Deep Learning for Multimedia Data in IoT

Srinidhi Hiriyannaiah, B. S. Akanksh, A. S. Koushik, G. M. Siddesh and K. G. Srinivasa

Abstract With the advent of Internet leading to proliferation of large amounts of multimedia data, the analytics of the aggregated multimedia data is proven to be one of the active areas of research and study. Multimedia data includes audio, video, images associated with applications like similarity searches, entity resolution, and classification. Visual data mining is now one of the active learning fields that include surveillance applications for object detection, fraud detection, crime detection, and other applications. Multimedia data mining includes many challenges like data volume, variety, and unstructured nature, nonstationary, and real time. It needs advanced processing capabilities to make decisions in near real time. The existing traditional database systems, data mining techniques cannot be used because of its limitations. Hence, to process such large amounts of data advanced techniques like machine learning, deep learning methods can be used. Multimedia data also includes sensor data that is widely generated. Most of the healthcare applications include sensors for detecting heart rate, blood pressure, and pulse rate. The advancement of the smartphones has resulted in fitness based applications based on the number of steps walked, calories count, kilometers ran, etc. All these types of data can be classified as Multimedia data for Internet of Things (IoT). There are many interfacing devices that are interconnected to each other with backbone as a computer network when sensor data is involved. The main aim of this chapter is to highlight the importance and convergence of deep learning techniques with IoT. Emphasis is laid on classification

S. Hiriyannaiah (✉) · B. S. Akanksh · A. S. Koushik · G. M. Siddesh
Ramaiah Institute of Technology, Bengaluru, India
e-mail: srinidhi.hiriyannaiah@gmail.com

B. S. Akanksh
e-mail: bsakanksh@gmail.com

A. S. Koushik
e-mail: askoushik4@gmail.com

G. M. Siddesh
e-mail: siddeshgm@gmail.com

K. G. Srinivasa
National Institute of Technical Teacher Training Research, New Delhi, India
e-mail: kgsrinivasa@gmail.com

S. Tanwar et al. (eds.), *Multimedia Big Data Computing for IoT Applications*, Intelligent Systems Reference Library 163,
https://doi.org/10.1007/978-981-13-8759-3_4

of IoT data using deep learning and the essential fine-tuning of parameters. A virtual sensor device implemented in python is used for simulation. An account of protocols used for communication of IoT devices is briefly discussed. A case study is provided regarding classification of Air Quality Dataset using deep learning techniques.

Keywords Multimedia data · IoT · Air quality analysis · Deep learning · IoT analytics

1 Introduction to Multimedia and IoT

Data Mining has gained a greater significance in recent days due to the large amount of data being collected and its availability over the Internet. The significant advances in the big data technology and tools have lead to the development of analytics and research in this area over the years. There is a paradigm shift of data mining to big data analytics that involves various stages of processing and analysis. Big data analytics consists of exploration of data, identifying the relationships among the different features in the data and visualize it. The applications of big data analytics include various multidisciplinary fields such as machine learning, information retrieval, databases, and visualization. These significant advances have lead to the evolution of multimedia management systems.

Multimedia data includes image, video, audio, and text. Due to the advent of smartphones and Internet, this kind of multimedia data has to lead to several applications and sharing platforms [1]. The valuable sources of multimedia data are social media such as Instagram, YouTube, Twitter, and Facebook [2]. The volume of data being collected at these sites is significantly increasing day by day. For example, in Instagram users have uploaded over 20 billion photos, 100 videos are uploaded every minute per day on YouTube, 500 million tweets on twitter. Such huge amount of information plays an important source for analyzing different patterns and developing applications. This rich source of information is termed as "Big data". Big data posses three essential features namely variety, velocity, and volume. The proliferation of big data in terms of volume, velocity, and variety has reached to different domains as shown in Fig. 1. In this chapter, the focus is on IoT applications and multimedia.

Internet of Things (IoT) is a network of things that are connected to each other which is supported by Internet for communication. Most of the devices nowadays are sensor based and is connected to one or more usually. It is estimated in most of the reviews that the total devices that will be connected to each other will 20 billion by 2020 [3]. With the increase in the number of devices that are connected to each other, the generation of multimedia data also increases. The scope of IoT is not limited to the sensor data alone but also relates to multimedia. For example, CCTV camera captures the video data for surveillance purposes which can be categorized as multimedia data. The fingerprint data can also be regarded as the multimedia data since it is in the form image. The applications of such data are utilized in smart

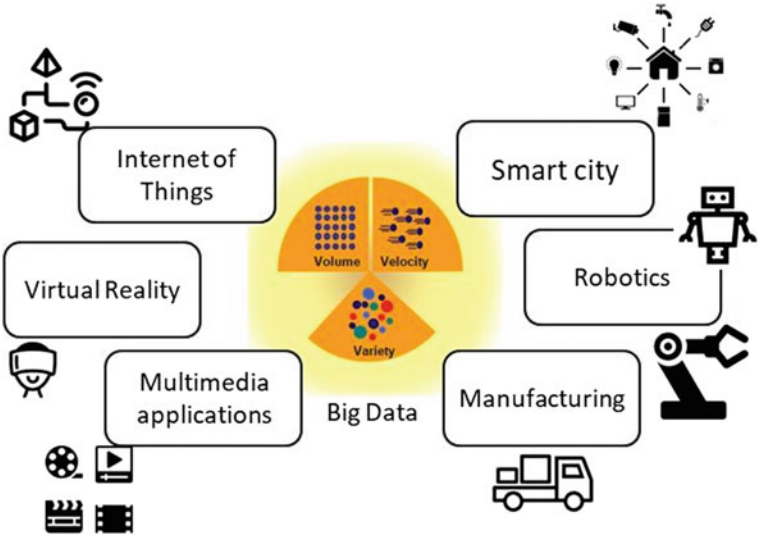

Fig. 1 Big data and applications

communities such as smart city, smart power, smart grid, etc. [3]. Hence, multimedia analytics play a crucial role in the field of IoT as well.

Multimedia analytics can be carried out using conventional analytical approaches such as clustering, decision tree methods. However, a major challenge faced with the conventional approaches is the scalability of the methods and the execution time required to produce the results [4]. In this regard, Deep learning and GPU computing methods are used for multimedia analytics. Deep learning methods that involve deep forward networks, recurrent networks are used most of the times for multimedia analytics. The features that are present in the multimedia data need to be extracted so that clear decision can be made. The main challenge of deep learning is feature extraction process involves stages of object recognition, object detection from the edges identified in the image.

Multimedia applications need a different kind of approach for analysis compared to conventional analytical techniques [4]. The conventional techniques of data analysis such as clustering, regression, naïve bayes classification can be used on the stationary data. Since multimedia data and IoT data is nonstationary, techniques that are based on streaming analysis are useful. The different types of enabling technologies for multimedia analysis and IoT data classification are discussed in the further sections.

The focus of this chapter is to introduce the advances in multimedia data, challenges in multimedia analytics and how deep learning methods can be used for multimedia analytics. An illustration regarding communication of IoT devices is provided. A brief account regarding MQTT protocol and its fundamental operations such as publish and subscribe is discussed. In the final section of the chapter, a case

study on IoT data classification is presented. The chosen dataset for this purpose is Air Quality dataset. It deals with creating a deep learning model to classify IoT data along with the associated code.

2 Advances in Multimedia Data

In the data analytics world, data has been proliferated in different ways. Multimedia data is one of such data that has been recently generated in huge volumes in the Internet. It is being captured by various multimedia computing devices like computers, tablets, mobile phones, and cameras [5]. The nature of the data being captured is also ubiquitous. Initially, started with audio, video it has now reached animations in the form of gifs. The advances in multimedia data are summarized as follows.

- **Visual data**: The most common form of multimedia data found is the video data. A visual data consists of a sequence of image sequences that has to analyze one-by-one. Most of the unstructured information exists in the form of visual data and contains very rich information. Visual data analytics process involves extracting meaningful information from the different image sequences that are present in the visual data. However, the main challenge lies in the size of the visual data. Recent technologies such as cloud computing, high-performance computing have enabled the visual data and analytics research in areas such as video surveillance systems, autonomous systems, and healthcare. The advances in visual data and analytics are challenging the human brain and its computation. In [5] one of the competitions held, machine outperformed the humans in image classification.
- **Audio data**: One more type of multimedia data that is mostly used is audio/speech data. Real-time audio analytical applications are needed in social media, healthcare, and industry. Audio analytics involves extracting useful information from the different pieces of the information present in the audio data. Call centers are one of industry applications that need audio analytics for interaction with the customer and training the persons involved in the call center. Big data platforms such as Spark, Hadoop, and different libraries are widely used for such audio analytical applications.
- **Text data**: Multimedia data may be embedded in the textual context in the form of web pages, surveys, feeds, and metadata. The analysis of such data helps to gain interesting insights. Multimedia data in the form of text can be structured or unstructured. The structured kind of data can be analyzed with the help of traditional relational database techniques of query retrieval. However, the multimedia data in the form of feeds are unstructured and needs to be transformed into a structured format for further analysis [6]. One of the example applications of multimedia text analytics is based on a particular situation like election, natural disaster, stock market, etc. The emotions behind the text can be analyzed with the help of multimedia data analysis. In this way, different kinds of information can be extracted from different sources of multimedia textual data.

- **Sensor data**: IoT is playing a significant role nowadays and sensors are present almost everywhere. The sensors are equipped with not only capturing the data but also apply analytics in real time [7]. With the advances in the hardware and the technologies of cloud computing, sensor data are increasing enormously. It is highly challenging to analyze such data and develop an analytical application based on that. Sensor data applications are highly seen in astronomical sciences for meteorological patterns, satellite conditioning, and monitoring. In healthcare, most of the applications are based on sensor data. Therefore, with the advances in the sensor data, it is highly essential to develop analytical applications based on it.
- **Social networks**: The main source for multimedia data is social networks. The advances in social networking and sharing that started from a normal text have now reached to image, video, live video, public groups, etc. Recommendation applications [7] widely use the multimedia content available in the social networks to provide recommendations by analyzing the shared messages, video, audio, and text. Personalization services are more widely used by the users of smartphones based on the subscriptions made by them.

The main characteristic nature of the multimedia data is variety. It exists in different forms and at various sources. These advances in multimedia data and analytics have enabled various challenges and technologies for developing different applications. Though, there are various technologies that exist for multimedia analytics the challenges that are put forth for analysis are more and have to be addressed carefully. In the next section, the different challenges that exist for multimedia analytics are discussed followed by the enabling technologies for multimedia analytics.

3 Challenges in Multimedia Data

Multimedia data involves the data from various sources such as cameras, social network, sensors and other that heterogeneous in nature. In order to carry out analytics for such data, the heterogeneity nature of such data has to be transformed for analysis purposes. The transformation of the data involves converting the data from different formats into a singular format for analytics. Some of the challenges that are generally seen with multimedia analytics are discussed below.

3.1 Data Acquisition and Volume

Multimedia data takes a large amount of storage for analytics and involves the acquisition of data from various sources. There are different heterogeneous sources that are involved such as social media, sensors, mobile devices, cameras, and virtual worlds [4]. The heterogeneous data that are generated from these sources are mul-

tistructural and multimodal in nature. Each of the source exhibit different kind of characteristics and highly complex in acquisition both in terms of quantity and quality. New techniques and technologies are needed to understand the varying nature of the multimedia data for acquisition [4, 6].

With the unprecedented growth of multimedia data, there is a huge challenge of storage and making it available for multimedia applications. There are many investigations done for addressing this challenge in multimedia data. In [7], a big data processing pipeline is introduced along with MapReduce that consists of stages such as data preprocessing, data recognition and load reduction. NoSQL databases play a significant role in the storage of multimedia data that allows flexibility of the schema involved in the storage. The different enabling technologies that can handle this issue are addressed in the next section.

3.2 Feature Extraction

Multimedia data such as audio and video includes numerous features. The process of extracting the features from this data plays a significant role in multimedia analytics. The different features of the video can be color, edge, shape, and texture of the image sequences in the video. The extraction of features is divided into two categories namely local feature extraction and global feature extraction. Global feature extraction techniques involve obtaining the color histogram of different objects that are present in the image sequence of the video. The development of color histograms helps in identifying the edges of the objects to distinguish among them. However, it involves a great amount of time since the data will be huge in volume and need to be compressed. In some of the situations, the color histogram may not be successful. Hence, feature extraction is one of the main challenges faced in Multimedia analytics [3]. The different types of techniques for feature extraction are discussed in the next section.

3.3 Synchronization and Computing

Multimedia analytics deals with both audio and video. The main challenge of multimedia analytics is in the synchronization of audio and video. For example in healthcare systems, 3-D based multimedia data need to be taken care of for analytics [8]. A two-tier based synchronization is required for some of the multimedia systems. The research in computing for analytics on multimedia data involves many big data computing platforms like Hadoop, Cassandra, and NoSQL databases. Pipelines of such platforms are required for computation and analysis.

The computational methods for multimedia analytics are mostly based on machine learning techniques like support vector machines (SVM), clustering. However, two or more techniques are usually combined together for multimedia analytics. GPU

computing is utilized in most of the multimedia analytics since the amount of the data is huge in volume. Parallel computing is carried out for multimedia analytics with the GPU to reduce the time of computation and to deal with the nonstationary multimedia data. However, computational methods that are scalable are still a challenge for multimedia analytics.

3.4 Security

Multimedia analytics involves data that are user-centric and contains some of the sensitive information of the users. The analysis performed on such data should have necessary techniques to not utilize the user identity information but carry out analysis [3]. For example, in the sentimental analysis of social network data, the user identity should not be revealed but the textual content can be utilized. Here, the textual content needs to be used rather than the identity of the users. However, geospatial analytics might reveal the geographical information about the users. In such cases, analytical techniques should be careful in revealing the information based on the analysis.

4 Enabling Technologies

Multimedia analytics is analyzing multimedia data such as audio, video, and text to identify the patterns in data and take decisions based on the patterns found. There are various stages of multimedia analytics like data preprocessing, feature extraction and analysis. In each stage, different types of technologies are used for leveraging the analytics and decision-making process. This section focuses on the different enabling technologies and platforms for multimedia analytics. In this regards, multimedia data is categorized into three parts namely text analytics, video analytics, and audio analytics. The enabling technologies in each of the category are summarized as shown in Table 1.

Table 1 Enabling technologies for multimedia analytics

Enabling technology	Language	Open source
NLTK	Python	Yes
Hadoop and Spark	Java, Python, Scala	Yes
Tableau	Java, Python	Partially
Graph databases	Python	Partially
LibROSA	Python	Yes
OpenCV	Python	Yes

4.1 Text Analytics

The most common form of multimedia data is text. It is highly volatile in nature since the text written by humans is unstructured and differs from person to person. There are various stages of text analytics such as removal of stop words like "is", "was", "this". For example, in the case of sentimental analysis, the key phrases of the text are important for analysis rather than the stop words in the text. Some of the enabling technologies that are available for text analytics are discussed below.

- **NLTK**
  Natural language Toolkit (NLTK) is one of the famous python based tools used for natural language processing. Most of the text analytics stages such as preprocessing, tokenization, stemming, tagging, parsing, and semantic reasoning can be performed using various modules that are available within NLTK. Corpus information is most needed while performing text analytics. For example, in the scenario of sentimental analysis a corpus of information that identifies the positive and negative sentiments are needed. WordNet a corpus is available in NLTK for text analytics. The main advantage of NKTK is python based that helps in easy understanding and analysis [9].

- **Deep Learning**
  The recent hype of Artificial intelligence and robotics has paved the way for different deep learning frameworks. Deep learning platforms such as Pytorch and Tensorflow are widely used for text analytics in multimedia systems. The texts that cannot be captured through the World Wide Web are analyzed using deep learning techniques like CNN and RNN architectures. Some of the examples include detection of actions in video, identifying disasters in the web crawled datasets.

- **Scikit learn**
  It is one of the scientific tool kits available in Python platform. Most of the machine learning techniques are provided in the form of APIs that helps in ease of programming. In text analytics, a bag of words is required to identify the most significant words that are present in the text dataset considered. Scikit learn provides such functions like "bag-of-words", "tf-idf" which are needed for text analytics. The term document frequency (TF) is mainly needed to identify the most occurred terms in the text. In this way, Scikit learn is one of the enabling technologies for text analytics [10].

- **Hadoop and Spark**
  Hadoop is one of the open source platforms that can be used to analyze any format of data. MapReduce programming is used to extract the data in the form of text present in Hadoop file system for analysis. It helps in aggregating the information that is present in the form of text. For example, a file containing the users and the channels subscribed by them can be analyzed with the help of MapReduce to find the top subscribed channels [11]. Spark is one of the open-source platforms that

provides numerous machine learning libraries in the form of Spark MLib [12] for performing different types of analytics.

4.2 Video Analytics

Major chunk of the multimedia data is in the form of video. Visual data analytics is a convoluted process since it involves a number of image sequences [13]. Other than the image sequences, visual data can be in the form of graphs and networks of text. There are some enabling technologies that help in visual analytics that are summarized below.

- **Convolution neural networks (CNN) using deep learning**
  CNN is the widely used technique for video analytics. The applications of CNN to video analytics include object detection, crime detection, etc. In CNN, the image sequence is first divided into various pixels and then brought down to scale for object detection. The image is first divided into various filters that represents a matrix of values of the image [14]. The filters are convolved into a matrix of less value by product and sum of two matrices. For example, a 2D image convolution is as shown below.

2-D Image					**Convolved Feature**		
1×0	1×1	1×0	0×1	0×0	2	3	2
0×1	1×0	1×1	1×0	0×0	2	3	3
0×1	0×1	1×0	1×1	1×0	2	3	3
0×0	0×1	1×1	1×0	0×1			
0×1	1×0	1×1	0×0	0×1			

  Once the convolved feature matrix is obtained, a series of convolution layers are built to obtain the final feature matrix and carry out the object detection.
  In this way, CNN architectures are used in video analytics.

- **Tableau**
  Multimedia data involves social network data that can be in the form of feeds or tweets. Visualization of such data helps in revealing the interesting patterns about the data. Tableau is one of such tools that help in understanding the data by visualizing first the data and then to carry out analysis. Tableau is the leading Data Visualization and Business Intelligence tool that helps in creating interactive visualizations. The visualization tool provides an easy to use interface to create dashboards and help solve complex business problems in a fast and effective way. Tableau also can connect to a vast set of files, databases, data warehouse, etc., to carry out analysis on the data from multiple sources.

- **OpenCV**
  OpenCV is one of the platforms widely used for video analytics. It stands for Open source computer vision library. It provides the necessary libraries for real-time object detection within a video. The applications of OpenCV includes facial recognition system, gesture recognition, motion tracking, augmented reality, mobile robotics [15]. It includes most of the machine learning algorithms like k-means clustering, Bayesian classification, decision tree learning, random forest classifier, and artificial neural networks.

4.3 Audio Analytics

Multimedia data that is in the form of audio needs extensive computing and time for arriving at results. The data present in the audio format need to be analyzed stage by stage since it differs from time to time. Some of the enabling technologies for audio analytics are summarized as below.

- **LibROSA**
  It is one of the libraries that is available in python for analysis of various audio files [16]. It helps in the development of various audio applications based on different formats. It helps in extracting the features of the music files like rhythm, tempo, beats, audio time series, power spectrogram of the audio, roll of frequency, spectral flatness, etc.
- **Deep learning**
  Deep learning is used for audio analysis using the artificial neural networks like Recurrent Neural Nets (RNN), Convolutional Neural Networks (CNN), and feed-forward networks. Recent developments in the computer science has paved the way for different applications in audio processing using deep learning. One of such experiments has been done in [14] to develop a framework for audio event recognition based on the web data. A CNN is developed as a part of the experiment with five hidden layers. Some of the learning methods are explored in the further sections.

5 Deep Learning Methods

In any dataset features/attributes play an important role to develop learning algorithms. In deep learning methods, these features play an important role in developing appropriate learning algorithms. A feature can be defined as the interesting part of the dataset that acts as the starting point for learning. The desirable property of any learning algorithm is the repetitive process of detecting the features for the dataset that is not trained. The important features for an image dataset are temporal, spatial, and textural. There are three stages in any feature detection algorithm namely

extraction, selection, and classification. The output of the extraction phase is a representation vector of the dataset/image considered. The representation vectors can be different for different datasets. It is used for classification purposes using deep learning methods like CNN, RNN, and feed-forward networks [17, 18].

5.1 Local Feature Extraction Methods

A feature is a function that represents the quantified value of an object/dataset. It represents the significant features of the object such as color, texture, and pixels. Local features of a dataset/image refer to the characteristics of the image that adheres to edge detection, image segmentation, and the features that are calculated on the results of the subdivision of the image. The most common methods that are used for local feature extraction using deep learning are as follows.

- **Haris Corner detection**
  Corner detection is used in the computer vision systems to infer some of the interesting facts of the image and its corners. It is usually used for applications like motion detection, video tracking, 3D modeling, and object recognition. One of the methods that are used is Haris corner detection that is based on the intensity of the corners [19]. The basic idea is to observe a large change in appearance when there is a shift in the window in any direction. The normal three scenarios of corner detection are as shown in Fig. 2. The change in the shift of the intensity says [u, v] is calculated as shown in the Eq. 1.
- **Scale Invariant Feature Transformation (SIFT)**
  It is used to detect local features in the image for applications such as gesture recognition, video tracking, 3D modeling, etc. A set of key features in the image are extracted and stored in a database. For a new image, the features are compared

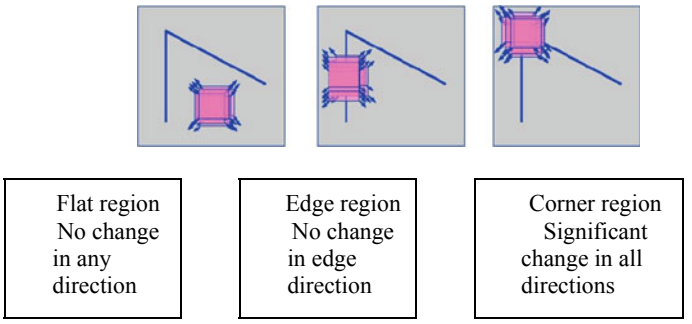

Fig. 2 Haris corner detection

against the existing database to determine the best candidate matching using the Euclidean distance of the feature vectors [20]. In order to filter out the matching from the database, a subset of the keypoints that are matched with location, scale, and orientation are filtered out from the database. The drawback of SIFT is the computation involves calculation of pixel patch value based on the histogram of gradients. Hence, if the image size is large then computation becomes expensive.

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + u) - I(x, y)]^2 \quad (1)$$

- **Oriented FAST and Rotated BRIEF(ORB)**
  ORB is one of the local feature extraction methods that overcomes the drawback of SIFT and reduces the computation time. It is based on the BRIEF (Binary robust independent elementary features) and FAST point detector [21]. The points are detected in the image based on the orientation. Initially, the point is located at the center of the corner of the image. The orientation of the image is known from this point. The invariance is improved using the circular region of the image. The description of the images is set using the points detected using FAST point detector. A matrix S ($2 \times n$) defines the coordinates of the pixels from the feature set. The orientation Θ found using FAST point rotates the matrix S to a steered version S_{Θ}. A lookup table is first precomputed using the angle $2\Pi/30°$ to produce the correct set of points S_{Θ}. A small snippet of the code is as shown below.
  The image is read initially inputted in grayscale but can be converted to RGB later. The ORB local feature extractor is defined and to specify the number of features that are needed to pick up. Initially, it's set to the top 1000 features, if left blank if finds all the possible features. All the points of interest in the image are computed and stored in the variable "kp". A plot is defined for these feature points stored in "kp" on the original image, these points are marked in a single color as specified in the parameters list. In the end, output can be viewed using the imshow() function in matplotlib. The output of a sample image is as shown in Fig. 3.

Input Image ORB features

Fig. 3 ORB sample output

```
import cv2
import matplotlib.pyplot as plt
filename='5.jpg' #filename of the image
img = cv2.imread('5.jpg',0)
orb = cv2.ORB(nfeatures=1000) #defines how many features
to pick
kp = orb.detect(img,None)
kp, des = orb.compute(img, kp)
img2 = cv2.drawKeypoints(img,kp,color=(255,0,0), flags=0)
#plot all the points of interest on the image
plt.figure(figsize=(16, 16))
plt.title('ORB Interest Points')
plt.imshow(img2)  #print image with highlights
plt.show()
```

5.2 *Global Feature Extraction Methods*

Once the local features such as edges of the images/dataset are extracted from the image global features are computed for the entire image. These global features help in the final classification of the image using the computed elements of the local features. The common methods that are used in extracting global features of the dataset/image are as follows:

- **Histogram of Oriented Gradients (HOG)**
  HOG is used as feature descriptor in image processing and computer vision for object detection. It is based on the histogram of gradients that acts as a global feature for object detection. In local portions of the image, it counts the occurrences of gradient orientation to produce the histogram. The intensity of the distribution of histograms present in the local object appearances guides the production of the histograms [22]. Basically, in an image, each pixel is drawn into a group of cells. A histogram of gradients is compiled for each pixel in the cell. A descriptor forms the concatenation of all these histograms. The different types of gradients that are followed in HOG are listed as follows.
  - **Image gradients**
    A gradient is a vector quantity that has both magnitude and direction. An image gradient gives the intensity change at the particular location of the image. An illustrative image gradient is as shown in Fig. 4.
  - **Histogram of oriented gradients**
    When the gradients of the image are oriented, it is called as oriented gradients. A histogram of oriented gradients gives the intensity of the gradients of the

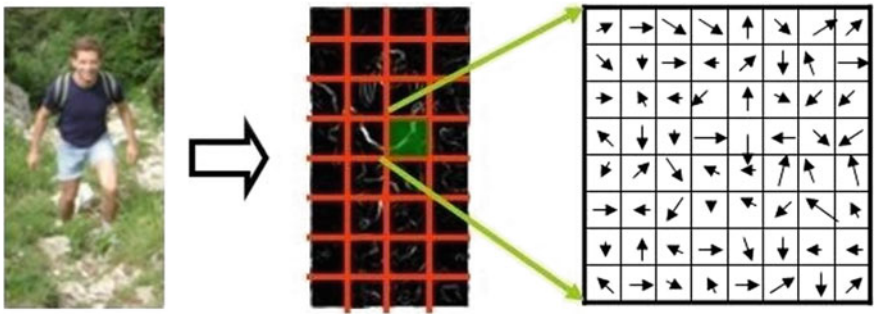

Fig. 4 Image gradient

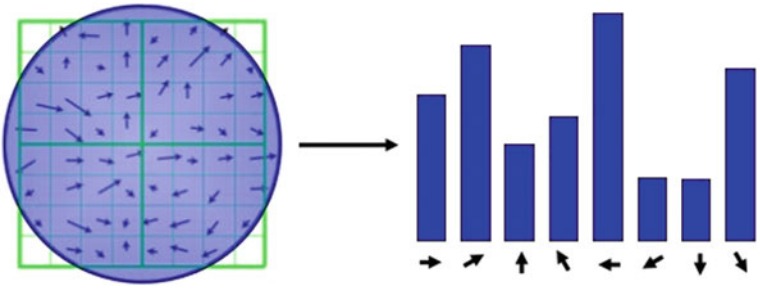

Fig. 5 Histogram of oriented gradients

small portion of the image. It helps to find the probability of a gradient in the particular portion of the image as shown in Fig. 5.

A small snippet of Histogram of oriented gradients is as shown below. The required libraries are imported, including the HOG detector function from the skimage library. Here, img is the image being read, orientation is the number of directions for features in terms of intensity fluctuation, pixels_per_cell divides picture into cells and define the number of pixels per cell, cells_per_block is the number of cells taken per block for mapping, visualize is to get final image of feature detection, feature_vector is to output the final feature vector, "feat" is the feature set and "im" is the histogram. The output of a sample image is as shown in Fig. 6a–c.

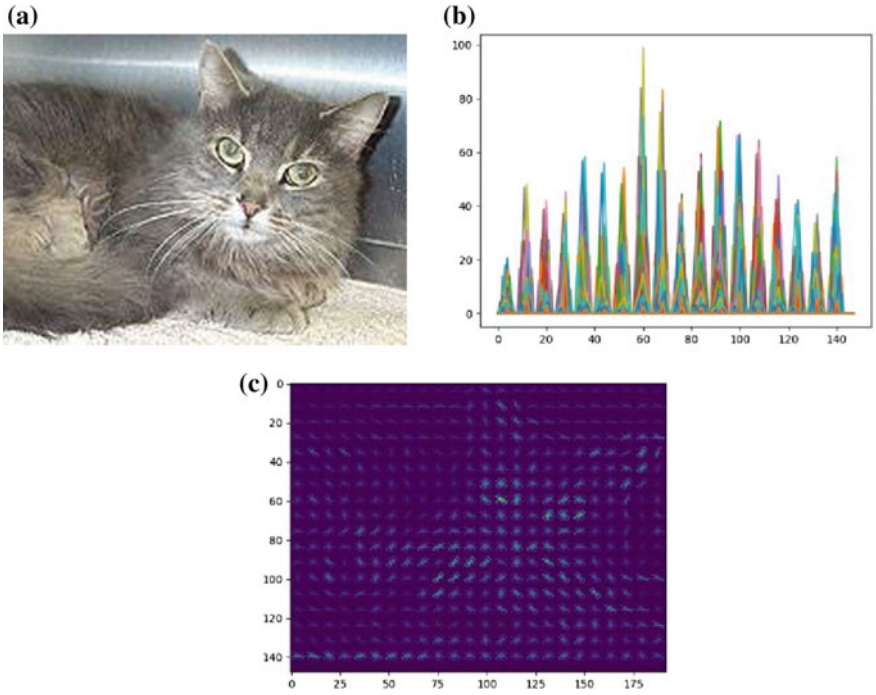

Fig. 6 **a** Input for HOG **b** Histogram for HOG sample output **c** Features for HOG sample output

```
import cv2
import matplotlib.pyplot as plt
from skimage.feature import hog
img = cv2.imread("7.jpg",0) #read image
feat,im=hog(img,orientations=9,pixels_per_cell=(8,8),cell
s_per_block=(2,2), visualise=True,feature_vector=True)
#img is image being read
#orientation the number of directions for features in
terms of intensity fluctuation
#pixels_per_cell divide picture into cells and define the
pixel density
#cells_per_block number of cells taken per block for
mapping
#visualise get image of feature detection
#feature_vector
plt.plot(im) #plot histogram for detection
plt.show() #show histpgram graph
```

6 Deep Learning for IoT Data Classification

6.1 IoT Data and Its Types

Internet of things is composed of various devices capable of receiving data and may or may not possess computing power. These devices are interconnected and are usually in constant communication with each other. Thus as a natural consequence always keep generating data. The devices participating in an IoT network can be majorly classified into two types. The first of them which forms the building blocks of the network are the sensors and second one of them are the devices which consume the data acquired from the sensors. These devices that consume the data can be further distinguished as ones which perform analysis based on the retrieved data or initiate an action based on the acquired data. On the whole, IoT data that is being generated by sensors play a major role in powering various kinds of applications.

From the natural understanding of the behavior of devices that participate in IoT, a norm can be derived for classifying IoT data. Thus this basic classification of IoT data yields two categories namely

1. Data used for analysis purposes (Analysis data)
2. Data used to initiate an action (Actuation data or action causing data)

6.1.1 Analysis Data

The volume of data generated from a wide array of sensors provides an excellent platform to perform analysis. Various kinds of sensor data are available ranging right from trivial data obtained from simple motion sensors to a more sophisticated form of data such as location data obtained from smartphones. The data obtained will usually be in the form of time series data which has a natural consequence of having a higher volume.

A classical example that can be illustrated in the case of analysis of IoT data is that of study of meteorological data. It consists of study of atmosphere and its constituents and various related analysis such as weather patterns, rainfall magnitudes, and so on. Various predictions are also undertaken based on this data which is clearly evident in the form of weather updates, rainfall predictions, and temperature range suggestions. Another similar example will be the analysis of chemical constituents of air and analyzing the overall air quality.

Various other domains also provide good scope for data analysis which includes multimedia sources. This vast domain has a huge disposal of data which exists min various forms. This includes real-time traffic data, text, and images from various social media platforms. Another domain which provides a platform to obtain statistics are the various routers, switches, and other networking devices deployed across the globe. Modern SDN systems have device statistics built into them to enable analysis and to infer congestion levels and other such parameters. A variety of medical

equipments also participate in generating data such as electrocardiography, blood test reports, x-rays, etc., which can be used for analysis purpose when viewed at a large scale. Call records of mobile phones and the usage times of different applications can also be considered as a contributor. The insights obtained after performing analysis are usually used for commercial purposes or can be utilized by the government.

6.1.2 Actuation Data or Action Causing Data

Apart from aiding various analytics applications, IoT data is also being used to actuate other devices in order to obtain a desirable action or a mechanism which drives a decision. This form of utility is being exploited in areas such as automation systems and in networking scenarios, as a solution mechanism for congestion problems.

As an example, in the case of automation systems, a simple home automation system serves the purpose of understanding decision driving mechanisms. The heating and illumination systems present in a typical house such as normal light bulb, or a fan can be made alternate between on and off state based on the data collected by sensors deployed in the house which include motion sensors, thermostats, and related sensors.

As an example in the case of networking scenarios, the routing information stored in the edge devices and the records of data or packets passing through the device can be retrieved. This information can be used to enable rerouting bringing about load balancing and thereby enabling congestion control.

6.2 Case Study on IoT Data Classification Using Deep Learning

6.2.1 IoT Dataset

The dataset chosen for classification purpose is “Air Quality Dataset” [23]. It is a fairly sized dataset of 9358 instances. The quality of air is determined by the relative composition of its constituent elements. The data collected here can be categorized as IoT data since it has accumulated by the various sensors. The device used to form the dataset is an Air Quality Chemical Multisensor Device. It is composed of five metal oxide chemical sensors which are arranged to form the device.

Data was gathered on an hourly basis in order to obtain a sizeable dataset and which represents adequate variation. The sensor was placed in an Italian city which was subject to a relatively higher level of pollution due to the presence of various pollutants. The duration of data capture ranged from March 2004 to February 2005. This data represents one whole year of recording which is regarded as the longest duration in which chemical sensors were deployed which captured air quality.

The dataset consists of the following attributes:

1. Date (DD/MM/YYYY)
2. Time (HH.MM.SS)
3. True hourly averaged concentration CO in mg/mˆ3 (reference analyzer)
4. PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)
5. True hourly averaged overall Non-Metanic HydroCarbons concentration in microg/mˆ3 (reference analyzer)
6. True hourly averaged Benzene concentration in microg/mˆ3 (reference analyzer)
7. PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)
8. True hourly averaged NOx concentration in ppb (reference analyzer)
9. PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)
10. True hourly averaged NO_2 concentration in microg/mˆ3 (reference analyzer)
11. PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO_2 targeted)
12. PT08.S5 (indium oxide) hourly averaged sensor response (nominally O_3 targeted)
13. Temperature in Â°C
14. Relative Humidity (%)
15. AH Absolute Humidity

IoT data most of the times is captured with the help of various sensors. These sensors are usually physical which are deployed in sites. They can also be simulated via a python script for experimental or analysis purposes. Theses scripts which are deployed in edge devices act as an interface to simulate different kinds of sensors. The script shown below can be used to pump in sensor data.

The sensors deployed in edge devices need to pump in data to a central database. IoT devices communicate using a lightweight protocol called mqtt protocol. Two basic operations or functions associated with mqtt protocol are published and subscribed. These terms are analogous to write and read of an I/O device. A device or a sensor that needs to pump in data has to publish with a particular type under a predetermined topic. Similarly, devices that need to analyze or read this data need to subscribe to the topic. The mqtt broker plays a vital role or is essentially the heart of the protocol. It is responsible for sending and receiving messages and delivering it to the clients thereby responsible for the fundamental publish/subscribe model. In the script shown below data needs to published and hence associated code has been incorporated.

```
import time
import paho.mqtt.client as mqtt
import json import ssl
import random
broker = "<broker_url here>"
port = 8883
topic = "iot-2/evt/test/fmt/json"
username = "<username>"
password = "<password >"
organization = "<org id here>"
deviceType = "Python_Client"

mqttc = mqtt.Client("<client id>") print(username+'
'+password)
mqttc.username_pw_set(username,password)
mqttc.tls_set_context(context=None)
mqttc.tls_insecure_set(False)
mqttc.connect(broker,8883,60)
 for i inrange(0,10):
        print('Sending sample key value pairs')
        msg={'key':i,'value':random.randrange(1,50)}
        mqttc.publish(topic,json.dumps(msg))
        time.sleep(1)
```

The supporting libraries for implementing the mqtt protocol and establishing a secure channel for communication have been imported. A suitable port number and a desired device type has been chosen. Among the various mqtt brokers available, an apt broker according to the application is to be chosen. The above script simulates a sample sensor and hence accordingly a generic key-value pair is taken as the reading of the sensor and a sample topic name is chosen. The key-value pairs usually denote the (time period, sensor value) of a typical sensor. This generic sensor can be considered as a temperature gauge, or an accelerometer, in simple terms a gravity sensor which is present on most modern day smartphones. These virtual sensors can be used for sample analytics purposes and to suit real life requirements, they are replaced by actual sensors that are deployed in the region of study.

6.2.2 Building Neural Network for Classification

Many problems like Email spam, Handwriting digit recognition, Fraud transaction detection are important day to day example of Classification problems. Classification in simple terms, is grouping of things by a common features, characteristics, and qualities. To solve this problem there are many classification algorithms like decision tree, logistic regression, and k-nearest neighbors. However, Neural Network has few benefits when compared to others therefore, it is the most popular choice to handle real-time big data. The benefits of neural network are:

– Normal Machine learning algorithm will reach a plateau, and will not improve much with more data after a certain point. However, neural network improves with more data to train
– Neural Network is a nonlinear model, which helps us to define complex hyper plane, nontrivial decision boundaries for solving convoluted classification problem.
– Neural network is highly adaptive can deal with nonstationary environments.
– Neural network has high Fault tolerance. Corruption of few neurons does not restrict it from generating the output.
– Every neuron has a potential to influence every other neuron in the network. Therefore, Contextual information is dealt naturally.

Due to all the reasons above, artificial neural network is being actively employed in lot of real-world problems like classifying spam emails, Bankruptcy prediction, flagging fraud transaction, etc., and are getting better results than conventional algorithms.

6.2.3 Experiment and Results

A sample dataset is as shown in Table 2. Artificial neural network is a network of neurons. They process the training data one at time and learn by comparing their classification predicted with the actual classification. The errors from initial classification are fed back into the network to correct itself and modify weights for further iteration. Therefore, the major difficulty in building neural network is to find the most appropriate grouping of training, learning, and transfer functions to get the best result.

Before we can build a network, we have process the dataset by classifying all NO2(GT) columns to two classes. 1 for higher and 0 for lesser than 100 air quality index as shown in Table 3.

Table 2 Air quality data set

Date	Time	CO(GT)	PT08.S1 (CO)	NMHC(GT)	C6H6(GT)	PT08.S2 (NMHC)
10-03-2004	18.00.00	2.6	1360	150	11.9	1046
10-03-2004	19.00.00	2	1292	112	9.4	955
10-03-2004	20.00.00	2.2	1402	88	9	939
10-03-2004	21.00.00	2.2	1376	80	9.2	948
10-03-2004	22.00.00	1.6	1272	51	6.5	836

Table 3 Preprocessing of data

Date	Time	NO2(GT)	T
10-03-2004	18.00.00	1	13.6
10-03-2004	19.00.00	0	13.3
10-03-2004	20.00.00	1	11.9
10-03-2004	21.00.00	1	11.0
10-03-2004	22.00.00	1	11.2

```
data=df[['Date','Time','NO2(GT)', 'T']].copy()
for i in range(len(data)):


    if(data.iloc[i]['NO2(GT)']<100):


      data.at[i,'NO2(GT)']=0 else:
      data.at[i,'NO2(GT)']=1
```

Next, we need input features like weekday and hour to extract from date and time column as shown in Table 4. We can split the dataset into training set and testing set.

```
for i in range(len(data)):
  time=datetime.datetime.strptime(data.iloc[i]['Date'
   ]+" "+ data.iloc[i]['Time'], "%d/%m/%Y %H:%M:%S")
  weekday=time.weekday()
  hour=time.hour
  data.at[i,'Weekday']=weekday
  data.at[i,'hour']=hour
  data.Weekday = data.Weekday.astype(int)
  data.hour = data.hour.astype(int)
  train, test = train_test_split(data, test_size=0.2)
  train.to_csv("train.csv",mode="w",index=False)
  test.to_csv("test.csv",mode="w",index=False)
```

Table 4 Preprocessing of data with day and hour

Date	Time	NO2(GT)	T	Weekday	Hour
10-03-2004	18.00.00	1	13.6	2	18
10-03-2004	19.00.00	0	13.3	2	19
10-03-2004	20.00.00	1	11.9	2	20
10-03-2004	21.00.00	1	11.0	2	21
10-03-2004	22.00.00	1	11.2	2	22

We will use tensor flow high level API to build a simple neural network with an input layer (4 nodes) and a output layer. To that, we have to first write a function to extract the training dataset.

```
def load_traindata(label_name='NO2(GT)'):
   train_path="train.csv" # For Training NN model
   CSV_COLUMN_NAMES=
   ['Date','Time','NO2(GT)','T','Weekday','hour']
   train = pd.read_csv(filepath_or_buffer=train_path,
   names=CSV_COLUMN_NAMES, # list of column names
   header=0, # ignore the first row of the CSV file.
      skipinitialspace=True,
      #skiprows=1
      )
   train.pop('Time')
   train.pop('Date')
   train['hour'] = train['hour'].astype(str)
   train.Weekday= train.Weekday.astype(str)
   train['T']= train['T'].astype(float)
   train_features,train_label=train,train.pop(label_n
   ame)
   return (train_features,train_label)

#getting training features and labels
(train_feature,train_label)= load_traindata()
```

To give a input features to we have to give it in the form of tf.feature column. This feature consist can be categorical (Weekday, hour) or numerical (Temperature T) also. If two or more input features are closely related, it can be a separate feature in itself. This type of columns is known as crossed column (weekday x_hour), which is mixture of two or more input features. After deciding the input features we create a Simple DNN classifier with base columns and crossed columns. Hidden unit indicates the no of nodes in each layer (for example [3–5] indicates a neural network with input layer with 4 nodes, hidden layer with 3 nodes and hidden layer 2 with 2 nodes). We need a hidden layer if the data is not linearly separable. Number of input nodes depends on number of input features and, similarly, number of output nodes depends on number of classes being in output label.

```
#creating normal features and crossed features for the
nn model
Weekday =
tf.feature_column.categorical_column_with_vocabulary_li
st ('Weekday', ['0', '1', '2', '3', '4','5','6'])
hour    =
tf.feature_column.categorical_column_with_vocabulary_li
st('hour', [ '0','1','2', '3','4','5','6','7','8','10',
'12','13','14','15','16','17','18','19','20','21','22',
'23'])
T=
tf.feature_column.numeric_column(key='T',dtype=tf.float
64)
base_columns = [
    tf.feature_column.indicator_column(Weekday),
    tf.feature_column.indicator_column(hour),
    T
    ]
Weekday_x_hour = tf.feature_column.crossed_column(
  ['Weekday', 'hour'], hash_bucket_size=1000)
crossed_columns = [
  tf.feature_column.indicator_column(Weekday_x_hour)
    ]
# Running Dnn classifer model with the features
designed
# Above and with and input layer with 4 nodes

classifier =
tf.estimator.DNNClassifier(feature_columns=
base_columns+crossed_columns,hidden_units=[4],n_
classes=2)
```

We have to train the classifier by passing a train_input function which shuffles the input training data features. We have to iterate through the process (epoch) until the classifier reaches its minimum error rate. Here we have chosen to iterate 50 times.

```
#training the nnmodel
def train_input_fn(features, labels, batch_size):
    dataset =
    tf.data.Dataset.from_tensor_slices((dict(features) ,
    labels))
    dataset=
    dataset.shuffle(buffer_size=1000).repeat(count=None
    ).batch(batch_size)
   return
   dataset.make_one_shot_iterator().get_next()
```

```
  classifier.train(
    input_fn=lambda:train_input_fn(train_feature,
    train_label, 50 ),steps=1000)
```

Neural Network Training

```
  INFO:tensorflow:Callingmodel_fn.
  INFO:tensorflow:Done calling model_fn.
  INFO:tensorflow:CreateCheckpointSaverHook.
  INFO:tensorflow:Graph was finalized.
  INFO:tensorflow:Runninglocal_init_op.
  INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1 into
C:\Users\Guest\AppData\Local\Temp\tmprvwl47wc\model.ckpt.
  INFO:tensorflow:step = 1, loss = 42.49231
  INFO:tensorflow:global_step/sec: 48.6003
  INFO:tensorflow:step = 101, loss = 30.568665 (2.062
sec)
  INFO:tensorflow:global_step/sec: 55.7494
  INFO:tensorflow:step = 201, loss = 25.75341 (1.793 sec)
  INFO:tensorflow:global_step/sec: 57.967
  INFO:tensorflow:step = 301, loss = 24.957882 (1.726
sec) INFO:tensorflow:global_step/sec: 57.7436
  INFO:tensorflow:step = 401, loss = 29.967522 (1.732
sec) INFO:tensorflow:global_step/sec: 58.4679
  INFO:tensorflow:step = 501, loss = 27.571487 (1.711
sec)
  INFO:tensorflow:global_step/sec: 53.1789
  INFO:tensorflow:step = 601, loss = 25.81527 (1.875 sec)
  INFO:tensorflow:global_step/sec: 54.6941
  INFO:tensorflow:step = 701, loss = 19.551216 (1.833
sec) INFO:tensorflow:global_step/sec: 56.3506
  INFO:tensorflow:step = 801, loss = 27.794727 (1.776
sec) INFO:tensorflow:global_step/sec: 59.6893
  INFO:tensorflow:step = 901, loss = 21.918167 (1.673
sec)
INFO:tensorflow:Saving checkpoints for 1000 into
    C:\Users\Guest\AppData
    \Local\Temp\tmprvwl47wc\model.ckpt.
  INFO:tensorflow:Loss for final step: 24.991734.
```

After training the classifier we can test it by passing it test data input features, in a similar way as we train. We have to pass a function which evaluate_input function to pass only the input feature to classifier and classifier predicts the output.

```
  #predicting for all time intevals in a day with the
trained                                                model
test_df=pd.read_csv("test.csv",parse_dates=True)
predict_x= {
      'T':[],
      'Weekday':[],
      'hour':[ ],
      }
predict_x['T'] = test_df['T'].astype(float)
predict_x['hour']= test_df.hour.astype(str)
predict_x['Weekday']= test_df.Weekday.astype(str)
test_label=test_df['NO2(GT)']

def eval_input_fn(features, labels=None,batch_size=None):
   """An input function for evaluation or prediction"""
   if labels is None:
         # No labels, use only features.
         inputs = features
   else:
         inputs = (features, labels)
         # Convert inputs to a tf.dataset
         object.
         dataset=tf.data.Dataset.from_tensor_
         slices (inputs)
         # Batch the examples
         assert batch_size is not None, "batch_size must
         not be None"
         dataset = dataset.batch(batch_size)
         # Return the read end of the pipeline.
         return
         dataset.make_one_shot_iterator().get_next()

predictions = classifier.predict(
input_fn=lambda:eval_input_fn(predict_x,
      labels=None, batch_size=50))

pred_label=[]
for pred_dict in zip(predictions):
   if pred_dict[0]['classes'] == [b'1']:
         pred_label.append(1)
   else:
        pred_label.append(0)
```

Since the problem chosen is a binary classification we can check the results by finding the confusion matrix for the test data. Confusion matrix is table which contains true positives, false positive, true negative, and false negative as shown in Table 5.

Table 5 Confusion matrix

	Predicted positive	Predicted negative
Actual positive	692	303
Actual negative	169	708

We get the sensitivity and specificity with which classifier classifies the test data. We can also represent the confusion matrix in the form of a graph as shown in Fig. 7.

```
#ploting a confusion matrix with the tested data def
plot_confusion_matrix(cm,     names,     title='Confusion
matrix', cmap=plt.cm.Blues):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(names))
plt.xticks(tick_marks, names, rotation=45)
plt.yticks(tick_marks, names)
plt.tight_layout()plt.ylabel('True label')
plt.xlabel('Predicted label')
con_mat =tf.confusion_matrix(test_label,pred_label)
cm=[]
with tf.Session():
  cm=tf.Tensor.eval(con_mat,feed_dict=None,
   session=None)
```

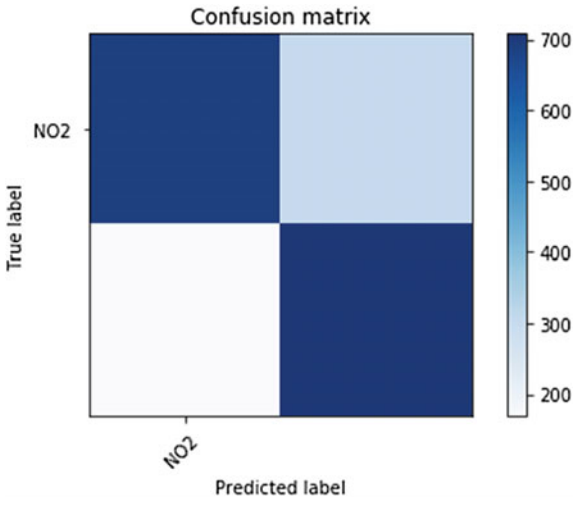

Fig. 7 Visual representation of confusion matrix

```
plt.figure()


diagnosis=["NO2 "]
 plot_confusion_matrix(cm, diagnosis)
```

We can also test our results of our classifier using a ROC (Receiver Operating Characteristic) curve. ROC curve is a graphical way to show the cutoff between sensitivity (fraction of true positives) and specificity (fraction of true negatives). The area under Roc curve is a measure of usefulness of test, therefore greater AOC means better result as shown in Fig. 8.

```
# Plot an ROC. pred - the predictions, y - the expected
output. def plot_roc(pred,y):
fpr, tpr, _ = roc_curve(y, pred)
roc_auc = auc(fpr, tpr)
print("Auc of classifer is ")
print(roc_auc)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)'
        % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
plot_roc(pred_label,test_label)
```

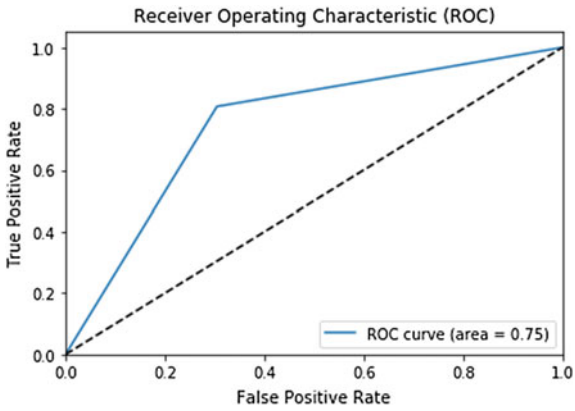

Fig. 8 Visual representation of ROC curve

7 Conclusion

Since IoT devices are increasing day by day multimedia data from different sensors such as video, audio, phone, and others need to be processed in real time. If the data generated is not collected and analyzed then the value of the data may not be known. Thus, deep learning methods are very essential for analysis of multimedia data and IoT for different applications. In this chapter, a brief introduction to multimedia data and its types were initially mentioned. The different deep learning methods with local feature extraction and global feature extraction were discussed with small snippets of examples. Finally, a case study on air quality monitoring presented the deep learning method of classification and analysis of the data.

References

1. A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, K.-K.R. Choo, Multimedia big data computing and Internet of Things applications: a taxonomy and process model. J. Netw. Comput Appl. (2018)
2. P.K. Atrey, M. Anwar Hossain, A. El Saddik, M.S. Kankanhalli, Multimodal fusion for multimedia analysis: a survey. Multimed. Syst. **16**(6), 345–379 (2010)
3. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
4. C.A. Bhatt, M.S. Kankanhalli, Multimedia data mining: state of the art and challenges. Multimed. Tools Appl. **51**(1), 35–76 (2011)
5. F. Venter, A. Stein, Images & videos: really big data. Anal. Mag. 14–47 (2012)
6. D. Che, M. Safran, Z. Peng, From big data to big data mining: challenges, issues, and opportunities, in *Database Systems for Advanced Applications* (Springer, Wuhan, China, 2013), pp. 1–15
7. Z. Wu, M. Zou, An incremental community detection method for social tagging systems using locality-sensitive hashing. Neural Netw. **58**(1), 12–28 (2014)
8. P.K. Atrey, N.C. Maddage, M.S. Kankanhalli, Audio based event detection for multimedia surveillance, in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5 (IEEE, 2006)
9. NLTK, https://www.nltk.org/
10. Scikit learn, http://scikit-learn.org/
11. Hadoop, https://hadoop.apache.org/
12. Spark, https://spark.apache.org
13. J. Herrera, G. Molto, Detecting events in streaming multimedia with big data techniques, in *2016 Proceedings of 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, Heraklion Crete, Greece (2016), pp. 345–349
14. S. Hershey, S. Chaudhuri, D.P. Ellis, J.F. Gemmeke, A. Jansen, R.C. Moore, M. Plakal, D. Platt, R.A. Saurous, B. Seybold et al., CNN architectures for large-scale audio classification, in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2017), pp. 131–135, https://www.tableau.com/ (14. Tableau)
15. OpenCV, https://opencv.org/
16. Librosa, https://librosa.github.io/librosa/
17. K. Alex, I. Sutskever, E.H. Geoffrey, *ImageNet Classification with Deep Convolutional Neural Networks* (2012), pp. 1097–1105
18. J. Schmidhuber, Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)

19. C. Harris, M. Stephens, A combined corner and edge detector, in *Alvey Vision Conference*, vol. 15, no. 50 (1988), pp. 10–5244
20. T. Lindeberg, Scale invariant feature transform (2012)
21. E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, in *2011 IEEE International Conference on Computer Vision (ICCV)* (IEEE, 2011), pp. 2564–2571
22. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, CVPR 2005*, vol. 1 (IEEE, 2005), pp. 886–893
23. Air quality data, https://archive.ics.uci.edu/ml/datasets/Air+quality