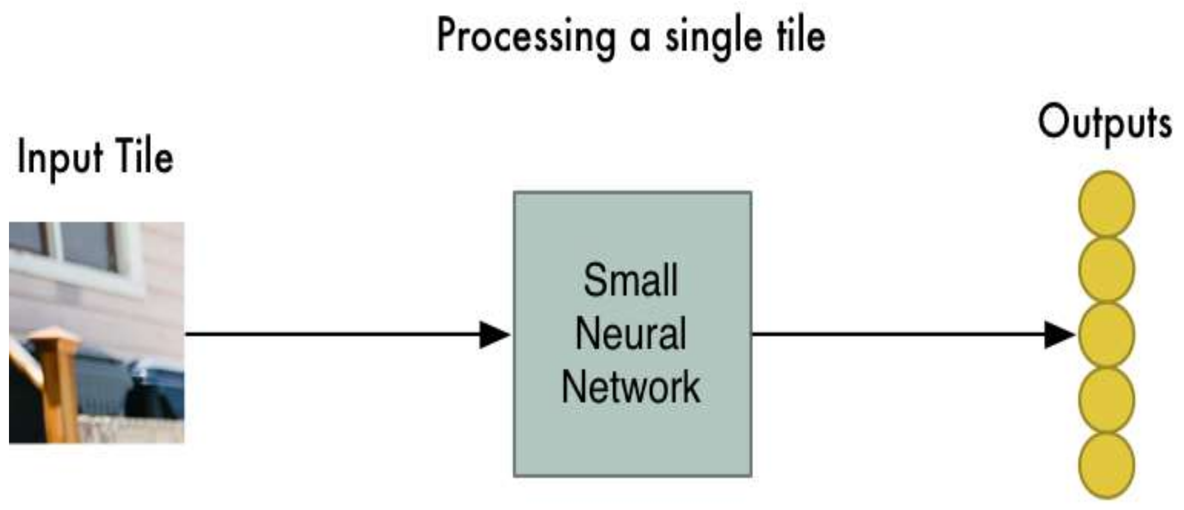


Figure-28[15]

Like once for each tile, repeat this 77 times.

However, there can be a big twist: we will keep the same neural network weights for each tile in the same original image. We are going to treat each mosaic image equally. If something interesting appears in a given mosaic, we will mark it as interesting.

Step 3: save the results of each tile in a new matrix. We do not want to lose track of the layout of the original chips. So we save the result of processing each tile in a grid in the same layout as the original image. Does it look like this:

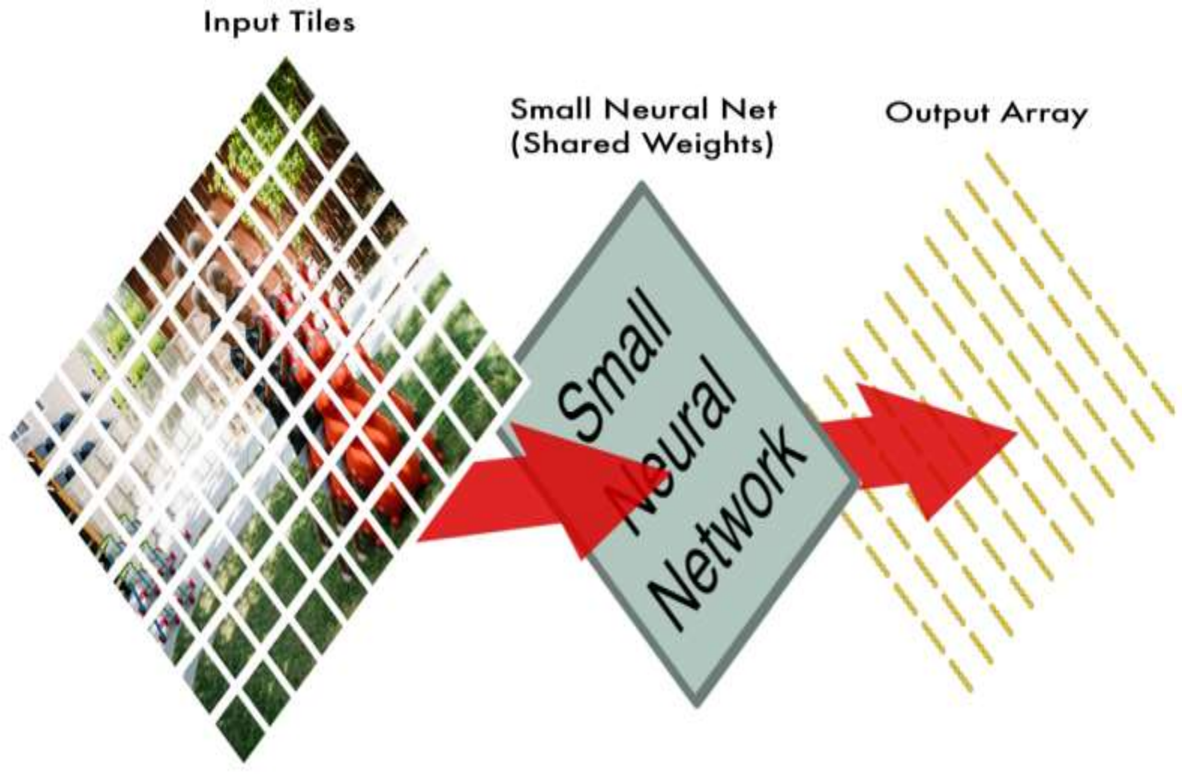


Figure-29[15]

In other words, we started with a large image and then ended up with a slightly smaller matrix that records which sections of our original image were the most interesting.

Step 4: reduction of samples. The result of Step 3 was a matrix that states which parts of the original image are the most interesting. But that set is still quite large:

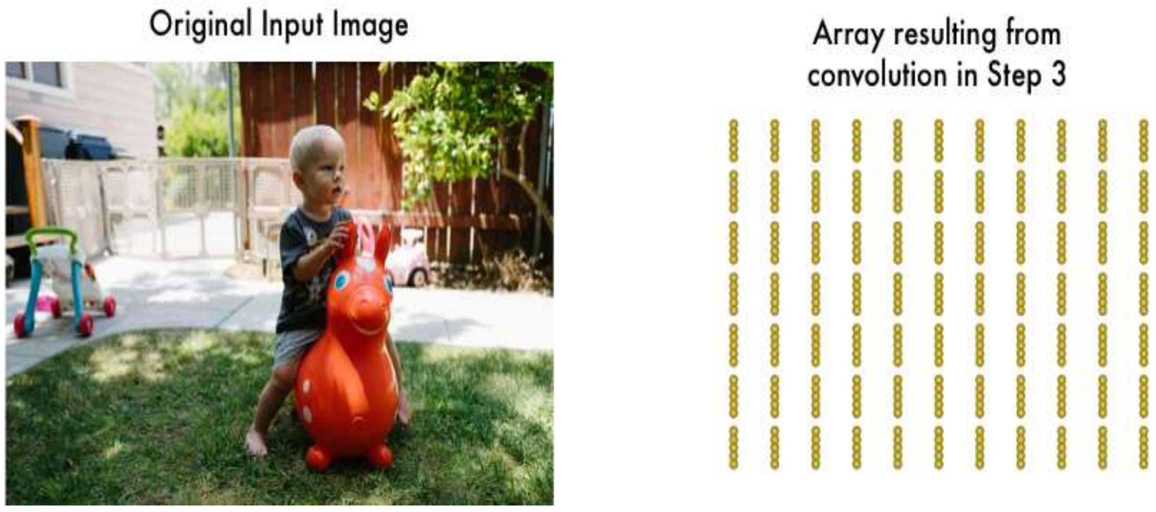


Figure-30[15]

To reduce the size of the matrix, we reduce it by means of an algorithm called maximum grouping. It sounds elegant, but it's not for nothing!

We will see each square of 2x2 of the matrix and keep the largest number:

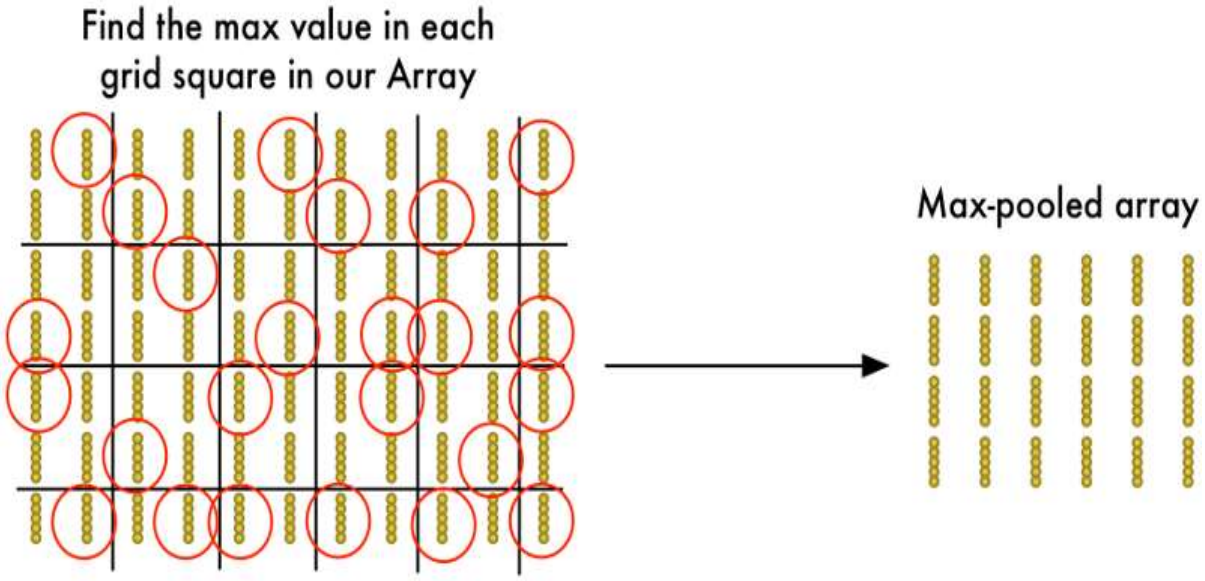


Figure-31[15]

Incase we find something interesting in any of the four input tiles that make up each 2x2 grid, we will keep the most interesting bit. This reduces the size of our matrix while keeping the most important bits.

Final step: make a prediction

So far, we have reduced a giant image to a fairly small matrix.

Guess what? That matrix is just a group of numbers, so we can use that small matrix as input into another neural network. This final neural network will decide whether the image is or does not match. To differentiate it from the convolution step, we call it a "totally connected" network.

Therefore, from start to finish, our entire five-step pipeline looks like this:

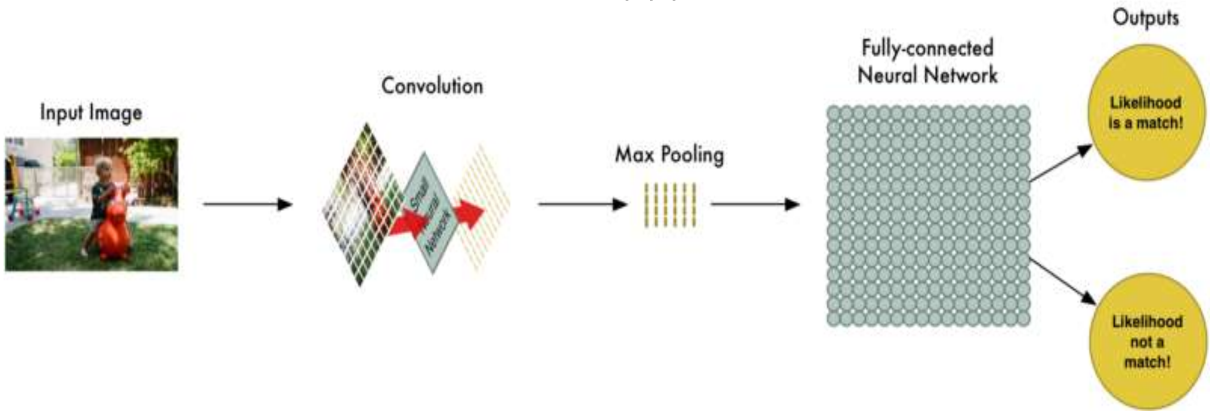


Figure-32[15]

Add even more steps

Our line of image processing is a series of steps: convolution, maximum use and finally a fully connected network.

When solving problems in the real world, these steps can be combined and stacked as many times as you want. It can have two, three or even ten layers of convolution. You can add the maximum grouping where you want to reduce the size of your data.

The basic idea is to start with a large image and gradually reduce it, step by step, until it finally has a single result. However, the more complicated features your network can recognize if more convolution steps you have.

For example, the first convolution step could learn to recognize sharp edges, the second convolution step could recognize peaks using its knowledge of sharp edges, the third step could recognize whole birds using their knowledge of peaks, etc.

This is what seems to be a more realistic deep convolutional network (like the one you would find in a research paper):

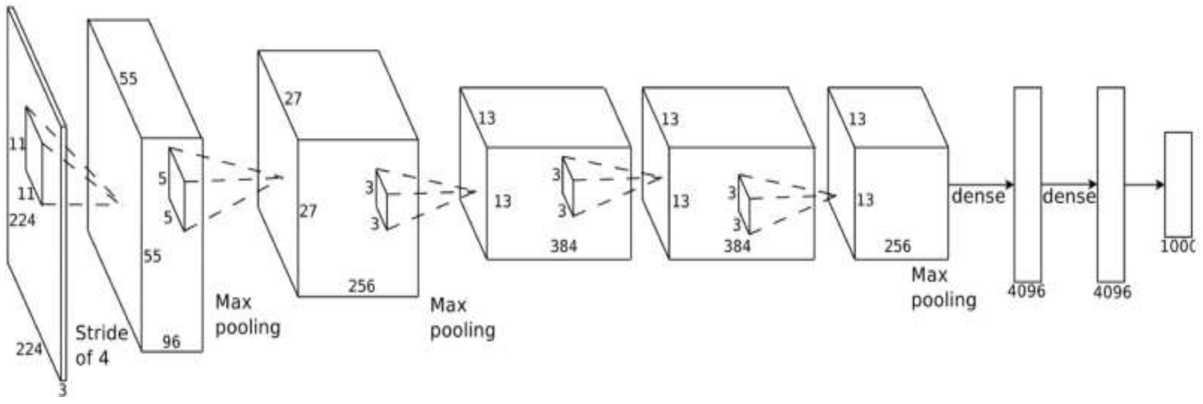


Figure-33[15]

In this case, they start an image of 224 x 224 pixels, apply the convolution and the maximum grouping twice, apply the convolution 3 more times, apply the maximum combination and then have two totally connected layers. The final result is that the image is classified in one of the 1000 categories.

Convolutional Neural Networks (CNNs / ConvNets)

The convolutional neural networks are very similar to the ordinary neural networks. They are formed by neurons and that have weights and biases that can be learned. Each neuron receives some inputs, makes a knitted product and optionally follows it with a non-linearity. The entire network still expresses a unique distinguishable scoring function: from the pixels of the unformatted image on one end to the class scores on the other. And they still have a loss function (for example, SVM / Softmax) in the last layer (fully connected) and all the tips / tricks that we developed to learn regular neural networks are still applied.
The ConvNet architectures explicitly assume that the entries are images, however, that allows to encode certain properties in the architecture. These make the forwarding function more efficient to implement and greatly reduce the number of parameters in the network.

Architecture Overview:

Recall: regular neural networks. However, each hidden layer is made of a set of neurons and each neuron is completely connected to all the neurons in the previous layer, and where the neurons in a single layer operate completely independently and do not share any connection. The last fully connected layer is called the "output layer" and in the classification configuration it represents the class scores.

Regular neural networks do not adapt well to complete images. In CIFAR-10, the images are only 32x32x3 (32 wide, 32 high, 3 color channels), so a single neuron totally connected in a hidden first layer of a normal neural network would have 32 * 32 * 3 = 3072 pesos. This amount seems manageable, but it is evident that this completely connected structure does not adapt to larger images. For example, a more respectable size image, p. 200x200x3, would lead to neurons that have 200 * 200 * 3 = 120,000 pesos. On the other hand, it is almost certain that we would like to have several of these neurons, so the parameters would add up quickly. Clearly, this complete connectivity is a waste and the large number of parameters would quickly lead to an excessive adjustment.

3D volumes of neurons. In CNN, the input consists of images and restricts the architecture in a more sensible way. In particular, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the previous layer, instead of all the neurons in a totally connected way. In addition, the final output layer for CIFAR-10 has dimensions of 1x1x10, because at the end of the ConvNet architecture we will reduce the entire image in a single vector of class scores. Here is a visualization:

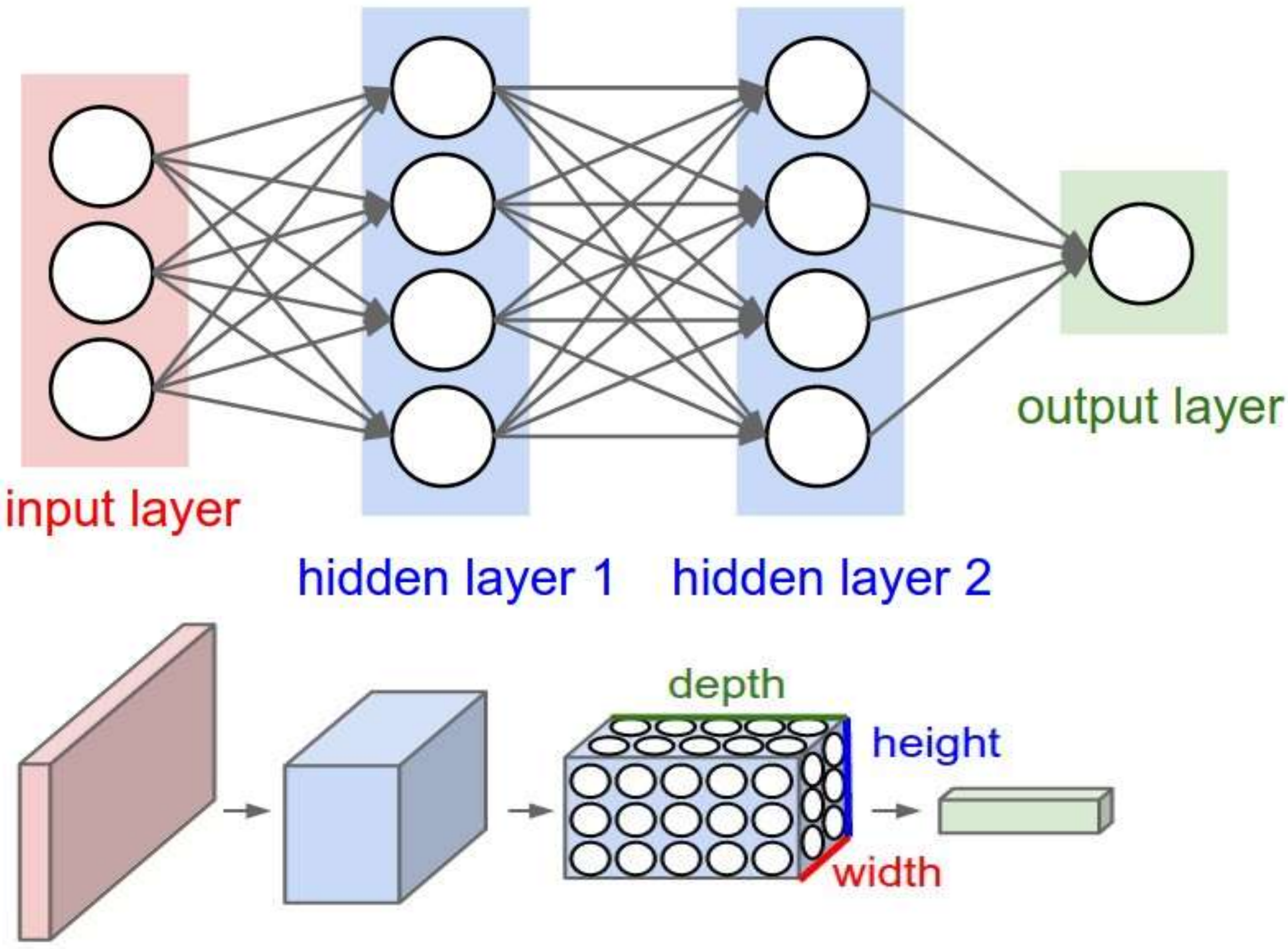


Figure-34[17]

Left: a 3-layer regular neural network. Right: A ConvNet organizes its neurons in three dimensions (width, height, depth), as shown in one of the layers. Each layer of a ConvNet transforms the 3D input volume into a 3D output volume of neuronal activations.

A ConvNet is composed of layers. Layers used to build ConvNets

A simple ConvNet is a sequence of layers, volume of activations into another through function. We use three main types of layers to build ConvNet architectures: convolutional layer, grouping layer and fully connected layer (exactly as seen in normal neural networks). We will stack these layers to form a complete ConvNet architecture.

Example architecture: general description. We will go into more details below, but a simple ConvNet for the CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. More details:

INPUT of similar [32x32x3] will support the pixel values without formatting the image, in this case an image of width 32, height 32 and with three color channels R, G, B.

The CONV layer will calculate the output of the neurons that are connected to the local regions in the input, each calculating a scalar product between its weights and a small region to which they are connected in the input volume. This can generate a volume like [32x32x12] if we decide to use 12 filters.

The RELU layer will apply an activation function for elements, such as maximum (0, x) maximum (0, x) at zero. This leaves the volume size unchanged ([32x32x12]).

The POOL layer will perform a sampling reduction operation along the spatial dimensions (width, height), resulting in a volume such as [16x16x12].

The FC layer (ie fully connected) will calculate the class scores, which will result in a volume size [1x1x10], where each of the 10 numbers corresponds to a class score. As with ordinary neural networks, and as the name implies, each neuron in this layer will connect to all the numbers in the previous volume.

With this approach, from the original pixel values to the final class scores, ConvNets transforms the original image layer by layer Note that some layers contain parameters and others do not. In particular, the CONV / FC layers perform transformations that are a function not only of the activations in the input volume, but also of the parameters (the weights and biases of the neurons). The parameters in the CONV / FC layers will be trained with gradient slope so that the class scores that ConvNet computes are consistent with the labels in the training set for each image.

In summary:
A ConvNet architecture is, in the simplest case, a list of layers that transform the volume of the image into an output volume (for example, maintaining the class scores)

There are some different types of layers (for example, CONV / FC / RELU / POOL are by far the most popular)

Each layer accepts a 3D input volume and transforms it into a 3D output volume through a differentiable function

Each layer may or may not have parameters (for example, CONV / FC do, RELU / POOL no)

Each layer may or may not have additional hyperparameters (for example, CONV / FC / POOL do, RELU does not)

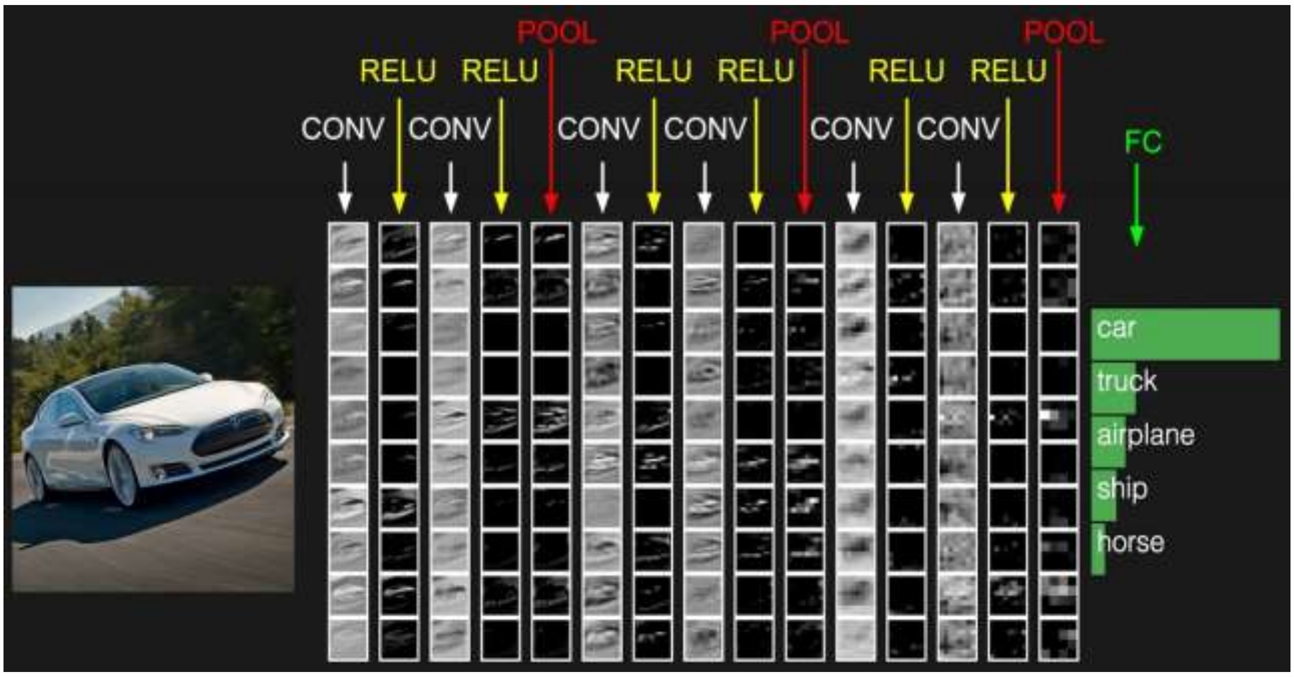


Figure-35[17]

Activations of an example ConvNet architecture. The initial volume stores the pixels of the raw image (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is displayed as a column. Since it is difficult to visualize volumes in 3D, we place the divisions of each volume in rows. The last layer volume contains the scores of each class, but here we only visualize the 5 main classified scores and print the labels of each one. The complete web-based demo is shown in the header of our website. The architecture shown here is a small VGG network, which will be discussed later.

Now we describe the individual layers and the details of their hyperparameters and their connectivities.

Convolutional Layer

The Conv layer is the basic component of a Convolutional Network that performs most of the computational heavy lifting.
Overview and intuition without brain things. First, let's analyze what the CONV layer calculates without the brain / neuron analogies. The parameters of the CONV layer consist of a set of filters that can be learned. Each filter is spatially small (wide and high), but extends through the total depth of the input volume. For example, a typical filter in a first layer of a ConvNet could have a size of 5x5x3 (that is, 5 pixels wide and high, and 3 because the images have depth 3, the color channels). During the forward pass, we slide (more precisely, we convolve) each filter across the width and height of the input volume and calculate the point products between the filter inputs and the input at any position. As we slide the filter over the width and height of the input volume we will produce a two-dimensional activation map that provides the responses of that filter in each spatial position. Intuitively, the network will learn the filters that are activated when they see some kind of visual characteristic, such as an edge of some orientation or a spot of some color in the first layer, or eventually whole patterns in the shape of a wheel or honeycomb in the upper layers of network . Now, we will have a complete set of filters in each CONV layer (for example, 12 filters), and each of them will produce a two-dimensional activation map separately. We will stack these activation maps along the depth dimension and produce the output volume.

The brain view: If you are a fan of the brain / neuron analogies, each input in the 3D output volume can also be interpreted as an output of a neuron that only looks at a small region in the input and shares parameters with all the neurons on the left and spatially right (since all these numbers result from applying the same filter). Now we will discuss the details of the neural connectivity, its disposition in space and its distribution scheme of parameters.

Local Connectivity: It is not practical to connect the neurons to all the neurons in the previous volume when it comes to high-dimensional inputs such as images. Instead, we will connect each neuron to only one local region of the input volume. The hyperparameter calls the receptive field of the neuron. The connectivity along the depth axis is always. It is always important to emphasize again on this asymmetry in the way we are going to treat the spatial dimensions and the depth dimension: the connections are local in space (width and height), but always along the entire depth of the input volume. Example 1. Suppose, for example, that the input volume has a size [32x32x3], (for example, a RGB image CIFAR-10). If the receptive field (or filter size) is 5x5, then each neuron in the convective layer will have weights in a region [5x5x3] in the input volume, for a total of 5 * 5 * 3 = 75 pesos (y + 1 bias) parameter). Note that the extension of the connectivity along the depth axis must be 3, since this is the depth of the input volume.

Example 2. Suppose that an input volume has a size [16x16x20]. Then, using an example receptive field size of 3x3, each neuron in the convective layer would now have a total of 3 * 3 * 20 = 180 connections to the input volume. Note that, once again, the connectivity is local in space (for example, 3x3), but complete along the entry depth (20).

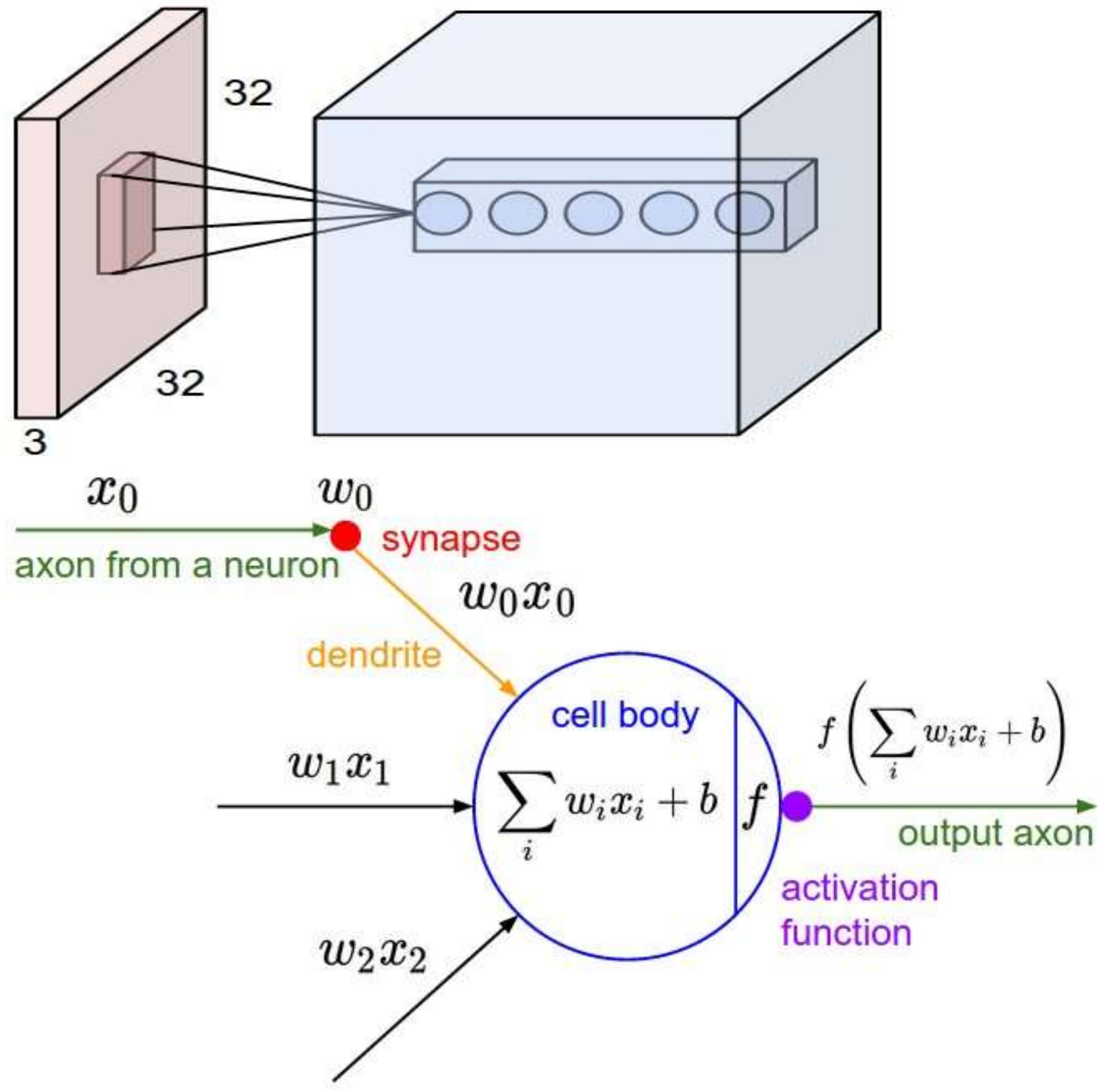


Figure-36[17]

Left: a sample input volume in red (for example, a CIFAR-10 image of 32x32x3) and an example volume of neurons in the first Convolutional layer. In a convolutional layer each neuron is connected in full depth to a local region. Note that there are multiple neurons (5 in this example) along the depth, all looking at the same region at the entrance; see the explanation of the depth columns in the following text. Right: the neurons of the neural network chapter remain unchanged: they still calculate a point product of their weights with the input followed by a non-linearity, but now their connectivity is restricted to being spatially local.

Spatial arrangement: Already explained the input volume of each neuron in a CNN with the and not yet discussed about how many neurons there are in the output. Three hyperparameters control the size of the output volume: depth, stride and zero fill. We talk about these below:

Depth: First, the depth of the output volume is a hyperparameter: it corresponds to the number of filters that we would like to use, each one learning to look for something different in the input. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may be activated in the presence of several

oriented edges. Here we refer to a set of neurons that belong at the same region of the entrance as a column of depth.
Stride: Second, we must specify the stride with which we slide the filter. When the stride is 1, we move the filters one pixel at a time. When the stride is two.
Zero-padding: Zero-padding: As we will see soon, it will sometimes be convenient to fill the input volume with zeros around the edge. The size of this zero fill is a hyperparameter. The good feature of the zero fill is that it will allow us to control the spatial size of the output volumes.
We can calculate the spatial size of the output volume as a function of the size of the input volume (WW), the size of the receptive field of the Conv Layer neurons (FF), the stride with which they are applied (SS) and the amount of zero padding used (PP) on the edge. You can convince yourself that the correct formula for calculating how many neurons "fit" is given by (WF + 2P) / S + 1 (WF + 2P) / S + 1. For example, for a 7x7 input and a 3x3 filter with Stride 1 and pad 0 would get a 5x5 output. With stride 2 we would obtain a 3x3 output. Let'salso see another graphic example:

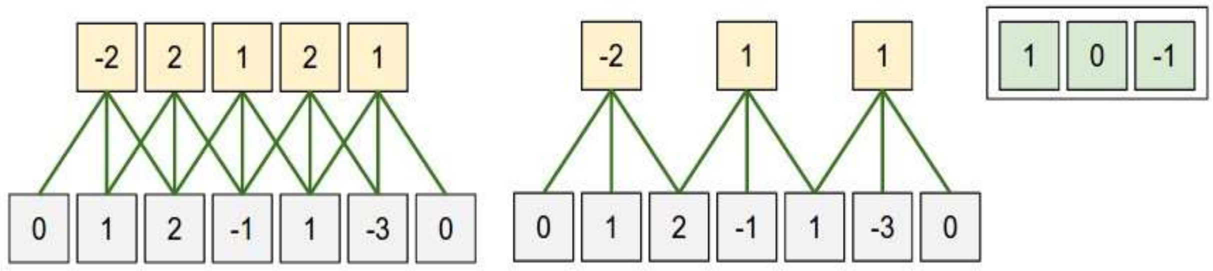


Figure-37[17]

Illustration of the spatial arrangement. In this example, there is only one spatial dimension (x axis), one neuron with a receptive field size of F = 3, the input size is W = 5, and there is no fill of P = 1. Left: the neuron with striae through the stride entrance of S = 1, giving size output (5 - 3 + 2) / 1 + 1 = 5. Right: the neuron uses stride of S = 2, giving size output (5 - 3) + 2) / 2 + 1 = 3. Note that stride S = 3 could not be used, since it would not fit perfectly in the volume. In terms of the equation, this can be determined since (5 - 3 + 2) = 4 is not divisible by 3.

The weights of the neurons are in this example [1,0, -1] (shown on the right), and their bias is zero. These weights are shared in all yellow neurons (see the shared use of parameters below).

Use of zero fill. In the previous example on the left, note that the input dimension was 5 and the output dimension was the same: also 5. This worked because our receptive fields were 3 and we used zero fill of 1. If no fill was used zero, then the output volume would have had a spatial dimension of only 3, because that's the number of neurons that would "fit" into the original input. In general, setting the zero fill as P = (F-1) / 2P = (F-1) / 2 when the stride is S = 1S = 1 ensures that the input volume and output volume will be spatially the same size . It is very common to use zero padding in this way and we will discuss the full reasons when we talk more about ConvNet architectures.

Restrictions on progress. Notice again that the hyperparameters of spatial arrangement have mutual restrictions. For example, when the input has a size W = 10W = 10, zero padding is not used P = 0P = 0, and the filter size is F = 3F = 3, then it would be impossible to use stride S = 2S = 2, since (WF + 2P) / S + 1 = (10-3 + 0) /2+1=4.5 (WF + 2P) / S + 1 = (10-3 + 0) / 2 + 1 = 4.5, that is, it is not an integer, which indicates that the neurons do not "fit" neatly and symmetrically through the input. Therefore, this configuration of the hyperparameters is considered invalid, and a ConvNet library could throw an exception or zero pad the rest so that it fits, or cut the entry to fit, or something. As we will see in the ConvNet architectures section, sizing the ConvNets properly so that all the dimensions "work" can be a real headache, that the use of zero fill and some design guidelines will significantly alleviate.

Example of the real world. The architecture that won the ImageNet challenge in 2012 accepted images of size [227x227x3]. In the first convolutional layer, he used neurons with receptive field size F = 11F = 11, stride S = 4S = 4 and no zero fill P = 0P = 0. As (227 - 11) / 4 + 1 = 55, and as the Conv layer had a depth of K = 96K = 96, the output volume of the Conv layer had a size [55x55x96]. Each of the 55 * 55 * 96 neurons in this volume was connected to a region of size [11x11x3] in the input volume. In addition, the 96 neurons in each depth column are connected to the same region [11x11x3] of the input, but of course with different weights. As a diversion aside, if you read the current document, it states that the input images were 224x224, which is undoubtedly incorrect because (224 - 11) / 4 + 1 is quite clear that it is not a whole number. This has confused many people in the story of ConvNets and little is known about what happened.

Parameter Sharing

The parameter sharing scheme is used in Convolutional layers to control the number of parameters. Using the previous real-world example, we see that there are 55 * 55 * 96 = 290,400 neurons in the first Conv Layer, and each has 11 * 11 * 3 = 363 weights and 1 bias. Now, this adds 290400 * 364 = 105,705,600 parameters. Truely this number is very high.

It turns out that we can drastically reduce the number of parameters by making a reasonable assumption: that if a characteristic is useful for calculating at some spatial position (x, y), then it should also be useful to calculate at a different position (x2), y2). In other words, by denoting a single two-dimensional portion of depth as a depth cut (for example, a volume of size [55x55x96] has 96 depth cuts, each of size [55x55]), we will restrict the neurons in each depth segment to use the same weights and biases. With this parameter distribution scheme, the first convection layer in our example would now have only 96 unique sets of weights (one for each depth segment), for a total of 95 * 11 * 11 * 3 = 34,848 unique weights or 34,944 parameters (+96 biases). Alternatively, all 55 * 55 neurons in each depth sector will now use the same parameters. During backpropagation, each neuron in the volume will calculate the gradient for its weights, but these gradients will be added in each depth sector and will only be updated a single set of weights/segment.

Note that if all neurons in a single depth segment are using the same weight vector, then the forward pass of the CONV layer can be calculated in each depth segment as a convolution of the neuron weights with the volume input (hence the name: Convolutional Layer) That is why it is common to refer to sets of weights as a filter (or a core), which is convoluted with the input.

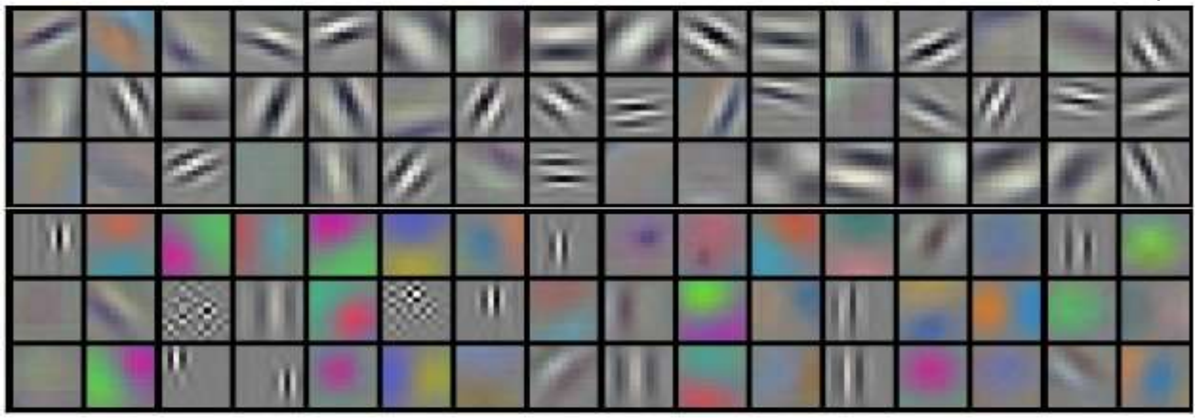


Figure-38[17]

Sample filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each is shared by the 55 * 55 neurons in a depth segment. Note that the assumption of sharing parameters is relatively reasonable: if the detection of a horizontal edge is important at some location in the image, it should also be intuitively useful at another location due to the invariant structure of translation of the images. Therefore, it is not necessary to re-learn to detect a horizontal edge in each of the 55 * 55 different locations in the output volume of the Conv layer.

Keep in mind that sometimes the assumption of sharing parameters may not make sense. This is especially the case when the input images to a ConvNet have a specific centered structure, where we should expect, for example, that completely different characteristics would be learned on one side of the image than on the other. A practical example is when the input are faces that have been centered on the image. It is expected that different specific characteristics of the eye or hair can be learned (and should be) in different spatial locations. In that scenario, it is a common way to relax the parameters exchange scheme and, instead, simply call the layer Layer locally connected.

Numpy examples: For the above discussion to be more concrete, let us express the same ideas, but in the code and with a specific example. Suppose the input volume is a numpy matrix X. Then:

A column of depth (or a fiber) in the position (x, y) would be the activations X [x, y,:].

A depth cut, or equivalent to an activation map in depth d, would be the activations X [:,:, d].

Conv Layer Example. Suppose that the input volume X has the form X.shape: (11,11,4). Suppose further that we do not use zero fill (P = 0P = 0), that the size of the filter is F = 5F = 5, and that the stride is S = 2S = 2. The output volume, therefore, would have a size spatial (11-5) / 2 + 1 = 4, giving a volume with width and height of 4. The activation map in the output volume (call it V), it would look like this (only some of the elements are computed in this example):

V [0,0,0] = np.sum (X [: 5 ,: 5,:] * W0) + b0

V [1,0,0] = np.sum (X [2: 7 ,: 5,:] * W0) + b0

V [2,0,0] = np.sum (X [4: 9 ,: 5,:] * W0) + b0

V [3,0,0] = np.sum (X [6: 11 ,: 5,:] * W0) + b0

Remember that in numpy, the operation * above denotes the multiplication by elements between the matrices. Note also that the weight vector W0 is the weight vector of that neuron and b0 is the bias. Here, we assume that W0 has the form W0.shape: (5,5,4), since the size of the filter is 5 and the depth of the input volume is 4. Note that at each point, we are calculating the scalar product as seen before in ordinary neural networks. In addition, we see that we are using the same weight and bias (due to the sharing of parameters), and where the dimensions along the width are increasing in steps of 2 (ie, the stride). To build a second activation map on the output volume, we would have:

V [0,0,1] = np.sum (X [: 5 ,: 5,:] * W1) + b1

V [1,0,1] = np.sum (X [2: 7 ,: 5,:] * W1) + b1

V [2,0,1] = np.sum (X [4: 9 ,: 5,:] * W1) + b1

V [3,0,1] = np.sum (X [6: 11 ,: 5,:] * W1) + b1

V [0,1,1] = np.sum (X [: 5,2: 7,:] * W1) + b1 (example of going y)

V [2,3,1] = np.sum (X [4: 9,6: 11,:] * W1) + b1 (or both)

we are indexing in the second depth dimension in V because we are computing the second activation map readily, and that we now use another different set of parameters (W11). In the previous example, we are for brevity omitting some of the other operations that Conv Layer would perform to fill in the other parts of the output matrix V. Also, remember that these activation maps are often followed element through an activation function as ReLU, but this is not shown here.

Summary. To summarize, the coexistence layer:

Accepts a volume size W1 × H1 × D1W1 × H1 × D1

Requires four hyperparameters:

o Number of KK filters,

or its spatial extension FF,

or the SS stride,

or the amount of zero padding PP.

Produces a volume size W2 × H2 × D2W2 × H2 × D2 where:

or W2 = (W1-F + 2P) / S + 1W2 = (W1-F + 2P) / S + 1

or H2 = (H1-F + 2P) / S + 1H2 = (H1-F + 2P) / S + 1 (ie the width and height are calculated equally by symmetry) or D2 = KD2 = K

Now, enter weights E·F·D1F·E·D1 per filter, for a total of (E·F·D1) ·K (E·F·D1) ·K1 weights and KK1.

In the output volume, the depth division dd-th (of size W2 × H2W2 × H2) is the result of performing a valid convolution of the dd-th filter on the input volume with a stride of SS, and then compensates with dd bias

A common configuration of hyperparameters is F = 3, S = 1, P = 1F = 3, S = 1, P = 1. There are common way for conventions and rules that motivate these hyperparameters. See the ConvNet architectures section below.

Convolution Demo: Below is a demonstration in execution of a CONV layer. Because 3D volumes are difficult to visualize, all volumes (input volume (in blue), weight volumes (in red), output volume (in green) are displayed with each depth sector stacked in rows. The input volume is of size W1 = 5, H1 = 5, D1 = 3W1 = 5, H1 = 5, D1 = 3, and the parameters of the CONV layer are K = 2, F = 3, S = 2, P = 1K = 2, F = 3, S = 2, P = 1. That is, we have two filters of size 3 × 33 × 3 and apply with a stride of 2. Therefore, the size of the output volume has a spatial size (5 - 3 + 2) / 2 + 1 = 3. Also, note that a fill of P = 1P = 1 is applied to the input volume, making the outer edge of the input volume zero. display iterates over the output activations (green) and shows that each element is calculated by elements multiplying the highlighted input (blue) with the filter (red), summarizing it and then compensating the result for the bias.

Implementation as Matrix Multiplication: Note that the convolution operation essentially performs point products between the filters and the local regions of the input. A common implementation pattern of the CONV layer is to take advantage of this fact and formulate the forward step of a convolutional layer as a large matrix multiplied as follows:

The regions of a input image are stretched in columns by an operation commonly called im2col. For example, if the entry is [227x227x3] and it will be convoluted with 11x11x3 filters in step 4, we would take [11x11x3] blocks of pixels in the input and stretch each block in a column vector of size 11 * 11 * 3 = 363 When iterating this process in the entry in stride of 4 we obtain (227-11) / 4 + 1 = 55 locations along width and height, which leads to an output matrix X_col of im2col of size [363 x 3025], where each column is a stretched receptive field and there are 55 * 55 = 3025 of them in total. Note that since the receptive fields overlap, each number in the input volume can be duplicated in multiple different columns.

The weights of the CONV layer extend similarly in rows. For example, if there are 96 filters of size [11x11x3] this would give a W_row array of size [96 x 363].

The result of a convolution is now equivalent to making a large matrix multiply np.dot (W_row, X_col), which evaluates the product of points between each filter and each receptive field location. In our example, the output of this operation would be [96 x 3025], giving the output of the scalar product of each filter in each location.

The result must finally be reconfigured in its proper output dimension [55x55x96].
This approach has the disadvantage that it can use a lot of memory, since some values in the input volume are replicated several times in X_col. However, the benefit is that there are many very efficient implementations of the Matrix Multiplication that we can take advantage of (for example, in the commonly used BLAS API). In addition, the same idea of im2col can be reused to perform the grouping operation, which we will discuss below.

Backpropagation: The backward pass for a convolution operation (for data and weights) is also a convolution (but with spatially inverted filters). This is easy to obtain in the one-dimensional case with a toy example (not expanded at the moment).

Convolution 1x1. On the other hand, several documents use 1x1 convolutions, as first investigated by the network in the network. Some people get confused at the beginning when they see the 1x1 convolutions, especially when they come from the processing of background signals. Normally, the signals are two-dimensional, so the 1x1 convolutions do not make sense. In a ConvNets this is not the scenario as it operates in three-dimensional volumes where as that the filters always extend through the total depth of the input volume. For example, if the input is [32x32x3], then making 1x1 convolutions would be to make three-dimensional products (since the input depth is 3 channels).

Extended Circumbances: A recent development introduces a hyperparameter to the CONV layer called dilation. So far we have only discussed the CONV filters. However, it is possible to have filters that have spaces between each cell, called dilation. As an example, in a dimension, a filter w of size 3 would calculate in the input x the following: w [0] * x [0] + w [1] * x [1] + w [2] * x [2] . This is dilation of 0. For dilation 1, the filter would calculate w [0] * x [0] + w [1] * x [2] + w [2] * x [4]; In other words, there is a space of 1 between the applications. This can be very useful in some environments to use together with filters with expansion 0 because it allows you to combine the spatial information between the inputs much more aggressively with fewer layers. For example, if you stack two CONV 3x3 layers one on top of the other, you can convince yourself that the neurons in the 2nd layer are a function of a 5x5 patch of the input (we could say that the effective receptive field of these neurons is 5x5) . If we use dilated convolutions, this effective receptive field will grow much faster.

Pooling Layer

It is common to periodically insert a Pooling layer between successive layers of Conv in a ConvNet architecture. Pooling Layer works as independently in each depth sector of the input and resizes it spatially manner by using the MAX operation there. The most common form is a grouping layer with filters of size 2x2 applied with a stride of 2 descending samples each cut of depth in the entrance by 2 along width and height, discarding 75% of the activations. Each MAX operation in this case would take a maximum of 4 numbers (small 2x2 region in some depth segment). The depth dimension remains unchanged. More generally, the grouping layer:
Accepts a volume size W1 × H1 × D1W1 × H1 × D1

Requires two hyperparameters:

or its spatial extension FF,

or the SS stride,

Produces a volume size W2 × H2 × D2W2 × H2 × D2 where:

or W2 = (W1-F) / S + 1W2 = (W1-F) / S + 1

or H2 = (H1-F) / S + 1H2 = (H1-F) / S + 1

or D2 = D1D2 = D1

Enter zero parameters since it calculates a fixed function of the input

Note that it is not common to use zero fill to group layers

It is worth noting that there are only two commonly observed variations of the maximum grouping layer encountered in practice: A grouping layer with F = 3, S = 2F = 3, S = 2 (also called overlap grouping), and most commonly F = 2, S = 2F = 2, S = 2. Gathering sizes with larger receptive fields is too destructive.

General grouping. In addition to the maximum grouping, the grouping units can also perform other functions, such as the average grouping or even the standard grouping L2. The average grouping is often used historically, but has recently become out of date compared to the maximum grouping operation, which has been shown to work best in practice.

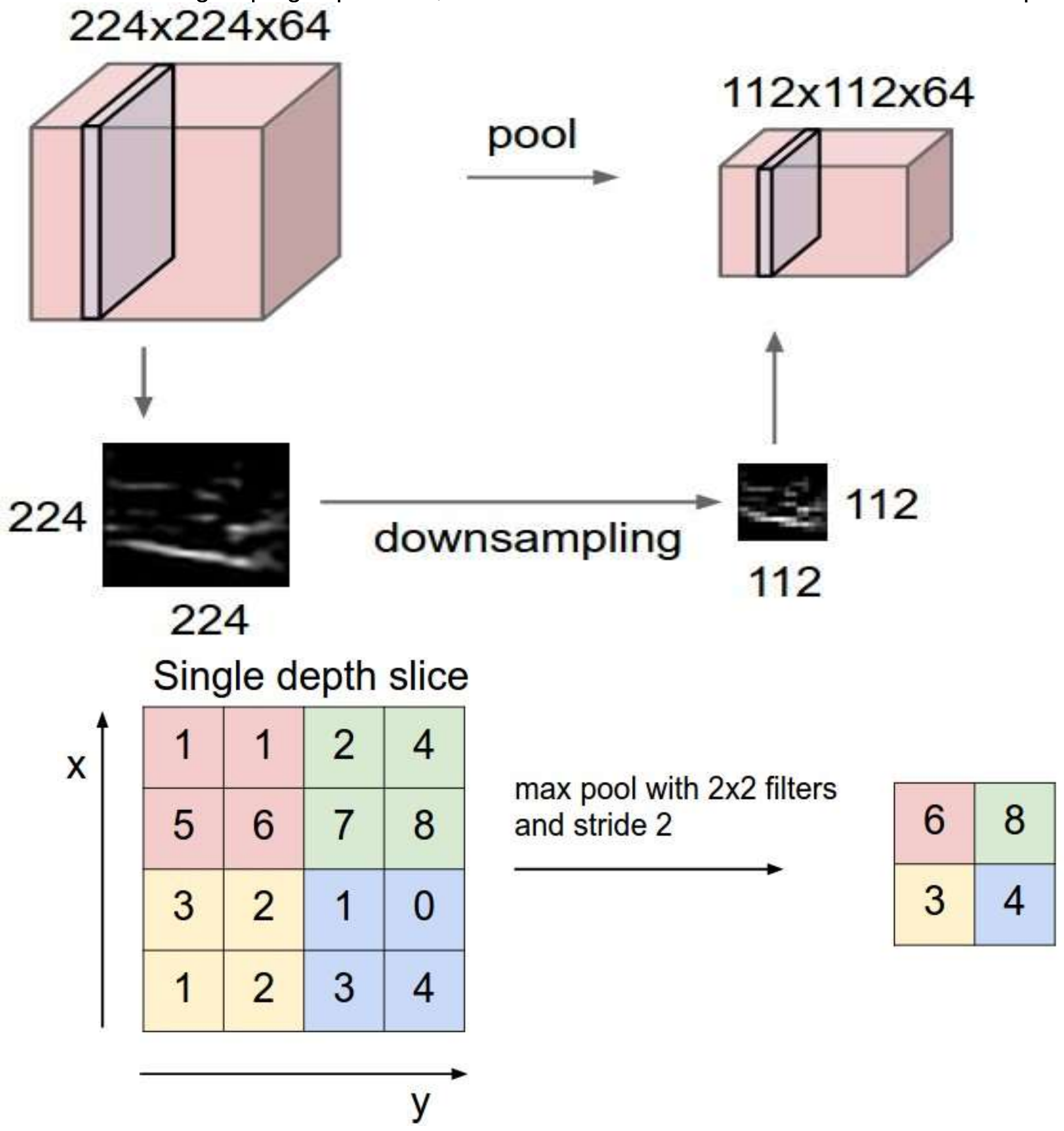


Figure-39[17]

Pooling downsamples reduces the volume spatially, independently in each depth sector of the input volume. Left: In this example, the size input volume [224x224x64] is combined with the size of filter 2, step 2 on the size output volume [112x112x64]. Note that the depth of the volume is preserved. Right: the most common sampling operation is maximum, which results in the maximum accumulation, which is shown here with a step of 2. That is, each maximum is taken in 4 numbers (small square of 2x2).

Backpropagation: Remember, from the backpropagation section, that the backward pass for a max (x, y) operation has a simple interpretation, since it only routes the gradient to the entry that had the highest value in the direct pass. Therefore, during the forward step of a clustering layer it is common to track the maximum activation rate (sometimes also called the switches) so that the gradient routing is efficient during inverse propagation.

Get rid of the commonwealth. Many people do not like the grouping operation and think we can escape without it. In case of striving for Simplicity, all Convolutional Net proposes to discard the grouping layer in favor of the architecture that only consists of repeated CONV layers. To reduce the size of the representation, they suggest using a larger stride in the CONV layer from time to time. It has also been found that discarding layers of pooling is important for the training of good generative models, such as variational autoencoders (VAE) or generative adversarial networks (GAN). It seems likely that future architectures present very few or no grouping layers.

Normalization Layer

To use in ConvNet architecture, many types of normalization layers have been proposed already. Sometimes with the intention of implementing inhibition schemes observed in the biological brain. However, these layers have fallen out of favor since then because in practice it has been shown that their contribution is minimal, if any. For several types of normalizations, see the discussion in the cuda-convnet library API by Alex Krizhevsky.

Fully-connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in normal neural networks. Therefore, their activations can be calculated with a multiplication of matrices followed by a displacement of bias. See the Neural Network section of the notes for more information.

Converting FC layers to CONV layers

There is only difference between the FC and CONV layers is that the neurons in the CONV layer are connected only to a local region at the input. However, the neurons in both layers still compute dot products, so their functional form is identical. It is possible to convert between FC and CONV layers:

For any CONV layer there is an FC layer that implements the same forwarding function. The weighting matrix would be a large matrix that is mostly zero, except in certain blocks (due to local connectivity) where the weights in many of the blocks are equal (due to the sharing of parameters).

On the contrary, any FC layer can be converted to a CONV layer. For example, an FC layer with K = 4096K = 4096 that is looking at an input volume of size 7 × 7 × 5127 × 7 × 512 can be expressed equivalently as a CONV layer with F = 7, P = 0, S = 1, K = 4096F = 7, P = 0, S = 1, K = 4096. In other words, we are setting the filter size to be exactly the size of the input volume and, therefore, the output will be simply 1 × 1 × 40961 × 1 × 4096 since only one depth column "fits" in the input volume. giving the same result as the initial FC layer.

FC-> CONV conversion. From these two conversions, the ability to convert an FC layer to a CONV layer is particularly useful in practice. Let take an image of 224x224x3 of ConvNet architecture, and after that uses a series of CONV layers and POOL layers to reduce the image to a volume of activations of size 7x7x512. From there, an AlexNet uses two FC layers of size 4096 and finally the last FC layers with 1000 neurons that calculate the class scores. We can convert each of these three FC layers into CONV layers as described above:

Replace the first FC layer that looks at the volume [7x7x512] with a CONV layer that uses a filter size F = 7F = 7, giving an output volume [1x1x4096].

Replace the second FC layer with a CONV layer that uses a filter size F = 1F = 1, giving an output volume [1x1x4096]

Replace the last FC layer in a similar way, with F = 1F = 1, giving final output [1x1x1000]

Each of these conversions could in practice involve the manipulation (e.g., remodeling) of the WW weight matrix in each FC layer in CONV layer filters. With this conversion helps us to slide the real ConvNet very appropriately in many spatial positions in a larger image, in a single forward pass.

For example, if the image 224x224 gives a volume size [7x7x512], that is, a reduction of 32, forwarding an image of size 384x384 through the converted architecture would give the volume equivalent in size [12x12x512], since 384 / 32 = 12. To continue with the next 3 CONV layers that we just converted from the FC layers would now give the final size volume [6x6x1000], since (12 - 7) / 1 + 1 = 6. Note that instead From a single vector of class size scores [1x1x1000], we now get a full set of 6x6 class scores in the 384x384 image.

The evaluation of the original ConvNet (with FC layers) independently in 224x224 cultures of the 384x384 image in 32-pixel strides gives an identical result to the forwarding of the converted ConvNet once.

Of course, forwarding the converted ConvNet only one time is much more efficient than iterating the original ConvNet in all those 39 locations, since the 39 evaluations share the calculation. This trick is often used in practice to get better performance, where, for example, it is common to change the size of an image to enlarge it, use a converted ConvNet to evaluate class scores in many spatial positions and then average the scores of class.

Finally, what would happen if we wanted to efficiently apply the original ConvNet on the image, but at a step smaller than 32 pixels? We could achieve this with multiple passes forward. For example, note that if we wanted to use a 16-pixel stride we could do it by combining the volumes received when converting ConvNet converted twice: first on the original image and second on the image but with the image spatially changed by 16 pixels along of width and height.

ConvNet Architectures

We have seen that Convolutional Networks are commonly composed of only three types of layers: CONV, POOL (we assume Max pool unless otherwise indicated) and FC (abbreviation of fully connected). We will also explicitly write the activation function RELU as a layer, which applies non-linearity by elements. In this section we will discuss how these are commonly stacked to form complete ConvNets.

Layer Patterns

The most common form of a ConvNet architecture is CONV-RELU layers, followed by with POOL layers and repeats this pattern until the image has spatially merged to a small size. At some point, it is common to make the transition to completely connected layers. The last fully connected layer contains the output, like the class scores. In other words, the most common ConvNet architecture follows the pattern:

INPUT -> [[CONV -> RELU] * N -> POOL?] * M -> [FC -> RELU] * G -> FC

where * indicates repetition and the POOL? indicates an optional grouping layer. In addition, N> = 0 (and usually N <= 3), M> = 0, G> = 0 (and usually G <3). For example, here are some common ConvNet architectures that you can see that follow this pattern:

INPUT -> FC, implements a linear classifier. Here N = M = K = 0.

INPUT -> CONV -> RELU -> FC

INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC. Here we see that there is a single CONV layer between every POOL layer.

INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*3 -> [FC -> RELU]*2 -> FC, here we can see two CONV layers stacked before every POOL layer. This is generally a good idea for larger and deeper networks, because multiple stacked CONV layers can develop more complex features of the input volume before the destructive pooling operation.

It prefers a small CONV filter stack to a large CONV layer of the receptive field. Suppose you stack three CONV 3x3 layers one on top of the other (with nonlinearities between them, of course). Each neuron in the first CONV layer has a 3x3 pattern of the input volume, in this arrangement. A neuron in the second layer CONV has a 3x3 view of the first CONV layer, and therefore, by extension, a 5x5 view of the input volume. Similarly, a neuron in the third layer CONV has a 3x3 view of the second CONV layer, and therefore a 7x7 view of the input volume. Suppose that instead of these three layers of 3x3 CONV, we would only want to use a single CONV layer with 7x7 receptive fields. These neurons would have a receptive field size of input volume that is identical in spatial extent (7x7), but with several disadvantages. First, the neurons would be computing a linear function on the input, while the three CONV layer stacks contain nonlinearities that make their features more expressive. Second, if we assume that all volumes have CC channels, then it can be seen that the single CONV 7x7 layer would contain C × (7 × 7 × C) = 49C2C × (7 × 7 × C) = 49C2 parameters, while the three CONV 3x3 layers would only contain 3 × (C × (3 × 3 × C)) = 27C23 × (C × (3 × 3 × C)) = 27C2 parameters. Intuitively, stacking CONV layers with small filters instead of having a CONV layer with large filters allows us to express more powerful characteristics of the input and with fewer parameters. As a practical disadvantage, we may need more memory to keep all the intermediate results of the CONV layer if we plan to do a new propagation.

Recent departures It should be noted that the conventional paradigm of a linear list of layers has recently been questioned, in the Google Inception architectures and also in the current residual networks of Microsoft Research Asia. Both (see the details below in the case studies section) present more intricate and different connectivity structures.

In practice: use what works best in ImageNet. If you feel a little tired when thinking about architectural decisions, you will be pleased to know that in 90% or more of the applications you should not worry about this. I would like to summarize this point as "do not be a hero": instead of shooting your own architecture for a problem, you should look at the architecture that works best in ImageNet, download a pre-established model and fine-tune it in your data.

You should rarely train a ConvNet from scratch or design one from scratch. I also made this point at the Deep Learning school.

Layer Sizing Patterns

So far we have omitted the mention of common hyperparameters used in each of the layers in a ConvNet. First we will establish the common general rules for sizing the architectures and then follow the rules with a discussion of the notation:

The input layer (which contains the image) should be divisible by 2 many times. Common numbers include 32 (for example, CIFAR-10), 64, 96 (for example, STL-10) or 224 (for example, Common ImageNet Connets), 384 and 512.

Conv layers should use small filters (eg 3x3 or maximum 5x5), using a step of S = 1S = 1, and the most important thing is to fill the input volume with zeros so that the conv layer does not disturb the spatial dimensions of the entrance. That is, when F = 3F = 3, the use of P = 1P = 1 will retain the original size of the input. When F = 5F = 5, P = 2P = 2. For a general FF, it can be seen that P = (F-1) / 2P = (F-1) / 2 preserves the input size. If you must use larger filter sizes (such as 7x7 or so), it is common to see this in the first conv layer that is looking at the input image.

The group layers are responsible for reducing the resolution of the spatial dimensions of the entrance. The most common configuration is to use the maximum combination with 2x2 receptive fields (that is, F = 2F = 2) and with a stride of 2 (that is, S = 2S = 2). Note that this discards exactly 75% of the activations in an input volume (due to the reduction of sampling in 2 in width and height). Another slightly less common configuration is to use 3x3 receptive fields with a step of 2, but this does. It is unlikely to see receptive field sizes for the maximum combination that are larger than three because the grouping is too slow and aggressive. This usually leads to worse performance.

Reduce size headaches. The scheme presented above is nice because all the CONV layers retain the spatial size of their input, while the POOL layers alone are in charge of sampling the volumes spatially. In an alternative scheme where we use strides greater than 1 or do not fill with zeroes the input in CONV layers, we would have to follow very carefully the input volumes in the entire CNN architecture and make sure that all the steps and filters work. out ", and that the ConvNet architecture is wired nicely and symmetrically.

Why use stride of 1 in CONV?

Small steps work best in practice. In addition, as already mentioned, trench 1 allows us to leave all the spatial descending sampling in the POOL layers, with the CONV layers transforming the input volume only in depth.

Why use padding?

In addition to the aforementioned benefit of maintaining constant spatial sizes after CONV, doing this actually improves performance. If the CONV layers did not fill the entries with zeros and only make valid convolutions, then the size of the volumes would be reduced by a small amount after each CONV, and the information on the edges would be "washed" too quickly.
Commitment based on memory restrictions. In some cases (especially at the beginning of the ConvNet architectures), the amount of memory can be accumulated very quickly with the general rules presented above. For example, filtering a 224x224x3 image with three 3x3 CONV layers with 64 filters each and a fill 1 would create three size activation volumes

[224x224x64]. This can equate to a total of around 10 million activations process, or 72 MB of memory. However, GPUs are usually downed by memory. In practice, people prefer to engage only in the first CONV layer of the network. For example, a compromise could be to use a first CONV layer with filter sizes of 7x7 and step of 2 (as seen in a ZF network). As another example, an AlexNet uses filter sizes of 11x11 and step of 4.

6. Caffe Framework

Caffe provides a complete set of tools for training, testing, adjusting and implementing models, with well-documented examples for all these tasks. As such, it is an ideal starting point for researchers and other developers seeking to access state-of-the-art machine learning. At the same time, it is probably the fastest available implementation of these algorithms, so it is immediately useful for industrial implementation.

6.1. Highlights of Caffe

Modularity: The software is designed from the start to be as modular as possible, allowing easy extension to new data formats, network layers and loss functions. Many of the layers and loss functions are already implemented, and abundant examples show how they are composed of trainable recognition systems for various tasks.

Separation and implementation: Caffe model are defined as configuration files used by the buffer languages of protocol. Caffe also supports a directed acyclic graphics form of network architectures. Upon instantiation, Caffe reserves exactly as much memory as necessary for the network and abstracts from its underlying location on the host or GPU. The change between a CPU and GPU implementation is exactly a function call.

Test coverage. Each module in Caffe has a test, and no new code is accepted in the project without the corresponding tests. This allows rapid improvements and refactoring of the code base, and imparts a welcome sense of reassurance to researchers using the code.

Links of Python and MATLAB: For rapid prototyping and the interface with the existing research code, Caffe provides Python and MATLAB links. Both languages can be used to build networks and classify entries. Python links also expose the solution module for easy prototyping of new training procedures.

Pre-trained reference models: Caffe provides (for academic and non-commercial use, no BSD license) reference models for visual tasks, including the "AlexNet" ImageNet model with variations and the R-CNN detection model. More are scheduled for the launch. Caffe model are defined as configuration files used by the buffer languages of protocol. Caffe supports a directed acyclic graphics form of network architectures. Upon instantiation, Caffe reserves exactly as much memory as necessary for the network and abstracts from its underlying location on the host or GPU. The change between a CPU and GPU implementation is exactly a function call.
Test coverage. Each module in Caffe has a test, and no new code is accepted in the project without the corresponding tests. This allows rapid improvements and refactoring of the code

base, and imparts a welcome sense of reassurance to researchers using the code.

Framework	License	Core language	Binding(s)	CPU	GPU	Open source	Training	Pretrained models	Development
Caffe	BSD	C++	Python, MATLAB	✓	✓	✓	✓	✓	distributed
cuda-convnet	unspecified	C++	Python		✓	✓	✓		discontinued
Decaf	BSD	Python		✓		✓	✓	✓	discontinued
OverFeat	unspecified	Lua	C++,Python	✓				✓	centralized
Theano/Pylearn2	BSD	Python		✓	✓	✓	✓		distributed
Torch7	BSD	Lua		✓	✓	✓	✓		distributed

Table 1: Comparison of popular deep learning frameworks[2]

7. Euclidean Distance

The Euclidean distance is the simple distance of straight line between two points on a Euclidean space. Then the Euclidean space becomes a metric space with this distance. The associated norm is called the Euclidean norm. Older literature refers to the metric as Pythagorean metric.

The Euclidean distance between points p and q is the length of the line segment that connects them $(\mathbf{pq})$. In Cartesian coordinates, if p = (p1, p2, ..., pn) and q = (q1, q2, ..., qn) are two points in euclidean e-space, then the distance (d) of p to q, or q to p is given by the formula of Pythagoras:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

The position of a point in a Euclidean n-space is a Euclidean vector. Then, p and q are Euclidean vectors, starting from the origin of space, and their suggestions indicate two points. The Euclidean norm, or Euclidean length, or the magnitude of a vector measures the length of the vector:

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} = \sqrt{\mathbf{p}\cdot\mathbf{p}},$$

where as the last equation includes the dot product.

A vector can be stated as a line segment started from the origin of the Euclidean space to vector point. If we consider that its length is really the distance from its tail to its tip, then the Euclidean norm of a vector is only a special case of Euclidean distance, whereas the Euclidean distance between its tail and its tip.
The distance between points p and q can have an address (for example, from p to q), so it can be represented by another vector, given by

$$\mathbf{q}-\mathbf{p} = (q_1-p_1, q_2-p_2, \cdots, q_n-p_n).$$

One dimension

In Euclidean geometry, setting two points on a line and choosing one to be the origin to established metric. The length of the line segment between these points defines the unit of distance and the direction from the origin to the second point is defined as the positive direction. This line segment can be translated along the line to construct longer segments whose lengths correspond to multiples of the distance of the unit. In this way, real numbers can be associated with points on the line (such as the distance from the origin to the point) and these are the Cartesian coordinates of the points on what can now be called the real line. As an alternative way to set the metric, instead of choosing two points on the line, choose a point to be the origin, a unit of length, and an address along the line to call positive. The second point is determined uniquely as the point on the line that is at a distance from a positive unit of the origin.
The distance between any two points on the real line is the absolute value of the numerical difference of its coordinates. It is common to identify the name of a point with its Cartesian coordinate. Therefore, if p and q are two points on the real line, then the distance between them is given by:

$$\sqrt{(q-p)^2} = |q-p|.$$

Two dimensions

In the Euclidean plane, if p = (p1, p2) and q = (q1, q2) then the distance is given by

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2}.$$

Three dimensions

In three-dimensional Euclidean space, the distance is

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + (p_3-q_3)^2}.$$

n dimensions

In general, for an n-dimensional space, the distance is

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_i-q_i)^2 + \cdots + (p_n-q_n)^2}.$$

8. Change Point Analysis

Abrupt changes respect of time series is known as change point and detecting this point is Change point detection. The change point detection is applied in weather forecasting, detecting bank transaction fraud, image analyses, and classified data in data analysis, human activity and medical science. Many methodologies are applied to detect change point in time series. Since all methods are capturing Change Point, it is difficult to detect which method is appropriate for a particular field. So, that we are learning and getting the basic concept behind those methods.

There are many questions that a researcher can take into account when performing a change point analysis. Some of these include:

- Has a change occurred?
- If yes, where is the change?
- What is the difference between the data of previous and subsequent change?
    - This may be the exchange rate
    - and / or the values of the parameters before and after the change.
- What is the probability that a change has occurred?
- How safe are we from the location of the change point?
- How many changes have occurred (+ all of the above for each change)?
- Why has there been a change?

Here we are going to discuss about E-divisive, Multirank, KCP and DeCon methods.

8.1. E-divisive

E-divisive detects change points by quantifying how different are the characteristic functions of the distributions of later segments of the time series. In fact, the characteristic functions uniquely describe a probability distribution, the changes in the distribution of characteristic distribution of the signal. This method combines the measure of multivariate divergence.

First we discuss multivariate divergence measure here.

8.1.1 Measuring Differences in Multivariate Distributions

For complex-valued functions φ(·), the complex conjugate of $\overline{\varphi}$ is denoted by φ, and the absolute square |φ|2 is defined as $\overline{\varphi\varphi}$. The Euclidean norm of x ∊ Rd is |x|d, or simply |x| when there is no ambiguity. A primed variable such as Xj is an independent copy of X; that is, X and Xj are independent and identically distributed.

$$\int_{\mathbb{R}^d} |\phi_x(t) - \phi_y(t)|^2 \, w(t) \, dt, \qquad \text{…. (i)}$$

In which w(t) denotes here an arbitrary positive weight function, for which the above integral exists. We use the following weight function,

$$w(t;\alpha) = \left(\frac{2\pi^{d/2}\Gamma(1-\alpha/2)}{\alpha 2^{\alpha}\Gamma((d+\alpha)/2)} |t|^{d+\alpha} \right)^{-1}, \qquad \text{….(ii)}$$

for some fixed constant α ∊ (0, 2). Then, if E|X|α, E|Y |α < ∞, a characteristic function based divergence measure may be defined as

$$\mathcal{D}(X,Y;\alpha) = \int_{\mathbb{R}^d} |\phi_x(t) - \phi_y(t)|^2 \left(\frac{2\pi^{d/2}\Gamma(1-\alpha/2)}{\alpha 2^\alpha \Gamma((d+\alpha)/2)} |t|^{d+\alpha} \right)^{-1} dt.$$

… (iii)

Suppose X, Xj , Fx and Y, Y j Fy, and that X, Xj, Y, and Y j are mutually independent. If E|X|α, E|Y |α < ∞, then we may employ an alternative divergence measure based on Euclidean distances,

$$\mathcal{E}(X,Y;\alpha) = 2E|X-Y|^\alpha - E|X-X'|^\alpha - E|Y-Y'|^\alpha.$$

..(iv)

Lemma 1, for any pair of independent random vectors X, Y ∈ Rd, and for any α ∈ (0, 2), if E(|X|α + |Y |α) < ∞, then E(X, Y ; α) = D(X, Y ; α), E(X, Y ; α) ∈ [0, ∞), and E(X, Y ; α) = 0 if and only if X and Y are identically distributed.

The Lemma 1 motivates a simple empirical divergence measure for multivariate distributions based on a U -statistics. Let Xn = {Xi : i = 1, . . . , n} and Y m = {Yj : j = 1, . . . , m} be independent iid samples from the distribution of X, Y ∈ Rd, respectively, such that E|X|α, E|Y |α < ∞ for some α ∈ (0, 2). Then an empirical divergence measure analogous to Equation (4) may be defined as

$$\widehat{\mathcal{E}}(\boldsymbol{X}_n, \boldsymbol{Y}_m; \alpha) = \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}|X_i - Y_j|^\alpha - \binom{n}{2}^{-1}\sum_{1\le i<k\le n}|X_i - X_k|^\alpha - \binom{m}{2}^{-1}\sum_{1\le j<k\le m}|Y_j - Y_k|^\alpha.$$

….(v)

Additionally, under the null hypothesis of equal distributions, i.e., E(X, Y ; α) = 0, we note that $\frac{2}{\mathrm{mn}}$E (Xn, Y m; α) converges in distribution to a non-degenerate random variable as m ∧ n → ∞.

Further, under the alternative hypothesis of unequal distributions, i.e., E(X, Y ; α) > 0, we note that mn E (Xn, Y m; α) → ∞ almost surely as m ∧ n → ∞. These asymptotic Estimating the Location of a Change Point

$$\widehat{\mathcal{Q}}(\boldsymbol{X}_n, \boldsymbol{Y}_m; \alpha) = \frac{mn}{m+n}\widehat{\mathcal{E}}(\boldsymbol{X}_n, \boldsymbol{Y}_m; \alpha)$$

….(vi)

8.1.2. KCP

The Kernel Change Point (KCP) method proposed for detects change points by evaluating how similar or dissimilar the scores at the observed time points are to each other. To this end, the observations are transformed to similarities by means of a kernel function.

1. Compute pairwise similarities using a Gaussian kernel function.

For each pair of observations, X i and X j , the pairwise similarity is computed using a Gaussian kernel function,

$$k(\boldsymbol{X}_i,\ \boldsymbol{X}_j) = \exp\left(\frac{-\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2}{2h^2}\right)$$ … (vii)

The similarities take on values close to 0 when X i and X j are distant and values close to 1 when X i and X j are similar. The bandwidth, h, is a smoothing parameter that indicates how strict one is when deciding if two observations are similar. Here we determined the bandwidth using the procedure of Arlot et al.

1. For different numbers of change points K, minimize the total intra-phase scatter to detect their location.

For varying numbers of change points, K = 0, …, K max , KCP minimizes the following criterion across all possible change point locations (τ 1, τ 2, …, τ K):

$$\widehat{R}(\tau_1, \tau_2, \ldots, \tau_K) = \frac{1}{n}\sum_{k=0}^{K}\widehat{V}_k$$

Where, V̂k is the intra-phase scatter. V̂k measures how homogeneous the corresponding phase is,

$$\widehat{V}_k = (\tau_k - \tau_{k-1}) - \frac{1}{\tau_k - \tau_{k-1}}\sum_{i=\tau_{k-1}+1}^{\tau_k}\sum_{j=\tau_{k-1}+1}^{\tau_k} k(\boldsymbol{X}_i, \boldsymbol{X}_j)$$

8.1.3. Multirank

Multirank makes use of a homogeneity statistic. Multirank only takes the rank order of the scores per variable into account. The method consists of two steps.

1. Check whether the time series contains at least one significant change point.

Considering all possible τ-values, the sequence is divided into two phases X 1 : τ and X τ + 1 : n . For each τ-value, the dissimilarity of these phases is determined by computing the following homogeneity statistic

$$\widehat{T}(\tau) = \frac{4}{n^2}\left[(\tau)\,\boldsymbol{R}_1'\widehat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{R}_1 + (n-\tau)\,\boldsymbol{R}_2'\widehat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{R}_2\right]$$ … (viii)

Where, $\widehat{\Sigma}$ is the empirical covariance matrix of the rank orders of the scores, and $\overline{\mathrm{R}}$ is a phase specific vector containing deviations of the observed mean phase ranks from the expected mean phase rank if the whole sequence is homogeneous. In case of homogeneity, the rank order of a score is completely random and, thus, the expected mean rank within a phase equals $\frac{n+1}{2}$. However, if a change point segments the sequence into phases with different distributions, the rank orders would no longer be random but would depend on the distributions. Consequently, deviations from the middle phase are classified from the expected range under homogeneity, and therefore also T, would be large. Therefore, to decide whether the time series contains at least one change point, the importance of the highest T value is tested by computing the associated asymptotic p value under the assumption of homogeneity. Details on this calculation, which is based on the Bessel functions of the first type and the gamma function.

Equations below shows the T values that were obtained for our illustrative example using different values τ, and indicates that τ = 25 gives the highest T value. This implies a possible point of change in the twenty-sixth observation. Specifically, the maximum homogeneity statistic is equal to 40.27, since

$$\overline{\boldsymbol{R}}_1 = \begin{bmatrix} \frac{R_1^{(1)}+R_2^{(1)}+R_3^{(1)}+\cdots+R_{25}^{(1)}}{25} - \frac{n+1}{2} \\ \frac{R_1^{(2)}+R_2^{(2)}+R_3^{(2)}+\cdots+R_{25}^{(2)}}{25} - \frac{n+1}{2} \\ \frac{R_1^{(3)}+R_2^{(3)}+R_3^{(3)}+\cdots+R_{25}^{(3)}}{25} - \frac{n+1}{2} \end{bmatrix} = \begin{bmatrix} \frac{31+26+33+\cdots+30}{25} - 25.5 \\ \frac{14+22+15+\cdots+16}{25} - 25.5 \\ \frac{19+16+24+\cdots+15}{25} - 25.5 \end{bmatrix} = \begin{bmatrix} -9.22 \\ -12.30 \\ -12.50 \end{bmatrix}$$

$$\overline{\boldsymbol{R}}_2 = \begin{bmatrix} \frac{R_{26}^{(1)}+R_{27}^{(1)}+R_{28}^{(1)}+\cdots+R_{50}^{(1)}}{25} - \frac{n+1}{2} \\ \frac{R_{26}^{(2)}+R_{27}^{(2)}+R_{28}^{(2)}+\cdots+R_{50}^{(2)}}{25} - \frac{n+1}{2} \\ \frac{R_{26}^{(3)}+R_{27}^{(3)}+R_{28}^{(3)}+\cdots+R_{50}^{(3)}}{25} - \frac{n+1}{2} \end{bmatrix} = \begin{bmatrix} \frac{12+49+25+\cdots+50}{25} - 25.5 \\ \frac{29+49+31+\cdots+40}{25} - 25.5 \\ \frac{29+48+31+\cdots+41}{25} - 25.5 \end{bmatrix} = \begin{bmatrix} 9.22 \\ 12.30 \\ 12.50 \end{bmatrix}$$

and

$$\widehat{\Sigma}^{-1} = \begin{bmatrix} 8.48 & -0.83 & -6.09 \\ -0.83 & 11.75 & -9.45 \\ -6.09 & -9.45 & 16.03 \end{bmatrix}.$$

In Step 1, $\overline{\mathrm{R}}2$ is always equal to $-\overline{\mathrm{R}}1$, since we are looking for one change point. When considering multiple change points, this property will of course not hold. The associated p-value for the maximal $\widehat{T}$ is 1.38 × 10− 7, confirming that the change point, T = 26, is highly significant. Henceforward, we will denote the maximal $\widehat{T}$ as $\widehat{T}$max to decide on the number of change points and on their location.

If the change point obtained in Step 1 is found to be significant, multiple change point detection is conducted by computing the generalized form of the homogeneity statistic in Eq. (iii), where Kdenotes the number of change points, τ 0 = 0 and τ K + 1 = n:

$$\widehat{T}\left(\tau_1, \tau_2, \ldots \tau_K\right) = \frac{4}{n^2} \sum_{k=0}^{K} \left(\tau_{k+1} - \tau_k\right) \overline{\boldsymbol{R}}_k' \widehat{\boldsymbol{\Sigma}}^{-1} \overline{\boldsymbol{R}}_k \quad \text{(ix)}$$

8.1.4. DeCon

DeCon based detection of the point of change in the identification of outliers using robust statistics. The method slides a time window of size W along the time series by sequentially deleting the first time point in the window and adding a new observation as the last time point. By window, it is determined if the last point of time is an outlier with respect to the distribution of the other time points in the window. If the latter is the case for multiple consecutive windows, this indicates that the observations that are added to the window may come from a different distribution and, therefore, that a change point occurred. Specifically, DeCon consists of the following four steps.1.

Apply Robust PCA in each time window and determine “outlyingness” of the last time point.

By time window, DeCon calculates a robust multivariate center, μ w, and a covariance matrix, Σ w, to determine the distribution of regular observations (standardized by variable since we are interested in correlations instead of covariances), and generates a peripheral measure for the last time point of the window. To this end, the robust principal components approach (ROBPCA) of Hubert et al. it is used [24]. In this document, we keep all the main components to avoid the question of how to choose the optimal number of components.1 Since we use all the components, the measure of distance is the so-called distance of score, which is equal to the Mahalanobis distance between the last time point X is the last and the robust specific center of the window μ w:

$$SD_{last} = \sqrt{\left(\boldsymbol{X}_{last} - \boldsymbol{\mu}_w\right)^T \sum\nolimits_w^{-1} \left(\boldsymbol{X}_{last} - \boldsymbol{\mu}_w\right)} \quad \text{(x)}$$

9. Problems with previous approaches

➔ Can’t distinguish shot-breaks with

- Cannot apply to all types of video streams.
- Fast object motion or Camera motion.
- Fast Illumination changes
- Reflections from glass, water
- Flash photography

➔ Fails to detect long and short gradual transitions

9.1. Experiment Details

9.1.1. Proposed approach

Video segment detection can be determined online and offline both ways. Here we proposed a new and efficient way of segment detection online/automatic. A video is consists of different segment of frames. However, each segment can be classified any of the following,
i) Sharp/Hard transition
ii) Gradual transition and
iii) No transition
Here, we used *sliding window* technique to do the image classification. A sliding window is a rectangular shape region of a fixed width and height that “slides” across an image. We can do that iteratively for rest of the images also.
We have prepared a learning DB with millions of images. But we did the experiment on a set of sample 500(five hundreds) frames(prepared manually with a mix-up of hard and gradual cuts). We used sliding window technique on those 500 frames to detect Hard-cuts among those.

To make the proposed technique more wide and efficient we introduced a custom CNN model, created by us only. This CNN consists of five convolutional layers(refer the table below). There all the convolutional layers are followed by ReLU(Rectified Linear Unit). Also, there are three fully-connected layers FC6, FC7 and FC8. Those are containing 4096 neurons each.

We used *Caffe* to extract and prepare image DB as learning data set. Also the code development of Change Point Detection algorithm is done partially.

Layer	Kernel	Followed by
Data	label	
Conv1	(11x4x0x3) x 96	ReLU
Pool1		
Conv2	(5 x 1 x 2 x 96) x 256	ReLU
Pool2		
Conv3	(3 x 1 x 1 x 256) x 384	ReLU
Conv4	(3 x 1 x 1 x 384) x 384	ReLU
Conv5	(3 x 1 x 1 x 384) x 256	ReLU
Pool5		
FC6	4096	ReLU Dropout
FC7	4096	ReLU Dropout
FC8	1000	Dropout
Softmax	Label	

Table 2: Proposed custom CNN model

Our experiment followed the steps mentioned below,

Step 1: Extract video and prepare image DB(Train & Test data set)

Identified any video format(e.g. *mp4, .mpg, .wav* etc.)
Extracted video into several frames.
Created Caffe model(CNN) for preparing training data set.
Prepared train data set by creating mean image file and .*lmbd* file.
Extracting feature from another video frames to prepare test data set and saved in a file.

Step 2: Calculate distance between two images by Euclidean Distance algorithm
Used Euclidean distance algorithm to measure distance between two feature points of a particular frame.
Apply Log(10) on the output of the above to get the scalable values.
Generate graphical representation of the distance data.
Compare with a manually prepared tabular data(data of 500 frames) of Hard-cut and Gradual-cut detection with the above graph. To determine the accuracy.

Step 3: Apply Change Point Detection algorithm to determine the Hard-cut automatically
Apply Change Point Detection (CPD) algorithm on the data generated by Euclidean Distance program.
Generating graphical representation of CPD data by a standard available tool.
Compare with a manually prepared tabular data (data of 500 frames) of Hard-cut and Gradual-cut detection and the graph generated by Euclidean distance program with the above graph. To determine the accuracy and efficiency.

We did the above experiment steps for sample 500 frames. In future we will do the continuation of the above steps for next set of test data frames.

9.1.2. Source Code

<<caffe_feature_extractor.py>>

```
import numpy as np
import os, sys, getopt

# Main path to your caffe installation
caffe_root = 'caffe/'

# Model prototxt file : (Model definition: A prototxt file containing the model definition (like the one we
had earlier))
model_prototxt = caffe_root + 'models/bvlc_reference_caffenet/deploy.prototxt'

# Model caffemodel file ( Learning algorithm: A prototxt file describing the parameters for the
stochastic gradient algorithm. This is called the solver file.)
model_trained = caffe_root +
'examples/imagenet/training_05.05.18/model_train_anni005_iter_1000.caffemodel'
```

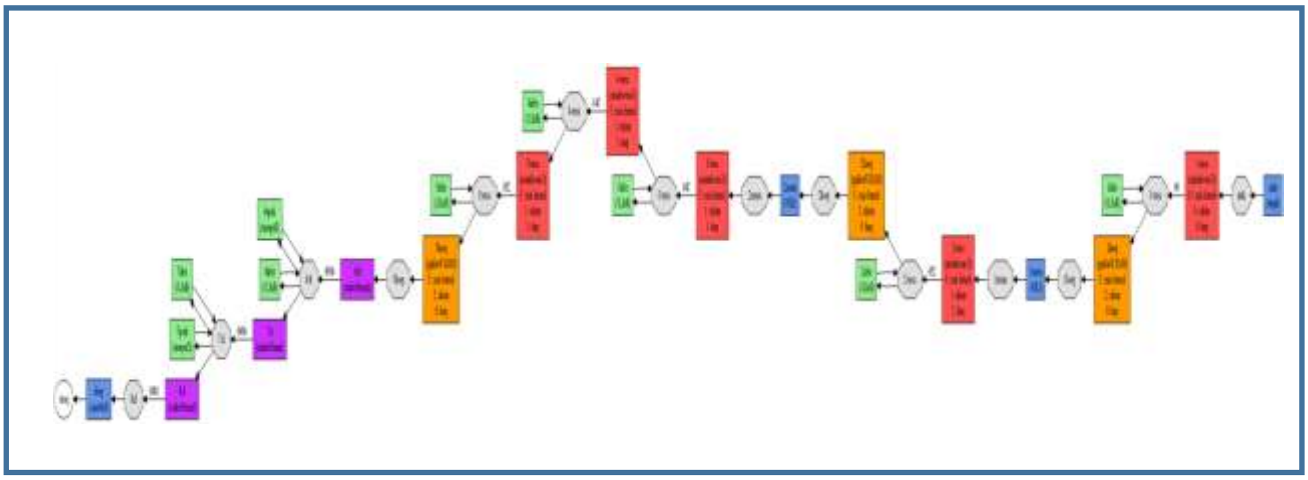
```
# Path to the mean image (used for input processing)(  Mean image: We need to compute the mean
image of the training dataset)
#mean_path = caffe_root + 'python/caffe/imagenet/ilsvrc_2012_mean.npy'
mean_path = caffe_root + 'python/caffe/imagenet/ilsvrc_2012_mean.npy'


# Name of the layer we want to extract

layer_name = 'fc8'
sys.path.insert(0, caffe_root + 'python')
import caffe

def main(argv):
inputfile = ''
outputfile = ''

try:
opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
except getopt.GetoptError:
print ('caffe_feature_extractor.py -i Wildlife.wmv -o hi.txt')
sys.exit(2)

for opt, arg in opts:
if opt == '-h':
print ('caffe_feature_extractor.py -i Wildlife.wmv -o hi.txt')
sys.exit()
elif opt in ("-i"):
inputfile = arg
elif opt in ("-o"):
outputfile = arg

print ('Reading images from "', inputfile)
print ('Writing vectors to "', outputfile)

# Setting this to CPU, but feel free to use GPU if you have CUDA installed
    caffe.set_mode_cpu()
    # Loading the Caffe model, setting preprocessing parameters
    net = caffe.Classifier(model_prototxt, model_trained,
                           mean=np.load(mean_path).mean(1).mean(1),
                           channel_swap=(2,1,0),
                           raw_scale=255,
                           image_dims=(256, 256))

# Processing one image at a time, printint predictions and writing the vector to a file
    with open(inputfile, 'r') as reader:
        with open(outputfile, 'wb') as writer:
            writer.truncate()
            for image_path in reader:
                image_path = image_path.strip()
                input_image = caffe.io.load_image(image_path)
                prediction = net.predict([input_image], oversample=False)
                #print (os.path.basename(image_path), ' : ' , labels[prediction[0].argmax()].strip() , ' (',
prediction[0][prediction[0].argmax()] , ')')
                np.savetxt(writer, net.blobs[layer_name].data[0].reshape(1,-1), fmt='%f')

if __name__ == "__main__":
    main(sys.argv[1:])
```

<<euclidist.py>>

```
#!/usr/bin/python2.7

#import file
import numpy as np
import sys
import math
import collections
#import matplotlib.pyplot as plt
from scipy.spatial import distance

#count number of line in file
num_lines = sum(1 for line in open('output_test_new_anni005_9.txt'))
#initialise two list , 1st for current image , 2nd fr previous image
list1 = list()
list2 = list()

file = open("df_anni005_9.txt", "a")
for i in range(0,num_lines):
        #print ('i =',i)
        frame=i
        prv_frame=frame-1
        if prv_frame != 0:

                with open('output_test_new_anni005_9.txt') as f:
                        for j, line in enumerate(f,1):
                                if j == frame: #here frame is line number
                                        break
                #for save perticular line caontainting strng
                with open('output_test_new_anni005_9.txt') as f:
                        for j, pre_line in enumerate(f,1):
                                #here prv_frame is line number
                                if j == prv_frame:
                                        break

                list1 = line.split()
                list2 = pre_line.split()

                length = len(list1)
                print("length= ",length)

                aList = list()
                bList = list()
                #string to float conversion
                for p in range(0,length):
                        u = float(list1[p])
                        v = float(list2[p])
                        # new list containg float type data
                        aList.append(u);
                        # new list containg float type data
         List.append(v);
                        # caluate euclidean distance beween two list
                        dist = distance.euclidean(aList,bList)

                        #log of difrence bewteen two images
                # log of  negative or Zero  value not allow
                if dist > 0.0:
         df_file_print_value=str(math.log(dist))
                        file.write(df_file_print_value)
```

```
                    file.write('\n')
file.close()
```

9.1.3.Proposed custom CNN model



9.1.4. Experimental Result

We used as many as data set of 500 frames to do the experiment. The frames are contained many video clips and from various situations. The output from our proposed algorithm were good and it was able to detect almost all cuts. We observed from the CPD graph that the two cuts are distinguished automatically as two high peaks. Our proposed approach that use Euclidean distance and Change Point Detection algorithm with Caffe to detect video shot boundary. The overall detection percent is almost accurate and can be applied for all types of video data.

The below data prepared with visually identified cuts manually on the first 500 frames to compare with the Euclidean Distance output followed by Change Point Detection graph to measure accuracy of our proposed approach.

Frame No.	Frame Name	Cut	Euclidean Distance
1	anni001_1.jpg		2.372311149
2	anni001_2.jpg		2.472317729
3	anni001_3.jpg		3.292649839
4	anni001_4.jpg		3.008475481
5	anni001_5.jpg		3.091585992
6	anni001_6.jpg		3.083551949
7	anni001_7.jpg		3.196543483
8	anni001_8.jpg		2.990283976
9	anni001_9.jpg		2.686553106
10	anni001_10.jpg		2.677382687
11	anni001_11.jpg		2.546326467
12	anni001_12.jpg		2.275234168

13	anni001_13.jpg		2.645647537
14	anni001_14.jpg		2.534478245
15	anni001_15.jpg		2.425457159
16	anni001_16.jpg		2.427176684
17	anni001_17.jpg		2.683631753
18	anni001_18.jpg		2.584035578
19	anni001_19.jpg		2.328441865
20	anni001_20.jpg		2.618888173
21	anni001_21.jpg		1.608908404
22	anni001_22.jpg	Hard cut	4.59233453
23	anni001_23.jpg		3.277518624
24	anni001_24.jpg		2.287889863
25	anni001_25.jpg		1.468299913
26	anni001_26.jpg		1.346178963
27	anni001_27.jpg		3.213042358
28	anni001_28.jpg		1.81430212
29	anni001_29.jpg		2.3486974
30	anni001_30.jpg		2.199510776
31	anni001_31.jpg	Gradual	1.766347775
32	anni001_32.jpg	Gradual	2.914009321
33	anni001_33.jpg	Gradual	2.128705341
34	anni001_34.jpg	Gradual	2.915136852
35	anni001_35.jpg	Gradual	2.765753878
36	anni001_36.jpg	Gradual	3.162491566
37	anni001_37.jpg	Gradual	2.079885362
38	anni001_38.jpg	Gradual	2.919569166
39	anni001_39.jpg	Gradual	2.308416366
40	anni001_40.jpg	Gradual	3.019915858
41	anni001_41.jpg	Gradual	2.117173293
42	anni001_42.jpg	Gradual	2.468202951
43	anni001_43.jpg	Gradual	2.676650214
44	anni001_44.jpg	Gradual	2.501996021
45	anni001_45.jpg	Gradual	3.292066244
46	anni001_46.jpg	Gradual	2.723324978
47	anni001_47.jpg	Gradual	2.607081764
48	anni001_48.jpg	Gradual	2.946700328
49	anni001_49.jpg	Gradual	3.478223503
50	anni001_50.jpg	Gradual	2.60265213
51	anni001_51.jpg	Gradual	2.422435248
52	anni001_52.jpg	Gradual	2.722859268
53	anni001_53.jpg	Gradual	3.807084156
54	anni001_54.jpg	Gradual	2.484325919
55	anni001_55.jpg	Gradual	2.554507882
56	anni001_56.jpg	Gradual	2.814518037
57	anni001_57.jpg	Gradual	2.635845012
58	anni001_58.jpg	Gradual	3.160277832
59	anni001_59.jpg	Gradual	2.35751777

60	anni001_60.jpg	Gradual	3.039805922
61	anni001_61.jpg		1.875702485
62	anni001_62.jpg		2.818319634
63	anni001_63.jpg		1.709959607
64	anni001_64.jpg		1.961265361
65	anni001_65.jpg		2.304118512
66	anni001_66.jpg		2.268603176
67	anni001_67.jpg		3.011075958
68	anni001_68.jpg		2.139190293
69	anni001_69.jpg		2.912950989
70	anni001_70.jpg		2.424098782
71	anni001_71.jpg		2.939618561
72	anni001_72.jpg		2.040064726
73	anni001_73.jpg		2.164802965
74	anni001_74.jpg		2.853242306
75	anni001_75.jpg		2.94785229
76	anni001_76.jpg		1.574795553
77	anni001_77.jpg		2.010116893
78	anni001_78.jpg		3.009430543
79	anni001_79.jpg		2.238432241
80	anni001_80.jpg		2.997192861
81	anni001_81.jpg		1.96341512
82	anni001_82.jpg		2.737577768
83	anni001_83.jpg		2.064766462
84	anni001_84.jpg		2.678618999
85	anni001_85.jpg		1.710117954
86	anni001_86.jpg		2.114387792
87	anni001_87.jpg		2.586694238
88	anni001_88.jpg		2.887509345
89	anni001_89.jpg		1.501170273
90	anni001_90.jpg		1.759680405
91	anni001_91.jpg		2.545562561
92	anni001_92.jpg		2.078365156
93	anni001_93.jpg		2.467979355
94	anni001_94.jpg		2.049099989
95	anni001_95.jpg		2.639517466
96	anni001_96.jpg		2.157615009
97	anni001_97.jpg		2.271067626
98	anni001_98.jpg		1.626891914
99	anni001_99.jpg		2.061541127
100	anni001_100.jpg		2.637538338
101	anni001_101.jpg	Gradual	2.037270321
102	anni001_102.jpg	Gradual	2.828735749
103	anni001_103.jpg	Gradual	2.262074936
104	anni001_104.jpg	Gradual	2.591043238
105	anni001_105.jpg	Gradual	2.040332379
106	anni001_106.jpg	Gradual	2.390193177

107	anni001_107.jpg	Gradual	1.677438875
108	anni001_108.jpg	Gradual	1.84501546
109	anni001_109.jpg	Gradual	2.576320622
110	anni001_110.jpg	Gradual	2.92903641
111	anni001_111.jpg	Gradual	2.470384973
112	anni001_112.jpg	Gradual	2.414413622
113	anni001_113.jpg	Gradual	2.847917678
114	anni001_114.jpg	Gradual	2.607051655
115	anni001_115.jpg	Gradual	2.897224676
116	anni001_116.jpg	Gradual	2.918955478
117	anni001_117.jpg	Gradual	2.80967482
118	anni001_118.jpg	Gradual	2.140603217
119	anni001_119.jpg	Gradual	3.051608433
120	anni001_120.jpg	Gradual	2.984702314
121	anni001_121.jpg	Gradual	3.101980611
122	anni001_122.jpg	Gradual	3.152897768
123	anni001_123.jpg	Gradual	3.205286507
124	anni001_124.jpg	Gradual	3.102123972
125	anni001_125.jpg	Gradual	2.731059328
126	anni001_126.jpg	Gradual	3.195172172
127	anni001_127.jpg	Gradual	3.274224193
128	anni001_128.jpg	Gradual	2.759603423
129	anni001_129.jpg	Gradual	2.964772462
130	anni001_130.jpg	Gradual	3.290638678
131	anni001_131.jpg	Gradual	3.142344008
132	anni001_132.jpg	Gradual	2.650602808
133	anni001_133.jpg	Gradual	3.129092004
134	anni001_134.jpg	Gradual	3.046432328
135	anni001_135.jpg		3.132325632
136	anni001_136.jpg		2.632660087
137	anni001_137.jpg		2.831439151
138	anni001_138.jpg		2.936047985
139	anni001_139.jpg		2.940735758
140	anni001_140.jpg		2.864244205
141	anni001_141.jpg		1.716812703
142	anni001_142.jpg		2.754905683
143	anni001_143.jpg		2.879451241
144	anni001_144.jpg		3.100254558
145	anni001_145.jpg		2.43382038
146	anni001_146.jpg		2.85575177
147	anni001_147.jpg		2.880969742
148	anni001_148.jpg		2.703234085
149	anni001_149.jpg		2.767660791
150	anni001_150.jpg		1.652258482
151	anni001_151.jpg		2.903561407
152	anni001_152.jpg		2.890983316
153	anni001_153.jpg		2.562448527

154	anni001_154.jpg		1.675581158
155	anni001_155.jpg		2.519680089
156	anni001_156.jpg		2.671782734
157	anni001_157.jpg		2.879550763
158	anni001_158.jpg		3.19177633
159	anni001_159.jpg		2.358425252
160	anni001_160.jpg		2.537391953
161	anni001_161.jpg		2.605561721
162	anni001_162.jpg		2.502102595
163	anni001_163.jpg		1.999379215
164	anni001_164.jpg		2.713181991
165	anni001_165.jpg		2.701987599
166	anni001_166.jpg		2.677063935
167	anni001_167.jpg		1.333125398
168	anni001_168.jpg		2.659251638
169	anni001_169.jpg	Hard cut	4.147412158
170	anni001_170.jpg		2.340463684
171	anni001_171.jpg		2.334040049
172	anni001_172.jpg		2.148572166
173	anni001_173.jpg		2.189329431
174	anni001_174.jpg		2.525779676
175	anni001_175.jpg		2.209203903
176	anni001_176.jpg		2.286863783
177	anni001_177.jpg		1.858857809
178	anni001_178.jpg		2.083425053
179	anni001_179.jpg		2.590954559
180	anni001_180.jpg		2.250517734
181	anni001_181.jpg		1.472130203
182	anni001_182.jpg		1.663879673
183	anni001_183.jpg		1.922767911
184	anni001_184.jpg		2.221062751
185	anni001_185.jpg		2.151477698
186	anni001_186.jpg		2.252025301
187	anni001_187.jpg		2.300662441
188	anni001_188.jpg		2.066069344
189	anni001_189.jpg		2.272311232
190	anni001_190.jpg		1.593685018
191	anni001_191.jpg		2.154367557
192	anni001_192.jpg		1.852209914
193	anni001_193.jpg		2.160617184
194	anni001_194.jpg		1.548258308
195	anni001_195.jpg	Gradual	2.076659122
196	anni001_196.jpg	Gradual	1.985669505
197	anni001_197.jpg	Gradual	1.928019176
198	anni001_198.jpg	Gradual	2.138688441
199	anni001_199.jpg	Gradual	1.500894935
200	anni001_200.jpg	Gradual	2.458917429

201	anni001_201.jpg	Gradual	2.195483596
202	anni001_202.jpg	Gradual	2.034488418
203	anni001_203.jpg	Gradual	1.713843435
204	anni001_204.jpg	Gradual	2.29696519
205	anni001_205.jpg	Gradual	2.087086247
206	anni001_206.jpg	Gradual	2.207304464
207	anni001_207.jpg	Gradual	2.529495068
208	anni001_208.jpg	Gradual	2.31011824
209	anni001_209.jpg	Gradual	2.566313904
210	anni001_210.jpg	Gradual	2.330319092
211	anni001_211.jpg	Gradual	2.452178364
212	anni001_212.jpg	Gradual	2.301886957
213	anni001_213.jpg	Gradual	2.61262965
214	anni001_214.jpg	Gradual	2.877291339
215	anni001_215.jpg	Gradual	2.447600416
216	anni001_216.jpg	Gradual	2.68785057
217	anni001_217.jpg	Gradual	2.923446921
218	anni001_218.jpg	Gradual	3.128421377
219	anni001_219.jpg	Gradual	3.215979284
220	anni001_220.jpg	Gradual	2.805527558
221	anni001_221.jpg	Gradual	3.567243138
222	anni001_222.jpg	Gradual	3.033979689
223	anni001_223.jpg	Gradual	2.788983576
224	anni001_224.jpg	Gradual	2.636418607
225	anni001_225.jpg	Gradual	2.92239941
226	anni001_226.jpg	Gradual	2.654488178
227	anni001_227.jpg	Gradual	2.933049668
228	anni001_228.jpg	Gradual	3.026882014
229	anni001_229.jpg	Gradual	3.019275576
230	anni001_230.jpg	Gradual	2.736718337
231	anni001_231.jpg	Gradual	2.860627072
232	anni001_232.jpg	Gradual	2.56626739
233	anni001_233.jpg	Gradual	2.005917123
234	anni001_234.jpg	Gradual	2.659270687
235	anni001_235.jpg	Gradual	3.110597221
236	anni001_236.jpg		2.944252075
237	anni001_237.jpg		2.223209163
238	anni001_238.jpg		3.303311044
239	anni001_239.jpg	Hard cut	4.701153093
240	anni001_240.jpg		2.584570166
241	anni001_241.jpg		2.813803063
242	anni001_242.jpg		2.770093364
243	anni001_243.jpg		2.915722563
244	anni001_244.jpg		2.76197826
245	anni001_245.jpg		2.871506267
246	anni001_246.jpg		2.649379311
247	anni001_247.jpg		2.983792542

248	anni001_248.jpg		2.63886502
249	anni001_249.jpg		2.535198604
250	anni001_250.jpg		2.935642901
251	anni001_251.jpg		2.701523238
252	anni001_252.jpg		3.248673289
253	anni001_253.jpg		3.0025607
254	anni001_254.jpg		2.587986152
255	anni001_255.jpg		2.522247542
256	anni001_256.jpg		2.721716681
257	anni001_257.jpg		2.97171513
258	anni001_258.jpg	Gradual	3.324338853
259	anni001_259.jpg	Gradual	2.831616718
260	anni001_260.jpg	Gradual	3.148719376
261	anni001_261.jpg	Gradual	3.019624462
262	anni001_262.jpg	Gradual	2.937183512
263	anni001_263.jpg	Gradual	3.011531027
264	anni001_264.jpg	Gradual	3.454267224
265	anni001_265.jpg	Gradual	3.247352974
266	anni001_266.jpg	Gradual	3.086704992
267	anni001_267.jpg	Gradual	3.106781883
268	anni001_268.jpg	Gradual	3.247654688
269	anni001_269.jpg	Gradual	3.411258337
270	anni001_270.jpg	Gradual	3.524932015
271	anni001_271.jpg	Gradual	3.880762245
272	anni001_272.jpg	Gradual	3.141304068
273	anni001_273.jpg	Gradual	3.182835021
274	anni001_274.jpg	Gradual	2.761760597
275	anni001_275.jpg	Gradual	3.046690797
276	anni001_276.jpg	Gradual	2.837515227
277	anni001_277.jpg	Gradual	3.169517264
278	anni001_278.jpg	Gradual	3.107585955
279	anni001_279.jpg	Gradual	2.520286612
280	anni001_280.jpg	Gradual	2.518530961
281	anni001_281.jpg	Gradual	2.772013925
282	anni001_282.jpg	Gradual	2.744363623
283	anni001_283.jpg	Gradual	2.603922433
284	anni001_284.jpg	Gradual	2.92508199
285	anni001_285.jpg	Gradual	3.085840884
286	anni001_286.jpg		2.656820798
287	anni001_287.jpg	Hard cut	3.980587018
288	anni001_288.jpg		2.658011324
289	anni001_289.jpg		2.763673596
290	anni001_290.jpg		2.633128989
291	anni001_291.jpg		4.513672839
292	anni001_292.jpg		4.336782323
293	anni001_293.jpg		2.120078046
294	anni001_294.jpg		3.524311777

295	anni001_295.jpg		3.305471463
296	anni001_296.jpg		3.349304773
297	anni001_297.jpg		3.457564341
298	anni001_298.jpg		1.906911403
299	anni001_299.jpg		3.224323135
300	anni001_300.jpg		3.223893284
301	anni001_301.jpg		3.521114741
302	anni001_302.jpg		3.736766198
303	anni001_303.jpg		2.289843457
304	anni001_304.jpg		3.450125513
305	anni001_305.jpg		3.383423063
306	anni001_306.jpg		3.290757089
307	anni001_307.jpg		3.339556711
308	anni001_308.jpg		1.832907185
309	anni001_309.jpg		3.230362434
310	anni001_310.jpg		3.345363697
311	anni001_311.jpg		3.475629748
312	anni001_312.jpg		3.321058267
313	anni001_313.jpg		1.56414426
314	anni001_314.jpg		3.465978968
315	anni001_315.jpg		3.534825832
316	anni001_316.jpg		3.445752338
317	anni001_317.jpg		3.320346553
318	anni001_318.jpg		1.972777532
319	anni001_319.jpg	Hard cut	4.666694186
320	anni001_320.jpg		1.750252109
321	anni001_321.jpg		2.183243127
322	anni001_322.jpg		2.362855551
323	anni001_323.jpg		2.313705135
324	anni001_324.jpg		2.195096966
325	anni001_325.jpg		1.897261255
326	anni001_326.jpg		1.732360402
327	anni001_327.jpg		1.723206658
328	anni001_328.jpg		1.552562322
329	anni001_329.jpg		1.220062188
330	anni001_330.jpg		1.380062188
331	anni001_331.jpg		2.684653144
332	anni001_332.jpg		2.986197699
333	anni001_333.jpg		3.039407139
334	anni001_334.jpg	Hard cut	4.796156191
335	anni001_335.jpg		2.567391351
336	anni001_336.jpg		3.119526007
337	anni001_337.jpg		2.780154007
338	anni001_338.jpg		2.90756686
339	anni001_339.jpg		2.829858299
340	anni001_340.jpg		2.629474746
341	anni001_341.jpg		2.800379968

342	anni001_342.jpg		2.983239113
343	anni001_343.jpg		2.836368251
344	anni001_344.jpg		1.887336265
345	anni001_345.jpg		2.817252387
346	anni001_346.jpg		2.486828932
347	anni001_347.jpg		2.469392777
348	anni001_348.jpg		2.758954434
349	anni001_349.jpg		2.417125631
350	anni001_350.jpg		2.615312412
351	anni001_351.jpg		2.561398891
352	anni001_352.jpg		2.459860697
353	anni001_353.jpg		2.596329231
354	anni001_354.jpg		2.317900357
355	anni001_355.jpg	Hard cut	4.519323829
356	anni001_356.jpg		3.183719063
357	anni001_357.jpg		2.033341615
358	anni001_358.jpg		2.854286158
359	anni001_359.jpg		3.034498845
360	anni001_360.jpg		2.939404825
361	anni001_361.jpg		3.225346566
362	anni001_362.jpg		1.611197958
363	anni001_363.jpg		3.295164882
364	anni001_364.jpg		2.516517881
365	anni001_365.jpg		2.767790863
366	anni001_366.jpg		3.018506661
367	anni001_367.jpg		2.339271914
368	anni001_368.jpg		3.200407972
369	anni001_369.jpg		3.267656976
370	anni001_370.jpg		3.507793075
371	anni001_371.jpg		3.224150167
372	anni001_372.jpg		2.585158331
373	anni001_373.jpg		3.291968448
374	anni001_374.jpg		3.048063295
375	anni001_375.jpg		3.099580452
376	anni001_376.jpg		2.992662311
377	anni001_377.jpg		1.636960066
378	anni001_378.jpg		2.830103816
379	anni001_379.jpg		3.06325817
380	anni001_380.jpg		3.114730145
381	anni001_381.jpg		2.813871432
382	anni001_382.jpg		-1.06448156
383	anni001_383.jpg		3.326713936
384	anni001_384.jpg		3.338700505
385	anni001_385.jpg		3.195491234
386	anni001_386.jpg		3.017529482
387	anni001_387.jpg		2.280879888
388	anni001_388.jpg		3.152706563

389	anni001_389.jpg	2.88819778
390	anni001_390.jpg	3.037484959
391	anni001_391.jpg	2.989974569
392	anni001_392.jpg	-0.051703995
393	anni001_393.jpg	3.259849758
394	anni001_394.jpg	3.006661979
395	anni001_395.jpg	2.78997002
396	anni001_396.jpg	2.93551714
397	anni001_397.jpg	2.326238468
398	anni001_398.jpg	2.680012833
399	anni001_399.jpg	3.114899459
400	anni001_400.jpg	2.903527763
401	anni001_401.jpg	3.018198248
402	anni001_402.jpg	2.283037948
403	anni001_403.jpg	2.93237011
404	anni001_404.jpg	2.91451357
405	anni001_405.jpg	2.786041225
406	anni001_406.jpg	2.803447652
407	anni001_407.jpg	0.422783449
408	anni001_408.jpg	2.936764618
409	anni001_409.jpg	2.80100797
410	anni001_410.jpg	3.018263604
411	anni001_411.jpg	2.977481416
412	anni001_412.jpg	0.956293486
413	anni001_413.jpg	2.808101793
414	anni001_414.jpg	2.734336933
415	anni001_415.jpg	2.741472301
416	anni001_416.jpg	2.907122723
417	anni001_417.jpg	2.42661871
418	anni001_418.jpg	2.982313914
419	anni001_419.jpg	3.029977244
420	anni001_420.jpg	3.005568367
421	anni001_421.jpg	2.592378448
422	anni001_422.jpg	1.418205088
423	anni001_423.jpg	3.125272339
424	anni001_424.jpg	2.652093463
425	anni001_425.jpg	3.264052872
426	anni001_426.jpg	3.019995211
427	anni001_427.jpg	1.306703361
428	anni001_428.jpg	3.011402735
429	anni001_429.jpg	3.088428324
430	anni001_430.jpg	2.923539224
431	anni001_431.jpg	2.900605647
432	anni001_432.jpg	2.174440089
433	anni001_433.jpg	2.96711783
434	anni001_434.jpg	3.154700349
435	anni001_435.jpg	2.900972499

436	anni001_436.jpg		2.846293657
437	anni001_437.jpg		0.818224579
438	anni001_438.jpg		2.879949203
439	anni001_439.jpg	Hard cut	4.102789221
440	anni001_440.jpg		2.806057254
441	anni001_441.jpg		2.918788565
442	anni001_442.jpg		3.075637325
443	anni001_443.jpg		1.483798794
444	anni001_444.jpg		2.989685335
445	anni001_445.jpg		2.707994591
446	anni001_446.jpg		3.13364272
447	anni001_447.jpg		3.230616009
448	anni001_448.jpg		1.518499123
449	anni001_449.jpg		2.997395882
450	anni001_450.jpg		3.122717398
451	anni001_451.jpg		2.51382282
452	anni001_452.jpg		2.553068398
453	anni001_453.jpg		1.68036669
454	anni001_454.jpg		2.60579831
455	anni001_455.jpg		2.692388249
456	anni001_456.jpg		2.790047996
457	anni001_457.jpg		2.793579728
458	anni001_458.jpg		0.424253366
459	anni001_459.jpg		2.902753629
460	anni001_460.jpg		2.854491894
461	anni001_461.jpg		2.793538446
462	anni001_462.jpg		3.041261582
463	anni001_463.jpg		1.112083229
464	anni001_464.jpg		2.846045007
465	anni001_465.jpg		3.177928771
466	anni001_466.jpg		2.927281436
467	anni001_467.jpg		3.12391314
468	anni001_468.jpg		1.749959156
469	anni001_469.jpg		2.808125737
470	anni001_470.jpg		2.533982914
471	anni001_471.jpg		2.942232885
472	anni001_472.jpg		2.558958191
473	anni001_473.jpg		0.24003615
474	anni001_474.jpg		2.781610921
475	anni001_475.jpg		3.035608993
476	anni001_476.jpg		2.829487565
477	anni001_477.jpg		2.845182934
478	anni001_478.jpg		-0.37820916
479	anni001_479.jpg		3.334509614
480	anni001_480.jpg		2.850943233
481	anni001_481.jpg		2.488451781
482	anni001_482.jpg		2.772169674

483	anni001_483.jpg		2.163649427
484	anni001_484.jpg		2.729053808
485	anni001_485.jpg		2.599587211
486	anni001_486.jpg		3.047769154
487	anni001_487.jpg		3.250167239
488	anni001_488.jpg		0.45777339
489	anni001_489.jpg		2.743200732
490	anni001_490.jpg		2.891894077
491	anni001_491.jpg		2.344353025
492	anni001_492.jpg		2.613043542
493	anni001_493.jpg		0.123665649
494	anni001_494.jpg		2.583484435
495	anni001_495.jpg		2.724907864
496	anni001_496.jpg		2.80515425
497	anni001_497.jpg		2.78452747
498	anni001_498.jpg		2.281789633
499	anni001_499.jpg		2.67457847

Table-3: Frames data prepared manually

The graph generated on Euclidean Distances data from the above table mentioned in Appendix 1 in comparison with another graph generated by Change Point Detection algorithm on the above Euclidean Distances as input. This is to measure and show the efficiency and accuracy of our proposed approach.

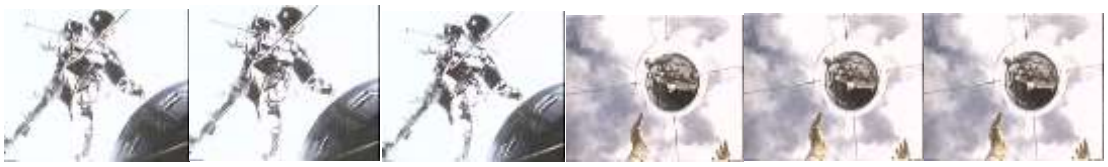


Figure 40-A : Hard Cut detect

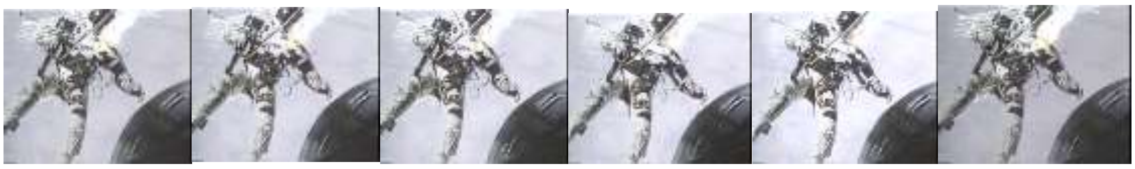


Figure 40-B: Gradual Cut detect

Figure 40: Shot boundary detected in video frames

Here in this section, we summarize the testing results of our proposed method using different video image sequences. Also, the performance of the proposed system is measured using the following different TRECVID evaluation metrics those are defined as

$$\text{Recall (R)} = \frac{\text{Correct}}{\text{Correct} + \text{Miss}} \text{x } 100$$

$$\text{Precision (P)} = \frac{\text{Correct}}{\text{Correct} + \text{False}} \text{x } 100$$

$$\text{F} - \text{Measure (F)} = \frac{\text{2 x Precsion x Recall}}{\text{Precision} + \text{Recall}} \text{x } 100$$

Video	Number of Shots	Number of Frames	Yoo's System (H.W. Yoo, H. J. Ryoo, and D. S. Jang, 2006)			Wenjing Tong's System(Li Song, Xiaokang Yang, Hui Qu, Rong Xie, 2015)			Proposed Method		
			P	R	F	P	R	F	P	R	F
anni001	10	870	0.67	0.67	0.65	1	0.881	0,94	0.99	0.94	0.96
anni005	38	11362	0.84	0.88	0.86	1	0.895	0.932	0.99	0.93	0.96
anni009	38	12305	0.86	0.94	0.90	1	0.821	0.901	0.98	0.94	0.96

Table-4: Comparison with other methods

We downloaded the above used benchmark databases from NIST (http://trecvid.nist.gov/) as it has a commonly used huge evaluation database.

10. Conclusions

This work introduces an efficient and robust system for detecting video scene changes, an essential task in fully content analysis systems. We faced many difficulties when during the installation process. It took almost 1 month to install the necessary libraries. Our module receives frame differences as inputs, then recalls the information stored into the neural network weights to determine the outputs. The algorithm has been tested on varieties of videos. Better generalization of the neural network can be achieved by increasing the number of video clips used in the training phase and by varying their contents. The effectiveness of the proposed paradigm has been proven as a robust and efficient way to identify scene changes in any type of compressed video streams.

11. References

[1] Nikita Sao, Ravi Mishra, A survey based on Video Shot Boundary Detection techniques

https://github.com/BVLC/caffe/

[2] http://caffe.berkeleyvision.org/
[3]Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv preprint arXiv:1408.5093, 2014.

[4]Waleed E. Farag, University of Pennsylvania, USA Hussein Abdel-Wahab, Old Dominion University, USA, Video Shot Boundary Detection

[5]http://mathonline.wikidot.com/the-distance-between-two-vectors

[6]Christian Szegedy, Wei Liu, YangqingJia , Pierre Sermanet, Scott Reed ,DumitruErhan,Vincent Vanhoucke. Andrew Rabinovich ,Going deeper with convolutions

[7]http://searchnetworking.techtarget.com/definition/neural-network

[8]https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm

[9]https://www.slideshare.net/nilmani14/neural-network-3019822

[10]John S. Boreczky Lawrence A. Rowe , Comparison of video shot boundary detection techniques , Journal of Electronic Imaging 5(2), 122–128 (April 1996)

[11]YangqingJia , Evan Shelhamer , Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell, Caffe: Convolutional Architecture for Fast Feature Embedding,ProceedingMM '14 Proceedings of the 22nd ACM international conference on Multimedia , Pages 675-678

[12] Source: https://stfalcon.com/en/blog/post/deep-learning-benefits-and-challenges

[13]http://www.coldvision.io/2016/07/29/image-classification-deep-learning-cnn-caffe-opencv-3-x-cuda/

[14] https://en.wikipedia.org/wiki/Euclidean_distance

[15]https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

[16]https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

[17] http://cs231n.github.io/convolutional-networks/#architectures

[18] MSP Waghmare, AS Bhide - Citeseer, Video Shot Boundary Detection Techniques ,International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 3, Issue 11, November 2014

[19] https://elearningindustry.com/how-digital-video-works-digital-video-101

[20] http://danielhancockphotography.com/uncategorized/histogram/

[21] https://scialert.net/fulltextmobile/?doi=jas.2006.1679.1685

[22] https://www.researchgate.net/figure/Hierarchical-Structure-of-Video_fig1_236142063

[23]https://medium.com/swlh/ill-tell-you-why-deep-learning-is-so-popular-and-in-demand-5aca72628780

[24] Hubert, M., Rousseeuw, P. J., & Vanden Branden, K. (2005). ROBPCA: A new approach to robust principal component analysis. Technometrics, 47, 64–79.

[25] Bulteel, K., Ceulemans, E., Thompson, R., Waugh, C., Gotlib, I., Tuerlinckx, F., & Kuppens, P. (2014). DeCon: A tool to detect emotional concordance in multivariate time series data of emotional responding. Biological Psychology, 98(1), 29–42.

[26] Lung-Yut-Fong, A., Lévy-Leduc, C., & Cappé, O. (2012). Homogeneity and change-point detection tests for multivariate data using rank statistics. Retrieved from

[27] Arlot, S., Celisse, A., & Harchaoui, Z. (2012). Kernel change-point detection. Retrieved from http://arxiv.org/abs/1202.3878

[28] https://arxiv.org/pdf/1306.4933.pdf

[29] https://link.springer.com/article/10.3758%2Fs13428-016-0754-9#Abs1

[30] H. W. Yoo, H. J. Ryoo, and D.S. Jang, 2006, Gradual shot boundary detection using localized edge blocks, Multimedia Tools Appl., vol. 28, pp. 283–300

[31] Wenjing Tong1, Li Song1, Xiaokang Yang1, Hui Qu1 and Rong Xie1, CNN-Based Shot Boundary Detection and Video Annotation

APPENDIX-1

The below graph is generated on the Euclidean Distance algorithm applied between two frames. This distance measured on extracted feature points of frame of the video considered.

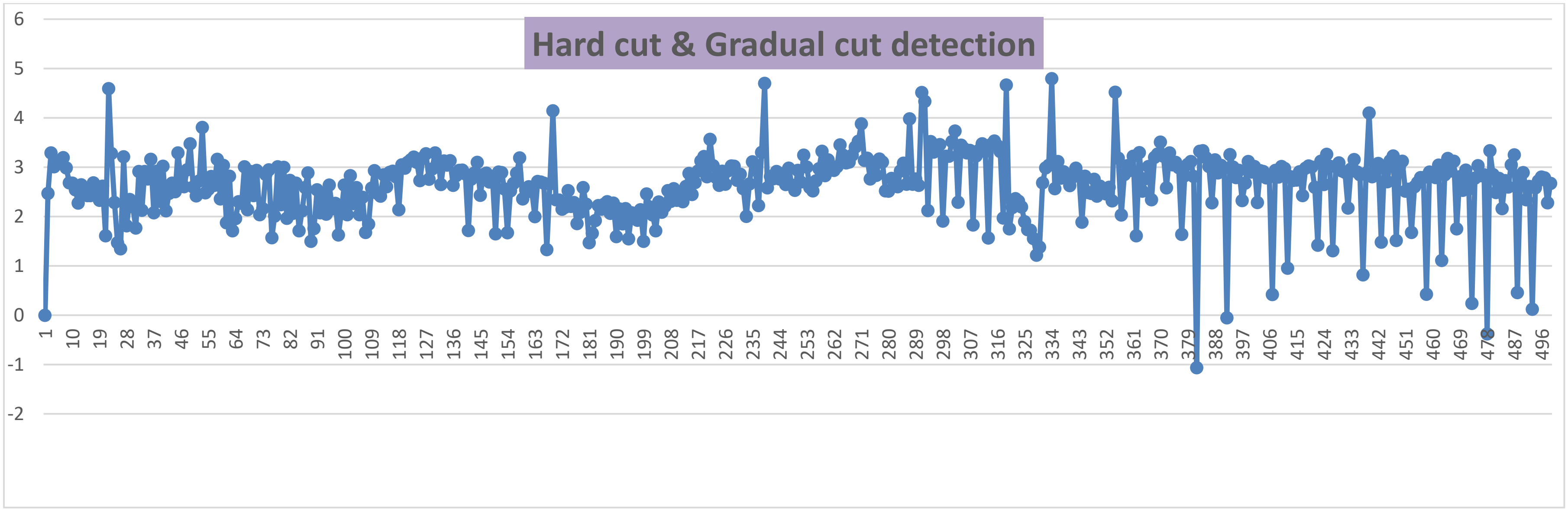


Figure 41: Hard cut and Gradual cut detection

The below graph is generated by the Change Point Detection(CPD) algorithm where earlier calculated Euclidean Distance values between two frames were passed as input.

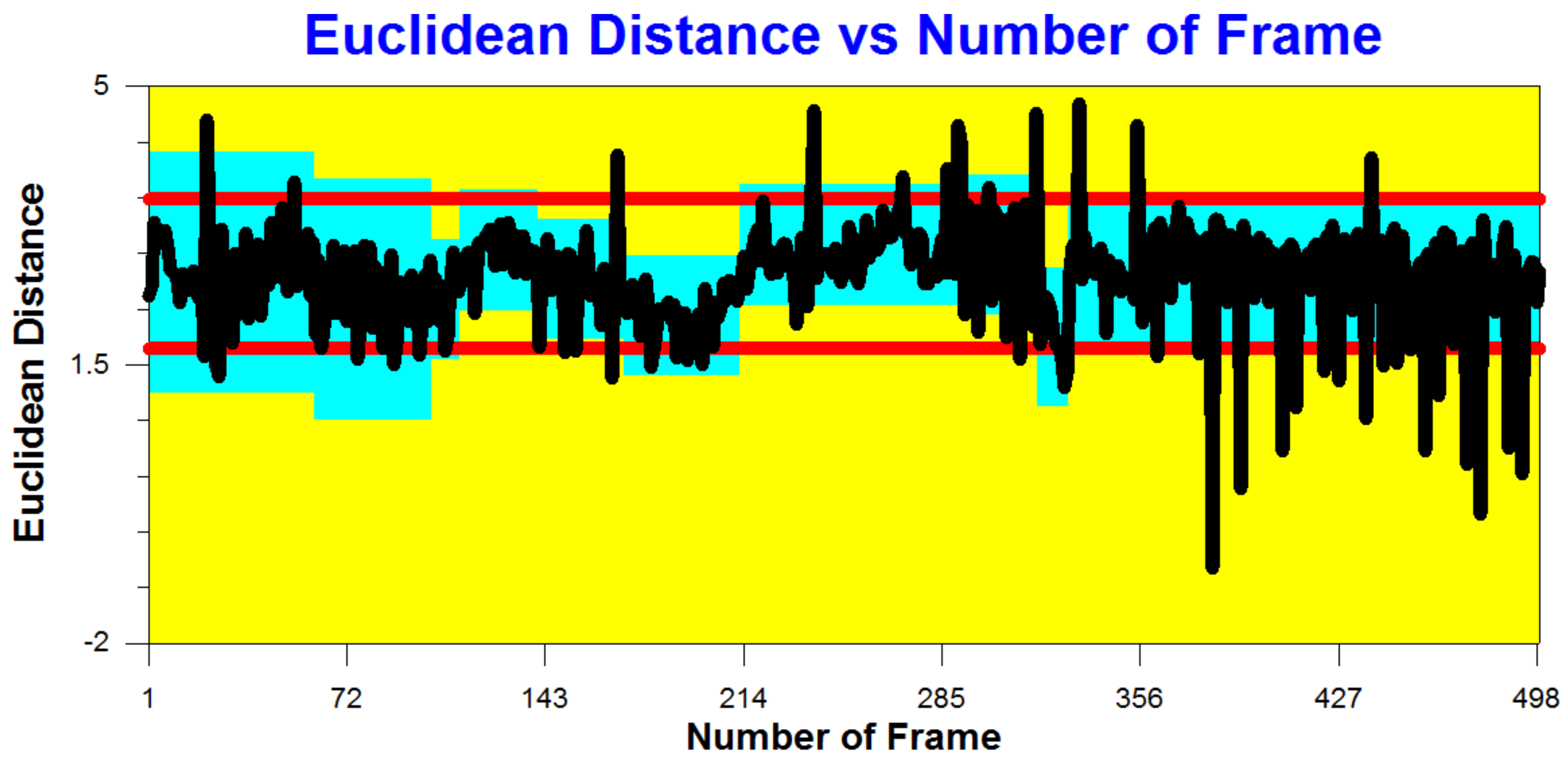


Figure 42: Hard cut and Gradual cut detection by CPD algorithm