

EFFICIENT QUALITY ASSESSMENT OF BRICKS USING ARTIFICIAL NEURAL NETWORK

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering degree in Electronics and Communication Engineering

By

GUNASEKARAN K (40130074)

GIRITHAR G B (40130066)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-1 UNIVERSITY BY UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

APRIL 2024



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
[DEEMED TO BE UNIVERSITY]**

Accredited A++ Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **K. Gunasekaran (40130074)** and **G.B. Girithar (40130066)** who carried out the Project entitled **“Efficient Quality Assessment of Bricks using Artificial Neural Network”** under my supervision from November 2023 to April 2024.

**Internal Guide
Dr. M. S. GODWIN PREMI M.E., Ph.D.**

**Head of the Department
Dr. T. RAVI, M.E., Ph.D.**

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **K. Gunasekaran (40130074)**, **G.B. Girithar (40130066)** hereby declare that the project entitled “**Efficient Quality Assessment of Bricks using Artificial Neural Networks**” is done by us under the guidance of **Dr.M.S.Godwin Premi, M.E., Ph.D.**, Professor, Department of Electronics and Communication Engineering, Sathyabama Institute of Science And Technology, Chennai is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr.N.M. Nandhitha, M.E., Ph.D.**, Dean, School of Electrical and Electronics and **Dr. T. Ravi, M.E., Ph.D.**, Head of the Department, Department of Electronics and Communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide, **Dr.M.S. Godwin Premi, M.E., Ph.D.**, Professor, Department of Electronics and Communication Engineering for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Now a days deep learning plays a vital role in quality assessment. In this project, we propose a novel efficient deep learning technique to identify the quality of bricks used for construction. Review of literature shows that the binary classification of bricks is done using convolution neural networks and support vector machines with an accuracy of 86%. Our novel technique will classify the bricks into 3 classes, good quality, medium quality and bad quality with an accuracy of classification more than 86%. We planned to use the true colour image of the bricks and consider the presence of cracks and holes for assessing the quality

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	
	LIST OF FIGURES	vi
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	EXISTING SYSTEM	7
	3.1 Proposed System	8
	3.2 Block Diagram	9
4	METHODS AND ALGORITHM	11
	4.1 Artificial Neural Network	11
	4.2 Basic Neural Network	12
	4.2.1 input Layer	14
	4.2.2 Hidden Layer	14
	4.2.3 Output Layer	15
5	RESULTS AND PERFORMANCE ANALYSIS	24
	5.1 Program code	24
	5.2 Features that determine the quality	27
	5.3 How to train neural network with tensorflow	29
6	CONCLUSION	40
7	REFERENCES	41

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Brick	8
3.2	Brick Test	9
3.3	Block Diagram	10
4.1	Block Diagram Basic Neural Network	12
4.2	Source	16
4.3	RGB Source	17
4.4	Artificial Neural Network	18
4.5	Output Layer	20
4.6	Output	20
4.7	Flow	22
4.8	Feedback	22
5.1	Analysis	30
5.2	Code 1	34
5.3	Code 2	35
5.4	Efficient Quality assessment of bricks using ANN	37
5.5	Confusion Matrix	38
5.6	Probability of Quality	39

CHAPTER 1

INTRODUCTION

Bricks belong to the oldest building elements in the world. The earliest findings of bricks can be dated to 7000 BC in southern Turkey, near the city of Jericho. Ancient bricks were made by hardening mud bricks in sunlight, in the warm regions of the world. The usage of fired bricks started in about 3500 BC, which also made it possible to manufacture bricks in cooler climate regions [1]. The objective of this thesis was to examine the quality of masonry bricks, with an appropriate machine learning model. The topic of this thesis was suggested by a local engineering company, which received a commission from a brick factory to improve the defect detection of masonry bricks. The plan was to implement quality inspection from high quality images, captured by a machine vision system. The research questions were firstly to examine whether a quality inspection can be conducted using machine learning. Secondly, if a feasible model was found, the challenge was to create a useful application. Also, guidelines for appropriate imaging setup were to be planned, to be able to collect images for inspection. In factory facilities, masonry bricks are inspected manually by an operator and then divided into three different categories. There is growing interest in upgrading the automation level in the brick factory to enable more efficient production. In the factory in question, there are over 10 brick color options to choose from. Additionally, the brick size varies, and the outer surface can have patterns. One brick type was chosen for examination. The plan was to train the learning model using one brick type and then, if successful, it would be easy to train other types with the same model. The early plan for the final implementation is illustrated in Figure 1. The configuration consists of robots, conveyor belts and an imaging system on the production line. A robot arm would pick a single brick at a time from the pile, as it is discharged from the oven. This is followed by an imaging setup including line scan cameras, where bricks are imaged from three directions. The three images are combined into one image, which is processed by a learning algorithm. Eventually, the brick is sent to the correct conveyor, according to the brick's classification result.

CHAPTER 2

LITERATURE SURVEY

2.1 CLASSIFICATION OF UNDERLYING TRAINING SET

Ali A, Shamsuddin SM, Ralescu AL (2015) Classification with class imbalance problem: a review. *Int J Adv Soft Compute Appl* 7:176–204. Most existing classification approaches assume the underlying training set is even distributed. In class imbalanced classification, the training set for one class (majority) far surpassed the training set of the other class (minority), in which, the minority class is often the more interesting class. In this paper, we review the issues that come with learning from imbalanced class data sets and various problems in class imbalance classification. A survey on existing approaches for handling classification with imbalanced datasets is also presented. Finally, we discuss current trends and advancements which potentially could shape the future direction in class imbalance learning and classification. We also found out that the advancement of machine learning techniques would mostly benefit the big data computing in addressing the class imbalance problem which is inevitably presented in many real-world applications especially in medicine and social media.

2.2 SURVEY ON DEEP LEARNING APPROACHES

Alom MZ, Taha TM, Yakopcic C et al (2018) The history began from AlexNet: a comprehensive survey on deep learning approaches. *arXiv*: Deep learning has demonstrated tremendous success in variety application domains in the past few years. This new field of machine learning has been growing rapidly and applied in most of the application domains with some new modalities of applications, which helps to open new opportunity. There are different methods have been proposed on different category of learning approaches, which includes supervised, semi-supervised and unsupervised learning. The experimental results show the convenient form of the abstract and the state-of-the-art performance.

2.3 MACHINE LEARNING FOR PREDICTING PROPERTIES

Alqahtani N, Alzubaidi F, Armstrong RT (2020) Machine learning for predicting properties of porous media from 2d X-ray images. Segmentation of 3D micro-Computed Tomographic (μCT) images of rock samples is essential for further Digital Rock Physics (DRP) analysis, however, conventional methods such as thresholding, watershed segmentation, and converging active contours are susceptible to user-bias. Deep Convolutional Neural Networks (CNNs) have produced accurate pixelwise semantic segmentation results with natural images and μCT images; however, physical accuracy is not well documented. The use of deep learning over traditional machine learning approaches in the field of Image Processing, Computer Vision, Speech Recognition, Machine Translation, Art, Medical imaging, medical information processing, Robotics and control, Bio-informatics, Natural Language Processing (NLP), Cyber security, and many more. This report presents a Brief survey development of DL approaches, including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), Auto-Encoder (AE) Deep Belief Network (DBN), Generative Adversarial Network (GAN), and Deep Reinforcement (DRL). In addition, we have included recent development of proposed advanced variant DL techniques based on the mentioned DL approaches. Furthermore, DL approaches have explored and evaluated in different application domains are also included in this survey. We have also comprised recently developed frameworks, SDKs, and benchmark datasets that are used for implementing and evaluating deep learning approaches. There are some surveys have published on Deep Learning in Neural Networks [1, 38] and a survey on RL [234]. However, those papers have not discussed the individual advanced techniques for training large scale deep learning models and the module recently developed method of generative models. performance of 4 CNN architectures is tested for 2D and 3D cases in 10 configurations permeabilities and connectivity orders of magnitude off. A

new network architecture is also introduced. Manually segmented uCT images of Mt. Simon Sandstone are treated as ground truth and used as training and validation data, with a high voxel wise accuracy (over 99%) achieved. Downstream analysis is then used to validate physical accuracy. The topology of each segmented phase is calculated, and the absolute permeability and multiphase flow is modelled with direct simulation in single and mixed wetting cases. These physical measures of connectivity, and flow characteristics show high variance and uncertainty, with models that achieve 95%+ in voxel wise accuracy possessing d as a hybrid fusion of U-net and resnet, combining short and long skip connections in a Network-in-Network configuration.

2.4 BRAIN MR IMAGE SEGMENTATION USING CNN

Bao S, Chung ACS (2018) Multi-scale structured CNN with label consistency for brain MR image segmentation. *Compute Methods Biomech Biomed Eng Imaging vis* 6:113–117. In this paper, a novel method for brain MR image segmentation has been proposed, with deep learning techniques to obtain preliminary labelling and graphical model to produce the result. A specific architecture, namely multi-scale structured convolutional neural networks (MS-CNN), is designed to capture discriminative features for each sub-cortical structure and to generate a label probability map for the target image. Due to complex background in brain images and the lack of constraints among testing samples, the initial result obtained with MS-CNN is not smooth. To deal with this problem, dynamic random walker with decayed region interest is then proposed to enforce label consistency. Comprehensive evaluations have been carried out on two publicly available data-sets and experimental results indicate that the proposed method can obtain better segmentation quality efficiently.

2.5 BRICK DEFECT DETECTION AND CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

By A. E. Sadik and A. T. Sadiq. This paper proposes a CNN-based approach for detecting and classifying defects in bricks, such as cracks, chips, and color variations. It discusses the use of transfer learning and fine-tuning techniques to leverage pre-trained CNN models, as well as the use of data augmentation and feature visualization techniques to improve the model's performance.

2.6 AUTOMATIC BRICK DEFECT RECOGNITION USING CONVOLUTIONAL NEURAL

By H. Sezgin, et al. This paper presents a CNN-based system for automatically detecting defects in bricks, such as cracks, holes, and shape deformations. It discusses the design of the CNN architecture, including the use of multiple convolutional and pooling layers, and the optimization of hyperparameters. It also presents experimental results and performance evaluation of the proposed system.

2.7 DEEP LEARNING FOR QUALITY INSPECTION OF INDUSTRIAL BRICKS

By P. Faustino, et al. This paper presents a CNN-based approach for quality inspection of industrial bricks, specifically focusing on detecting surface defects, such as cracks, holes, and color variations. It discusses the use of different CNN architectures, including VGG and ResNet, and compares their performance in terms of accuracy and computational efficiency. It also presents a detailed analysis of the dataset used for training and evaluation. convolutional neural network (CNN). Although core is more representative of reservoir lithology than well log data, employing machine learning for automatic lithology identification from core is limited

convolutional neural network (CNN). Although core is more representative of reservoir lithology than well log data, employing machine learning for automatic lithology identification from core is limited. A CNN-based approach is developed to classify core images into three lithologies: sandstone, limestone, and shale. The model is based on ResNeXt-50 architecture, which had better performance in lithology prediction than ResNet-18 and Inception-v3 architectures. Our method processes core tray images and outputs lithology logs in a fully automated procedure and predicts the lithologies.

2.8 Automatic Visual Inspection of Clay Bricks using Convolutional Neural Networks.

This paper presents a CNN- based system for automatic visual inspection of clay bricks, including the detection of surface defects, such as cracks, holes, and roughness. It discusses the design of the CNN architecture, including the use of different convolutional and pooling layers, and the optimization of hyperparameters. It also presents experimental results and performance evaluation of the proposed system

CHAPTER 3

EXISTING SYSTEM

- In order to remove noises such as irregularly illuminated conditions, shading and a blemish, we propose two preprocessing for automatic crack detection. The proposed method consists of two pre-processing and one processing.
- First, the subtraction pre-processing is used to remove slight variation like irregularly illuminated conditions and shading.
- Secondly, the line emphasis pre-processing is used to emphasize line structures associated with cracks against blob structures such as blemish. After two pre-processing, thresholding processing is used to separate cracks from the background.
- In proposed system order to remove cracks such as irregularly illuminated conditions, shading and a blemish, we propose two pre processing for automatic crack detection. The proposed method consists of two pre-processing and one processing. First, the subtraction pre-processing is used to remove slight variation like irregularly illuminated conditions and shading. Secondly, the line emphasis pre-processing is used to emphasize line structures associated with cracks against blob structures such as blemish. After two pre-processing, thresholding processing is used to separate cracks from the background. In existing system, they used to find the crack detection through convolutional neural networks.

Disadvantages

- This paper proposed a method to detect cracks from images using convolutional neural networks.
- The Ann architecture of binary-class output for crack detection was designed through modifying the AlexNet. 1455 images with 4160×3120 -pixel resolutions captured by a smartphone were used to train, validate, and test the ANN where all of the images were taken from real concrete surfaces.

- To build the training set and validation set, 1250 images were cropped into 60000 smaller images of 256×256 -pixel resolutions

3.1 ANN

- In the proposed system we are going to quality check of bricks by ANN (artificial neural network).
- artificial neural network-based solution, using a pretrained VGG-16 deep learning architecture, is proposed.
- Overall accuracy of 88 % was obtained when considering all three quality classes. When only discarding class 3 bricks, i.e., those that are not suitable for any construction work, the accuracy was 93 %.

Advantages

- ANNs, a type of network architecture of deep learning algorithms, are especially used for image recognition and tasks that involve processing pixel data. There are other types of neural networks in deep learning, but ANNs are the best network architecture for identifying and recognizing objects.

Different Types of Tests on Bricks

1. Water Absorption Test on Bricks

Water absorption test for bricks can be performed by using two methods,

(i) 24-hour immersion cold water test

(ii) 5-hour boiling test

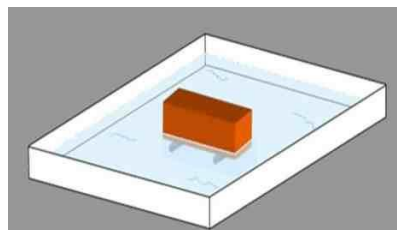


Fig:3.1 brick

2. Efflorescence Test

- Soluble salt available in brick causes efflorescence. The presence of such salts can be determined as follows.
- Place one end of the bricks in a 25 mm depth of water in a dish with a minimum diameter of 150 mm and depth of 30 mm.
- The dish is made of porcelain, glass, or of glazed stonework.
- The experiment is performed in a well-ventilated room between 20 to 30 degrees C till all the water in the dish is either absorbed by the specimen or evaporated.
- After the specimen has dried, add a similar quantity of water to the dish and let it be absorbed by the specimen for efflorescence after the second evaporation.



Fig:3.2 brick test

3. Hardness Test on Brick

The Hardness Test of Brick can be judged with the help of the scratch of the fingernail. Try to scratch the figure on the brick, if no scratch is left on the surface of the brick, it is considered to be having sufficient hardness.

3.2 FLOW CHART

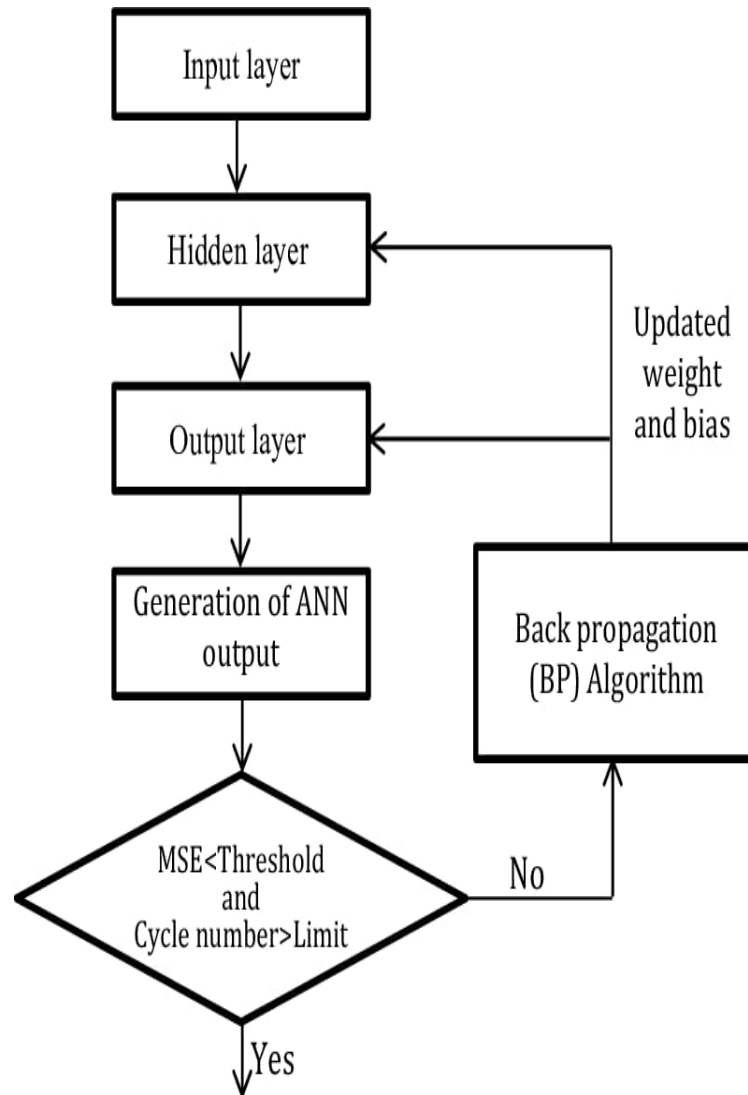


Fig:3.3 Flow Chart

CHAPTER 4

METHODS AND ALGORITHM

4.1 NEURAL NETWORKS

Neural Networks are multi-layered networks consisting of neurons or nodes. These neurons are the core processing units of the neural network. They are designed to act like human brains. They take in data, train themselves to recognize the patterns in the data and then predict the output.

A basic neural network has three layers:

1. Input layer
2. Hidden layer
3. Output layer

The working mechanism of Artificial Neural Network

Artificial Neural Networks work in a way similar to that of their biological inspiration. They can be considered as weighted directed graphs where the neurons could be compared to the nodes and the connection between two neurons as weighted edges. The processing element of a neuron receives many signals (both from other neurons and as input signals from the external world).

Signals are sometimes modified at the receiving synapse and the weighted inputs are summed at the processing element. If it crosses the threshold, it goes as input to other neurons (or as output to the external world) and the process repeats.

Usually, the weights represent the interconnection strength between the neurons. The activation function is a transfer function that is used to get the desired output for the problem designed. Say, the desired output is zero or one in case of a binary classifier. Sigmoid function could be used as the activation function.

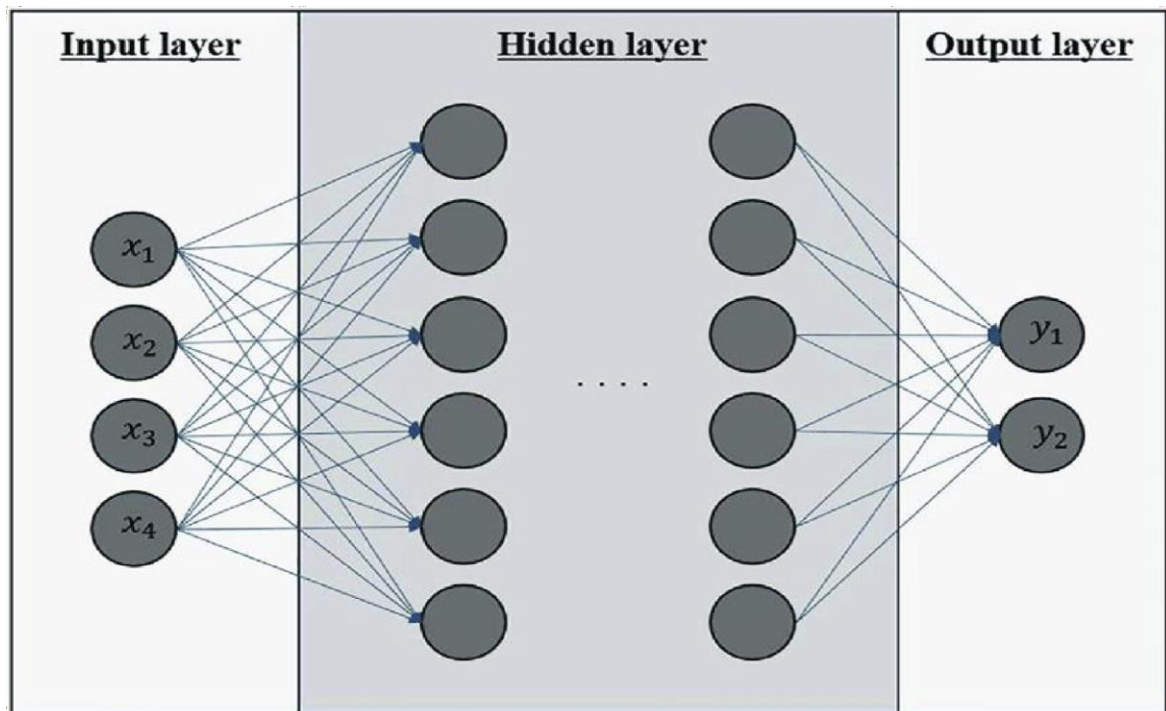


Fig:4.1: Block diagram Basic neural network

4.2 Basic neural network

The input layers receive the input, the output layer predicts the output and the hidden layers do most of the calculations. The number of hidden layers can be modified according to the requirements. There should be at least one hidden layer in a neural network.

The basic working of the neural network is as follows:

Let us consider an image, each pixel is fed as input to each neuron of the first layer. Neurons of one layer are connected to neurons of the next layer through channels. Each of these channels is assigned a numerical value known as weight.

1. The inputs are multiplied by the corresponding weights and this weighted sum is then fed as input to the hidden layer.
2. The output from the hidden layers is passed through an activation function which

will determine whether the neuron will be activated or not.

3.The activated neurons transmit data to the next hidden layers. In this manner, data is propagated through the network; this is known as Forward Propagation. In the output layer, the neuron with the highest value predicts the output. These outputs are the probability values.

4.The predicted output is compared with the actual output to obtain the error. This information is then transferred back through the network, the process is known as Backpropagation.

5. Based on this information, the weights are adjusted. This cycle of forward and backward propagation is done several times on multiple inputs until the network predicts the output correctly in most of the cases.

6.This ends the training process of the neural network. The time to train the neural network may get high in some cases.

7.In the output layer, the neuron with the highest value predicts the output. These outputs are the probability values

8.The predicted output is compared with the actual output to obtain the error. This information is then transferred back through the network; the process is known as Back propagation.

9.The predicted output is compared with the actual output to obtain the error. This information is then transferred back through the network, the process is known as Backpropagation.

10. Based on this information, the weights are adjusted. This cycle of forward and backward propagation is done several times on multiple inputs until the network predicts the output correctly in most of the cases.

Here are some guidelines to prepare data for image processing:

- More data needs to be fed to the model to get the better results.
- Image dataset should be of high quality to get more clear information, but to process them you may require deeper neural networks.
- In many cases RGB images are converted to grayscale before feeding them in neural network.

4.2.1 INPUT LAYER:

The input layer of a neural network is composed of artificial input neurons, and brings the initial data into the system for further processing by subsequent layers of artificial neurons. The input layer is the very beginning of the workflow for the artificial neural network. One of the distinct characteristics of the input layer is that artificial neurons in the input layer have a different role to play – experts explain this as the input layer being constituted of “passive” neurons that do not take in information from previous layers because they are the very first layer of the network. In general, artificial neurons are likely to have a set of weighted inputs and function based on those weighted inputs.

However, in theory, an input layer can be composed of artificial neurons that do not have weighted inputs, or where weights are calculated differently, for example, randomly, because the information is coming into the system for the first time. What is common in the neural network model is that the input layer sends the data to subsequent layers, in which the neurons do have weighted inputs.

4.2.2 HIDDEN LAYER:

A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. Hidden neural network layers are set up in many ways. In some cases, weighted inputs are randomly assigned. In other cases, they are fine-tuned and calibrated through a process called backpropagation. Either way, the artificial neuron in the hidden layer works like a biological neuron in the brain – it takes in its probabilistic input signals, works on them, and converts them into an output corresponding to the biological neuron's axon.

Many analyses of machine learning models focus on the construction of hidden layers in the neural network. There are different ways to set up these hidden layers to generate various results – for instance, convolutional neural networks that focus on image processing, recurrent neural networks

that contain an element of memory and simple feedforward neural networks that work in a straightforward way on training data sets.

4.2.3 OUTPUT LAYER:

The output layer is the final layer in the neural network where desired predictions are obtained. There is one output layer in a neural network that produces the desired final prediction. It has its own set of weights and biases that are applied before the final output is derived.

The activation function for the output layer may be different than the hidden layers based on the problem. For example, SoftMax activation is used to derive the final classes in a classification problem. The output is a vector of values that may need further post-processing to convert them to business related values. For example, in a classification problem, the output is a set of probabilities that needs to be mapped to the corresponding business classes. How do we determine the number of nodes in the output layer? It depends on the problem. In a binary classification problem, there is only 1 node that provides a probability of the positive outcome. In the case of n-class classification, there are n nodes, each producing the probability for a given class. For regression, there is only 1 node that produces the output. In this way, the number of nodes may vary based on the type of problem being solved.

Image Processing in Python: Algorithms Tools, and Methods You Should Know

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as **Image Processing**. It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

Let's get into it!

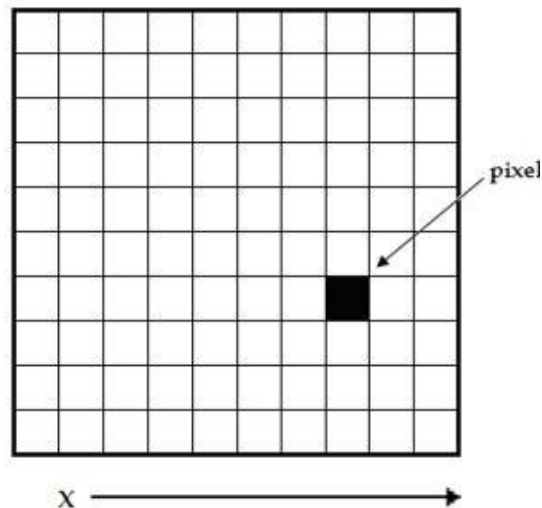


Fig:4.2: [Source](#)

The RGB color model is one of the most widely used color representation method in computer graphics. It uses a color coordinate system with three primary colors:

R(red), G(green), B(blue)

Each primary color can take an intensity value ranging from 0(lowest) to 1(highest). Mixing these three primary colors at different intensity levels produces a variety of colors. The collection of all the colors obtained by such a linear combination of red, green and blue forms the cube shaped RGB color space.

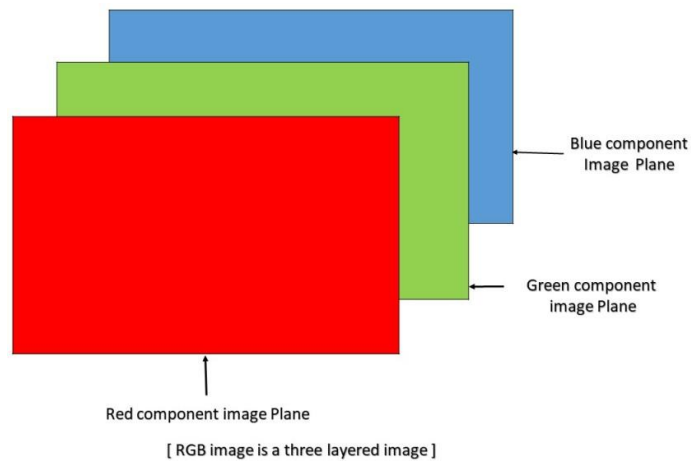


Fig:4.3: RGB *Source*

There are various types of images:

- RGB image: It contains three layers of 2D image, these layers are red, green, and blue channels.
- Grayscale image: These images contain shades of black and white and contain only a single channel.

Artificial Neural Network

Introduction to Artificial Neural Networks

Artificial Neural Networks are the most popular ***machine learning algorithms*** today. The invention of these Neural Networks took place in the 1970s but they have achieved huge popularity due to the recent increase in computation power because of which they are now virtually everywhere. In every application that you use, Neural Networks power the intelligent interface that keeps you engaged.

Block Diagram of Neural Architecture

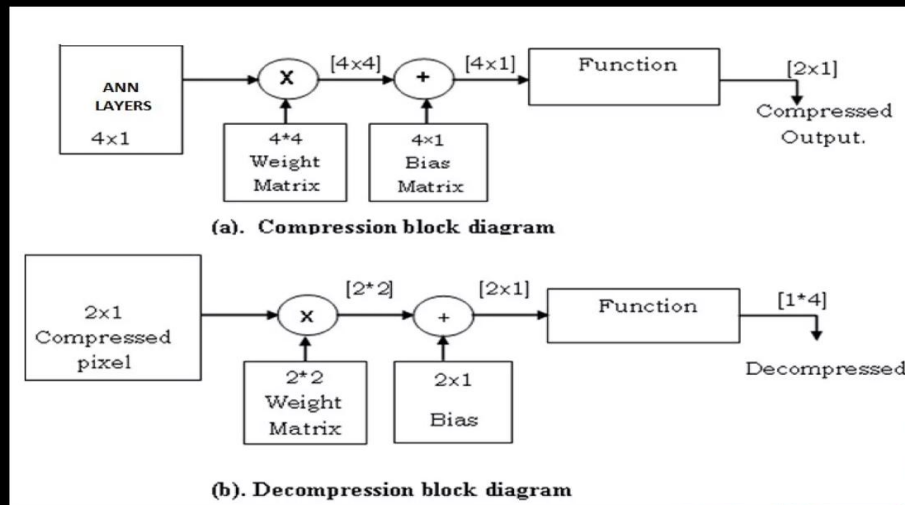


Fig:4.4: Block Diagram of Artificial Neural Network

Artificial Neural Networks are a special type of machine learning algorithms that are modeled after the human brain. That is, just like how the neurons in our nervous system are able to learn from the past data, similarly, the ANN is able to learn from the data and provide responses in the form of predictions or classifications.

ANNs are nonlinear statistical models which display a complex relationship between the inputs and outputs to discover a new pattern. A variety of tasks such as image recognition, speech recognition, machine translation as well as medical diagnosis makes use of these artificial neural networks.

An important advantage of ANN is the fact that it learns from the example data sets. Most commonly usage of ANN is that of a random function approximation. With these types of tools, one can have a cost-effective method of arriving at the solutions that define the distribution. ANN is also capable of taking sample data rather than the entire dataset to provide the output result. With ANNs, one can enhance existing data analysis techniques owing to their advanced predictive capabilities.

Don't forget to check DataFlair's leading blog on [Recurrent Neural Networks](#).

Artificial Neural Networks Architecture

The functioning of the Artificial Neural Networks is similar to the way neurons work in our nervous system. The Neural Networks go back to the early **1970s** when **Warren S McCulloch and Walter Pitts** coined this term. In order to understand the workings of ANNs, let us first understand how it is structured. In a neural network, there are three essential layers –

Input Layers

The *input layer* is the first layer of an ANN that receives the input information in the form of various texts, numbers, audio files, image pixels, etc.

Hidden Layers

In the middle of the ANN model are the *hidden layers*. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These hidden layers perform various types of mathematical computation on the input data and recognize the patterns that are part of.

Output Layer

In the *output layer*, we obtain the result that we obtain through rigorous computations performed by the middle layer.

In a neural network, there are multiple parameters and hyperparameters that affect the performance of the model. The output of ANNs is mostly dependent on these parameters. Some of these parameters are weights, biases, learning rate, batch size etc. Each node in the ANN has some weight.

Each node in the network has some weights assigned to it. A transfer function is used for calculating the weighted sum of the inputs and the bias.

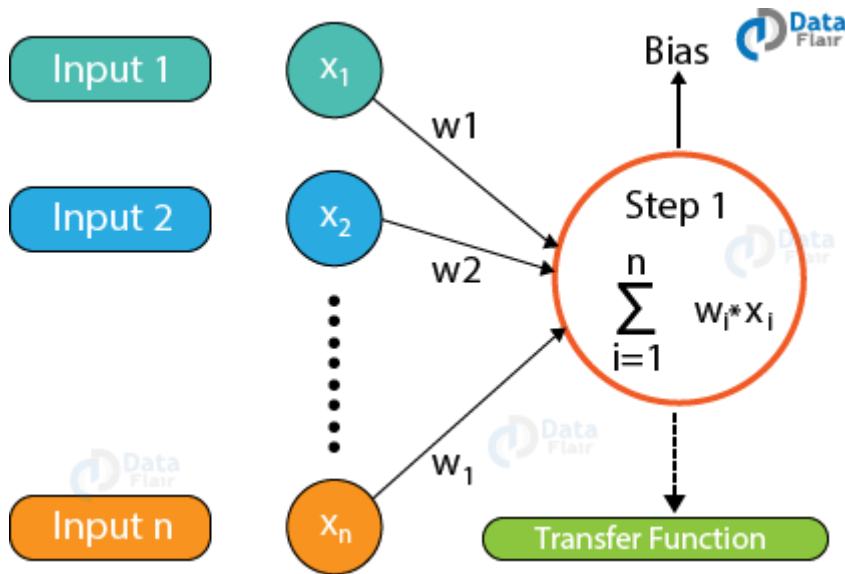


Fig:4.5: Output Layer

After the transfer function has calculated the sum, the activation function obtains the result. Based on the output received, the activation functions fire the appropriate result from the node. **For example**, if the output received is above 0.5, the activation function fires a 1 otherwise it remains 0.

Some of the popular activation functions used in Artificial Neural Networks are Sigmoid, RELU, SoftMax, tanh etc.

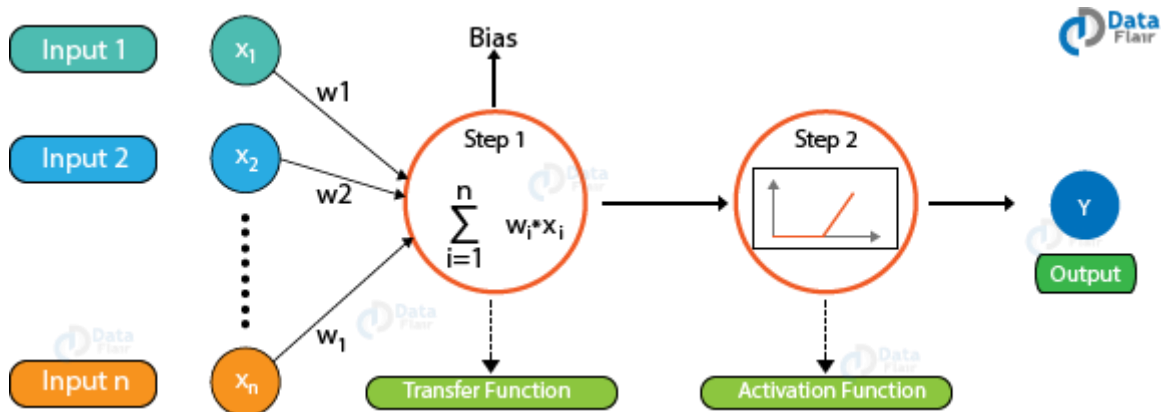


Fig:4.6: output

Based on the value that the node has fired, we obtain the final output. Then, using the error functions, we calculate the discrepancies between the predicted output and resulting output and adjust the weights of the neural network through a process known as **backpropagation**.

ANNs are part of an emerging area in Machine Learning known as Deep Learning.

Many people are confused between Deep Learning and Machine Learning. Are you among one of them? Check this easy-to-understand article on [Deep Learning vs Machine Learning](#).

Back Propagation in Artificial Neural Networks

In order to train a neural network, we provide it with examples of input-output mappings. Finally, when the neural network completes the training, we test the neural network where we do not provide it with these mappings. The neural network predicts the output and we evaluate how correct the output is using the various error functions. Finally, based on the result, the model adjusts the weights of the neural networks to optimize the network following gradient descent through the chain rule.

Types of Artificial Neural Networks

There are two important types of Artificial Neural Networks –

- Feedforward Neural Network
- Feedback Neural Network

Feedforward Artificial Neural Networks

In the feedforward ANNs, the flow of information takes place only in one direction. That is, the flow of information is from the input layer to the hidden layer and finally to the output. There are no feedback loops present in this neural network. These types of neural networks are mostly used in ***supervised learning*** for instances such as classification, image recognition etc. We use them in cases where the data is not sequential in nature.

Flow of information

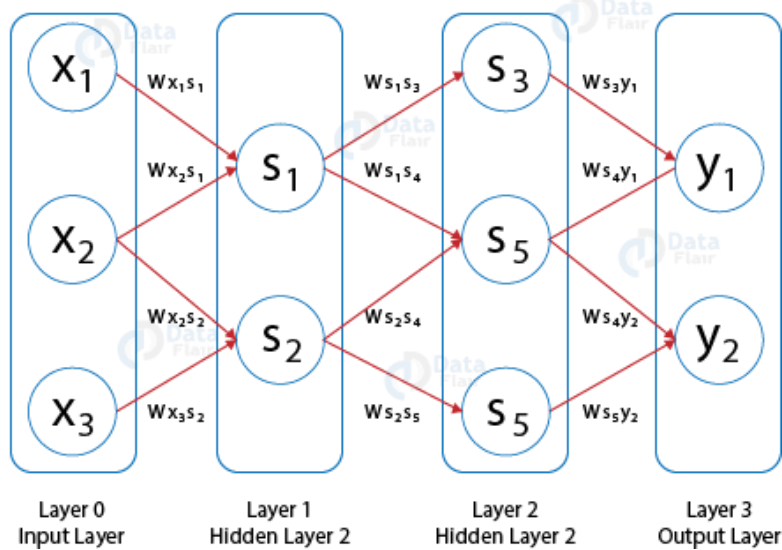


Fig:4.7 Flow

Feedback Artificial Neural Networks

In the feedback ANNs, the feedback loops are a part of it. Such type of neural networks are mainly for memory retention such as in the case of recurrent neural networks. These types of networks are most suited for areas where the data is sequential or time-dependent.

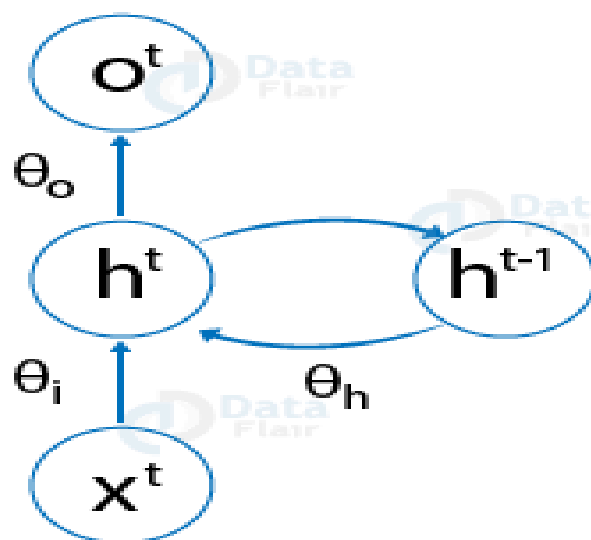


Fig:4.8: Feedback

Bayesian Networks

These types of neural networks have a probabilistic graphical model that makes use of Bayesian Inference for computing the probability. These types of Bayesian Networks are also known as Belief Networks. In these Bayesian Networks, there are edges that connect the nodes representing the probabilistic dependencies present among these types of random variables. The direction of effect is such that if one node is affecting the other then they fall in the same line of effect. Probability associated with each node quantifies the strength of the relationship. Based on the relationship, one is able to infer from the random variables in the graph with the help of various factors.

The only constraint that these networks have to follow is it cannot return to the node through the directed arcs. Therefore, Bayesian Networks are referred to as Directed Acyclic Graphs (DAGs).

These **Bayesian Networks** can handle the multivalued variables and they comprise of two dimensions –

- Range of Propositions
- Probability that each proposition has been assigned with.

Assume that there is a finite set of random variables such that each variable of the finite set is denoted by $X = \{x_1, x_2 \dots x_n\}$ where each variable X takes from the values present in the finite set such that $\text{value}\{x_1\}$. If there is a directed link from the variable X_i to the variable X_j , then X_i will be the parent of X_j that shows the direct dependencies between these variables.

With the help of Bayesian Networks, one can combine the prior knowledge as well as the observed data. Bayesian Networks are mainly for learning the causal relationships and also understanding the domain knowledge to predict the future event. This takes place even in the case of missing data.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 PROGRAM CODE 1:

```
import streamlit as st
import numpy as np
import functions
from PIL import Image

# App Title
st.title("Efficient Quality assessment of bricks using ANN")

# Introduction text
st.markdown(unsafe_allow_html=True, body="<p>Welcome to quality checking
of bricks APP.</p>"

            "<p>This is a basic app built with Streamlit."

            "With this app, you can upload the bricks image
whethere the quality is good or not.</p>"

            "<p>The model used is a Artificial Neural Network
(ANN) and in this "

            "moment has a test accuracy of "

            "<strong>90.7%.</strong></p>")

st.markdown("First, let's load an bricks image.")
```

```

# Loading model

# Img uploader
img = st.file_uploader(label="Load bricks image", type=['jpeg', 'jpg', 'png'],
key="xray")

if img is not None:
    # Preprocessing Image
    p_img = functions.preprocess_image(img)

    if st.checkbox('Zoom image'):
        image = np.array(Image.open(img))
        st.image(image, use_column_width=True)
    else:
        st.image(p_img)

# Loading model
loading_msg = st.empty()
loading_msg.text("Predicting...")
model = functions.load_model()

# Predicting result
prob, prediction = functions.predict(model, p_img)

loading_msg.text("")

if prediction:
    st.markdown(unsafe_allow_html=True, body="<span style='color:red; font-
size: 50px'><strong><h4>bad quality!
:slightly_frowning_face:</h4></strong></span>")
else:

```

```
st.markdown(unsafe_allow_html=True, body="<span style='color:green; font-size: 50px'><strong><h3>good quality!:smile: </h3></strong></span>")
```

```
st.text(f"*Probability of quality is {round(prob[0][0] * 100, 3)}%")
```

PROGRAM CODE 2:

```
import keras
import numpy as np
import streamlit as st
from keras import layers, models, optimizers # modeling
from PIL import Image

MODEL = "ann_augmented_dropout_90_7.h5"

@st.cache(allow_output_mutation=True)
def load_model():
    print("loading model")
    model = keras.models.load_model(f"model/{MODEL}", compile=True)

    return model

def preprocess_image(img):
    image = Image.open(img).convert("RGB")
    p_img = image.resize((224, 224))

    return np.array(p_img) / 255.0
```

```
def predict(model, img):  
    prob = model.predict(np.reshape(img, [1, 224, 224, 3]))  
  
    if prob < 0.8:  
        prediction = True  
    else:  
        prediction = False  
    return prob, prediction
```

5.2 Some of the features that determine the quality of our neural network are

1. Layers
2. Activation function
3. Loss function
4. Optimizer

Layers

Layers in a neural network are very important as we saw earlier an artificial neural network consists of 3 layers an input layer, hidden layer, output layer. The input layer consists of the features and values that need to be analysed inside a neural network. Basically, this is a layer that will read our input features onto an Artificial neural network.

A hidden layer is a layer where all the magic happens when all the input neurons pass the features to the hidden layer with a weight and a bias each and every neuron inside the hidden layer will sum up all the weighted features from all the input layers and apply an activation function to keep the values between 0 and 1 for easier learning. Here we need to choose the number of neurons in each layer manually and it must be the best value for the network.

Here the real decision-makers are the weights between each layer which will finally pass a value of 0 to 1 to the output layer. Till this, we have seen the importance of each level of layers in an artificial neural network. There are many types of layers in TensorFlow but the one that we will use a lot is Dense

Activation function

Activation functions are simply mathematical methods that bring all the values inside a range of 0 to 1 so that it will be very easier for the machine to learn the data in its process of analyzing the data. There are a variety of activation functions that are supported by the Tensor flow. Some of the commonly used functions are,

1. Sigmoid
2. Relu
3. Softmax
4. Swish
5. Linear

Each and every activation function has its own specific use cases and drawbacks. But the activation function that is used in the hidden and input layers is 'Relu' and another this which will have a greater impact on the result is losses.

After this, we can see the params in the compilation of the model in TensorFlow,

Losses

Loss functions are a very important thing to notice while creating a neural network because loss functions in the neural network will calculate the difference between the predicted output and the actual result and greatly help the optimizers in the neural nets to update the weights on its backpropagation.

There are many loss functions that were supported by the TensorFlow library, and again commonly used few are,

1. Mean Absolute
2. MeanSquaredError
3. Binary Crossentropy
4. Categorical Crossentropy
5. Sparse Categorical Crossentropy

Note: Again, the choice of loss is completely dependent on the type of problem and result we expect from the Neural network.

Optimizers

Optimizers are a very important thing because this is the function that helps the neural network to change the weights on the backpropagation so that the difference between the actual and predicted result will decrease at a gradual pace and obtain that point where the loss is very minimum and the model is able to predict more accurate results.

Again TensorFlow supports many optimizers to mention a few,

1. Gradient descent
2. SDG – Stochastic Gradient Descent
3. Adagrad
4. Adam

After compiling the model, we need to fit the model with the dataset for training,

5.3How to Train a Neural Network with TensorFlow:

Step 1: Importing the libraries

We are going to import the required libraries.

```
# Importing the libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
from tensorflow import keras
```

```
from tensorflow.keras import layers
```

```
from sklearn.model_selection import train_test_split
```

Step 2: Importing the data

The data we used for this example are generated randomly with Numpy. In this data x and y are the point of coordinates and the color feature is the target value that was generated randomly which is in binary representing Red – 1 , Blue – 0.

The data will look like

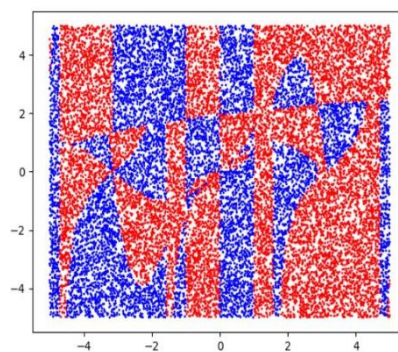


Fig: 5.1 Analysis

Step 3: Splitting the data

Now we are going to split the dataset into train and test splits to evaluate the model with the unseen data and check its accuracy.

```
# split the data into train and test set
```

```
train, test = train_test_split(
```

```
    df, test_size=0.2, random_state=42, shuffle=True)
```

Step 4: Constructing the input

In this step, we are going to construct the input we need to feed into a network. For simplicity and for the model's sake we are going to stack the two features of the data into x and the target variable as y. We use `numpy.column_stack()` to stack the

```
# Constructing the input
x = np.column_stack((train.x.values, train.y.values))
y = train.color.values
```

Step 5: Building a model

Now we are going to build a simple neural network to classify the color of the point with two input nodes and a hidden layer and an output layer with relu and sigmoid activation functions, and sparse categorical cross-entropy loss function and this is going to be a **fully connected feed-forward network**

```
# Defining the model

model = keras.Sequential([

    keras.layers.Dense(4, input_shape=(2,), activation='relu'),

    keras.layers.Dense(2, activation='sigmoid')

])

# Compiling the model

model.compile(optimizer='adam',

              loss=keras.losses.SparseCategoricalCrossentropy(),

              metrics=['accuracy'])
```

```
# fitting the model

model.fit(x, y, epochs=10, batch_size=8)
```

Step 6: Building a better model

Now we are going to improve the model with a few extra hidden layers and a better activation function 'softmax' in the output layer and built a better neural network.

```
# Defining the model

model_better = keras.Sequential([

    keras.layers.Dense(16, input_shape=(2,), activation='relu'),

    keras.layers.Dense(32, activation='relu'),

    keras.layers.Dense(32, activation='relu'),

    keras.layers.Dense(2, activation='softmax')

])
```

```
# Compiling the model

model_better.compile(optimizer='adam',

loss=keras.losses.SparseCategoricalCrossentropy(),

metrics=['accuracy'])
```

```
# Constructing the input

x = np.column_stack((train.x.values, train.y.values))
```

```
y = train.color.values
```

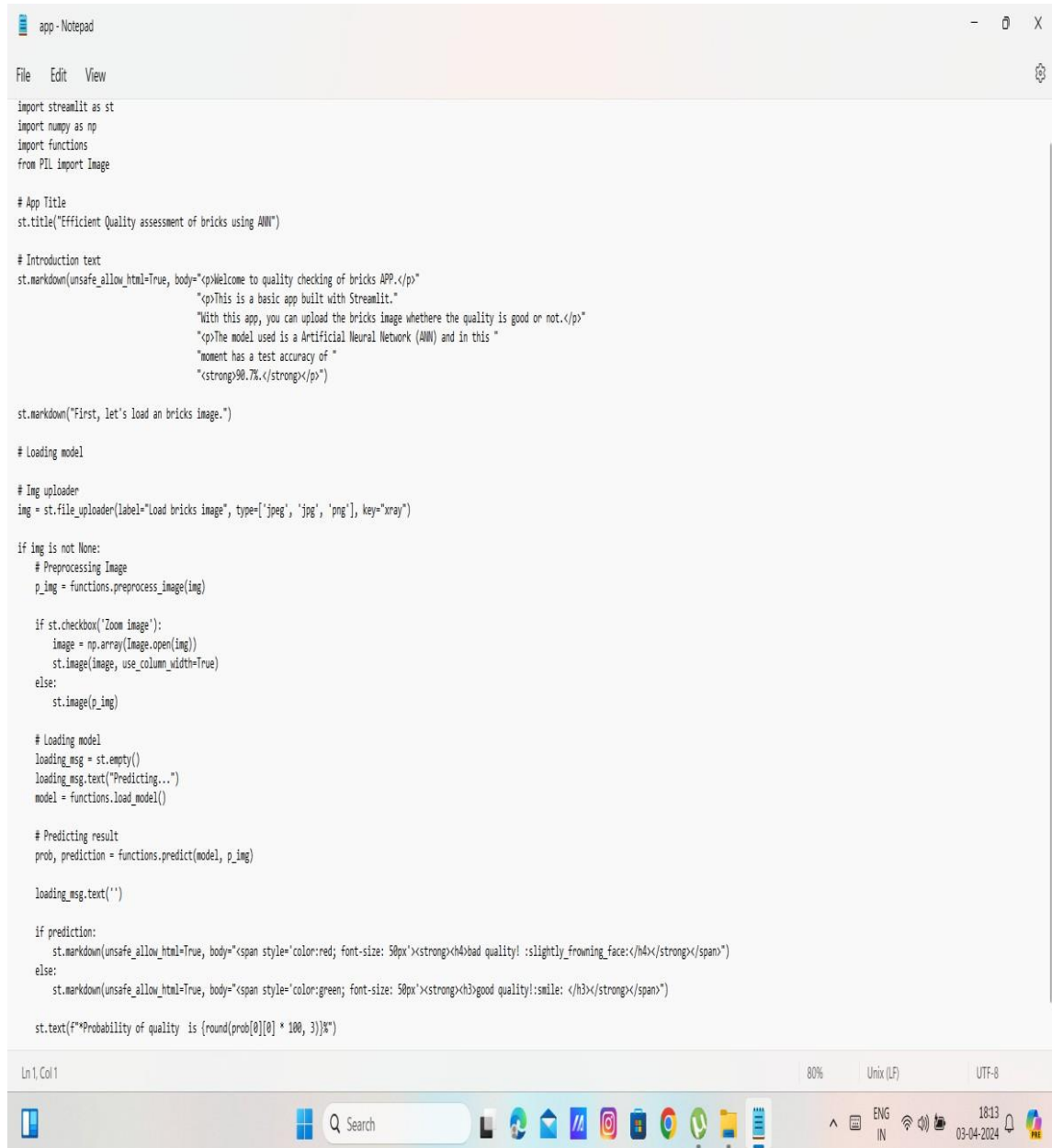
```
# fitting the model
```

```
model_better.fit(x, y, epochs=10, batch_size=8)
```

Step 7: Evaluating the model

Finally, if we evaluate the model, we can clearly see that the accuracy of the model on unseen data has been improved. So, we built an efficient model.

CODE 1:



```
app - Notepad
File Edit View
import streamlit as st
import numpy as np
import functions
from PIL import Image

# App Title
st.title("Efficient Quality assessment of bricks using ANN")

# Introduction text
st.markdown(unsafe_allow_html=True, body="
<p>Welcome to quality checking of bricks APP.</p>
<p>This is a basic app built with Streamlit.</p>
<p>With this app, you can upload the bricks image whether the quality is good or not.</p>
<p>The model used is a Artificial Neural Network (ANN) and in this
moment has a test accuracy of "
<strong>90.7%</strong></p>")

st.markdown("First, let's load an bricks image.")

# Loading model

# Img uploader
img = st.file_uploader(label="Load bricks image", type=['jpeg', 'jpg', 'png'], key="xray")

if img is not None:
    # Preprocessing Image
    p_img = functions.preprocess_image(img)

    if st.checkbox("Zoom image"):
        image = np.array(Image.open(img))
        st.image(image, use_column_width=True)
    else:
        st.image(p_img)

    # Loading model
    loading_msg = st.empty()
    loading_msg.text("Predicting...")
    model = functions.load_model()

    # Predicting result
    prob, prediction = functions.predict(model, p_img)

    loading_msg.text('')

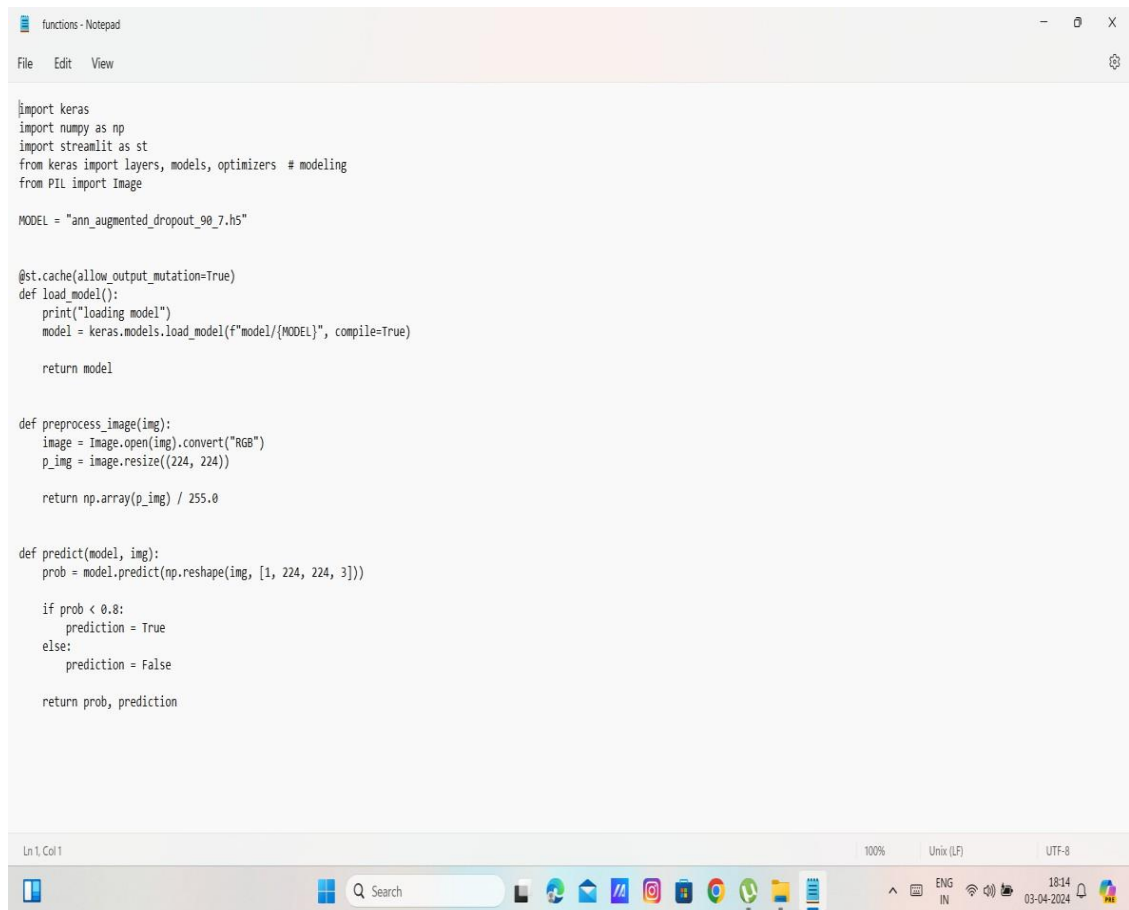
    if prediction:
        st.markdown(unsafe_allow_html=True, body="
<span style='color:red; font-size: 50px'><strong><h4>bad quality! :slightly_frowning_face:</h4></strong></span>")
    else:
        st.markdown(unsafe_allow_html=True, body="
<span style='color:green; font-size: 50px'><strong><h3>good quality!:smile: </h3></strong></span>")

    st.text(f"**Probability of quality is {round(prob[0][0] * 100, 3)}%")

Ln 1, Col 1
80% Unix (LF) UTF-8
```

Fig:5.2:Code 1

CODE 2:



```
functions - Notepad
File Edit View

import keras
import numpy as np
import streamlit as st
from keras import layers, models, optimizers # modeling
from PIL import Image

MODEL = "ann_augmented_dropout_90_7.h5"

@st.cache(allow_output_mutation=True)
def load_model():
    print("loading model")
    model = keras.models.load_model(f"model/{MODEL}", compile=True)

    return model

def preprocess_image(img):
    image = Image.open(img).convert("RGB")
    p_img = image.resize((224, 224))

    return np.array(p_img) / 255.0

def predict(model, img):
    prob = model.predict(np.reshape(img, [1, 224, 224, 3]))

    if prob < 0.8:
        prediction = True
    else:
        prediction = False

    return prob, prediction

Ln 1, Col 1 100% Unix (LF) UTF-8
```

Fig:5.3: Code 2

OUTPUT:

Step 1: Type `python.main.py` in cmd in bricks folder and the chapter will open and give `import data` and give `import data` after that data will get imported

Step 2: After that total training bricks will get trained and showed in the output screen

Step 3: It will check and take the inputs from the datasets.

Step 4: And give pre -process data and after preprocessing overall datasets will get processed

Step 5: And finally give train data to give train all type of data and every time we take the sample pictures and added into the datasets and follow the same procedure.

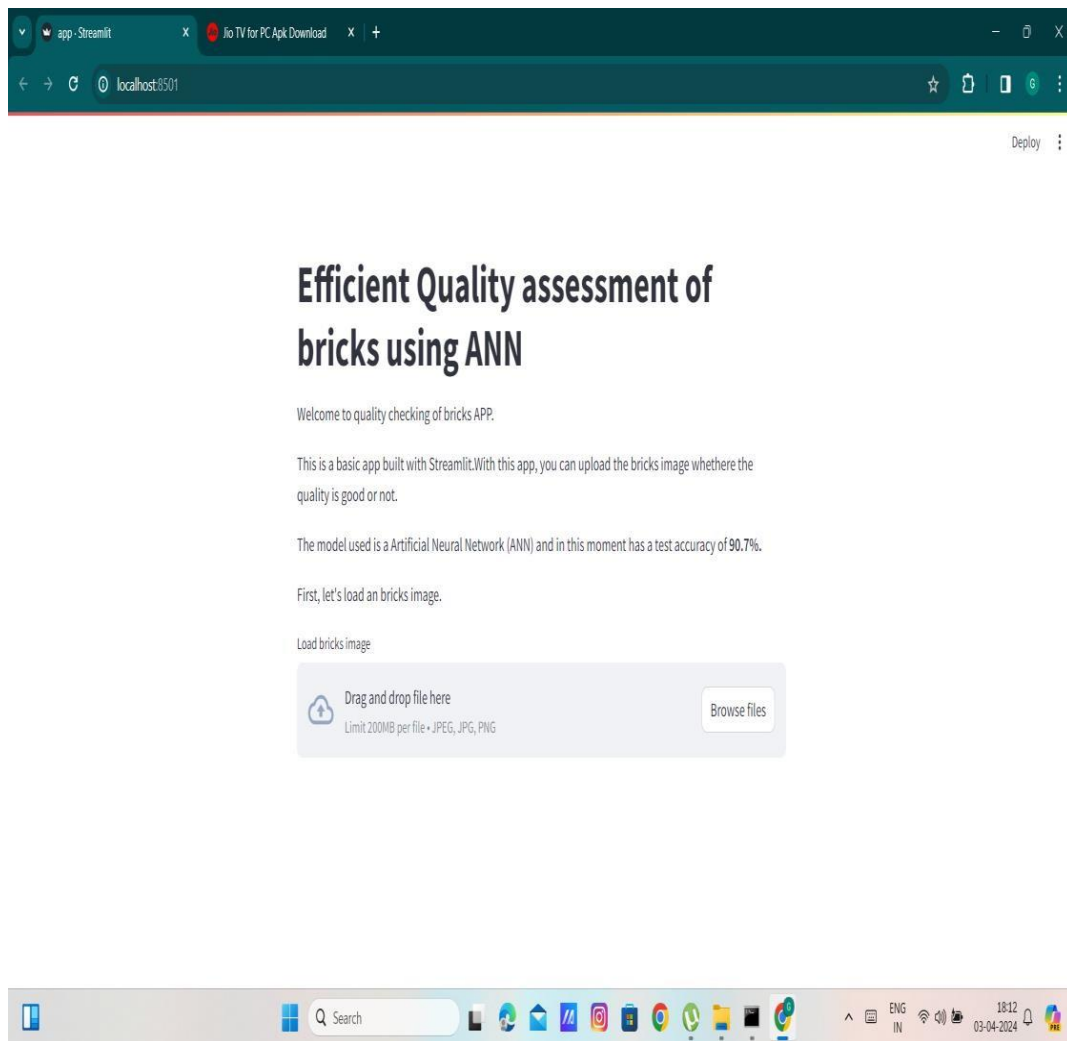


Fig:5.4 Efficient Quality assessment of bricks using ANN

CONFUSION MATRIX

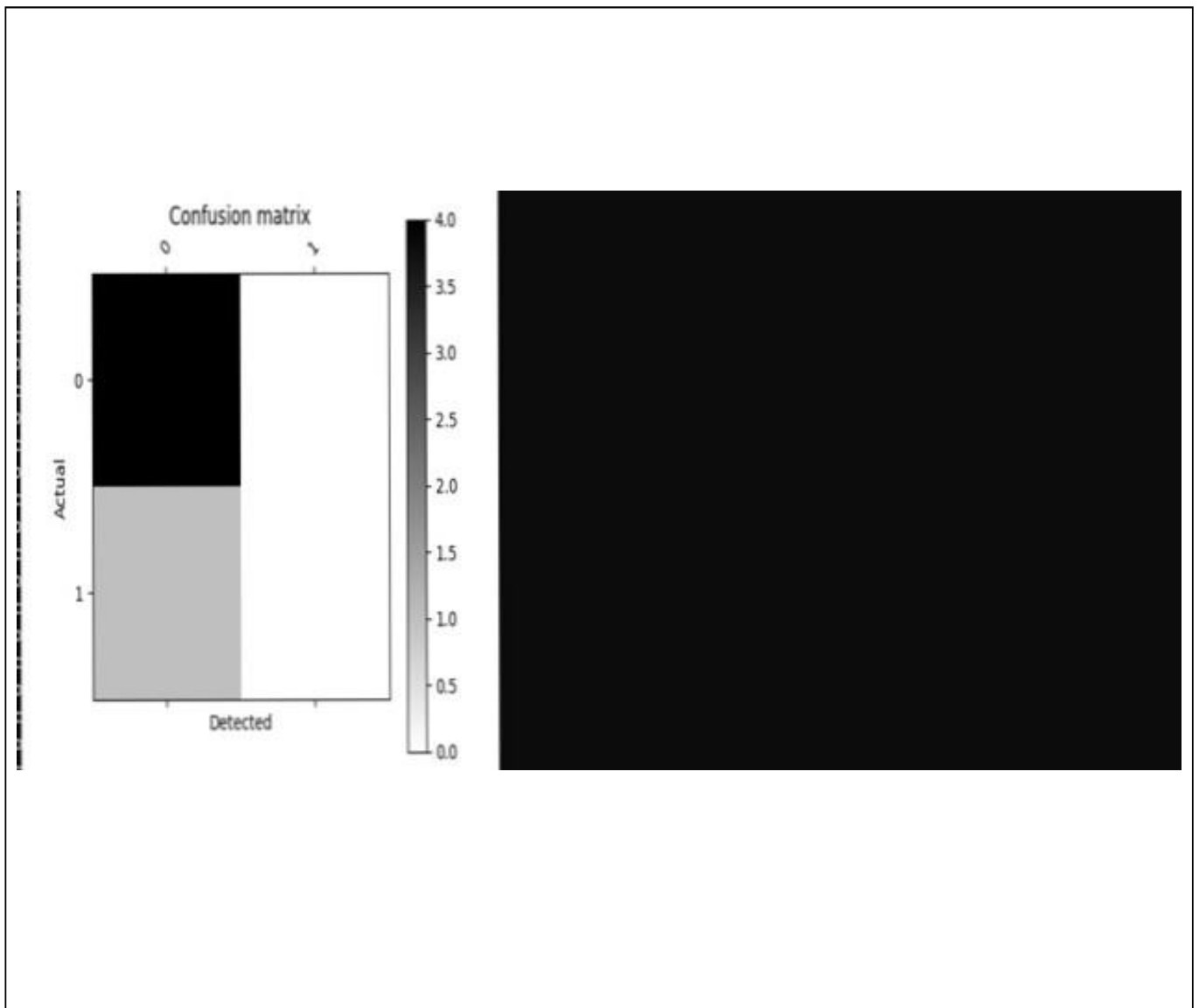


Fig:5.5: Confusion Matrix

STEP 6: This notifies the confusion matrix graph which shows the confusion matrix between the actual matrix

STEP 7: This picture shows the Accuracy checking

of the graph out of 100 and it will depict for each of the sample product.

PROBABILITY OF QUALITY

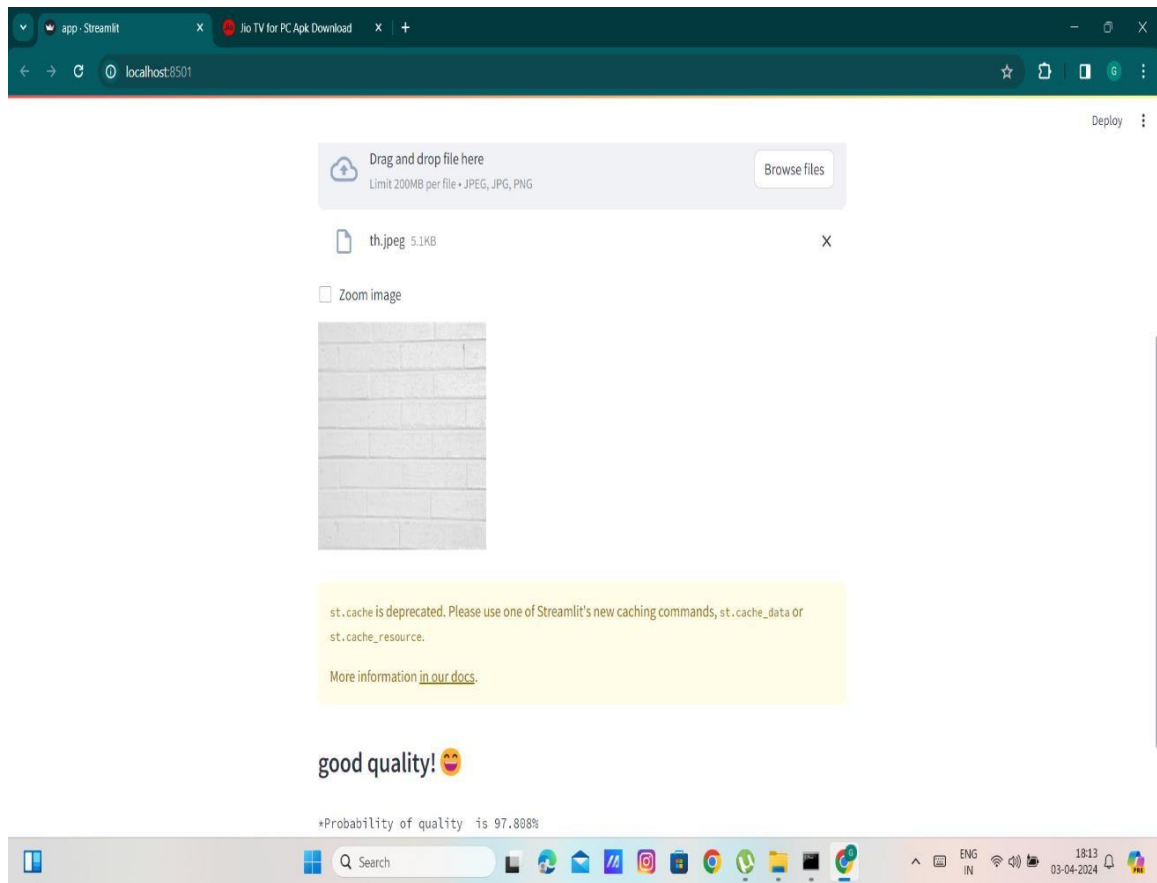


Fig:5.6: Probability of Quality

CHAPTER-6

Conclusion

An ANN model was trained using a set of 6400 training images obtained from sandstone core trays from different industry. The method was applied to automatically calculate the RQD by identifying intact core fragments and eliminating fractures, crushed cores, empty tray regions, and other objects present in the core trays. The RQD was calculated for each row in the core tray, resulting in an average error rate of less than 3% for new unseen sandstone tray images. In addition, the proposed method yielded accurate results when applied to limestone tray images, with an average error rate of approximately 3%. And in Conclusion we using a Artificial neural network to find the quality of a bricks

CHAPTER – 7

Reference

1. Ali A, Shams Uddin SM, Ralescu AL (2015) Classification with class imbalance problem: a review. *Int J Adv Soft Comput Appl* 7:176–204
2. Alom MZ, Taha TM, Yakopcic C et al (2018) The history began from AlexNet: a comprehensive survey on deep learning approaches. *arXiv:1803.01164*
3. Alqahtani N, Alzubaidi F, Armstrong RT et al (2020) Machine learning for predicting properties of porous media from 2d X-ray images. *J Pet Sci Eng* 184:106514. <https://doi.org/10.1016/j.petrol.2019.106514>
4. Alzubaidi F, Mostaghimi P, Swietojanski P et al (2021) Automated lithology classification from drill core images using convolutional neural networks.
5. *J Pet Sci Eng* 197:107933. <https://doi.org/10.1016/j.petrol.2020.107933>
6. Bao S, Chung ACS (2018) Multi-scale structured CNN with label consistency for brain MR image segmentation. *Comput Methods Biomech Biomed Eng Imaging vis* 6:113–117. <https://doi.org/10.1080/21681163.2016.1182072>
7. Barton N, Lien R, Lunde J (1974) Engineering classification of rock masses for the design of tunnel support. *Rock Mech* 6:189–236. <https://doi.org/10.1007/BF01239496>
8. Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyperparameter optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ (eds) *Advances in neural information processing systems*, vol 24. Curran Associates, Inc.,
9. Granada, Spain Bergstra J, Komer B, Eliasmith C et al (2015) Hyperopt: a Python library for model selection and hyperparameter optimization.

10. Sadiq M., Wen F., Dagestani A.A. Environmental footprint impacts of nuclear energy consumption: The role of environmental technology and globalization in ten largest ecological footprint countries. *Nucl. Eng. Technol.* 2022 doi: 10.1016/j.net.2022.05.016.
11. Golden A.P., Cohen S.S., Chen H., Ellis E.D., Boice J.D., Jr. Evaluation of statistical modeling approaches for epidemiologic studies of low-dose radiation health effects. *Int. J. Radiat. Biol.* 2022;98:572–579. doi: 10.1080/09553002.2018.1554924.
12. Hasegawa A., Tanigawa K., Ohtsuru A., Yabe H., Maeda M., Shigemura J., Ohira T., Tominaga T., Akashi M., Hirohashi N., et al. Health effects of radiation and other health problems in the aftermath of nuclear accidents, with an emphasis on Fukushima. *Lancet.* 2015;386:479–488. doi: 10.1016/S0140-6736(15)61106-0.
13. Desouky O., Ding N., Zhou G. Targeted and non-targeted effects of ionizing radiation. *J. Radiat. Res. Appl. Sci.* 2015;8:247–254. doi: 10.1016/j.jrras.2015.03.003.
14. Akkurt I., Akyildirim H., Mavi B., Kilincarslan S., Basyigit C. Gamma-ray shielding properties of concrete including barite at different energies. *Prog. Nucl. Energy.* 2010;52:620–623. doi: 10.1016/j.pnucene.2010.04.006.
15. Ahmad I., Shahzada K., Ahmad M.I., Khan F., Badrashi Y.I., Khan S.W., Muhammad N., Ahmad H. Densification of Concrete using Barite as Fine Aggregate and its Effect on Concrete Mechanical and Radiation Shielding Properties. *J. Eng. Res.* 2019;7:81–95.
16. Maslehuddin M., Naqvi A., Ibrahim M., Kalakada Z. Radiation shielding properties of concrete with electric arc furnace slag aggregates and steel shots. *Ann. Nucl. Energy.* 2013;53:192–196. doi: 10.1016/j.anucene.2012.09.006.
17. González-Ortega M., Cavalaro S.H.P., Aguado A. Influence of barite aggregate friability on mixing process and mechanical properties of concrete. *Constr. Build. Mater.* 2014;74:169–175. doi: 10.1016/j.conbuildmat.2014.10.040.

18. Saidani K., Ajam L., Ben Ouezdou M. Barite powder as sand substitution in concrete: Effect on some mechanical properties. *Constr. Build. Mater.* 2015;95:287–295. doi: 10.1016/j.conbuildmat.2015.07.140.
19. Osman G., Witold B., Cengiz O., Muuml F. An investigation on the concrete properties containing colemanite. *Int. J. Phys. Sci.* 2010;5:216–225.
20. Lv Y., Qin Y., Wang J., Li G., Zhang P., Liao D., Xi Z., Yang L. Effect of incorporating hematite on the properties of ultra-high performance concrete including nuclear radiation resistance. *Constr. Build. Mater.* 2022;327:126950. doi: 10.1016/j.conbuildmat.2022.126950.