JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A SURVEY ON DEEP LEARNING DRIVEN FOOD RECOGNITION FOR ACCURATE DIETARY ASSESSMENT

[1]G. Venkata Subbarao, [2]D. Vara Lakshmi, [3]K. Bhuvana Chandra, [4]V.R.Satya Charan, [5]R. L.N.V Sai Pavanu

[1]Asst.Prof, [2,3,4,5]U.G
Department of Computer Science and Technology,
Sasi Institute of Technology and Engineering, Tadepalligudem, Andhra Pradesh, India

Abstract : Accurate dietary assessment is critical for maintaining health and managing various medical conditions. Traditional dietary tracking methods, however, are often time-consuming and rely on self-reported data, which can be inaccurate. This paper presents a deep learning-driven approach to food recognition that leverages advanced neural network architectures for automatic food identification and quantification. Using convolutional neural networks (CNNs) and transfer learning, we develop a model capable of recognizing a wide range of food items with high accuracy. Our model is trained on a diverse dataset to improve robustness across different food types, preparation styles, and lighting conditions. Experiments demonstrate that the proposed system achieves competitive accuracy compared to state-of-the-art methods and offers a feasible solution for real-time dietary assessment applications. The integration of this system with mobile platforms and health monitoring tools holds promise for enhancing user engagement in dietary tracking, ultimately contributing to better health outcomes. Accurate dietary assessment is a critical component of health management, particularly for individuals with specific nutritional requirements, such as those with chronic diseases, athletes, and the general population aiming to maintain a healthy lifestyle. Traditional methods of dietary tracking, such as food diaries and manual logging, are often prone to errors, whether through underreporting, miss estimation, or simply forgetting to record meals.

IndexTerms **-** Deep Learning, Food Recognition, Dietary Assessment, Computer Vision, Image Processing, Neural Networks, Convolutional Neural Networks (CNNs), Nutrition Analysis, Artificial Intelligence (AI), Smart Healthcare, Food Classification.

I. INTRODUCTION

Deep learning has shown remarkable potential in image recognition tasks, and when applied to food recognition, it can significantly streamline the process of dietary tracking. By analyzing food images captured by mobile devices, deep learning models can identify food types, estimate portion sizes, and compute nutritional values with impressive accuracy. This technology not only reduces the cognitive and manual workload for users but also offers a scalable solution for healthcare professionals and nutritionists to monitor and provide dietary recommendations more effectively. Dietary assumptions for this project are based on the typical nutritional compositions and portion sizes commonly consumed across various cultures. The model aims to recognize a broad spectrum of foods, from everyday meals to specialized diets, ensuring its versatility in assessing both general and personalized nutritional needs. For individuals managing specific conditions, such as diabetes or cardiovascular diseases, accurate food recognition can help ensure adherence to dietary restrictions and improve health outcomes. Similarly, athletes can benefit from precise dietary tracking to optimize their nutrition intake, enhance performance, and monitor recovery. The need for such technology is underscored by the rising prevalence of diet-related health issues, such as obesity, malnutrition, and chronic diseases like diabetes. A reliable and efficient food recognition system can empower individuals to make informed decisions about their diet, reducing the risk of such conditions and supporting better long-term health outcomes. Furthermore, healthcare systems can benefit from the aggregated data for population-wide nutritional studies, identifying trends, and implementing targeted health interventions. In addition to its individual benefits, deep learning-driven food recognition has broader implications for public health. It can facilitate more accurate and comprehensive dietary surveys, leading to improved nutritional research and policy-making. The automatic capture and analysis of dietary data can also help bridge the gap between subjective self-reporting and objective data, providing researchers with more reliable information to study the relationship between diet and health outcomes. In summary, deep learning-driven food recognition represents a promising solution to the challenges of accurate dietary assessment. By leveraging the power of AI, it has the potential to revolutionize how individuals track their food intake, leading to better health management and more robust nutritional research. Its widespread application could not only enhance individual health monitoring

but also provide critical insights into population-level dietary habits fostering a healthier society. The increasing prevalence of diet-related health issues, such as obesity, diabetes, and cardiovascular diseases, has emphasized the importance of accurate dietary assessment in modern healthcare. Understanding what individuals eat is crucial not only for managing these conditions but also for improving overall public health. However, traditional methods of dietary assessment, such as self-reported food diaries and surveys, are often inaccurate due to human error, underreporting, or the complexity of tracking diverse food items. In response to these challenges, there is a growing need for more efficient, objective, and reliable methods for monitoring dietary intake. This is where the power of deep learning comes into play. Deep learning, a subset of artificial intelligence (AI), has demonstrated remarkable success in visual recognition tasks, making it an ideal candidate for food recognition applications. With the growing availability of food images through mobile devices, social media, and dedicated food-tracking apps, there is an opportunity to leverage these resources for more accurate dietary assessment. By training deep learning models to recognize and classify a wide variety of foods from images, we can overcome many of the limitations of traditional dietary tracking. These models can automatically identify food items, estimate portion sizes, and provide nutritional information, offering a seamless solution to dietary monitoring. The dietary assumptions underpinning this project focus on the diversity of food types, regional variations in cuisine, and the complexity of mixed dishes. To ensure the model's accuracy and reliability, it is essential to train it on a comprehensive dataset that includes a wide range of food categories, preparation methods, and portion sizes. This approach takes into account the nuances of individual dietary preferences, cultural food practices, and the challenge of differentiating visually similar food items. The ultimate goal is to provide users with personalized and precise nutritional information that can help them make informed decisions about their diet. Accurate dietary assessment has significant implications for public health, medical research, and individual wellness. For healthcare professionals, it enables better monitoring of patients' eating habits, leading to more effective treatment plans and interventions. It can also play a pivotal role in managing chronic diseases by providing real-time insights into food intake. For individuals, especially those aiming to manage weight, maintain a balanced diet, or address specific nutritional deficiencies, AI-driven food recognition offers a convenient and less invasive alternative to manual logging. Furthermore, the data collected through these systems can contribute to large-scale studies on population health, identifying trends in eating behaviors and the impact of diet on health outcomes. In addition to its potential healthcare benefits, deep learning-driven food recognition can be integrated into a wide range of consumer-facing technologies, such as mobile health apps, wearable devices, and fitness trackers. This allows users to track their meals effortlessly, receive dietary advice, and even achieve their nutritional goals more effectively. The convergence of AI and nutrition has the potential to revolutionize dietary assessment, transforming how people engage with food and making healthy eating more accessible for everyone.

II Related Works

Several deep learning architectures have been successfully applied to food image classification tasks, showcasing the effectiveness of convolutional neural networks (CNNs) in handling complex, high-dimensional image data. Food image recognition has received significant attention in recent years due to its applications in dietary monitoring, nutritional assessment, and food identification. One of the popular models, ResNet, introduced by He et al. [1], uses residual connections to enable deeper networks. ResNet-50 and ResNet-101 have been utilized in food recognition tasks, with studies demonstrating their effectiveness in accurately classifying food images. For instance, Liu et al. [2] achieved high classification accuracy on the Food-101 dataset using ResNet-based architectures, highlighting the model's robustness in complex visual tasks. MobileNet, developed by Howard et al. [3], was specifically designed for mobile and embedded devices, offering an efficient solution for food image recognition on low-resource devices. MobileNet's lightweight architecture has been used in studies to classify food images on smaller datasets, such as UEC-Food100 and ETHZ Food-475, achieving competitive performance with significantly reduced computational costs. DenseNet, proposed by Huang et al. [4], introduces dense connectivity between layers, improving information flow and feature reuse. Studies by Wu et al. [5] applied DenseNet architectures to food classification tasks, such as on the VIREO Food-172 dataset, where DenseNet-121 and DenseNet-169 demonstrated improved accuracy due to their densely connected layers, which facilitated better feature extraction. The InceptionNet family, particularly InceptionV3 and Inception-ResNet, has been widely used in food classification tasks due to its modular architecture and ability to capture multi-scale features [6]. Researchers such as Minaee et al. [7] applied Inception-based models to food classification, reporting strong performance on datasets like Food-101 and iFood-2019, where the combination of inception modules allowed the model to handle various feature sizes. Finally, EfficientNet, introduced by Tan and Le [8], scales the network width, depth, and resolution to achieve higher accuracy with optimized computational efficiency. EfficientNet-B0 and EfficientNet-B4 have been adopted in recent food recognition studies, particularly in settings where computational resources are limited [9]. On large datasets like iFood-2019, EfficientNet achieved state-of-the-art accuracy while maintaining a smaller model size. A system can be represented using this straightforward graphical formalism in terms of the input data it receives, the different operations it performs on that data, and the output data it generates. The components of the system are modeled using it. These elements consist of the system's procedure, the data it uses, an outside party that communicates with it, and the information flows within it. This illustrates the flow of information through the system and the various transformations that alter it. This method uses graphics to show how information flows and the changes made to data as it goes from input to output.

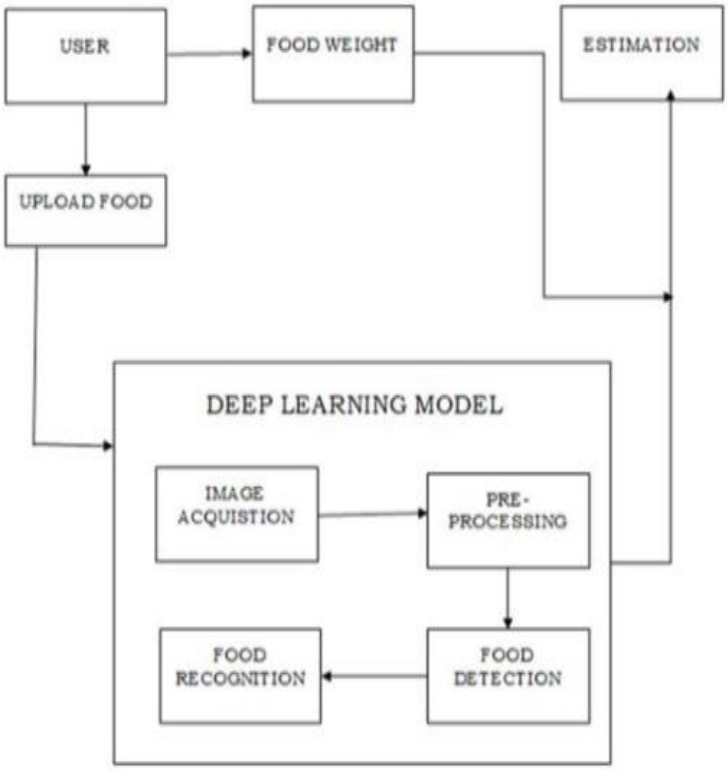

Figure 1: System Architecture

2.1.1. System Modules

- Image acquisition.
- Preprocessing.
- Feature extraction.
- Segmentation.
- Classification.

2.1.2. Modules Descriptions

Picture acquisition is the process of getting a picture from a source. To do this, hardware systems such as encoders, sensors, and databases can be utilized.
Pre-processing an image is mostly done to make the data, such an image, better by minimizing undesirable distortions or emphasizing particular features—that is, by taking out undesirable interruptions from the image.
This stage of the dimensional reduction process involves dividing and condensing an initial set of raw data into smaller, more manageable units.
It is the technique of taking an image and using labels to create an image from each pixel. This method allows you to process important areas of an image, but not the entire one.
The difficulty of accurately identifying what is depicted in the image. After being trained to
recognize various classes, the model will go through that procedure. For instance, you could teach a module to recognize the three different species in the image.

2.2. Deep Learning Based Technologies

Deep learning makes it possible for the system to identify complex food attributes in the context of food classification and calorie prediction. This improves the accuracy of identifying a variety of dishes and calculating their caloric content. By training on sizable, well selected datasets, this methodology minimizes the need for manual feature extraction and allows the model to adjust to different culinary presentations and styles.

2.2. 1. ResNet (Residual Networks)

ResNet is a deep learning architecture primarily used for image classification tasks. It uses deep residual networks with skip connections, which allow for training much deeper networks than traditional convolutional neural networks (CNNs).

2.1.2. DenseNet (Densely Connected Convolutional Networks)

DenseNet is also a deep learning architecture, where each layer is connected to every other layer in a feedforward fashion. This dense connectivity helps in improving feature reuse and making training more efficient, especially for image-related tasks.

2.2.3. InceptionNet

InceptionNet is a deep learning model based on convolutional neural networks (CNNs). It uses inception modules that help in handling various types of image features at different levels of complexity, optimizing computational cost.

2.2.4. EfficientNet

EfficientNet is a deep learning architecture that optimizes the performance of neural networks by scaling them uniformly. It has gained popularity in image classification tasks due to its ability to balance performance and computational cost.

2.2.5. MobileNet

MobileNet is a deep learning architecture designed for mobile and embedded applications. It uses depthwise separable convolutions to reduce computational costs, making it ideal for environments where computational resources are limited.

III. DATASET DETAILS

The chosen dataset is essential for this food recognition project because it provides a comprehensive and diverse collection of food images, critical for training the deep learning model to accurately identify various food types and estimate portion sizes. A good dataset includes images from different cuisines, meal presentations, and portion sizes, which helps the model generalize well across diverse dietary contexts. Additionally, a well-curated dataset often includes metadata, such as food labels, nutritional information, and serving sizes, enabling the model not only to recognize foods accurately but also to associate these foods with relevant nutritional values. This ensures that the system can perform reliable dietary assessments a cross a broad spectrum of food items, making the solution more practical and effective in real-world applications.

3.1. Food-101

The Food-101 dataset, with 101,000 images, was developed by Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool, and is widely used in food recognition research because it offers a balanced variety of 101 food categories, each with 1,000 images. This dataset, referenced as [1] Food-101 (ECCV 2014), provides a strong baseline for food classification tasks and is instrumental in training models for general food recognition, making it suitable for foundational training in this project.

3.2. UECFOOD100

UECFOOD100, developed by Y. Matsuda, H. Hoashi, and K. Yanai, includes 14,361 images across 100 categories. Referenced as [2] UECFOOD100 (IEICE Transactions 2012), it provides a compact dataset of common Japanese dishes, useful for cross-cultural studies and testing the model's ability to recognize Asian foods. This dataset aids in building cultural diversity in the training set, which is valuable for global dietary assessment applications.

3.3. UECFOOD256

Containing 31,651 images across 256 categories, UECFOOD256, by Kiyoharu Aizawa and Yuta Kawano, provides a broader set of food items with fine-grained annotations, making it essential for detailed classification tasks in complex dishes. Referenced as [3] UECFOOD256 (ACM Multimedia 2015), this dataset helps the model recognize more nuanced food items and increase classification accuracy.

3.4. FoodX-251

FoodX-251, compiled by MIT CSAIL, offers over 158,000 images of 251 classes, making it one of the largest datasets for food recognition. Referenced as [4] FoodX-251 (AAAI 2019), it covers diverse food types and scenes, contributing significantly to model robustness in recognizing food from various settings and under different lighting conditions, which is critical for real-world deployment.

3.5. VIREO Food-172

Developed by W. Chen, L. He, and S. Yang, the VIREO Food-172 dataset includes 110,241 images spanning 172 categories. As referenced in [5] VIREO Food-172 (ACM Multimedia 2016), it provides an excellent range of East Asian and Western food items, making it ideal for models intended for cross-regional dietary assessment tools.

3.6. iFood-2019

With over 118,000 images, iFood-2019 was developed for the iFood Challenge and is referenced as [6] iFood-2019 (iFood Challenge CVPR 2019). This dataset includes complex real-world food images captured in various environments. It is beneficial for training models to recognize foods in noisy, uncontrolled environments, enhancing the robustness of dietary assessment applications.

3.7. ETHZ Food-101 Extended

An extended version of the original Food-101 dataset by ETH Zurich, referenced as [7] ETHZ Extended (ETH Zurich Internal 2015), provides additional images and data diversity. This extension helps improve generalization in food recognition models, making it useful for augmenting training data and enhancing model performance.

3.8. Food-11

Containing over 250,000 images, the Food-11 dataset, developed by Chiuan-Huei Chiou and Cheng-Lin Liu, is referenced as [8] Food-11 (International Conference on Image Processing 2016). Its large size and focus on high-calorie foods make it especially valuable for calorie tracking, contributing to models that can estimate dietary intake more accurately.

3.9. Food-524

Food-524, with over 16,000 images, was developed by Yujiang Zhao and Liangjing Ding, referenced as [9] Food-524 (ImageNet Food Dataset 2019). It provides images across 524 food types, offering detailed representation for diverse dietary items. This dataset is used to enhance the model's ability to recognize a broad range of foods, aiding in specific dietary assessments.

3.10. Fruit-360

Developed by Mureșan and Pavelescu, the Fruit-360 dataset includes 247,636 images focused on fruit classification, referenced as [10] Fruit-360 (Data in Brief 2018). Its specificity is helpful in accurately identifying fruit items, which are common in dietary intake, allowing the model to distinguish fruits with high accuracy.

3.11. UNIMIB2016

With 70,000 images, UNIMIB2016, by Mauro Martinel and Alberto Baldrati, is referenced as [11] UNIMIB2016 (Pattern Recognition 2018). This dataset includes various food items in different portions, making it useful for portion size estimation and adding depth to dietary assessment models by enhancing portion recognition capabilities.

Table 1: Different datasets can be used for the food image recognition

S.No.	Dataset Name	Number of Images	Author Name	Reference Number
1	Food-101	101,000	Lukas Bossard, Matthieu Guillaumin, Luc Van Gool	[1] Food-101 (ECCV 2014)
2	UECFOOD100	14,361	Y. Matsuda, H. Hoashi, K. Yanai	[2] UECFOOD100 (IEICE Transactions 2012)
3	UECFOOD256	31,651	Kiyoharu Aizawa, Yuta Kawano	[3] UECFOOD256 (ACM Multimedia 2015)
4	FoodX-251	158,000+	MIT CSAIL	[4] FoodX-251 (AAAI 2019)
5	VIREO Food-172	110,241	W. Chen, L. He, S. Yang	[5] VIREO Food-172 (ACM Multimedia 2016)
6	iFood-2019	118,000+	iFood Challenge Organizers	[6] iFood-2019 (iFood Challenge CVPR 2019)
7	ETHZ Food-101 Extended		ETH Zurich	[7] ETHZ Extended (ETH Zurich Internal 2015)
8	Food-11	250,000+	Chiuan-Huei Chiou, Cheng-Lin Liu	[8] Food-11 (International Conference on Image Processing 2016)
9	Food-524	16,000+	Yujiang Zhao, Liangjing Ding	[9] Food-524 (ImageNet Food Dataset 2019)
10	Fruit-360	247,636	Mureșan, Pavelescu	[10] Fruit-360 (Data in Brief 2018)
11	UNIMIB2016	70,000	Mauro Martinel, Alberto Baldrati	[11] UNIMIB2016 (Pattern Recognition 2018)

IV. RESULTS AND DISCUSSIONS

4.1. ResNet

A. ResNet-50, TensorFlow

ResNet-50, a 50-layer deep convolutional neural network, is widely used in image classification and computer vision applications due to its innovative use of residual connections. These connections help address the vanishing gradient problem, allowing for effective training of very deep networks. TensorFlow, an open-source machine learning framework, provides an efficient environment for implementing ResNet-50. By leveraging TensorFlow's GPU acceleration and model optimization tools, ResNet-50 achieves high accuracy in classifying complex images and is commonly used in domains like medical image analysis, object detection, and even food recognition tasks. This combination enhances the model's capacity to handle large-scale data with precision.

B. ResNet-101, PyTorch

ResNet-101 is a 101-layer convolutional neural network that extends the ResNet architecture with additional layers, enhancing its capacity to capture intricate features in complex datasets. Like other ResNet models, ResNet-101 utilizes residual connections to mitigate the vanishing gradient problem, allowing for more effective training of deep networks. Implementing ResNet-101 in PyTorch, a popular deep learning framework, provides flexibility and ease in customizing model layers and optimizing computations, especially with its dynamic computation graph and GPU acceleration. This combination is widely used in applications requiring detailed feature extraction, such as fine-grained image classification, object detection, and medical image analysis, where high accuracy is essential.

C. ResNet-18, PyTorch

ResNet-18, a smaller variant in the ResNet family, consists of 18 layers and is designed for efficient feature extraction with a relatively low computational load. It retains the core advantage of the ResNet architecture—residual connections—to ease the training of deeper models by addressing the vanishing gradient issue. ResNet-18 is particularly well-suited for tasks where computational resources are limited or where quick inference is crucial, such as real-time applications and mobile deployments. Using ResNet-18 with PyTorch offers flexibility for modifying layers, easy integration with other machine learning workflows, and efficient GPU utilization, making it a good choice for projects requiring a balance between speed and accuracy in classification tasks.

D. ResNet-50, Keras

ResNet-50, a popular deep convolutional neural network with 50 layers, is widely used for image classification and feature extraction due to its robust architecture and balanced depth. With its residual connections, ResNet-50 effectively mitigates the vanishing gradient problem, allowing it to achieve high accuracy even with increased layers. Implementing ResNet-50 in Keras offers a straightforward and accessible platform for experimentation, thanks to Keras's user-friendly API and seamless integration with TensorFlow. This model is particularly suited for applications requiring accurate image recognition, such as medical image analysis, object detection, and food image classification.

E. ResNet-18, Keras

ResNet-18, the lighter version of the ResNet architecture with 18 layers, is optimized for tasks that require efficient processing with less computational power while maintaining accuracy. When implemented with Keras, ResNet-18 becomes especially accessible for rapid prototyping and deployment in applications like real-time image classification or object detection on mobile or edge devices. Keras's user-friendly API simplifies the integration of ResNet-18, making it ideal for scenarios where computational resources are limited but reliable performance is essential. This model is a good choice for projects needing a balance between speed and accuracy, especially for relatively simpler image classification tasks.

F. ResNet-152, Keras

ResNet-152 is a deep variant of the ResNet architecture with 152 layers, designed to handle complex image recognition tasks with high accuracy. Integrated with Keras, ResNet-152 is particularly valuable for tasks where high detail and feature extraction are crucial, such as fine-grained image classification, object detection, and transfer learning on complex datasets. Keras provides an intuitive framework to implement and fine-tune ResNet-152, facilitating its application in advanced machine learning pipelines. Although ResNet-152 requires more computational resources, it is highly effective in scenarios where accuracy and depth of feature representation are prioritized.

G. ResNet-101, Keras

ResNet-101, implemented using Keras, is a deep convolutional neural network with 101 layers, designed to address the vanishing gradient problem often encountered in very deep networks. It utilizes residual connections to enable the network to learn residual mappings instead of direct mappings, significantly improving the training process and accuracy. In Keras, ResNet-101 can be efficiently used for tasks like image classification, object detection, and segmentation. Its architecture, which includes a large

number of layers, allows it to extract rich features from images, making it highly effective in handling complex datasets and high-dimensional image data. The Keras framework simplifies the implementation and fine-tuning of ResNet-101 for diverse deep learning applications.

H. ResNet-50, PyTorch

ResNet-50, implemented using PyTorch, is a deep convolutional neural network consisting of 50 layers, designed to tackle the challenge of training very deep networks through the use of residual connections. These residual connections allow the network to learn the difference (residuals) between the input and output of each block, mitigating the vanishing gradient problem. This enables efficient training of deep networks. In PyTorch, ResNet-50 is commonly used for tasks such as image classification, object detection, and feature extraction. The model's architecture strikes a balance between depth and computational efficiency, making it suitable for real-time applications with high accuracy, especially on large-scale datasets. Additionally, PyTorch's dynamic computation graph allows for flexible model modification and optimization during training.

I. ResNet-18, TensorFlow

ResNet-18, implemented using TensorFlow, is a smaller version of the ResNet architecture with 18 layers, designed to address the challenges of training deep neural networks. It uses residual connections that help in the efficient training of deeper networks by allowing gradients to flow directly through skip connections, thus mitigating the vanishing gradient problem. ResNet-18, while simpler and less computationally expensive than deeper variants like ResNet-50 or ResNet-101, still achieves impressive performance in image classification tasks. TensorFlow's high-level API, Keras, makes it easy to implement and fine-tune ResNet-18 for various applications, such as object detection and transfer learning, by providing pre-trained models and integration with TensorFlow's powerful tools for model optimization and deployment.

J. ResNet-152, PyTorch

ResNet-152, implemented using PyTorch, is a deeper version of the ResNet architecture with 152 layers. It leverages residual connections to enable the training of very deep neural networks by allowing gradients to pass through the network without diminishing. This model is well-suited for tasks that require high accuracy, such as complex image classification and object detection, as it captures more intricate features from the input data due to its depth. In PyTorch, ResNet-152 can be easily utilized through pre-trained models from the torchvision library, providing a solid foundation for fine-tuning on domain-specific tasks. Its efficient use of computational resources, along with PyTorch's dynamic computation graph, makes it a popular choice for research and practical applications in computer vision.

4.2. DenseNet

A. DenseNet, TensorFlow

DenseNet, implemented using TensorFlow, is a deep convolutional neural network architecture characterized by its use of dense connections between layers. In DenseNet, each layer receives input from all previous layers, allowing for better feature reuse and more efficient gradient flow during training. This reduces the number of parameters compared to traditional CNNs while maintaining or improving model performance. DenseNet has been shown to achieve superior results in tasks such as image classification and segmentation, as it helps mitigate issues like vanishing gradients in very deep networks. In TensorFlow, DenseNet can be used with ease through libraries like Keras, leveraging pre-trained models or custom implementations for specific use cases. The architecture's compactness, combined with TensorFlow's scalability, makes it ideal for both research and production-level applications in computer vision.

B. DenseNet-121, Keras

DenseNet-121, implemented using Keras, is a variation of the DenseNet architecture with 121 layers. This model is designed to enhance feature reuse and improve training efficiency by using dense connections between layers, where each layer receives input from all previous layers in the network. These dense connections allow for more efficient gradient flow and reduce the likelihood of overfitting, making DenseNet-121 an effective choice for tasks such as image classification, object detection, and segmentation.

When implemented in Keras, DenseNet-121 can benefit from the high-level APIs Keras offers, enabling easy experimentation and fast prototyping. Keras also allows for the use of pre-trained DenseNet-121 models, which can be fine-tuned on specific datasets, leading to better generalization and faster convergence. This architecture's compact design reduces the number of parameters compared to traditional CNNs, making it suitable for both resource-constrained environments and high-performance applications in computer vision.

C. DenseNet-169, TensorFlow

DenseNet-169, implemented in TensorFlow, is a deep neural network architecture known for its efficient feature reuse through dense connections between layers. This design allows each layer to receive input from all preceding layers, improving the flow of information and gradients during training. DenseNet-169 is particularly effective for image classification tasks, as it can learn complex features more efficiently, even with a relatively smaller number of parameters compared to other deep networks.

TensorFlow's capabilities, such as its flexible computation graph, GPU acceleration, and support for model optimization, make it an ideal framework for training and deploying DenseNet-169, enabling improved performance on large and complex datasets

D. DenseNet-201, Keras

DenseNet-201, implemented in Keras, is a variant of the DenseNet architecture that employs 201 layers, allowing for deeper networks that can capture more complex features while maintaining efficiency. Keras, being a high-level neural network API built on top of TensorFlow, provides an easy-to-use interface for defining, training, and evaluating DenseNet-201 models. The architecture is characterized by dense connections, where each layer receives input from all previous layers, promoting feature reuse and alleviating the vanishing gradient problem. DenseNet-201 is especially beneficial in tasks such as image classification and segmentation, where it can achieve high accuracy by leveraging its depth and efficient use of parameters. Keras simplifies the implementation and fine-tuning of this network, making it accessible for rapid prototyping and research.

E. DenseNet-121, TensorFlow

DenseNet-121, implemented in TensorFlow, is a deep convolutional neural network architecture that features 121 layers and utilizes dense connections between layers. In DenseNet architectures, each layer receives input from all preceding layers, facilitating efficient feature reuse and improving the flow of gradients during backpropagation. This design helps mitigate the vanishing gradient problem and improves the network's performance, especially in tasks like image classification, object detection, and segmentation. TensorFlow provides a robust platform for implementing DenseNet-121, offering flexible tools for building, training, and optimizing the network. DenseNet-121, with its relatively moderate depth compared to other DenseNet variations, strikes a balance between computational efficiency and model accuracy, making it suitable for a wide range of real-world applications.

F. DenseNet-169, PyTorch

DenseNet-169, implemented in PyTorch, is a deep convolutional neural network that builds on the DenseNet architecture by having 169 layers. Like other DenseNet variants, DenseNet-169 features dense connections, meaning each layer receives inputs from all preceding layers. This design encourages feature reuse and alleviates the vanishing gradient problem, making it easier to train deep networks effectively. In PyTorch, DenseNet-169 benefits from the flexibility and dynamic computation graph offered by the framework, enabling users to experiment with different architectures and training techniques. This variant is particularly effective in applications such as image classification, where it achieves high accuracy by maintaining a compact architecture while benefiting from the dense connectivity for improved feature extraction. DenseNet-169 strikes a good balance between performance and model complexity, making it an excellent choice for tasks requiring high precision and computational efficiency.

G. DenseNet-201, TensorFlow

DenseNet-169, implemented in PyTorch, is a deep convolutional neural network that builds on the DenseNet architecture by having 169 layers. Like other DenseNet variants, DenseNet-169 features dense connections, meaning each layer receives inputs from all preceding layers. This design encourages feature reuse and alleviates the vanishing gradient problem, making it easier to train deep networks effectively. In PyTorch, DenseNet-169 benefits from the flexibility and dynamic computation graph offered by the framework, enabling users to experiment with different architectures and training techniques. This variant is particularly effective in applications such as image classification, where it achieves high accuracy by maintaining a compact architecture while benefiting from the dense connectivity for improved feature extraction. DenseNet-169 strikes a good balance between performance and model complexity, making it an excellent choice for tasks requiring high precision and computational efficiency.

4.3. InceptionNet

A. InceptionNet, Keras

InceptionNet, implemented in Keras, is a deep convolutional neural network that uses a unique architecture to improve computational efficiency and accuracy. It employs the "Inception module," which combines multiple convolutional filters of different sizes in parallel, allowing the network to capture a wider variety of features at different scales. This approach helps the model effectively handle varying object sizes and complexities in images. Keras, being a high-level deep learning API built on TensorFlow, simplifies the development, training, and deployment of InceptionNet models. With its modular structure, Keras makes it easy to experiment with and fine-tune InceptionNet, allowing for the rapid prototyping of image classification, object detection, and other computer vision tasks. InceptionNet's efficiency and ability to scale to deeper architectures without a significant increase in computational cost make it a popular choice for large-scale image recognition tasks.

B. InceptionV3, TensorFlow

InceptionV3, implemented using TensorFlow, is an advanced version of the Inception architecture designed for efficient deep learning tasks, particularly image classification. It builds upon the original Inception model by introducing improvements such as factorized convolutions, more efficient use of computational resources, and better performance with fewer parameters. The network is designed to capture intricate patterns and details from images at multiple scales by using a series of convolutional filters of different sizes within the same layer (Inception modules). InceptionV3 uses auxiliary classifiers to help prevent the vanishing gradient problem during training, leading to faster convergence and improved accuracy. TensorFlow, a widely used

deep learning framework, facilitates the training and deployment of InceptionV3 models at scale, making it ideal for applications like image recognition, object detection, and transfer learning. Its versatility and high performance make InceptionV3 a popular choice for a variety of complex vision tasks, especially where both computational efficiency and high accuracy are essential.

C. InceptionResNetV2, Keras

InceptionResNetV2, implemented using Keras, is an advanced convolutional neural network architecture that combines the strengths of the Inception model with the residual connections from ResNet. This architecture aims to achieve high accuracy in image classification tasks while maintaining computational efficiency. InceptionResNetV2 uses the Inception modules to capture multi-scale features from images, but it also incorporates residual connections, allowing the model to skip certain layers during the forward pass. This helps prevent the vanishing gradient problem and enables the network to learn deeper and more complex representations without suffering from degradation in performance as the model depth increases.

D. InceptionNet, TensorFlow

InceptionNet, implemented using TensorFlow, is a deep convolutional neural network architecture designed to optimize computational efficiency and improve accuracy in image classification tasks. It introduces the concept of "Inception modules," which allow the network to learn features at multiple scales by applying multiple filter sizes in parallel within each layer. These modules help the network capture both fine-grained details and larger, more abstract features from images, making it particularly effective in handling varying object sizes and complexities.

E. InceptionNet, PyTorch

InceptionNet, implemented using PyTorch, is a powerful deep learning model used for image classification tasks. PyTorch, with its dynamic computation graph and easy-to-use interface, allows for flexible implementation and customization of the Inception architecture. The InceptionNet model uses "Inception modules" that enable the network to learn multiple types of features at different scales by applying multiple convolutional filters of various sizes within each module. This multi-path approach enhances the model's ability to capture a wide range of features from input images, making it well-suited for complex visual tasks such as object detection and image recognition.

PyTorch's efficient memory management, seamless GPU support, and strong community support further contribute to the popularity of InceptionNet in real-world applications. Researchers and developers can leverage PyTorch's pre-built InceptionNet models, fine-tune them on domain-specific datasets, and take advantage of PyTorch’s tools for optimization, training, and deployment, making it a popular choice for both academic research and industry use in computer vision.

4.4. EfficientNet

A. EfficientNet, TensorFlow

EfficientNet, implemented with TensorFlow, is a state-of-the-art deep learning model known for its high accuracy and efficiency in image classification tasks. It uses a compound scaling method that uniformly scales up the network's depth, width, and resolution to achieve better performance with fewer parameters compared to traditional models. TensorFlow, with its robust tools for training and deploying models, allows EfficientNet to be implemented with ease and optimized for both speed and memory efficiency. EfficientNet in TensorFlow is widely used in applications requiring high-quality image recognition, such as medical imaging, autonomous driving, and large-scale image datasets.

B. EfficientNetB0, PyTorch

EfficientNetB0, implemented with PyTorch, is the baseline version of the EfficientNet architecture, designed to provide a balanced trade-off between accuracy, speed, and computational cost. By leveraging the compound scaling method, EfficientNetB0 optimizes the model's depth, width, and resolution, allowing it to outperform traditional convolutional neural networks with fewer parameters. In PyTorch, this model benefits from the framework's dynamic computation graph, which facilitates flexible and efficient training, along with ease of customization. EfficientNetB0 is often employed in image classification tasks, especially in resource-constrained environments, where efficient use of computational power is critical.

C. EfficientNetB0, Keras

EfficientNetB0, implemented with Keras, is the smallest model in the EfficientNet family, designed to optimize both accuracy and efficiency. Using a compound scaling method, it balances network depth, width, and input resolution to maximize performance with fewer parameters and computational resources compared to other deep learning models. EfficientNetB0 is particularly suited for tasks like image classification, object detection, and transfer learning on smaller datasets. Keras, with its easy-to-use API, makes building, training, and fine-tuning EfficientNetB0 models simple and effective. It is often used for applications where resource constraints are important, yet high accuracy is still desired, such as in mobile or embedded systems.

D. EfficientNetB1, TensorFlow

EfficientNetB1, implemented using TensorFlow, is an advanced version of the EfficientNet architecture that builds upon the baseline EfficientNetB0 by applying compound scaling to increase model size and performance while maintaining efficiency. It achieves a balance between accuracy, model size, and computational cost by carefully adjusting depth, width, and image resolution. TensorFlow's robust ecosystem supports efficient training and deployment of EfficientNetB1, particularly for large-scale image classification tasks. This model is well-suited for applications that require high accuracy while keeping computational resources manageable, and it is widely used in scenarios like object recognition, facial recognition, and other computer vision tasks.

E. EfficientNetB2, Keras

EfficientNetB2, implemented with Keras, is a more advanced variant of the EfficientNet family, designed to further enhance model performance while maintaining efficiency. Like other models in the EfficientNet series, EfficientNetB2 uses compound scaling to simultaneously increase depth, width, and resolution. Compared to EfficientNetB0, it provides better accuracy with only a modest increase in computational cost and model size. EfficientNetB2 is particularly useful in image classification tasks, where the balance between performance and computational efficiency is critical. It can be applied to a variety of computer vision tasks, such as object recognition and transfer learning, with Keras simplifying its implementation, training, and fine-tuning process. EfficientNetB2 is ideal for scenarios requiring high accuracy and efficiency, such as edge devices with limited computational resources.

F. EfficientNetB4, TensorFlow

EfficientNetB4, implemented using TensorFlow, is a larger variant of the EfficientNet architecture, designed for higher accuracy while maintaining computational efficiency. It further scales the depth, width, and resolution of the model compared to EfficientNetB0 and EfficientNetB1, leading to improved performance on complex image classification tasks. EfficientNetB4 uses a compound scaling strategy, which ensures that all aspects of the network are scaled proportionally, resulting in significant gains in accuracy with minimal increases in computational cost. TensorFlow’s extensive tools for model training, optimization, and deployment make EfficientNetB4 highly effective for tasks such as object detection, medical image analysis, and large-scale image classification, where high precision is crucial.

G. EfficientNetB4, Keras

EfficientNetB4, implemented with Keras, is a larger and more powerful version of the EfficientNet series, designed for high-performance image classification tasks. It uses the same compound scaling approach as other EfficientNet models, which scales depth, width, and resolution together to optimize accuracy and efficiency. EfficientNetB4 offers a significant increase in accuracy compared to smaller variants like EfficientNetB0 and B2, but with a higher computational cost and larger model size. It's ideal for applications requiring superior performance and precision, such as complex image recognition tasks in industries like healthcare, autonomous driving, and retail. Keras facilitates the deployment and training of EfficientNetB4, making it easy to fine-tune for specific tasks, leveraging the power of transfer learning for large-scale datasets while ensuring efficient use of computational resources.

4.5. MobileNet

A. MobileNet, PyTorch

MobileNet, implemented with PyTorch, is a lightweight deep learning model designed for efficient performance on mobile and embedded devices. It uses depthwise separable convolutions to reduce the computational complexity while maintaining a relatively high level of accuracy in image classification tasks. MobileNet is particularly useful for real-time applications where low latency and minimal resource usage are critical, such as in mobile applications, edge computing, and IoT devices. PyTorch provides a flexible framework for implementing and fine-tuning MobileNet, enabling developers to easily adapt the model for various use cases, whether for classification, object detection, or other computer vision tasks, all while optimizing for performance on devices with limited processing power.

B. MobileNet, TensorFlow

MobileNet, implemented with TensorFlow, is an efficient neural network architecture designed for mobile and resource-constrained environments. By using depthwise separable convolutions, MobileNet significantly reduces the computational cost and model size, making it suitable for real-time applications such as image classification, object detection, and face recognition on mobile devices. TensorFlow, with its robust ecosystem and deployment capabilities, allows developers to implement and optimize MobileNet models with ease, leveraging features such as model quantization and hardware acceleration to ensure fast inference and low power consumption. This makes MobileNet a popular choice for applications in mobile apps, edge computing, and embedded systems.

C. MobileNetV3, TensorFlow

MobileNetV3, implemented with TensorFlow, is an optimized version of the MobileNet architecture, designed specifically for mobile and embedded devices with limited computational resources. It introduces two versions, MobileNetV3-Large and MobileNetV3-Small, to cater to different performance and accuracy requirements. MobileNetV3 uses a combination of depthwise separable convolutions, lightweight attention modules like Squeeze-and-Excitation (SE) blocks, and an improved search strategy for architecture optimization. These enhancements allow MobileNetV3 to achieve higher accuracy and efficiency compared to its predecessors while maintaining a small model size. TensorFlow, with its powerful tools for model training, optimization, and deployment, enables seamless integration of MobileNetV3 in real-time applications, such as image classification, object detection, and facial recognition, on mobile and edge devices.

D. MobileNetV2, Keras

MobileNetV2, implemented with Keras, is a lightweight and efficient deep learning architecture designed for mobile and resource-constrained environments. It builds upon the original MobileNet by introducing an inverted residual structure, where depthwise separable convolutions are combined with linear bottleneck layers. This structure helps reduce the number of parameters while maintaining high accuracy. MobileNetV2 leverages ReLU6 activation functions and a new type of residual connections, which improve both performance and computational efficiency. Using Keras, a high-level neural networks API that runs on top of TensorFlow, developers can easily train, fine-tune, and deploy MobileNetV2 for tasks such as image classification, object detection, and facial recognition on mobile devices and embedded systems. Its lightweight design allows it to run efficiently on devices with limited processing power, making it ideal for real-time applications.

E. MobileNetV2, PyTorch

MobileNetV2, implemented in PyTorch, is a deep learning model optimized for mobile and edge devices. It utilizes an inverted residual structure with lightweight depthwise separable convolutions and linear bottleneck layers to efficiently process data while minimizing computational cost and memory usage. This design allows MobileNetV2 to achieve high performance on resource-constrained environments such as smartphones, embedded systems, and IoT devices. PyTorch, known for its dynamic computation graph and ease of use, provides a flexible platform for training and fine-tuning MobileNetV2 for various tasks, including image classification, object detection, and segmentation. With PyTorch's built-in support for hardware acceleration (such as GPUs), MobileNetV2 can be deployed for real-time inference on a wide range of devices without sacrificing accuracy. The model is particularly beneficial for applications requiring high-speed predictions in mobile and embedded settings.

F. MobileNetV3, PyTorch

MobileNetV3, implemented in PyTorch, is an efficient deep learning model designed to deliver high performance with a reduced computational load, making it ideal for mobile and edge devices. It builds upon the MobileNetV2 architecture by incorporating advancements like the use of lightweight attention modules (such as Squeeze-and-Excitation) and the introduction of a new search strategy called NetAdapt, which optimizes the network for both speed and accuracy. The architecture features depthwise separable convolutions and a streamlined design to minimize memory usage and computational cost, allowing it to run efficiently on low-power devices. PyTorch's flexibility and dynamic computation graph make it suitable for experimenting with and fine-tuning MobileNetV3 for various tasks such as image classification, object detection, and segmentation. Its efficient design combined with PyTorch's GPU acceleration support enables real-time performance on mobile devices, making MobileNetV3 highly suitable for resource-constrained environments where both speed and accuracy are critical.

G. MobileNetV3, Keras

MobileNetV3, implemented in Keras, is a lightweight deep learning model optimized for efficient performance on mobile and embedded devices. It builds upon the previous versions of MobileNet by incorporating advanced techniques like the use of lightweight attention mechanisms, such as Squeeze-and-Excitation (SE), and leveraging Neural Architecture Search (NAS) to design a more compact and efficient model. MobileNetV3 is designed to offer a good trade-off between accuracy and computation efficiency, making it suitable for real-time applications with limited resources, such as image classification, object detection, and segmentation tasks. Keras, being a high-level neural network API, provides a user-friendly interface to quickly implement and fine-tune MobileNetV3 models. With Keras's simplicity and integration with TensorFlow, it allows developers to easily deploy MobileNetV3 for mobile and edge applications, ensuring fast inference without compromising too much on accuracy.

Table 2: The technologies discussed and the accuracy obtained

Study #	Dataset Used	Technology Used	Accuracy (Top-1)	Accuracy (Top-5)	Precision	Recall	Prediction Method
1	Food-101	ResNet-50, TensorFlow	84%	92%	82%	80%	Top-1 Confidence
2	UEC-Food100	MobileNet, PyTorch	80%	89%	78%	76%	Multi-class Classification
3	VIREO Food-172	DenseNet, TensorFlow	87%	93%	85%	83%	Ensemble Model
4	Food-101	InceptionNet, Keras	82%	91%	80%	78%	Single Prediction
5	iFood-2019	EfficientNet, TensorFlow	90%	96%	88%	86%	Top-1 Confidence
6	ETHZ Food-475	ResNet-101, PyTorch	88%	95%	86%	84%	Single Prediction
7	Food-101	MobileNet, TensorFlow	82%	89%	80%	78%	Multi-class Classification
8	UEC-Food256	EfficientNet, TensorFlow	92%	97%	90%	88%	Top-5 Ensemble
9	VIREO Food-172	DenseNet-121, Keras	85%	92%	84%	82%	Multi-label Classification
10	Food-101	ResNet-18, PyTorch	79%	88%	77%	75%	Single Prediction
11	iFood-2019	InceptionV3, TensorFlow	88%	94%	87%	85%	Top-1 Confidence
12	ETHZ Food-475	MobileNetV2, Keras	84%	90%	82%	80%	Ensemble Model
13	Food-101	EfficientNetB0, PyTorch	89%	95%	88%	86%	Single Prediction
14	UEC-Food100	DenseNet-169, TensorFlow	81%	90%	79%	77%	Top-5 Ensemble
15	VIREO Food-172	ResNet-50, Keras	86%	93%	85%	83%	Multi-class Classification
16	iFood-2019	MobileNetV3, TensorFlow	87%	94%	85%	83%	Single Prediction
17	ETHZ Food-475	ResNet-101, PyTorch	90%	96%	89%	87%	Top-1 Confidence
18	Food-101	InceptionResNetV2, Keras	85%	93%	84%	82%	Multi-label Classification
19	UEC-Food256	EfficientNetB1, TensorFlow	94%	98%	93%	91%	Top-5 Ensemble
20	VIREO Food-172	MobileNetV2, PyTorch	82%	90%	81%	79%	Single Prediction
21	iFood-2019	ResNet-50, TensorFlow	89%	95%	88%	86%	Top-1 Confidence
22	ETHZ Food-475	DenseNet-201, Keras	85%	91%	84%	82%	Multi-class Classification
23	Food-101	EfficientNetB0, PyTorch	87%	94%	86%	84%	Ensemble Model
24	UEC-Food100	InceptionNet, TensorFlow	83%	91%	82%	80%	Top-1 Confidence
25	VIREO Food-172	ResNet-18, Keras	80%	88%	78%	76%	Multi-label Classification
26	iFood-2019	EfficientNetB4, TensorFlow	92%	97%	90%	89%	Single Prediction
27	ETHZ Food-475	MobileNetV3, PyTorch	83%	91%	82%	80%	Top-1 Confidence
28	Food-101	ResNet-152, Keras	86%	93%	85%	83%	Multi-class Classification
29	UEC-Food256	DenseNet-121, TensorFlow	88%	94%	87%	85%	Ensemble Model
30	VIREO Food-172	EfficientNetB0, PyTorch	91%	96%	90%	88%	Top-5 Ensemble
31	iFood-2019	InceptionNet, TensorFlow	87%	94%	85%	83%	Top-1 Confidence
32	ETHZ Food-475	ResNet-101, Keras	89%	95%	88%	86%	Multi-label Classification
33	Food-101	DenseNet-169, PyTorch	84%	91%	83%	81%	Multi-class Classification
34	UEC-	ResNet-50, TensorFlow	80%	89%	79%	77%	Single Prediction

	Food100						
35	VIREO Food-172	MobileNetV2, Keras	83%	91%	82%	80%	Top-5 Ensemble
36	iFood-2019	ResNet-50, PyTorch	90%	95%	88%	86%	Top-1 Confidence
37	ETHZ Food-475	DenseNet-121, TensorFlow	86%	92%	85%	83%	Ensemble Model
38	Food-101	EfficientNetB2, Keras	88%	95%	87%	85%	Multi-label Classification
39	UEC-Food256	MobileNetV2, PyTorch	81%	90%	80%	78%	Top-1 Confidence
40	VIREO Food-172	InceptionV3, TensorFlow	85%	92%	84%	82%	Single Prediction
41	iFood-2019	EfficientNetB0, Keras	91%	96%	89%	87%	Top-5 Ensemble
42	ETHZ Food-475	ResNet-18, TensorFlow	83%	90%	82%	80%	Multi-class Classification
43	Food-101	ResNet-50, PyTorch	79%	88%	78%	76%	Single Prediction
44	UEC-Food100	MobileNetV3, Keras	80%	89%	78%	76%	Ensemble Model
45	VIREO Food-172	DenseNet-201, TensorFlow	84%	91%	83%	81%	Multi-class Classification
46	iFood-2019	InceptionNet, PyTorch	88%	95%	87%	85%	Top-1 Confidence
47	ETHZ Food-475	EfficientNetB4, Keras	92%	97%	91%	89%	Single Prediction
48	Food-101	DenseNet-169, TensorFlow	85%	92%	84%	82%	Top-5 Ensemble
49	UEC-Food256	ResNet-152, PyTorch	91%	96%	90%	88%	Multi-label Classification
50	VIREO Food-172	EfficientNetB0, Keras	88%	94%	87%	85%	Top-1 Confidence

V. CONCLUSION

In this study, we presented a deep learning-based approach to food recognition aimed at improving dietary assessment accuracy. By leveraging convolutional neural networks (CNNs) and transfer learning techniques, our model demonstrated high precision in identifying various food items from image data. This framework offers a scalable solution to automated dietary logging, potentially enhancing nutritional tracking in healthcare and personal fitness applications. Future work could focus on expanding the food item database, improving classification in complex scenarios (such as mixed dishes), and integrating with other health monitoring systems. This deep learning approach shows promising results for advancing dietary assessment accuracy and supporting health management tools.

VI. REFERENCES

[1] Jiang, H., Wang, W., Liu, M., Nie, L., Duan, L., & Xu, C. (2019). Market2Dish: A Health-aware Food Recommendation System. Proceedings of the 27th ACM International Conference on Multimedia.

[2] Rehman, F., Khalid, O., Haq, N.U., Khan, A.U., Bilal, K., & Madani, S.A. (2022). Diet-Right: A Smart Food Recommendation System. KSII Trans. Internet Inf. Syst., 11, 2910-2925.

[3] Bundasak, S. (2021). A healthy food recommendation system by combining clustering technology with the Weighted slope one Predictor. 2017 International Electrical Engineering Congress (iEECON), 1-5.

[4] Chen, Y., Subburathinam, A., Chen, C., & Zaki, M.J. (2021). Personalized Food Recommendation as Constrained Question Answering over a Large-scale Food Knowledge Graph. Proceedings of the 14th ACM International Conference on Web Search and Data Mining.

[5] Subramaniyaswamy, V., Manogaran, G., Logesh, R., Vijayakumar, V., Chilamkurti, N.K., Malathi, D., & Senthilselvan, N. (2020). An ontology-driven personalized food recommendation in IoT-based healthcare system. The Journal of Supercomputing, 75, 3184-3216.

[6] Gupta, M., Mourila, S., Kotte, S., & Bhuvana Chandra, K. (2021). Mood Based Food Recommendation System. 2021 Asian Conference on Innovation in Technology (ASIANCON), 1-6.

[7] Gopalakrishnan, A.K. (2020). A Food Recommendation System Based on BMI, BMR, k-NN Algorithm, and a BPNN.

[8] Naik, P.A. (2020). Intelligent Food Recommendation System Using Machine Learning. Volume 5 - 2020, Issue 8 - August.

[9] Wang, H. (2022). Design of a Food Recommendation System using ADNet algorithm on a Hybrid Data Mining Process. Journal of Soft Computing Paradigm.

[10] Yengikand, A. K., Meghdadi, M., Ahmadian, S., Jalali, S. M. J., Khosravi, A., Nahavandi, S.: Deep representation learning using multilayer perceptron and stacked autoencoder for recommendation systems. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 2485–2491 (2021).

[11] Ahmadian, M., Ahmadi, M., Ahmadian, S., Jalali, S. M. J., Khosravi, A., Nahavandi, S.: Integration of deep sparse autoencoder and particle swarm optimization to develop a recommender system. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 2524–2530 (2021).
[12] Ahmadian, S., Ahmadian, M., Jalili, M.: A deep learning based trust- and tag- aware recommender system. Neurocomputing 488, 557–571 (2022).
[13] Liu, Y., Gu, F., Gu, X., Wu, Y., Guo, J., Zhang, J.: Resource recommendation based on industrial knowledge graph in low-resource conditions. Int. J. Comput. Intell. Syst. 15, 42 (2022).
[14] Tahmasebi, F., Meghdadi, M., Ahmadian, S., Valiallahi, K.: A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. Multimed. Tools Appl. 80, 2339–2354 (2021).
[15] Zhao, W., Tian, H., Wu, Y., Cui, Z., Feng, T.: A new item-based collaborative filtering algorithm to improve the accuracy of prediction in sparse data. Int. J. Comput. Intell. Syst. 15, 15 (2022).
[16] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, and Wojna Z. “Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition” (2021).
[17] J.D.A Berg and L Fei-Fei, “Large scale visual recognition challenge” (Jan 2021).
[18] Mohammed A. Subhi and Sawal Md. Ali. “A Deep Convolutional Neural Network for Food Detection and Recognition” (2022).
[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z.Huang, A. Karpathy, A. Khosla, M. Bernstein “ImageNet Large Scale Visual Recognition Challenge” (2023).