Pixel to Plate :

Deep Learning Based - Recipe Discovery and Recommendation

Sai Tejasri Yerramsetti, Anurima Saha, Radhika Ravindra, Sayandip Pal, Pritkumar Chakalasiya

Project under
Prof. Ryan Rafler
BDA 602 - MACHINE LEARNING

Abstract

This paper presents an innovative approach that integrates machine learning, deep learning, and optical character recognition (OCR) techniques to automate the identification of grocery items and provide intelligent recipe recommendations. The main objective is to develop an automated system capable of accurately recognizing a wide range of fruits, vegetables, and packaged goods using advanced computer vision models and OCR methods. Our methodology involves the use of transfer learning to fine-tune pre-trained vision models on the Fruits and Vegetables dataset from Kaggle, alongside the integration of state-of-the-art OCR solutions to extract text from multilingual packaged grocery items, enabling comprehensive ingredient recognition. Additionally, natural language processing techniques have been incorporated to offer personalized recipe recommendations based on the identified ingredients, utilizing methods such as word embeddings, word vectorization, and similarity measures. Our holistic approach has yielded highly accurate results, with the fine-tuned vision models achieving an impressive 96.38% accuracy on the test set. This research aims to revolutionize the grocery shopping experience by providing seamless item identification and intelligent recipe recommendations, thereby enhancing culinary exploration and convenience for consumers.

Keywords: Deep Learning, Transfer Learning, Optical Character Recognition (OCR), Recipe Recommendation, Grocery Item Identification, Vision Transformers, Convolutional Neural Networks (CNNs), Natural Language Processing (NLP), Word Embeddings, Vectorization, Multilingual Text Recognition.

1. Introduction

In an era characterized by rapid technological advancement, our research focuses on revolutionizing the culinary industry through the integration of cutting-edge technologies such as deep learning and natural language processing. Our research aims to harness the power of deep learning and natural language processing to enhance the grocery shopping experience and culinary recipe exploration. By integrating these advanced technologies, we seek to improve grocery item identification and provide personalized recipe recommendations tailored to individual preferences.

Our approach revolves around leveraging sophisticated deep learning algorithms and optical character recognition (OCR) techniques to navigate the complexities of the grocery marketplace. We explore image classification methods to accurately identify a wide range of fruits, vegetables, and packaged goods. Using convolutional neural network (CNN) architectures and transfer learning, we aim to enhance model performance and ensure accurate classification.

Moving to text recognition, we tackle challenges in extracting multilingual text from packaged grocery items. Our investigation into OCR methodologies emphasizes the importance of thorough data preprocessing and translation efforts for seamless integration of ingredient information into our recommendation system. In recipe recommendation, we combine natural language processing (NLP) and machine learning techniques. Through methods like TF-IDF vectorization and cosine similarity metrics, we analyze recipe composition to offer personalized suggestions based on user-specified ingredients. Our recommendation system showcases the potential of machine learning to revolutionize culinary exploration.

By combining these technologies, we aim to:

Enhance Grocery Item Identification: We utilize sophisticated deep learning algorithms, OCR techniques, and image classification methods to accurately identify a wide range of grocery items, including fruits, vegetables, and packaged goods. By doing this, we strive to achieve precise and reliable identification of diverse products for the user.

Personalized Recipe Recommendations: Our approach extends to providing personalized recipe recommendations tailored to individual preferences. Leveraging TF-IDF vectorization and cosine similarity metrics, we analyze recipe compositions and generate personalized recommendations for users. This personalized recommendation system aims to revolutionize culinary exploration by providing tailored guidance, increasing engagement and enriching the experience to users.

2. Exploratory Data Analysis

For this project we have focused on the following three datasets:

2.1 The Freiburg Groceries Dataset

Source - Kaggle

Link - https://github.com/PhilJd/freiburg_groceries_dataset

Description - This dataset consists of 5000 256x256 RGB images of 25 food classes like nuts, oil, pasta, etc. It has imbalanced class sizes ranging from 97 to 370 images per class. All the images were taken in various aspect ratios and padded to squares.

2.2 Recipe Dataset

Source - Kaggle

Link - https://github.com/Glorf/recipenlg/tree/c8e3681133b05af48bb5a2a9c19d1a0cf7636e1d

Description - The dataset we publish contains 2231142 cooking recipes (>2 million). It's processed more carefully and provides more samples than any other dataset in the area. It results in approx 1.6M recipes of better quality. Some of the features are: 'title', 'ingredients', 'directions', 'link', 'source' and ‘NER’.

2.3 Fruits and Vegetables Dataset

Source - Kaggle

Link - https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition

Description - This dataset contains images scraped from *Bing Image Search* organized into three primary folders: train (approximately 100 images per category), test (10 images per category) and validation (10 images per category). Each dataset folder contains subfolders dedicated to *thirty six different classes* of fruits and vegetables. The first image from each categorical folder in the train set is displayed below.

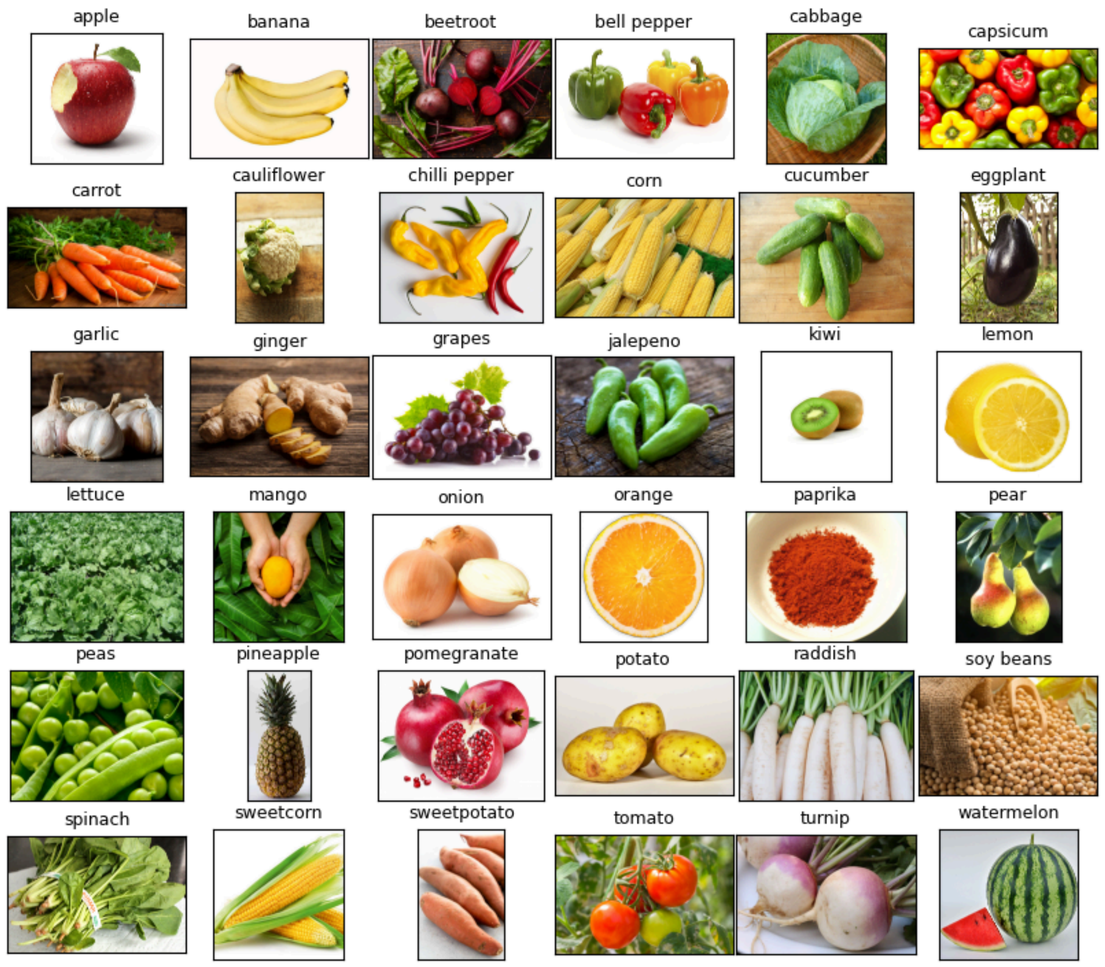


Fig 1: Plotting the first image along with label under each class of of 'Fruits and Vegetables Dataset'

The following histogram shows the distribution of samples in different classes of the training set.

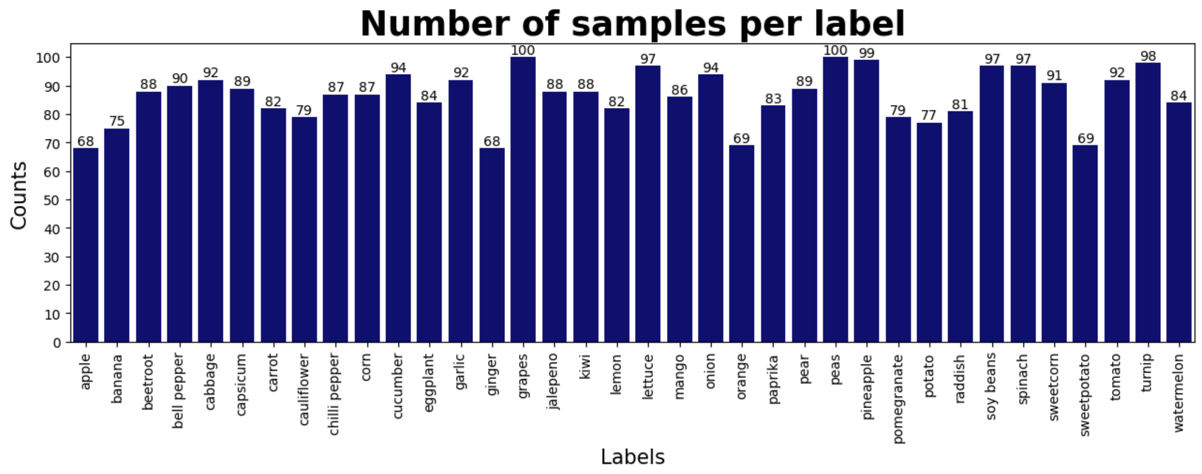


Fig 2: Histogram showing class label and distribution of samples across each class in the training set

We see that the frequency distribution ranges from a minimum of 68 (in case of apple and garlic) to a maximum of 100 (in case of grapes and peas).

3. Methodology

3.1 Text Recognition and extraction using Paddle OCR

In our methodology, in the grocery dataset, we utilized OCR methods through Keras and Paddle to detect and recognize text from labeled ingredients within the dataset. Keras facilitated direct annotations on images, while Paddle presented information in a convenient list format for streamlined processing. A significant challenge we encountered was the dataset being in German, requiring translation to English for compatibility with our tools. Additionally, we converted HEIC images from iPhones to jpg/png using the HEIC2PNG library to ensure compatibility with Paddle OCR, enabling accurate recognition and labeling of ingredients. To ensure logical recommendations, we extracted keywords specific to ingredients by comparing translated English text with the dataset's ingredient column using Python's set method. This enabled the identification of common elements, further processed after converting all extracted text to lowercase for easier mapping. We implement a sliding window approach on a separate dataset containing fruits and vegetables, albeit encountering memory issues.

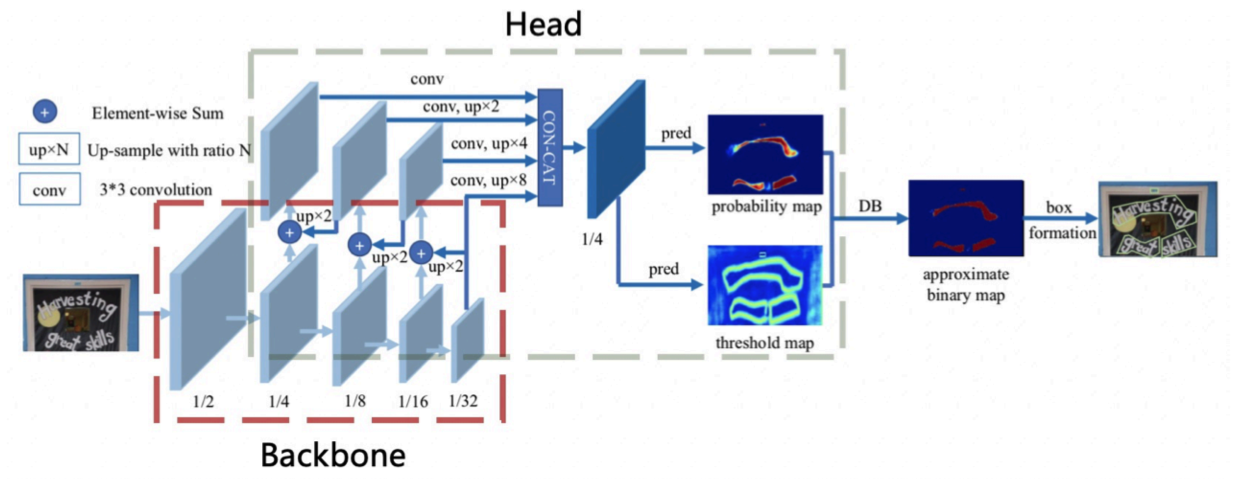


Fig 3: Paddle OCR Architecture

3.2 Image Classification using Vision Transformer

Vision Transformer (ViT) is a transformer model that is based on an encoder-only architecture. Although transformers are traditionally used for Natural Language Processing tasks, ViT has emerged as a strong alternative for state-of-the-art CNN-based models with its enhanced computational efficiency and accuracy. This algorithm applies a transformer-like architecture on patches of fixed-size images along with linear and positional embeddings to produce highly competitive benchmarks for computer vision applications.

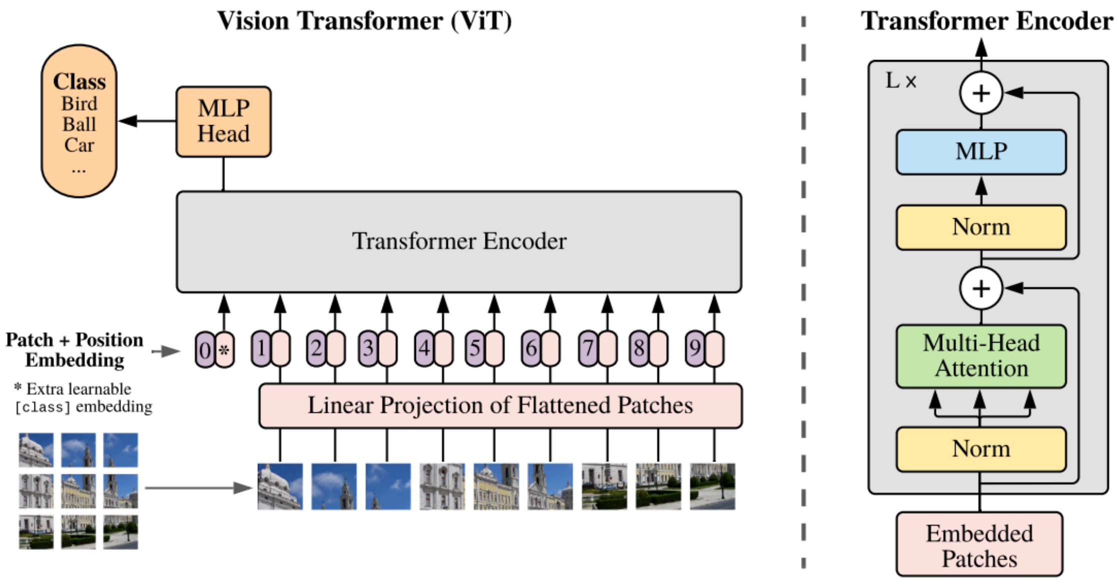


Fig 4: Vision Transformer Architecture

As seen in the figure above, 2D images of resolution HxWxC are reshaped into N number of flatten 2D patches of resolution PxP such that $N = HW/P^2$. This is followed by learnable linear projection that outputs patch embedding. To retain positional information, position embeddings are added to patch embeddings. This creates a sequence of token embeddings that are effective inputs to a standard transformer encoder. The encoder block is made up of alternate layers of multihead self-attention block and MLP block with layer norm applied before and residual connections applied after every block.

3.3 Image Classification using Convolutional Neural Network Models

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily used for processing and analyzing visual data, such as images and videos. They have become the backbone of various computer vision tasks due to their ability to automatically learn hierarchical patterns and features directly from raw pixel data. They are inspired by the visual cortex of the human brain, where neurons respond to overlapping regions of the visual field. Just like the human brain, CNNs apply convolutional operations to extract local features from input images, which are then aggregated and processed hierarchically to recognize similar or more complex patterns.

3.3.1 VGG 16

VGG16 comprises a series of convolutional layers followed by max-pooling layers, gradually reducing spatial dimensions while increasing the depth of feature maps. The VGG16 architecture consists of 16 weight layers, including 13 convolutional layers and 3 fully connected layers. The input images, typically of size HxWxC, are passed through convolutional layers with small receptive fields (3x3) and a stride of 1, followed by rectified linear unit (ReLU) activation functions. Max-pooling layers with a 2x2 window and a stride of 2 are applied to downsample feature maps, preserving the most important information.

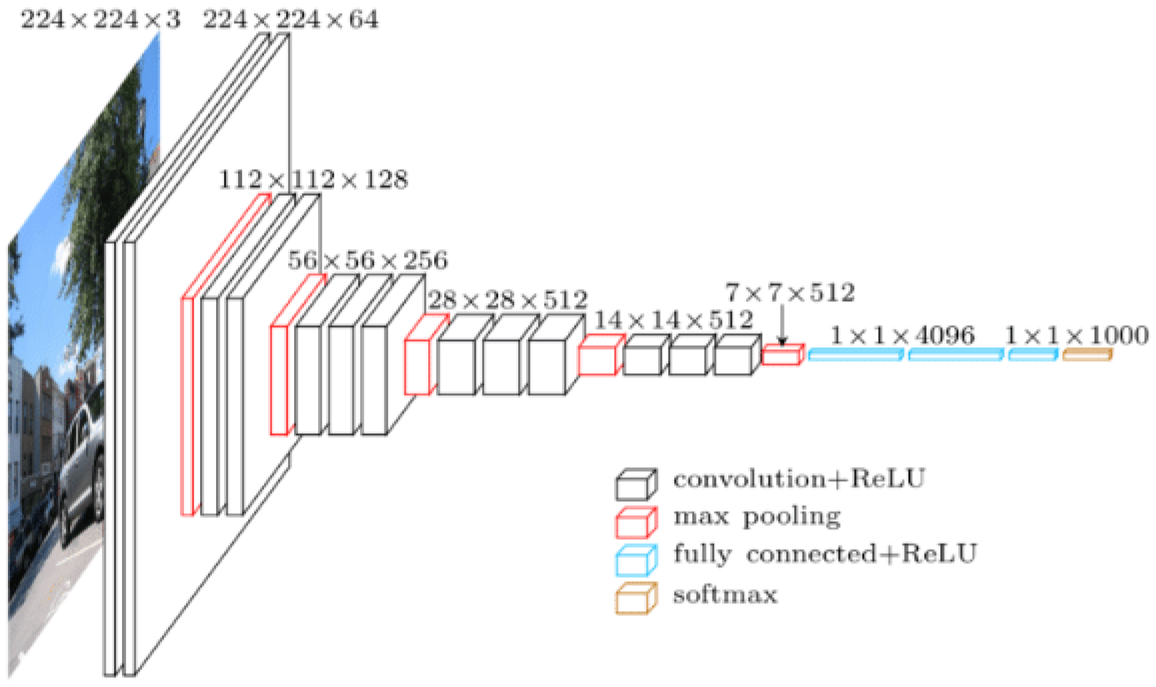


Fig 5: VGG16 Architecture

3.3.2 DenseNet

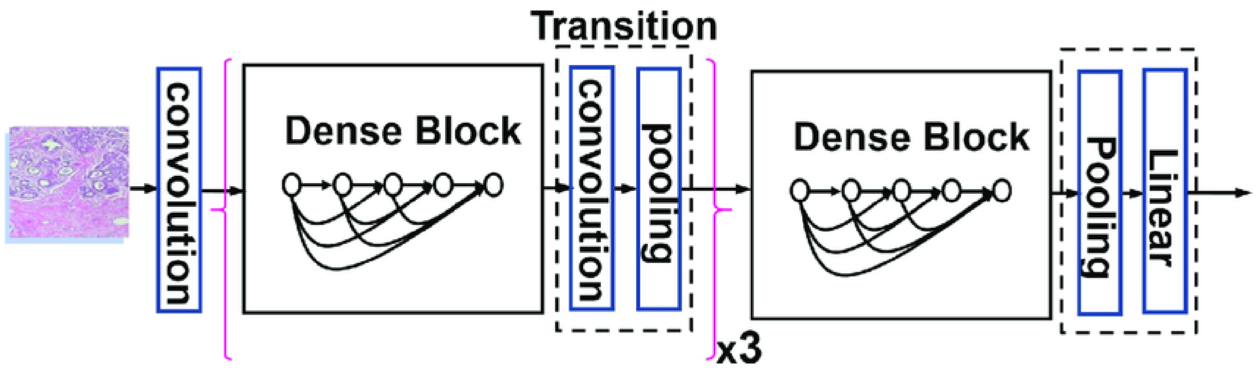


Fig 6: DenseNet121 Architecture

DenseNet121 is a deep CNN architecture known for its dense connectivity pattern. Unlike traditional CNN architectures where each layer is connected only to the subsequent layer, it establishes direct connections from each layer to every other layer in a feed-forward fashion. The input images of size HxWxC are processed through multiple dense blocks, each containing several convolutional layers. Within each dense block, feature maps from all preceding layers are concatenated along the channel dimension, allowing for direct information flow throughout the network.

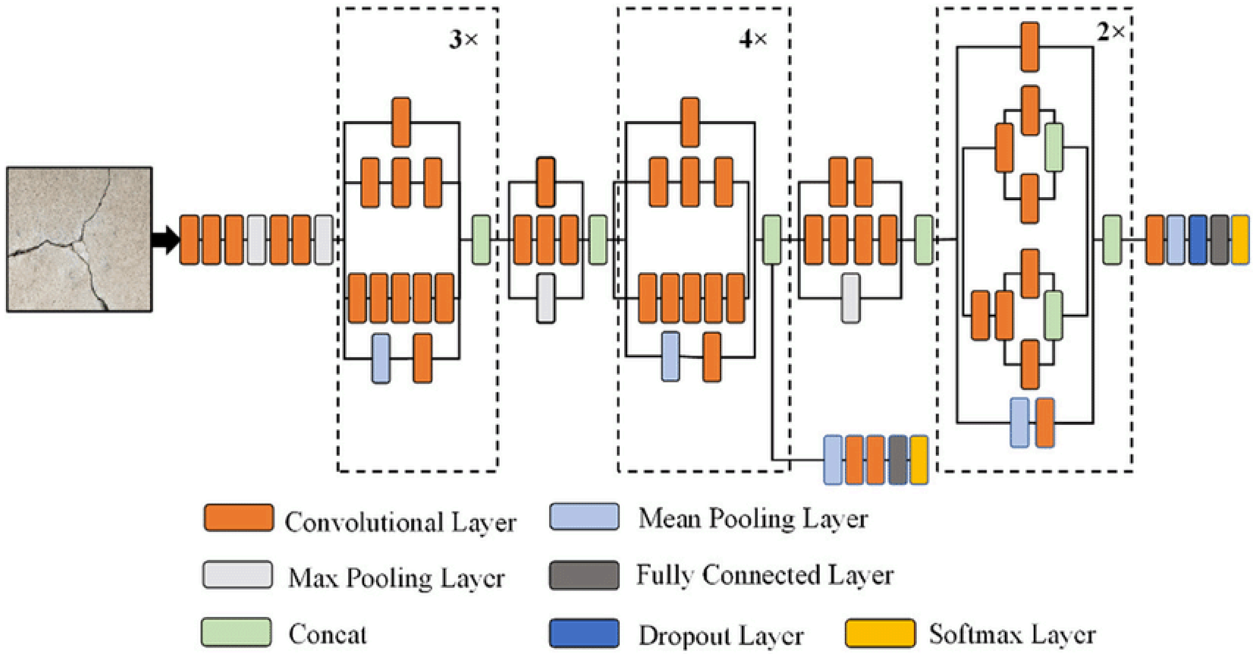


Fig 7: InceptionV3 Architecture

3.3.3 - Inception V3

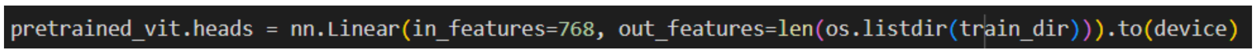
InceptionV3 is a convolutional neural network architecture renowned for its Inception modules. Unlike traditional convolutional layers, which rely on fixed-size filters. Inception modules employ filters of multiple sizes (1x1, 3x3, and 5x5) in parallel, enabling the network to capture features at various spatial scales. The input images, typically of size HxWxC, are first processed through convolutional layers and pooling operations to extract hierarchical features. Inception modules are then strategically placed throughout the network, facilitating efficient feature extraction and dimensionality reduction.

4. Analysis

4.1 Fine tuning Vision Transformers for image classification

According to the paper 'An Image Is Worth 16x16 Words' , ViT has been pre-trained on large datasets and is transferable on mid-sized to small image recognition benchmarks like ImageNet, CIFAR-100, VTAB, etc for downstream classification tasks. For the purpose of our analysis, we have followed a similar methodology using transfer learning to fine-tune the pre-trained Vision Transformer model to classify image inputs into 36 different classes of fruits and vegetables.

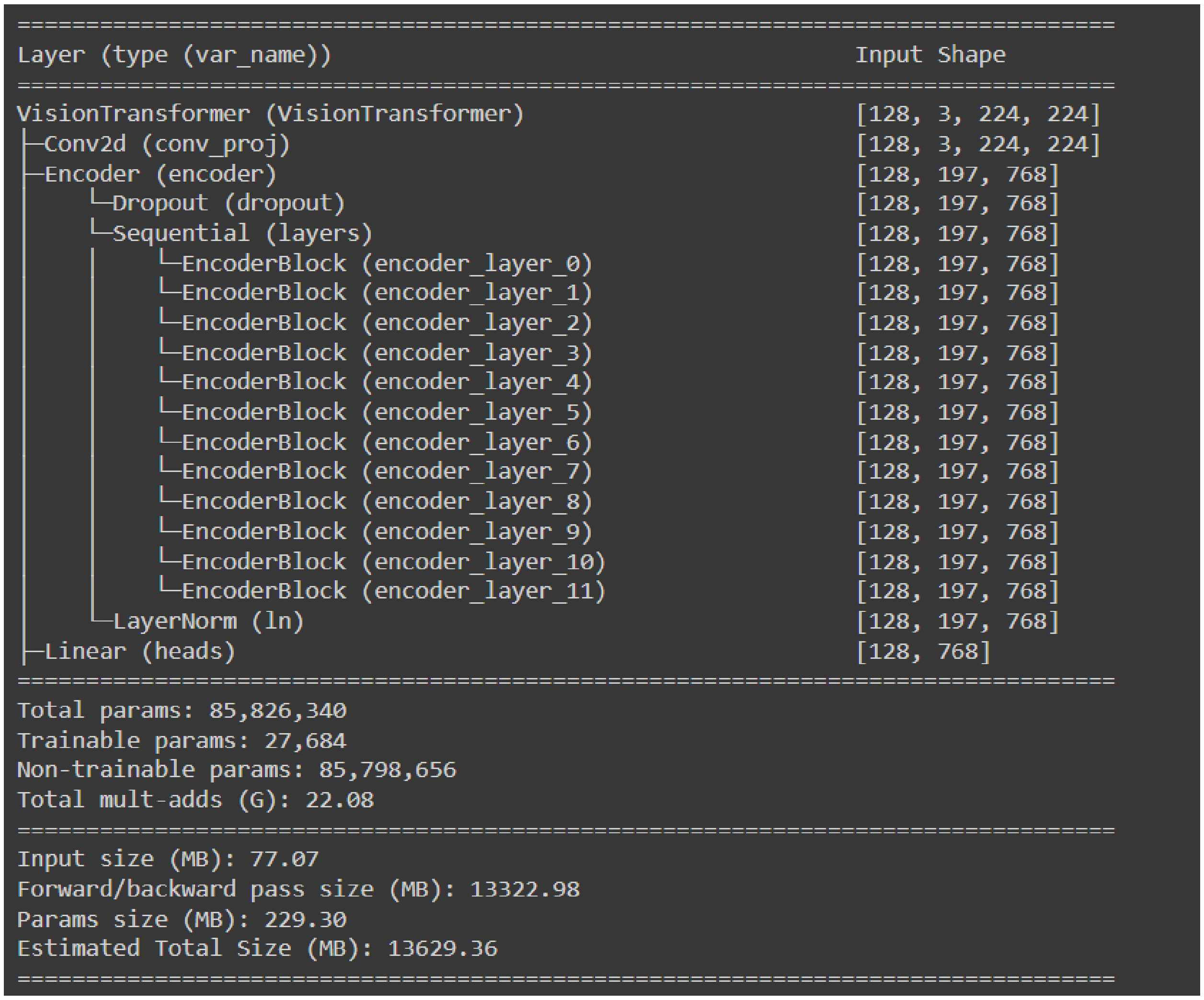
We begin by removing the pre-trained prediction head and initializing a Dx36 feedforward layer with zero weights.



```
pretrained_vit.heads = nn.Linear(in_features=768, out_features=len(os.listdir(train_dir))).to(device)
```

Fig 8: Initializing training head for ViT base model

We have used the ViT base model with 86M parameters as summarized below.



```
==========================================================================================
Layer (type (var_name))                                          Input Shape
==========================================================================================
VisionTransformer (VisionTransformer)                            [128, 3, 224, 224]
├─Conv2d (conv_proj)                                             [128, 3, 224, 224]
├─Encoder (encoder)                                              [128, 197, 768]
│    └─Dropout (dropout)                                         [128, 197, 768]
│    └─Sequential (layers)                                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_0)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_1)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_2)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_3)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_4)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_5)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_6)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_7)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_8)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_9)                       [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_10)                      [128, 197, 768]
│    │    └─EncoderBlock (encoder_layer_11)                      [128, 197, 768]
│    └─LayerNorm (ln)                                            [128, 197, 768]
├─Linear (heads)                                                 [128, 768]
==========================================================================================
Total params: 85,826,340
Trainable params: 27,684
Non-trainable params: 85,798,656
Total mult-adds (G): 22.08
==========================================================================================
Input size (MB): 77.07
Forward/backward pass size (MB): 13322.98
Params size (MB): 229.30
Estimated Total Size (MB): 13629.36
==========================================================================================
```

Fig 9 : ViTBase model instance as used for training image classification model for "Fruits and Vegetables Dataset"

4.1.1 Data Augmentation

The following data augmentation techniques were used to train the model:

1. torchvision.transforms.RandomHorizontalFlip(p=0.5) - Horizontally flip the given image randomly with a given probability
2. torchvision.transforms.RandomVerticalFlip(p=0.5) - Vertically flip the given image randomly with a given probability.
3. torchvision.transforms.RandomRotation(degrees, interpolation=InterpolationMode.NEAREST, expand=False, center=None, fill=0) -Rotate the image by angle of 40 degrees
4. torchvision.transforms.Resize(size, interpolation=InterpolationMode.BILINEAR, max_size=None, antialias=True) - Resize the input image to the (224,224) with bilinear interpolation
5. torchvision.transforms.Normalize(mean, std, inplace=False) - Normalize a tensor image with mean and standard deviation of 0.5

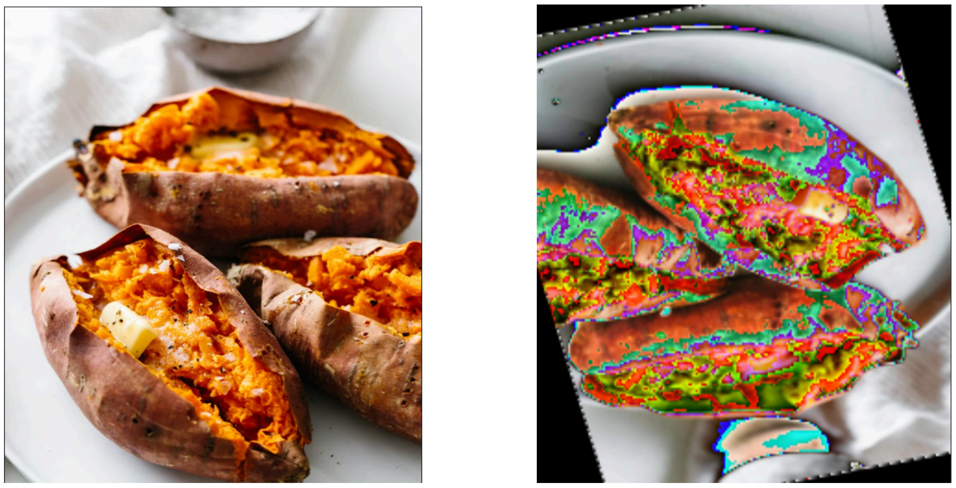


Figure 10 : Image transformation after applying data augmentation as mentioned in above - Original image (left) vs transformed image (right)

4.1.2 Model Training

Model was trained for 30 epochs with batch size of 128 and the following parameters:

1. Optimizer - torch.optim.Adam with a learning rate of $1e^{3}$
2. Loss function - torch.nn.CrossEntropyLoss()

The train and test loss for the final epoch was approximately 0.1901. We obtained accuracy of 94.59% on the train set and 95.06% accuracy on the test in the final epoch. The training loss and accuracy changes over the 30 epochs has been shown in the plots below:

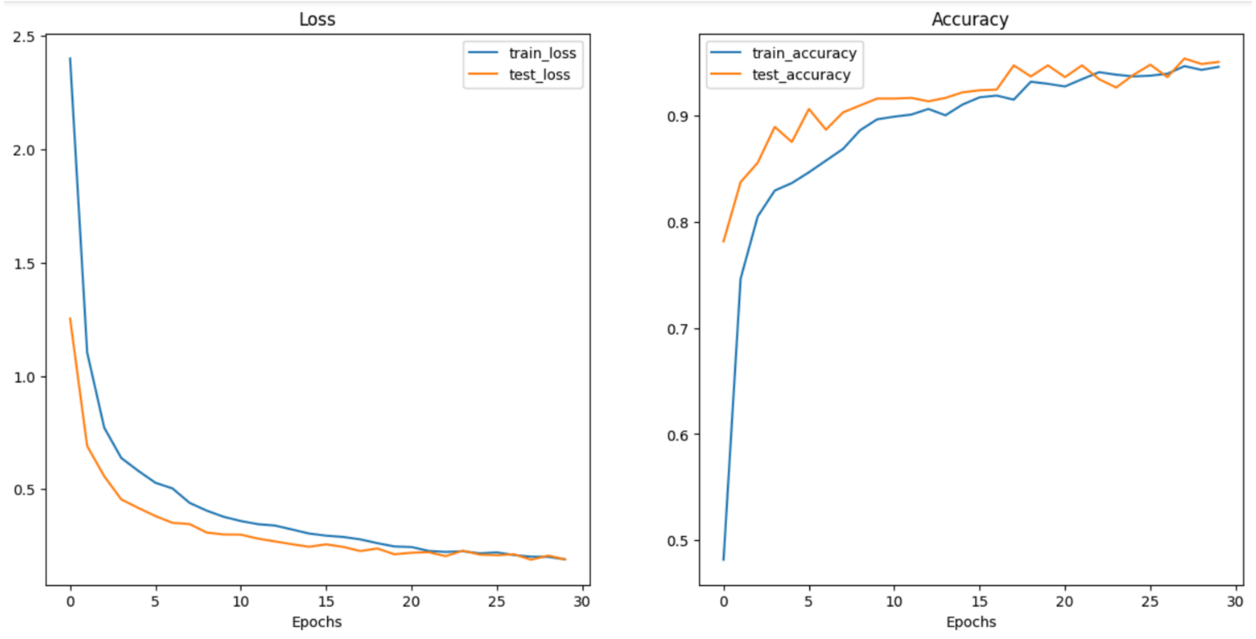


Fig 11 : Summarizing Training ViT base model on ‘Fruits and Vegetables Dataset’

4.2 Fine tuning CNNs for image classification

According to established practices, VGG16, DenseNet121, and InceptionV3 have been pre-trained on large-scale datasets such as ImageNet. In alignment with this methodology, we have employed transfer learning by initializing the pre-trained models without their final classification layers. Specifically, we remove the pre-trained prediction heads and replace them with custom feedforward layers tailored to our classification task. This approach leverages the weights and biases in the pre-trained models' feature representations while allowing for fine-tuning on our specific classification problem. After initializing the pre-trained models without their final classification layers, we augment them with custom feedforward layers tailored to our specific classification task. In our implementation, we utilize a Flatten layer followed by a Dense layer(s). This is then connected to an output layer consisting of 36 units, aligns with our classification problem where we aim to categorize input images into 36 distinct classes of fruits and vegetables.

4.2.1 Data Augmentation

Due to the unavailability of enough images for each class, image augmentation techniques were used to increase the dataset size significantly.

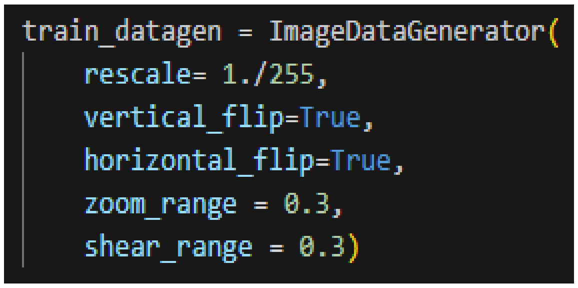

```
train_datagen = ImageDataGenerator(
    rescale= 1./255,
    vertical_flip=True,
    horizontal_flip=True,
    zoom_range = 0.3,
    shear_range = 0.3)
```

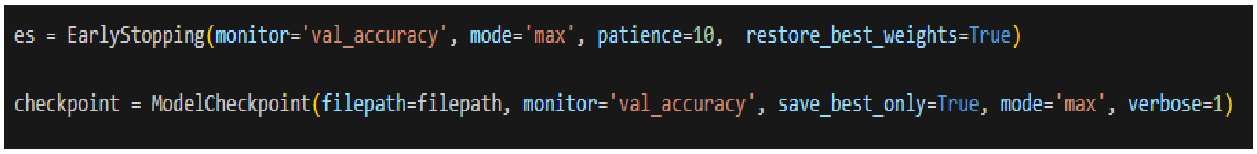
Fig 12 : Image Augmentation technique

4.2.2 Model Training

The models were trained using the following callback functions:-

1. EarlyStopping() - The model is monitored whether the validation accuracy is increasing with the number of epochs.
2. ModelCheckPoint() - Saves best model weights and prevents underfitting or overfitting.

By using these techniques, an infinite number of epochs can be set, and the training will automatically stop when best weights are recorded.


```
es = EarlyStopping(monitor='val_accuracy', mode='max', patience=10,  restore_best_weights=True)

checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)
```

Fig 13 : Callback functions

Number of epochs required for each model:-

1. VGG16 - Early Stopping function halted the training after 55 epochs.
2. DenseNet121 - Early Stopping function halted the training after 333 epochs.
3. InceptionV3 - Early Stopping function halted the training after 361 epochs.

4.3 Ensemble Learning

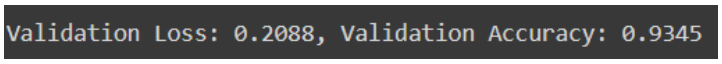
We have used ensemble learning to combine the predictions from three CNN based models mentioned above to enhance overall performance. In our approach, we have employed a technique known as ensemble averaging, where predictions from three pre-trained models - VGG16, DenseNet121, and InceptionV3 - are combined to form a consolidated prediction.

Each individual model independently processes the input image and generates its own set of predictions, represented as probabilities for each class. These predictions are then aggregated using ensemble averaging, where we extract the maximum probability for each class across all models.

5. Analysis

5.1 Vision Transformer Image Classification Results

We have used our final model to obtain classification results on the validation (with 360 images - 10 images per class). The validation loss from the final model is 0.2088. The model predicts different classes of fruits and vegetables with an accuracy of 93.45%.



```
Validation Loss: 0.2088, Validation Accuracy: 0.9345
```

Fig 14 : Results on the validation set

We have provided the following images (sourced from the internet and not a part of training or validation set) as an input to the model to obtain more interpretable results. The following images show the predicted class alongside the probability of obtaining the predicted classes.

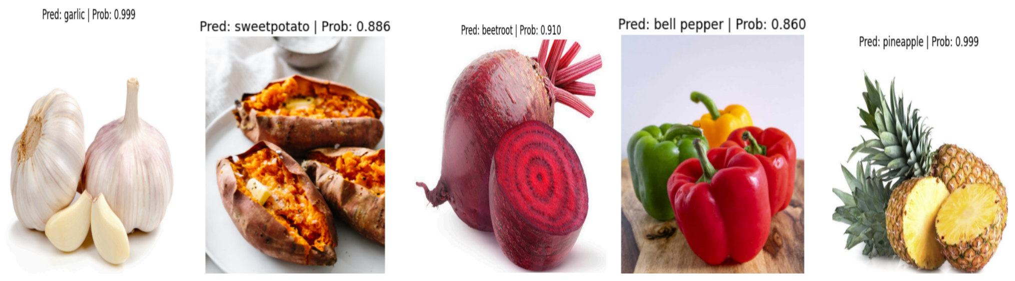

Pred: garlic | Prob: 0.999
Pred: sweetpotato | Prob: 0.886
Pred: beetroot | Prob: 0.910
Pred: bell pepper | Prob: 0.860
Pred: pineapple | Prob: 0.999

Fig 15 : Images of fruits and vegetables labeled with the prediction and prediction probabilities

We have also made a visual comparison of different kinds of images for fruits and vegetables belonging to the same class to understand the scope of our model over a given range. The results for classes potato and carrot have been highlighted in the figure below.

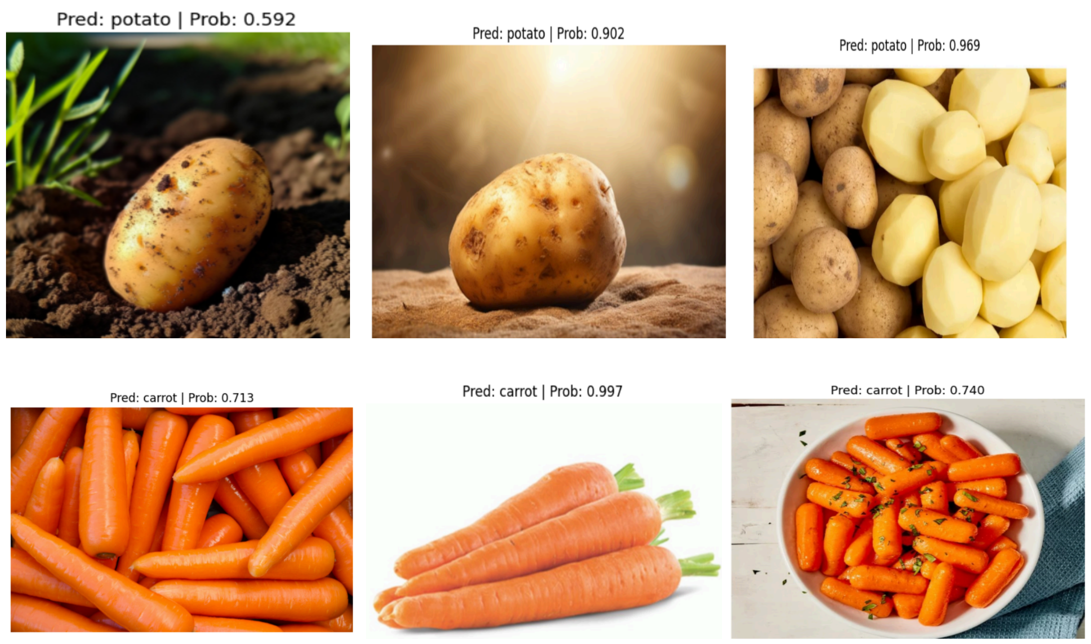
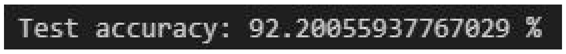

Pred: potato | Prob: 0.592
Pred: potato | Prob: 0.902
Pred: potato | Prob: 0.969
Pred: carrot | Prob: 0.713
Pred: carrot | Prob: 0.997
Pred: carrot | Prob: 0.740

Fig 16 : Testing model for different types of images of the same class to test the visual range of the predictive power of the model - testing class potatoes and carrots

5.2 Ensemble Model Image Classification Results

The following accuracy results after evaluating the individual models on the validation set.

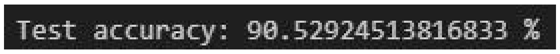
5.2.1 VGG 16 - The VGG 16 model provides an accuracy of 92.20% on the validation set after training it for 55 epochs.



```
Test accuracy: 92.20055937767029 %
```

Fig 17 : Validation accuracy of VGG16

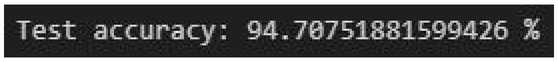
5.2.2 DenseNet121 - DenseNet121 model gives an accuracy of 90.53% on the validation set after training it for 55 epochs.



```
Test accuracy: 90.52924513816833 %
```

Fig 18 : Validation accuracy of DenseNet121

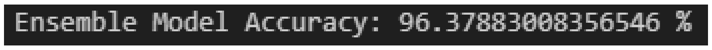
5.2.2 Inception V3 - The highest accuracy of 94.71% was achieved by training Inception V3 model for 361 epochs



```
Test accuracy: 94.70751881599426 %
```

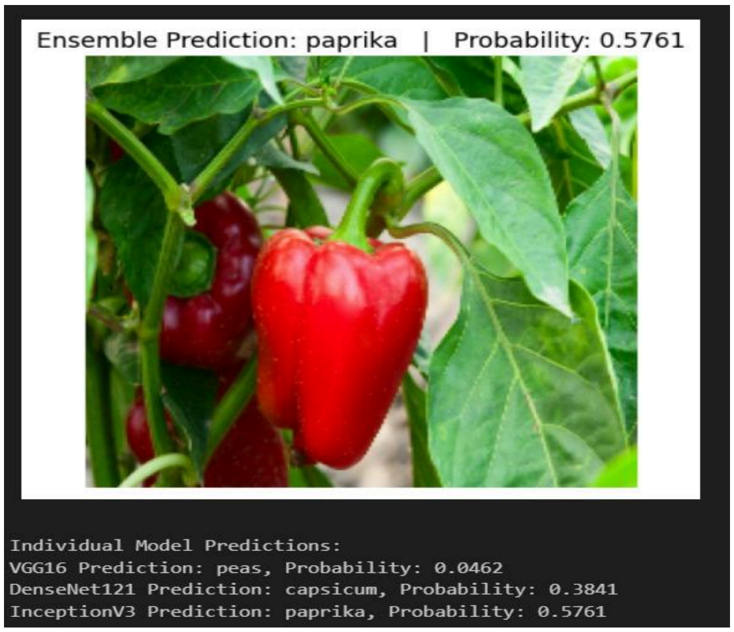
Fig 19 : Validation accuracy of InceptionV3

Using an ensemble of the three CNN models ,we have achieved an overall accuracy of 96.38% on the validation section dataset as shown below.



```
Ensemble Model Accuracy: 96.37883008356546 %
```

Fig 20 : Accuracy after Ensemble Learning



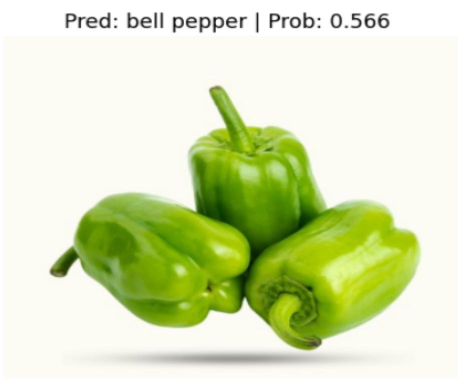
```
Individual Model Predictions:
VGG16 Prediction: peas, Probability: 0.0462
DenseNet121 Prediction: capsicum, Probability: 0.3841
InceptionV3 Prediction: paprika, Probability: 0.5761
```

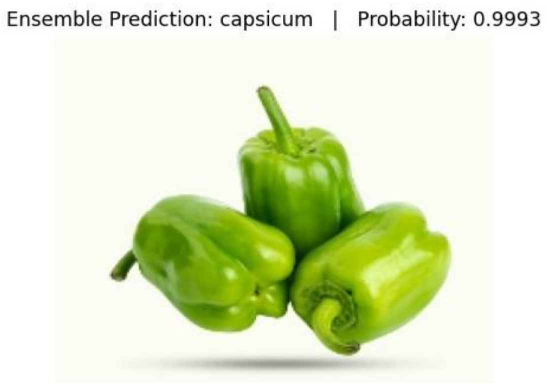
Fig 21 : Superior prediction accuracy for ensemble model

The above image shows the three models working in the ensemble. Here we notice the individual probabilities of each model along with their predictions. Even though VGG16 and DenseNet121 were making a wrong prediction, InceptionV3 successfully made a correct prediction.

5.3 Model Comparison and Selection

Although the Vision Transformer model is providing a higher accuracy on the validation set after training for only 30 epochs, it fails to beat the ensemble of the three CNN based models providing an overall increase in accuracy by approximately 2%. We have also tried to compare the accuracy results on individual images sourced from the internet as shown below.





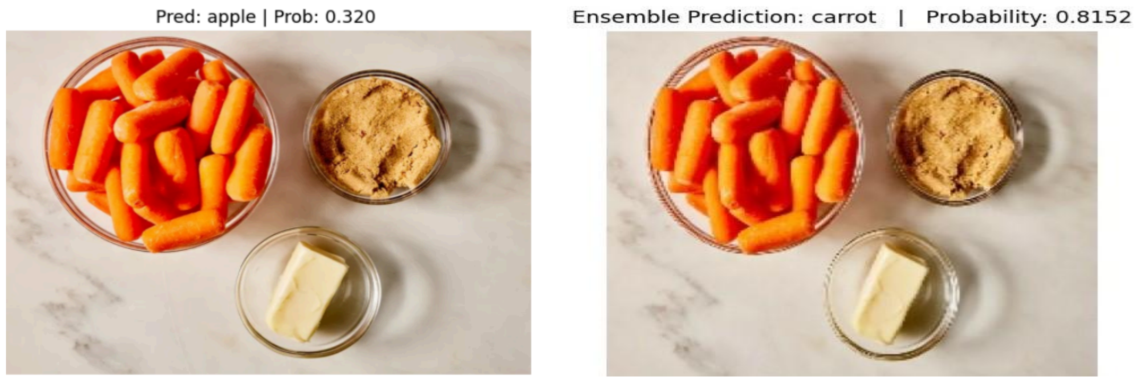

Pred: apple | Prob: 0.320
Ensemble Prediction: carrot | Probability: 0.8152

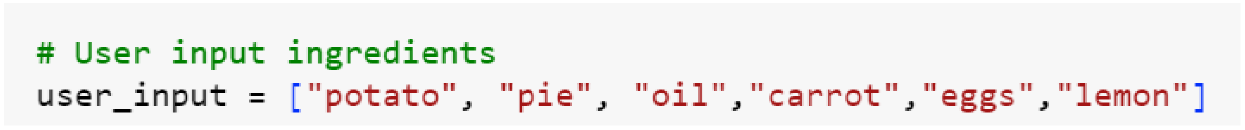
Fig 22 : Comparison of prediction of ViT (on the left) and Ensemble (on right) on the same images

From the above image comparison we note that the ViT model fails to recognize intricate differences in images like distinguishing between "bell pepper" and "capsicum" while the ensemble model excels at it. Furthermore in the second case where the fruit or vegetable is only a small part of the entire image the ensemble model recognizes it correctly with 81.5% probability whereas the ViT image completely mis-predicts it.

5.3 Paddle OCR and MultiLingual text

We utilized OCR methods such as Keras and Paddle to effectively detect and recognize text from labeled ingredients on bottles/cans within the dataset. Due to the annotations being in image format, we opted for Paddle OCR, which conveniently provided the extracted text in a list format, facilitating further processing. Overcoming the obstacle of the dataset being in German, we implemented translation methods to ensure compatibility with our tools. Additionally, to resolve compatibility issues with Paddle OCR, we successfully employed the HEIC2PNG library to convert HEIC images from iPhones to jpg/png formats. Furthermore, to ensure the logical correctness of our recommendations, we extracted specific keywords related to ingredients by comparing translated English text with dataset columns using Python's set method. This approach efficiently identified common elements. Lastly, for streamlined mapping, we converted all extracted text to lowercase. Through this meticulous process, we derived accurate insights to address our research question, supported by relevant visualizations and outputs from top-performing ML algorithms.

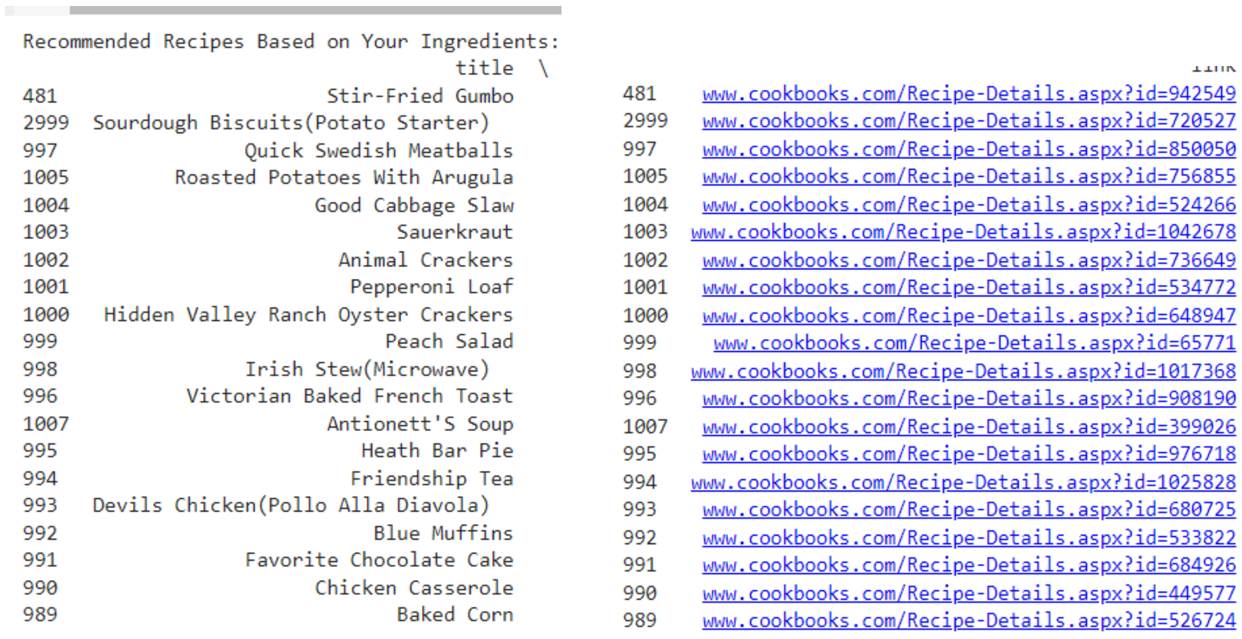
5.4 TF-IDF model and matrix results



```
# User input ingredients
user_input = ["potato", "pie", "oil","carrot","eggs","lemon"]
```

Fig 23 : Sample User input

Fig 22 : Recommended Recipes based on user input



```
Recommended Recipes Based on Your Ingredients:
                                        title  \
481                          Stir-Fried Gumbo
2999     Sourdough Biscuits(Potato Starter)
997                    Quick Swedish Meatballs
1005              Roasted Potatoes With Arugula
1004                         Good Cabbage Slaw
1003                                Sauerkraut
1002                           Animal Crackers
1001                             Pepperoni Loaf
1000         Hidden Valley Ranch Oyster Crackers
999                                Peach Salad
998                       Irish Stew(Microwave)
996                 Victorian Baked French Toast
1007                            Antionett'S Soup
995                             Heath Bar Pie
994                             Friendship Tea
993      Devils Chicken(Pollo Alla Diavola)
992                               Blue Muffins
991                     Favorite Chocolate Cake
990                          Chicken Casserole
989                                 Baked Corn

                                                  link
481    www.cookbooks.com/Recipe-Details.aspx?id=942549
2999   www.cookbooks.com/Recipe-Details.aspx?id=720527
997    www.cookbooks.com/Recipe-Details.aspx?id=850050
1005   www.cookbooks.com/Recipe-Details.aspx?id=756855
1004   www.cookbooks.com/Recipe-Details.aspx?id=524266
1003  www.cookbooks.com/Recipe-Details.aspx?id=1042678
1002   www.cookbooks.com/Recipe-Details.aspx?id=736649
1001   www.cookbooks.com/Recipe-Details.aspx?id=534772
1000   www.cookbooks.com/Recipe-Details.aspx?id=648947
999     www.cookbooks.com/Recipe-Details.aspx?id=65771
998   www.cookbooks.com/Recipe-Details.aspx?id=1017368
996    www.cookbooks.com/Recipe-Details.aspx?id=908190
1007   www.cookbooks.com/Recipe-Details.aspx?id=399026
995    www.cookbooks.com/Recipe-Details.aspx?id=976718
994   www.cookbooks.com/Recipe-Details.aspx?id=1025828
993    www.cookbooks.com/Recipe-Details.aspx?id=680725
992    www.cookbooks.com/Recipe-Details.aspx?id=533822
991    www.cookbooks.com/Recipe-Details.aspx?id=684926
990    www.cookbooks.com/Recipe-Details.aspx?id=449577
989    www.cookbooks.com/Recipe-Details.aspx?id=526724
```

6. Conclusion

With highly accurate results achieved through a fine-tuned vision model and advanced OCR capabilities, augmented by intelligent recipe recommendations driven by NLP techniques, this holistic solution enhances the grocery experience by enabling integrating item identification with personalized culinary guidance. It promises enhanced culinary exploration and convenience for consumers.

The primary challenge faced in creating and integrating the above models was a dearth of computational resources. With a greater availability of computational capabilities, we can train a customized OCR model for our specific needs. Additionally, we can think about training the Vision Transformer model further to test for improvement in predictive accuracy. As a future scope, we can add more classes of fruits and vegetables to give our application an universal appeal. Finally, we can shift our focus from single item images (one at a time approach) to multiple ingredients being identified and classified from one image through simultaneous exploration of object detection and object classification algorithms.

About the Authors

Anurima Saha (asaha8669@sdsu.edu) Anurima Saha is a graduate student pursuing her Master's in Big Data Analytics at San Diego State University. For the PIXEL-TO-PLATE project, she worked on the fruits and vegetables dataset, leveraging vision transformers for training the image classification model using transfer learning.. With a strong background in machine learning and computer vision, Anurima brings her expertise to develop innovative solutions combining cutting-edge technology with practical applications.

Sayandip Pal (spal6554@sdsu.edu) Sayandip Pal is a Master's student in the department of Electrical and Computer Engineering at San Diego State University. His role in the project involved working on the fruits and vegetables dataset alongside Anurima, utilizing an ensemble of the CNN models for image recognition tasks. Sayandip's interests lie in leveraging data mining and deep learning techniques to solve real-world problems.

Sai Tejasri (syerramsetti9523@sdsu.edu) Sai Tejasri is pursuing her Master's in Big Data Analytics at San Diego State University. Along with Radhika, she worked on the grocery dataset, employing Paddle and Keras OCR for text recognition from images. With a passion for cooking, dancing, socializing, and sustainable practices, Sai Tejasri aims to create a user-friendly recipe recommendation system that efficiently utilizes available ingredients.

Radhika Ravindra (rravindra0463@sdsu.edu) Radhika Ravindra is a graduate student in the Big Data Analytics program at San Diego State University. She collaborated with Sai Tejasri on the project, focusing on the grocery dataset and utilizing Paddle and Keras OCR for text extraction from images. Radhika explores innovative ways to apply data science techniques, developing efficient data pipelines for multimodal data.

Prit Chakalasiya (pchakalasiya9736@sdsu.edu) Prit Chakalasiya is pursuing his Master's in Big Data Analytics at San Diego State University. With a strong background in software engineering and deep learning, he was responsible for developing the recipe recommendation system for the project. Prit aims to create user-friendly applications that simplify daily tasks through intelligent solutions.

Project source code -

https://github.com/saitejasri1/Shared-ML-project

References

[1]Marín, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., & Torralba, A. (2019). Recipe 1M+: A dataset for learning cross-modal embeddings for cooking recipes and food images. IEEE Transactions on Pattern Analysis and Machine Intelligence. https://doi.org/10.1109/TPAMI.2019.2937218

[2]Kul, S., & Sayar, A. (2022). A smart recipe recommendation system based on image processing and deep learning. In S. Misra et al. (Eds.), Innovations in Smart Cities Applications Volume 5 (pp. 1023-1033). Springer. https://doi.org/10.1007/978-3-030-94191-8_83

[3]Rokon, M. S. J., Morol, M. K., Hasan, I. B., Saif, A. M., & Khan, R. H. (2022). Food recipe recommendation based on ingredients detection using deep learning. arXiv. https://arxiv.org/abs/2203.06721

[4]Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I., & Torralba, A. (2017). Learning cross-modal embeddings for cooking recipes and food images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3020-3028). https://openaccess.thecvf.com/content_cvpr_2017/html/Salvador_Learning_Cross-Modal_Embeddings_CVPR_2017_paper.html

[5]Reigues, L., Fidalgo, F., & Oliveira, . (2023). A food image to recipe retrieval system using deep learning. Applied Sciences, 13(13), 7880. https://doi.org/10.3390/app13137880

[6] Seth K. , Fruits and Vegetables Dataset. Kaggle. https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition

[7] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[8] RecipeNLG dataset source - https://recipenlg.cs.put.poznan.pl/

[9] Densely Connected Convolutional Networks, by Gao Huang, Zhuang Liu, Laurens van der Maaten and Kilian Q. Weinberger

[10] Very Deep Convolutional Networks For Large-Scale Image Recognition, by Karen Simonyan & Andrew Zisserman , Visual Geometry Group, Department of Engineering Science, University of Oxford

[11] Rethinking the Inception Architecture for Computer Vision, by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens and Zbigniew Wojna