Food Image To Recipe Converter

[1] K.Arhant , [2] G.Naveen , [3] S.Aasritha Reddy
dept. Artificial Intelligence
Amrita School of Engineering
Bengaluru, Karnataka

[1] bl.en.u4aie22123@bl.students.amrita.edu, [2]bl.en.u4aie22115@bl.students.amrita.edu, [3]bl.en.u4aie22144@bl.students.amrita.edu.

***Abstract*—The project aims to generate food recipe by classifying the image present in the dataset and along with that shall also give calorie, degree of the food, based on the image using multiple machine learning models**

Keywords— food recipe, degree of food, calorie

I. Introduction

Food is indeed a very important aspect in the life of a human being. It helps us gain energy, help sustain living and also provides rich nutrients to help us do our day- to-day activities. Food is thus an essential component in a human's life. Tasty food adds more nutrients and interest to children especially in their growing ages. But due to lack of finding the proper recipe or taking the worst case not knowing the dish's name can cause mother's and many people limit their food exploration capacity, making them limit to their daily food, which would have bored them. When children and even teenagers lose interest in eating food, they tend to lose their appetite and thus effects their health and doesn't let them enjoy food anymore, thus it is a sheer necessity to eat proper food. To make proper food, modern age people seek for recipes in the internet which do not actually tell them the quantity to enjoy quality food. Health freaks who are very much concerned about their health often tend to analyze their daily intake of food based on the number of calories and the protein thus giving a health-conscious recipe. Thus, we have are introducing Recepto. Recpeto is a image classification model that gives the statistics of food and also its recipe for a healthy diet through machine learning models. We take different parameters since we are very much health conscious and believe in the ideology of "a fitter world, makes a better world", thus we use different machine learning models to classify the image of the food and give the recipe based on that.

II. LITERATURE SURVEY

Food computing is becoming a major area of research, with the ultimate goal of creating machine learning systems that can automatically generate recipes based on food images. Current methods rely on finding similar recipes in a database, which limits their success. This paper proposes approaches, which uses powerful language models to directly generate recipes from images. The classification works in three major steps : (1)**BLIP Model** : Bootstrapping Language-Image Pre-training, is a state-of-the-art model used in the FIRE system for generating titles for food images. Here's a breakdown of its key aspects: BLIP is a multimodal model, meaning it can process both visual information (images) and textual information (language).It's specifically trained on a large dataset of images and their corresponding captions. (2) **Vision transformer with a decoder** : To extract the proper ingredients that are required for the creation of the recipe we need a tool that extracts the ingredients. There are multiple approaches but the best approach is through Vision transformer that is specified to bring out the ingredients that are necessary for the food. (3) **T5 Model** : Well last but not the least, once we are ready with the ingredients it is time to cook the food, thus we use the T5 model that is used in generating the recipe of the food and make it feel tasty.**[1]**

We tend to create a web and a mobile application platform that can be useful in making the model. Thus, to implement it as a dual service we have planned to blend in machine learning models that shall suffice both the sources. To come up with that we have used the following approaches : (1)**Advanced Search**: Forget keyword limitations! Feast In's innovative search algorithm helps you refine your recipe search based on available ingredients, dietary restrictions, cooking time, and even cuisine. No more sifting through irrelevant recipes – find the perfect fit for your needs. (2)**Search by Image**: Ever seen a mouthwatering dish but don't know what it is or how to make it? Feast In's powerful image recognition technology comes to the rescue. Simply upload a picture, and Feast In will suggest similar recipes, unlocking a world of culinary possibilities.**[2]**

This paper, by Obaid et al. (2020), delves into the world of deep learning models for image classification. As the authors highlight, the "big data age" has fueled the development of increasingly complex deep learning algorithms, capable of surpassing traditional machine learning methods in feature learning and expression. This has led to significant advancements in image classification tasks within the field of computer vision. The paper provides a comprehensive overview, starting with an introduction to deep learning itself. It then delves into various deep learning models used for image classification, including: **Convolutional Neural Networks (CNNs):** These are the workhorses of image classification, adept at extracting spatial features from images. **Recurrent Neural Networks (RNNs):** While less common, RNNs can be useful for handling sequential data like image sequences. **Transformers:** Newer architectures like transformers are gaining traction, offering advantages in handling long-range dependencies within images. Theauthors compare and contrast these models based on their strengths and weaknesses, highlighting factors like accuracy, computational efficiency, and suitability for specific tasks. They also discuss the impact of factors like dataset size and diversity on model performance. Finally, the paper focuses on two popular benchmark datasets for image classification: CIFAR-10 and CIFAR-100. The authors compare the performance of various deep learning models on these datasets, providing insights into their relative effectiveness. Overall, this paper offers a valuable resource for anyone interested in understanding the current landscape of deep learning models for image classification. It provides a clear and concise overview of the key concepts and models, along with insights into their strengths, weaknesses, and practical considerations. **[3]**

This research, by Gulzar (2023), investigates the use of MobileNetV2 with deep transfer learning for classifying fruit images. This approach leverages pre-trained models to improve accuracy and efficiency in tasks with limited data. The study focuses on a dataset of 40 different fruits and utilizes MobileNetV2, a lightweight and efficient CNN architecture. By applying transfer learning, the pre-trained weights of MobileNetV2 are fine-tuned on the fruit image dataset, allowing the model to learn fruit-specific features. **[4]**

A new study uses machine learning to predict the degree of processing in any food, revealing that over 73% of the US food supply is ultra-processed. This has concerning health implications, with increased reliance on ultra-processed food linked to higher risks of metabolic syndrome, diabetes, and other issues. Replacing these foods with less processed alternatives could significantly improve health outcomes, highlighting the importance of better informing consumers about processing levels.**[5]**

This paper provides a conceptual and technical foundation for building a food image to recipe converter using machine learning. This paper, mentions that the system outputs the recipe names of the food, ingredients, and cooking procedures using machine learning datasets. This paper, mentions using the convolutional Neutral networks (CNN) to categorize food images into various categories and output matched recipes. this paper helps our project to how to offering insights into model architecture using CNN, dataset considerations, and output structures, enhancing the practical applicability and performance of our project**.[6]**

This paper is on automatic food recognition and nutrient estimation from food images using some datasets and computer versions which is relevant to our food image to recipe converter project. It tells the state-of-the-art methods for processing food images, including classification, segmentation, and volume estimation. The systematic review of these methods, along with insights into the strengths and limitations, can inform the design of our model. By this paper, we can enhance the accuracy and effectiveness of ourfood image to the recipe converter project**.[7]**

This paper provides the use of a subset of the 1M+ dataset and it involves using a mobile application with an approach to search encodings of food images generated through a DenseNet-121 CNN. This approach simplifies the model and uses CNN directly for mapping, it calculates the similarity index for input food image and image in the dataset. It uses DenseNet-121, a deep CNN for processing the food images. It also tells the use of distance metrics and the KNN algorithm to find the closest matching result. Finally, this paper informsand guides various aspects of our project covering dataset selection, model architecture, nutritional value integration, and similarity index calculation. **[8]**

This paper provides the limitations of existing food-logging tools and aims to improve precision and convenience. this paper uses leveraging advanced machine learning techniques for food recognition to potentially improve the accuracy of recipe conversion. In their paper, these optimized models are integrated into an Android app named Food Insight. finally, this paper can assist our food image to recipe converter project by providing insights into addressing challenges, leveraging advanced machine learning techniques,integrating knowledge bases, handling contextual information, and offering a practical example of Android appimplementation**. [9]**

This paper provides creating standardized recipe datasets for machine learning is tough because recipes come in various formats and languages. In this study, we collected recipe datasets available publicly and made them consistent using dictionary and rule-based methods. We also used specific resources to convert measurements. This gave us two sets of data—one with ingredient embeddings and the other with recipe embeddings. When we tested a machine learning system to predict nutrients using these datasets, we found that combining embeddings using domain knowledge worked better than the usual methods**.[10]**

III. METHODOLOGY

A1. Implementing perceptron to learn AND gate logic

The code A1 implements a perceptron which is given to us in the question and we are using it to learn AND gate logic. The approach was such that we created two functions "step_function()" and a "perceptron_function()" which do their respective works. The step function was used to generate 1 if the input is greater than or equal to 0 else it would generate 0. The perceptron_gate() function was the main function where all the logic comes in. It is used to train the perceptron for a given set of inputs and outputs. Overall, the code trains a single layer perceptron for learning AND gate and finally generates weights

A2. Activation functions

The code A2 aims to do the same functionality as that of code A1 but will little addition. This time we were told to use activation functions which are as follow :

1. Bi-Polar Step function
2. Sigmoid function
3. ReLU function

To implement these functions we formulated their respective formulae and returned the final outputs, by also comparing the iterations taken to converge against each of the activation function.

A3. Varying learning rates

In the third question we had to follow the same logic as we did for the first question. This time the change was that we had to consider the weights to be same but the learnings had to be taken differently. A set of learnings were provided to us as follow : {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Thus we came up with the iterations and plotted a graph based on the learning rates that the question had asked us to implement.

A4. Learning XOR Gate

The code A4 has the same questions as that of A1-A3 but this time instead of considering AND gate we had to do the same with XOR gate. We used the same functions but the perceptron gate logic changed as per that of XOR gate logic thus it learned XOR Gate as well.

A5. Building a perceptron to classify a dataset

The code A5 deals with building a perceptron and classifies a certain data. Based on the data we created a dataset and thus inserted the same values that were present in the data, after which we built the perceptron. In the end the training accuracy and finals weights are produced.

A6. Matrix pseudo-inverse

This code reads transaction data from our project dataset, preprocesses the data by converting a binary column, and applies the simple perceptron learning algorithm for binary classification. compares the weights with those obtained using the pseudo-inverse method matrix. perceptron learning function adjusts weights iteratively based on the prediction error, this will separate the two classes in the data. finally, we get the weights from perception learning and weights from the pseudo-inverse matrix by using the code.

A7. Weights of networks using Back-propagation algorithm

This code implements a neural network to solve the XOR problem. it uses the sigmoid activation function, stochastic gradient descent, and backpropagation for training a specific number of epochs. Network weights are taken randomly and training continues until the mean error falls below a predefined threshold. the trained network is used to predict results for XOR inputs. Finally, for the given neutral networks we find the weights for the network using the back-propagation algorithm and also by AND gate logic.

A8. Learning rate and activation function

This code implements a perceptron for the XOR logic gate using a perceptron learning algorithm. The code applies this perceptron to the XOR gate problem, providing random initial weights, a learning rate, and a maximum number of epochs. The final weights and errors during training are printed, for the perceptron's ability to learn the XOR gate. finally, it displays the weights, final weights for the XOR gate, and also errors during the training same as question A1 where we did it with the AND gate logic, and here XOR gate logic.

A9. Zero output of lagic gate maps

The code says about a simple neural network class called `NeuralNetwork` which uses the sigmoid activation function and backpropagation for training. The class has different methods for forward propagation , backward propagation , and training. It defines the sigmoid activation function and its derivative.The `NeuralNetwork` class is instantiated three times to train neural networks for the AND, OR, and XOR gates. The training data for each gate is defined, and then the `train` method is called to train each neural network. The trained models are tested by printing their predictions for the given input data.

A10. MLP network from Sci-Kit

The code A10 aims to use scikit-learn `MLPClassifier` which helps to create and train neural network models for the AND and XOR logical gates. Also it defines input and output pairs for AND and XOR gates, which helps to create `MLPClassifier` instances with a single hidden layer fof 4 neurons and logistic activation function.And trains them using the input and output pairs.It also tests the trained models by predicting outputs for the input data and printing the predictions.

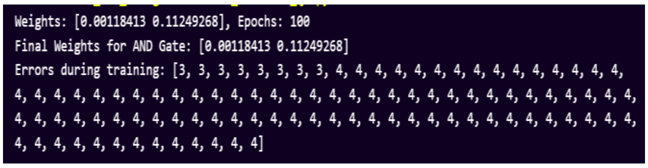
A11. MLP classifier on our project dataset

The code A11 aims to demonstrates how to train a multi-layer perceptron (MLP) classifier using scikit-learn's Pipeline and ColumnTransformer for preprocessing. At first it loads the dataset containing both numerical and categorical features, preprocesses the data by scaling numerical features and one-hot encoding categorical features, constructs an MLP classifier with two hidden layers, trains the classifier on the training data, makes predictions on the test data, evaluates the classifier's performance using accuracy score, and prints the accuracy achieved by the classifier.

IV. RESULT

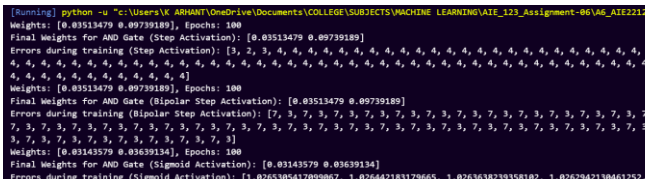
The project is currently under development and we are still under exploration to blend in as much as functionality required to make it a user-friendly application. The insight on machine learning is indeed a very helpful aspect in the complete project since that shall help users determine what they shall be eating and exploring. The project aims to provide an insight of the food quality and quantity of the user. It helps the user give detailed insights of what ingredients were used for making the food.

Below, we can see the results or output we can get by running the codes A1 to A9 in the codes we have done the testing and training and we calculate the accuracy, f1 score, recall, and precision for the question asked accordingly using our dataset which we are going to use in our project. we printed the confusion matrix and classification report as asked in some of the questions.



```
Weights: [0.00118413 0.11249268], Epochs: 100
Final Weights for AND Gate: [0.00118413 0.11249268]
Errors during training: [3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

Fig A1



```
[Running] python -u "c:\Users\K ARHANT\OneDrive\Documents\COLLEGE\SUBJECTS\MACHINE LEARNING\AIE_123_Assignment-06\A6_AIE2212
Weights: [0.03513479 0.09739189], Epochs: 100
Final Weights for AND Gate (Step Activation): [0.03513479 0.09739189]
Errors during training (Step Activation): [3, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Weights: [0.03513479 0.09739189], Epochs: 100
Final Weights for AND Gate (Bipolar Step Activation): [0.03513479 0.09739189]
Errors during training (Bipolar Step Activation): [7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7
7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3
3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3]
Weights: [0.03143579 0.03639134], Epochs: 100
Final Weights for AND Gate (Sigmoid Activation): [0.03143579 0.03639134]
Errors during training (Sigmoid Activation): [1.0265305417099067, 1.026442183179665, 1.0263638239358102, 1.0262942130461252
```

```
Final Weights for AND Gate (Sigmoid Activation): [0.03143579 0.03639134]
Errors during training (Sigmoid Activation): [1.0265305417099067, 1.026442183179665, 1.0
0261770438135855, 1.026127728745695, 1.026083611696053, 1.026044075346072, 1.026008582022
1.0258985081358682, 1.0258772743179063, 1.0258580190826028, 1.025840530725349, 1.0258246
0257848338423714, 1.0257737860449037, 1.02576366694707, 1.0257543870218477, 1.0257458665
1.0257180675234814, 1.0257124179290609, 1.025707199048822, 1.0257023739293234, 1.0256979
0256863906559732, 1.025683093718011, 1.025680032391498, 1.0256771881412579, 1.0256745446
1.0256656804952125, 1.0256638308240422, 1.0256621063813935, 1.0256604980364723, 1.025658
0256550672991194, 1.0256539259283117, 1.0256528591541976, 1.0256518617957653, 1.02565092
0256484758274635, 1.0256477603527545, 1.025647090333364, 1.0256464627259922, 1.02564587
0256443227917402, 1.0256438686219722, 1.0256434426170677, 1.0256430429453576, 1.02564266
0256416751621829, 1.0256413837682488, 1.025641110056737, 1.0256408529055636, 1.025640611
0256399699674672, 1.025639781223732, 1.0256396037050908, 1.0256394367146975, 1.025639279
0256388616182461, 1.025638738295946, 1.0256386221701927, 1.0256385128032774, 1.025638409
0256381351045714, 1.0256380538769125, 1.0256379773051818, 1.025637905111659, 1.025637837
Weights: [0.35714279 0.35714292], Epochs: 100
```

```
Weights: [0.35714279 0.35714292], Epochs: 100
Final Weights for AND Gate (ReLU Activation): [0.35714279 0.35714292]
Errors during training (ReLU Activation): [0.8839501159097484, 0.6332200313207735, 0.505847396167521, 0.441693566180236
39421914549373605, 0.3868489522382799, 0.38353234719381135, 0.3821759520507114, 0.3817340912130577, 0.381693148519497266
38196406630307644, 0.38211549890232477, 0.3822445357253228, 0.38234780812832614, 0.38242753398022267, 0.38248771705307
3825654662663628, 0.38258959472965515, 0.38260716484050816, 0.3826199166478665, 0.3826291495723866, 0.38263582330648815
382644116451856, 0.3826466213928825, 0.3826484261007447, 0.3826497258311415, 0.3826506616066348, 0.382651335181540
382652168709761, 0.38265241962690, 0.3826526001071204, 0.38265272990102767, 0.3826528232265608, 0.382652890317221
3826529731872112, 0.3826529980786562, 0.3826530159548294, 0.38265302878862417, 0.38265303799891, 0.382653044605973
382653052738230, 0.382653055169611, 0.3826530569097128, 0.382653058154087, 0.3826530590431518, 0.382653059677701
3826530604520998, 0.382653060681004, 0.38265306083414, 0.38265306095840, 0.38265306103962, 0.38265306109682
382653061165049, 0.382653061184583, 0.38265306119809, 0.382653061207384, 0.382653061213710, 0.382653061217972
3826530612226534, 0.382653061223828, 0.38265306122454956, 0.38265306122496, 0.38265306122518, 0.382653061225274
3826530612252494, 0.382653061225188, 0.382653061225115, 0.382653061225039, 0.382653061224965, 0.382653061224897
```

Fig A2

```
Weights: [0.05978764 0.07633394], Epochs: 100
Final Weights for AND Gate: [0.05978764 0.07633394]
Errors during training: [3, 3, 3, 2, 2, 2, 2, 3, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Epochs for different learning rates: [100, 100, 100]
```

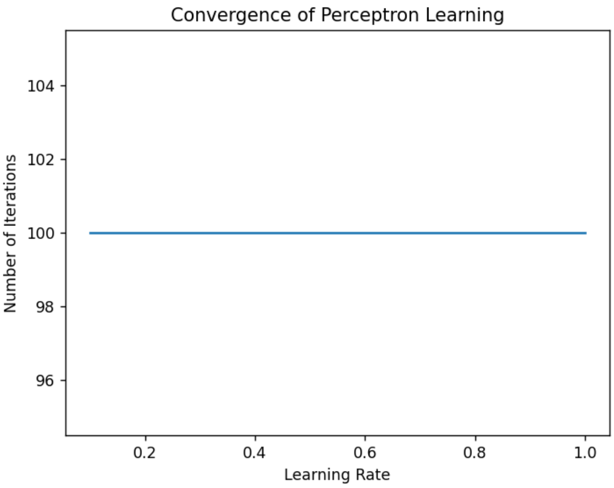


Fig A3

```
Weights: [-0.05230436 -0.06572775], Epochs: 100
Final Weights for XOR Gate: [-0.05230436 -0.06572775]
Errors during training: [2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Weights: [-0.08426004 -0.08200674], Epochs: 100
Final Weights for XOR Gate (Step Activation): [-0.08426004 -0.08200674]
Errors during training (Step Activation): [2, 2, 2, 2, 2, 2, 2, 2, 3, 3,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Weights: [-0.08426004 -0.08200674], Epochs: 100
```

```
Final Weights for XOR Gate (Step Activation): [-0.08426004 -0.08200674]
Errors during training (Step Activation): [2, 2, 2, 2, 2, 2, 2, 2, 3, 3,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Weights: [-0.08426004 -0.08200674], Epochs: 100
Final Weights for XOR Gate (Bipolar Step Activation): [-0.08426004 -0.082
Errors during training (Bipolar Step Activation): [10, 2, 10, 2, 10, 2, 1
10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10,
2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2,
10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10, 2, 10,
```

```
Weights: [-0.03401117 -0.03383179], Epochs: 100
Final Weights for XOR Gate (Sigmoid Activation): [-0.03401117 -0.033831
Errors during training (Sigmoid Activation): [1.0271236496280234, 1.026
0268062616313784, 1.026676323024596, 1.026562230512612, 1.0264619150487
1.02629571144004095, 1.0262269444741283, 1.0261661338390213, 1.026112279
0260645128933508, 1.0260220830630775, 1.0259843362898866, 1.025950704778
0259206945388706, 1.0258938753214724, 1.0258698720097663, 1.025848357177
1.0258116842288534, 1.0257960564737199, 1.0257819690810974, 1.025769253
0257577600072907, 1.0257473589186925, 1.0257379344659368, 1.025729384747
025721619646712, 1.025714559336817, 1.0257081330162077, 1.0257022777875
1.025692062874934, 1.0256876088425724, 1.0256835358370093, 1.025679808
0256763942612057, 1.025673265169749, 1.0256703952674642, 1.025667761391
```

```
Final Weights for XOR Gate (ReLU Activation): [0.28571428 0.2
Errors during training (ReLU Activation): [2.0174654838615997
7257966813856704, 1.64145267668507, 1.5950248354149574, 1.569
1.5453948319463549, 1.5401440537252211, 1.5368988660182505, 1
5335012646767117, 1.5326120551844793, 1.532010489153739, 1.53
5313103758700588, 1.5311091011205575, 1.5309669756770483, 1.5
5307942077507406, 1.5307428394395675, 1.5307060559169243, 1.5
5306607372021528, 1.5306471291290826, 1.5306373458783211, 1.5
5306252471955815, 1.5306216043742067, 1.5306189825935235, 1.5
5306157369985312, 1.5306147590552377, 1.530614055011537, 1.53
5306131832109546, 1.530612920473044, 1.5306127313070224, 1.53
5306124970497508, 1.5306124264469085, 1.5306123756131669, 1.5
```

```
1.5306122244897961, 1.5306122448979607, 1.5306122448979602, 1.530612
Weights: [-0.04521573 -0.03220784], Epochs: 100
Final Weights for XOR Gate: [-0.04521573 -0.03220784]
Errors during training: [2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Epochs for different learning rates: [100, 100, 100]
```

Fig A4

```
LEARNING\AIE_123_Assignment-06\A6_AIE22123_Q5.py:19: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-x))
Training accuracy: 100.00%
Final weights:
[ -52.08130608 -809.76895582  109.47056719 -285.60349171   70.24938901]
```

Fig A5

```
      dtype='object')
Weights from Perceptron Learning: [  -8.84 -114.84   18.4   -49.28   10.12]
Weights from Matrix Pseudo-Inverse: [-0.7720194  -0.05579846  0.02947478 -0.08646498  0.00894496]
```

Fig A6

```
[[0.03078322]
 [0.32291545]
 [0.32301408]
 [0.32341902]]
```

Fig A7

```
Weights: [-0.08594714 -0.09160469], Epochs: 100
Final Weights for XOR Gate: [-0.08594714 -0.09160469]
Errors during training: [2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

Fig A8

```
Training AND gate neural network:
Epoch: 0
Loss: 0.5876718958341327
Epoch: 1000
Loss: 0.24211654135115257
Epoch: 2000
Loss: 0.10968566399746801
Epoch: 3000
Loss: 0.06895569485077173
Epoch: 4000
Loss: 0.05267434782734461
Epoch: 5000
Loss: 0.0436934797920971
Epoch: 6000
Loss: 0.03789016685125578
Epoch: 7000
Loss: 0.03377839679335236
Epoch: 8000
Loss: 0.030684634785458458
Epoch: 9000
Loss: 0.028256237012047103

Training OR gate neural network:
Epoch: 0
Loss: 0.5207537309825327
Epoch: 1000
Loss: 0.13453441914026998
Epoch: 2000
Loss: 0.07620811090348689
Epoch: 3000
Loss: 0.05718882208970682
Epoch: 4000
Loss: 0.047353572701890875
Epoch: 5000
Loss: 0.041174421914995714
Epoch: 6000
Loss: 0.03685655517768602
Epoch: 7000
Loss: 0.03363032972788979
Epoch: 8000
```

```
Training XOR gate neural network:
Epoch: 0
Loss: 0.499851460811796
Epoch: 1000
Loss: 0.4998767620008471
Epoch: 2000
Loss: 0.49967675626704267
Epoch: 3000
Loss: 0.49811538130650856
Epoch: 4000
Loss: 0.45284381834881005
Epoch: 5000
Loss: 0.2919549246216559
Epoch: 6000
Loss: 0.17900828331510912
Epoch: 7000
Loss: 0.13406155431026429
Epoch: 8000
Loss: 0.110513799886942
Epoch: 9000
Loss: 0.09574358093648813

Testing the trained models:
AND gate predictions: [[0.99595701 0.00398432]
 [0.97150881 0.02848312]
 [0.97199063 0.02757301]
 [0.04497122 0.95524074]]
OR gate predictions: [[0.95594944 0.04350554]
 [0.02595876 0.97438038]
 [0.02604563 0.97427449]
 [0.01416576 0.98611736]]
XOR gate predictions: [[0.87557768 0.12235742]
 [0.08041429 0.9210583 ]
 [0.08903324 0.912439  ]
 [0.9490662  0.04996079]]
```

Fig. A9

```
AND gate predictions: [0 0 0 1]
XOR gate predictions: [0 0 0 0]
```

Fig.A10

Observations :If we look through the questions, what we have observed is that we had trained a perceptron to learn AND and OR gates using logics and based on that we had also understood how to build a perceptron when we have a set of data that is available with us. This led to the explorations of many concepts in machine learning that are actually useful for our project and also in the development of further innovations as well.

.

References

[1] Chhikara, P., Chaurasia, D., Jiang, Y., Masur, O. and Ilievski, F., 2024. FIRE: Food Image to REcipe generation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 8184-8194).J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Ann, E.T.L., Hao, N.S., Wei, G.W. and Hee, K.C., 2021. Feast In: A Machine Learning Image Recognition Model of Recipe and Lifestyle Applications. In MATEC Web of Conferences (Vol. 335, p. 04006). EDP Sciences.

[3] Obaid, K.B., Zeebaree, S. and Ahmed, O.M., 2020. Deep learning models based on image classification: a review. International Journal of Science and Business, 4(11), pp.75-81.

[4] Gulzar, Y., 2023. Fruit image classification model based on MobileNetV2 with deep transfer learning technique. Sustainability, 15(3), p.1906.

[5] Menichetti, G., Ravandi, B., Mozaffarian, D. and Barabási, A.L., 2023. Machine learning prediction of the degree of food processing. Nature Communications, 14(1), p.2312.

[6] A. Channam, B. R. Swarup and S. G. Rao, "Extraction of Recipes from Food Images by Using CNN Algorithm," 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2021, pp. 1308-1315, doi: 10.1109/I-SMAC52330.2021.9640741.

[7] F. S. Konstantakopoulos, E. I. Georga and D. I. Fotiadis, "A Review of Image-Based Food Recognition and Volume Estimation Artificial Intelligence Systems," in IEEE Reviews in Biomedical Engineering, vol. 17, pp. 136-152, 2024, doi: 10.1109/RBME.2023.3283149

[8] Mane, A., Bawankar, M., Goradia, M., Chaudhari, S. (2023). Image to Recipe and Nutritional Value Generator Using Deep Learning. In: Ranganathan, G., Fernando, X., Rocha, Á. (eds) Inventive

Communication and Computational Technologies. Lecture Notes in Networks and Systems, vol 383. Springer, Singapore.

[9] Z. Hakguder, Y. He, W. Chai and J. Cui, "Smart Diet Management through Food Image and Cooking Recipe Analysis," 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Las Vegas, NV, USA, 2022, pp. 2603-2610, doi: 10.1109/BIBM55620.2022.9994888.

[10] G. Ispirova, T. Eftimov and B. K. Seljak, "Predefined domain specific embeddings of food concepts and recipes: A case study on heterogeneous recipe datasets," 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 2022, pp. 4074-4083, doi: 10.1109/BigData55660.2022.10020698.