

Lung Disease Pattern using ANN-BPN

A PROJECT REPORT

Submitted by

C.NIKHIL REDDY [RA1711003011013]

G.MUNI REDDY [RA1711003011033]

Under the Guidance of

DR. P. VELMURUGAN

(Associate Professor, Department of Computer Science and Engineering)

In Partial Fulfilment of the Requirements for the Degree of



BACHELOR OF TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**FACULTY OF ENGINEERING AND TECHNOLOGY SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603 203**

MAY 2021



**SRM INSTITUTE OF SCIENCE AND
TECHNOLOGY KATTANKULATHUR-603203**

BONAFIDE CERTIFICATE

Certified that this project report titled "LUNG DISEASE PATTERN BASED ON ANN-BPN" is the bonafide work of Mr CHIRRA NIKHIL REDDY and Mr G.MUNI REDDY who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

<<Signature of Guide>>

<<Signature of HOD>>

Dr P. VELMURUGAN

Associate Professor

Department of Computer Science
and Engineering
SRMIST
603 203

Dr B. AMUTHA

Professor and Head

Department of Computer Science
and Engineering
SRMIST
603 203

Signature of Internal Examiner

Signature of External Examiner



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Department of Computer Science and Engineering

SRM Institute of Science & Technology

Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/Course : B. Tech

Student Name : C. NIKHIL REDDY, G. MUNI REDDY

Registration Number: RA1711003011013, RA1711003011033

Title of Work : LUNG DISEASE PATTERN BASED ON ANN-BPN

I / We hereby certify that this assessment complies with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated and that I / we have met the following conditions:

- References / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data, etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged inappropriate places any help that I have received from others (e.g.: fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University's policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except, where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **C. Muthamizhchelvan**, Vice Chancellor (I/C), SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for his invaluable support. We wish to thank **Dr B. Amutha**, Professor and Head, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Academic Advisor **Dr. P. Vishalakshi**, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for their great support at all the stages of project work. We would like to convey our thanks to our Panel Head, **Dr V.V Ramalingam, Professor**, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for his / her inputs during the project reviews.

We register our immeasurable thanks to our Faculty Advisor, **Dr A.Jackulin Mahariba**, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr P.Velmurugan**, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under his/her mentorship. He / She provided me the freedom and support to explore the research topics of my interest. Her / His passion for solving real problems and making a difference in the world has always been inspiring.

We sincerely thank the staff and students of the Computer Science and Engineering Department, SRM Institute of Science and Technology, for their help during my research. Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement.

C. Nikhil Reddy (RA1711003011013)

G. Muni Reddy (RA1711003011033)

Lung Disease Pattern Based on ANN-BPN

ABSTRACT

Lung diseases are a common occurrence in people with a variety of medical conditions, and they are often caused by smoking, drinking, or being exposed to toxic gases. The use of image processing to detect lung cancer is an effective diagnostic method. Researchers were able to detect a cancerous cell in a lung using image pre-processing methods such as the Gabor filter and image segmentation. Using Python and Tensor flow to remove features is a good way to go. GLCM attribute extraction and several segmentation techniques were used on various CT images of the lung to define the lesion portion of the lung, and the performance was in JPG format. The technique's performance was evaluated as a result. According to the findings, image segmentation combined with CNN Classifier and Fuzzy Clustering yielded improved results for lung cancer cell detection than other segmentation strategies.

Image Segmentation, Artificial Neural Network, Machine Learning, and Fuzzy Clustering are some of the terminology used in this article.

TABLE OF CONTENTS

TITLE PAGE	1
BONAFIDE CERTIFICATE	2
OWN WORK DECLARATION	3
ACKNOWLEDGEMENT	4
ABSTARCT	5
1. INTRODUCTION	8
1.1 CLUSTERING	8
1.2 K-MEANS CLUSTERING	9
1.3 EXISTING METHOD	11
2. SYSTEM ANALYSIS	11
2.1 BACKGROUND	11
3. LITERATURESURVEY	13
4. SYSTEM REQUIRMENTS	15
4.1 HARDWARE REQUIRMENTS	15
4.2SOFTWARE REQUIRMENTS	15
4.4 PROPOSED SYTEM	15
4.5 BLOCK DIAGRAM	15
4.6 DESCRIPTIONS	17
5. METHODOLOGY	17
5.1 DIGITAL IMAGE PROCESSING	17
5.2 CLASSIFICATION OF IMAGES	20
5.3 PRE PROCESSING	23
5.4 CLUSTERING MODEL	25

5.5 ARTIFICIAL NEURAL NETWORK	33
5.6 BACK PROPAGATION ALGORITHM	39
6. CONCLUSION	41
7. APPENDICES	42
7.1 PROJECT CODE	42
7.2 SCREENSHOTS	55
8. PLAGARISM REPORT	57

CHAPTER 1

INTRODUCTION

1.1 CLUSTERING

Clustering is perhaps the most significant unstructured training task [12], and it involves identifying a pattern in a set of unlabeled information. As a result, a clustering is a group of things that are "related" to one another but "different" to items from other clustering.

The following are the different types of learning techniques.

- Probability - based Clustering
- Layered Networking

In the first scenario, information are categorized in an exclusionary fashion, meaning that if a data corresponds to one group, it can be incorporated in someone else. The second kind, intersecting grouping, on the other hand, groups data using linguistic variables, allowing each point to correspond to different groups with varying class labels. In this situation, information will be linked to a participation function that is suitable. The merger of the two closest groupings is the foundation of a grouping technique. Organizing each datum as clusters achieve the starting requirement..

Thresholding

The adaptive threshold approach is the most basic picture classification methodology. To convert a grey scale picture to a binary picture this approach uses a clip-phase (or a minimum number). The most important part of this strategy is deciding on a predefined threshold. The greatest entropy approach, Otsu's approach (highest variation), and k-

meaning clustering are all common approaches in business. Techniques for filtering computerized tomography (CT) pictures have lately been established. The fundamental concept is that, contrary to Otsu's technique, the boundaries are calculated using radiography rather than the (recreated) picture.

Design Steps:

- (1) Establish a preliminary threshold $T = (\text{maximum picture brightness} + \text{lowest picture brightness})/2$.
- (2) Split the picture applying T to produce 2 pairs of pixel intensities: B (all number of pixels are below T) and N (all intensity values are larger than T).
- (3) Determine the median values of B and N individually, as well as the means u_b and u_n .
- (4) Determine the revised thresholds by using the following formula: $T = (u_b + u_n)/2$
 $T = (u_b + u_n)/2$
- (5) Continue procedures 2–4 until iteration requirements are fulfilled and the required section from the liver picture is obtained.

1.2 K-MEANS CLUSTERING:

Clustering analysis, a key data extraction technique, is a powerful tool for analyzing and extracting meaningful insights from large amounts of information. The cluster method divides the information into categories or groups, with items within a group sharing a high degree of resemblance while being quite different to things in other groupings. The characteristic variables that describe the items are used to determine differences. Distance measurements are often employed. Grouping, being a type of facts and figures and an instance of uncontrolled training, offers us with a precise and subtle analytic instrument from a mathematical standpoint.

In cluster analysis, the K-means algorithms is a prominent partition technique. The squared-error criteria is the most often used grouping mistake criteria, and it may be expressed as:

$$J_c = \sum_{j=1}^c \sum_{k=1}^{n_j} \|x_k^{(j)} - m_j\|^2$$

Here J is the most common value or mean of cluster, is the total of square-mistake for all items in the dataset and is the region of location reflecting a certain item. K-means, which uses the squared-mistake criteria, performs well when the groupings are dense structures that are relatively well isolated from each other, but it is not ideal for finding groupings with non-convex forms or groupings of extreme size. It will split the elements in one group into two or multiple groups in order to reduce the square-error criteria. Furthermore, the ideal clusters correlates to the extremum when using this square-mistake criteria to assess grouping findings. Because the functional form contains numerous locally minimum elements, the method will end at a locally optimal if the activation findings are precisely around the localized minimum position. As a result, choosing a starting clusters location at randomly is a simple way to get to the locally optimal rather than the overall optimal. To overcome the fact that the square-error criteria makes it difficult to discern significant differences across groups, a methodology centered on representational point-focused techniques has been devised. Furthermore, there are a variety of techniques to addressing the challenge with the most basic being recurrence with randomly generated picks, which varies depending on the original beginning circumstances. To prevent becoming stuck in a local optimum, several techniques use the simulated anneal approach. The notion is that numerous sub-specimens from the database are grouped separately, and then these answers are grouped repeatedly. Finally, the revised initial centre is selected as the answer with the least amount of distortions among all answers. This document introduces an innovative enhanced K automated system that is premised on successful methods of multi-samplings and once-groupings to seek the optimised control parameters of pattern centres addressing the reliance on initial state and the restriction of the

K-means dataset that uses the square-error requirement to monitor the effectiveness of cluster formation. Our findings show that the new method achieves superior consistency and outperforms the old K-means in grouping outcomes.

1.3 Current Methodology:

- assessment of the Primary Componen
- Binary structure and form characteristics in the locality
- Classification KNN and FNN

Draw backs of Existing method:

- Low discriminating ability and a large processing burden.
- The texture analysis section is not differentiated by LBP.
- For a big form factor, FNN is a sluggish learning model.
- Segmentation precision is lower.

CHAPTER 2

SYSTEM ANALYSIS

2.1BACKGROUND

Implement an automated decision support system based on an Artificial Neural Network using Back Propagation Network for CT lung cancer diagnosis and cancer detection using a fuzzy clustering method.

Lung disease is one of the most common contributors of mortality worldwide. ILDs are a category of approximately 200 disorders that includes pulmonary fibrosis, pulmonary edema, pneumonia, and many others. With more than 50,000 instances identified each year, idiopathic pulmonary fibrosis (IPF) is emerging a community

healthcare problem and lung cancer is by far the largest incidence of cancer fatality in both males and females, responsible for roughly 25% of all cancer fatalities. Meanwhile, the globe is battling the Covid epidemic, which has resulted in a large amount of pneumonia instances. With far more than 50,000 instances identified annually, unexplained pulmonary fibrosis (IPF) has become a global healthcare problem and lungs cancer is by far the largest incidence of cancers fatality in both males and females, contributing for roughly 25% of all cancers fatalities. Meanwhile, the globe is dealing with the Covid-19 epidemic, which has resulted in a large lot of incidents of pneumonia. Lungs cancer monitoring and treatment is an international issue now more now than ever.

The traditional diagnosis procedures of ct scans and computerized tomography (CT) are the most extensively utilized in the detection and treatments of lungs diseases? A chest x-ray can detect cancer, infections, or air accumulation around a lung, which can trigger the lungs to fall. Emphysema and cystic fibrosis are two chronic lung diseases that can be discovered. The sharpness and consistency of ct scans are low, and inter-examiner variability is significant. CT scans are utilized to figure out the size and shape of an organ, as well as if it's polluted, hard, or fluid-filled. They're likewise performed to make sure you don't have influenza. This method can identify tumors, cancer, and other irregularities in the individual bodies. Regardless of the fact that a chest CT scan is more delicate than a radiograph in sensing lungs investigates and is considered the gemstone benchmark for treating pneumonia, it is not frequently utilised clinically assumed pneumonia for a variety of reasons, which include cost, inaccessibility, and radioactivity issues.

CHAPTER 3

LITERATURE SURVEY

- A. Using an adaptive hierarchical heuristic mathematical model and deep learning for lung cancer on a computer tomography image.

Lung cancer prediction, the machine learning, a deep neural network, statistical model,

ZHIQING ZHOU, HENG YU, AND QIMING WANG 2020.

There is no accurate photograph.

- B. Candidate Classification of Lung Nodule Detection Using Multi-Resolution CNN and Information Transfer.

WANGXIA ZUO, LIN WANG 2020

Multi-resolution model, convolutional neural network, lung nodule nominee classification

Difficult to identify

- C. A Profound System for Lungs Cancer Kind Recognition Using Densely Coupled Convolutional Circuits and Achieving Effectiveness.

XIAINJIN XIE 2020, SHENCHAN PANG, YAQIEN ZHENG, MAI DENG, XUN WENG

Data enhancement, ad improve algorithm, densely connected convolutional networks

Accurate findings are difficult to come by.

- D. Using Gabor filters and the watershed segmentation technique, an advanced image processing analysis for the identification of lung cancer has been developed.

S. Avinash 2016

Image processing/Fast Fourier Transform (FFT)

Poor Quality.

- E. Automatic Detection and Segmentation of Lung Nodule on CT Images.

Yangchuan 2018

Image processing / Fully Convolutional Network (FCN)

Poor detection.

CHAPTER 4

SYSTEM REQUIREMENT

4.1 HARDWARE REQUIREMENTS:

1. System : 64-bit Operating System
2. Processor : x64 processor
3. Hard Disk : 100 GB
4. RAM : 4GB

4.2 SOFTWARE REQUIREMENTS:

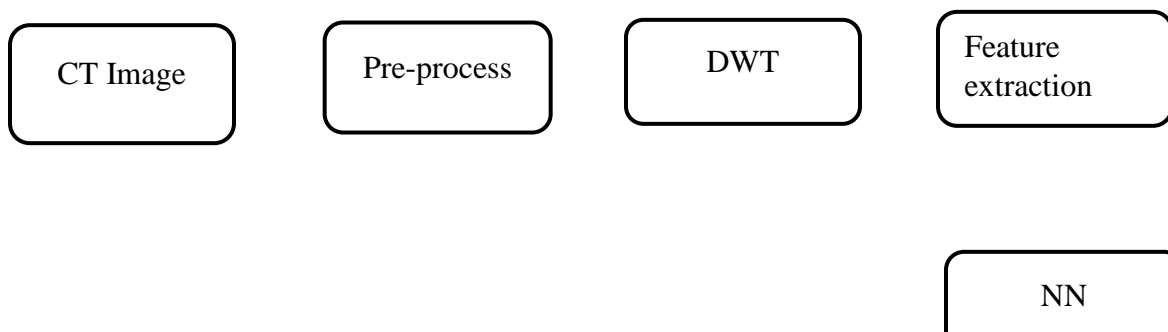
1. Operating System : Microsoft Windows 7 and above
2. Coding Language : Python
3. Application : Jupyter Notebook, Anaconda Navigator, Pycharm

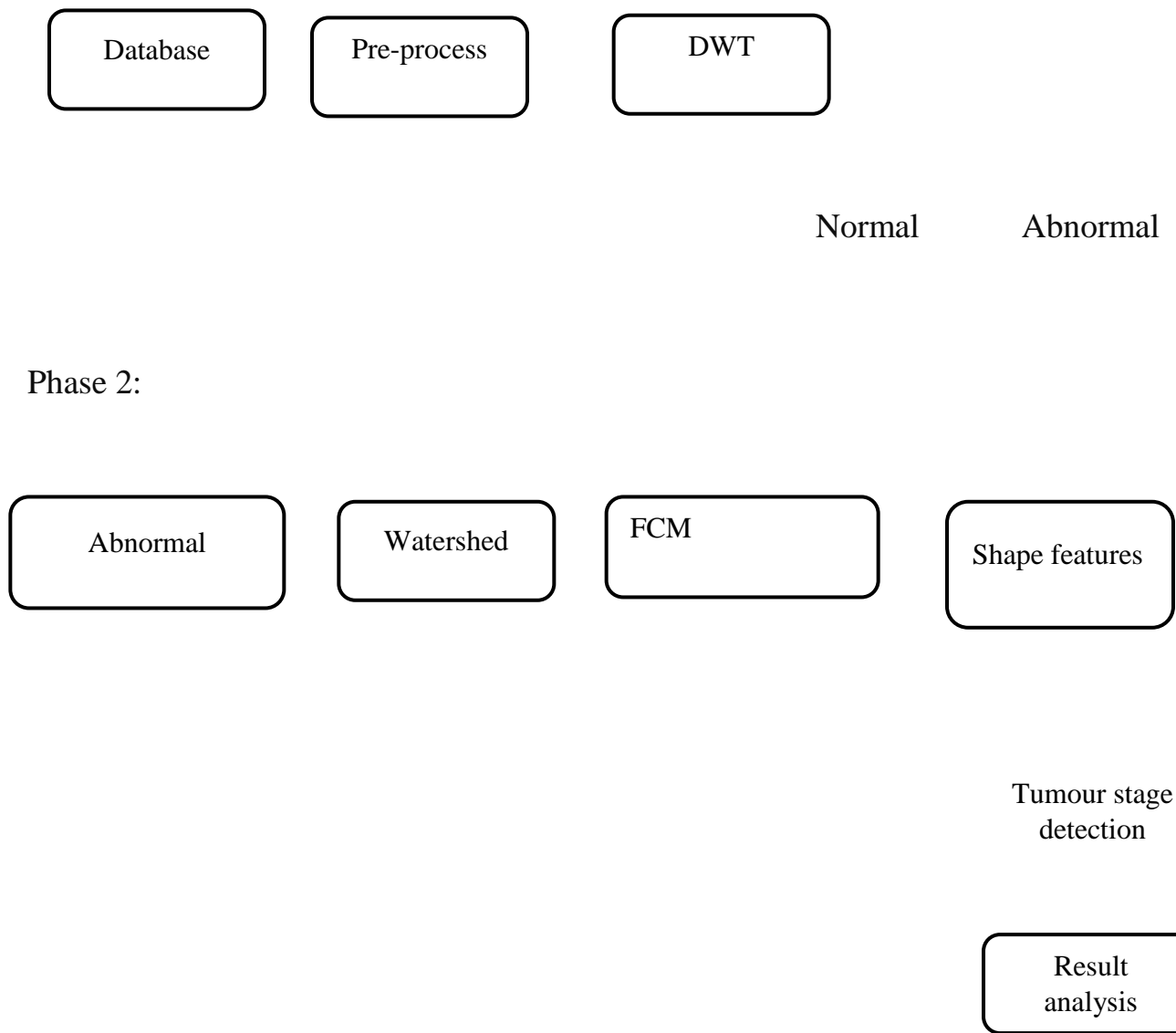
4.3 PROPOSED SYSTEM

- Colour classifiers and textural classifiers are used in mixed characteristics
- Classifiers based on neural networks

4.4 Block diagram:-

Phase 1:





4.6 Descriptions

- The project intends to use a multimodal classification algorithm and morphology method to detect tumors in CT inspected clinical pictures.
- The procedure of splitting a digitized picture into different sections is referred to as segmented.

- The disturbance in the brains Scans is reduced using filters, and then the separation of CT brain pictures is done utilizing a spatially Fuzzy C implies Grouping method
- The tumour area will be smoothed out from the chaotic backdrop using the morphology technique.
- For telemedicine applications, the divided main and secondary areas are compacted using hybridized approaches.

CHAPTER 5

METHODOLOGY

5.1 DIGITAL IMAGE PROCESSING:

Picture computing methods such as sound insulation would most likely be used to identify elements in a photograph accompanied by (reduced) features separation to detect edges, sections, and potentially places with specific texturing.

Clusters of these forms can be interpreted as single things, such as automobiles on a roadway, items on a production line, or malignant cells on a cover slip. Because an item might seem substantially differently when seen from multiple distances or under varied illumination, this is an AI challenge. Additional issue is determining which characteristics correspond to which item and which are backdrop, reflections, and so on. These functions are generally performed instinctively by the humans graphical systems but to reach physical movement, a machine needs competent coding and a considerable amount of computational capacity Manipulation of information in the manner of a picture using a variety of approaches. A picture is often portrayed by designs like those found on a photography prints slides, display screen, or cinema screen, and is generally read as a double-

dimensional matrix of intensity value. A machine may analyze a picture either visually or electronically.

To electronically analyze an image, the photograph must first be reduced to a set of statistics that the computers can alter. A photographic component, or pixels, is an integer that represents the lighting value of an image at a certain position. Whilst considerably bigger pictures are emerging more frequent, a typical digital image may contain 512 512 megapixels, or around 250,000 pixel resolution. There's many three main procedures that may be done on the picture in the computers after it has been digitised. A image size in the outgoing picture requires on a particular intensity values in the incoming picture for a company oriented. For localized computations the frequency of each output picture pixel is determined by multiple neighbouring images in the input picture. All of the input pixel values contributes to the resulting picture pixels value in an international operations.

These processes, either alone or in combinations, are used to augment, recover, or condense the picture. An picture is improved when it is changed to make the contents it holds more visible, but it may also be upgraded by rendering it more graphically attractive

Vibration softening is one instance. A 3by3 pixels screen can be used to apply average filtration to a speech signal to smoothing it out. This implies that the value of each pixels in the speech signal, as well as the frequencies of its eight closest neighbours, is captured These nine integers are then arranged accordingly to magnitude, and the midpoint is picked as the figure for the pixels in the current picture The processed picture is generated by moving the 3by3 windows one pixel at a moment over the noisy picture.

Another type of augmentation is contrasting modification, in which the values of each image in the current photo is exclusively determined by the values of that pixels in the old picture; in other terms, this is a single operations. Setting the intensity and contrasting adjustments on a tv screen, or manipulating the exposures and developing times in printing, are popular ways to manipulate contrasts. Pseudo-coloring a black and white picture by allocating random colors to the gray stages is another points operations. Warmer temperatures elements (with high image pixels) are allocated one color (for instance, red), while cooler elements (with lower image pixels) are allocated another color (for instance, blue), with additional colors allocated to intermediary levels.

Computational intelligence has long sought to recognize item types in real-life-world pictures. Due to considerable appearance differences across item examples pertaining to the identical category, this is theoretically problematic. Furthermore, illusions caused by backdrop clutter, size, and perspective differences can cause even the identical item instances to seem very differently. Interparty resemblance, which occurs when examples from distinct classes appears to be quite identical, adds to the difficulties. As a result, bounding boxes representations must be versatile sufficient to account for category heterogeneity but discriminatory enough to separate actual object categories from crowded photos. Identification is challenging due to these seemingly contradictory criteria of an objects modelling process. The two aims of identification addressed in this study are picture categorization and feature categorization Picture classification determines if a class diagram is represented in a picture, whereas item recognition locates all occurrences of that class inside that image. The major contributions of this research is a methodology for bounding boxes detection that only uses edge data to achieve these aims. Our technique is unique in that we describe contours using extremely basic

and general line segmentation and ellipse form colonialists, as well as a customizable mechanism for learning exclusionary fundamental configurations. The lines segment represents a straightforward shape, whereas the ellipse represents a curving shape We go with an ellipse since it's one of the most basic round geometries while yet being flexible enough to portray curving objects. The features of these shaped primitives are quite appealing. They facilitate conceptual and cognitively coherent argumentation such as concurrency and proximity, contrast edge-focused classifiers. Furthermore, unlike contoured fragmentation characteristics, these primitives' collection requirements are unaffected by scale factor and may be effectively expressed with four factors for a line and five variables for an elliptical.

Furthermore, unlike contoured fragmentation, which need assessments between single pixel values, matched between primitives may be easily calculated (e.g., utilizing geometrical characteristics Lastly, because geometrical attributes can be readily scaled standardized, they make cross-scaled matching easier. Contoured fragmentation, on the other hand, are not scaling independent, therefore one must either rescale remnants, which produces aliasing (e.g., when pixel values are driven away), or adjust a picture before removing bits and pieces, which reduces picture quality.

5.2 CATEGORISATION OF IMAGES:

In electronic, picture analysis there are three sorts of pictures. They really are.

1. Images in Binary
2. Images in Grayscale
3. Images in Colors

BINARY IMAGE:

A binary picture is a computer picture in which each pixels has just two potential readings. A binary picture is often made up of two colours: black and white, however any two shades can be utilized The forefront colour is utilized for the element(s) in the photograph, while the backdrop colour is utilized for the remainder of the pictures

Binary pictures are sometimes known as two-staged or bi-staged pictures. This implies that every pixels is represented by a single bit (0 or 1). This idea is typically referred to as black and white, monochromatic, although it may also refer to pictures with only one specimen per pixels, such as grey level pictures.

In electronic, imaging computing binary pictures frequently appear as filters or as the consequence of procedures such as categorization, thresholding, and wobbling. Only bi-levelled pictures can be handled by some instruction technologies, such as laser printers, faxes, and bi-staged computer screens.

IMAGE IN GRAY SCALE:

A monochrome image A digitized picture is one in which each screen's values is an unique sampling meaning it only contains intensities values This type of image, often called black, is made up entirely of different hues of gray (0-256), ranging from black at the shortest wavelength to white (245) at the greatest.

Grey level pictures are different from one-bits black-and-white pictures, which are pictures with just two colours, black and white, in the setting of desktop photography (also termed bi-levelled or binary pictures). There are various hues of gray in gray scale photographs. Monochrome pictures are grey level pictures that are devoid of any chromatic variations.

Grey level pictures are frequently created by monitoring the illumination at every pixels in a particular part of the electrical spectra (e.g., infra, noticeable sight, uv, etc.), and they are monochrome when just one wavelength is caught. They may also be created from a full-coloured image; check the sections on graylevel conversion.

COLOUR IMAGE:

A (computerized) colour picture is one in which each pixel contains colour detail every pixels has a unique value that affects the color it appears to be. This quantity is accompanied by three digits that represent the color combinations breakdown into the three major hues of Red, Green, and Blue. This method may be used to symbolize any hue accessible to the naked eye A value between 0 and 255 represents the breakdown of a colour into its three main hues. For instance, white will be $R = 254, G = 254, \text{ and } B = 254$; black will be $(R,G,B) = (0,0,0)$; and brilliant pink will be : $(254,0,254)$.

To put it another way, a picture is a massive dual-dimensional arrays of colours, megapixels, each of which is encoded with three bytes and represents the three basic colours. This enables the picture to have a maximum of 16.8 million distinct colours $(254 \times 254 \times 254)$. This method is also known as RGB decoding, and it is designed exclusively for human eyesight.

It is clear that the sentiments of the individuals with whom we wish to contact have a significant impact on our behaviour and social interactions. As a result, a good emotion detection technology might have a significant influence on enhancing individual-computer interactions systems by making them more customer-friendly and human-centred.

Sentiment analysis can also be useful in a variety of situations, involving identity verification, high-end-techno monitoring and safety technologies picture extraction, and passively socio - demographic information gathering. It is undeniable that the face is one of the most distinguishing features of living individuals. We can not only know who someone is by staring at their appearance, but we can also infer a wealth of details about them, such as their moods, occupations, and sexes. This is why, during the last twenty years, the computerized visual study group has been so interested in emotions identification by appearance.

5.3 PREPROCESSING

Picture recovery is the process of approximating the clear actual picture from a corrupted image. Motion blurring sound, and lens concentration are all examples of corrupt practices. Picture augmentation differs from image reconstruction in that the latter is intended to accentuate characteristics of the picture that make it more attractive to the spectator, rather than automatically producing true information from a technical standpoint. Picture enhancing methods supplied by "Visualization programs" do not require an a posteriori description of the activity that formed the image (such as contrasting bending or de-blurring via a closest neighbours algorithm). Disturbance may be successfully eliminated using picture improvement by surrendering considerable quality, but this is not desirable in many situations. Performance in the z-directional of an Electron Microscopy is already poor. To retrieve the item, more complex methodologies must be used. Image restoration methods such as de-convolution are one instance. It is eligible to enhance clarity, particularly in the axis orientation as well as reducing noise and boosting contrasts.

Dual-tree complex wavelet transforms (DT-CWT)

The double-tree complicated wavelets is a new modification to the discontinuous waveform transformation (DWT) that has many key advantages: In two and further directions, it is almost shifting insensitive and spatially selected. For d-dimensional transmissions, it does this with a redundant estimate of just $2d$, which is significantly smaller than the truncated DWT. The multifunctional (M-D) double-tree CWT is not recoverable, but it is built on a separable filter bank that is operationally economical. The idea underlying double-tree transformations demonstrates how complicated waveforms with desirable characteristics may be developed, as well as a variety of digital image analysis implementations.

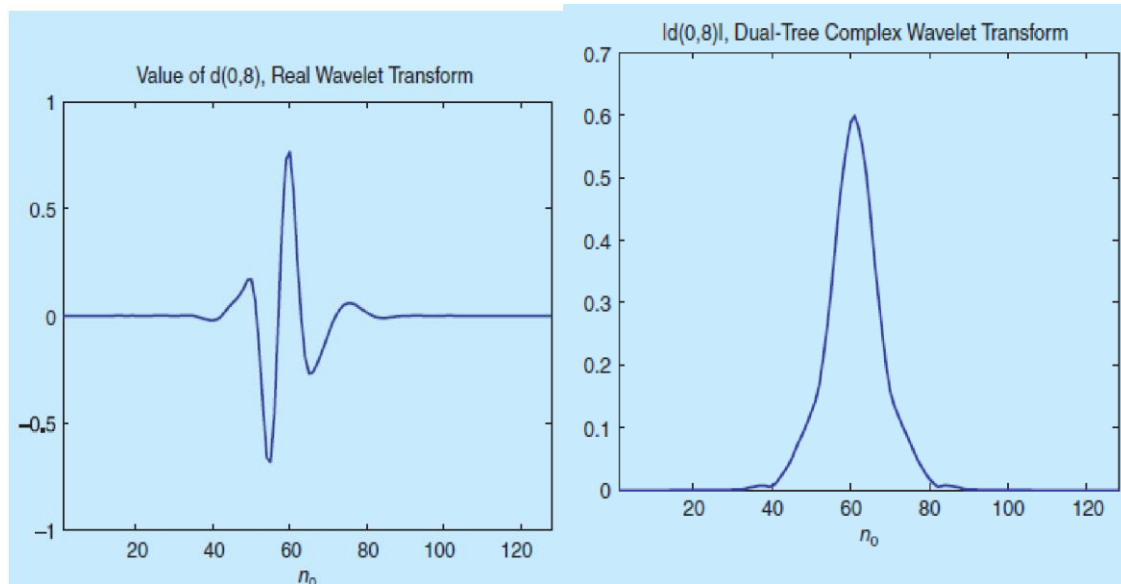


Fig: the value of the wavelet coefficient in “Real-Valued Discrete Wavelet Transform and Filter Banks

The genuine DWT creates both big and tiny wavelets in the vicinity of an edges. The (roughly) analytical CWT, on the other hand, generates variables whose dimensions are more closely connected to their closeness to the border. The testing signals is a stepped edges at $n = n_0, x(n) = u(n - n_0)$. The wave front coefficients $d(0, 8)$ (the eight component at phase three in "Real-Estimated Discrete Wavelets Transform and Filtering Banking") is plotted as a relationship of n_0 . The actual coefficients $d(0, 8)$ is determined utilizing the traditional real DWT in the superior surface. The dual-tree CWT is used to calculate the complicated coefficients $(0, 8)$ in the lower right corner.

5.4 CLUSTERING MODEL

Grouping is the most significant uncontrolled training issue since it involves discovering a framework in a set of unstructured information. As a result, a cluster is a group of things that are "typical" to one another but "different" to items from other groups.

FUZZY CLUSTERING MODEL

In the fields of information processing and imprecise models validation, imprecise grouping serves a significant function. There have been a number of imprecise grouping approaches suggested, the most of them are predicated on proximity requirements. The imprecise c (FCM) method is a commonly used technique. It computes ambiguous parameters using inverse proximity. The new FCFM technique is a more powerful technique. It employs Gaussian values to determine the clusters, big starting models, and elimination, grouping, and combining algorithms. The FCM and FCFM algorithms are discussed and compared in the subsequent subsections.

After considering the grouping distributions in the neighbourhood, the participation weighted of each group is updated using the Geographical Fuzzy C Means approach, which adds spatial data. To compute the participation function in the frequency domain, the first step is the equivalent as in normal FCM. The participation characteristics of each pixels is translated to the frequency domains in the second viewing, and the geographical functions is calculated from there. The current subscription that is combined with the geographical functions is used in the FCM repetition. When the largest discrepancy between clusters cores or memberships parameters between successive lines is less than a minimum predefined threshold, the repetition is ended.

J. C. Bezdek [2] proposed the imprecise c (FCM) method. FCM is based on the concept of applying weights to reduce total weighting mean deviation:

$$J(\mathbf{w}_{qk}, \mathbf{z}^{(k)}) = \sum_{k=1, K} \sum_{k=1, K} (\mathbf{w}_{qk}) \|\mathbf{x}^{(q)} - \mathbf{z}^{(k)}\|^2$$

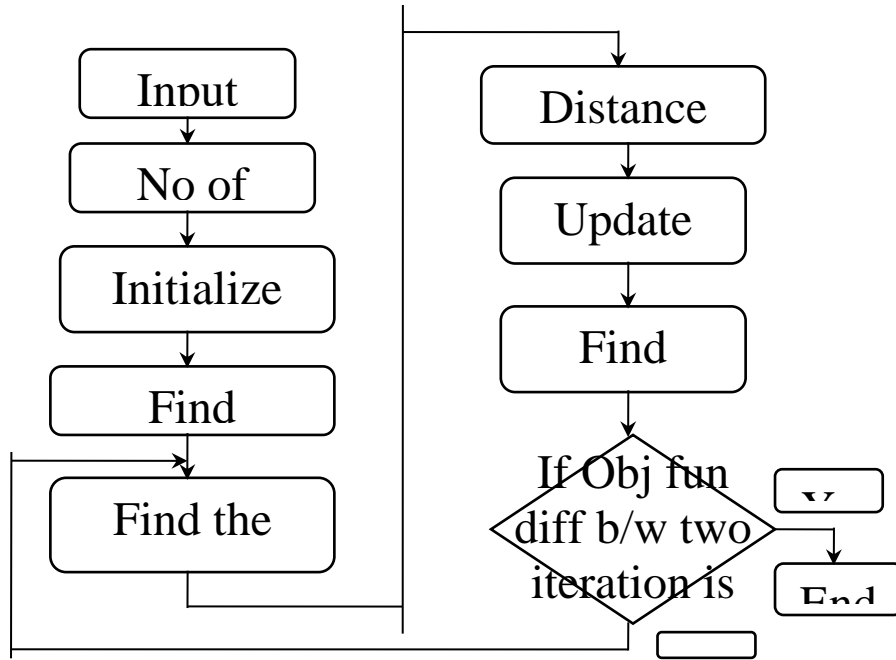
$$\sum_{k=1, K} (\mathbf{w}_{qk}) = 1$$

$$\mathbf{w}_{qk} = (1/(\mathbf{D}_{qk})^2)^{1/(p-1)} / \sum_{k=1, K} (1/(\mathbf{D}_{qk})^2)^{1/(p-1)}, p > 1$$

The FCM enables each characteristic vectors to correspond to any cluster that has an imprecise validity (between 0 and 1) determined utilizing Equations (4). The approach allocates a feature representation to a clusters based on the characteristic element's greatest weight across all groups.

A New Fuzzy c-means Implementation

Algorithm Flow



Set the Irregular Weights to their default values. Our method shows the available between utilizing features matrices or randomized to initialize the values when matching the FCM and FCFM. The initialization of the weights using features matrices allocates the first (user-provided) characteristic matrices to models, after which the values are computed applying Equations. (4).

Standardize the Weights over Q. The calculated clusters come nearer and nearer over the FCM cycle. We utilize before normalizing the values across Q to prevent quick converging and always clustering into one bunch. Where w_{max} and w_{min} are the highest and lowest values for all characteristic matrices for the given category design, respectively.

$$w[q,k] = (w[q,k] - w_{min}) / (w_{max} - w_{min})$$

Eliminating Empty Clusters. We introduce a stage (Stage 8) just after hierarchical based process to remove the vacant groupings. This stage comes after the imprecise grouping cycle and before the enhanced XB authenticity computation. The minimal separation of the model pairing utilized in Equations (8) might be the length of an unfilled clustered pairing if the reduction is not performed. We use the way of protecting tiny groupings by giving it a value of 0 to ensure that it only removes vacant groups.

We include Stage 9 to compute the clusters and the updated Xie-Beni grouping authenticity after the fcm repetition for the objective of comparability and to choose the best outcome

Robustness and dispersion parameters contribute to the Xie-Beni reliability. Equations is used to determine the robustness-to-separation proportion.(6).

$$\mathcal{J} = \{(1/K) \sum_{k=1, K} \mathcal{I}_k^2\} / D_{\min}^2$$

$$\mathcal{I}_k^2 = \sum_{q=1, Q} w_{qk} \| \mathbf{x}^{(q)} - \mathbf{c}^{(k)} \|^2$$

D_{\min} is the minimal length between the clusters point.

The updated Xie-Beni authenticity termed as

$$| = D_{\min}^2 / \{ \sum_{k=1, K} \mathcal{I}_k^2 \}$$

In contradiction to the conventional Xie-Beni authenticity assessment, the variability of each group is derived by combining just the individuals of each clusters rather than all Q for each group.

$$\int_k^2 = \sum_{\{q: q \text{ is in cluster } k\}} w_{qk} \| \mathbf{x}^{(q)} - \mathbf{c}^{(k)} \|^2$$

As seen in Equations, the geographical functional is incorporated into the memberships measure.

$$u'_{ij} = \frac{u_{ij}^p h_{ij}^q}{\sum_{k=1}^c u_{kj}^p h_{kj}^q}$$

CO-OCCURRENCE MATRIX

- The co-happening matrix form of textural characteristics first developed by R.M. Horlick, investigates the grey layer spatial dependency of textures [2]. The following is a quantitative explanation of the co-happening matrices [4]:
 - Assume P is a positioning operation (i,j),
 - Assume A is a n x n matrices.
 - - whose component The amount of instances elements with gray scale (intensities) g[i] appear in the location given by P, compared to locations with gray scale (density) g[j], is A[i][j]. **g[j]**.
- Define C as the n x n matrices obtained by splitting A by the overall amount of points pairings fulfilling P. The combined likelihood that a combination of locations meeting P will have quantities g[i], g[j] is measured by C[i][j].
-

A co-occurring matrices, or C, is a structure described by P.

I above j,” I one place to the right and two beneath j,” and so on are instances of the operators P.

This may also be demonstrated in the following way... Let t be a translator, and then a matrices of co-occurrences. For each greyish (a, b), Ctof a limit is identified by [1]:

$$C_t(a,b) = \text{card}\{(s, s+t) \in R^2 | A[s] = a, A[s+t] = b\}$$

Ct(a, b) is the quantity of site-coupled separated by a translations vectors t, indicated as (s, s + t), with a being the gray-level of s and b becoming the gray-level of s + t.

With an 8-level picture representations and a vectors t that only examines one neighbour, for instance, we would discover [1]:

```

1  2  1  3  4
2  3  1  2  4
3  3  2  1  1

```

Figure: Image example

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	0	0	0	0	0
2	0	1	0	2	0	0	0	0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

Figure: Classical Co-occurrence matrix

The co-occurring matrix is built first, depending on the picture components' orientations and proximity. The textural representation is then created by extracting useful parameters from the matrices. Haralick offered the texturing option below:

1. power
2. dissimilarity
3. connection
4. Homogeneity

As a result, we get a co-occurring matrix for each Haralick texturing element the geographic allocation and dependency of the gray values within a small region are represented by these co-occurring matrices. The chance of moving from one pixels with a gray scale of i to another with a gray scale of j across a given angular displacement is represented by the (i,j) th accession in the matrix. Sets of objective methods, known as characteristic matrices, are constructed from these combinations.

Power: It is a grey-scale picture textural uniformity altering metric that reflects the picture grey-scale consistency of mass and textured distributions.

$$E = \sum_x \sum_y p(x, y)^2$$

$p(x,y)$ is the GLC M

Dissimilarity: Contrasting is the major diagonally near the point of tension which measures the values of the matrices and pictures of local variations in numbers, representing picture purity and shadows depths textures.

Contrast
$$I = \sum \sum (x - y)^2 p(x, y)$$

Correlation Coefficients: The combined likelihood of the provided pixels pairings occurring is calculated.

Connection:
$$\text{sum}(\text{sum}((x - \mu_x)(y - \mu_y)p(x, y)/\sigma_x\sigma_y))$$

Homogeneity: The distance between the GLCM diagonally and the allocation of components in the GLCM.

Homogeneity =
$$\text{sum}(\text{sum}(p(x, y)/(1 + [x - y])))$$

Drawbacks

- Poor discriminatory power
- High computational load
- Edge information are lost owing to the shifting variation attribute.

KNN Classifier

The k-closest neighbour algorithms (k-NN) is an information processing approach for categorizing items predicated on the characteristic space's nearby learning samples k-NN is a sort of instance-focused education, often known as lazy teaching methods, in which the functions is only estimated temporarily and all computing is postponed until after categorization. The k-closes neighbour method is one of the most straightforward of all machines training methodologies an item is categorized by a plurality voting of its neighbours, with the item being allocated to the most basic category amongst the k closest neighbouring nodes (k is a positive number, generally tiny). If $k = 1$, the item is typically considered as an early neighbour's house category.

The same procedure may be utilized for reversion by basically allocating the element's characteristic variable to the averages of itsk closest neighbours' numbers. It may be beneficial to weigh the participation of neighbours such that the neighbours who are closer give more to the aggregate than those who are farther away. (A popular weighting system is to provide a weight of $1/d$ to each neighbour, where d is the proximity between them.) This method is a broadening of regular approximation.

5.5 ARTIFICIAL NEURAL NETWORK

Neural Network:

Neural channels are prediction frameworks that are partially founded on biological neuronal function

One of the great PR triumphs of the twentieth century was the choice of the word neural networks “A structure of balanced, cumulative numbers with nonlinear transmission operations seems a lot more fascinating than “A system of graded, cumulative numbers with nonlinear transmission operations Despite its name, neural connections aren't "reading computers" or "unnatural lungs." A 100 cells make up a common artificial neural networks The human nervous systems on the other hand, is thought to have around 3×10^{10} cells. We're still a long way from "Information."

Frank Rosenblatt created the initial "Perceptron" framework in 1958. Rosenblatt's paradigm included 3 levels: (1) a "retina" that dispersed inputs to the second layer, (2) "identification elements" that combined inputs with weighting and triggered a minimum stepping mechanism that fed into the output nodes, and (3) the output vector that combined the results. Regrettably, because the neurons used a phase mechanism training senses was challenging or unachievable Marvin Minsky and Seymour Papert released a comprehensive critique of perceptron's in 1969, pointing out a variety of fundamental flaws, and curiosity in perceptron's declined for a while.

When David Rumelhart, Geoffrey Hinton, and Ronald Williams released “Learners Inner Depictions by Mistake Propagation in 1986, it reignited interest in neural nets. They created a multilayered computer program with nonlinear yet distinguishable transmission characteristics that eliminated the drawbacks of the factor levels used in the initial perceptron. They also offered a pretty good neuronal networks learning technique.

Kinds of Neural Systems:

1) A neuronal system that is unnatural

2) Systems of back propagating

3) Neuronal Systems for Generalized Regression

DTREG provides the most commonly utilized neuronal system models:

a) Perceptron Systems with Multiple Layers (also referred as multilayered feed-forwards systems,

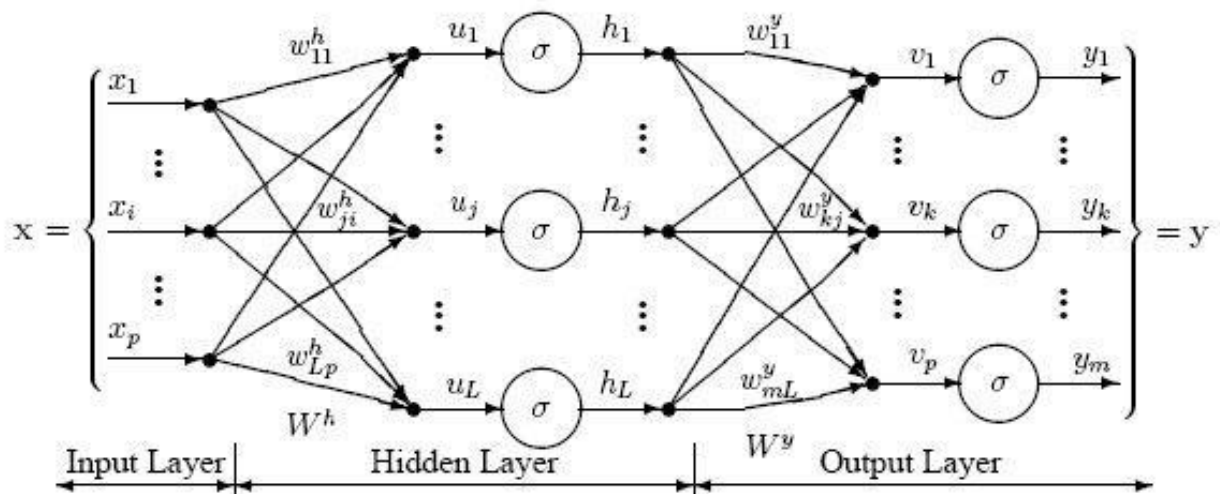
b) Neural Systems with Cascades Coupling

Back propagation neural systems are a type of system that allows information to spread backwards (BPN)

c) Neural Systems for Regression (GRNN)

The Multi layered Perceptron Neuronal System Framework

A three-layered perceptron connectivity is depicted in the figure below:



This system comprises three cells in the input nodes, three cells in the camouflaged layers (in the centre), and three cells in the output nodes (on the right hand side).

Each predictors has its own neurons in the input neurons. When dealing with explanatory data, $N-1$ synapses are utilized to symbolize the variable's N divisions.

Input Layered — The input stratum is given a matrix of predictors variables ($x_1 \dots x_p$). The input layers (or the procedure that comes before it) provide a standard those numbers so that each variable's spectrum is -1 to 1 . The numbers are distributed to each of the nodes in the input layers by the input layers. A continuous input of 1.0 , termed the biases, is provided to each of the concealed levels in addition to the predictors factors the prejudice is compounded by a load and appended to the total flowing into the neurons.

Hidden Layered — When the number from each inputs neuron reaches a concealed layers neuron, it is assigned a weight (w_{ji}), and the weighed numbers are joined altogether to receive a total value u_j . The summation (u_j) is sent via a transferring operator which returns h_j . The output layers receive the signals from the concealed layers.

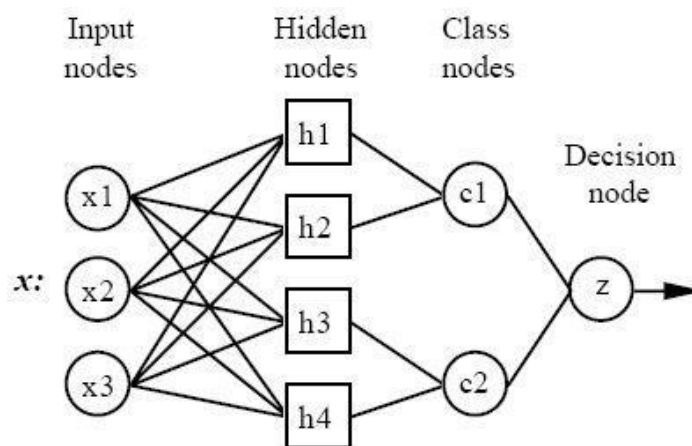
Output Layered — When the number from each concealed layers neurons reaches a neuronal in the output nodes, it is assigned a weight (w_{kj}), and the real impacts are joined altogether to get a composite values v_j . The weighed estimate (v_j) is sent via a transferring functions which produces the number y_k . The channel's outputs are represented by the y variables.

When a constant aim factor is used in a predictive study the output layers has a single node that outputs a single scale factor. In categorization situations with classified destination factors the output layers has N neurons that produce N numbers one for each of the destination variable's N subcategories.

Back propagation networks (BPN):

The structures of Back Propagation and Generalized Regression Neuronal Networking (GRNN) are identical, however there is a key distinction: When the objective variables is categorized probability models do categorization, whereas generic regression neuronal networking conduct regress when the objective variables is extended DTREG will autonomously identify the relevant sort of networks dependent on the kind of destination variables if you choose a BPNN networks.

Framework of a BPN:



All BPN networks have four layers:

1. **Input layers** — Each explanatory variables has its own neurons in the input nodes. $N-1$ neurons are utilized for categorizing data where N is the integer of subcategories. By removing the midpoint and divided by the standard deviation, the inputs neurons (or processes preceding the inputs layer) normalize the scope of data The numbers are then sent to each of the cells in the concealed layers by the neural network.

2. **Hidden layers** — Each instance in the learning information matrix contains one neuronal in this stratum. Together with the goal number the neurons keeps the quantities of the case's predictors. When given the x vectors of input parameters from the input nodes, a concealed neurons calculates the Euclidean distances of the testing ground from the neurotransmitter midpoint and then uses the sigma number to apply the RBF kernel function (s). The resultant value is sent to the patterning material's synapses.
3. **Pattern layers / Summation layers** — For BPNN systems, the following stratum in the system is distinct. Each subcategory of the targeted variables has one patterned neurons in BPN systems. Each concealed neuronal stores the preferred aim classification of each learning event the weighed number output by a concealed neuronal is only supplied to the patterned neuronal that correlates to the concealed neurotransmitter categorization. The scores for the category that the patterns cells indicate are added together (therefore, it is a weighed voting for that class).
4. **Decision layers** — For BPN systems, the deciding layers is distinct. The determination phase in BPN systems analyzes the weighed values gathered in the design level for each targeted model and predicts the targeted class based on the highest majority.

5.6 BACK PROPAGATION ALGORITHM

Considering a system with only one genuine inputs x , and the networking functions F . The $F'(x)$ derivatives is calculated in two stages:

The inputs x is supplied into the networks in a feed-forwards fashion. At each nodes, the fundamental equations and their modifications are assessed. The compounds are kept on hand.

Back propagation: The outputs layer receives the consistent or unchanging 1 and the networking is reversed. The number recorded in the left side of the component is amplified by the sum of receiving input to a network. The output is sent to the command's left side. The derivatives of the networking functions with regard to x is the information obtained at the inputs layer.

STAGES OF THE ALGORITHMS

After arbitrarily selecting the channel's values, the back propagation neural technique is utilized to determine the required adjustments the four main stages can be used to breakdown the automated system:

I Feed-forwards calculus

ii) Transmission from the input layers to the output layers

iii) Weighted modifications

iv) Back propagation to the buried level

CHAPTER 6

CONCLUSION

Lungs infection is a global public healthcare issue all over the world, so reliable treatment and secure surveillance of lungs infection are becoming increasingly critical. This has been particularly significant throughout the COVID-19 pandemic. The portability, cost-effectiveness, and safety offered by lung analysis allow for assessment at the bedside and repeat exams during follow-up. The ability to perform this treatment at the bedside can minimize the need for the patient to be transported, CT radiation exposure, and the likelihood of the COVID-19 infection spreading among healthcare staff. This could lead to the implementation of an automatic "COVID-19 lung rating" and a more accurate, quantitative method of measuring superficial lung stiffness. This method of predictive measurement will allow for precise statistical analysis of a patient's lung condition (hour-by-hour or day-by-day) to ascertain the efficacy of disease treatment.

CHAPTER 7

7.1 PROJECT CODE

```

import time
import math
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import dataset
import cv2
import os

from sklearn.metrics import confusion_matrix
from datetime import timedelta

# Convolutional Layer 1.
filter_size1 = 3
num_filters1 = 32

# Convolutional Layer 2.
filter_size2 = 3
num_filters2 = 32

# Convolutional Layer 3.
filter_size3 = 3
num_filters3 = 64

# Fully-connected layer.
fc_size = 128          # Number of neurons in fully-connected layer.

# Number of color channels for the images: 1 channel for gray-scale.
num_channels = 3

# image dimensions (only squares for now)
img_size = 32

# Size of image when flattened to a single dimension
img_size_flat = img_size * img_size * num_channels

# Tuple with height and width of images used to reshape arrays.
img_shape = (img_size, img_size)

# class info
classes = ['Abnormal', 'Normal', '']
num_classes = len(classes)

```

```

# batch size
batch_size = 1

# validation split
validation_size = .16

# how long to wait after validation loss stops improving before terminating training
early_stopping = None

train_path = 'data/'
test_path = 'test/'
checkpoint_dir = "models/"

data = dataset.read_train_sets(train_path, img_size, classes, validation_size=validation_size)
test_images, test_ids = dataset.read_test_set(test_path, img_size)

print("Size of:")
print("- Training-set:/t/t{}".format(len(data.train.labels)))
print("- Test-set:/t/t{}".format(len(test_images)))
print("- Validation-set:/t/t{}".format(len(data.valid.labels)))

# Get some random images and their labels from the train set.
images, cls_true = data.train.images, data.train.cls

# Plot the images and labels using our helper-function above.
plot_images(images=images, cls_true=cls_true)

##Helper-functions for creating new variables
def new_weights(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.05))
def new_biases(length):
    return tf.Variable(tf.constant(0.05, shape=[length]))
def new_conv_layer(input,          # The previous layer.
                   num_input_channels, # Num. channels in prev. layer.
                   filter_size,      # Width and height of each filter.
                   num_filters,      # Number of filters.
                   use_pooling=True): # Use 2x2 max-pooling.

```

```

shape = [filter_size, filter_size, num_input_channels, num_filters]

# Create new weights aka. filters with the given shape.
weights = new_weights(shape=shape)

# Create new biases, one for each filter.
biases = new_biases(length=num_filters)
layer = tf.nn.conv2d(input=input,
                     filter=weights,
                     strides=[1, 1, 1, 1],
                     padding='SAME')

# Add the biases to the results of the convolution.
# A bias-value is added to each filter-channel.
layer += biases

if use_pooling:
    layer = tf.nn.max_pool(value=layer,
                           ksize=[1, 2, 2, 1],
                           strides=[1, 2, 2, 1],
                           padding='SAME')

layer = tf.nn.relu(layer)
return layer, weights

def flatten_layer(layer):
    # Get the shape of the input layer.
    layer_shape = layer.get_shape()
    num_features = layer_shape[1:4].num_elements()
    layer_flat = tf.reshape(layer, [-1, num_features])
    return layer_flat, num_features

def new_fc_layer(input,
                 num_inputs,
                 num_outputs,
                 use_relu=True):

    # Create new weights and biases.
    weights = new_weights(shape=[num_inputs, num_outputs])
    biases = new_biases(length=num_outputs)
    layer = tf.matmul(input, weights) + biases

```

```

# Use ReLU?
if use_relu:
    layer = tf.nn.relu(layer)

return layer

x = tf.placeholder(tf.float32, shape=[None, img_size_flat], name='x')
x_image = tf.reshape(x, [-1, img_size, img_size, num_channels])
y_true = tf.placeholder(tf.float32, shape=[None, num_classes], name='y_true')
y_true_cls = tf.argmax(y_true, dimension=1)

##Convolutional Layer 1
layer_conv1, weights_conv1 = \
    new_conv_layer(input=x_image,
                    num_input_channels=num_channels,
                    filter_size=filter_size1,
                    num_filters=num_filters1,
                    use_pooling=True)

layer_conv1    # layer 1 output image data

layer_conv2, weights_conv2 = \
    new_conv_layer(input=layer_conv1,
                    num_input_channels=num_filters1,
                    filter_size=filter_size2,
                    num_filters=num_filters2,
                    use_pooling=True)

layer_conv3, weights_conv3 = \
    new_conv_layer(input=layer_conv2,
                    num_input_channels=num_filters2,
                    filter_size=filter_size3,
                    num_filters=num_filters3,
                    use_pooling=True)

layer_conv2
layer_conv3
##Flatten Layer

```

```

layer_flat, num_features = flatten_layer(layer_conv3)
## Fully-Connected Layer
layer_fc1 = new_fc_layer(input=layer_flat,
                          num_inputs=num_features,
                          num_outputs=fc_size,
                          use_relu=True)
layer_fc2 = new_fc_layer(input=layer_fc1,
                          num_inputs=fc_size,
                          num_outputs=num_classes,
                          use_relu=False)

## class prediction
y_pred = tf.nn.softmax(layer_fc2)
y_pred_cls = tf.argmax(y_pred, dimension=1)

## Cost-function to be optimized
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=layer_fc2,
                                                          labels=y_true)

##Optimization Method
cost = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate=1e-4).minimize(cost)
##Performance Measures
correct_prediction = tf.equal(y_pred_cls, y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

####TENSORFLOW SESSION
session = tf.Session()

session.run(tf.initialize_all_variables())
train_batch_size = batch_size

##Helper-function to perform optimization iterations
def print_progress(epoch, feed_dict_train, feed_dict_validate, val_loss):
    # Calculate the accuracy on the training-set.
    acc = session.run(accuracy, feed_dict=feed_dict_train)
    val_acc = session.run(accuracy, feed_dict=feed_dict_validate)
    msg = "Epoch {0} --- Training Accuracy: {1:>6.1%}, Validation Accuracy: {2:>6.1%}, Validation Loss: {3:.3f}"
    print(msg.format(epoch + 1, acc, val_acc, val_loss))

# Counter for total number of iterations performed so far.
total_iterations = 0

```



```

def optimize(num_iterations):
    # Ensure we update the global variable rather than a local copy.
    global total_iterations

    # Start-time used for printing time-usage below.
    start_time = time.time()

    best_val_loss = float("inf")
    patience = 0

    for i in range(total_iterations,
                   total_iterations + num_iterations):
        x_batch, y_true_batch, _, cls_batch = data.train.next_batch(train_batch_size)
        x_valid_batch, y_valid_batch, _, valid_cls_batch = data.valid.next_batch(train_batch_size)

        # Convert shape from [num examples, rows, columns, depth]
        # to [num examples, flattened image shape]

        x_batch = x_batch.reshape(train_batch_size, img_size_flat)
        x_valid_batch = x_valid_batch.reshape(train_batch_size, img_size_flat)

        # Put the batch into a dict with the proper names
        # for placeholder variables in the TensorFlow graph.
        feed_dict_train = {x: x_batch,
                           y_true: y_true_batch}

        feed_dict_validate = {x: x_valid_batch,
                              y_true: y_valid_batch}
        session.run(optimizer, feed_dict=feed_dict_train)

        # Print status at end of each epoch (defined as full pass through training dataset).
        if i % int(data.train.num_examples/batch_size) == 0:
            val_loss = session.run(cost, feed_dict=feed_dict_validate)
            epoch = int(i / int(data.train.num_examples/batch_size))

            print_progress(epoch, feed_dict_train, feed_dict_validate, val_loss)

            if early_stopping:
                if val_loss < best_val_loss:
                    best_val_loss = val_loss
                    patience = 0
                else:
                    patience += 1

            if patience == early_stopping:
                break

```

```

# Update the total number of iterations performed.
total_iterations += num_iterations

# Ending time.
end_time = time.time()

# Difference between start and end-times.
time_dif = end_time - start_time

# Print the time-usage.
print("Time elapsed: " + str(timedelta(seconds=int(round(time_dif)))))

print(total_iterations)
##Helper-function to plot example errors
def plot_example_errors(cls_pred, correct):
    incorrect = (correct == False)

    # Get the images from the test-set that have been
    # incorrectly classified.
    images = data.valid.images[incorrect]

    # Get the predicted classes for those images.
    cls_pred = cls_pred[incorrect]

    # Get the true classes for those images.
    cls_true = data.valid.cls[incorrect]

    # Plot the first 9 images.
    ##     plot_images(images=images[0:9],
    ##                 cls_true=cls_true[0:9],
    ##                 cls_pred=cls_pred[0:9])
def plot_confusion_matrix(cls_pred):
    cls_true = data.valid.cls

    # Get the confusion matrix using sklearn.
    cm = confusion_matrix(y_true=cls_true,
                          y_pred=cls_pred)

    # Print the confusion matrix as text.
    print(cm)
    # Plot the confusion matrix as an image.
    plt.matshow(cm)

```



```

# Make various adjustments to the plot.
plt.colorbar()
tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, range(num_classes))
plt.yticks(tick_marks, range(num_classes))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

##Helper-function for showing the performance
def print_validation_accuracy(show_example_errors=False,
                             show_confusion_matrix=False):

    # Number of images in the test-set.
    num_test = len(data.valid.images)
    cls_pred = np.zeros(shape=num_test, dtype=np.int)
    i = 0

    while i < num_test:
        # The ending index for the next batch is denoted j.
        j = min(i + batch_size, num_test)

        # Get the images from the test-set between index i and j.
        images = data.valid.images[i:j, :].reshape(batch_size, img_size_flat)

        # Get the associated labels.
        labels = data.valid.labels[i:j, :]

        # Create a feed-dict with these images and labels.
        feed_dict = {x: images,
                     y_true: labels}

        # Calculate the predicted class using TensorFlow.
        cls_pred[i:j] = session.run(y_pred_cls, feed_dict=feed_dict)

        # Set the start-index for the next batch to the
        # end-index of the current batch.
        i = j

    cls_true = np.array(data.valid.cls)
    cls_pred = np.array([classes[x] for x in cls_pred])

    # Create a boolean array whether each image is correctly classified.
    correct = (cls_true == cls_pred)
    correct_sum = correct.sum()
    acc = float(correct_sum) / num_test

```

```

# Print the accuracy.
msg = "Accuracy on Test-Set: {0:.1%} ({1} / {2})"
print(msg.format(acc, correct_sum, num_test))

# Plot some examples of mis-classifications, if desired.
if show_example_errors:
    print("Example errors:")
    plot_example_errors(cls_pred=cls_pred, correct=correct)

# Plot the confusion matrix, if desired.
if show_confusion_matrix:
    print("Confusion Matrix:")
    plot_confusion_matrix(cls_pred=cls_pred)

optimize(num_iterations=400) # We performed 100 iterations above.
print_validation_accuracy(show_example_errors=True)
print_validation_accuracy(show_example_errors=True, show_confusion_matrix=True)

#### TESTING IMAGE INPUT ####

inputface = cv2.imread('03.jpg')
cv2.imshow("frame",inputface)
inputface = cv2.resize(inputface, (img_size, img_size), cv2.INTER_LINEAR) / 255
##plt.imshow(inputface.reshape(img_size, img_size, num_channels))

def sample_prediction(test_im):

    feed_dict_test = {
        x: test_im.reshape(1, img_size_flat),
        y_true: np.array([[2,1,0]])
    }

    test_pred = session.run(y_pred_cls, feed_dict=feed_dict_test)
    return classes[test_pred[0]]

print("output test data: {}".format(sample_prediction(inputface)))

def plot_conv_weights(weights, input_channel=0):
    # Assume weights are TensorFlow ops for 4-dim variables
    # e.g. weights_conv1 or weights_conv2.

    # Retrieve the values of the weight-variables from TensorFlow.
    # A feed-dict is not necessary because nothing is calculated.

```

```

# At this stage it is not necessary, because nothing is calculated.
w = session.run(weights)

# Get the lowest and highest values for the weights.
# This is used to correct the colour intensity across
# the images so they can be compared with each other.
w_min = np.min(w)
w_max = np.max(w)

# Number of filters used in the conv. layer.
num_filters = w.shape[3]

# Number of grids to plot.
# Rounded-up, square-root of the number of filters.
num_grids = math.ceil(math.sqrt(num_filters))

# Create figure with a grid of sub-plots.
fig, axes = plt.subplots(num_grids, num_grids)

# Plot all the filter-weights.
for i, ax in enumerate(axes.flat):
    # Only plot the valid filter-weights.
    if i < num_filters:
        # Get the weights for the i'th filter of the input channel.
        # See new_conv_layer() for details on the format
        # of this 4-dim tensor.
        img = w[:, :, input_channel, i]

        # Plot image.
        ax.imshow(img, vmin=w_min, vmax=w_max,
                  interpolation='nearest', cmap='seismic')

    # Remove ticks from the plot.
    ax.set_xticks([])
    ax.set_yticks([])

# Ensure the plot is shown correctly with multiple plots
# in a single Notebook cell.
plt.show()

def plot_conv_layer(layer, image):
    # Assume layer is a TensorFlow op that outputs a 4-dim tensor
    # which is the output of a convolutional layer,
    # e.g. layer_conv1 or layer_conv2.

    image = image.reshape(img_size_flat)

```

```

def plot_conv_layer(layer, image):
    # Assume layer is a TensorFlow op that outputs a 4-dim tensor
    # which is the output of a convolutional layer,
    # e.g. layer_conv1 or layer_conv2.

    image = image.reshape(img_size_flat)

    # Create a feed-dict containing just one image.
    # Note that we don't need to feed y_true because it is
    # not used in this calculation.
    feed_dict = {x: [image]}

    # Calculate and retrieve the output values of the layer
    # when inputting that image.
    values = session.run(layer, feed_dict=feed_dict)

    # Number of filters used in the conv. layer.
    num_filters = values.shape[3]

    # Number of grids to plot.
    # Rounded-up, square-root of the number of filters.
    num_grids = math.ceil(math.sqrt(num_filters))

    # Create figure with a grid of sub-plots.
    fig, axes = plt.subplots(num_grids, num_grids)

    # Plot the output images of all the filters.
    for i, ax in enumerate(axes.flat):
        # Only plot the images for valid filters.
        if i < num_filters:
            # Get the output image of using the i'th filter.
            # See new_conv_layer() for details on the format
            # of this 4-dim tensor.
            img = values[0, :, :, i]

            # Plot image.
            ax.imshow(img, interpolation='nearest', cmap='binary')

            # Remove ticks from the plot.
            ax.set_xticks([])
            ax.set_yticks([])

    # Ensure the plot is shown correctly with multiple plots
    # in a single Notebook cell.
    plt.show()

```

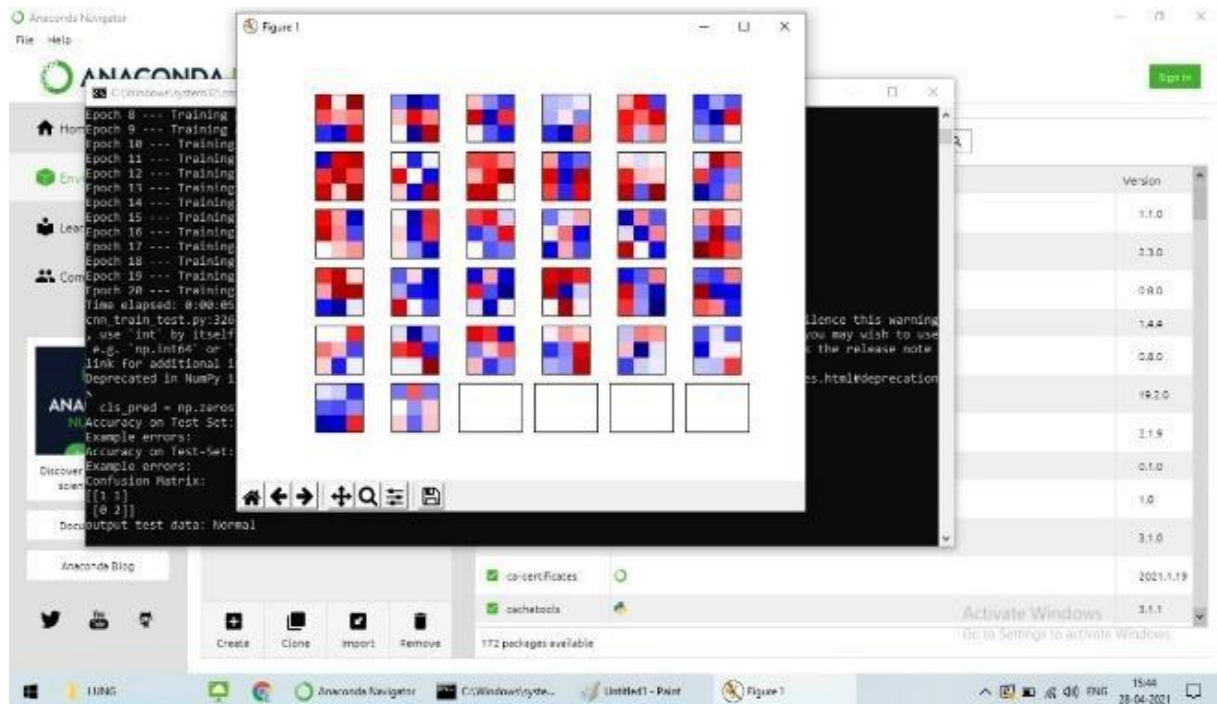
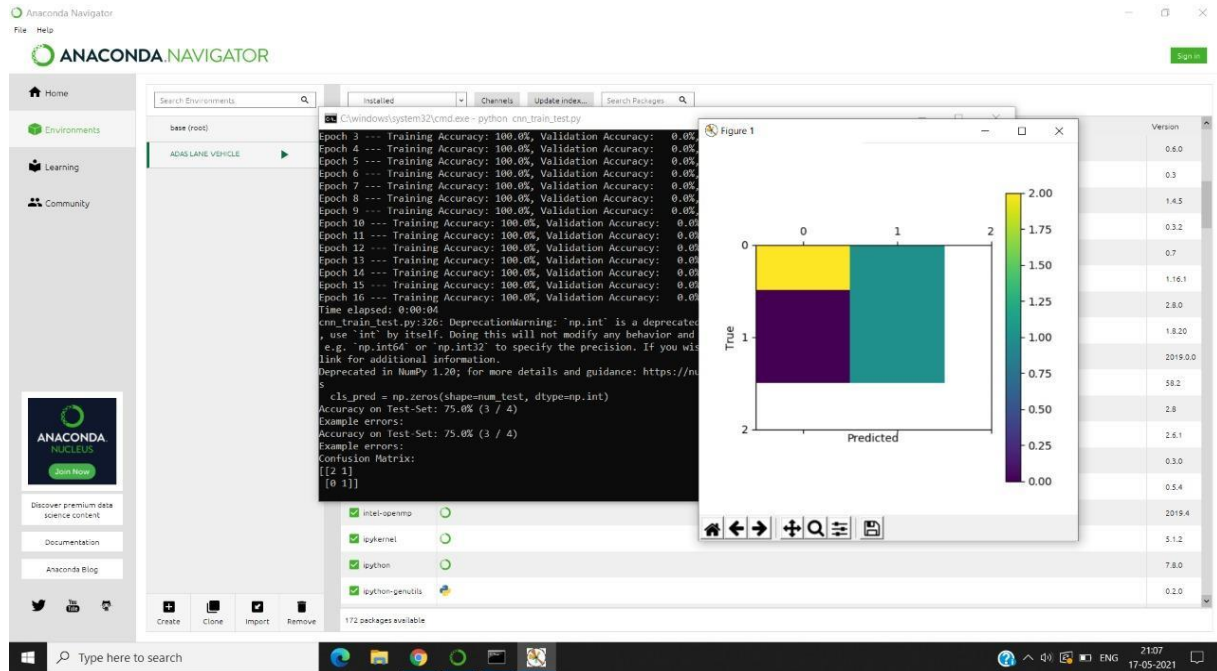
```
def plot_image(image):
    plt.imshow(image.reshape(img_size, img_size, num_channels),
                interpolation='nearest')
    plt.show()

image1 = test_images[0]
plot_image(image1)

plot_conv_weights(weights=weights_conv1)
plot_conv_layer(layer=layer_conv1, image=image1)
plot_conv_weights(weights=weights_conv2, input_channel=0)
plot_conv_layer(layer=layer_conv2, image=image1)
session.close()

cv2.waitKey(0)
cv2.destroyAllWindows()
```


7.2 SCREENSHOTS



ANACONDA NAVIGATOR

File Help

C:\anaconda\python\python.exe - python: cnn_train_test.py

Epoch 8 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 9 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 10 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 11 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 12 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 13 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 14 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 15 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 16 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 17 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 18 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 19 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Epoch 20 --- Training Accuracy: 100.0%, Validation Accuracy: 0.0%, Validation Loss: 11.14
Time elapsed: 0:00:05
cnn_train_test.py:320: UserWarning: Using a single loop for training is deprecated in NumPy 1.16.0 and will be removed in NumPy 1.17.0. Use np.nditer instead.
cls_pred = np.zeros(1000)
Accuracy on Test Set: 0.0
Example errors:
Accuracy on Test-Set:
Example errors:
Confusion Matrix:
[[1 1]
 [0 2]]
Decoded output test data: Normal

time


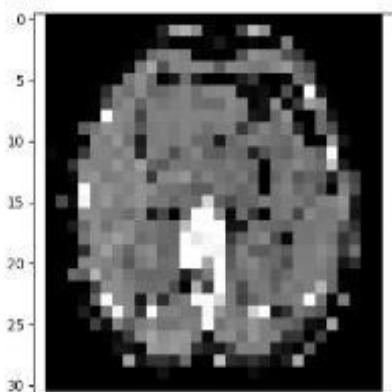


Figure 1



co-certificates
cachetools
172 packages available

Create Clone Import Remove

Activate Windows
Go to Settings to activate Windows.

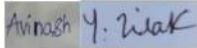
15:42
28-04-2021

CHAPTER 8

PLAGARISM REPORT

Format - I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY <small>(Deemed to be University u/s 3 of UGC Act, 1956)</small>		
Office of Controller of Examinations		
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES (To be attached in the dissertation/ project report)		
1	Name of the Candidate (IN BLOCK LETTERS)	C. Nikhil Reddy , G. MUNI REDDY
2	Address of the Candidate	J-1504, Aparna Sarovar, Nallagandla Hyderabad, Telangana, 500046 Jonnada, Alimuri Mandal, East Godavari Andhra Pradesh , 533233 Mobile Number : 9100382620, 9533699997
3	Registration Number	RA1711003011013, RA1711003011033
4	Date of Birth	9-11-1999 07-03-1999
5	Department	Computer Science
6	Faculty	P.Velmurugan
7	Title of the Dissertation/Project	Lung Disease Pattern using ANN-BPN
8	Whether the above project/dissertation is done by	Individual or group : group (Strike whichever is not applicable) a) If the project/ dissertation is done in group, then how many students together completed the project : 2 b) Mention the Name & Register number of other candidates : C.Nikhil Reddy RA1711003011013
9	Name and address of the Supervisor / Guide	Dr P.Velmurugan velmurup1@gmail.com 8171416311 Mail ID : Mobile Number :
10	Name and address of the Co-Supervisor / Co- Guide (if any)	Mail ID : Mobile Number :

11	Software Used	Anaconda		
12	Date of Verification	28-05-2021		
13	Plagiarism Details: (to attach the final	report from the software)		
Chapter	Title of the Chapter	percentage of similarity index (including self citation)	percentage of similarity index (excluding self citation)	% of plagiarism after excluding Quotes, Bibliography, etc..
1	Introduction	<1	<1	<1
2	Literature Survey	<1	0	0
3	System Architecture	<1	<1	0
4	System Requirement	<1	0	0
5	Methodology	<1	<1	0
6	Result And Discussion	<1	0	0
7				
8				
9				
10				
Appendices				
I / We declare that the above information have been Our Knowledge.		verified and found true to the best of my /		
 Signature of the Candidate		Name & Signature of the Staff (9fho uses the plagiarism check software)		
name & Signature of the Supervisor /Guide		Name & Signature of the Co-Supervisor/Co-Guide		
Name & Signature of the HOD				

ORIGINALITY REPORT

8%

SIMILARITY INDEX

7%

INTERNET SOURCES

4%

PUBLICATIONS

3%

STUDENT PAPERS

PRIMARY SOURCES

1

www.slideshare.net

Internet Source

2%

2

media.neliti.com

Internet Source

1%

3

www.arpnjournals.com

Internet Source

1%

4

www.cse.unr.edu

Internet Source

1%

5

www.ijert.org

Internet Source

1%

6

www.ijetr.org

Internet Source

1%

7

"Disease Identification in Maize Plant Leaf",
International Journal of Innovative Technology
and Exploring Engineering, 2020

Publication

1%

8

ijetsr.com

Internet Source

<1%

Submitted to Chicago State University

9

Student Paper

<1%

10

Logesh Kumar, S., M. Swathy, S. Sathish, J.
Sivaraman, and M. Rajasekar. "Identification
of Lung Cancer Cell using Watershed
Segmentation on CT Images", Indian Journal
of Science and Technology, 2016.

Publication

<1%

11

"Brain Tumour Segmentation Based on SFCM
using Back Propagation Neural Network",
International Journal of Innovative Technology
and Exploring Engineering, 2019

Publication

<1%

12

ijarcsse.com

Internet Source

<1%

13

C. Senthilkumar, R. K. Gnanamurthy. "A Fuzzy
clustering based MRI brain image

<1%