Personalized Class Incremental Context-Aware Food Classification for Food Intake Monitoring

Hassan Kazemi Tehrani

Thesis in the Department of Electrical and Computer Engineering

Concordia University

Montréal, Québec, Canada

September 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Hassan Kazemi Tehrani**

Entitled: **Personalized Class Incremental Context-Aware Food Classification for Food Intake Monitoring**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Dongyu Qiu

_____ External Examiner
Dr. Wen-Fang Xie

_____ Examiner
Dr. Dongyu Qiu

_____ Supervisor
Dr. Jun Cai

Approved by _____
Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2024

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Personalized Class Incremental Context-Aware Food Classification for Food Intake Monitoring

Hassan Kazemi Tehrani

Accurate food intake monitoring is essential for maintaining a healthy diet and preventing nutrition-related diseases. Traditional food classification models struggle with the diverse range of foods across cultures and the continuous introduction of new food types due to relying on fixed-sized datasets. Moreover, studies show that people consume only a small range of foods across the existing ones. These limitations necessitate the model to adapt itself as new classes appear. Additionally, the model needs to pay more attention to certain food classes. While existing class-incremental models have low accuracy for the new classes and lack personalization, this work introduces a personalized, class-incremental food classification model designed to address these challenges and enhance food intake monitoring systems. Our approach dynamically adapts to new real-world food classes, maintaining accuracy for both new and existing classes through personalization. The model prioritizes foods based on individuals eating habits by considering meal frequencies, times, and locations. We employ a modified dynamic support network (DSN), the personalized dynamic support network (PDSN), to handle new food classes and propose a comprehensive framework integrating this model into a food intake monitoring system. The system analyzes meal images, estimates food weight with a smart scale, calculates macro-nutrient content, and creates a dietary profile via a mobile app. Experimental evaluations on the personalized datasets FOOD101-Personal and VIPER-FoodNet-Personal (VFN-Personal) demonstrate the model's effectiveness in improving classification accuracy, addressing the limitations of conventional and class-incremental food classification models.

Acknowledgments

I am deeply grateful to Dr. Jun Cai for his guidance and support, which helped me navigate the challenges of my graduate studies.

I also extend my heartfelt appreciation to my parents for their unwavering support throughout my life, nurturing my curiosity and ambition to pursue my goals.

Contribution of Authors

This dissertation is the research work of Hassan Kazemi Tehrani and it is submitted under the supervision of Dr. Jun Cai for the degree of Master of Applied Science at Concordia University. The dissertation is original and unpublished.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the necessary background information to understand the concept and importance of food intake monitoring and personalized food image classification.

1.1 Motivation

In recent years, maintaining a healthy diet to improve the quality of life has become increasingly important. Accurate estimation of daily nutritional intake is crucial for maintaining a healthy lifestyle and preventing nutrition-related diseases such as diabetes and heart disease. This practice can help control weight and boost both physical and mental health. New technologies and tools, such as wearable devices, have enabled food intake monitoring systems to track dietary habits, analyze nutritional information, and recommend dietary advice [1].

With the advent of smartphones and mobile technologies, individuals frequently take pictures of their meals and snacks, leading to an increase in the number of food images and a growing demand for food image classification systems [2]. Food image classification, a subcategory of image classification, uses computer vision to categorize or label food images based on their contents [3]. Existing works have shown remarkable outcomes on fixed-sized datasets [4, 5, 6, 7, 8], However, in multicultural urban settings where food diversity is at its peak, these models often struggle to adapt to new food categories not present in their fixed training datasets.

Consequently, there is a pressing need for models that can dynamically adapt and incorporate new food items continuously, a process referred to as class-incremental learning [9]. Class-incremental learning models, which learn new classes without forgetting previous ones, provide a viable solution to address the drawbacks of conventional methods. However, while state-of-the-art methods achieve good overall accuracy, their performance on incremental classes is noticeably low [10]. This limitation hinders precise dietary monitoring, especially when user-specific food classes appear incrementally.

To address these challenges, we propose a novel personalized class-incremental food classification model, a modified version of dynamic support network (DSN) [11], that enhances the adaptability and accuracy of food intake monitoring systems. By integrating contextual factors such as meal times, locations, and specific eating habits, our model aims to achieve universal accuracy across diverse dietary preferences. This personalized approach improves the classification accuracy of both new and existing food items, enhancing the applicability of the system in real-world scenarios.

1.2 Objectives

The primary objective of this thesis is to develop a personalized class-incremental food classification model that efficiently learns new food classes while retaining high accuracy for previously learned classes. The model aims to dynamically adapt to the introduction of new food items, addressing the limitations of static models that struggle with food diversity in multicultural settings. By focusing on personalization, the model will cater to individual dietary habits and preferences, thereby improving the overall accuracy and relevance of food intake monitoring systems.

Another key objective is to design a robust framework that integrates the proposed personalized class-incremental model as its core component. This framework will leverage advanced computer vision and machine learning techniques to accurately track dietary habits, analyze nutritional information, and provide personalized dietary recommendations. The framework aims to be user-friendly, accessible, and capable of operating in real-world scenarios, ensuring practical applicability and ease of use for end-users.

Finally, this work aims to conduct a thorough evaluation of the proposed model using new

benchmark datasets, FOOD101-Personal and VIPER-FoodNet-Personal (VFN-Personal) [12]. The evaluation will involve comparing the performance of the proposed model with existing baseline models in terms of classification accuracy, adaptability, and personalization. By demonstrating the effectiveness of the proposed model, this objective aims to establish its superiority and practical utility in real-world food classification tasks.

1.3 Contribution

The main contributions of this work can be summarized as follows:

Design and Development of a Novel Personalized Class-Incremental Food Classification Model

This thesis proposes a groundbreaking personalized class-incremental food classification model that effectively addresses the limitations of existing methods. The model is designed to achieve high accuracy for both existing and new food classes, dynamically adapting to the continuous introduction of new food items while maintaining robust performance on previously learned classes by leveraging personalization.

Design and Implementation of a Comprehensive Food Intake Monitoring Framework

We design and implement a comprehensive framework for food intake monitoring systems that integrates the proposed personalized model. This framework leverages advanced machine learning and computer vision techniques to provide accurate, dynamic, and personalized monitoring of dietary habits. The key contributions in this framework include the implementation of our personalized class incremental food image classifier, a smart scale for precise weight measurement, the development of a user-friendly mobile application that serves as the interface for capturing meal images and receiving dietary recommendations, and the integration of the system with a comprehensive nutrient database to ensure accurate nutritional analysis. These components work together to enhance the overall user experience, making the framework a valuable tool for effective health and nutrition management.

Extensive Evaluation on New Benchmark Datasets

The proposed model is rigorously evaluated and compared with existing baseline models using new benchmark datasets, FOOD101-Personal and VFN-Personal. The contribution includes the implementation of baseline models, which are then integrated with our personalized class-incremental method. We evaluated these integrated models using new benchmark datasets. This evaluation highlights the superior performance of our method in classification accuracy, adaptability, and personalization, demonstrating its effectiveness and practical utility in real-world food classification tasks.

It is worth mentioning that this work has resulted in the publication of the below research paper:

- Tehrani et al. "Personalized Class Incremental Context-Aware Food Classification for Food Intake Monitoring Systems" (2024)

1.4 Thesis Structure

The content of the thesis is organized as follows:

Chapter 2 presents the literature review, covering various aspects of food image classification, class incremental classification, and personalized food classification. This chapter lays the foundation by discussing the existing research and methodologies relevant to these topics.

Chapter 3 details the food intake monitoring system. This includes a comprehensive explanation of the personalized food intake monitoring framework, the integration and functionality of the personalizer plug-in, and the concept and implementation of the personalized dynamic support network. The mathematical models and algorithms introduced in this work are also elaborated in this chapter.

Chapter 4 addresses the implementation and experimental setup. This section describes the datasets used, baseline models, and specific implementation details. The challenges encountered during development are discussed, along with the tools and programming frameworks employed.

Chapter 5 evaluates the performance of the proposed methods through experimental results. It includes improvements observed with the personalized plug-in, enhancements over the original

dynamic support network, and insights from the ablation study. This chapter analyzes the test results, providing explanations for the observed performance, whether it leads to outperformance or underperformance.

Chapter 6 concludes the thesis, summarizing the contributions of this work and discussing potential future research directions to further improve the proposed system's performance.

Chapter 2

Literature Review

2.1 Food Image Classification

The increase in available food datasets has further advanced the field of food image classification. Identifying food types directly from images is highly valuable for a range of food-related applications [13]. According to [14], several traditional machine learning methods are capable of classifying food images such as support vector machines (SVM) [15], k-nearest neighbors (KNN) [16], multiple kernel learning (MKL) [17], and random forests [18]. These traditional methods use handcrafted features including color, texture and scale-invariant feature transform [19]. Several papers have explored these techniques in the context of food image classification, each contributing unique perspectives and advancements.

In [20], researchers introduced the first visual dataset of fast foods, comprising a total of 4,545 still images, 606 stereo pairs, 303 videos captured using a 360° camera for structure-from-motion applications, and 27 privacy-preserving videos documenting eating events of volunteers. This work was motivated by the need to enhance fast food recognition for dietary assessment. The data collection involved obtaining three instances of 101 different food items from 11 popular fast-food chains and capturing images and videos in both restaurant conditions and controlled lab settings. To evaluate the dataset, the researchers employed two standard approaches: color histograms and bag of SIFT (Scale-Invariant Feature Transform) features, combined with an SVM classifier. These benchmarking methods were designed to stimulate further research in the area by providing a robust

starting point for evaluating recognition systems. The dataset, along with the benchmarks, has been made freely available to the research community, encouraging collaboration and innovation. This contribution is expected to drive advancements in the accuracy and efficiency of fast food recognition technologies, ultimately aiding in dietary assessment.

The proposed system in [21] utilizes the MKL method to integrate various image features such as color, texture, and SIFT. MKL is particularly advantageous as it allows for the estimation of optimal weights to combine these image features adaptively for each food category. This adaptability is crucial because the most effective features for recognizing different foods can vary significantly. For example, color might be a critical feature for identifying ”potage,” while texture could be more relevant for recognizing ”hamburger.” A prototype system implementing this method was developed to recognize food images captured by cellular phone cameras. The system achieved a 61.34% classification rate for 50 kinds of foods in experimental evaluations. Notably, when considering the top three candidate categories in terms of the output values of the 1-vs-rest classifiers, the classification rate increased to 80.05%. This result demonstrates the system's potential for practical application. The integration of various image features using MKL represents a significant advancement in food image recognition, building on previous progress in generic object recognition. By applying MKL to food image recognition, [21] effectively bridged the gap between machine learning techniques and practical dietary assessment tools.

Another significant advancement is proposed in [22], where the authors introduce a new representation of food items that leverages the spatial relationships between different ingredients. This approach addresses the inherent difficulty in food recognition due to the deformable nature of food items and their significant variations in appearance. The proposed method calculates pairwise statistics between local features computed over a soft pixel-level segmentation of the image into eight ingredient types. These statistics are accumulated in a multi-dimensional histogram, which serves as a feature vector for an SVM classifier. This innovative representation significantly improves the accuracy of food recognition compared to existing methods. The research is driven by the observation that a food item can largely be characterized by its ingredients and their relative spatial relationships. For example, a sandwich might consist of a layer of meat between slices of bread, while a

salad comprises various greens with a diverse spatial layout. Despite potential unreliability in detecting individual ingredients, aggregating pairwise statistics about ingredient types and their spatial arrangement provides sufficient information to reliably identify food items, both at coarse and fine levels. Implementing this approach involves assigning a soft label (distribution over ingredients) to each pixel in the image using a semantic texton forest. Subsequently, a multi-dimensional histogram feature is constructed, where each bin represents a pair of ingredient labels and discretized geometric relationships between two pixels. This histogram, aggregated from many pixel pairs in the image, captures the spatial distribution of ingredients. Finally, the histogram is used as a feature vector in a multi-class SVM classifier to recognize the food item. The proposed method's strength lies in its ability to cope with the significant intra-class variations and occlusions typical of food images.

Food naturally comes in a wide variety of looks, which contributes to high intra-class diversity: various foods within the same category might have quite diverse appearances. On the other hand, there is little inter-class variation. Therefore dietary groups may seem to be comparable to one another. These qualities put traditional techniques to the test since they frequently fail to identify the intricate details required for precise food identification [23].

Convolutional neural networks (CNNs) excel at image classification. CNNs automatically learn features through 2D convolutional layers, eliminating the need for manual feature extraction. This allows CNNs to perform well in detailed feature detection and makes them particularly accurate for computer vision tasks [24]. There are some famous CNN architectures like AlexNet [25], VGGNet [26], GoogLeNet (Inception v1) [27], ResNet [28], DenseNet [29], and EfficientNet [30] ordered by publication date respectively, which can be used for image classification tasks including food image classification. Multiple pieces of research have been conducted to utilize CNNs to classify food images with various network architectures and types of food datasets [31].

In [32], researchers address the limitation of existing image-based diet assessment methods that typically focus on either food classification or food portion size estimation, but not both simultaneously. They propose an end-to-end multi-task framework capable of performing both tasks concurrently, thereby enhancing the practical applicability of these methods in real-life scenarios

where multiple tasks need to be processed together. The proposed framework leverages deep learning techniques and introduces a novel dataset collected from a nutrition study, where registered dietitians provided the ground truth for food portions. The multi-task learning approach utilizes L2-norm-based soft parameter sharing to simultaneously train the classification and regression tasks. This method allows each task to maintain its own feature space while regularizing the lower layers of the two models. Experimental results demonstrate that this approach improves both food classification accuracy and the mean absolute error for portion size estimation, indicating significant potential for advancing the field of image-based dietary assessment. A key innovation in this work is the use of cross-domain feature adaptation combined with normalization techniques to enhance the performance of food portion size estimation. By concatenating feature vectors from both the classification and regression networks, the model can utilize prior knowledge from the classification task to better inform portion size estimation based on the known food category. The researchers also highlight the importance of data availability for the success of deep learning methods. They note the lack of existing datasets that include both food categories and corresponding portion sizes, which has hindered progress in developing comprehensive end-to-end dietary assessment systems. To address this gap, they introduce a new dataset comprising both food categories and portion sizes, collected during eating occasions and annotated by registered dietitians. This dataset provides a valuable resource for future research in the field.

In [33], researchers tackle the challenge of accurately measuring food and energy intake, which is crucial for combating obesity. They propose an innovative assistive calorie measurement system designed to help patients and healthcare professionals manage diet-related health conditions effectively. This system operates on smartphones, enabling users to capture images of their food and automatically measure calorie intake. The core of the system's food recognition capability relies on deep convolutional neural networks (DCNNs). By training the system with 10,000 high-resolution food images, the researchers achieve an impressive 99% accuracy in recognizing single food portions. This high level of accuracy is vital for the practical application of the system in real-world settings. The significance of this work lies in its potential to provide a convenient and intelligent solution for dietary assessment. By leveraging modern deep learning techniques, the system not only classifies food items but also estimates their caloric content, offering a comprehensive approach to

diet monitoring. The integration of DCNN with the mobile application allows the system to handle training and testing requests efficiently, even on low-powered mobile devices, making it accessible and user-friendly. The researchers also highlight the broader implications of accurate dietary assessment methods. Improved accuracy in dietary information can strengthen the ability of researchers to identify relationships between diet, diseases, and genetics. Moreover, the widespread adoption of such assistive technologies can empower individuals to make informed dietary choices, potentially leading to long-term health improvements.

The study presented in [34] demonstrates a significant improvement in food recognition accuracy by integrating DCNN features with conventional hand-crafted image features. Specifically, the researchers combined DCNN features with fisher vectors using histogram of oriented gradients (HoG) and color patches to achieve a top-1 accuracy of 72.26% and a top-5 accuracy of 92.00% on the UEC-FOOD100 [35] dataset. This performance markedly surpasses the previously reported best classification accuracy of 59.6% for the same dataset. In preliminary experiments, the researchers found that training DCNNs solely on the UEC-FOOD100 dataset did not yield better performance than conventional methods, primarily due to the limited amount of training data. To address this issue, they adopted a strategy of using a pre-trained DCNN on a large-scale dataset as a feature extractor. This approach involves extracting DCNN features from the output signals of the layer just before the final one of the pre-trained network. The results of this study highlight the potential of combining DCNN features with conventional image features to enhance food recognition accuracy, even for fine-grained datasets. This integrated approach leverages the strengths of both deep learning and traditional image processing techniques, offering a robust solution for practical food recognition applications.

In [36], the authors address the significant challenge of improving dietary assessment accuracy, which is crucial for effective weight management. They propose a novel approach that leverages deep learning-based visual food recognition algorithms integrated with an edge computing paradigm. This combination aims to overcome the inherent limitations of traditional dietary assessment methods, such as reliance on memory and inaccurate calorie tracking. The study focuses on two primary objectives, enhancing food recognition accuracy through advanced deep learning techniques and designing a system that utilizes edge computing to optimize performance, minimize

response time, and reduce energy consumption. The researchers develop and test DCNNs for food image classification, demonstrating a substantial improvement in accuracy over conventional methods. The system achieves remarkable results with food recognition accuracy surpassing previous benchmarks and response times comparable to the best existing solutions. One of the key innovations of this work is the application of edge computing, which distributes computational tasks between mobile devices and cloud servers. This approach addresses challenges related to system latency and battery life, making the technology more practical for real-world use. The authors' implementation on edge devices, such as smartphones, and a GPU cluster for server-side processing, effectively balances computational load and enhances system efficiency. The significance of this research lies in its potential to revolutionize dietary assessment through the integration of cutting-edge technologies. By utilizing deep learning for accurate food recognition and edge computing for optimized system performance, the proposed system offers a comprehensive and efficient solution for dietary tracking.

In [37], the authors introduce a new dataset specifically designed for evaluating food recognition and dietary assessment systems, the Mediterranean Greek Food (MedGRFood) dataset. This dataset comprises 42,880 images of Mediterranean cuisine, with a focus on Greek dishes, categorized into 132 distinct food classes. The dataset's comprehensive nature and large volume make it a valuable resource for advancing the field of food image recognition. The study leverages the EfficientNetB2 architecture, a member of the EfficientNet [30] family of CNNs, to achieve high performance in food recognition tasks. The proposed deep learning schema incorporates fine-tuning, transfer learning, and data augmentation techniques, resulting in an impressive top-1 accuracy of 83.4% and top-5 accuracy of 97.8% on the MedGRFood dataset. By providing a dataset tailored to Mediterranean cuisine, the authors aim to enhance the performance of food recognition systems, thereby supporting better dietary management and health outcomes.

In [38], the authors address the challenge of improving food recognition and retrieval systems through the use of DCNNs and propose a significant advancement in the field with the introduction of the Food-475 database. This dataset contains 247,636 images across 475 food classes, created by merging four existing food databases. This large-scale and heterogeneous dataset is crucial for enhancing the robustness and generalization of food recognition models. The study emphasizes

the importance of domain-representativeness in training effective CNNs. The Food-475 database is evaluated in terms of its food-domain representativeness, considering factors such as the total number of images, the number of food classes, and the number of examples per class. By training a ResNet [28] with 50 layers (ResNet-50) on this extensive dataset, the authors demonstrate that features extracted from the Food-475 database achieve superior performance in food classification and retrieval tasks, compared to features obtained from other databases. The authors note that existing food recognition methods often rely on single databases, which can limit the generalization of the recognition algorithms due to potential biases in image acquisition conditions or composition. The introduction of the Food-475 database addresses this limitation by offering a more heterogeneous and representative set of food images. The paper also introduces a semi-automatic merging procedure to refine food class definitions, improving upon previous datasets like Food-524 [39], which did not account for semantically equivalent food classes.

In [40], the authors address the challenge of classifying food ingredients, a relatively underexplored area compared to food meal classification, by proposing a novel framework named DeepFood. This framework leverages deep learning techniques, particularly CNNs, to enhance feature extraction and multi-class classification accuracy for food ingredient images. The DeepFood framework integrates several advanced machine learning techniques to achieve superior performance. It employs transfer learning algorithms with CNNs for deep feature extraction, followed by a multi-class classification algorithm to analyze these features. Specifically, the framework utilizes ResNet [28] deep feature sets, information gain (IG) [41] for feature selection, and the SVM with a sequential minimal optimization (SMO) [42] algorithm for classification. The experimental results on a dataset consisting of 41 food ingredient classes with 100 images per class demonstrate the effectiveness of the DeepFood framework in improving classification accuracy compared to existing methods.

In [1], the authors focus on food image classification and recognition, essential steps for dietary assessment, leveraging a GoogLeNet [27] model to achieve high accuracy in food/non-food classification and food category recognition. The authors highlight the evolution of dietary assessment techniques, emphasizing the role of multimedia approaches, especially food image analysis. The paper presents two experimental approaches: food/non-food image classification and food category

recognition. The authors created two custom datasets by aggregating images from existing datasets, social media, and mobile devices. They employed a fine-tuned GoogLeNet [27] model within the Caffe deep learning framework, achieving a remarkable 99.2% accuracy in food/non-food classification and 83.6% accuracy in food category recognition. The paper underscores the difficulty in accurately recognizing food items due to their visual similarities and the complexity of mixed food presentations. The authors propose that while perfect classification may be challenging, recognizing general food types can still provide valuable information for estimating dietary values and monitoring nutritional intake.

Recently, with the outstanding achievements of transformers in the field of natural language processing (NLP) introduced by [43], researchers got attracted to the use of these types of architecture in computer vision tasks. [44] proposed an architecture called Vision Transformer which treats image patches as tokens and processes them using a transformer model. Moreover, [45] introduced Swin Transformer, a hierarchical transformer utilizing shifted window mechanisms, designed to handle image inputs of various sizes and achieve strong performance across multiple vision tasks. Also, [46] in an architecture called ConvNeXt, updated traditional CNNs with transformer-like architectures to enhance performance. Ultimately, in the realm of food classification, various researchers have employed these types of architectures to effectively categorize different foods.

In [47], the authors address the challenges of food image classification in computer vision and machine learning, particularly when dealing with foods that have similar shapes but different nutritional values. The paper introduces a high-accuracy food image classification method that enhances features using a Vision Transformer-based approach, called AlsmViT. ViT models divide images into small patches and utilize self-attention mechanisms to capture local features, ultimately generating a global image representation. This approach has shown promising results in image classification tasks, particularly through semi-supervised learning, which enhances the model's performance and generalization ability. However, ViT models have limitations, particularly their dependency on large datasets, which can lead to overfitting when trained on smaller datasets like Food-101 [48] and Vireo Food-172 [49]. To address these limitations, the authors propose the AlsmViT model, which enhances Vision Transformer performance through data augmentation and feature enhancement techniques. The AlsmViT model incorporates Augmentplus, LayerScale, and a multi-layer

perception mechanism to locally enhance features, mitigating issues like overfitting and premature saturation. The model was trained and validated on the Food-101 and Vireo Food-172 datasets, achieving validation set accuracies of 95.17% and 94.29%, respectively. This represents a significant improvement of 5.26% and 5.12% over the ViT-L model, demonstrating the effectiveness of the proposed enhancements in handling foods with similar shapes but different nutritional values.

In [50], the authors propose a novel approach, the Swin-DR network, which integrates a deep convolutional module for local feature enhancement with the Swin Transformer for global feature extraction. This combination allows the model to achieve a more detailed and accurate recognition of food items with similar macro characteristics but differing in fine details. The Swin-DR network utilizes a depthwise separable residual convolutional block (DRConvBlock) to further refine local features and a multi-layer perceptron based on global average pooling and dropout (MLP-GD) for end-to-end classification. This architecture enhances the model's ability to discern subtle inter-class variations and improves classification accuracy. The proposed method demonstrates superior performance on the Foodx-251 [51] and UEC Food-256 [52] datasets, achieving validation accuracies of 81.07% and 82.77%, respectively.

In [53], the authors recognized that, although the Swin Transformer is effective at capturing both local and global features, it tends to be biased towards global features. To improve local feature learning, they introduce several enhancements: the Local Feature Extraction Network (L-FEN), Convolution Patch-Merging (CP), Multi-Path (MP), and Multi-View (MV) modules. The L-FEN module enhances the Swin Transformer's capability to capture detailed local features. The CP technique adapts the Swin Transformer's Patch Merging for a more localized and hierarchical approach. The MP method aggregates features across various stages of the Swin Transformer to better highlight local details. The MV module replaces traditional Swin blocks with those that incorporate diverse receptive fields, broadening the scope of local feature capture. The proposed architecture, named Global–Local Swin Transformer (GL-Swin), is evaluated on fine-grained food classification tasks across three major datasets: ISIA Food-500 [54], UEC Food-256 [52], and Food-101 [48]. The GL-Swin achieves accuracies of 66.75%, 85.78%, and 92.93% on these datasets, respectively.

2.2 Class Incremental Classification

Continual learning, also known as incremental learning or lifelong learning is an area of machine learning that aims to learn continuously and adapt to new data and tasks over time without forgetting previous knowledge, which is a common issue known as catastrophic forgetting [55].

Continual learning can be categorized as instance-incremental learning (IIL), domain-incremental learning (DIL), task-incremental learning (TIL), class-incremental learning (CIL), task-free continual learning (TFCL), online continual learning (OCL), and continual pre-training (CPT) [56].

In class incremental learning, new classes appear over time and the model should detect new classes without losing the capability of classifying the ones. There are works focusing on using regularization-based approaches to achieve incremental learning [57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69], while some other works have used replay-based methods and [70, 71, 72, 73, 74, 75, 76, 77, 78, 79] and parameter isolation methods [80, 81, 82, 83, 84, 85].

Moreover, in food classification, several works utilize continual learning to improve their food classification model further. In [86], the authors address two significant challenges in deploying deep learning-based food recognition systems in real-world scenarios: continual learning with the arrival of new food classes and handling class-imbalanced data. They recognize that as new food types emerge over time, the model must learn these new classes without forgetting previously learned ones (catastrophic forgetting). Additionally, food image datasets are often long-tailed, with a few popular food types being more frequent than many rare types, requiring improved generalization for rare classes. To tackle these issues, the authors introduce two new benchmark datasets, VFN-INSULIN and VFN-T2D, which reflect real-world food consumption patterns for insulin users and individuals with type 2 diabetes, respectively. They propose a novel end-to-end framework designed for long-tailed continual learning that incorporates feature-based knowledge distillation and a new data augmentation technique combining class-activation-map (CAM) and CutMix. The knowledge distillation process is enhanced with an additional prediction head to address representation misalignment and preserve learned knowledge. The data augmentation method improves generalization for rare classes by integrating crucial features from CAM with images from more common classes. The proposed framework is evaluated on several datasets, including Food101-LT,

VFN-LT, VFN-INSULIN, and VFN-T2D. The results demonstrate substantial improvements over existing methods, particularly in handling long-tailed distributions and continual learning without the need for detached training stages. The authors also conduct an ablation study to evaluate each component's effectiveness and suggest further improvements for real-world deployment.

In [3], the authors highlight that traditional food classification systems often rely on static datasets with fixed classes and uniform distributions, which limits their effectiveness in real-world scenarios where food consumption patterns are dynamic and influenced by a range of factors such as cultural, economic, and personal preferences. To address this challenge, the paper introduces Online Class Incremental Learning (OCIL) as a more adaptable approach. OCIL methods are designed to continuously learn from an ongoing stream of data, allowing the system to adapt to new information while minimizing the risk of losing previously acquired knowledge when new data is introduced. One of the key techniques within OCIL is Experience Replay (ER), which involves storing a small portion of past data to improve learning. However, existing OCIL methods often assume that the data is perfectly balanced, which is not the case in real-world food consumption scenarios. To bridge this gap, the authors propose an innovative framework that includes the Realistic Data Distribution Module (RDDM) and the Dynamic Model Update (DMU) module. The RDDM framework is designed to simulate realistic food consumption patterns, providing a more accurate representation of the dynamic nature of food data. Complementing this, the DMU module is developed to enhance existing ER methods by optimizing the selection of the most relevant training samples. This addresses common issues such as data repetition and imbalanced sample distributions, which are prevalent in real-world food classification tasks. The paper demonstrates how their approach, applied to various food consumption scenarios—short-term, moderate-term, and long-term—using challenging datasets like Food-101 [48] and VFN [87], leads to significant improvements in classification performance.

2.3 Personalized Food Classification

The area of personalized food classification is relatively under-explored, presenting a compelling opportunity to advance food classification based on individual eating habits. In its early

stages, [88] proposed an innovative approach that incorporates temporal information into the classification process. The proposed method leverages recursive Bayesian estimation to incrementally learn from an individual's eating history. By integrating this temporal aspect, the system can build a more accurate profile of a person's dietary habits over time. This approach significantly enhances food classification performance, achieving an 11% improvement in accuracy compared to existing methods of that time. The paper also outlines the development of a dietary assessment system, known as the mobile Food Record (mFR). This system aims to automatically estimate food type, volume, nutrients, and energy from food images captured via mobile devices. The mFR system is composed of a web-based user interface, a mobile application, and a backend infrastructure that includes a computational server and a database system.

Also, [89] introduces a novel approach to enhancing food image classification through incremental updates to a Bayesian network model. This paper focuses on improving the accuracy of a food log system, which leverages image processing to detect and analyze food images, estimating nutritional balance and allowing users to modify the analysis results when errors are detected. The authors address the challenge of food image classification, which is often complicated by factors such as diverse food appearances and variations in serving styles. They present a method to incrementally update the Bayesian network classifier using user corrections, enhancing the system's performance. The proposed method significantly improves the classifier's performance by integrating corrections made by users. This iterative learning process enhances the system's accuracy from 89% to 92%. The paper compares the performance of the Bayesian network with traditional support vector machines and highlights the advantage of using user feedback to refine the model. This incremental learning approach allows the food log system to personalize its analysis for each user, adapting to individual variations and improving overall classification effectiveness.

Furthermore, several research papers achieved personalized food classification by leveraging the nearest class mean classifier and the 1-nearest neighbor classifier [90, 91, 92].

In [90], the authors introduce a novel approach, the Sequential Personalized Classifier (SPC) framework, that is designed to incrementally adapt to individual users' dietary patterns using a very limited number of samples. The SPC combines the nearest class mean classifier with a 1-nearest neighbor classifier based on deep features. This combination allows the classifier to handle the

dynamic nature of real-world data effectively, bridging the gap between controlled laboratory conditions and practical applications. The proposed method was evaluated using a dataset of daily food images collected through a food-logging application, which captures the variability and incremental growth of food classes over time. The results show that the SPC framework significantly outperforms existing methods by effectively personalizing the classifier for each user and adapting to new classes without the need for extensive retraining.

Finally, in more recent approaches, [12] utilizes self-supervised learning to classify two new benchmark datasets, FOOD101-Personal and VFN-Personal, incorporating personalization. Traditional methods often use deep neural networks trained on static datasets, which do not reflect the dynamic nature of real-world food consumption patterns. These patterns involve sequential food images that capture an individual's dietary habits over time, which is crucial for accurate nutrient analysis. To address this gap, the paper introduces two benchmark datasets designed to better capture individualized food consumption patterns: Food101-Personal and VFN-Personal. These datasets were created from real-world dietary surveys and studies, providing a more realistic foundation for testing personalized food classification methods. The authors propose a new framework that combines self-supervised learning with temporal image feature extraction. This approach updates the classifier incrementally as new food images are introduced, enabling the model to adapt to changes in an individual's diet over time. By leveraging self-supervised learning, the model can continuously improve its feature extraction process, while the temporal contextual information is enhanced through a sliding window technique that captures the evolving nature of food consumption. The evaluation of this framework on the newly introduced datasets demonstrates a significant improvement in classification performance compared to existing methods. This work advances the field of personalized food image classification by addressing key limitations of previous approaches, including the lack of dynamic learning and the need for temporal context in dietary patterns. The datasets and methods introduced in this paper offer valuable resources and techniques for developing more accurate and personalized dietary assessment tools.

Despite their innovative approaches and promising results, these previous models focused solely on food images and sequences of food images, neglecting crucial contextual information like meal

time, meal location, and meal frequency. As a result, their accuracy fell short for practical application in real-world scenarios. These limitations motivate us to introduce a new approach designed to address these issues.

Chapter 3

Food Intake Monitoring System

3.1 Personalized Food Intake Monitoring Framework

This section introduces a personalized food intake monitoring framework, as illustrated in Fig. 3.1. The core component of this framework is the personalized dynamic support network (PDSN), which classifies food images collected via our mobile application by integrating user profile history data, including meal frequency, meal time, and meal location.

Upon each food detection, user feedback regarding the accuracy of the detection is solicited, enabling continuous updates to our personalized module. This feedback mechanism allows the model to determine whether the detected food item is already included in the existing class set. If the food item is part of the existing classes, the system retrieves nutritional information from the nutrient database, including macronutrient content such as protein, carbohydrate, fat, and calorie values.

Subsequently, the mobile application interfaces with a smart scale via Bluetooth to ascertain the weight of the meal. By combining the meal weight with the nutritional data from the nutrient database, we accurately calculate the user's macronutrient intake for that particular meal. In instances where the detected food item is not part of the existing class set, the continual learning capability of PDSN is employed to train the model to recognize new classes. Additionally, food image segmentation techniques are applied to identify the ingredients of the new food item, enabling the calculation of its macronutrient content. This new information is then incorporated into the nutrient

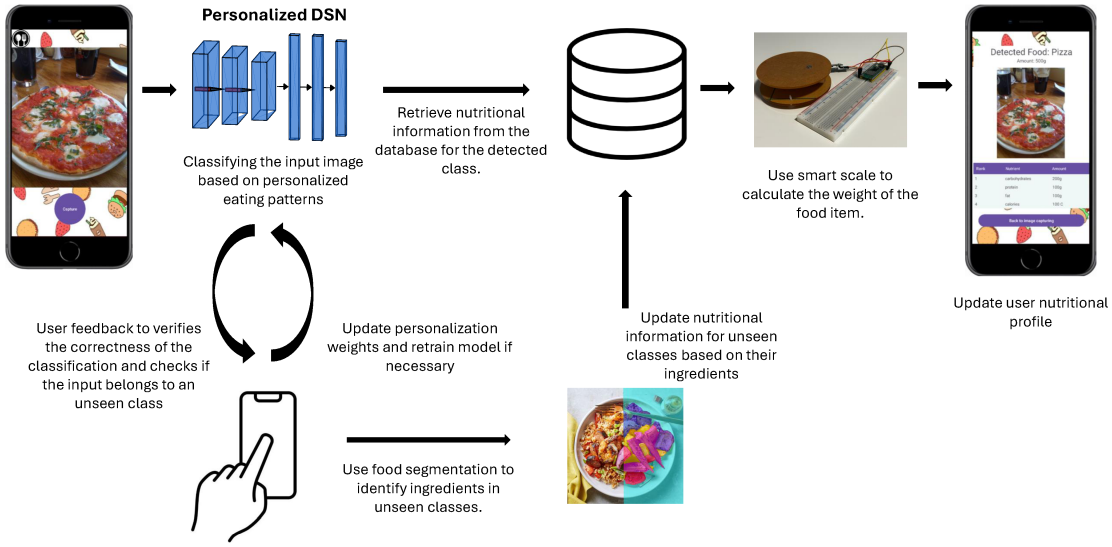


Figure 3.1: Overview of our framework for food intake monitoring system leveraging PDSN.

database.

This personalized approach ensures that the monitoring framework adapts to individual eating habits and preferences, thereby providing accurate and tailored nutritional information to the user.

3.2 Personalizer Plug-in

Existing personalized food classification models often utilize eating patterns as time series data to personalize the classifier [88, 89, 12]. However, information such as meal frequency, meal time, and meal location are specific to the user and can be leveraged to achieve even greater accuracy and personalization of the model. To utilize this information, we define the following:

Let F be the set of existing food classes, T be the discrete set of times for eating, and L be the discrete set of locations where the user is eating.

For each user $u \in U$, we introduce a vector $\mathbf{MF}^u$ that holds the probability $P(F = f)$ of consuming a specific food, considering the frequency of appearance of that food in the user's profile history. This vector is denoted as $\mathbf{MF}^u \in \mathbb{R}^{|F|\times 1}$, where MF^u_f represents the probability of

consuming food type $f \in F$ for user u. The sum of MF_f^u for f in the range 0 to $|F|$ is calculated as

$$\sum_{f=0}^{|F|} \text{MF}_f^u = 1, \tag{1}$$

where MF_f^u is uniformly distributed across all food classes at the initialization phase.

We define the matrix $\mathbf{MT}^u \in \mathbb{R}^{|F| \times |T|}$ that represents the conditional probability $P(T = t \mid F = f)$ of consuming a meal at a specific time $t \in T$ for a given food type, considering the user's profile history, where $\text{MT}_{f,t}^u$ represents the probability of consuming food type f at time t for user u. The sum of $\text{MT}_{f,t}^u$ for t in the range 0 to $|T|$ is calculated as

$$\sum_{t=0}^{|T|} \text{MT}_{f,t}^u = 1, \tag{2}$$

where $\text{MT}_{f,t}^u$ is uniformly distributed across all times for each food class at the initialization phase.

We define the matrix $\mathbf{ML}^u \in \mathbb{R}^{|F| \times |L|}$ that represents the conditional probability $P(L = l \mid F = f)$ of consuming a meal at a specific location $l \in L$ for a given food type, considering the user's profile history, where $\text{ML}_{f,l}^u$ represents the probability of consuming food type f at location l for user u. The sum of $\text{ML}_{f,l}^u$ for l in the range 0 to $|L|$ is calculated as

$$\sum_{l=0}^{|L|} \text{ML}_{f,l}^u = 1, \tag{3}$$

where $\text{ML}_{f,l}^u$ is uniformly distributed across all locations for each food class at the initialization phase.

Upon receiving each input image $I \in \mathbb{R}^{n \times n}$, it undergoes classification processes using classifiers C to generate a probability distribution P across the classes. Specifically, $C(I) = P$ represents the transformation of input image I to the probability distribution P. Let P_f^u denote the probability of class f for user u based on the input image.

The personalized probability distribution PP^u can be expressed as

$$PP_f^u = P_f^u \cdot \text{MF}_f^u \cdot \text{MT}_{f,t_i}^u \cdot \text{ML}_{f,l_i}^u, \tag{4}$$

where $u \in U$, $t_i \in T$, and $l_i \in L$ represent the user, time, and location associated with the input image, respectively.

The detected class is the one with the maximum value in the personalized probability distribution and expressed as

$$\text{Detected Class} = f_i = \arg\max_f (PP_f^u). \tag{5}$$

After detecting the food type from the input image, if the model's prediction is incorrect, the correct class will be requested. Based on this correct class, the personalization vectors will then be updated as follows.

Let α_f, α_t, and α_l be the forgetting factors for meal frequency, meal time, and meal location, respectively. These values will be used to update the personalization vectors in such a way that the probability of detecting foods more important and specific to the user will increase, while the probability of detecting less relevant foods will decrease by the forget factors.

Updating of the food probability vector $\mathbf{MF}_f^u$ expressed as

$$\mathbf{MF}_f^u = \begin{cases} \mathbf{MF}_f^u + \alpha_f \cdot (1 - \mathbf{MF}_f^u), & \text{if } f = f_i, \\ \mathbf{MF}_f^u \cdot (1 - \alpha_f), & \text{otherwise.} \end{cases} \tag{6}$$

Updating of the time probability matrix $\mathbf{MT}_{f,t}^u$ expressed as

$$\mathbf{MT}_{f,t}^u = \begin{cases} \mathbf{MT}_{f,t}^u + \alpha_t \cdot (1 - \mathbf{MT}_{f,t}^u), & f = f_i\,, t = t_i, \\ \mathbf{MT}_{f,t}^u \cdot (1 - \alpha_t), & f = f_i\,, t \neq t_i, \\ \mathbf{MT}_{f,t}^u, & \text{otherwise.} \end{cases} \tag{7}$$

Updating of the location probability matrix $\mathbf{ML}_{f,l}^u$ expressed as

$$\mathbf{ML}_{f,l}^u = \begin{cases} \mathbf{ML}_{f,l}^u + \alpha_l \cdot (1 - \mathbf{ML}_{f,l}^u), & f = f_i\,, l = l_i, \\ \mathbf{ML}_{f,l}^u \cdot (1 - \alpha_l), & f = f_i\,, l \neq l_i, \\ \mathbf{ML}_{f,l}^u, & \text{otherwise.} \end{cases} \tag{8}$$

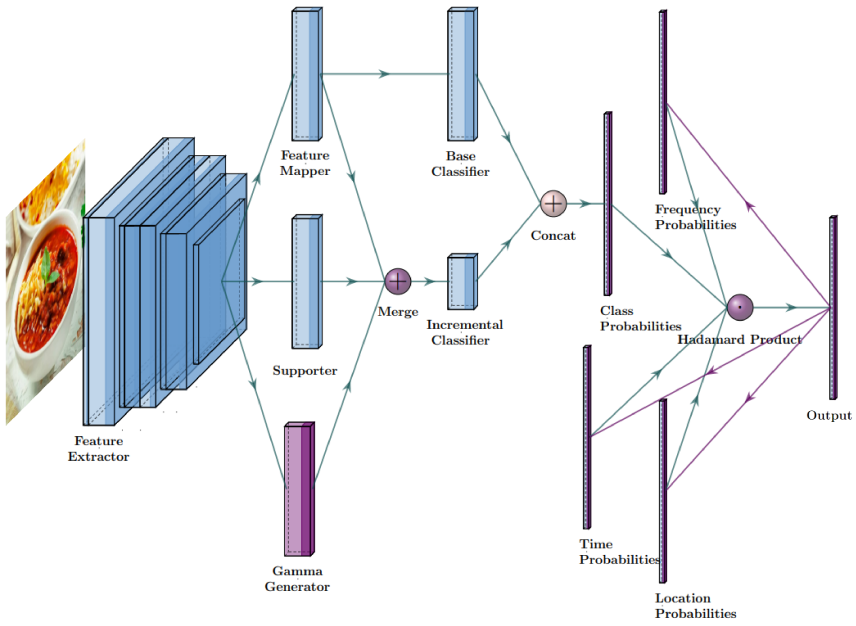


Figure 3.2: Architecture of PDSN showcasing its innovative adaptive food image classification design.

In the mentioned equations, the probability of detecting less relevant food types will be decreased by a forgetting factor, and this reduction will be reallocated to the probabilities of more relevant foods for the user. Moreover, in this way, we ensure that equations (1) to (3) remain the same, and by incorporating these user-specific probabilities, our method aims to enhance the accuracy of personalized food classification models by leveraging detailed and individualized information on meal frequency, meal time, and meal location.

3.3 Personalized Dynamic Support Network

In this section, we propose a modified version of DSN [11]. The overall architecture follows the DSN structure but introduces an improvement for personalized food classification through incremental learning, as illustrated in Fig. 3.2.

The proposed model consists of the following key components:

Backbone Feature Extraction

The model utilizes a backbone feature extractor, such as ResNet [28], to extract features from the input images that can be used for classifying the input. The extracted features can be expressed as

$$\mathbf{h} = \text{backbone}(x), \tag{9}$$

where x is the input image, and *backbone* represents the model used for feature extraction.

Feature Mapping

The extracted features are mapped to a new feature space using a fully connected layer, converting the raw features into a form suitable for classification, as follows

$$\mathbf{z} = \mathbf{W}_{fm}\mathbf{h}, \tag{10}$$

where $\mathbf{W}_{fm}$ is the weight matrix of the fully connected layer. These features are then normalized to ensure consistent scale and improve the performance of the classification model, as follows

$$\mathbf{z} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}. \tag{11}$$

Base Class Classification

The normalized features are passed through another fully connected layer for the classification of base classes as follows

$$\text{output}_0 = \frac{\mathbf{W}_0\mathbf{z}}{\|\mathbf{W}_0\|_2\|\mathbf{z}\|_2}, \tag{12}$$

where $\mathbf{W}_0$ is the weight matrix of the base classifier.

Gamma Generation for Incremental Sessions

The output of the feature extracting layer is also passed to another fully connected neural network to generate gamma values for incremental sessions as follows

$$\gamma = \text{relu}(\mathbf{W}_\gamma \mathbf{h}), \tag{13}$$

where $\mathbf{W}_\gamma$ is the weight matrix of the gamma generator.

Session-specific Classifiers

For each incremental session $i \geq 1$, a support mechanism maps the features to the appropriate space for incremental classes. These mapped features are then merged with the features generated for the base classes to incorporate their information, regulated by the responsible γ. Finally, the output is calculated using the merged features as follows

$$\text{supporter}_i = \frac{\mathbf{W}_{s,i}\mathbf{h}}{\|\mathbf{W}_{s,i}\|_2 \|\mathbf{h}\|_2}, \tag{14}$$

$$\mathbf{z}_i = \gamma_i \mathbf{z} + \text{supporter}_i, \tag{15}$$

$$\mathbf{z}_i = \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2}, \tag{16}$$

$$\text{output}_i = \frac{\mathbf{W}_i \mathbf{z}_i}{\|\mathbf{W}_i\|_2 \|\mathbf{z}_i\|_2}, \tag{17}$$

where $\mathbf{W}_{s,i}$ is the weight matrix of the supporter for session i, and $\mathbf{W}_i$ is the weight matrix of the classifier for session i.

Concatenation of Outputs

The class probabilities is a concatenation of the outputs from all sessions obtained as

$$\text{output} = [\text{output}_0 \| \ldots \| \text{output}_i \| \ldots \| \text{output}_{sess}] \tag{18}$$

where output_i represents the output at session i, $\|$ denotes the concatenation operation, and $sess$

denotes the total number of sessions.

Personalizer

At the end, the personalizer plug-in will be used to incorporate contextual information such as meal frequency, meal time, and meal location further to enhance the model's performance for that specific user.

Our approach's primary improvement over the original DSN is the personalized plug-in and network's dynamic generation of gamma. In the original DSN, gamma is treated as a hyperparameter requiring manual tuning. Our approach enables the network to dynamically adjust the impact of the base class feature mapper and the incremental class feature mappers, thereby improving classification performance and adaptability to new data.

By learning gamma values, the network can better decide the relative importance of base and incremental class features for each specific data point, leading to more accurate and personalized food classification.

This methodology outlines the design and implementation of our personalized dynamic support network (PDSN) model highlighting the architectural innovations and the theoretical foundations supporting its improved performance in incremental learning scenarios.

Chapter 4

Implementation and Experimental Setup

4.1 Datasets

Our experiment utilized two primary datasets: the Food-101 dataset and the VFN dataset. Additionally, we integrated personalized versions of these datasets, referred to as Food-101-Personal and VFN-Personal, for evaluating personalized food image classification tasks.

The Food-101 dataset [48] is a widely used benchmark for food image classification, comprising 101,000 food images, with each class having 750 training images. It covers a wide spectrum of food categories essential for training machine learning models in food image classification tasks.

The Food-101-Personal dataset was derived through an online survey conducted by [12] using the Food-101 dataset. Participants simulated one week of food consumption patterns by selecting foods from the FOOD101 categories. This personalized dataset includes 20 patterns with an average of 44 food classes and 6000 images in total. It comprises 300 images per individual, representing long-term food consumption patterns for enhanced evaluation in personalized food image classification tasks.

The VFN dataset [87] comprises 14,991 online food images sourced from the What We Eat In America (WWEIA) database [93]. The dataset is designed specifically for food recognition and includes images from various food categories. The characteristics of VFN make it highly relevant for evaluating the generalizability of our models. This dataset covers 82 food categories selected based on high intake frequency from the WWEIA database, reflecting the most commonly consumed

foods in the United States.

The VFN-Personal dataset originated from a dietary examination conducted by [12] involving healthy individuals aged 18 to 65, employing an image-based dietary evaluation approach. Participants documented their food intake over a three-day period. Much like the Food-101-Personal dataset, the VFN-Personal dataset was crafted utilizing techniques to mimic prolonged food consumption behaviors. The dataset comprises 26 patterns, each encompassing an average of 29 distinct classes and similar to the Food-101-Personal dataset, it includes 300 images per individual.

4.2 Baseline Models

Our study centers on assessing the performance of seven widely recognized architectures known for their image classification capabilities: AlexNet [25], VGGNet [26], GoogLeNet [27], ResNet [28], DenseNet [29], Vision Transformer [44], and Swin Transformer [45].

We selected these architectures based on their track record of high performance in image classification tasks. To ensure a fair comparison, we utilized pretrained versions of these models trained on the ImageNet [94] dataset. Fine-tuning was performed specifically for food image classification, involving 20 epochs of training to adapt the models to our dataset.

We conducted a comprehensive comparison between the baseline models and their enhanced versions. The enhancements include the incorporation of our proposed personalized plugin, designed to improve classification accuracy by tailoring models to individual eating habits.

To assess the effectiveness of our model in learning new classes incrementally, we compared DSN [11] and PDSN architectures based on the accuracy of different datasets using the same backbone to ensure a fair comparison. This setup allowed us to study how our approach facilitates better learning of new classes over time.

4.3 Implementation Details

4.3.1 PDSN

We utilized various pretrained architectures implemented in Python with the PyTorch [95] library as the backbones of DSN and PDSN to extract features. Each architecture was fine-tuned for food image classification tasks with a batch size of 32 and trained for 20 epochs. Input images were resized to 224×224 pixels to accommodate the requirements of different architectures. For optimization, we employed the Stochastic Gradient Descent (SGD) optimizer with a learning rate (lr) of 0.001, momentum of 0.9, weight decay of 0.0005, and nesterov momentum enabled. To incorporate meal time and location information, each input image had a fixed probability of being associated with different meal times and locations based on the type of food. This probabilistic approach introduced variability in meal times and locations, mitigating potential overfitting. Also forget factors, α_f, α_t, and α_l were set to 0.003, 0.04, and 0.04, respectively. We trained the models on the FOOD101 and VFN datasets and subsequently evaluated their performance using the FOOD101-Personal and VFN-Personal datasets, which contain personalized eating patterns across 300 meal sessions. The development codes of PDSN are illustrated in appendix A. The core of the model is encapsulated in the "PersonalizedDSN" class in "model.py" file, which inherits from "nn.Module", making it a PyTorch model. Below is a breakdown of how the model operates:

(1) Initialization

- The model dynamically selects a backbone architecture (e.g., ResNet, AlexNet, DenseNet, etc.) based on the args passed during initialization. Each architecture is initialized with pretrained weights for better feature extraction.
- Depending on the dataset (FOOD101 or VFN), the model sets a session length for personalization.
- The final layer of each backbone architecture is customized (replaced or reconfigured) to produce a fixed-size feature vector (node of size 512).
- The model defines a series of classifiers and supporting layers to handle different sessions of incremental classification as explained in 3.3.

(2) Forwarding

- The input image is passed through the selected backbone to extract features.
- The features are then passed through a feature-mapper layer and normalized using the l2norm method.
- A gamma generator layer produces gamma values that modulate the influence of different sessions in the incremental classification.
- The output from the base session classifier is generated first. For subsequent sessions, a new node is computed by combining the current session's features and a supporting layer's output.
- The final output is a concatenation of the base and personalized session outputs.

The model leverages meal frequency, meal time, and location data to influence the classification process. This is implemented in the script for personalization in "pdsn.py", where the output of the model is element-wise multiplied by MF, MT, and ML matrices. The MF, MT, and ML matrices are updated incrementally with each input, where the corresponding entries are updated based on the true label, enabling personalization. Below is a breakdown of how the pdsn.py operates:

- The script loads the personalized DSN model and initializes it with the specified backbone and dataset.
- It applies various transformations to the input images and feeds them through the model.
- The script reads meal time and location probabilities from CSV files, which indicate the likelihood of a user consuming certain foods at specific times of the day or locations.
- the script simulates the eating patterns of different users to test the model's performance in a personalized context.
- The adjustments explained in 3.2 are applied by element-wise multiplication of the model's output with the MF, MT, and ML matrices.
- After each prediction, the script updates the MF, MT, and ML matrices to reflect the new information about the user's eating habits.

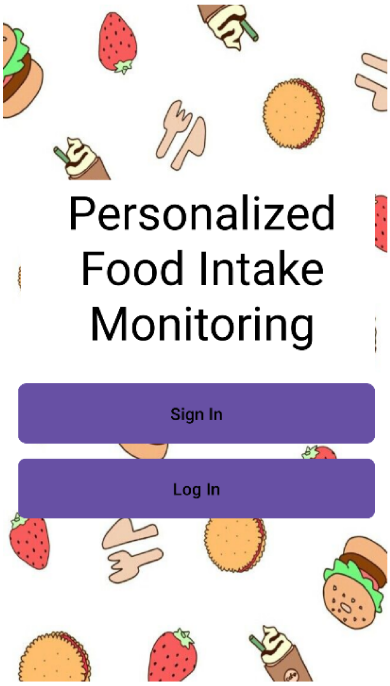


Figure 4.1: Authentication Page of Mobile Application

- The script calculates the accuracy of the model's predictions, both with and without personalization.

4.3.2 Food Intake Monitoring Framework

Our framework and mobile application are designed to seamlessly integrate cutting-edge technology with user-friendly features to deliver a robust and efficient solution. In this section, we delve into the implementation of our framework and its core components highlighting how they work together to provide a seamless user experience.

Mobile Application

In this section, we provide a detailed overview of our mobile application, developed using Kotlin and designed to run on the widely used Android OS. The application serves as the primary interface between the user and our underlying framework, enabling seamless interactions and functionalities.

We begin with the user authentication process. As illustrated in Fig. 4.1, users can choose to sign up or log in to our system. As shown in Fig. 4.2, new users must provide their email, full name,

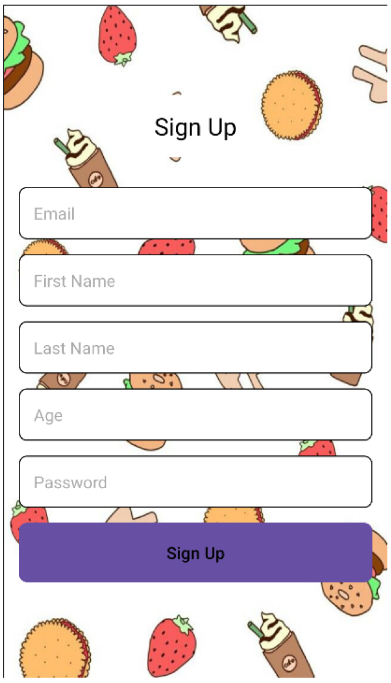


Figure 4.2: Sign Up Page of Mobile Application

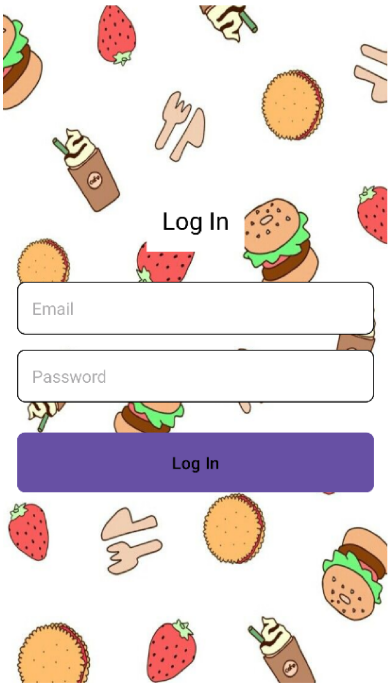


Figure 4.3: Login Page of Mobile Application

age, and password to create an account. This information is used to personalize their experience and securely store their data. For returning users, Fig. 4.3 demonstrates the login process where users

enter their email and password to access their accounts.

Upon successful login, users are directed to the main page, depicted in Fig. 4.4. This page serves as the hub for key functionalities, including:

- Taking pictures of meals
- Viewing their profile
- Connecting to a nearby smart scale

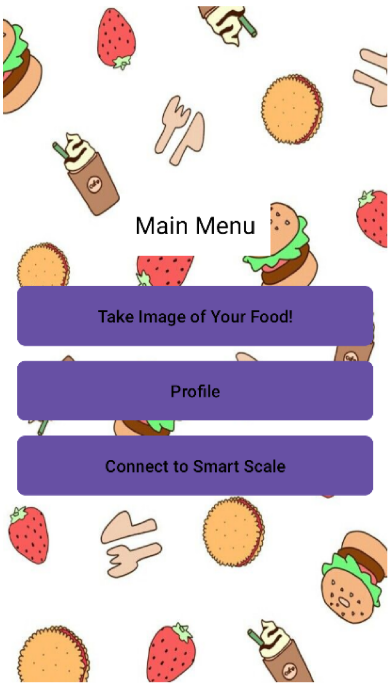


Figure 4.4: Main Page of Mobile Application

To connect to a smart scale, users can simply click the relevant button on the main page. As shown in Fig. 4.5, the application automatically scans the network for available smart scales. If a connection is established successfully, the button indicator turns green, signalling a successful pairing.

The user profile page, illustrated in Fig. 4.6, provides insights into the user's nutritional intake. Here, users can view:

- The amount of macronutrients consumed over a selected period

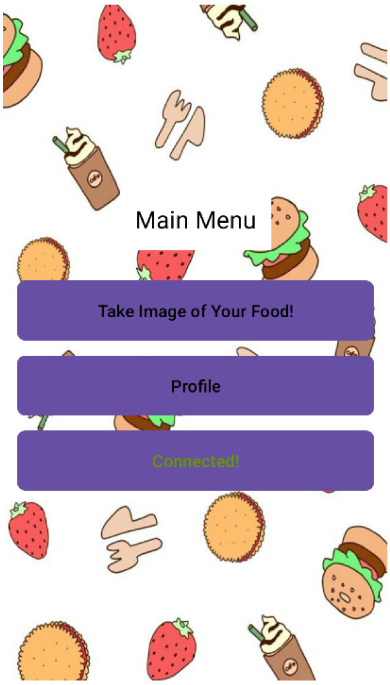


Figure 4.5: Main Page of Mobile Application after Connecting to Smart Scale

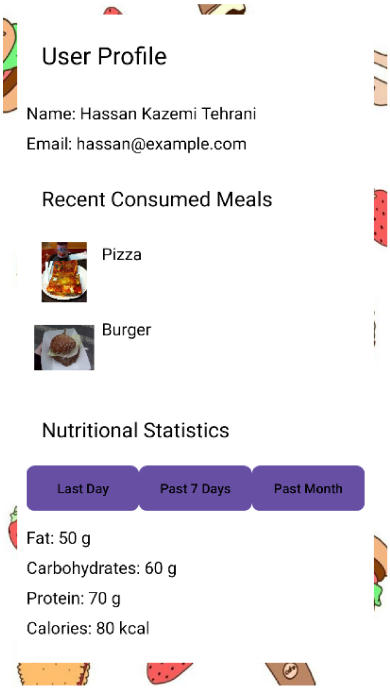


Figure 4.6: Profile Page of Mobile Application

- The list of foods consumed in each meal

When users begin capturing images of their meals, they are directed to a dedicated page for

image capture, as shown in Fig. 4.7. This step allows users to take a picture of their food for analysis.

Figure 4.7: Image Capture Page of Mobile Application

After capturing an image, the application processes it and displays the predicted food type, weight, and estimated macronutrients on the next page, as depicted in Fig. 4.8. Users can review this information to ensure accuracy.

- If the predicted food type is correct, users can click ”Submit” to finalize the entry.
- If the prediction is incorrect, users can click ”Modify” to direct to the modify page as shown in Fig. 4.9. This option allows them to correct the food type manually and specify if the food is new. If marked as new, the model will retrain itself to include this new food type in the classification set.

Once the food information is confirmed and submitted, the user profile is updated with the new data. Additionally, as explained in chapter 3, the server-side model associated with the user is updated to reflect any changes or new data.

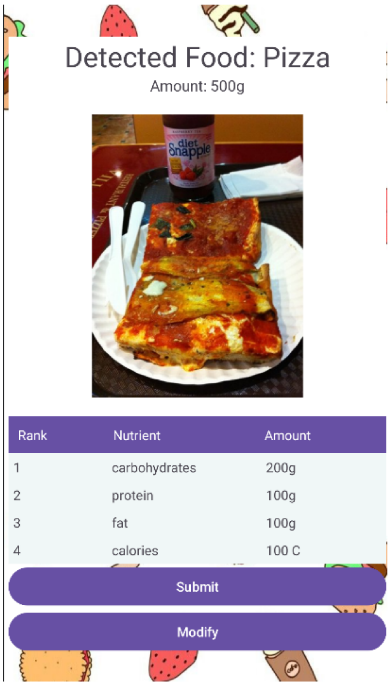


Figure 4.8: Result Page of Mobile Application

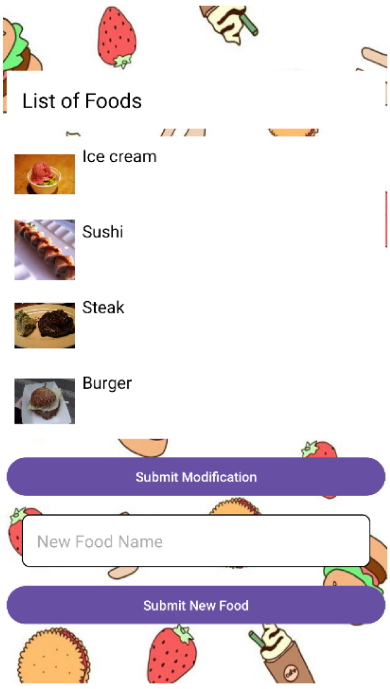


Figure 4.9: Modify Page of Mobile Application

Smart Scale

In this part, we delve into the implementation of our smart scale, which is designed to interface seamlessly with our mobile application. Fig. 4.10 illustrates the smart scale design and component

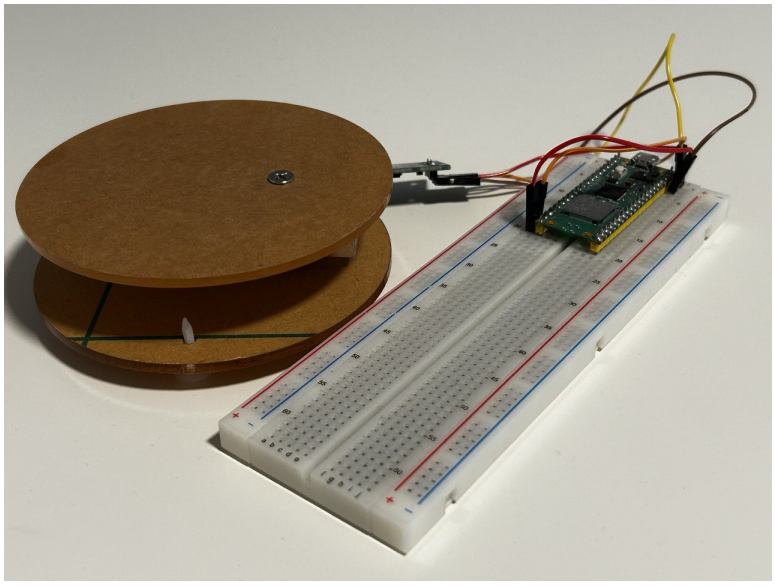


Figure 4.10: Smart Scale Design and Component Assembly

assembly. The smart scale system is built using Micropython, a lightweight Python implementation for microcontrollers, and leverages a Raspberry Pi Pico as the microcontroller. The setup also includes a 5kg load cell and an HX711 module to accurately measure weight.

The Raspberry Pi Pico, shown in Fig. 4.11, is a versatile microcontroller based on the RP2040 chip. It provides sufficient processing power and connectivity options to handle the smart scale's functions. It is chosen for its affordability and ease of integration with various sensors and modules.

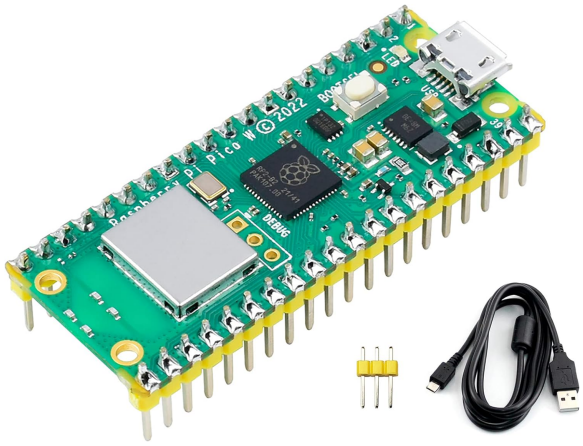


Figure 4.11: Raspberry Pi Pico

The 5kg load cell, depicted in Fig. 4.12, converts the physical weight into an electrical signal.

This signal is then processed by the microcontroller. The load cell is essential for accurate weight measurement and ensures that the scale can handle a range of weights typically encountered in food measurement scenarios.

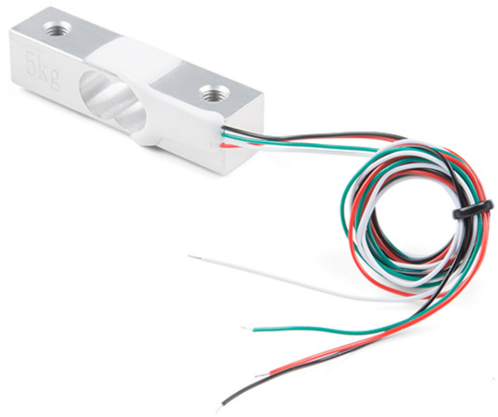


Figure 4.12: 5Kg Load Cell

The HX711 module, illustrated in Fig. 4.13, is an analog-to-digital converter designed specifically for weighing scales. It amplifies the signal from the load cell and converts it into a digital format that the Raspberry Pi Pico can process. This module is crucial for obtaining precise and stable weight readings from the load cell.

The smart scale system is programmed using Micropython, which allows for efficient scripting on the Raspberry Pi Pico. The Micropython code handles the following tasks:

- Initialization: The code initializes the HX711 module and calibrates the load cell to ensure accurate measurements.
- Weight Measurement: It continuously reads data from the load cell via the HX711 and converts the analog signal into a digital weight reading.
- Network Communication: The Raspberry Pi Pico connects to the network and listens for incoming connections from the mobile application. This is achieved using Micropython's networking libraries.

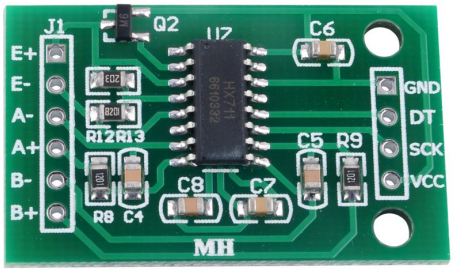


Figure 4.13: HX711 ADC Module

The smart scale is designed to be network-aware. It performs the following steps to communicate with the mobile application:

- Connection: Once powered on, the smart scale connects to the network and waits for incoming connections from the mobile application. It listens on a specified port for connection requests.
- Communication: After establishing a connection, the smart scale listens for commands from the mobile application. When a command to measure weight is received, the smart scale triggers the measurement process.
- Measurement and Response: The smart scale measures the weight and returns the result to the mobile application. This response is sent back over the network, ensuring that the weight data is transmitted accurately and promptly.

The smart scale remains in a listening state, continuously monitoring for commands from the mobile application. This setup ensures that the scale is ready to measure and provide weight data whenever requested, facilitating real-time interaction and data collection.

Chapter 5

Experimental Results

5.1 Improvements with Personalized Plug-In

We evaluated the performance of our PDSN model alongside seven baseline architectures for food classification using the FOOD101-Personal and VFN-Personal datasets. Tables 5.1 and 5.2 present the comparative performance metrics of these models, both with and without our personalizer plug-in.

Initially, we trained all baseline models—AlexNet, VGGNet, GoogLeNet, ResNet, DenseNet, Vision Transformer, and Swin Transformer—on the standard FOOD101 and VFN datasets. Table 5.3 summarizes the accuracy results of these baseline models after training them for 20 epochs on FOOD101 and VFN. After evaluating the baseline models, we integrated our personalizer plug-in into each architecture and re-evaluated their performance on the personalized versions of these datasets, FOOD101-Personal and VFN-Personal, with and without enabling personalization.

FOOD101, being a well-populated dataset, demonstrated moderate improvements with the personalizer plug-in due to its ample data availability. Conversely, the VFN dataset, which lacks extensive data and initially had lower baseline model accuracy, benefited substantially from the personalizer plug-in. In some instances, we observed performance boosts of up to 10% in classification accuracy.

Our personalizer plug-in significantly improved the models' performance by customizing the classification process according to individual eating patterns. This personalization leverages unique

Table 5.1: Average Top-1 Classification Accuracy ± std for FOOD101-Personal at different time steps

Model	**FOOD101-Personal**			
	$\mathbf{t}_{75}$	$\mathbf{t}_{150}$	$\mathbf{t}_{225}$	$\mathbf{t}_{300}$
AlexNet	92.26 ± 0.03	92.16 ± 0.02	91.37 ± 0.02	91.35 ± 0.02
VGGNet	94.93 ± 0.02	94.80 ± 0.01	94.53 ± 0.01	94.53 ± 0.01
GoogLeNet	94.73 ± 0.02	94.33 ± 0.01	94.06 ± 0.01	93.75 ± 0.01
ResNet	94.60 ± 0.02	94.63 ± 0.01	94.37 ± 0.01	94.26 ± 0.01
DenseNet	95.53 ± 0.01	95.63 ± 0.01	95.35 ± 0.01	95.30 ± 0.01
Vision Transformer	97.46 ± 0.01	96.90 ± 0.01	96.88 ± 0.01	96.91 ± 0.01
Swin Transformer	97.46 ± 0.01	97.36 ± 0.01	96.95 ± 0.01	96.93 ± 0.01
(Our Work)				
P-AlexNet	92.86 ± 0.03	93.53 ± 0.02	93.55 ± 0.02	93.65 ± 0.02
P-VGGNet	95.46 ± 0.02	95.63 ± 0.01	95.71 ± 0.01	95.80 ± 0.01
P-GoogLeNet	95.66 ± 0.02	95.33 ± 0.01	95.51 ± 0.01	95.23 ± 0.00
P-ResNet	95.26 ± 0.02	95.63 ± 0.01	95.71 ± 0.01	95.46 ± 0.01
P-DenseNet	95.93 ± 0.01	96.46 ± 0.01	96.44 ± 0.01	96.38 ± 0.01
P-Vision Transformer	97.73 ± 0.01	97.53 ± 0.01	97.37 ± 0.01	97.35 ± 0.00
P-Swin Transformer	$\mathbf{97.86 \pm 0.01}$	$\mathbf{98.06 \pm 0.01}$	$\mathbf{97.73 \pm 0.01}$	$\mathbf{97.45 \pm 0.01}$

The "P-" prefix denotes the integration of our personalizer plug-in to the baseline models.

patterns in a user's food consumption, allowing the model to focus on the most relevant foods for each individual. By integrating personalization data into the classification process, our approach effectively reduces the negative effects of intra-class diversity and inter-class similarity by considering additional factors specific to each user's eating habits. This tailored learning process helps the models differentiate between similar food images in different classes and distinguish diverse food images within the same class more effectively.

5.2 Improvement Over Original Dynamic Support Network

To demonstrate the efficacy of our PDSN model in incremental learning scenarios, we conducted experiments, comparing to the original DSN architecture. Both models were initially trained on the complete set of classes from FOOD101 and VFN datasets separately, serving as the base session. Subsequently, we introduced two new classes to each dataset and evaluated their performance on

Table 5.2: Average Top-1 Classification Accuracy ± std for VFN-Personal at different time steps

Model	**VFN-Personal**			
	$\mathbf{t}_{75}$	$\mathbf{t}_{150}$	$\mathbf{t}_{225}$	$\mathbf{t}_{300}$
AlexNet	58.46 ± 0.08	58.74 ± 0.06	58.87 ± 0.07	58.51 ± 0.07
VGGNet	62.66 ± 0.07	62.71 ± 0.06	63.00 ± 0.06	62.91 ± 0.06
GoogLeNet	62.15 ± 0.07	62.58 ± 0.07	63.14 ± 0.08	63.03 ± 0.08
ResNet	62.46 ± 0.07	63.12 ± 0.07	62.90 ± 0.08	62.55 ± 0.07
DenseNet	67.94 ± 0.08	69.30 ± 0.07	69.55 ± 0.07	69.20 ± 0.07
Vision Transformer	71.94 ± 0.06	73.00 ± 0.07	73.43 ± 0.07	73.14 ± 0.07
Swin Transformer	70.97 ± 0.07	71.79 ± 0.07	71.91 ± 0.07	71.61 ± 0.06
(Our Work)				
P-AlexNet	62.71 ± 0.07	66.28 ± 0.08	68.88 ± 0.08	69.78 ± 0.08
P-VGGNet	68.87 ± 0.07	72.48 ± 0.07	74.58 ± 0.07	75.35 ± 0.07
P-GoogLeNet	67.69 ± 0.08	71.46 ± 0.08	73.77 ± 0.07	74.19 ± 0.07
P-ResNet	67.12 ± 0.07	71.69 ± 0.08	73.98 ± 0.07	74.60 ± 0.07
P-DenseNet	73.48 ± 0.08	76.58 ± 0.08	78.56 ± 0.07	78.69 ± 0.07
P-Vision Transformer	76.61 ± 0.06	**79.76 ± 0.07**	**81.50 ± 0.06**	81.66 ± 0.07
P-Swin Transformer	**76.87 ± 0.07**	79.64 ± 0.07	81.45 ± 0.06	**81.74 ± 0.06**

The "P-" prefix denotes the integration of our personalizer plug-in to the baseline models.

Table 5.3: Best Top-1 Accuracy of Classification Architectures on FOOD101 and VFN after 20 Epochs

Model	**FOOD101**	**VFN**
AlexNet	64.49	86.07
VGGNet	76.65	**86.46**
GoogLeNet	75.74	86.34
ResNet	75.08	86.04
DenseNet	81.80	86.07
Vision Transformer	84.37	86.34
Swin Transformer	**87.86**	85.75

Table 5.4: Accuracy Breakdown of DSN and PDSN Models on FOOD101 and FOOD101-Personal

Model	**FOOD101**			**FOOD101-Personal**		
	Base	**New**	**Total**	**Base**	**New**	**Total**
DSN	**74.85**	73.33	**74.82**	94.26	72.42	92.28
PDSN (Our Work)	74.73	**76.66**	74.77	**95.26**	**77.54**	**93.65**

four datasets: FOOD101, VFN, FOOD101-Personal, and VFN-Personal.

Table 5.5: Accuracy Breakdown of DSN and PDSN Models on VFN and VFN-Personal

Model	**VFN**			**VFN-Personal**		
	Base	**New**	**Total**	**Base**	**New**	**Total**
DSN	85.95	70.00	85.57	62.54	69.74	63.20
PDSN (Our Work)	**85.98**	**70.00**	**85.59**	**74.05**	**70.40**	**73.71**

Tables 5.4 and 5.5 summarize the results of our incremental learning experiment. In standard datasets (FOOD101 and VFN), the improvement with PDSN was marginal, indicating that it learned new classes slightly better than DSN. This improvement can be attributed to the dynamic gamma generation in PDSN, which allows the model to effectively balance and integrate information from the base classifier when learning new classes.

In personalized versions (FOOD101-Personal and VFN-Personal), where our personalizer plug-in enhances adaptability, the performance improvement was notable. Specifically, PDSN achieved approximately a 5% improvement in detecting new classes in FOOD101-Personal and an overall accuracy improvement of about 10% in VFN-Personal compared to DSN.

This experiment underscores the effectiveness of our model's personalized approach, particularly in scenarios with limited data (VFN dataset) and for tasks requiring robust adaptation to new information.

5.3 Ablation Study

To investigate the impact of different meal-related factors (meal frequency, meal time, and meal location) on personalization and performance improvement, we conducted an ablation study using our PDSN model. Fig. 5.1 illustrates the performance of five scenarios over time, each evaluating different configurations of these factors.

The experiments were structured as follows. We first evaluated the base model without considering any meal factors. Subsequently, we separately assessed each factor's impact (meal frequency, meal time, meal location). Finally, we evaluated the model's performance when considering all factors simultaneously.

As depicted in Fig. 5.1, integrating all meal factors into the personalization process yielded the

best overall performance improvement. Specifically:

- considering all meal factors collectively resulted in the highest performance gains, indicating the synergistic effect of comprehensive personalization. Meal frequency had the most significant individual impact on performance improvement, demonstrating its critical role in enhancing the model's adaptability.
- meal time also contributed significantly to performance enhancement, although to a lesser extent compared to meal frequency.
- meal location, while contributing to improvement, had the least pronounced effect among the factors studied.
- the base model, which did not consider any meal factors, exhibited the lowest performance improvement, highlighting the necessity of personalized adaptation in food intake monitoring scenarios.

This ablation study underscores the importance of integrating contextual meal information into the model's learning process, emphasizing the role of meal frequency as a key determinant in enhancing classification accuracy and adaptability.

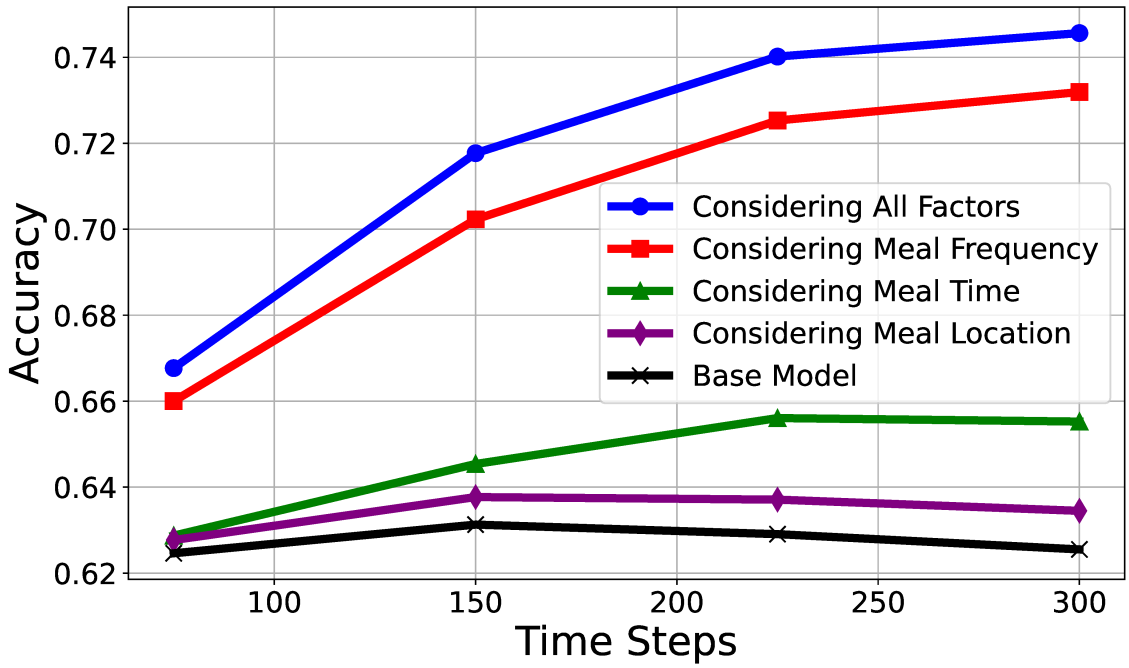


Figure 5.1: Accuracy over Time Steps Considering Different Factors

Chapter 6

Conclusion and Future Work

This chapter summarizes the contributions of this work and outlines potential future developments for the proposed food intake monitoring system.

6.1 Conclusion

In this study, we introduced PDSN, a novel method tailored for personalized food intake monitoring systems. Our method mitigated the adverse impacts of both intra-class variation and inter-class similarities by incorporating user-specific factors related to individual eating behaviors. Moreover, our approach leverages dynamic gamma generation through the network, allowing for adaptive feature weighting between base and incremental classes. These innovations address the limitations of traditional DSN by enhancing classification accuracy and adaptability in the context of evolving dietary habits.

Through comprehensive experimentation, we evaluated PDSN with different baseline architectures on the FOOD101 and VFN datasets, demonstrating consistent performance improvements across personalized datasets (FOOD101-Personal and VFN-Personal). Our results underscored the effectiveness of the personalizer plug-in in enhancing classification accuracy, particularly in scenarios with limited data (VFN), where we observed up to a 10% improvement over baseline models.

Furthermore, our study showcased PDSN's capability in incremental learning, where it outperformed traditional DSN by effectively classifying new food classes introduced after initial training

sessions. This capability was most pronounced in personalized datasets, highlighting the model's robustness in adapting to individual dietary preferences over time.

Additionally, our ablation study highlighted the pivotal role of meal-related factors (meal frequency, meal time, and meal location) in further enhancing PDSN's performance. Integrating these factors significantly improved classification accuracy, with meal frequency proving to be the most influential factor in the personalization process.

In conclusion, PDSN represents a significant advancement in personalized food intake monitoring systems, offering robust performance improvements through dynamic feature adaptation and contextual meal information integration. Our findings not only contribute to the field of machine learning-driven dietary assessment but also pave the way for future research in personalized AI-driven healthcare applications.

6.2 Future Work

In the pursuit of advancing personalized food classification, several avenues for future work present themselves:

Incorporating Additional User-Specific Features

Future research should consider integrating a wider array of user-specific features such as age, gender, and nationality. These factors can significantly influence eating habits and preferences, potentially enhancing the accuracy and relevance of personalized food classification models.

Implementing Reinforcement Learning for Personalization Updates

Future research could explore the use of reinforcement learning to dynamically update the weights of personalization in real-time. By continuously learning from user interactions and feedback, the system could adapt more effectively to changes in user preferences and behavior over time. This approach could result in a more responsive and personalized experience, as the model would refine its predictions and recommendations based on ongoing user engagement.

Removing Hardware Dependencies

To streamline the framework and make it more accessible, future efforts should focus on eliminating the dependency on a smart scale for weight measurement. Instead, weight could be estimated through software approaches, leveraging computer vision and machine learning techniques to infer portion sizes and food weights from images.

Expanding Personalization to Other Domains

The concept of personalized classification can be extended beyond food. Future work could explore its application in other domains such as personalized healthcare, fitness, and e-learning. By tailoring recommendations and content to individual users based on their unique characteristics and behaviors, these systems can provide more effective and engaging experiences.

Exploring Collaborative Personalization Techniques

Integrating collaborative personalization techniques could complement the personalized classification system. By analyzing similarities between users, the system could suggest food items or recipes that have been well-received by users with similar profiles, further enhancing personalization.

By exploring these directions, future research can significantly enhance the personalization and utility of food classification systems, making them more adaptable, accessible, and effective in catering to individual user needs.

Bibliography

[1] Ashutosh Singla, Lin Yuan, and Touradj Ebrahimi. “Food/non-food image classification and food categorization using pre-trained googlenet model”. In: *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*. 2016, pp. 3–11.

[2] Abdulkadir Şengür, Yaman Akbulut, and Ümit Budak. “Food image classification with deep features”. In: *2019 international artificial intelligence and data processing symposium (IDAP)*. IEEE. 2019, pp. 1–6.

[3] Siddeshwar Raghavan, Jiangpeng He, and Fengqing Zhu. “Online Class-Incremental Learning For Real-World Food Image Classification”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 8195–8204.

[4] Doyen Sahoo, Wang Hao, Shu Ke, Wu Xiongwei, Hung Le, Palakorn Achananuparp, Ee-Peng Lim, and Steven CH Hoi. “FoodAI: Food image recognition via deep learning for smart food logging”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2260–2268.

[5] Simon Mezgec and Barbara Koroušić Seljak. “NutriNet: a deep learning food and drink image recognition system for dietary assessment”. In: *Nutrients* 9.7 (2017), p. 657.

[6] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. “Food image recognition using very deep convolutional networks”. In: *Proceedings of the 2nd international workshop on multimedia assisted dietary management*. 2016, pp. 41–49.

[7] Shuqiang Jiang, Weiqing Min, Linhu Liu, and Zhengdong Luo. “Multi-scale multi-view deep feature aggregation for food recognition”. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 265–276.

[8] Jiangpeng He and Fengqing Zhu. “Single-stage heavy-tailed food classification”. In: *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2023, pp. 1115–1119.

[9] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. “Re-evaluating continual learning scenarios: A categorization and case for strong baselines”. In: *arXiv preprint arXiv:1810.12488* (2018).

[10] Li-Jun Zhao, Zhen-Duo Chen, Zi-Chao Zhang, Xin Luo, and Xin-Shun Xu. “Bias Mitigating Few-Shot Class-Incremental Learning”. In: *arXiv preprint arXiv:2402.00481* (2024).

[11] Boyu Yang, Mingbao Lin, Yunxiao Zhang, Binghao Liu, Xiaodan Liang, Rongrong Ji, and Qixiang Ye. “Dynamic support network for few-shot class incremental learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2022), pp. 2945–2951.

[12] Xinyue Pan, Jiangpeng He, and Fengqing Zhu. “Personalized Food Image Classification: Benchmark Datasets and New Baseline”. In: *arXiv preprint arXiv:2309.08744* (2023).

[13] Weiqing Min, Zhiling Wang, Yuxin Liu, Mengjiang Luo, Liping Kang, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. “Large scale visual food recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[14] Mohammed Ahmed Subhi, Sawal Hamid Ali, and Mohammed Abulameer Mohammed. “Vision-based approaches for automatic food recognition and dietary assessment: A survey”. In: *IEEE Access* 7 (2019), pp. 35370–35381.

[15] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20 (1995), pp. 273–297.

[16] Thomas Cover and Peter Hart. “Nearest neighbor pattern classification”. In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[17] Serhat S Bucak, Rong Jin, and Anil K Jain. “Multiple kernel learning for visual object recognition: A review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2013), pp. 1354–1369.

[18] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[19] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60 (2004), pp. 91–110.

[20] Mei Chen, Kapil Dhingra, Wen Wu, Lei Yang, Rahul Sukthankar, and Jie Yang. “PFID: Pittsburgh fast-food image dataset”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 289–292.

[21] Taichi Joutou and Keiji Yanai. “A food image recognition system with multiple kernel learning”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 285–288.

[22] Shulin Yang, Mei Chen, Dean Pomerleau, and Rahul Sukthankar. “Food recognition using statistics of pairwise local features”. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 2249–2256.

[23] Sapna Yadav, Satish Chand, et al. “Automated food image classification using deep learning approach”. In: *2021 7th international conference on advanced computing and communication systems (ICACCS)*. Vol. 1. IEEE. 2021, pp. 542–545.

[24] Fotios S Konstantakopoulos, Eleni I Georga, and Dimitrios I Fotiadis. “A review of image-based food recognition and volume estimation artificial intelligence systems”. In: *IEEE Reviews in Biomedical Engineering* (2023).

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[26] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[29] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[30] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[31] Sefer Memiş, Berker Arslan, Okan Zafer Batur, and Elena Battini Sönmez. “A comparative study of deep learning methods on food classification problem”. In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pp. 1–4.

[32] Jiangpeng He, Zeman Shao, Janine Wright, Deborah Kerr, Carol Boushey, and Fengqing Zhu. “Multi-task image-based dietary assessment for food recognition and portion size estimation”. In: *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE. 2020, pp. 49–54.

[33] Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, and Shervin Shirmohammadi. “Food calorie measurement using deep learning neural network”. In: *2016 IEEE international instrumentation and measurement technology conference proceedings*. IEEE. 2016, pp. 1–6.

[34] Yoshiyuki Kawano and Keiji Yanai. “Food image recognition with deep convolutional features”. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. 2014, pp. 589–593.

[35] Yoshiyuki Kawano and Keiji Yanai. “Foodcam: A real-time food recognition system on a smartphone”. In: *Multimedia Tools and Applications* 74 (2015), pp. 5263–5287.

[36] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, Ma Yunsheng, Songqing Chen, and Peng Hou. “A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure”. In: *IEEE Transactions on Services Computing* 11.2 (2017), pp. 249–261.

[37] Fotios S Konstantakopoulos, Eleni I Georga, and Dimitrios I Fotiadis. “Mediterranean food image recognition using deep convolutional networks”. In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE. 2021, pp. 1740–1743.

[38] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. “CNN-based features for retrieval and classification of food images”. In: *Computer Vision and Image Understanding* 176 (2018), pp. 70–77.

[39] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. “Learning CNN-based features for retrieval of food images”. In: *New Trends in Image Analysis and Processing–ICIAP 2017: ICIAP International Workshops, WBICV, SSPandBE, 3AS, RGBD, NIVAR, IWBAAS, and MADiMa 2017, Catania, Italy, September 11-15, 2017, Revised Selected Papers 19*. Springer. 2017, pp. 426–434.

[40] Lili Pan, Samira Pouyanfar, Hao Chen, Jiaohua Qin, and Shu-Ching Chen. “Deepfood: Automatic multi-class classification of food ingredients using deep learning”. In: *2017 IEEE 3rd international conference on collaboration and internet computing (CIC)*. IEEE. 2017, pp. 181–189.

[41] J. Ross Quinlan. “Induction of decision trees”. In: *Machine learning* 1 (1986), pp. 81–106.

[42] John Platt. “Sequential minimal optimization: A fast algorithm for training support vector machines”. In: (1998).

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[44] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[45] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[46] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. "A convnet for the 2020s". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986.

[47] Xinle Gao, Zhiyong Xiao, and Zhaohong Deng. "High accuracy food image classification via vision transformer with data augmentation and feature augmentation". In: *Journal of Food Engineering* 365 (2024), p. 111833.

[48] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101–mining discriminative components with random forests". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*. Springer. 2014, pp. 446–461.

[49] Jingjing Chen and Chong-Wah Ngo. "Deep-based ingredient recognition for cooking recipe retrieval". In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 32–41.

[50] Zhiyong Xiao, Guang Diao, and Zhaohong Deng. "Fine grained food image recognition based on swin transformer". In: *Journal of Food Engineering* (2024), p. 112134.

[51] Parneet Kaur, Karan Sikka, Weijun Wang, Serge Belongie, and Ajay Divakaran. “Foodx-251: a dataset for fine-grained food classification”. In: *arXiv preprint arXiv:1907.06167* (2019).

[52] Yoshiyuki Kawano and Keiji Yanai. “Automatic expansion of a food image dataset leveraging existing categories with domain adaptation”. In: *Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part III 13*. Springer. 2015, pp. 3–17.

[53] Jun-Hwa Kim, Namho Kim, and Chee Sun Won. “Global–local feature learning for fine-grained food classification based on Swin Transformer”. In: *Engineering Applications of Artificial Intelligence* 133 (2024), p. 108248.

[54] Weiqing Min, Linhu Liu, Zhiling Wang, Zhengdong Luo, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. “Isia food-500: A dataset for large-scale food recognition via stacked global-local attention network”. In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 393–401.

[55] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. “A continual learning survey: Defying forgetting in classification tasks”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3366–3385.

[56] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. “A comprehensive survey of continual learning: Theory, method and application”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[57] Hippolyt Ritter, Aleksandar Botev, and David Barber. “Online structured laplace approximations for overcoming catastrophic forgetting”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[58] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. “Overcoming catastrophic forgetting by incremental moment matching”. In: *Advances in neural information processing systems* 30 (2017).

[59] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. “Progress & compress: A scalable framework for continual learning”. In: *International conference on machine learning*. PMLR. 2018, pp. 4528–4537.

[60] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. “Memory aware synapses: Learning what (not) to forget”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 139–154.

[61] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[62] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. “Riemannian walk for incremental learning: Understanding forgetting and intransigence”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 532–547.

[63] Janghyeon Lee, Hyeong Gwon Hong, Donggyu Joo, and Junmo Kim. “Continual learning with extended kronecker-factored approximate curvature”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9001–9010.

[64] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *International conference on machine learning*. PMLR. 2017, pp. 3987–3995.

[65] Sanyam Kapoor, Theofanis Karaletsos, and Thang D Bui. “Variational auto-regressive Gaussian processes for continual learning”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5290–5300.

[66] Zhizhong Li and Derek Hoiem. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[67] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. “Rotate your networks: Better weight consolidation and less catastrophic

forgetting". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 2262–2268.

[68] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. "Lifelong gan: Continual learning for conditional image generation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2759–2768.

[69] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. "Class-incremental learning via deep model consolidation". In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2020, pp. 1131–1140.

[70] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. "Learning to learn without forgetting by maximizing transfer and minimizing interference". In: *arXiv preprint arXiv:1810.11910* (2018).

[71] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. "Experience replay for continual learning". In: *Advances in neural information processing systems* 32 (2019).

[72] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. "icarl: Incremental classifier and representation learning". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 2001–2010.

[73] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Advances in neural information processing systems* 30 (2017).

[74] David Isele and Akansel Cosgun. "Selective experience replay for lifelong learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[75] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. "Rainbow memory: Continual learning with a memory of diverse samples". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 8218–8227.

[76] Matthias De Lange and Tinne Tuytelaars. “Continual prototype evolution: Learning online from non-stationary data streams”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 8250–8259.

[77] Kandan Ramakrishnan, Rameswar Panda, Quanfu Fan, John Henning, Aude Oliva, and Rogerio Feris. “Relationship matters: Relation guided knowledge transfer for incremental learning of object detectors”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 250–251.

[78] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. “End-to-end incremental learning”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 233–248.

[79] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. “Large scale incremental learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 374–382.

[80] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. “Expert gate: Lifelong learning with a network of experts”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3366–3375.

[81] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. “Pathnet: Evolution channels gradient descent in super neural networks”. In: *arXiv preprint arXiv:1701.08734* (2017).

[82] Arun Mallya and Svetlana Lazebnik. “Packnet: Adding multiple tasks to a single network by iterative pruning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 7765–7773.

[83] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. “Overcoming catastrophic forgetting with hard attention to the task”. In: *International conference on machine learning*. PMLR. 2018, pp. 4548–4557.

[84] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. “Progressive neural networks”. In: *arXiv preprint arXiv:1606.04671* (2016).

[85] Ju Xu and Zhanxing Zhu. “Reinforced continual learning”. In: *Advances in neural information processing systems* 31 (2018).

[86] Jiangpeng He, Luotao Lin, Jack Ma, Heather A Eicher-Miller, and Fengqing Zhu. “Long-tailed continual learning for visual food recognition”. In: *arXiv preprint arXiv:2307.00183* (2023).

[87] Runyu Mao, Jiangpeng He, Zeman Shao, Sri Kalyan Yarlagadda, and Fengqing Zhu. “Visual aware hierarchy based food recognition”. In: *International conference on pattern recognition*. Springer. 2021, pp. 571–598.

[88] Yu Wang, Ye He, Fengqing Zhu, Carol Boushey, and Edward Delp. “The use of temporal information in food image analysis”. In: *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops: ICIAP 2015 International Workshops, BioFor, CTMR, RHEUMA, ISCA, MADiMa, SBMI, and QoEM, Genoa, Italy, September 7-8, 2015, Proceedings 18*. Springer. 2015, pp. 317–325.

[89] Yuto Maruyama, Gamhewage C de Silva, Toshihiko Yamasaki, and Kiyoharu Aizawa. “Personalization of food image analysis”. In: *2010 16th International Conference on Virtual Systems and Multimedia*. IEEE. 2010, pp. 75–78.

[90] Shota Horiguchi, Sosuke Amano, Makoto Ogawa, and Kiyoharu Aizawa. “Personalized classifier for food image recognition”. In: *IEEE Transactions on Multimedia* 20.10 (2018), pp. 2836–2848.

[91] Qing Yu, Masashi Anzawa, Sosuke Amano, Makoto Ogawa, and Kiyoharu Aizawa. “Food image recognition by personalized classifier”. In: *2018 25th IEEE international conference on image processing (ICIP)*. IEEE. 2018, pp. 171–175.

[92] Seum Kim, Yoko Yamakata, and Kiyoharu Aizawa. “Boosting Personalized Food Image Classifier by Sharing Food Records”. In: *Proceedings of the 13th International Workshop on Multimedia for Cooking and Eating Activities*. 2021, pp. 29–32.

[93] U.S. Department of Agriculture. *What We Eat in America (WWEIA) Database*. https://data.nal.usda.gov/dataset/what-we-eat-america-wweia-database. Accessed: 2024-07-12.

[94] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[95] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

Appendix A

Implementation Materials (PDSN)

The appendix contains development codes of personalized dynamic support network

```python
from models.PersonalizedDSN import PersonalizedDSN
from torchvision import transforms
from config import settings
import PIL.Image as Image
import numpy as np
import torch
import statistics
import os
import json
import pandas as pd
import argparse
from utils.Translator import translate
import random

def get_arguments():
    parser = argparse.ArgumentParser(description='PDSN')
    parser.add_argument("--af", type=float, default=0.0003)
    parser.add_argument("--at", type=float, default=0.004)
    parser.add_argument("--al", type=float, default=0.004)
    parser.add_argument("--dataset", type=str, default='FOOD101')
    parser.add_argument("--backbone", type=str, default='resnet18')
    parser.add_argument("--incremental", type=bool, default=False)
    return parser.parse_args()


args = get_arguments()

```

Figure A.1: The code in pdsn.py file

```
# Load Model
network = PersonalizedDSN(args=args)
# Move To Cuda
network.cuda()
# Load Checkpoint
savedir = os.path.join("checkpoints", args.backbone, args.dataset+"_inc_old","Sess2.pth.tar")
checkpoint = torch.load(savedir)
network.load_state_dict(checkpoint['state_dict'], strict=False)
# Put network to evaluation mode
network.eval()

# Load Class Mapping
class_list = []
if(args.dataset == "FOOD101"):
    with open(settings.FOOD101_DIR + '/meta/classes.txt', 'r') as f:
        lines = f.readlines()
        for line in lines:
            class_list.append(line.strip())
    # Load Mean and Std for Normalization
    mean_vals = settings.mean_vals_food
    std_vals = settings.std_vals_food
    # Set transformations
    transforms = transforms.Compose([#transforms.ToPILImage(),
                                     transforms.Resize(settings.FOOD101_size),
                                     transforms.ToTensor(),
                                     transforms.Normalize(mean_vals, std_vals)
                                     ])
    # number of users
    num_users = 20
    # categories size
    categories = 103
    # base
    base_categories = 101
```

Figure A.2: The code in pdsn.py file

```
if(args.dataset == "VFN"):
    with open(settings.VFN_DIR + '/Meta/category_ids.txt', 'r') as f:
        lines = f.readlines()
        for line in lines:
            class_list.append(line.strip().split(" ")[1])
    # Load Mean and Std for Normalization
    mean_vals = settings.mean_vals_vfn
    std_vals = settings.std_vals_vfn
    # Set transformations
    transforms = transforms.Compose([#transforms.ToPILImage(),
                                     transforms.Resize(settings.VFN_size),
                                     transforms.ToTensor(),
                                     transforms.Normalize(mean_vals, std_vals)
                                     ])
    # number of users
    num_users = 26
    # categories size
    categories = 84
    # base
    base_categories = 82

# incremental classes
class_list += ["gheimeh", "ghormeh"]
```

Figure A.3: The code in pdsn.py file

```python
# Load the meal time and location probabilities
mealtime_data = pd.read_csv(f'./dataset/{args.dataset}_personal/{args.dataset}_MealTime_Probability.csv')
meallocation_data = pd.read_csv(f'./dataset/{args.dataset}_personal/{args.dataset}_MealLocation_Probability.csv')
# Meal times 0,1,2,3 and locations 0,1
meal_times = ['Morning', 'Afternoon', 'Evening', 'Night']
meal_locations = ['Home','Out']
accuracy_list_normal = {}
accuracy_list_personalized = {}
accuracy_list_normal_base = {}
accuracy_list_personalized_base = {}
accuracy_list_normal_inc = {}
accuracy_list_personalized_inc = {}
for user_id in range(1,num_users+1):
    # Load Eating Pattern
    pattern_image_directories = f'dataset/{args.dataset}_personal/benchmark_pattern_{args.dataset.lower()}/{user_id}/pattern_image'
    with open(pattern_image_directories, 'r') as file:
        pattern_image_directories = json.load(file)
    pattern_labels = f'dataset/{args.dataset}_personal/benchmark_pattern_{args.dataset.lower()}/{user_id}/pattern_semantic'
    with open(pattern_labels, 'r') as file:
        pattern_labels = json.load(file)
    for i in range(6,21):
        food1_dir = f'./incremental/sess1/{i}.jpg'
        food2_dir = f'./incremental/sess2/{i}.jpg'
        label1 = "gheimeh"
        label2 = "ghormeh"
        random_index1 = random.randint(0, len(pattern_image_directories))
        pattern_image_directories.insert(random_index1, food1_dir)
        pattern_labels.insert(random_index1, label1)
        random_index2 = random.randint(0, len(pattern_image_directories))
        pattern_image_directories.insert(random_index2, food2_dir)
        pattern_labels.insert(random_index2, label2)
```

Figure A.4: The code in pdsn.py file

```python
    # Number of Correct Predictions and total numbers
    correct = 0
    correct_personalized = 0
    correct_base = 0
    correct_inc = 0
    correct_p_base = 0
    corret_p_inc = 0
    total = 0
    total_base = 0
    total_inc = 0
    dataset_size = len(pattern_image_directories)
    # initialize MF vector
    MF = torch.full((1, categories), 1/categories)
    MF = MF.cuda()
    # initialize MT Matrix
    MT = torch.full((categories,len(meal_times)),1/len(meal_times))
    MT = MT.cuda()
    # initialize ML Matrix
    ML = torch.full((categories,len(meal_locations)),1/len(meal_locations))
    ML = ML.cuda()
    # forget factor
    alpha_f = args.af
    alpha_t = args.at
    alpha_l = args.al
    # Load Data and Inference
```

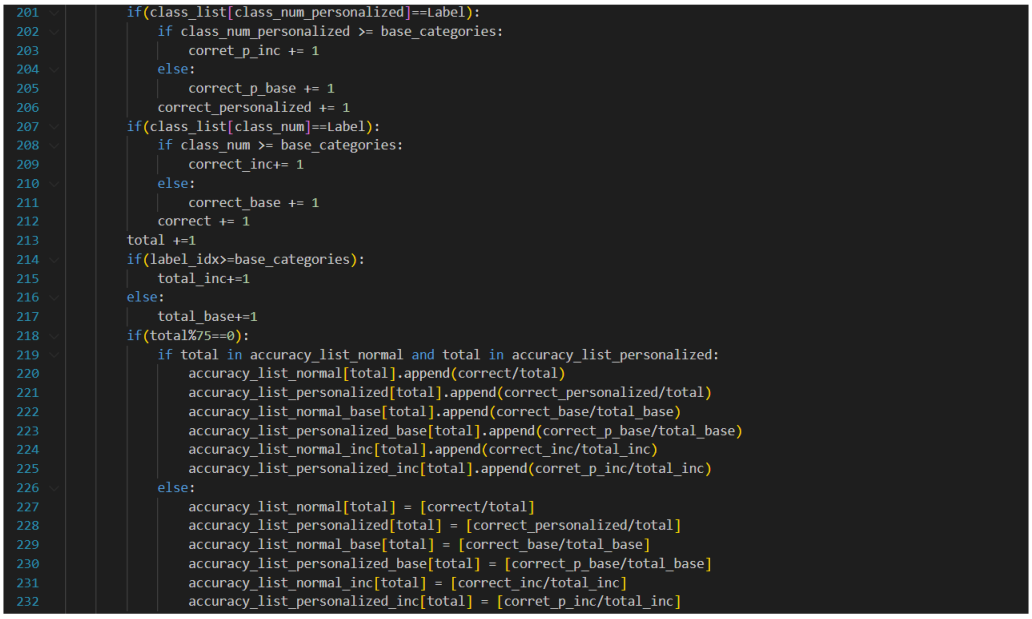
Figure A.5: The code in pdsn.py file

```python
    # Load Data and Inference
    for i in range(dataset_size):
        directory = f"./dataset/{args.dataset}_personal/" + pattern_image_directories[i][2:]
        img = Image.open(directory)
        # Do transforms
        image = transforms(img)
        # To Cuda
        image = image.cuda()
        Label = pattern_labels[i]
        if(args.dataset == "VFN"):
            Label = Label.replace(' ','_')
            Label = Label.lower()
            Label = translate(Label)
        label_idx = class_list.index(Label)
        # time
        mealtime_probabilities = mealtime_data[mealtime_data['Food'] == Label].iloc[0][['Morning', 'Afternoon', 'Evening', 'Night']]
        mealtime_probabilities = mealtime_probabilities.tolist()
        mealtime_probabilities = [float(prob) / sum(mealtime_probabilities) for prob in mealtime_probabilities]
        # Perform weighted random selection
        chosen_time = np.random.choice(meal_times, p=mealtime_probabilities)
        time_idx = meal_times.index(chosen_time)

        # location
        meallocation_probabilities = meallocation_data[meallocation_data['Food'] == Label].iloc[0][['Home', 'Out']]
        meallocation_probabilities = meallocation_probabilities.tolist()
        meallocation_probabilities = [float(prob) / sum(meallocation_probabilities) for prob in meallocation_probabilities]
        # Perform weighted random selection
        chosen_location = np.random.choice(meal_locations, p=meallocation_probabilities)
        location_idx = meal_locations.index(chosen_location)

        # Get output
        out, output = network(image.unsqueeze(0), sess = 2)
```

Figure A.6: The code in pdsn.py file

```python
        # MF impact
        output_p = output * MF
        # MT impact
        MTi = MT[:,time_idx].unsqueeze(0)
        output_p = output_p * MTi
        # ML impact
        MLi = ML[:,location_idx].unsqueeze(0)
        output_p = output_p * MLi
        # Personalized Out
        _, pred_personalized = torch.max(output_p, dim=1)
        # Normal Out
        _, pred = torch.max(output, dim=1)
        class_num = pred.item()
        class_num_personalized = pred_personalized.item()
        true_idx = class_list.index(Label)
        # update MF
        MF[0, true_idx] = MF[0, true_idx] + alpha_f * (1 - MF[0, true_idx])
        mask = torch.ones_like(MF, dtype=torch.bool)
        mask[0, true_idx] = False
        MF[mask] = MF[mask] * (1 - alpha_f)
        # update MT
        MT[true_idx,time_idx] = MT[true_idx,time_idx] + alpha_t*(1-MT[true_idx,time_idx])
        MT[true_idx, :time_idx] = MT[true_idx, :time_idx] * (1 - alpha_t)
        MT[true_idx, time_idx+1:] = MT[true_idx, time_idx+1:] * (1 - alpha_t)
        # update ML
        ML[true_idx,location_idx] = ML[true_idx,location_idx] + alpha_l*(1-ML[true_idx,location_idx])
        ML[true_idx, :location_idx] = ML[true_idx, :location_idx] * (1 - alpha_l)
        ML[true_idx, location_idx+1:] = ML[true_idx, location_idx+1:] * (1 - alpha_l)
```

Figure A.7: The code in pdsn.py file



```
            if(class_list[class_num_personalized]==Label):
                if class_num_personalized >= base_categories:
                    corret_p_inc += 1
                else:
                    correct_p_base += 1
                correct_personalized += 1
            if(class_list[class_num]==Label):
                if class_num >= base_categories:
                    correct_inc+= 1
                else:
                    correct_base += 1
                correct += 1
            total +=1
            if(label_idx>=base_categories):
                total_inc+=1
            else:
                total_base+=1
            if(total%75==0):
                if total in accuracy_list_normal and total in accuracy_list_personalized:
                    accuracy_list_normal[total].append(correct/total)
                    accuracy_list_personalized[total].append(correct_personalized/total)
                    accuracy_list_normal_base[total].append(correct_base/total_base)
                    accuracy_list_personalized_base[total].append(correct_p_base/total_base)
                    accuracy_list_normal_inc[total].append(correct_inc/total_inc)
                    accuracy_list_personalized_inc[total].append(corret_p_inc/total_inc)
                else:
                    accuracy_list_normal[total] = [correct/total]
                    accuracy_list_personalized[total] = [correct_personalized/total]
                    accuracy_list_normal_base[total] = [correct_base/total_base]
                    accuracy_list_personalized_base[total] = [correct_p_base/total_base]
                    accuracy_list_normal_inc[total] = [correct_inc/total_inc]
                    accuracy_list_personalized_inc[total] = [corret_p_inc/total_inc]
```

Figure A.8: The code in pdsn.py file

```
with open(f"results_{args.dataset}_{args.backbone}_inc_ours.txt", "w") as file:
    for key in accuracy_list_normal.keys():
        print(f"TimeStamp {key}, original model:")
        average_normal = statistics.mean(accuracy_list_normal[key])
        std_dev_normal = statistics.stdev(accuracy_list_normal[key])
        average_normal_base = statistics.mean(accuracy_list_normal_base[key])
        average_normal_inc = statistics.mean(accuracy_list_normal_inc[key])
        print(f"avg: {average_normal}, std: {std_dev_normal}, avg base: {average_normal_base}, avg inc: {average_normal_inc}")
        file.write(f"TimeStamp {key}, original model:\n")
        file.write(f"Average: {average_normal}\n")
        file.write(f"Average base: {average_normal_base}\n")
        file.write(f"Average inc: {average_normal_inc}\n")
        file.write(f"Standard Deviation: {std_dev_normal}\n\n")
        print(f"TimeStamp {key}, personalized model:")
        average_person = statistics.mean(accuracy_list_personalized[key])
        std_dev_person = statistics.stdev(accuracy_list_personalized[key])
        average_person_base = statistics.mean(accuracy_list_personalized_base[key])
        average_person_inc = statistics.mean(accuracy_list_personalized_inc[key])
        print(f"avg: {average_person}, std: {std_dev_person}, avg base: {average_person_base}, avg inc: {average_person_inc}")
        file.write(f"TimeStamp {key}, personalized model:\n")
        file.write(f"Average: {average_person}\n")
        file.write(f"Average base: {average_person_base}\n")
        file.write(f"Average inc: {average_person_inc}\n")
        file.write(f"Standard Deviation: {std_dev_person}\n\n")
```

Figure A.9: The code in pdsn.py file

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from models.backbone import Resnet
from config import settings
from torchvision import models

class PersonalizedDSN(nn.Module):
    def __init__(self, args):
        super(PersonalizedDSN, self).__init__()
        if args != None:
            #ResNet
            if(args.backbone == "ResNet"):
                self.backbone = Resnet.resnet18(pretrained=True)
            #AlexNet
            if(args.backbone == "AlexNet"):
                self.backbone = models.alexnet(models.AlexNet_Weights)
            # DenseNet
            if(args.backbone == "DenseNet"):
                self.backbone = models.densenet121(models.DenseNet121_Weights)
            # GoogLeNet
            if(args.backbone == "GoogleNet"):
                self.backbone = models.googlenet(models.GoogLeNet_Weights)
            #ViT
            if(args.backbone == "ViT"):
                self.backbone = models.vision_transformer.vit_b_16(models.ViT_B_16_Weights)
            # Swin Transformer
            if(args.backbone == "Swin"):
                self.backbone = models.swin_transformer.swin_v2_t(models.Swin_V2_T_Weights)
            # VGGNet
            if(args.backbone == "VGGNet"):
                self.backbone = models.vgg11(models.VGG11_Weights)

        self.node = 512
```

Figure A.10: The code in model.py file

```python
        if(args.dataset == 'FOOD101'):
            self.session_len = settings.FOOD101_SessLen
        if(args.dataset == 'VFN'):
            self.session_len = settings.VFN_SessLen

        # AlexNet
        if(args.backbone == "AlexNet"):
            num_features = self.backbone.classifier[6].in_features
            self.backbone.classifier[6] = nn.Linear(num_features, self.node, bias=False)
            self.featureMapper = nn.Linear(512, self.node, bias=False)
        # DenseNet
        if(args.backbone == "DenseNet"):
            self.backbone.classifier = torch.nn.Identity()
            self.featureMapper = nn.Linear(1024, self.node, bias=False)
        # GoogleNet
        if(args.backbone == "GoogleNet"):
            num_features = self.backbone.fc.in_features
            self.backbone.fc = nn.Linear(num_features, 512)
            self.featureMapper = nn.Linear(512, self.node, bias=False)
        # ViT
        if(args.backbone == "ViT"):
            self.backbone.heads = torch.nn.Identity()
            self.featureMapper = nn.Linear(768, self.node, bias=False)
        # ResNet
        if(args.backbone == "ResNet"):
            self.featureMapper = nn.Linear(512, self.node, bias=False)
        # Swin Transformer
        if(args.backbone == "Swin"):
            self.backbone.head = torch.nn.Identity()
            self.featureMapper = nn.Linear(768, self.node, bias=False)
        # VGGNet
        if(args.backbone == "VGGNet"):
            self.backbone.classifier[6] = torch.nn.Identity()
            self.featureMapper = nn.Linear(4096, self.node, bias=False)
        #-------
```

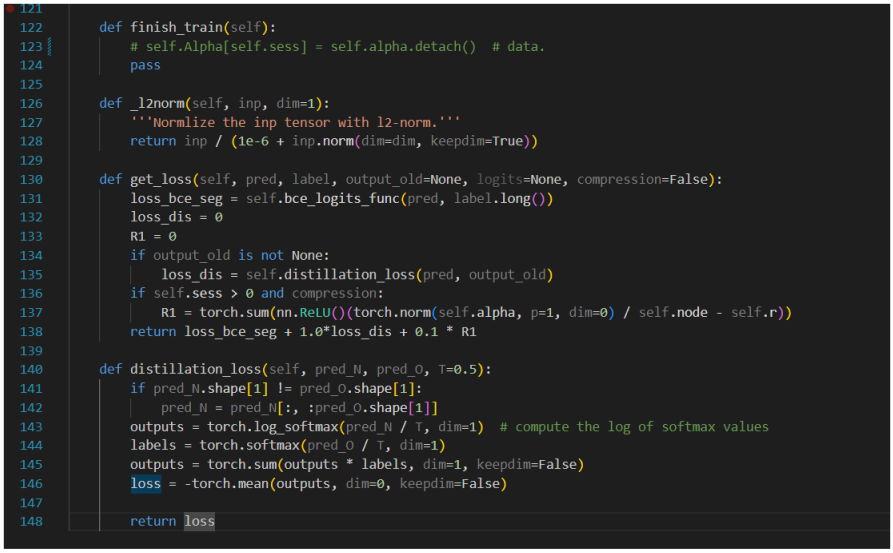
Figure A.11: The code in model.py file

```
        #-------
        self.baseClassifier = nn.Linear(self.node, self.session_len[0], bias=False)
        SessLen = len(self.session_len) # total number of sessions for the experiment
        for i in range(1, SessLen):
            exec('self.session' + str(i) + "Classifier" + '= nn.Linear(self.node, self.session_len[i], bias=False)')
        for i in range(1, SessLen):
            exec('self.session' + str(i) + "Supporter" + '= nn.Linear(512, self.node, bias=False)')

        self.gammaGenerator = nn.Linear(512, SessLen-1, bias=False)
        Alpha = torch.zeros(SessLen, self.node)
        Alpha[0] = Alpha[0] + 1
        self.register_buffer('Alpha', Alpha)
        self.r = nn.Parameter(torch.tensor(0.0))
        self.bce_logits_func = nn.CrossEntropyLoss()
        self.sess = 0

    def get_feature(self, x):
        x = self.backbone(x)
        x = x.view(x.size(0), -1)

        return x
```

Figure A.12: The code in model.py file

```
def forward(self, x, sess=0, epoch=0, Mode='train', IOF='image'):
    self.sess = sess
    if IOF == 'image':
        if sess > 0:
            with torch.no_grad():
                x = self.get_feature(x)
        else:
            x = self.get_feature(x)

    out = self.featureMapper(x)
    out = self._l2norm(out, dim=1)
    gammas = self.gammaGenerator(x)
    gammas = F.relu(gammas)
    for i in range(sess + 1): # 0,1,2
        if i == 0: # base session
            output = F.linear(F.normalize(out, p=2, dim=-1), F.normalize(self.baseClassifier.weight, p=2, dim=-1))
        else:
            gamma_i = gammas[:, i-1].unsqueeze(1)
            classifier = eval('self.session' + str(i) + "Classifier")
            supporter = eval('self.session' + str(i) + "Supporter")
            # normalize
            out_aux = F.linear(F.normalize(x.view(x.size(0), -1), p=2, dim=-1), F.normalize(supporter.weight, p=2, dim=-1))
            new_node = (out * gamma_i) + (out_aux)
            # new_node = (out * 0.2) + (out_aux)
            new_node = self._l2norm(new_node, dim=1)
            output = torch.cat([output, F.linear(F.normalize(new_node, p=2, dim=-1), F.normalize(classifier.weight, p=2, dim=-1))],
                               dim=1)  # concat classifiers

    # output = F.softmax(output, dim=1)
    return out, output
```

Figure A.13: The code in model.py file



```
    def finish_train(self):
        # self.Alpha[self.sess] = self.alpha.detach()  # data.
        pass

    def _l2norm(self, inp, dim=1):
        '''Normlize the inp tensor with l2-norm.'''
        return inp / (1e-6 + inp.norm(dim=dim, keepdim=True))

    def get_loss(self, pred, label, output_old=None, logits=None, compression=False):
        loss_bce_seg = self.bce_logits_func(pred, label.long())
        loss_dis = 0
        R1 = 0
        if output_old is not None:
            loss_dis = self.distillation_loss(pred, output_old)
        if self.sess > 0 and compression:
            R1 = torch.sum(nn.ReLU()(torch.norm(self.alpha, p=1, dim=0) / self.node - self.r))
        return loss_bce_seg + 1.0*loss_dis + 0.1 * R1

    def distillation_loss(self, pred_N, pred_O, T=0.5):
        if pred_N.shape[1] != pred_O.shape[1]:
            pred_N = pred_N[:, :pred_O.shape[1]]
        outputs = torch.log_softmax(pred_N / T, dim=1)  # compute the log of softmax values
        labels = torch.softmax(pred_O / T, dim=1)
        outputs = torch.sum(outputs * labels, dim=1, keepdim=False)
        loss = -torch.mean(outputs, dim=0, keepdim=False)

        return loss
```

Figure A.14: The code in model.py file