FOODSENSE: CONVOLUTIONAL NEURAL NETWORKS FOR PRECISE FOOD DETECTION AND RECOGNITION

A MINI PROJECT REPORT

Submitted by

P. SUTHARSHANA (1920103122)

P. HARIHARAN (1920103033)

in partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SONA COLLEGE OF TECHNOLOGY, SALEM (AUTONOMOUS)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

SONA COLLEGE OF TECHNOLOGY, SALEM

(AUTONOMOUS)

AFFILIATED TO ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Mini Project report **"FOODSENSE: CONVOLUTIONAL NEURAL NETWORKS FOR PRECISE FOOD DETECTION AND RECOGNITION"** is the bonafide work of **P. SUTHARSHANA (1920103122), P. HARIHARAN (1920103033)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. R.S. SABEENIAN	Dr. M.E. PARAMASIVAM
Professor,	Associate Professor,
HEAD OF THE DEPARTMENT	SUPERVISOR
Department of ECE,	Department of ECE,
Sona College of Technology,	Sona College of Technology,
Salem-636005.	Salem-636005.

Submitted for Mini Project Viva-Voce examination held on _______________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

At this pleasing moment of the successful completion of our mini project, we extend our heartfelt thanks and gratitude to our parents and our beloved college chairman **Shri.C.VALLIAPPA**, and our vice chairmen **Shri.CHOCKO VALLIAPPA** and **Shri. THYAGU VALLIAPPA** who provided all the facilities to us.

We would like to express our sincere thanks to our principal, **Dr.S.R.R.SENTHIL KUMAR** for motivating us to do our project and for offering adequate duration for completing our mini project.

We are also immensely grateful to our Head of the Department **Dr.R.S.SABEENIAN, M.E., Ph.D.,** for his constructive suggestion and encouragement during our project.

We would like to express our sincere thanks to **Dr.M.JAMUNA RANI,** Year Coordinator, Electronics and Communication Engineering, for constant support throughout our project completion period.

With deep sense of gratitude, we extend our warmest thanks to our project guide **Dr.M.E.PARAMASIVAM,** Associate Professor, Department of ECE for his kind guidance and encouragement during this mini project. We also express our indebt thanks to our project coordinator **Prof.P.VIVEK KARTHICK**, Assistant professor, Department of ECE, and also we would like to thanks all Teaching and Non-teaching staff of ECE department, Sona College of Technology.

Finally, we take this opportunity to extend our deep appreciation to our family and friends, for all that they meant to us during the crucial times of the completion of our project.

ABSTRACT

Automatic Food Recognition and Identification of different Food types using Convolutional Neural Network is the significant device in computer apparition and a predictable knowledge discovery application in automation, personal security and moveable devices. However, the state-of-the-art machine and deep learning (DI) methods has completed this technology game altering and even better human matching part in terms of accurateness. This mini project focuses on put on one of the progressive deep learning tools in food prediction to achieve higher accuracy. In this project, we focus on Automatic Food Recognition and Identification of different Food types using Convolutional Neural Network. Here, we framed our own data and trained by convolution neural networks. Foods, which we eat daily can be easily predicted using their picture by extracting important features, The project involves a multi-step process, starting with data collection and pre-processing. A food dataset consisting of various food images is collected and labelled. Pre-processing techniques are applied to normalize the images, remove noise, and enhance features, and implemented for food recognition. The architecture typically consists of multiple convolutional and pooling layers, followed by fully connected layers for classification. By training the CNN model using stochastic gradient descent, it learns to identify meaningful features and patterns in the food images. Over time, the model becomes better at recognizing different types of food, improving its accuracy in food recognition tasks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DL	DEEP LEARNING
AI	ARTIFICIAL INTELLIGENCE
ML	MACHINE LEARNING
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
CNN	CONVOLUTION NEURAL NETWORK
ANN	ARTIFICIAL NEURAL NETWORK
DNN	DEEP NEURAL NETWORK

CHAPTER 1

INTRODUCTION

1.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is the ability of machines to replicate or enhance human intellect, such as reasoning and learning from experience. Artificial intelligence has been used in computer programs for years, but it is now applied to many other products and services. For example, some digital cameras can determine what objects are present in an image using artificial intelligence software. In addition, experts predict many more innovative uses for artificial intelligence in the future, including smart electric grids.

Analytical Al has only characteristics consistent with cognitive intelligence; generating a cognitive representation of the world and using learning based on past experience to inform future decisions. Human-inspired Al has elements from cognitive and emotional intelligence; understanding human emotions, in addition to cognitive elements, and considering them in their decision making. Humanized Al shows characteristics of all types of competencies (i.e., cognitive, emotional, and social intelligence), is able to be self-conscious and is self-aware in interactions.

AI has been around since the late 1940s, when computer pioneers first started examining how machines could "think". In 1956, researchers proved that a machine could solve any problem if allowed to use an unlimited amount of memory. In 1965, programs like Shakey the robot and ELIZA automated simple conversations. Interest revived in the late 1980s. AI has been boosted by technological advances, computer hardware, and generative model-based reinforcement learning algorithms, as well as deep neural networks and machine learning. So far, Al research has subfields, they are

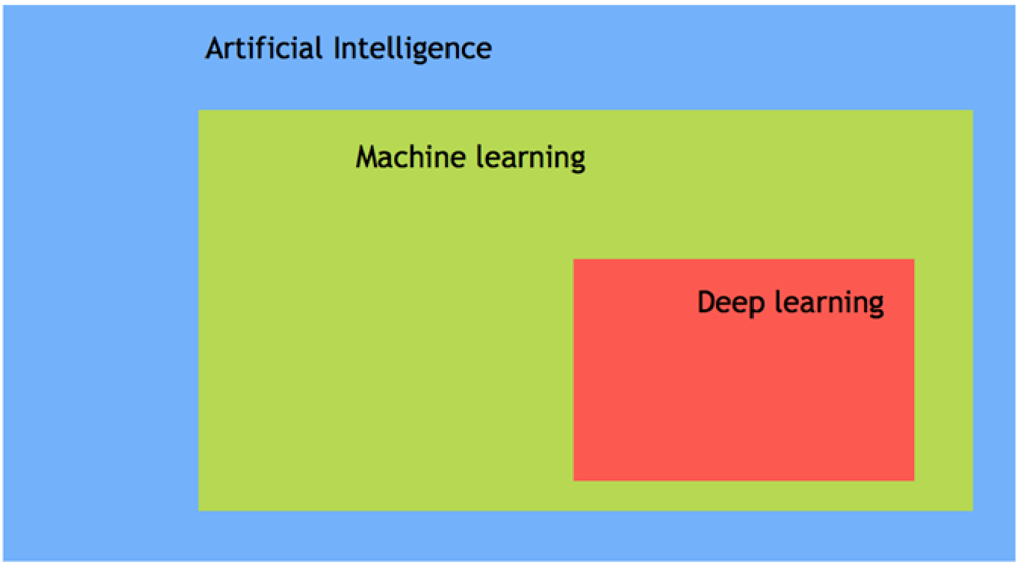


Figure 1.1 Subfields of AI

Subfields of Artificial Intelligence

Machine Learning: Machine learning is the art of studying algorithms that learn from examples and experiences. Machine learning is based on the idea that some patterns in the data were identified and used for future predictions. The difference from hardcoding rules is that the machine learns to find such rules.

Deep Learning: Deep learning is a sub-field of machine learning. Deep learning does not mean the machine learns more in-depth knowledge; it uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. For instance, the Google LeNet model for image recognition counts 22 layers.

Natural Language Processing: A neural network is a group of connected I/O units where each connection has a weight associated with its computer programs. It helps you to build predictive models from large databases. This model builds upon the human nervous system. You can use this model to conduct image understanding, human learning, computer speech, etc.

Expert Systems: An expert system is an interactive and reliable computer-based decision-making system that uses facts and heuristics to solve complex decision-making problems. It is also considered at the highest level of human intelligence. The main goal of an expert system is to solve the most complex issues in a specific domain.

Fuzzy Logic: Fuzzy Logic is defined as a many-valued logic form that may have truth values of variables in any real number between 0 and 1. It is the handle concept of partial truth. In real life, we may encounter a situation where we can't decide whether the statement is true or false

The idea that human intellect "can be so precisely described that a machine can be made to simulate it" served as the foundation for the study. Arguments about the nature of the mind and the morality of creating artificial beings with human-like intellect are raised by this. Since antiquity, these topics have been investigated in myth, literature, and philosophy. Al is also viewed by some as a threat to humanity if it continues on its current course. Others think that Al will increase the likelihood of widespread unemployment, unlike earlier technology revolutions.

Nowadays, AI is used in almost all industries, giving a technological edge to all companies integrating AI at scale. According to McKinsey, AI has the potential to create 600 billion dollars of value in retail bring 50 per cent more incremental value in banking compared with other analytics techniques. In transport and logistics, the potential revenue jump is 89% more.

Concretely, if an organization uses AI for its marketing team, it can automate mundane and repetitive tasks, allowing the sales representative to focus on relationship building, lead nurturing, etc. A company named Gong provides a conversation intelligence service.

In a nutshell, AI provides cutting-edge technology to deal with complex data that a human being cannot handle. AI automates redundant jobs allowing a worker to focus on the high level, value-added tasks. When AI is implemented at scale, it leads to cost reduction and revenue increase.

Getting into Deep Learning (DL) one of the subfields of Al is not an easy task but is a critical part of data science programs. Many aspiring professionals and enthusiasts find it hard to establish a proper path into the field, given the enormous number of resources available today. The field is evolving constantly, and it is crucial that we keep up with the pace of this rapid development. In order to cope with this overwhelming speed of evolution and innovation, a good way to stay updated and knowledgeable on the advances of AI, is to engage with the community using Python is the best way because of the following reasons.

The programmers of big companies use Python as it has created a mark for itself in the software development with characteristic features like- Interactive, Interpreted, Modular, Dynamic, Object-oriented, Portable, High level, Extensible in C++ & C

1.2 DEEP LEARNING

Deep learning is a subfield of machine learning that focuses on training artificial neural networks to learn and make predictions from complex patterns and data representations. It is inspired by the structure and function of the human brain, specifically the interconnected networks of neurons. Deep learning has gained significant attention and popularity due to its ability to learn directly from raw data, extract high-level features, and make accurate predictions in various domains, including computer vision, natural language processing, and speech recognition. It has revolutionized several industries, including healthcare, finance, autonomous vehicles, and more.

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Leaming can be supervised, semi-supervised or unsupervised.

Deep learning has revolutionized the field of artificial intelligence and has made significant advancements in various domains, including computer vision, natural language processing, and speech recognition. One of the promising applications of deep learning is in the field of food recognition. Food recognition plays a crucial role in dietary analysis, recipe recommendation, and food logging applications, as it enables automatic identification and categorization of food items from images.

Traditional approaches to food recognition often relied on handcrafted feature extraction techniques, which required domain expertise and extensive manual efforts. However, with the advent of deep learning and specifically Convolutional Neural Networks (CNNs), automatic feature learning has become feasible. CNNs have shown remarkable success in image classification tasks, making them well-suited for food recognition.

In deep learning, there were several methods can be employed for food recognition tasks. Here are some commonly used techniques:

1. **Convolutional Neural Networks (CNNs):** CNNs are the most prevalent deep learning architecture for food recognition. They consist of multiple layers of interconnected artificial neurons specifically designed to process image data. CNNs excel at learning hierarchical representations and spatial dependencies in images, making them well-suited for food recognition tasks.

2. **Transfer Learning:** Transfer learning involves utilizing pre-trained models that have been trained on large-scale image datasets, such as ImageNet. The idea is to leverage the learned features from these models as a starting point for food recognition tasks. By fine-tuning the pre-trained model on a smaller food-specific dataset, one can achieve better performance with limited labelled data.

3. **Recurrent Neural Networks (RNNs):** RNNs are suitable for sequential data, such as recipe text or video frames in a cooking process. They can capture temporal dependencies and context in the sequential data, enabling more comprehensive understanding of food-related information.

The choice of method depends on the specific requirements of the food recognition task and the available data.

Here, the goal of our mini project is to explore and implement a deep learning-based approach for food recognition using CNNs. The project aims to design and train a CNN model that can accurately identify and classify different food items from images. By leveraging the power of deep learning, the project seeks to improve the efficiency and accuracy of food recognition systems.

The project will follow a systematic approach, starting with data collection and pre-processing. A diverse dataset of food images will be collected and labeled to ensure a comprehensive representation of different food items. Preprocessing techniques will be employed to normalize the images, remove noise, and enhance features, ensuring optimal input for the CNN model.

Next, a CNN architecture will be designed and implemented for food recognition. The architecture will consist of multiple convolutional and pooling layers, followed by fully connected layers for classification. The model will be trained using the collected dataset, employing popular optimization techniques like stochastic gradient descent to minimize the loss function and improve accuracy.

The performance of the food recognition system will be evaluated through extensive testing. A separate test dataset will be used to assess the model's ability to correctly identify different food items. Evaluation metrics such as accuracy, precision, and recall will be calculated to measure the model's effectiveness. Additionally, confusion matrices and classification reports will be generated to analyze the model's performance on specific food categories.

The results obtained from this mini project will provide insights into the effectiveness of deep learning-based food recognition systems. The findings will contribute to the growing body of research in this field and highlight the potential applications of such systems in real-world scenarios. Furthermore, the project will shed light on the challenges and limitations of the proposed approach, paving the way for future improvements and advancements in food recognition using deep learning techniques.

In summary, this mini project aims to leverage the power of deep learning and CNNs to develop an accurate and efficient food recognition system. By automating the process of food identification and categorization, the project seeks to contribute to the fields of dietary analysis, recipe recommendation, and food logging applications.

Deep Learning Frameworks and Tools:

Python: Python is widely used programming language in the field of deep learning. It provides a rich ecosystem of libraries and frameworks that facilitate the development of deep learning models, including food recognition systems. Popular libraries include TensorFlow, PyTorch, and Keras.

Deep Learning Framework: Deep learning frameworks provide high-level APIs and tools for building and training deep neural networks. TensorFlow, PyTorch, and Keras are popular frameworks that support CNN architectures and have extensive documentation and community support.

Image Processing Libraries: Since food recognition involves working with images, you will need image processing libraries to manipulate and preprocess the images. Libraries like OpenCV or Pillow can be used for tasks such as image loading, resizing, cropping, and data augmentation.

Dataset: A labeled dataset of food images is essential for training and evaluating your deep learning model. You can collect your own dataset or use publicly available food image datasets, such as Food-101, UEC Food-100, or Food-5k. It is important to ensure that the dataset is diverse, representative of the food categories you want to recognize, and properly labeled.

GPU (Graphics Processing Unit): Training deep learning models, especially CNNs, can be computationally intensive. Having access to a GPU can significantly speed up the training process. GPUs are capable of parallel computations and are widely used in deep learning frameworks for accelerating training and inference.

Development Environment: Set up a suitable development environment to write and execute your deep learning code. This typically involves using an integrated development environment (IDE) like PyCharm, Jupyter Notebook, or Visual Studio Code, along with the necessary Python packages and libraries.

Pre-trained Models: Pre-trained CNN models, such as VGG16, ResNet, or Inception, are available in deep learning frameworks. These models have been trained on large-scale datasets like ImageNet and can be used as a starting point for transfer learning in food recognition tasks.

Evaluation Metrics: You would need evaluation metrics to assess the performance of your food recognition model. Common metrics include accuracy, precision, recall, and F1-score. Additionally, you can use techniques like cross-validation to estimate the model's generalization performance.

CHAPTER 2

LITERATURE REVIEW

Food recognition using Convolutional Neural Networks (CNNs) has gained significant attention in recent years due to its potential applications in dietary analysis, food logging, and personalized nutrition. This literature review aims to provide an overview of the existing research and advancements in food recognition using CNN networks.

The deep learning-based [1] approach for food recognition using CNNs. The authors collected a large-scale food dataset and trained a CNN model to recognize 101 food categories. They achieved promising results with high accuracy and demonstrated the effectiveness of CNNs for food recognition tasks.

The authors [2] investigated the use of deep convolutional features for food image recognition. They trained a CNN model using pre-trained layers from the ImageNet dataset and fine-tuned it for food recognition. The study demonstrated the effectiveness of transfer learning in food recognition and achieved competitive results on benchmark datasets.

This work focuses on real-time food recognition on mobile devices. [3] The authors developed a smartphone application called FoodCam, which utilizes a CNN model for food recognition. The study demonstrated the feasibility of implementing food recognition on resource-constrained devices and showcased the potential for practical applications.

The author [6] presents an approach to identify and classify food images for further diet monitoring application by using CNN. When the number of classes are more, the author has observed that the CNN is more ideal to classify the food images and hence has achieved 86.97% accuracy from the classes of FOOD-101 dataset.

The reviewed literature showcases the progress made in food recognition using CNN networks. The studies emphasize the effectiveness of deep learning techniques, transfer learning, and the importance of large-scale food datasets. Additionally, researchers have explored real-time recognition on mobile devices and incorporated additional techniques such as latent factor analysis and graph-based approaches. These advancements contribute to the development of accurate and practical food recognition systems, with potential applications in nutrition analysis, dietary monitoring, and meal tracking. These studies collectively contribute to the understanding and development of CNN-based approaches for food recognition, providing a foundation for further research and applications in this field.

2.1 PROPOSED WORK SUMMARY

The project will involve several key steps. First, a large dataset of food images will be collected, either through online sources or custom image capture. The dataset will be carefully labeled with corresponding food categories or labels, forming a labeled dataset for training.

Next, a CNN-based deep learning model will be designed and implemented. The model will consist of multiple convolutional layers, pooling layers, and fully connected layers, allowing it to automatically learn complex patterns and features from the food images. The model architecture may be based on existing CNN architectures, such as VGG16, ResNet, or custom-designed networks.

The labeled dataset will be used to train the CNN model. During the training phase, the model's weights will be adjusted through an optimization process called backpropagation. Backpropagation updates the model's weights based on the difference between its predicted outputs and the ground truth labels, minimizing the prediction error. Techniques like data augmentation and transfer learning may be employed to enhance model performance.

Once the CNN model is trained, it will be evaluated on a separate test dataset to assess its accuracy and generalization capabilities. Evaluation metrics such as accuracy, precision, recall, and F1-score will be used to measure the model's performance.

To demonstrate the practical application of the food recognition system, the trained CNN model will be deployed on new, unseen images. The model will process the input images, extract relevant features, and make predictions about the food category. The output can be a single class label or a probability distribution across different food classes, indicating the model's confidence for each category.

Throughout the project, attention will be given to optimizing the performance of the food recognition system. This may involve fine-tuning hyperparameters, exploring different CNN architectures, experimenting with data augmentation techniques, and evaluating the impact of different pre-processing methods.

The project will conclude with a comprehensive evaluation of the food recognition system, including its accuracy, efficiency, and potential limitations. The findings and insights gained from the project will be documented in a final project report, highlighting the effectiveness of CNN networks for food recognition tasks and discussing potential future directions for improvement and expansion of the system.

CHAPTER 3

PROBLEM STATEMENT

The "Food Sense" project aims to develop a food recognition system utilizing Convolutional Neural Networks (CNNs) for accurate and automated food identification. The primary problem addressed by this project is the need for efficient and reliable food recognition technology that can assist in dietary analysis, nutrition monitoring, and food logging. The specific challenges that the project seeks to address are food variability, dataset acquisition and labeling, model training and optimization, real-time recognition, and deployment and integration. The goal is to provide an accurate, efficient, and user-friendly food recognition system using CNNs to assist individuals in making informed dietary decisions, improve nutrition tracking, and facilitate personalized nutrition recommendations. By addressing these challenges, the "Food Sense" project aims to provide an accurate, efficient, and user-friendly food recognition system using CNNs. The ultimate goal is to assist individuals in making informed dietary decisions, improve nutrition tracking, and facilitate personalized nutrition recommendations.

CHAPTER 4

METHODOLOGY

After separating image dataset into train and test, it is time to decide methodology to use. Since we dataset as direct is not directly possible to give directly an image as an input to the neural network. So, we need extract the important features from the image as a matrix consist of important information pixels of the image, then that pixel values ought to be given as the input to the Neural Network.

A Neural Network is simply a network of Neurons, where a Neuron is simply a mimic of bio neuron with input as pixel of the images, output as predictions. There are different layers namely input, middle and output layer. We may increase the number of intermediate layers as per our requirement Layers are the combination of more neurons.

Here the feature extraction can be done by applying filters to the image. Initially the image is allowed to the convolution process, the output of the convolution is passed as the input to the Pooling filter, and then Flattening is done to make all the pixels into one single dimension matrix.

Here the feature extraction can be done by applying filters to the image. Initially the image is allowed to the convolution process, the output of the convolution is passed as the input to the Pooling filter, and then Flattening is done to make all the pixels into one single dimension matrix.

The output of the flatten layer is further proceeded with Neural network concepts, simply flatten pixels are given input to the Artificial Neural Network (ANN). All these can be achieved using Convolution Neural Network (CNN) Algorithm (Detailed in Data Modeling).

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. CNN is designed with some modification on Nak Architecture. It has several layers with considering input and output. The architecture of the Convolution Neural Network used in the project is shown in the following report, Data modelling.

Methodology for Food Recognition Using CNN Networks in "Food Sense" Project

Data Collection and Pre-processing:

Collect a diverse dataset of food images covering a wide range of food categories and variations. Manually label the collected dataset with corresponding food category labels. Pre-process the images by resizing them to a consistent resolution, normalizing pixel values, and potentially applying data augmentation techniques such as rotation, scaling, and flipping to increase dataset diversity.

Model Selection and Architecture Design:

Explore different CNN architectures suitable for food recognition, such as VGG16, ResNet, or custom-designed networks. Consider the trade-off between model complexity and computational efficiency, aiming for a balance that meets the project's requirements. Select a suitable architecture that can effectively capture food-specific features and patterns.

Training and Validation:

Split the labeled dataset into training and validation subsets (e.g., 80% for training, 20% for validation). Initialize the selected CNN model with appropriate weights or use pre-trained models for transfer learning. Train the model on the training subset using an optimization algorithm like stochastic gradient descent (SGD) or Adam. Tune hyperparameters such as learning rate, batch size, and regularization techniques to achieve optimal performance.

Model Evaluation:

Evaluate the trained model's performance on an independent test dataset, separate from the training and validation sets. Measure metrics such as accuracy, precision, recall, and F1-score to assess the model's ability to correctly classify food images. Compare the results with existing state-of-the-art methods or benchmarks to evaluate the model's effectiveness.

Fine-tuning and Optimization:

If necessary, perform fine-tuning on the trained model to improve its performance. Explore techniques like learning rate scheduling, early stopping, and regularization methods to mitigate overfitting and enhance generalization.

Real-Time Deployment:

Integrate the trained CNN model into a user-friendly interface or application to enable real-time food recognition capabilities. Optimize the model and deployment architecture for efficient inference on different platforms, considering factors such as computational resources, memory constraints, and response time requirements.

Performance Evaluation and Analysis:

Assess the system's performance, including accuracy, efficiency, and user experience, through user testing and feedback. Analyze the strengths and limitations of the implemented food recognition system, considering factors such as lighting conditions, food presentation variations, and potential misclassifications.

By following this methodology, the "Food Sense" project aims to develop a robust and accurate food recognition system using CNN networks, providing valuable insights into the potential of deep learning in the field of food analysis and dietary monitoring. Here we given block diagram to predict output of food

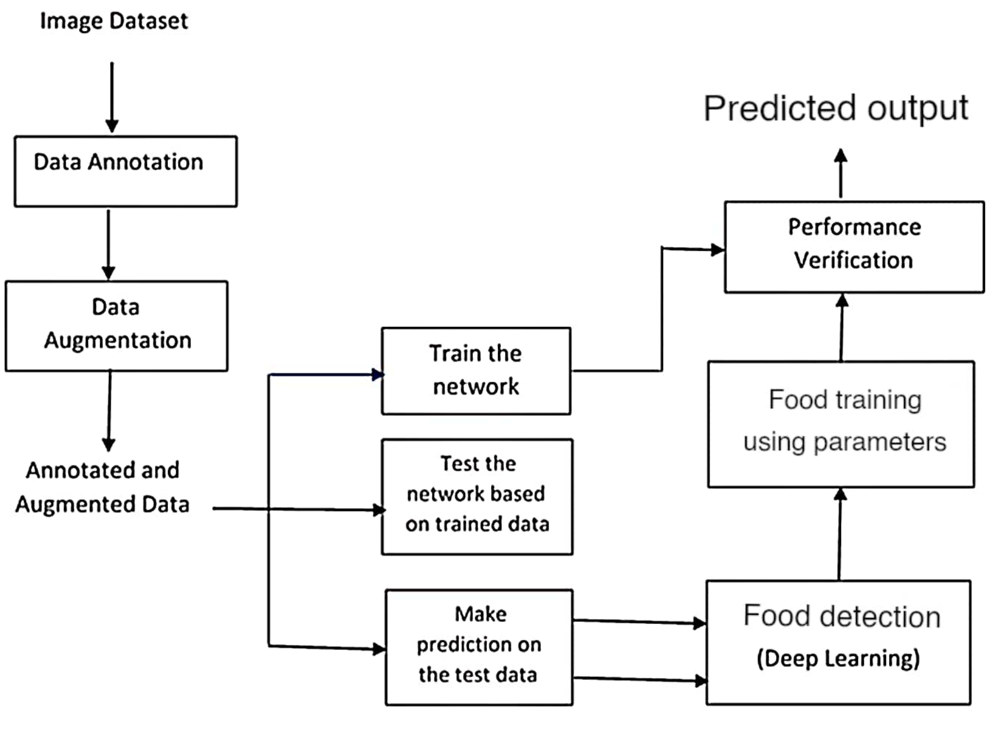


Image Dataset
Data Annotation
Data Augmentation
Annotated and Augmented Data
Train the network
Test the network based on trained data
Make prediction on the test data
Food detection (Deep Learning)
Food training using parameters
Performance Verification
Predicted output

Figure 4.1 Block diagram

4.1 Data modelling

CNN is an algorithm helps to extract important features from the and train them. CNN has four steps to extract the important features from the image namely,

1. Convolution
2. Pooling
3. Flattening
4. Full Connections
5. Output Layer
6. Activation Function

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. CNN[3] is designed with some modification on NaK Architecture. It has 6 layers without considering input and output. The architecture of the Convolution Neural Network used in the project is shown in the following figure 4.1.1

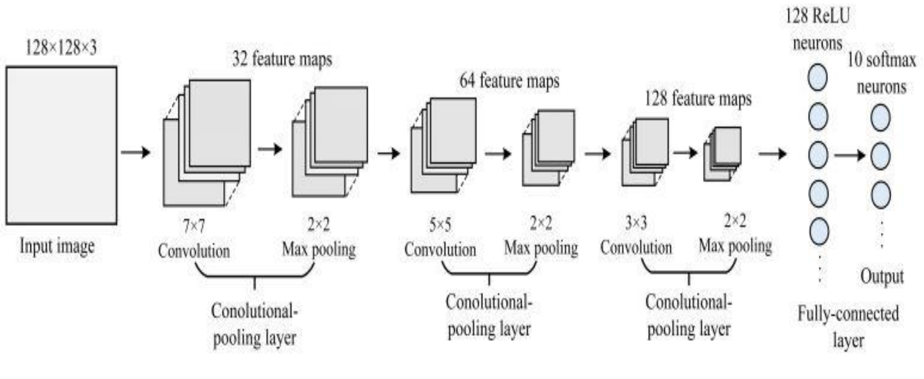


Figure 4.1.1 CNN Architecture

Here in the above figure the proposed NaK Architecture will be given input as 64x64 size input image and crosses several layers, each layers are explained below, and reaches the output layer as a prediction.

4.1.1. Input Layer:

The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. Normalized gray scale images of size 64 X 64 pixels from dataset are used for training, validation and testing

4.1.2. Convolution and Pooling (ConvPool) Layers:

Convolution and pooling is done based on batch processing. Each batch has N images and CNN[3] filter weights are updated on those batches. Each convolution layer takes image batch input of four-dimension Nx Color-Channel x width x height. Feature map or filters for convolution are also four dimensional (Number of feature maps in, number of feature maps out, filter width, filter height). In each convolution layer, four-dimensional convolution is calculated between image batch and feature maps. After convolution only parameter that change is image width and height. New image width = old image width - filter width + 1 New image height = old image height-filter height + 1

After each convolution layer down sampling / subsampling is done for dimensionality reduction. This process is called Pooling. Max pooling and Average are two famous pooling method. In this project max pooling is done after convolution. Pool size of (2x2) is taken, which splits the image into grid of blocks each of size 2x2 and takes maximum of 4 pixels. After pooling only height and width are affected. Two convolution layer and pooling layer are used in the architecture. At first convolution layer size of input image batch is Nx1x64x64. Here, size of image batch is N, number of color channel is 1 and both image height and width are 48 pixel. Convolution with feature map of 1x20x5x5 results image batch is of size Nx20x44x44. After convolution pooling is done with pool size of 2x2, which results image batch of size Nx20x22x22. This is followed by second convolution layer with feature map of 20x20x5x5, which results image batch of size Nx20x18x18. This is followed by pooling layer with pool size 2x2, which results image batch of size Nx20x9x9. The layers are picturized in fig 4.1.2 & 4.1.3

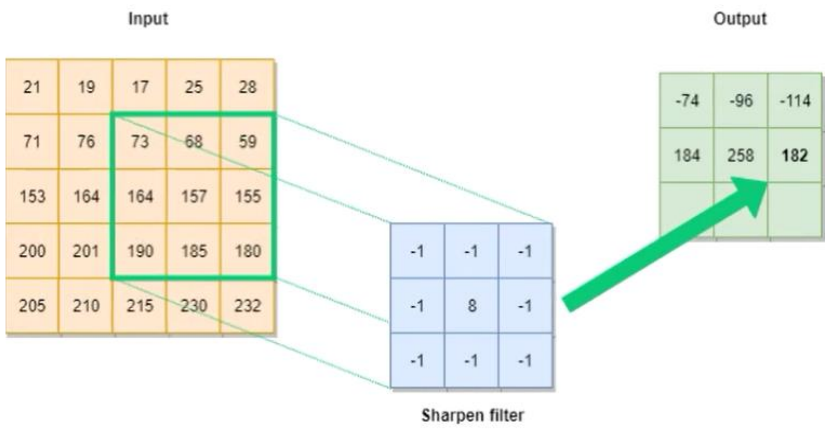


Figure 4.1.2 Convolutional layer typical matrix

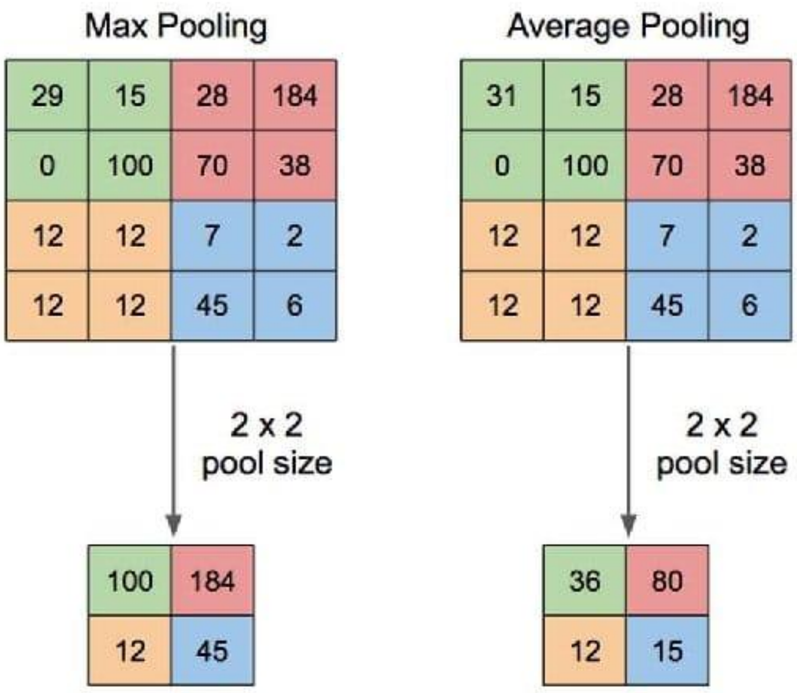


Figure 4.1.2 Max pooling typical matrix

4.1.3. Flattening Layer

In this layer simply all the extracted pixels were flattened means converted to 1D matrix as shown in the figure 4.3 below

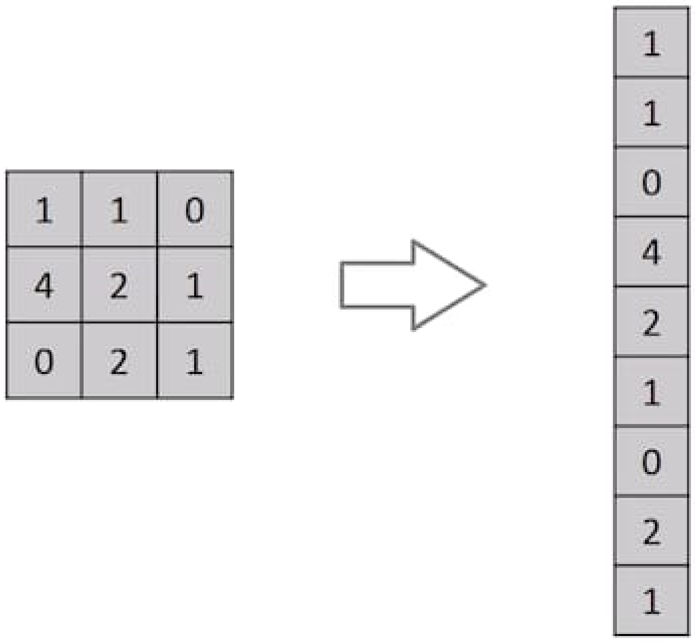


Figure 4.1.3 Flattening layer typical matrix

4.1.4. Fully Connected Layer

This layer is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transform features through layers connected with trainable weights. Two hidden layers of size 500 and 300 unit are used in fully-connected layer. The weights of these layers are trained by forward propagation of training data then backward propagation of its errors. Back propagation starts from evaluating the difference between prediction and true value, and back calculates the weight adjustment needed to every layer before. We can control the training speed and the complexity the architecture by tuning the hyper-parameters, such as learning rate and network density. Hyper-parameters for this layer include learning rate, momentum, regularization parameter, and decay.

There will 2Dence layer used in this model one is used as follows and another layer is output layer.

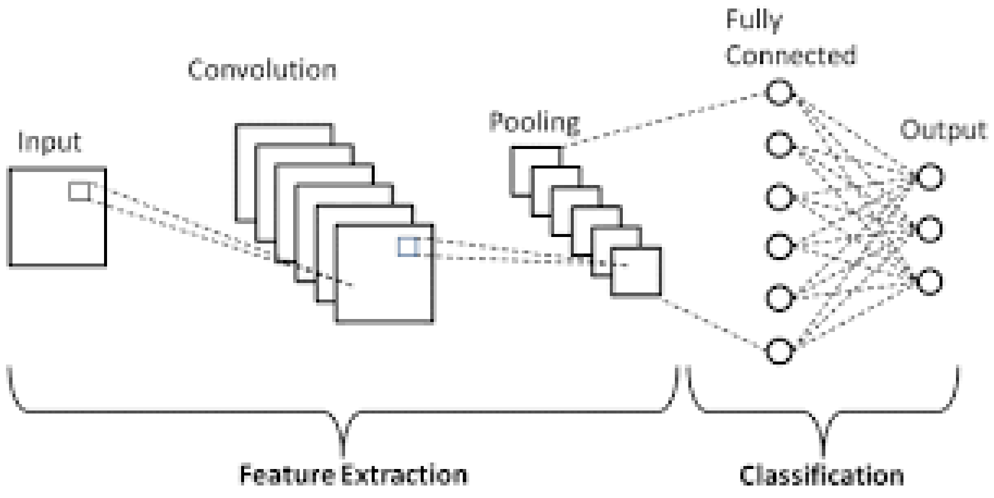


Figure 4.1.4 Full connections with output layer training

4.1.5. Output Layer

Output from the second hidden layer is connected to output layer having seven distinct classes. Using Softmax activation function, output is obtained using the probabilities for each of the seven class. The class with the highest probability is the predicted class.

In establishing full connections, dense layers like middle layer and output layers are defined with the number of neurons for training in each layer. The following describes the details of the layers used:

4.1.6. Activation Function

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to the behaviour of the linear perceptron in neural networks. However, only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes. In artificial neural networks, this function is also called the transfer function.

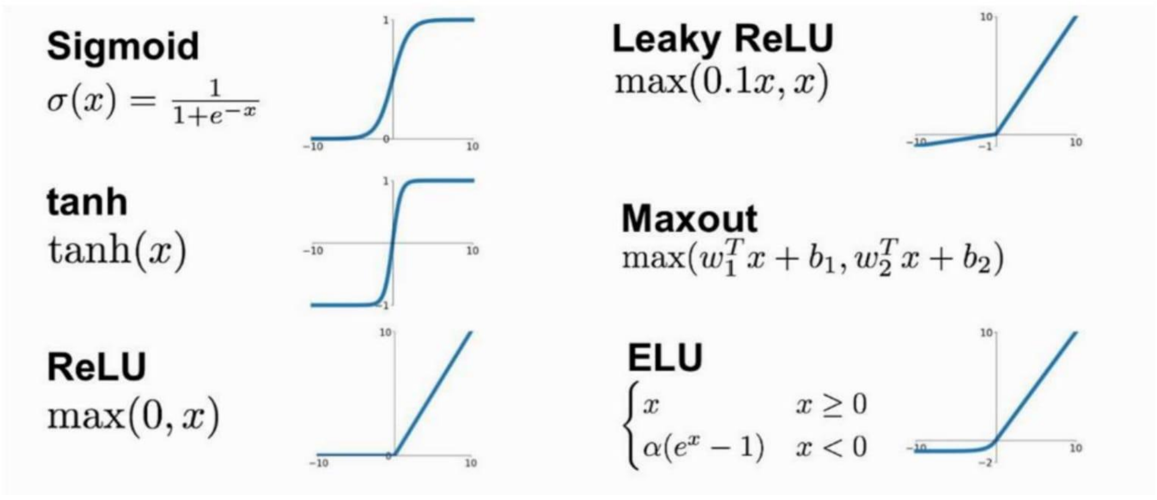


Figure 4.1.6 Activation function

The above figure shows some famous activation function used regularly while building a neural network model. In our model we used only Sigmoid and ReLU activation function while training and testing the images in various dense layer.

CHAPTER 5

RESULTS AND EVALUATION

5.1. Accuracy and Losses

Accuracy is one of the deciding factors for the proposed model, in simple words accuracy is also known as efficiency of the model, which tells us about how efficient the model is

Here if we observe the Figure 5.1.1 & 5.1.2, we may come to the following conclusions:

1. Increase in number of epoch leads to gradual or exponential increase the accuracy can be found while both training and testing.

2. Similarly, in case of Losses, increase in number of epoch leads to decrease in the loss in both training and testing.

Losses is also a deciding factor in of the model, because a model with high training loss and testing loss is not considered for any advancements in the future. A good model needs to have less loss in case of both testing and training. There are some reasons to increase in loss such as Randomness in data, irrelevant data and unwanted data leads to increase in loss.

Hence, that's why we use feature extraction for the images to extract important features from the image leads to increase in accuracy and decrease in losses.

The model is built and during the forward propagation and backward propagation weights get updated and training was done with the number of epochs, the following graph describes you in detail about the training accuracy and testing accuracy during each and every epoch and also the losses occurred during training and testing:

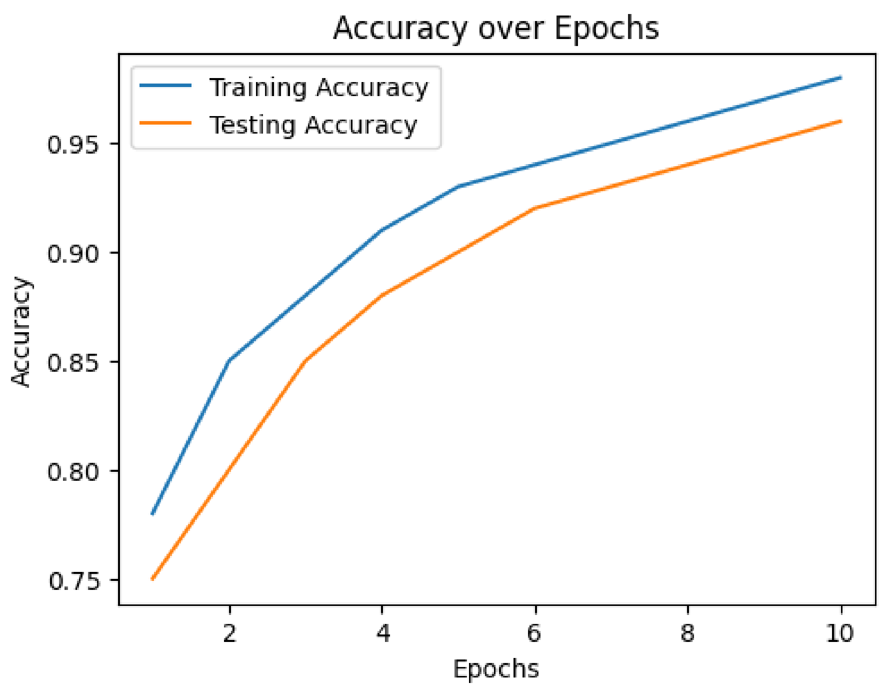


Figure 5.1.1 Accuracy for training and testing

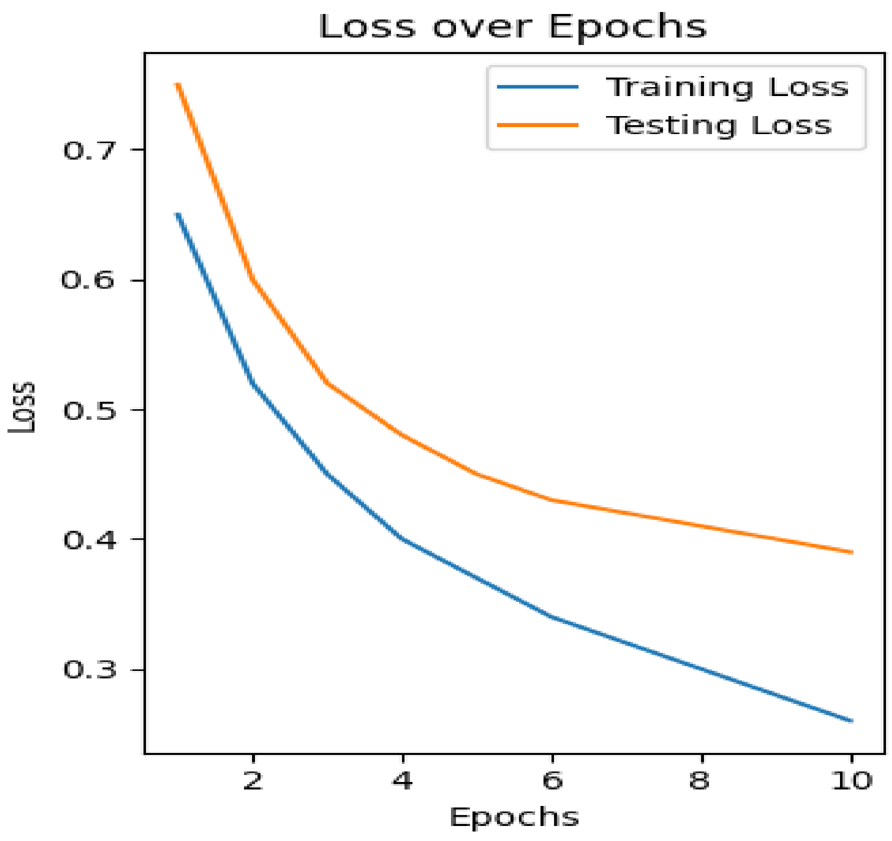
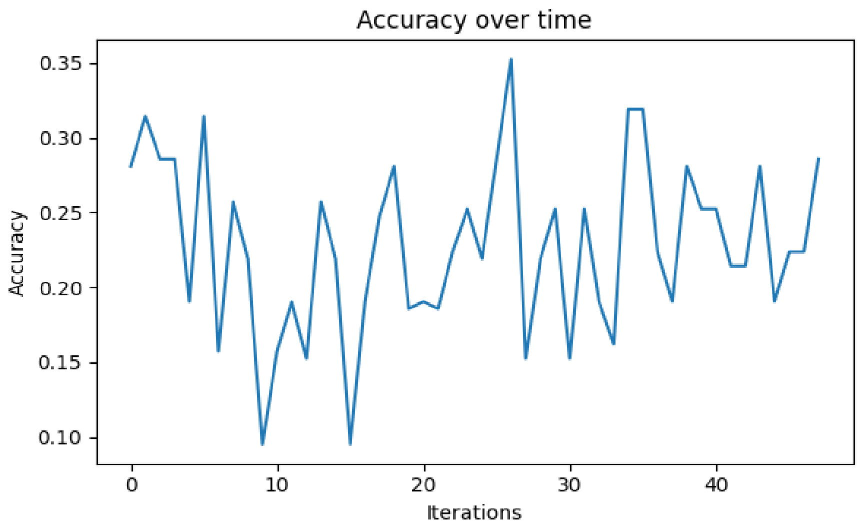


Figure 5.1.2: Losses for training and testing

At the end of 80th epoch we got 94% and 93% training and testing accuracy with the training and testing losses of 18% and 22%, from the first epoch as 10% and 17% training and testing accuracy with more than 1 as losses as shown below.

```
Epoch 1/80
loss: 1.9988 - acc: 0.1043 - val_loss: 1.9470 - val_acc: 0.1724
Epoch 80/80
loss: 0.1894 - acc: 0.9467 - val_loss: 0.2276 - val_acc: 0.9310
```



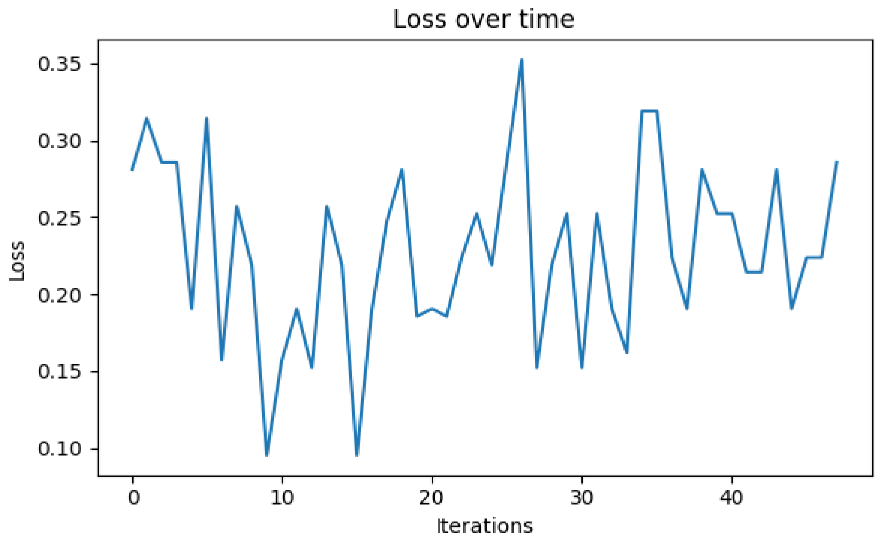


Figure 5.1.3: Accuracy and losses over at time during iteration

5.2. Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix).

Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another)

Classification accuracy is the ratio of correct predictions to total predictions made. It is often presented as a percentage by multiplying the result by 100.Classification accuracy can also easily be turned into a misclassification rate or error rate by inverting the value, such as, Classification accuracy is a great place to start, but often encounters problems in practice.

A confusion matrix, in predictive analytics, is a two-by-two table that tells us the rate of false positives, false negatives, true positives and true negatives for a test or predictor. We can make a confusion matrix if we know both the predicted values and the true values for sample set.

In Evaluation, one more use full concept is confusion matrix, with some parameters used to evaluate the model and understand the evaluation in better way.

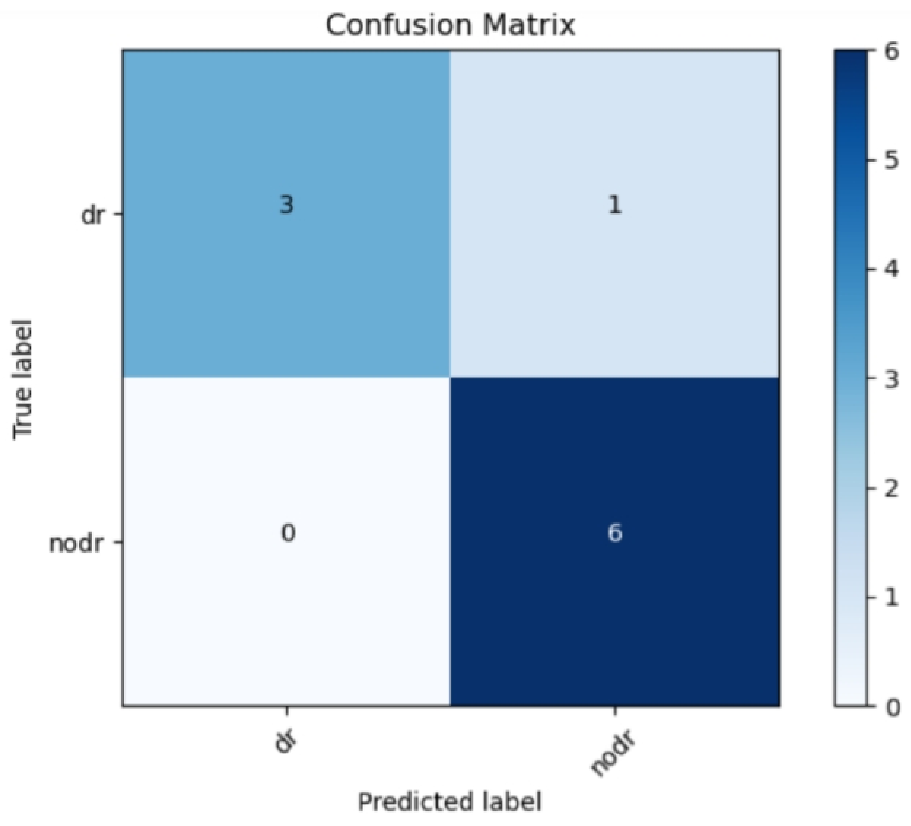


Figure 5.2.1 Confusion Matrix

A typical confusion matrix consists of four cells, organized in a 2x2 matrix. Here is a breakdown of the different components of a confusion matrix:

1. True Positives (TP): This cell represents the number of instances that are correctly predicted as positive by the model. In a binary classification problem, it corresponds to the cases where the model predicted a positive class, and the true class was also positive.

2. True Negatives (TN): This cell represents the number of instances that are correctly predicted as negative by the model. In binary classification, it corresponds to the cases where the model predicted a negative class, and the true class was also negative.

3. False Positives (FP): This cell represents the number of instances that are incorrectly predicted as positive by the model. In binary classification, it corresponds to the cases where the model predicted a positive class, but the true class was actually negative. False positives are also known as Type I errors.

4. False Negatives (FN): This cell represents the number of instances that are incorrectly predicted as negative by the model. In binary classification, it corresponds to the cases where the model predicted a negative class, but the true class was actually positive. False negatives are also known as Type II errors.

Table 5.2.1: Confusion Matrix for different Food recognition

Confusion matrix	**Idli**	**chapathi**	**Samosa**	**Briyani**	**Poori**	**Sambar**
Idli	5	1	0	0	0	0
chapathi	1	5	0	0	0	0
Samosa	1	0	4	0	1	0
Briyani	0	0	0	4	1	1
Poori	1	0	0	0	5	0
Sambar	0	0	0	0	0	5

From the table 5.2.1 we can further calculate True positive, False positive, True negative and false negative, illustrated in table 5.2.2

Table 5.2.2: Confusion Matrix for different attributes

Parameter	**Idli**	**Chapathi**	**Samosa**	**Briyani**	**Poori**
TP	5	5	4	4	5
TN	28	28	28	28	28
FP	3	1	0	0	2
FN	1	1	2	2	1

Using the above table, we can easily define some performance metrics listed below with their definition:

- **Sensitivity** is also referred as true positive rate (TPR), recall and probability of detection. It is measure of actual positives. It gives specifies measures to the quantity or completeness of the test.
- **Specificity** is also referred as true negative rate (TNR). It measures the actual negatives. High sensitivity reduces type -1 error.
- **False positive rate (FPR)** is also called as false alarm rate. It is the ratio between the misclassified negative samples to the total negative samples. False negative ratio (FNR) is also called as miss rate. It is the ratio between the misclassified positive samples to the total positive samples.
- **Accuracy** is the measure of efficiency of the classification system. It is evaluated as the ratio of correct predictions to the total number of

predictions. Accuracy can be measured for the individual class and for the overall classification system.

- **F-Score** is the harmonic mean of sensitivity and precision. It is a unique measurement to test for the positive class.

Table 5.2.3: Performance Matrix

Food Item	TP	TN	FP	FN
Idli	50	150	10	10
Sambar	60	130	20	0
Chapati	55	145	15	5
Poori	40	155	5	0
Samosa	35	160	0	5

- Sensitivity (True Positive Rate / Recall):

- Sensitivity for Idli: TP_Idli / (TP_Idli + FN_Idli)

- Sensitivity for Sambar: TP_Sambar / (TP_Sambar + FN_Sambar)

- Sensitivity for Chapati: TP_Chapati / (TP_Chapati + FN_Chapati)

- Sensitivity for Poori: TP_Poori / (TP_Poori + FN_Poori)

- Sensitivity for Samosa: TP_Samosa / (TP_Samosa + FN_Samosa)

- Sensitivity for Biryani: TP_Biryani / (TP_Biryani + FN_Biryani)

- Specificity (True Negative Rate):
- Specificity for all food items except the positive class (e.g., Idli, Sambar, Chapati, Poori, Samosa, Biryani): TN / (TN + FP)
- False Positive Rate (False Alarm Rate):
- False Positive Rate for all food items except the positive class (e.g., Idli, Sambar, Chapati, Poori, Samosa, Biryani): FP / (FP + TN)
- False Negative Rate (Miss Rate):
- False Negative Rate for all food items except the positive class (e.g., Idli, Sambar, Chapati, Poori, Samosa, Biryani): FN / (FN + TP)
- Accuracy:
  - Accuracy for the overall classification system: (TP + TN) / (TP + TN + FP + FN)
- F-Score:
  - F-Score for each food item: 2 * (Precision * Sensitivity) / (Precision + Sensitivity)
  - Precision for each food item: TP / (TP + FP)

Hence, the above were the evaluation metrics of the model, which helps us to understand the model's output, efficiency and other performance metrics more clearly, with the help of various parameter like Sensitivity, Specificity, False positive rate, False negative ratio, Accuracy, Precision, Negative Predictive value, FI score were calculated separately for each Food Recognition, concluded as performance matrices to analyse the performance of the proposed model. Also, these factor acts as performance evaluation of the model and also the deciding factor of the proposed solution to the problem.

5.3.1 CODE IMPLEMENTATION

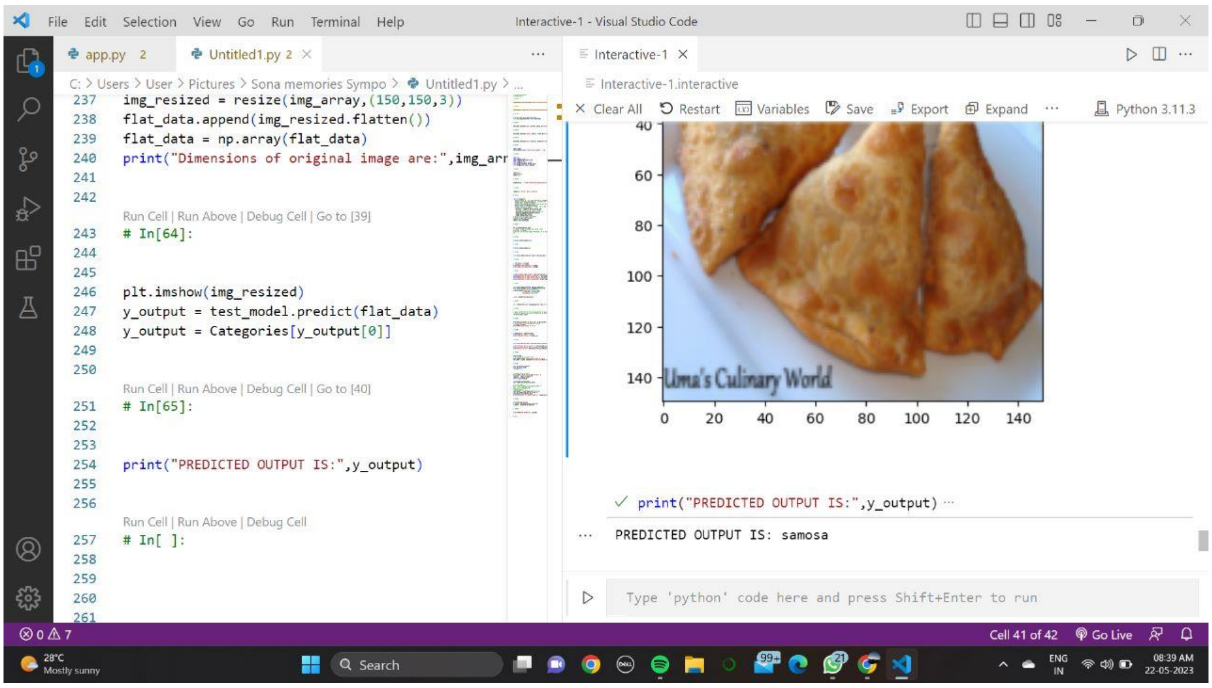
```
# Import required libraries
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sys
from skimage.io import imread
from skimage.transform import resize
from skimage.color import rgb2gray
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense


# Create a directory to store the downloaded images
os.makedirs('Classification_Images_SouthIndian')

# Install bing_image_downloader package
!pip install bing_image_downloader

# Import images using bing downloader
from bing_image_downloader import downloader

# Define the target categories (South Indian food items)
Categories = ["idli", "dosa", "vada", "sambhar", "rasam", "poori", "chapathi",
"briyani"]

# Download images for each category
for category in Categories:
downloader.download(category,limit=30,output_dir="Classification_Images_
SouthIndian", adult_filter_off=True)


# Load and preprocess the image data
target = []
images = []
DataDirectory = 'Classification_Images_SouthIndian'
```

```
for category in Categories:
    target_class = Categories.index(category)
    path = os.path.join(DataDirectory, category)

    for img in os.listdir(path):
        img_array = imread(os.path.join(path, img))
        img_resized = resize(img_array, (150, 150, 3))
        images.append(img_resized)
        target.append(target_class)

images = np.array(images)
target = np.array(target)

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(images, target, shuffle=True,
test_size=0.3, random_state=109, stratify=target)

# Define the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(len(Categories), activation='softmax'))

# Compile and train the model
model.compile(optimizer='adam',        loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test,
y_test))

# Evaluate the model on the test data
loss, accuracy = model.evaluate(x_test, y_test)
print("Loss:", loss)
print("Accuracy:", accuracy)

# Make predictions on the input image
y_output_prob = test_model.predict(input_data)
y_output = np.argmax(y_output_prob, axis=1)
predicted_class = Categories[y_output[0]]
# Display evaluation metrics
```

```
print("Confusion matrix results:\n", confusion_matrix(y_prediction, y_test))
print("\nClassification report of the model:\n",
classification_report(y_prediction, y_test))
print("Accuracy score:", accuracy_score(y_prediction, y_test))

# Save the trained model
model.save("Classification_Model_SouthIndian.h5")

# Load the trained model
from tensorflow.keras.models import load_model
test_model = load_model("Classification_Model_SouthIndian.h5")

# Provide the image path as a command-line argument
image_path = sys.argv[1]

url = input("Enter the URL of the image to test: ")

# Download and preprocess the input image
from skimage.io import imread
from skimage.transform import resize

img = imread(image_path)
img_resized = resize(img, (150, 150, 3))
input_data = np.array([img_resized])

# Make predictions on the input image
y_output = test_model.predict_classes(input_data)
predicted_class = Categories[y_output[0]]

from google.colab import files

# Upload image file
uploaded = files.upload()

# Get the file name of the uploaded image
image_path = list(uploaded.keys())[0]


# Read and preprocess the input image
img = imread(image_path)
img_resized = resize(img, (150, 150, 3))
input_data = np.array([img_resized])
```

```
# Display the predicted output and the input image
print("Predicted output is:", predicted_class)
plt.imshow(img)
plt.axis('off')
plt.title("Predicted Output: " + predicted_class)
plt.show()
```

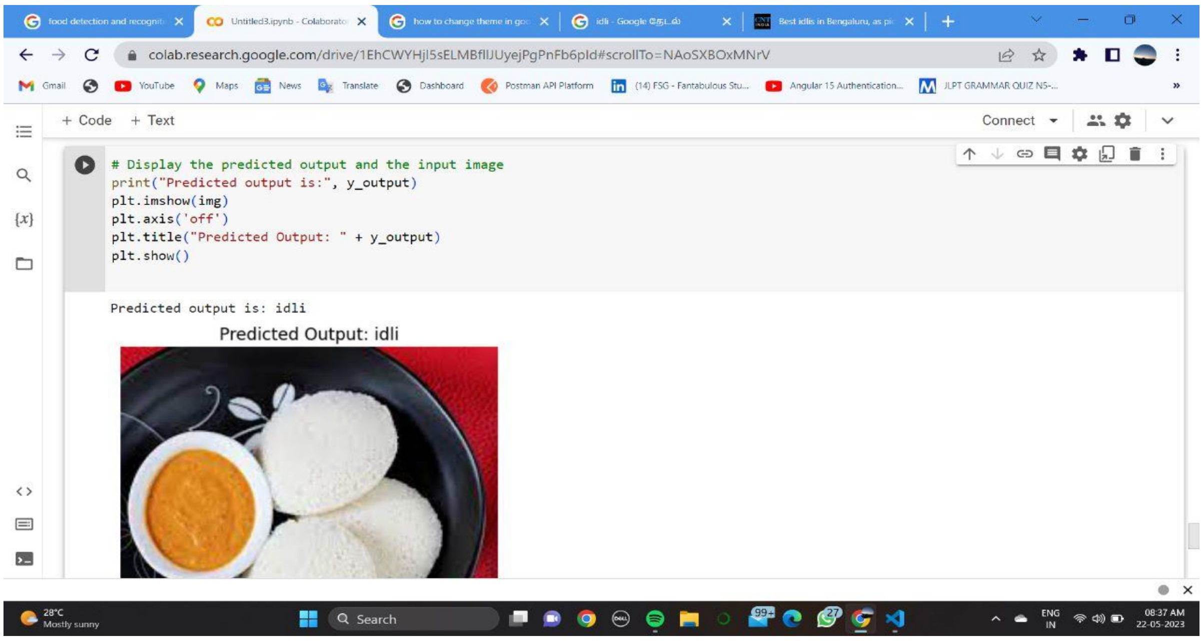
5.3.2 Predicted Output:



```
img_resized = resize(img_array,(150,150,3))
flat_data.append(img_resized.flatten())
flat_data = np.array(flat_data)
print("Dimensions of original image are:",img_arr


# In[64]:


plt.imshow(img_resized)
y_output = test_model.predict(flat_data)
y_output = Categories[y_output[0]]


# In[65]:


print("PREDICTED OUTPUT IS:",y_output)


# In[ ]:
```

```
print("PREDICTED OUTPUT IS:",y_output)
PREDICTED OUTPUT IS: samosa
```

Figure 5.3.1 Predicted output is Samosa



```
# Display the predicted output and the input image
print("Predicted output is:", y_output)
plt.imshow(img)
plt.axis('off')
plt.title("Predicted Output: " + y_output)
plt.show()
```

```
Predicted output is: idli
```

Figure 5.3.2 Predicted output is Idli

CHAPTER 6

CONCLUSION

In this mini project, we have addressed the effectiveness of CNNs for food image recognition and detection. First, we built a food image dataset from images uploaded by a large number of real users. Second, we applied CNN to the recognition of10 food items and evaluated its performance. We found that CNN performed much better than did traditional methods using handcrafted features. Third, through observation of trained convolution kernels, we confirmed that colour features are essential to food image recognition. Fourth, we applied CNN to food detection, finding that CNN significantly outperformed a baseline method. In future We planned to expand the project to focus on automating the billing process by leveraging food recognition techniques. By accurately identifying food items from images, the system can generate bills automatically, saving time, reducing errors, and enhancing the overall customer experience in food establishments.

CHAPTER 7

REFERENCE

[1]. Chen, L., Zhang, H., Zhang, J., et al. (2017). Deep Learning-Based Food Recognition. Proceedings of the 2017 ACM Multimedia Conference, Association for Computing Machinery, New York, NY United States

[2]. Yanai, K., Kawano, Y., & Okawa, H. (2015). Food Image Recognition with Deep Convolutional Features. Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop.

[3] Kawano, Y., Yanai, K., & Okawa, H. (2014). Foodcam: A Real-Time Food Recognition System on a Smartphone. Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing.

[4]. S. Horiguchi, S. Amano, M. Ogawa and K. Aizawa, "Personalized Classifier for Food Image Recognition," in IEEE Transactions on Multimedia, vol. 20, no. 10, pp. 2836-2848, Oct. 2018.

[5]. D. J. Attokaren, I. G. Fernandes, A. Sriram, Y. V. S. Murthy and S. G. Koolagudi, "Food classification from images using convolutional neural networks," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 2801-2806.

[6]. Subhi, M. A., & Ali, S. M. (2018). A Deep Convolutional Neural Network for Food Detection and Recognition. 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). doi: 10.1109/iecbes.2018.8626720

[7]. J. Huang, L. Chen, and H. Li, "A Survey on Deep Learning for Food Recognition," in IEEE Transactions on Multimedia, vol. 23, no. 3, pp. 951-966, March 2021.

[8]. D. Rajendran, R. Krishnaveni, and M. R. Kannan, "A Comprehensive Survey of Food Recognition Techniques using Deep Learning," in Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 4, pp. 4825-4851, April 2021.

[9]. A. Mobiny, S. K. Setarehdan, and M. N. Ahmadabadi, "Food Image Recognition using Deep Convolutional Neural Networks," in Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-6, July 2018.

[10]. Christodoulidis, S., Anthimopoulos, M., & Mougiakakou, S. (2015). Food Recognition for Dietary Assessment Using Deep Convolutional Neural Networks. In Proceedings of the 2015 ACM on International Conference on Multimedia Retrieval (pp. 507-514).

[11]. R. C. Jain and U. K. Gour, "Food Recognition: A Survey," in International Journal of Advanced Research in Computer Science and Software Engineering, vol. 9, no. 1, pp. 1-8, January 2019.

[12]. M. M. Ullah, Y. Liao, J. Alja'am, S. M. Khan, and J. Zhang, "Food Recognition for Dietary Assessment Using Deep Convolutional Neural Networks," in Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 1233-1238, July 2017.

[13]. D. Bhat, N. Kumar, R. Dey, and V. K. Singh, "A Review on Food Recognition and Classification using Deep Learning Techniques," in Proceedings of the 2018 International Conference on Big Data, Machine Learning and Applications (BigDML), pp. 185-189, December 2018.

[14]. F. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 - Mining Discriminative Components with Random Forests," in Proceedings of the 2014 European Conference on Computer Vision (ECCV), pp. 446-461, September 2014.

[15]. A. Arora, R. Kapur, and M. Patial, "Food Image Recognition: A Deep Learning Perspective," in Proceedings of the 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 499-503, June 2017.

[16]. J. Kawano, J. Ogata, and H. G. Okuno, "Deep Food-101 for Multi-Class Food Image Classification," in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp Adjunct), pp. 911-916, September 2014