Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Deep learning in food category recognition

Yudong Zhang [a,b], Lijia Deng [a], Hengde Zhu [a], Wei Wang [a], Zeyu Ren [a], Qinghua Zhou [a], Siyuan Lu [a], Shiting Sun [a], Ziquan Zhu [a], Juan Manuel Gorriz [c,d,*], Shuihua Wang [a,*]

[a] School of Computing and Mathematical Sciences, University of Leicester, Leicester, LE1 7RH, Leicestershire, UK
[b] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[c] Department of Signal Theory, Networking and Communications, University of Granada, Granada, Spain
[d] Department of Psychiatry, University of Cambridge, Cambridge CB21TN, UK

A R T I C L E I N F O

Keywords:
Machine learning
Deep learning
Data augmentation
Food category recognition
Computer vision
Semi-supervised learning
Convolutional neural network
Transfer learning

A B S T R A C T

Integrating artificial intelligence with food category recognition has been a field of interest for research for the past few decades. It is potentially one of the next steps in revolutionizing human interaction with food. The modern advent of big data and the development of data-oriented fields like deep learning have provided advancements in food category recognition. With increasing computational power and ever-larger food datasets, the approach's potential has yet to be realized. This survey provides an overview of methods that can be applied to various food category recognition tasks, including detecting type, ingredients, quality, and quantity. We survey the core components for constructing a machine learning system for food category recognition, including datasets, data augmentation, hand-crafted feature extraction, and machine learning algorithms. We place a particular focus on the field of deep learning, including the utilization of convolutional neural networks, transfer learning, and semi-supervised learning. We provide an overview of relevant studies to promote further developments in food category recognition for research and industrial applications.

1. Introduction

Throughout history, the multi-facet relationship between humankind and the food we consume has changed countless times. How food is stored, transported, processed, and consumed today vastly differs from that of the 20th century [1,2] and is unimaginable for hunter-gatherers of the Paleolithic period [3]. Similarly, how we interact with food in the next century may be beyond our imagination. However, from sickles and plows to automated harvesters; from drying and salting to refrigerated containers; the evolution of our interaction with food correlates with improvements in efficiency, where higher efficiency results in higher yield and availability for consumption. In our pursuit of ever-increasing efficiency, the next step may be the inclusion of artificial intelligence (AI) and "centaurs", i.e., AI-aided humans [4] between humans and food.

Food category recognition covers a range of tasks in the food "life cycle" of cultivation, processing, storage, delivery, and consumption. These tasks include (1) food quality and ingredients detection and (2) food type and quantity detection. Quality and ingredient detection are most relevant to food production, where identifying food quality and contents allow for filtering low-quality and substandard products [5–7]. Type and quantity detection are commonly applied in the delivery of food and in the process of individual consumption, i.e., consuming the right food in a measured quantity to prevent allergens and control

Abbreviations: ACC, accuracy; ANN, artificial Neural Network; BN, batch normalization; BPNN, back propagation neural network; CJ, color jittering; CNN, convolutional neural network; CVS, computer vision system; DA, data augmentation; DF, displacement field; DoG, difference-of-Gaussian; DSIA, dual-stage image analysis; EL, ensemble learning; ELM, extreme learning machine; FCM, fuzzy C-means; FT, Fourier transform; GAN, generative adversarial network; GConv, group convolution; GT, geometric transformation; HOG, histogram of oriented gradients; k-NN, k-nearest neighbors; LDA, linear discriminant analysis; LR, logistic regression; Macro-F1, macro-averaged F1 score; Micro-F1, micro-averaged F1 score; MKL, multi-kernel learning; MSI, multispectral image; NN, neural network; PDF, probability density function; PN, Poisson noise; POC, pair of coordinates; PST, PatchShuffle transform; RBF, radial basis function; ReLU, rectified Linear Unit; ResNet-152, residual neural network with 152 layers; RF, random forest; RVFL, random vector functional link; SIFT, scale-invariant feature transform; SMOTE, synthetic minority over-sampling technique; SPN, salt-and-pepper noise; SSD, small-size dataset; SSL, semi-supervised learning; STFT, short-time Fourier transform; SVM, support vector machine; UV, ultraviolet; VAEs, variational auto-encoders; ViT, vision Transformer.

* Corresponding authors.
E-mail addresses: gorriz@ugr.es (J.M. Gorriz), shuihuawang@ieee.org (S. Wang).

https://doi.org/10.1016/j.inffus.2023.101859
Received 26 March 2023; Received in revised form 25 May 2023; Accepted 26 May 2023
Available online 27 May 2023

nutritional intake [8,9].

From the food production perspective, which includes aspects of cultivation, processing, storage, and transport, technological advances paired with the socio-economic phenomenon of industrialization and globalization have made food production incredibly efficient. The reality supports this claim since human food production can sustain most of our 80 billion population. However, efficiency differs in different countries and regions due to varying socio-economic scenarios, and even in highly mechanized and automated processes, many roles are both subjective and labor intensive. Both characteristics impact the overall efficiency of the food production process.

A potential future solution to this problem is using data-oriented artificial intelligence, where integration with robotics can reduce labor costs and well-adjusted algorithms are less prone to intra-personnel and inter-personnel bias. The requirement for manual interactions in the food ”life-cycle” is not limited to the production processes but is also predominant in any individual's interaction with food. The prime scenario is diet control, as introduced in the prior survey by Zhu [10]. An imbalance in the human diet can cause obesity and is linked to a number of chronic diseases, including hypertension, diabetes, heart diseases, and multiple cancers [11]. The significant improvements in living standards from the 20th to 21st century, especially in first-world countries, have led to a significant climb in the percentage of the population that is overweight or obese [12]. An intuitive solution to a balanced diet is tracking and quantifying nutritional intake, often through mobile applications [9]. However, the current applications are inefficient and often require significant manual input.

Data-oriented artificial intelligence, especially deep learning, provides a potential solution to this problem through high-performing computer vision models, e.g., faster and more accurate food logging into diet monitoring applications, food safety, and quality detection. Data-orient artificial intelligence has gained significant attention after the plateau of innovations in general AI and expert systems. A characteristic of this more specified and targeted approach to AI is the significantly overparameterized models that rely on large, annotated datasets. This approach to AI has shown significant advancements in speech recognition [13], natural language processing [14], and computer vision [15].

However, the gap between research and practical application in many interdisciplinary fields, e.g., medicine, is the availability of quality data and annotations. The lack of data poses the problem of learning in the high-dimensional low sample size, which leads to overfitting and lack of generalization. Over-fitting occurs when a model provides decision boundaries too close to the available training data, causing an empirical performance gap between train and test results. The lack of data can also potentially cause an implicit difference between the distribution of the small test set and real-world data, resulting in high bias error and low generalizability to real-world applications.

Increasing food consumption is paired with the significant amount of data we produce in the food domain. Major contributing factors include our interactions with the internet and social media [16], e.g., food snapshots and health applications. At the same time, mass industrial food production also provides opportunities for data collection by the producers [17]. Most data available will be without annotation, but annotation costs are generally lower than in fields like medicine, e.g., everyday tasks like labeling food types and cuisines can be done with little or no expert knowledge [18]. If the data is utilized ethically, we can likely find sufficient data for training data-oriented AI. In this survey, we include an extensive survey of datasets available for food category recognition, focusing on tasks of recognizing processed foods, vegetables, and fruits. Grouped based on the food type and cuisines, we included 51 datasets that were published from 2009 to 2022. In datasets produced in recent years, we observe an evident increase in sample size and the number of categories, along with more attention to food nutrition and geographical information.

Multiple challenges exist in learning from food-based data, especially food images. These challenges include the lack of spatial uniformity that enables clear identification of differences [9], e.g., since food varies greatly in shape, color, and form, no geometric structure or pattern exists to identify differences quickly. Many types of food also lack rigid structures and can be displayed in a range of different states [19], causing significant variances between data sources and within the same category. Significant challenges exist in data collection, where access to the abundance of data on the internet provides no guarantee of consistency in data view and quality.

A potential solution to this problem is using preprocessing to provide more data and more variations of data to minimize the impact of the above challenges on our data-oriented algorithms and models. These processing methods can be generalized as data augmentation [20]. Under the assumption that the test sets provide a close estimate of the real-world distributions, data augmentation aims to reduce the distance between train and test sets through various image transformations. These transformations include geometric transforms, noise injection, photo-metric transforms, image mixing, and deep learning-based methods. This survey provides an in-depth survey of these transformations with examples of common food images.

After data collection and augmentation, the natural problem is extracting information provided by the dataset and its relevant transformations. Modern machine learning research has two predominant approaches: classical hand-crafted features and deep feature extraction [21]. Hand-crafted features are features constructed with a priori knowledge of the underlying task. In this survey, the target task of a model or algorithm is most often the classification of food images. Relevant features to this target class often include color, texture, and shape. In this survey, we cover standard feature extraction methods, e.g., color histograms for color features, histograms of oriented gradients for shape, and Gabor filters for texture features. We also include scale-dependent wavelet transforms; and scale-invariant feature transforms for invariance to resolution, illumination, and orientation. These methods cover the most common forms of hand-crafted features for food image data and can provide the basis for classification with either machine learning algorithms or neural networks.

Traditional machine learning algorithms [22] is the standard method for food category recognition before the 2010s and the rise of deep learning and large image datasets. Common methods we surveyed include support vector machine, logistic regression, K-nearest neighbors, K-means, tree-based methods, etc. The algorithms are grounded in statistics and probability theory and provide good performance for small datasets with hand-crafted features. However, the dependence of hand-crafted features on a priori knowledge limits its application to experts with domain knowledge in the food category recognition task. This scenario also increases the specificity of a feature extraction and classification pipeline, limiting its limits to the application and transfer to other domains. For example, a pipeline for the classification of apples may differ greatly from a pipeline for the grain quality measurement due to the difference between the task-specific features that may be included. A potential alternative to this approach is the deep learning approach, which uses neural networks for automated feature extraction.

In recent years, deep learning has risen to become a large field in the machine learning domain. It implements universal approximators of neural networks [23], a modern development of the perceptron proposed by Rosenblatt [24]. With chain-rule-derived gradient computation and back-propagation [25], the neural networks can be efficiently trained with a data-oriented approach. The sheer number of neurons coupled with vision-oriented architectures like convolutional layers enables neural networks to abstract latent feature representations with minimal manual interference. An example of recent works by Aguilar [26] has applied this method in food category recognition. Convolutional neural networks (CNN), like the model proposed by Teng [27], are the most common architecture in the field. The features extracted by CNN often show a hierarchy of low to high-level features that extend from lines, dots, or edges to objects or characteristic shapes [28]. CNN usually consists of standard components of the convolution layer, padding, pooling, and various activation functions and regularization methods. This survey includes detailed descriptions of these components and their application. In extension, we survey the most commonly applied deep learning method in the field of food category recognition, transfer learning, and a field of study with great potential, semi-supervised learning.

With the popularity of deep neural networks in interdisciplinary research, common challenges exist in practical applications [29]. These challenges include the availability of data and relevant labels. Transfer learning is the field of study for the transfer of knowledge across domains [30]. Current computer vision success is based on the ImageNet hierarchical database, which contains millions of annotated images [31]. While large quantities of food images are potentially available, the dataset for specific tasks, e.g., classification of a particular food category, may be limited. A potential solution to this problem is the transfer of knowledge across domains - transfer learning [30].

In image classification applications, transfer learning is commonly implemented through the transfer of model structure, weights, or parameters for classification in different feature spaces and distributions. Neural networks with transferred parameters have outperformed the same neural networks with randomized parameters in convergence, thus reducing the need for time-consuming hyperparameter searches [32]. We include an overview of the application of transfer learning to food category recognition and common pre-trained neural networks utilized in these processes.

The abundance of potential data for food category recognition presents both great opportunity and challenge. One potential approach to utilizing these potential data is through semi-supervised learning, which is an approach that lies between unsupervised and supervised learning. Based on the basic assumptions of the data and its manifold [33], semi-supervised learning aims to learn from both labeled and unlabeled data. This approach is especially applicable to many fields of food category recognition, where unlabeled data are potentially abundant. This survey uses the most commonly applied semi-supervised methods, including generative and diffusion modeling, consistency regularization, pseudo-labeling, and graph-based methods.

Prior surveys focused on food processing [10], food recognition for mobile applications, and volume estimation [9]. In this survey, we provide a more generalized view of food category recognition that includes applications throughout the food "life-cycle" and encompasses a range of recognition tasks, including food quality identification and food ingredients detection. This survey identifies the key impacts of this field of study, including the macro impacts of automation of the food industry with more efficient human labor and reduction of human error; and the micro impacts of food personification for better dietary health of individuals.

The survey is structured as follows: Section 2 will introduce the datasets, Section 3 will present existing food-oriented machine vision systems, Section 4 will present preprocessing through the lens of data augmentations, Section 5 will present a survey of hand-crafted features, which the traditional machine learning approaches will follow in Section 6. We will move toward deep learning from the traditional machine learning approach: Section 7 will present the essential CNN components,

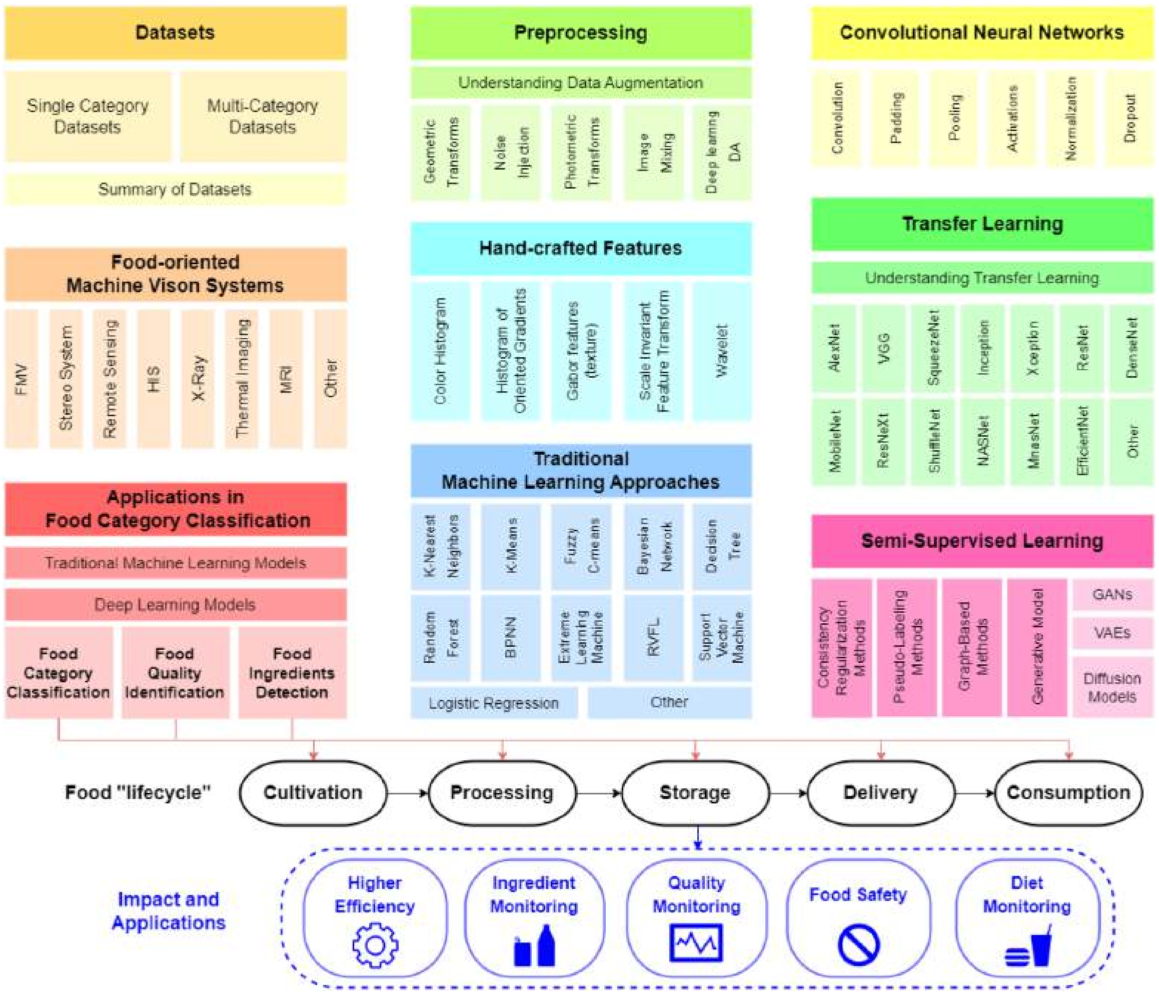


Fig. 1. Taxonomy of this survey.

Section 8 will introduce the common practice of transfer learning, and Section 9 will present semi-supervised learning as a potential pathway toward the future. Section 10 surveys the application of deep learning methods in practice, and we conclude the survey in Section 11. A taxonomy of the survey can be found in Fig. 1.

This survey is highly related to the journal Information Fusion. Numerous reasons for this case include: many food imaging vision systems can be fused, as shown in Section 3. Various data augmentation methods in Section 4 are commonly fused to create a more powerful hybrid fusion technique. Hand-crafted features in Section 5 can be fused to generate more efficient features. Traditional machine learning techniques in Section 6 can be fused to create more effective classifiers. Convolutional neural networks in Section 7 can be fused to improve classification by combining the strengths of multiple models. Pretrained models in Section 8 can be fused by combining the knowledge learned from different tasks. Fusion is inherent in semi-supervised learning, in Section 9, where we combine the information from labeled and unlabeled data to improve the performance of food category recognition models.

2. Datasets

Data has always been one of the key points to driving AI-based Food Recognition research. Different training data may lead to different training results. No matter the target, food image datasets are needed to evaluate the performance of the proposed algorithms. To this end, the above research work either uses existing datasets or introduces new datasets. Therefore, we hope to survey the existing datasets. This can help us select the appropriate dataset to improve the model's performance when doing research.

A clear and targeted single-category dataset may have better training effects for specific tasks. For example, a single-category of fruit dataset may achieve better training results for classifying fresh fruits. For example, Dhiman et al. (2021) [34] used a special citrus dataset to train a model for citrus quality classification. Chen, X. et al. (2017) [18], Chen et al. (2012) [35], and Chen et al. (2016) [36] successfully classified Chinese food, the most diverse and complex dishes, using the special Chinese food dataset. Tatsuya Miyazaki et al. (2011) [37] and Matsuda et al. (2012) [38] also contributed complete datasets to the identification of Japanese food.

On the other hand, when training in-depth learning models, the use of diverse and multi-categories datasets can better support widespread use and also support the migration performance of models. Using multi-categories data can improve the validity and accuracy of the data and make the model more generalizable to other datasets. In addition, a variety of datasets can better detect how a model behaves in exceptional situations. In this way, the neural network model can work more widely, accurately, and reliably to achieve the desired results. For example, Min et al. (2019) [39] and Min et al. (2020) [40] have contributed several datasets in this direction in order to continuously improve model performance.

In addition, Zhou et al. (2016) [41] have made full use of the correlation between different data, such as using geographic information, restaurant classification, and other non-food third-party information to assist in food classification. Or use ingredients as additional supervision information to improve the accuracy of food classification and even predict the nutrient content of food [42,43].

In general, at present, we can divide the existing datasets into single-category datasets and multi-categories datasets according to their data characteristics. Single-category datasets usually only contain foods from one cuisine, and these foods usually have certain similarities in the form or use of ingredients. This single-category dataset can maximize the recognition accuracy of a proprietary model in a specific area. However, a single-category of the dataset may have some limitations, such as regional and cultural differences or the use of different food forms and ingredients. This is also a particularly important reason for the multi-categories datasets, which can cover different regions, cultures, forms, and ingredients, thus improving the accuracy of model recognition and migration performance.

In general, we can divide the existing datasets into single-category datasets and multi-categories datasets according to their data characteristics. Single-category datasets usually only contain foods from one cuisine, and these foods usually have certain similarities in the form or use of ingredients. This single-category dataset can maximize the recognition accuracy of a proprietary model in a specific area. However, a single-category of the dataset may have some limitations, such as regional and cultural differences or the use of different food forms and ingredients. This is also a particularly important reason for the multi-categories datasets, which can cover different regions, cultures, forms, and ingredients, thus improving the accuracy of model recognition and migration performance.

2.1. *Single-category datasets*

2.1.1. *Western food*

FNDDS [44]: Bosch et al. (2011) [44] collected a large dataset on American food. This dataset includes about 7000 dishes in 11 categories. The image of each dish also includes the weight of various ingredients in the images, as well as the nutritional components. The dataset provides three interrelated data components: an image database containing data generated from food images, an experimental database containing data related to nutrition research and image analysis results, and an enhanced version of the nutrition database containing nutrition and a visual description of each food.

ETHZ Food-101 [45]: Bossard et al. (2014) [45] collected a real-world food dataset by downloading images over the Internet. These downloaded images contain labels for the location and type of food. The dataset contains 101,000 real-world images of 101 of the most popular dishes. 750 training images and 250 test images were randomly selected for each class. The image of the training set is cleaned up by additional manual effort, reducing errors and interference. However, the image of the training set is not cleaned up, which improves the robustness of the model.

FOOD201 [46]: Myers et al. (2015) [46] randomly selected 50,374 images from the food101 dataset for artificial semantic labeling. After combining similar semantics, the new dataset, called the Food201-MultiLabel dataset, contains 201 food categories. The dataset is divided into 35,242 training images and 15,132 test images according to the proportion of food101. About 12 K of the images are semantically segmented at the pixel level. This part of the data is called the Food201-segmented dataset. Therefore, this dataset can be used in both food classification and food segmentation studies.

2.1.2. *Chinese food*

ChineseFoodNet [18]: Chen et al. (2017) [18] collected 19.4 GB images of Chinese food from the Internet. Each image in the dataset is manually tagged to remove the error tags that corrected the original data. This dataset contains 185,628 images of 208 food categories. Images in the dataset maintain the original appearance of the image without any processing. The dataset is divided into training sets (145, 066), validation sets (20,254), and test sets (20,310) at a scale of 8:1:1.

Chen [35]: Chen et al. (2012) [35] built a food recognition platform through the Internet and collected 5000 images of food from mobile phones. This dataset contains 50 popular Chinese food categories, each with 100 images. The dataset was randomly divided into five groups through five-fold cross-validation: one group was retained as validation data, and four groups were used as training data.

Vireo-Food 172 [36]: Chen et al. (2016) [36] believed that since the number of ingredients is far less than food categories, identifying ingredients takes more time for network migration and expansion than identifying dishes. Training the model to learn the mutual but fuzzy relationship between ingredient recognition and food classification can improve the model's ability to deal with zero-shot retrieval problems. They obtained 172 categories of food images from two Chinese food websites. More than 100 images larger than 256 × 256 were collected in each class. The ingredients in the image were labeled by ten housewives. This dataset yielded 1041 images with 353 ingredient tags, with an average of 3 ingredients per image. On average, there were 640 positive samples in each food class and 745 positive samples in each ingredient.

ChinaMartFood-109 [47]: Ma et al. (2022) [47] collected 10,921 images of 109 kinds of food from the Chinese market. This dataset not only includes the classification of food but also includes the classification of 23 nutrients, covering 18 major food groups.

2.1.3. *Fast food*

PFID [48]: Chen et al. (2009) [48] collected 101 classes of fast food from 11 popular fast food chains. These examples were expanded in a restaurant environment and a laboratory-controlled environment. The dataset consisted of 4545 still images, 606 stereo images, 30,3360 motion structure videos, and 27 privacy-preserving videos of volunteer eating events. Each instance of each class consists of four packaged still images and four unpacked images in the dining room environment, six still images in the laboratory environment, two sets of stereo images along the long and short sides of the object, and a 360-degree video of the food on the turntable.

2.1.4. *Japanese food*

Food50 [49]: Taichi et al. (2009) [49] collected 5000 images of food shared by netizens from the Internet. The image cuts out the area where the target food is located to obtain a clear target food image. The dataset contains 5000 images from 50 popular Japanese foods. On average, each food class contains 100 images.

Food85 [50]: Hoashi et al. (2010) [50] collected 85 kinds of food popular in Japan on the Internet. 100 images were collected for each class. The dataset has 8500 images in total. These images place the food in the center of the image by clipping. Some food classes may not be included in other comprehensive datasets. This dataset is very helpful for training food recognition models used in Japan.

Foodlog [37]: Tatsuya Miyazaki et al. (2011) [37] collected daily food images uploaded by users through a public Web service called FoodLog. Users from Japan upload these images and basically represent Japanese daily food. Nutritionists label the images according to five classes: grains, vegetables, meat/beans, fruits, and dairy products. Each image shows the average calories provided by the food for each person. The dataset contains 6512 images in total, of which 5512 images are used as a training set, and the other 1000 images are used as a test set.

UEC-FOOD-100 [38]: Matsuda et al. (2012) [38] collected 9060 images of Japanese food. These images contain a total of 100 kinds of food. There is at least one food sample in each image. Each food sample is manually marked with a binding box. The test set consists of 1200 images containing only one food sample and 500 images containing a total of 1200 food sample points. The rest of the images are training sets.

UEC-FOOD-256 [51]: Kawano et al. (2014) [51] continued to collect more images based on the UEC-FOOD-100 dataset and proposed the UEC-FOOD-256 dataset. This dataset contains 256 food classes. Compared with the UEC-FOOD-100 dataset, the paper gives this dataset a download link.

FLD [52]: Yu et al. (2018) [52] downloaded all food images in the past two years from the foodlog website. These images were uploaded by users in Japan. The dataset contains not only 623,956 images, including 1,508,171 food uploaded by more than 20,000 ordinary users, but also the owner ID and timestamp of each image. The dataset includes 1870 general food classes defined by the system and 97,457 other food classes defined by the user. The dataset is divided into two subsets. The first subset FLD-469 consists of 469 main classes by default of the system, with 500 images for each class. Another subset of FLD-CLS consists of 209,700 images, including the first 300 food records from 699 user-defined classes.

SuShi-50 [53]: According to the Sushi Guide, Qiu et al. (2019) [53] collected a fine-grained food dataset from the Internet that contains 50 different types of sushi. Each class has more than 50 images. The dataset contains 3963 images, half of which are test sets, and the other side is training sets.

2.1.5. *Food of other nationalities*

Indian Food Database [54]: Pandey et al. (2017) [54] collected a special dataset for Indian food. This dataset includes 50 kinds of food that are popular in India. Each class contains 100 images. The same class of food may vary in color, texture, shape, and size because Indian food lacks any common layout. Eighty images from each class were randomly selected as training sets, and 20 were test sets.

Pakistani Food Dataset [55]: Tahir et al. (2020) [55] collected a new dataset for Pakistani food. This dataset contains a total of 100 foods. A total of 4928 images were collected, of which 4448 were training sets, and 480 were test sets. The high degree of food similarity in Pakistan makes this dataset more difficult to train than other datasets.

Rice dataset [56]: Stütz et al. (2014) [56] collected a dataset about rice. The dataset contains only images of rice. These images are tagged with the quality of the rice and the energy contained in the rice tagged by a nutritionist. This dataset is used to identify the quality and energy of rice.

THFood-50 [57]: Termritthikun et al. (2017) [57] collected 50 popular Thai food images from the Internet. Each Thai food contains about 200 to 700 images. The resolution of the image is uniformly adjusted to 256 × 256 pixels. 90% of the images in the dataset are in the training set, and the remaining 10% are in the test set.

Turkish-Foods-15 [58]: Güngör et al. (2017) [58] collected a large number of images of Türkiye's food from the Internet. These images include a total of 15 Türkiye dishes. Each class contains at least 500 images.

2.1.6. *Raw vegetables and fruits*

Fruits 360 [59]: Mureşan et al. (2018) [59] obtained 38,409 images containing 60 kinds of fruits by recording the fruits on the low-speed rotating motor and extracting screenshots. All the images are centered on fruit with white background. The fruit in the image is accurately segmented and extracted and scaled to a 100 × 100 size background-free image. This is a large and abundant dataset which can well train a robust fruit classification model.

VegFru [60]: Hou et al. (2017) [60] collected a dataset of fresh vegetable and fruit images in 2017. VegFru is a huge dataset containing vegetables and fruits closely related to everyone's daily life. It contains more than 160,000 images in total. VegFru classifies vegetables and fruits according to their edible characteristics and divides 160,731 images into 25 classes and 292 subclasses. In VegFru, there are 91,117 vegetable images and 69,614 fruit images. Images of vegetables and fruits are placed separately, so VegFru can also be divided into two datasets, Veg200 for vegetables and Fru92 for fruits. The number of images per subclass ranges from 200 to 2000. The first 100 images in each subclass are used for training sets, the next 50 images are verification sets, and the rest images are test sets

FruitVeg-81 [61]: Waltner et al. (2017) [61] collected 15,630 images of 81 different fruits and vegetables by taking photos of fresh fruits and vegetables and collecting online images. This dataset has a wide variety of classes and a huge number of images. This data is completely open access and an important dataset in the field of food classification.

HyperspectralFruVeg [62]: Using a hyperspectral camera, Steinbrener et al. (2019) [62] record 2700 images of 13 different kinds of fruits and vegetables with 16 spectral bands from 470 to 630 nm camera. This is the first hyperspectral dataset of fruit and vegetable images. The recorded images of 16 frequency bands are spliced into one image in order in the form of a grayscale. The recorded images are cut to 256 × 256 for easy training. Of these cropped images, 700 were randomly selected as the verification set. The rest 2000 images are used as training sets.

FruitNet [63]: This dataset was collect by Meshram et al. (2022) [63]. This dataset used three kinds of photographic equipment to photograph six common fruits in India from July to October: apple, banana, guava, lime, orange, and pomegranate. The dataset is divided into three folders: Good quality fruits, Bad quality fruits, and Mixed quality fruits. Each folder contains all six classes of fruits. The dataset has 19,526 images. Among them, the class of Good quality has 11,664 images, the class of Bad quality has 6778 images, and the class of Mix quality has 1074 images. The size of the image is 256 × 256.

Citrus Fruits [34]: Dhiman et al. (2021) [34] collected this dataset for citrus quality classification. The dataset collected 4136 from online sources and annotated by domain experts. The dataset contains four classes in total: health, Low Severity, middle Severity, and high Severity. There are 1466 images in the health class, which are divided into 1173 in the training set and 293 in the test set. The other three classes are basically about 700 training set images and 200 test set images. This dataset focuses on the quality of citrus. It is a detailed dataset related to food safety.

Fruits yield [64]: Behera et al. (2021) [64] collected a dataset on fruits from the different farming sites of Sambalpur and Sundargarh districts of Odisha, including five Indian fruits, namely mango, apple, orange, pomegranate, and tomato. The dataset has 1000 images taken on-site for training, including 200 images in each class, with a standard resolution of 800 × 600. In addition, a total of 100 images, with 20 images of each class in 5 different classes taken on-site, are used for the test set.

XiaotangshanVeg [65]: This dataset focuses on vegetable diseases collected by Zhou et al. (2021) [65]. The data includes 4284 images in 6 classes: tomato health, tomato powdery mildew, tomato early blight, cucumber health, cucumber powdery mildew, and cucumber downy mildew. The data collected the field images of plants in Xiaotangshan base at three different time periods: morning (7:00–8:00), noon (11:00–12:00), and evening (17:00–18:00). These images are divided into training, validation, and test subsets at a ratio of 7:2:1. Images in the dataset include simple images with obvious disease characteristics and difficult images with mixed target and background.

2.2. *Multi-categories datasets*

TADA [42]: This is a very early dataset. Mariappan et al. (2009) [42] collected 306 images in 2009, of which 50 were food replicas, and 256 were real food. Food replicas are images taken under specific conditions, such as placing food on a white plate on a black and white colored checkerboard pattern tablecloth. Tablecloths are used as benchmarks for estimating food. White plates are used to help split food. 17 images were used for training and 33 for testing. The image of the training set contains only one sample, and the image of the test set contains 2–3 samples. There are 32 food classes. For real food, 11 images are training sets, and 245 images are test sets. Half of the real food was taken under good lighting, while the other side was taken under poor lighting.

UNCIT-FD889 [66]: Farinella et al. (2015) [66] collected 889 different dishes from different nationalities, such as Italy, Britain, Thailand, India, and Japan, in four years. The dataset contains 3583 images in total, all of which are taken from real food. Each dish includes nearly four images from different perspectives or lighting. Since each dish has multiple presentation methods, building a more accurate food recognition model may be possible.

UPMC Food-101 [67]: After referring to ETHZ Food-101, Xin et al. (2015) [67] collected a comprehensive food dataset. This dataset includes 101 classes of food and 90,840 images in total. The number of images in different classes ranges from 790 to 956. In addition, 93,533 images in the dataset contain the original HTML source page. Another 86,574 images with text descriptions. Compared with other food101 datasets, the images of the UPMC Food-101 dataset no longer come from a single class of food but include food from different cuisines.

UNIMIB2015 [68]: Ciocca et al. (2015) [68] collected photos of the dishes before and after meals of 1000 people in real scenes. Each group of images includes 3 random dishes. The dataset includes 15 different categories of dishes. All photos are captured by an automated camera. Through the detection of the tray, the specific dish images were obtained from the original photos. All 2000 images have been manually annotated, including the ground truth for dish classification and the ground truth for residual estimation. Of the total 2000 images, 600 images are used as training sets, and the remaining 1400 images are test sets.

Dishes [69]: Herranz et al. (2015) [69] collected images from 187 restaurants and 701 unique food categories from a city on the Internet. More than 15 images are obtained for each dish, 10 of which are used as training sets. This dataset is innovative and hopes to link the classification of dishes, recognition of restaurants, and geographic information features.

Menu-Match [70]: Beijbom et al. (2015) [70] collected a group of real dining images from three local restaurants and built a dataset for classifying dishes and identifying nutrient elements. They chose an Italian restaurant, a Chinese restaurant, and a soup restaurant. The dishes are randomly selected by customers and photographed by photographers from multiple angles. Finally, they collected 646 images of 41 dishes. And 1386 food ingredients are annotated from these images. Nutritionists have provided near-real nutritional markers. Therefore, the dataset contains accurate nutrition information and real food images.

UNIMIB2016 [71]: Ciocca et al. (2017) [71] collected the UNIMIB2016 dataset on the basis of UNIMIB2015 in order to improve the model's ability to identify multiple foods. The images are from the canteen of Milan University. The image is captured by mobile phone. They collected a total of 1442 images that had passed the quality inspection stage. After removing overly blurred images and duplicate photos, this dataset has 1027 images with trays. These images include 73 food classes and a total of 3616 food samples. Compared with UNIMIB2015, UNIMIB2016 has added more fast-food images of canteens. In addition, many images also add interfering objects that do not belong to food, such as mobile phones, wallets, and keys.

Food-975 [41]: Zhou et al. (2016) [41] collected 37,885 food images from 6 restaurants in order to study ultra-fine grain image recognition. They hope that the neural network model can tell which restaurant these dishes come from from the subtle differences in images. These images contain a total of 975 different dishes. 4951 photos from a controlled environment and 351 images from a specific website constitute a complex test set to test the robustness of the model. Finally, the dataset contains a three-tier hierarchy. The top layer is a nutritional classification table based on 51 food ingredients; In the middle layer are 781 different kinds of dishes created by aggregating restaurant labels; At the bottom layer are 975 fine-grained labels.

Food500 [72]: Merler et al. (2016) [72] collected a huge food dataset to enable neural network models to learn as many food classifications as possible. They have collected over 150 K effective food images worldwide via the Internet. These images contain 508 different foods from different countries. An average of 292 frontal images were taken from each class. The smallest class has 26 images (yellow corn chips), and the largest class has 489 images (gulab jaamun). Despite a large number of images, each image has been labeled manually by three groups of labelers, reducing the probability of mislabelling. This huge open-source dataset is good for training a food classification model.

Food11 [73]: Singla et al. (2016) [73] collected an 11-classes food dataset by using and modifying the major food classes defined by the United States Department of Agriculture (USDA): bread, dairy products, desserts, eggs, fried food, meat, noodles/noodles, rice, seafood, soup, and vegetables/fruits. This dataset combines images from existing datasets with images of food on the web. For each class, the images in the dataset contain a wide variety of ingredients to make training more difficult. This dataset contains 16,643 images, of which 9866 images are training sets, 3430 images are validation sets, and the remaining 3347 images are test sets.

Food5K [73]: This dataset collects 5000 images from other publicly available datasets. These include 2500 images of food and 2500 images of people and other non-food items. There are 3000 images for the training set, of which 1500 are food images, and 1500 are non-food images. The test and validation sets have 1000 images, respectively, including 500 food and 500 non-food images. The dataset attempts to add a large number of non-food images that are visually similar to food images, thus increasing the difficulty of classification tasks.

UNCIT-FD1200 [74]: Farinella et al. (2016) [74] collected a dataset consisting of 4754 food images to study the food monitoring system. In order to show more geometric and photometric changes in the entire dataset, each plate of food was photographed from multiple angles many times. The dataset is divided into eight categories: appetizer, main course, second course, single course, side dish, dessert, breakfast, and fruit. Specifically, the dataset contains a total of 1200 images of different dishes. These images come from different dishes of different nationalities, for example, Britain, Japan, India, Italy, Thailand, etc.

Instagram 800k [75]: Rich et al. (2016) [75] collected a super huge dataset from the images they shared on Instagram. By studying the relationship between images and text on Instagram posts, the dataset can be used to understand the relationship between the content of the real image taken and the label. The dataset collects images with food-related labels from October 2014 to March 2015 for two different periods (winter and spring). The dataset contains images of the 43 most popular food-related labels on Instagram. The dataset also contains image-related metadata such as favorites, reviews, titles, GPS locations, and tags. The dataset consists of a total of 808,964 images.

EgocentricFood [76]: In real life, food is not necessarily in the center of the image. Bolaños et al. (2016) [76] improve the accuracy of food classification by simultaneously locating and identifying food. Using the wearable camera Narrative Clip, the dataset collected 5038 images from 9 different classes (glass, cups, cans, jars, cups, bottles, plates, food, and baskets). The image was labeled with 8573 boundary box annotations by manual annotation. This is the first image dataset to be used for locating and identifying food-related objects.

FOOD524DB [77]: Ciocca et al. (2017) [77] are concerned about the drawbacks of existing datasets in terms of the number of food classes or the number of images in a single class. By combining and collating existing Food50, Food-101, UECFOOD-256, and VIREO datasets, they obtained a very large Multicategory food dataset of 247,636 images. The dataset reduced the total number of food classes by combining duplicate classes describing approximations to 524. Each class contains more than 100 food images. There are 241 classes that contain 100 to 199 images, 58 classes that contain 200 to 499 images, 113 classes that contain 500 to 999 images, and the remaining 112 very popular food classes that contain more than 1000 images.

ISIA Food-200 [39]: Min et al. (2019) [39] built a vocabulary of food classes through the "food list by ingredients" in Wikipedia. Then they collected 197,323 images on the Internet according to the vocabulary. The images were divided into 200 food classes and 319 visible ingredients. Each food class has more than 500 images. The dataset is divided into the training set, verification set, and test set at the rate of 6:1:3.

FoodX-251 [78]: Kaur et al. (2019) [78] collected a total of 251 food classes, including 158k images collected from the Internet. 12k images are verification sets, 28k images are test sets, and the remaining 118k images are training sets. All images are labeled manually. These food classes are similar in fine-grained and visual terms, such as different types of cakes, sandwiches, puddings, soups, and pastries. The dataset includes 15 different types of cakes and 10 different types of pasta. This dataset can be used to train the model to recognize different types of food with similar appearances.

FoodAI-756 [79]: Sahoo et al. (2019) [79] collected a large number of popular food in Singapore through the Internet. The dataset defines 152 "superclasses" representing common types of food and beverage, including beer, fried rice, roast chicken, ice cream, etc. These superclasses are further subdivided into 756 kinds of food. There are at least 500 images (65,855 images in total) collected for each of the 100 visual foods in 8 classes (such as India, China, desserts, Malay, etc.). In general, there are 756 visual foods in this dataset, including about 400,000 images. Each food has at least 174 images and a maximum of 2312 images. The training set contains 377k images, the verification set contains 7.5k images, and the best test set contains 38k images.

ISIA Food-500 [40]: Min et al. (2020) [40] summarized the most representative 500 food classes from 4943 food classes in the network and existing datasets. There are 399,726 images in the dataset, with an average of 800 images in each class. The food classes in this dataset include oriental and western cuisines. According to the GSFA standard, the foods in our dataset and the existing typical dataset are mainly concentrated in 11 classes: meat, grains, vegetables, fish, fruits, dairy products, baked goods, fat, candy, beverages, and eggs. Compared with other datasets, the ISIA food-500 has added classes such as dairy products and beverages. The class tag of the dataset is represented by more than two words and connected with "-", such as "pea-soup".

Food2K [80]: Min et al. (2021) [80] collected more than 1 million images containing 2000 classes. Compared with existing food recognition datasets, Food2K exceeds them by an order of magnitude in terms of classes and images. This is a new and challenging benchmark for developing new food visual models. With the help of experts in the food field, they defined 12 superclasses, such as bread and meat. Then this superclass was subdivided into food material classes, such as pork and beef. Finally, they obtained 1710 kinds of oriental food and 290 kinds of western food.

Allrecipes [43]: Gao et al. (2020) [43] built a large-scale dataset containing food nutrition data from the formula-sharing platform Allrecipe.com. The dataset contains 68,768 users, 45 630 recipes, 33 147 ingredients, and 1093 845 interactions. Each recipe in the dataset has a corresponding nutritional fact bar that provides calorie information. A total of 52,821 images from 27 different food classes were included in the dataset, of which 60% were training sets, 10% were validation sets, and the remaining 30% were test sets. The images of the test set contain interactive information.

MAFood-121 [81]: Aguilar et al. (2019) [81] collected a food image data collection dish, cuisine, and classes from the Internet that contains three levels. The cuisine level is the largest base, with a total of 11 cuisines. Eleven traditional dishes are selected for each cuisine. Each dish belongs to at least one of the ten food classes: bread, eggs, fried food, meat, noodles/pasta, rice, seafood, soup, dumplings, and vegetables. The dataset consists of 121 dishes. Each dish has a maximum of 250 images. There were 21,175 images in the dataset, 72.5% for training, 12.5% for validation, and 15% for testing. Cuisine and dish can only take one value per image, while classes have multi-annotations.

FoodBase [82]: Popovski et al. (2019) [82] randomly selected 1000 semantic annotations of recipes with semantic tags from the Hansard corpus and post-processed the annotated semantic tags to generate the FoodBase dataset. Foodbase has collected 274,053 food entities in five classes (namely "appetizer/snack", "breakfast/lunch", "dessert", "dinner" and "drink"), and 13,079 unique food entities.

2.3. *Summary of datasets*

We investigated 51 datasets from 2009 to 2022 and classified them according to the food categories contained in the datasets. The specific classification and statistics are recorded in Table 1, and sample images of some representative datasets are listed in Fig. 2. Half of the datasets are multi-categories datasets. The single-category dataset mainly includes Chinese, Japanese, fast, and fresh vegetables and fruits. Among them, the number of datasets for Japanese food is the largest. Although there is no single-category dataset of French food and Italian food, the two most famous cuisines. However, we can find the data of these two cuisines from several large-scale multi-categories datasets.

Table 1
Summary of datasets.

Name	Food Category	NI	NC	Raw/ cooked	Annotation type	Collecting Method	Public	Time
FNDDS [44]	American	7000	11	cooked	classification	camera &existing food databases	√	2011
ETHZ Food-101 [45]	American	100,000	101	cooked	classification	Internet	√	2014
FOOD201 [46]	American	50,374	201	cooked	classification & segmentation	existing food databases	√	2015
ChineseFoodNet [18]	Chinese	185,628	208	cooked	classification	Internet	√	2012
Chen [35]	Chinese	500	50	cooked	classification	phone & Internet	×	2012
Vireo-Food 172 [36]	Chinese	110,241	172	cooked	ingredient recognition	Internet	√	2016
ChinaMartFood-109 [47]	Chinese	10,921	109	cooked	classification & nutrient element identification	Internet	√	2022
PFID [48]	Fast Foods	1388	15	cooked	classification	camera	√	2009
Indian Food Database [54]	Indian	5000	50	cooked	classification	Internet	√	2017
Food50 [49]	Japanese	5000	50	cooked	classification	Internet	×	2009
Food85 [50]	Japanese	8500	85	cooked	classification	Internet	×	2010
Foodlog [37]	Japanese	6512	2000	cooked	classification &energy regression	Acquired from user	√	2011
UEC-FOOD-100 [38]	Japanese	14,361	100	cooked	classification	Internet	×	2012
UEC-FOOD-256 [51]	Japanese	25,088	256	cooked	classification	Internet	√	2014
FLD [52]	Japanese	3,007,157	1195	cooked	classification	phone	√	2018
Sushi-50 [53]	Japanese	3963	50	cooked	classification	Internet	√	2022
Pakistani Food Dataset [55]	Pakistani	4928	100	cooked	classification	Internet & existing food databases	√	2020
Rice dataset [56]	rice	-	1	cooked	Energy regression	Acquired from user	×	2014
THFood-50 [57]	Thai	2500	50	cooked	classification	Internet	√	2017
Turkish-Foods-15 [58]	Turkish	7500	15	cooked	classification	Internet	√	2017
Fruits 360 [59]	Fruits	38,409	60	raw	classification	camera	√	2018
Fruits yield [64]	Fruits	1000	5	raw	classification	camera	×	2021
FruitNet [63]	Fruits	19,526	6	raw	classification	camera	√	2022
Citrus Fruits [34]	Fruits	4136	4	raw	classification	Internet	×	2022
XiaotangshanVeg [65]	Vegetables	4284	6	raw	classification	camera	√	2021
VegFru [60]	Vegetables & Fruits	160,731	292	raw	classification	Internet	√	2017
FruitVeg-81 [61]	Vegetables & Fruits	15,630	81	raw	classification	Internet	√	2017
HyperspectralFruVeg [62]	Vegetables & Fruits	2700	13	raw	classification	camera	√	2019
TADA [42]	Artificial and Generic Food	256	11	cooked	classification	camera	√	2009
UNCIT-FD889 [66]	MC	3583	899	cooked	classification	phone	√	2014
UPMC Food-101 [67]	MC	90,840	101	cooked	Classification	Internet	√	2015
UNIMIB2015 [68]	MC	2000	15	cooked	recognition and leftover estimation.	phone	√	2015
Dishes [69]	MC	117,504	1173	cooked	dishes classification, restaurant identification	Internet	√	2015
Menu-Match [70]	MC	646	41	cooked	classification & nutrient element identification	Internet	√	2015
UNIMIB2016 [71]	MC	1027	73	cooked	classification	camera	×	2016
Food-975 [41]	MC	37,785	375	cooked	classification	camera	×	2016
Food500 [72]	MC	148,408	508	cooked	classification	Internet	×	2016
Food11 [46]	MC	16,643	11	cooked	classification	existing food databases	√	2016
UNCIT-FD1200 [74]	MC	4754	1200	cooked	classification	phone	√	2016
Instagram 800k [75]	MC	808,964	43	cooked	Classification & geographic information	Internet	√	2016
EgocentricFood [76]	MC	5038	9	cooked	food localization and recognition	camera	√	2016
Food5K [73]	MC	5000	-	cooked	classification	Internet & existing food databases	√	2016
FOOD524DB [77]	MC	247,636	524	cooked	classification	Existing food databases	√	2017
ISIA Food-200 [39]	MC	197,323	200	cooked	classification	Internet	√	2019
FoodX-251 [78]	MC	158,846	251	cooked	classification	Internet	√	2019
FoodAI-756 [79]	MC	400,000	756	cooked	classification	Internet & existing food databases	×	2019
Allrecipes [43]	MC	52,821	27	cooked	classification & nutrient element identification	Internet	√	2019
MAFood-121 [81]	MC	21,175	121	cooked	classification	Internet	√	2019
FoodBase [82]	MC	274,053	13,079	cooked	classification	Internet & existing food databases	√	2019

(continued on next page)

Table 1 (*continued*)

Name	Food Category	NI	NC	Raw/cooked	Annotation type	Collecting Method	Public	Time
ISIA Food-500 [40]	MC	399,726	500	cooked	classification	Internet & existing food databases	√	2020
Food2K [80]	MC	1,036,564	2000	cooked	classification	Internet & existing food databases	√	2021

(MC = multi-categories; NC = number of classes; NI = number of images)

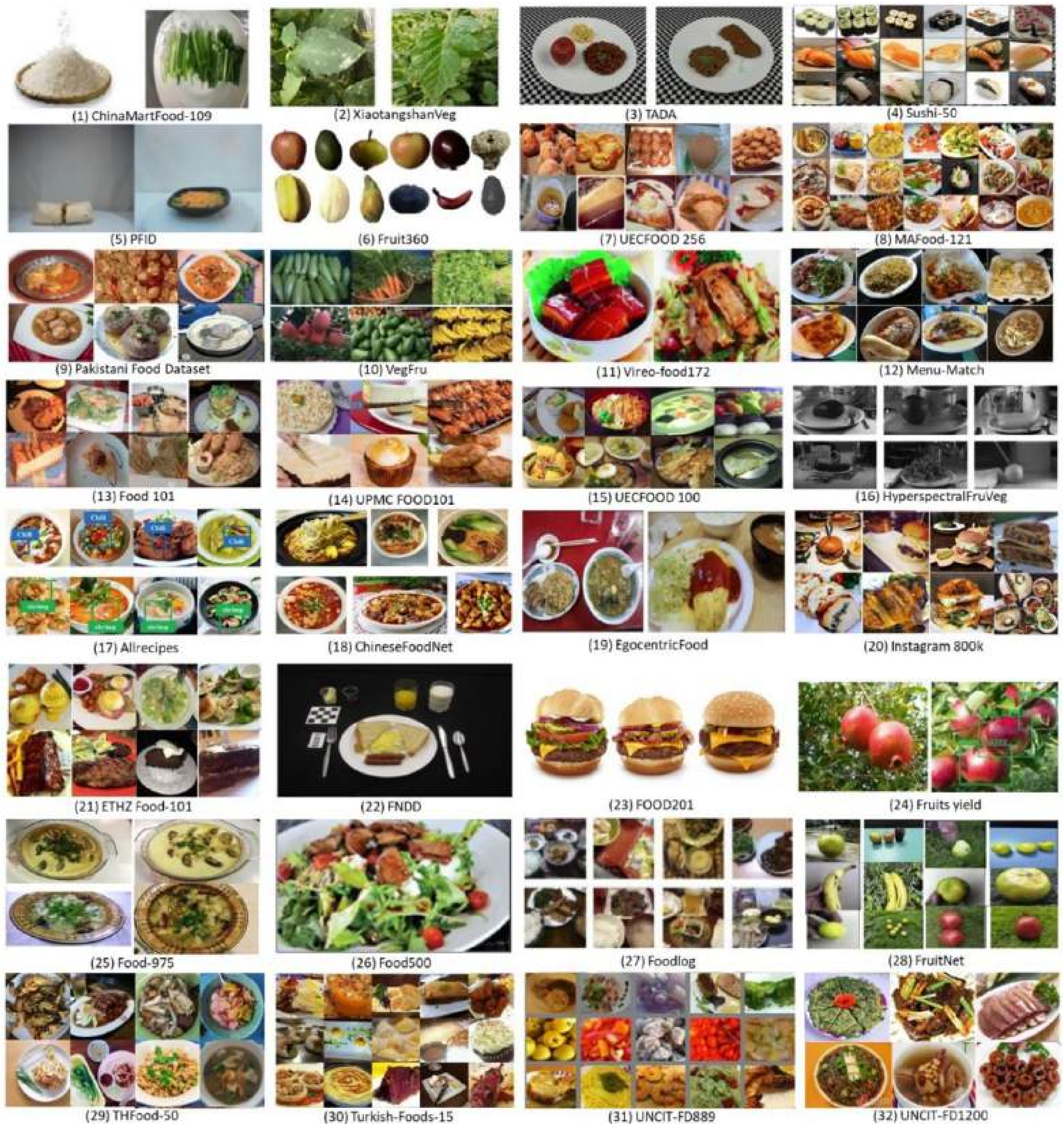


Fig. 2. Sample images from a few food datasets.

There is also a kind of fresh vegetables and fruits dataset in the single-category datasets. The classification of vegetables and fruits is of great significance in agriculture and retail trade. In addition to the basic classification of vegetables and fruits, the dataset of vegetables and fruits also cares about the maturity and quality of vegetables and fruits. This is the focus that other cooking datasets lack: food safety.

In recent years, datasets contain more and more images and more and more types of images. Rich data categories and quantities can better help us train a sufficiently flexible neural network model. Multi-categories datasets can provide more information to help the model classify the categories accurately to improve the accuracy and effectiveness of the model. In addition, the use of multi-categories datasets can effectively reduce the overfitting of the model to specific categories, thus reducing the system risk of the model under abnormal conditions. Therefore, Multi-categories datasets are particularly important in training machine learning models, which can better help us improve the accuracy and robustness of the models.

In addition, multi-categories datasets also pay more and more attention to food nutrition information and restaurant geographic information and also collect and analyze relevant consumer behavior, restaurant service quality, and other information so as to better explain the various factors affecting food classification and better help us understand the identification and analysis of food categories.

3. Food-oriented machine vision system

With the advances in deep learning and machine learning methods, a food-oriented machine vison system (MVS) is becoming increasingly significant to certain the delivery of products with high quality in terms of accurate packaging, food quality, and security. The food-oriented machine system will be able to capture the food information from different aspects, including size, dimensions, shape, surface, appearance, etc. Then, the captured data will be analyzed for monitoring purposes with the aim of reducing human labor and improving accuracy while reducing human errors.

The machine vision system should include cameras and data processing units to make next-stage decisions, as shown in Fig. 3. The captured data should be of the high quality that the data processing unit can process. So basically, the food-oriented MVS system contains two parts: (1) Data acquiring and (2) data processing units. Their roles are described in Table 2.

3.1. Stereo system, remote sensing, and hyperspectral imaging

A stereo system is a type of camera with two or more lenses that capture images for each lens [83,84]. It is mainly to mimic human binocular vision. The distance between cameras is similar to the distance between someone's eyes. It will therefore provide the depth information - the distance of an object to the camera together with the 2D image data to form the 3D data [85,86]. Compared to the traditional 2D camera, the stereo system is able to improve the accuracy of reconstructing real-world subjects [87]. With the advanced computing power, the stereo system is becoming increasingly popular for object recognition and environment modeling. For example, the stereo camera is popularly used in vehicle-to-vehicle distance estimation [88], hearing research [89], estimating the length composition of fish [90], etc. Nowadays, stereo cameras can be implemented by simply mounting cameras in pairs or one camera and one lens.

Remote sensing detects the physical object at a distance, for example, from satellites, aircraft, etc., by measuring the reflected and emitted radiation. Remoting sensing has been widely applied in many different fields, including geography [91,92], military [93,94], planning [95–97], economic development [98–100], commercial [101–103], and agriculture [104–106]. There are two different types of remoting sensing methods: active remote sensing and passive remote sensing. Their differences are shown in Table 3.

Different from traditional imaging methods that assign each pixel by primary colors (red, blue, and yellow), Hyperspectral image (HIS) is to

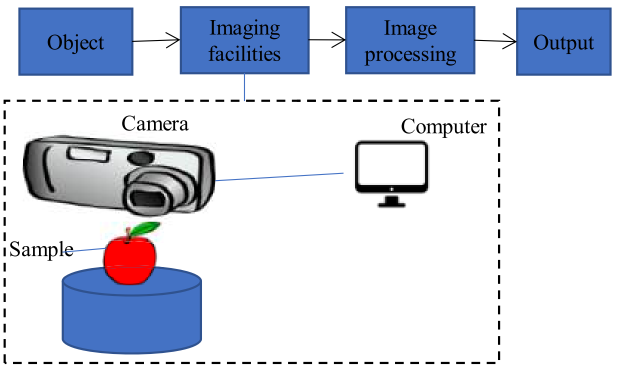


Fig. 3. Typical machine vision system.

Table 2
Parts of a food-oriented MVS system.

Part	Role
Data acquiring	The data-acquiring system determines the type and quality of the captured data. To certain the effectiveness of the output, the data-acquiring system should be of certain standards to ensure the captured data is valid.
Data processing	The data processing unit plays an important role as to process the captured data and then making the next stage decision.

Table 3
Active remote sensing versus passive remote sensing.

Type	Definition	Examples
Active	Active remote sensing is first to emit energy and then with a sensor to collect the energy reflected from the target	RADAR and LiDAR
Passive	Passive remote sensing is to detect the radiation emitted by the object/ surrounding areas	charge-coupled devices, radiometers, and infrared

analyze a wide range of spectrum of light. It is to divide the light that strikes each pixel into many different spectral bands to describe the collected information. HIS is developed by NASA's Jet Propulsion Laboratory in the late 1970s. It has become increasingly important in many different research fields in recent years. For example, Wieme et al. (2022) [107] stated that HIS could work as an effective imaging tool for assessing fruit, vegetables, and mushrooms. Porebski et al. (2022) [108] conducted a study to analyze the contribution of color imaging and hyperspectral imaging for texture classification.

3.2. X-Ray, thermal imaging, MRI, and additional imaging

X-Ray imaging is a painless imaging method that utilizes a penetrating form of high-energy electromagnetic radiation. John Hall-Edwards in Birmingham, England, on 11 January 1896, the first time used the X-ray for clinical conditions. He radiographed the hand of an associate with a needle stuck in. Then, on 14 Feb 1896, he first used the x-ray in a surgical operation. Nowadays, X-Ray has been widely used for the detection of bone fractures [109,110], dental problems [109,111], some types of injuries, pneumonia [112,113], some types of tumors [114,115], food inspection [116,117], etc. The food X-ray inspection system is mainly designed for the end of line quality control or to check the quality of the raw materials for further processing. It will help to detect the smallest contaminants and make sure the product will comply with all major food safety standards. Matsui, T. et al. (2022) [118] developed an automatic detection model based on X-ray imaging and image processing for stem end rots of 'Hass' avocado fruit.

Thermal imaging is to record the subject's temperature as it emits infrared energy and then assigns the temperature a shade of color, therefore, helping to visualize the heat which is not visible to human eyes [119,120]. The thermal camera is equipped with a heat sensor capable of detecting tiny temperature differences [121,122]. Therefore, the thermal camera is to collect the objects' infrared radiation and then creates an image based on the collected information to improve the visibility of objects in a dark environment.

Thermal imaging has been applied to many different research fields. McGinnis et al. (2022) [123] developed new thermal imaging to detect the microvasculature during surgery to implement real-time image acquisition. Sarhadi et al. (2022) [124] applied machine learning for damage detection in glass-epoxy composite materials based on thermal imaging. There are many other fields where thermal images are applied, including documenting and quantifying disease activity in rheumatoid arthritis (RA) [125], automatic detection, segmentation, and classification of breast lesions from thermal images [126], etc. Thermal imaging has also emerged as a powerful non-destructive measurement technique in the food industry [127].

Magnetic resonance imaging (MRI) is a noninvasive method widely used in clinical to examine organs, tissues, and skeletal systems [128, 129]. It produces high-resolution images of the inside of the body that help diagnose a variety of problems [130]. MRI scanners employ strong magnetic fields, magnetic field gradients, and radio waves to form images of the organs in the body [131,132]. During the imaging process, radio waves are sent from the MRI machine, moving atoms in the object out of their original position. Afterward, the atoms return to their original position and send back radio signals while the radio waves are turned off [133,134]. Therefore, the computers will convert the received signals into an image of the part of the object examined. MRI is widely used in clinical medicine and biomedical research.

Meanwhile, MRI also allows the visualization of the structure of food noninvasively and enables the determination of the characteristics or texture of the food as a non-destructive imaging method. For example, Nagata et al. (2016) [135] developed an outdoor MRI system to measure the sap flow in a living tree. Collewet et al. (2022) [136] used Multi-exponential MRI T2 to classify and characterize fruit tissues. Winisdorffer et al. (2015) [137] employed MRI to investigate the water status and distribution at the subcellular level in whole apple fruit.

Some imaging methods can be used to obtain information about the target subjects. For example, Cai et al. (2023) [138] utilized Raman scattering spectroscopic imaging for the Characterization and recognition of citrus fruit spoilage fungi. Zou et al. (2022) [139] investigated Mass spectrometry imaging (MSI) as a tool for food microbiology. Verdú et al. (2023) [140] proposed to use of laser scattering imaging combined with CNNs for the modeling of the textural variability in a vegetable food tissue.

4. Preprocessing: data augmentation

Food image datasets are commonly small in size, which will raise the problem that weakens the classification models' generalization competence, which stands for the so-called 'performance gap' of a classifier assessed on the test set S_{test} and training set S_{train}. This Chapter expatiates the definitions and techniques of DA, an effectual image-domain method to alleviate overfitting.

4.1. Small-size dataset and its solutions

For a small-size dataset (SSD) [141], as the training set is small, the model has fewer samples to learn from, thus increasing the risk of overfitting [142]. Fig. 4(a) shows the performance curve where the overfitting takes place at the 7th epoch. We can clearly spot that the test error E_{test} starts to increase at 7-th epoch while the training error E_{train}

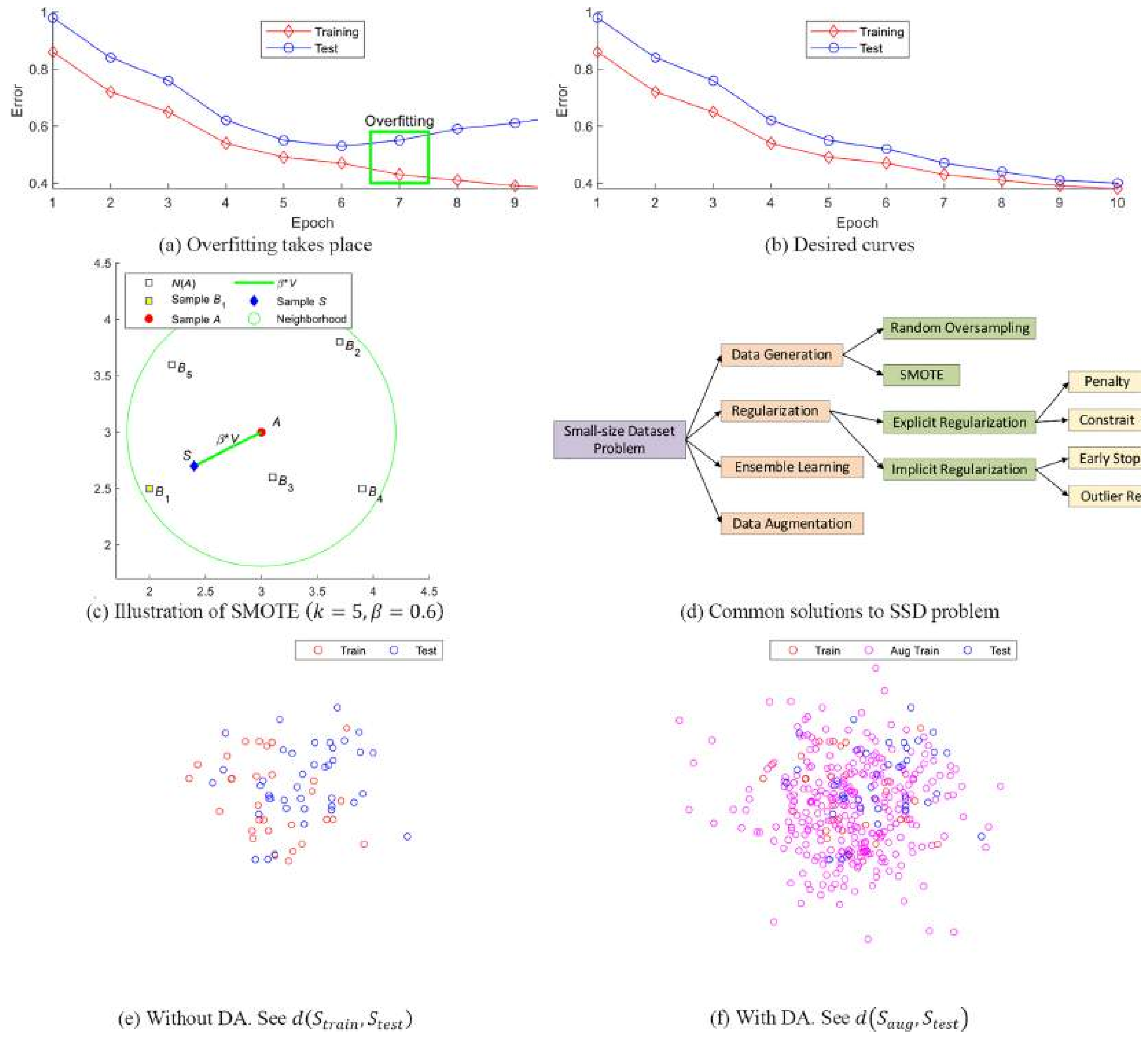


Fig. 4. Illustration of SSD and its solutions. (a and b) training and test performance curves. (c) SMOTE, (d) common solutions, (e and f) DA reduces $d(S_{train}, S_{test})$.

remains to diminish. Fig. 4(b) shows two favorite performance curves, in which both the curves of E_{train} and E_{test} drop till Epoch 10.

There are three types of classical solutions to SSD problems: (i) data generation (DG), (ii) regularization, and (iii) ensemble learning (EL) [143]. DG makes artificial data from a sampled data resource. Random oversampling is a robust method used to create multiple copies of some of the minority classes. The oversampling can be carried out more than once (2x, 5x, 10x, etc.). The synthetic minority over-sampling technique (SMOTE) [144] belongs to a representative DG method. For example, we take a sample A from the minority class and select its k nearest neighbors, $N(A) = \{B_1, B_2, \ldots, B_k\}$. We randomly select one neighbor sample from the k neighbor samples $N(A)$. Suppose we pick up B_1. SMOTE first draws a line V from sample A to B_1. V is defined as $V = B_1 - A$. Then, an artificial sample S is made by,

$$S = A + \beta \times V, \tag{1}$$

where $\beta \in [0, 1]$ is a random amount, obeying the uniform distribution.

Fig. 4(c) gives an illustration of the concept of SMOTE. The red circle stands for point A, which has $k = 5$ samples around it, each sample is with the shape of a square. The neighborhood is represented by a big green circle. The algorithm randomly selects a sample B_1, which is represented as a yellow square. Assuming $\beta = 0.6$, the sample S is represented as a sapphire diamond.

Regularization is largely used for the regularizing models' weights [145]. It makes the model to be more 'simpler'. Suppose a measure of the magnitudes of the weighs of a model $\mathbb{M}$ is denoted as $w(\mathbb{M})$, the large weights will create the model $\mathbb{M}$ unsteady. Assume the stability of $\mathbb{M}$ is denoted as $s(\mathbb{M})$, we have $w\uparrow \Rightarrow s\downarrow$. The reason is slight changes in the inputs yield big changes in the output for large-weight models. In contrast, small weights are considered more regular (viz., less specialized), thus, making the corresponding model more stable. $w\downarrow \Rightarrow s\uparrow$.

Regularization has two types: explicit regularization and implicit regularization [146]. The first one explicitly adds a penalty or constraint term to the optimization problem. The latter is all other regularization forms, such as early stopping, outlier removal, etc. EL approaches [147] employ various models to attain superior predictive performance to any individual model.

As shown in Fig. 4(d), DA is a method that works out SSD problem from its root, S_{train}. The augmented data by DA stand for a more all-inclusive training set. Accordingly, DA helps minimize the distance d between S_{train} and S_{test}, namely, $d(S_{aug}, S_{test}) < d(S_{train}, S_{test})$.

Fig. 4(e) illustrates the decrease of $d(S_{train}, S_{test})$ after DA, in which each dot stands for a sample food image. It indicates S_{train} cannot envelop the qualities of the S_{test}. Hence, the model based on the S_{train} may overfit. Fig. 4(f) illustrates the area of S_{train} is distended by data augmentation, now the augmented training set S_{aug} cover the area of S_{test}. So, we can conclude that $d(S_{aug}, S_{test})$ is now less than the original $d(S_{train}, S_{test})$ with the help of data augmentation.

4.2. Safe and unsafe DA

DA is commonly used for food category classification. The reason is that food image collection is rather costly and labor demand. Further, DA is also able to help food detection and semantic segmentation.

The safety of DA is another worthy aspect. Assume there is an image I, and its corresponding correct label is L. The definition of a safe DA D_{safe} is: $L[D_{\text{safe}}(I)] = L(I)$. Namely, a safe DA is label-preserving. However, in other instances, the unsafe DA D_{unsafe} changes the labels as $L[D_{\text{unsafe}}(I)] \neq L(I)$.

The definition of safe or unsafe DA is domain-dependent [20], and its accurate determination requires expertise and knowledge. For example, after comparing Fig. 5(a and b), we can conclude rotation is safe for the gammon steak. However, after comparing Fig. 5(c and d), we find rotation is unsafe for digit recognition, as six will be recognized as nine after a 180-degree rotation.

Meanwhile, the safe or unsafe of some DAs is also amount-dependent. Suppose we have the fries image, shown in Fig. 5(e). The injection of a small quantity of noise is safe, as shown in Fig. 5(f), but the injection of a huge quantity of noise is unsafe for the same fries image, as spotted in Fig. 5(g).

4.3. Geometric transforms

Geometric transformations (GTs) are prevalent DA resolutions to enlarge the number of images in S_{train}. The benefit of GT is its simplicity. The weakness is (i) surplus storage memory, (ii) additional computation

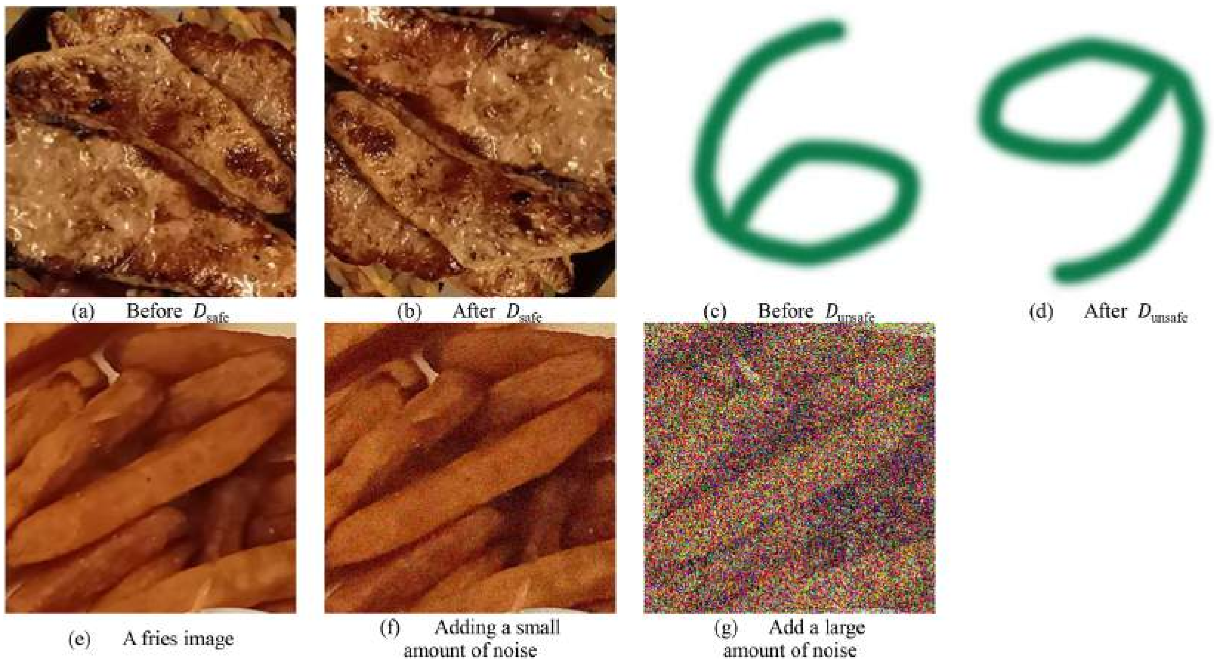


Fig. 5. Illustration of safe and unsafe DA. (a and b) a safe 180-degree rotation, (c and d) an unsafe 180-degree rotation, and (e-g) Noise injection may be safe or unsafe depending on the noise amount.

cost, and (iii) extra training period. GT should be examined wisely because a number of GTs possibly will change the images' labels.

4.3.1. Flipping, scaling, rotation, shear, and cropping

With flipping, we reflect the raw image I along a specified line, obtaining a mirror image of I. Flipping is one of the easiest and most clear-cut DA methods. Vertical flipping (VF) is less prevalent than horizontal flipping (HF) [148]. Tests on CIFAR-10, ImageNet, and other food datasets demonstrate the efficiency of flipping. Flipping results are unsafe on datasets, for example, MNIST or SVHN, because those two datasets contain transcripts and numerals. Fig. 6(a) displays an original rice image, Fig. 6(b) shows the horizontal flipping result, and Fig. 6(c) shows the vertical flipping result.

Scaling is a linear transform that either enlarges or shrinks the image by a scaling factor s which applies the same degree in both directions. Assume the pair of coordinates (POC) of the raw pixel is $[x_1, y_1]$, and the POC after the shear transform is $[x_2, y_2]$, the scaling is defined as

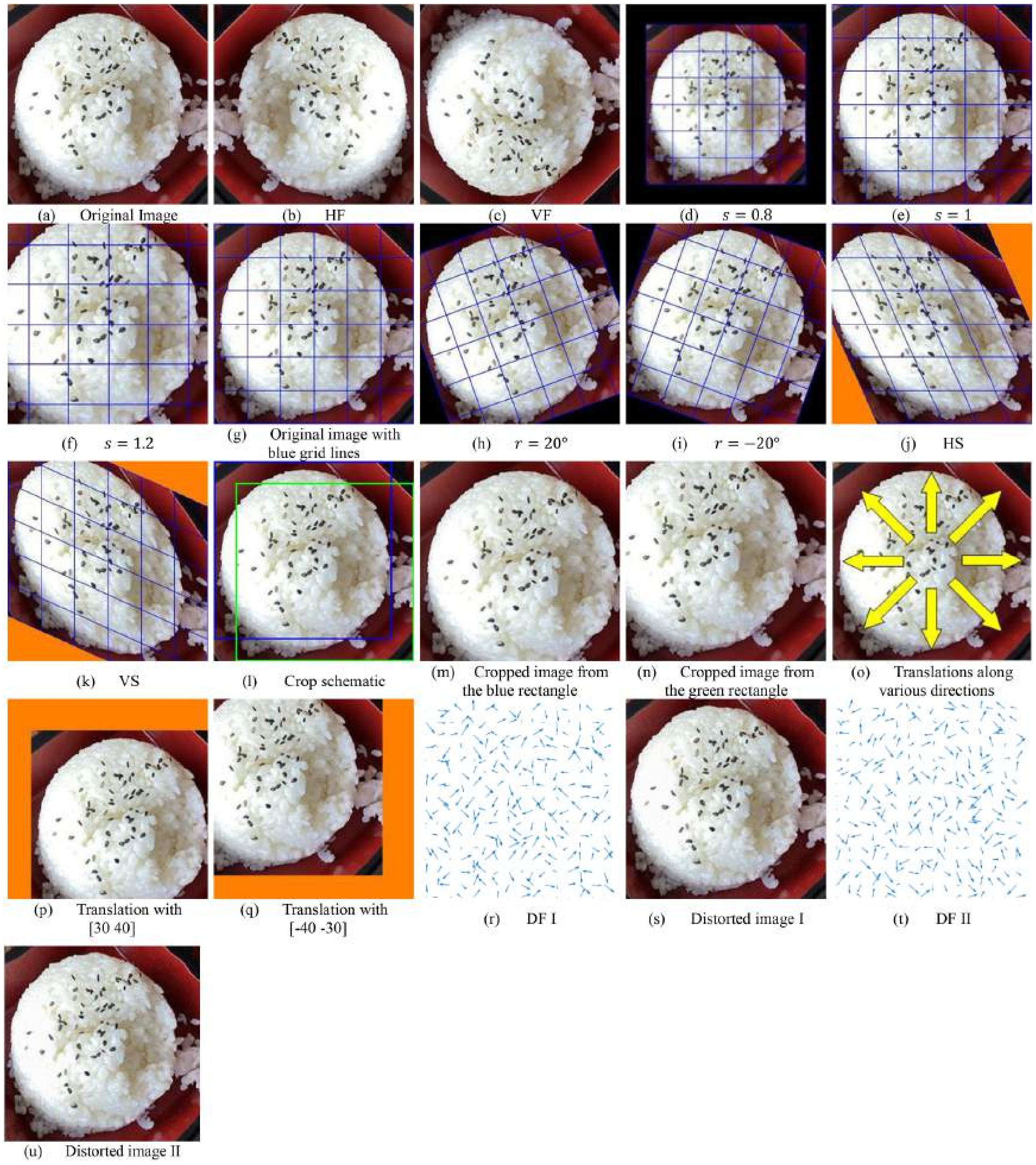

(a) Original Image
(b) HF
(c) VF
(d) $s = 0.8$
(e) $s = 1$
(f) $s = 1.2$
(g) Original image with blue grid lines
(h) $r = 20°$
(i) $r = -20°$
(j) HS
(k) VS
(l) Crop schematic
(m) Cropped image from the blue rectangle
(n) Cropped image from the green rectangle
(o) Translations along various directions
(p) Translation with [30 40]
(q) Translation with [-40 -30]
(r) DF I
(s) Distorted image I
(t) DF II
(u) Distorted image II

Fig. 6. Illustrations of geometric transforms. (a) original image, (b and c) horizontal and vertical flipping, (d-f) scaling with various s values, (g) raw rice image with blue grid lines, (h and i) rotation results, (j and k) horizontal and vertical shear, (l) crop schematic, (m) Cropped image from the blue rectangle, (n) Cropped image from the green rectangle, (o) Translations along various directions, (p and q) translation results, (r-u) displacement fields and distorted images.

$$[x_2, y_2, 1] = [x_1, y_1, 1] \times \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

When $s = 1$. Scaling returns an identical image. When $s > 1$, scaling is called dilation. When $s < 1$, scaling is called contraction. Fig. 6(d-f) show the scaling result with $s = 0.8$, $s = 1$, and $s = 1.2$, respectively. Note here blue grid lines are added for better vision.

Rotation DA is the movement of I around a given point. Frequently, a clockwise rotation corresponds to a negative rotation angle r, whereas an anticlockwise rotation corresponds to a positive rotation angle r.

In the DA circumstance, I rotates around the central pixel [149]. Small-angle rotations, for example, r within $[-15^\circ, 15^\circ]$ are normally safe for numeral and transcript recognition, whereas a large-angle rotation such as within $[-90^\circ, 90^\circ]$ might cause unsafety, viz., the labels are no more maintained. The definition of rotation is below:

$$[x_2, y_2, 1] = [x_1, y_1, 1] \times \begin{bmatrix} \cos(r) & -\sin(r) & 0 \\ \sin(r) & \cos(r) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

Fig. 6(g) displays the raw rice image with blue grid lines. Fig. 6(h and i) show the rotation DA results with r of 20° and -20°, respectively.

Shear DA relocates every pixel in a preset direction by an amount, which is proportional to its signed distance from the line passing through the origin and parallel to that direction [150]. There are two commonly available shear mapping: horizontal shear (HS) and vertical shear (VS). The HS is defined as

$$[x_2, y_2, 1] = [x_1, y_1, 1] \times \begin{bmatrix} 1 & 0 & 0 \\ z_{hs} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

where z_{hs} stands for the HS factor. VS is defined as:

$$[x_2, y_2, 1] = [x_1, y_1, 1] \times \begin{bmatrix} 1 & z_{vs} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

where z_{vs} is the VS factor. Fig. 6(j and k) show the corresponding cognate HS and VS outcomes.

In the traditional food category classification field, cropping DA is an effectual utensil to take patches out from a large or mixed-size image. The following classification AI models are operated on the patches $\{P_1, P_2, \ldots\}$ other than the raw image itself I. Specifically, cropping DA takes out the patches with a preset size out of I [151,152].

Fig. 6(l) displays the crop DA schematic, in which two squares (blue and green) mark out the areas to be cropped. Fig. 6(m and n) display the patches (P_1, P_2) within the blue and green squares.

4.3.2. Translation and elastic distortion

A translation moves every pixel by the same distance along the same angle [153]. The translation DA is generally exercised in food category classification. The food naturally gathers food images in nearly flawlessly centered places, necessitating the following classification algorithms to predict similarly centered food images.

By means of the translation DA [154], the dataset is added by other translated images (food not centered). The trained classifier is more reliable and can predict capably on other images in which foods are not centered.

If images are shifted outwards of the raw image coordinate, there are missing values v_m. Therefore, it is necessary to fill in v_m with either a constant or random noise n_{rand}.

$$v_m = \begin{cases} 0 \\ 255 \\ n_{rand} \end{cases}, \tag{6}$$

where 0 and 255 stand for black and white, respectively.

Fig. 6(o) displays an image from which the translation DA can displace it along various directions. Fig. 6 (p and q) show two translation DA results with the translation parameters of [30, 40] and [-40, -30], respectively.

The disparity between translation and cropping is (i) cropping DA reduces the image size, and (ii) translation DA preserves the image size. If the raw image size is $[W_0, H_0]$, then the size after cropping is $[W_{crop}, H_{crop}]$ and the size after translation is $[W_{tran}, H_{tran}]$, we have

$$\begin{cases} W_{tran} = W_0, H_{tran} = H_0 \\ W_{crop} < W_0, H_{crop} < H_0 \end{cases}. \tag{7}$$

Elastic distortion uses the displacement field (DF) [155] to generate the distorted image. Fig. 6(r and t) show two DFs [156] and Fig. 6(s and u) show the corresponding two distorted images. It is easily spotted that the brim of the bowl in the original image is straight, but the brims of the bowl in the distorted images are now zigzag.

4.4. Noise injection

Noise injection stands for adding noise to the training images to make the training of the classification model more robust. Scholars often use the probability density function (PDF) to describe the distribution of the noise.

4.4.1. Gaussian noise

The Gaussian noise [157] model is frequently utilized to replicate thermal noise. The PDF of the univariate Gaussian noise G is defined:

$$p_G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < +\infty, \tag{8}$$

where μ symbolizes the mean and σ^2 symbolizes the variance [158]. For input training images, x means the gray-level of added noise [159]. The noise-injected image is written as $y = I + G$, where I is the original image, and $G \sim p_G(x)$ is the added Gaussian noise image. Fig. 7(a) displays the original gammon image. Fig. 7(b-d) displays the Gaussian noise-injected image with $\sigma^2 = 0.01$, 0.02, and 0.05, respectively.

4.4.2. Salt-and-pepper, speckle, and Poisson noises

The salt-and-pepper noise (SPN) [160] stays another category of image noise frequently used in DA. The operation of salt-and-pepper is two steps. First, it assigns each pixel a random probability value $p(i,j)$ from a uniform distribution of the interval $[0, 1]$. The noise injected image y is defined as

$$y(i,j) = \begin{cases} N_p & p(i,j) < \frac{\gamma}{2} \\ N_s & \frac{\gamma}{2} \le p(i,j) < \gamma \\ I(i,j) & \gamma \le p(i,j) \end{cases}, \tag{9}$$

where $N_s = 255$ or the maximum grayscale value, standing for the salt noise, and $N_p = 0$ denoting the pepper noise. γ stands for the noisy density, a factor symbolizing how many percentages of all pixels are contaminated with SPN.

The SPN [161] does not influence the entire image but alters parts of the original pixels. Fig. 7(e-g) illustrates an example of salt-and-pepper noise injection with the noisy density values assigned as 0.05, 0.1, and 0.2, respectively.

Speckle noise (SN) is multiplicative [162] and frequently triggered by poor information channels. Since it is multiplicative with the raw images, the noise takes place with the image where pixel values $I(i,j)$ are nonzero and vanishes where the values of $I(i,j)$ are zero. SN is modelled as

$$y = I + N_m I + N_a, \tag{10}$$

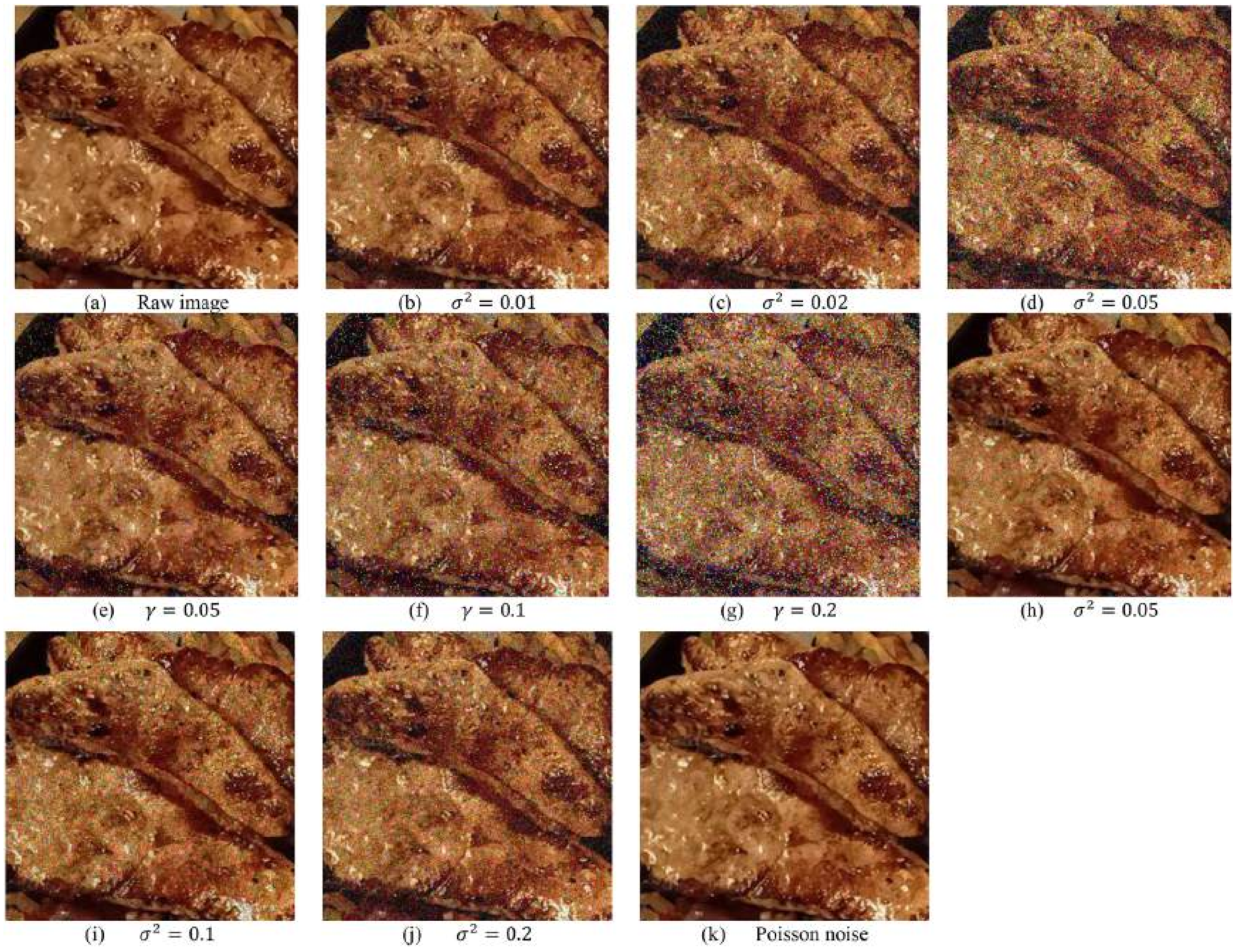


Fig. 7. Illustration of noise injection. (a) raw image, (b-d) Gaussian noise injection, (e-g) salt-and-pepper noise injection, (h-j) speckle-noise injection, (k) Poisson noise injection.

where I stands for the noise-free raw image. y is the noise-injected image. N_m and N_a mean the multiplicative noise and additive noise, respectively [163]. If there is no additive noise, then Eq. (10) degrades to $y = I + N_m I$. Suppose the variance of the speckle noise is denoted by σ^2, then three examples of speckle noise with the values of 0.05, 0.1, and 0.2 are given in Fig. 7(h-j).

Poisson noise (PN) obeys the Poisson process $P(\lambda)$.

$$P(\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}. \tag{11}$$

The mean of $P(\lambda)$ is λ. PN is caused by the uncertainty [164] related to the measurement of light, intrinsic to (i) the quantized mechanism of light and (ii) the independence of photon recognitions [165]. For example, if a pixel of the input image has a value of 20, then the value of the related output pixel is produced from a Poisson distribution $P(\lambda)$ with $\lambda = 20$. An example of PN injected image is show in Fig. 7(k).

4.5. Photometric transforms

Photometric transforms, identified as color space transform [166], are to operate the grayscale values of a grayscale image or the RGB values of a color image. The straightforward photometric transform technique is adding or subtracting a constant value, C_1 or C_2, to increase or decrease the image's pixel values, making it lighter or dimmer. The equation is:

$$\begin{cases} y_1 = I + C_1 \\ y_2 = I - C_2 \end{cases}, \tag{12}$$

where C_1 means a constant value to be added while C_2 means a constant value to be subtracted. y_1 and y_2 represent the brightened and darkened images, respectively.

Fig. 8(a) shows a garlic bread image, Fig. 8(b) presents the brightened image by adding $C_1 = 30$ to the original image, and Fig. 8(c) presents the darkened image by subtracting $C_2 = 30$ from the raw image.

4.5.1. Gamma correction and color jittering

Gamma correction [167] is a nonlinear process regulating the raw image's gray-scale or RGB values. Using the power-law formula, we can define GC as:

$$\begin{cases} y_o = C \times y_i^{\gamma} \\ y_i \in [0, 1] \\ y_o \in [0, 1] \end{cases}, \tag{13}$$

where y_i and y_o represent the input and output grayscale values, assuming y_i and y_o are normalized to the extent of $[0, 1]$. Furthermore, $C = 1$ helps to reserve the grayscale extent.

There are two vital thoughts: (i) Eq. (13) turns to the gamma compression $\mathbb{D}^{GC}_{com}$ when $\gamma < 1$; and (ii) Eq. (13) turns to the gamma

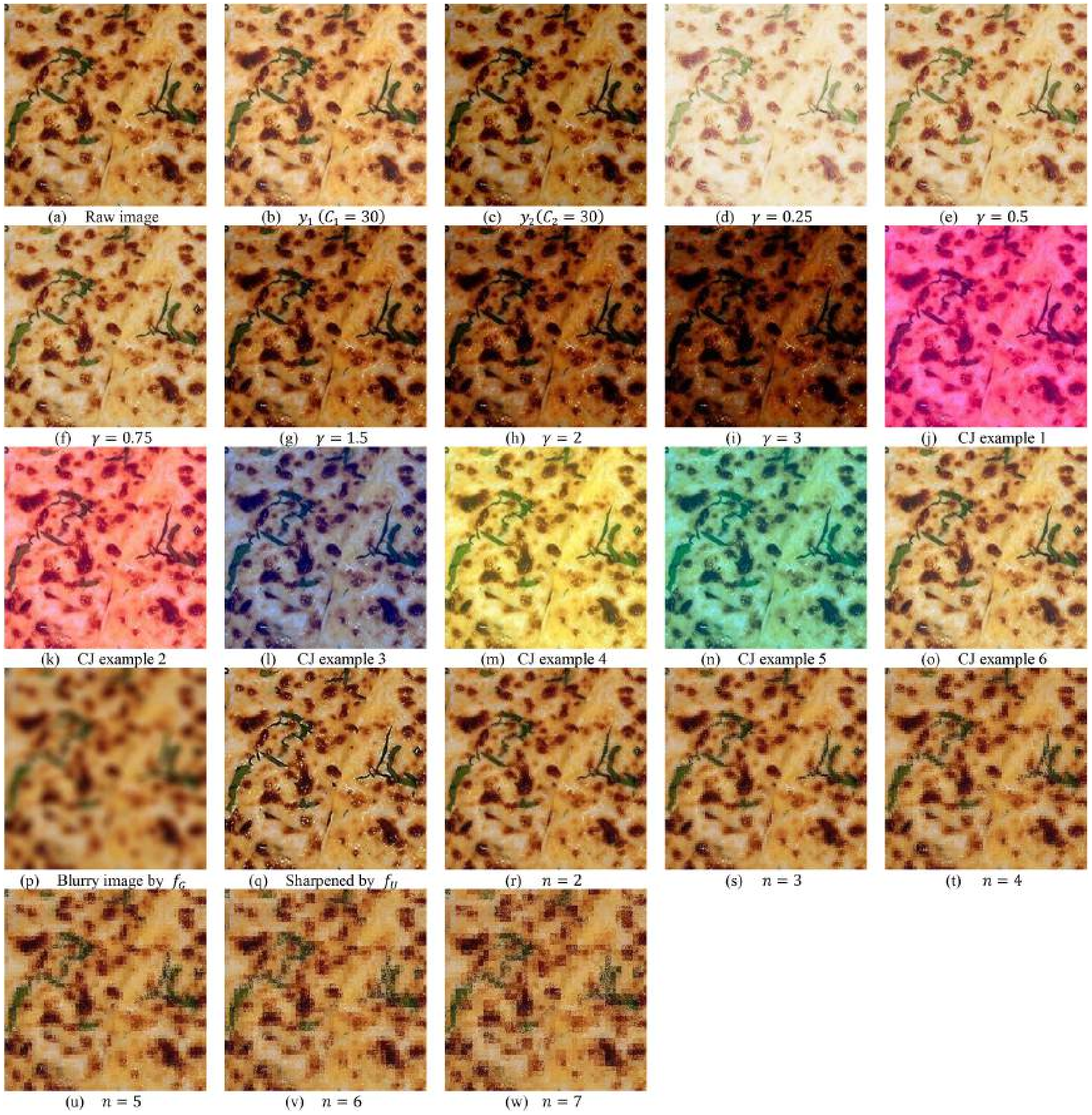


Fig. 8. Illustration of photometric transforms. (a) raw image, (b and c) simple photometric transform, (d-i) Gamma correction, (j-o) six OJ examples, (p) blurry image, (q) sharpened image, (r-w) PatchShuffle images.

expansion $\mathbb{D}^{GC}_{exp}$ when $\gamma > 1$ [168]. Fig. 8(d-f) present three results of $\mathbb{D}^{GC}_{com}$, i.e., $\gamma = (0.25, 0, 5, 0.75)$, respectively. Fig. 8(g-i) present three other results of $\mathbb{D}^{GC}_{exp}$ with $\gamma = (1.5, 2, 3)$, respectively.

Color jittering (CJ) changes the color values in the raw image by adding or subtracting a random value for each channel [169]. Another way is to make changes to the brightness, contrast, saturation, and hue channels. The advantage of CJ is it brings in randomness alterations to all the RGB channels; thus, CJ aids the manufacture of bogus color images.

Suppose the raw image is I, whose range is $[0, 1]$. We randomly generate three random numbers $(c_r, c_g, c_b) \in [-R, +R]$. The R is the maximum value of random numbers. Usually, it can be set as $R = 0.2$. The CJ operation is defined as:

$$\begin{cases} y_r = f_N(I_r + c_r) \\ y_g = f_N(I_g + c_g) \\ y_b = f_N(I_b + c_b) \end{cases}, \tag{14}$$

where the subscript (r, g, b) stand for the red, green, and blue channels. f_N is the normalization function to make sure the value is within the range of $[0, 1]$. That is

$$f_N(x) = \begin{cases} 0 & x < 0 \\ x & 0 \le x \le 1 \\ 1 & x > 1 \end{cases}. \tag{15}$$

Fig. 8(j-o) displays six CJ results of the original image in Fig. 8(a).

4.5.2. Sharpening, blurring, and PatchShuffle transform

Kernel filters are employed to sharpen and blur raw images as DA methods. The kernel filter method slides an $n \times n$ kernel along the raw image with either a Gaussian blur [170] filter f_G or an unsharp masking [171] filter f_U. Unsharp masking [172] is an image sharpening method. Its name derives from that the method uses an unsharp negative image to create the final sharpened image. The definition of kernel filter method is:

$$\begin{cases} y_G = I \otimes f_G \\ y_U = I \otimes f_U \end{cases}, \tag{16}$$

where y_G and y_U stand for the blurry and sharpened images. Obviously, f_G produces a blurry image, whereas f_U produces a sharpened image. Fig. 8(p and q) display y_G and y_U obtained by the filters f_G and f_U, respectively.

Instinctively, y_G helps the following classifiers better fight back the blur attack (motion, Gaussian, average, etc.) through the test. y_U introduces more contrast and edge minutiae for food category classifications. Both f_G and f_U kernels are popular in DA.

Kang, G. et al. (2017) [173] proposed a novel PatchShuffle transform (PST) technique. Within each minibatch, each image I is divided into nonoverlapped patches $\{b_{ij}\}$, and each patch b_{ij} undertakes a transformation so that pixels of the patch b_{ij} are shuffled. The authors have steered tests with various filter sizes n and various swapping probabilities p_{swap}.

Assume the raw image is I with a size of $N \times N$. Let I be divided into a patch matrix B with nonoverlapped patches $B = \{b_{ij}\}$,

$$I \mapsto B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1,N/n} \\ b_{21} & b_{22} & \cdots & b_{2,N/n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N/n,1} & b_{N/n,2} & \cdots & b_{N/n,N/n} \end{bmatrix}, \tag{17}$$

where b_{ij} stands for the patch at i-th row and j-th column of B.

The PST works on each patch as follows:

$$c_{ij} = p^r_{ij} \times b_{ij} \times p^c_{ij}, \tag{18}$$

where c_{ij} denotes the transformed patch, p^r_{ij} and p^c_{ij} stand for the row and column permutation matrixes, respectively [174]. The authors' simulation results exhibited that the optimal hyperparameter is the combination of $n = 2$ and $p_{swap} = 0.05$. Hence, the final PatchShuffled image is $C = \{c_{ij}\}$. Fig. 8(r-w) displays the PST results with $n = 2, \cdots, 7$. Kang, G. et al. (2017) [173] report PST can be harnessed on not only raw input images but also feature maps.

4.6. Image mixing

The above transformation techniques (geometric transforms, noise injection, and photometric transforms) are all single-image DA methods. This section discusses some new techniques for handling more than one image.

4.6.1. SamplePairing and mixup

Inoue, H. (2018) [175] proposed the SamplePairing method, synthesizing new training data from one image by randomly overlaying another image selected from the training set. Shortly, SamplePairing calculates the pixel-wise average of two patches or images.

Assuming an image D of Class 1 (C_1), and another randomly chosen image E of Class 2 (C_2). SamplePairing DA firstly produces two patches (F, G) from the image pair (D, E) by (i) random cropping and (ii) random horizontal flipping, as shown in Fig. 9(a).

$$\begin{cases} D \xrightarrow{Patch} F \\ E \xrightarrow{Patch} G \end{cases} \tag{19}$$

The Class 2 label (C_2) is cast off. Afterward, the patch pair (F, G) is blended to produce the mixed patch H by pixel-wisely averaging the intensities of the patch pair.

$$\underbrace{H(i,j)}_{C_1} = \left[\underbrace{F(i,j)}_{C_1} + \underbrace{G(i,j)}_{C_2} \right] \Big/ 2. \tag{20}$$

The mixed patch H is then employed for network training. Inoue, H. (2018) [175] stated that SamplePairing is able to produce M^2 new data from M-sized sample dataset [175]. Fig. 9(b-d) shows the image pair of the rice and the dam. Fig. 9(d) displays the synthesized image with the label "Rice".

Zhang, H. et al. (2018) [176] presented a data-agnostic DA technique, mixup. In their method, the parameter $\omega \in [0, 1]$ is initiated, meanwhile one-hot encoding is employed to exploit the information of classes of both images. Assuming (A, B) stands for the two randomly chosen images, the mixup image, and label C and t_C are obtained as:

$$\begin{cases} C = \omega \times A + (1 - \omega) \times B \\ t_C = \omega \times t_A + (1 - \omega) \times t_B \end{cases}, \tag{21}$$

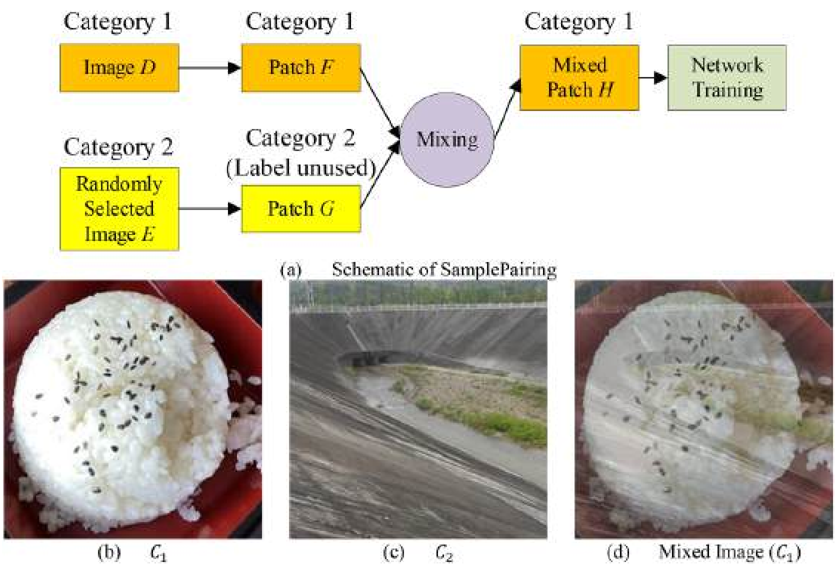


Fig. 9. Illustration of SamplePairing. (a) Schematic of SamplePairing, (b-d) A SamplePairing illustration of mixing the image pair of C_1 = Rice and C_2 = Dam.

where (t_A, t_B) are labels of Sample A and Sample B. Concisely, mixup stretches out S_{train} by linearly interpolating two randomly chosen samples (A,B). Fig. 10(a and b) display two randomly chosen samples: Rice and Swan. Fig. 10(c-i) show the mixup outcome with $\omega = 0.2, \cdots, 0.8$.

4.6.2. Nonlinear mixing and random erasing

Summers, C. et al. (2019) [177] expanded linear combination to a nonlinear mixing method. Supposing $\theta \in [0,1]$ is a random hyperparameter, the vertical concatenation (VC) mingles the top θ fraction of the image A and the bottom $(1-\theta)$ fraction of the image B, rather than pixelwise average. Assume (W,H) are the width and height of the image A, (w,h) are the width and height index, respectively, the VC output is defined:

$$C^{VC}(w,h) = \begin{cases} A(w,h) & h \leq \theta H \\ B(w,h) & \text{otherwise} \end{cases}. \tag{22}$$

The horizontal concatenation (HC) is defined:

$$C^{HC}(w,h) = \begin{cases} A(w,h) & w \leq \theta W \\ B(w,h) & \text{otherwise} \end{cases}. \tag{23}$$

Mixed concatenation (MC) combines HC and VC. That is, suppose there are $0 \leq \theta_1, \theta_2 \leq 1$ two random hyperparameters, then

$$C^{MC}(w,h) = \begin{cases} A(w,h) & h \leq \theta_1 H \wedge w \leq \theta_2 W \\ B(w,h) & h \leq \theta_1 H \wedge w > \theta_2 W \\ B(w,h) & h > \theta_1 H \wedge w \leq \theta_2 W \\ A(w,h) & h > \theta_1 H \wedge w > \theta_2 W \end{cases}. \tag{24}$$

Random column interval (RCI) chooses a random column interval. That interval part of the image A is substituted with the corresponding part of image B. Random row interval (RRI) carries out the same operation along the row direction. The random row (RR) chooses each row at random from either image A or B. The RR technique is considered as a higher frequency of VC. In the same way, we are able to infer the random column (RC) technique. Random square (RS) cuts out a random square from image A, and fills in the square with the cognate region from image B. Random pixel (RP) samples every pixel independently from images A and B. Fig. 10(j-r) display the results of nine nonlinear mixing DA techniques.

Zhong, Z. et al. (2020) [178] proposed the random erasing (RE) technique that randomly picks a rectangle region and deletes its pixels, and fills in with random values (RVs). The RE is suitable to win image recognition jobs related to occlusion, meaning several patches of the objects are clogged. RE compels the following AI models to learn global features from the unclogged patches.

Practically, RE randomly chooses an $n \times m$ patch p from an image A. It then fills the patch p with either 0 s, 255 s, or mean pixel value p_m, or RVs r. The patch fill (PF) procedure is written as:

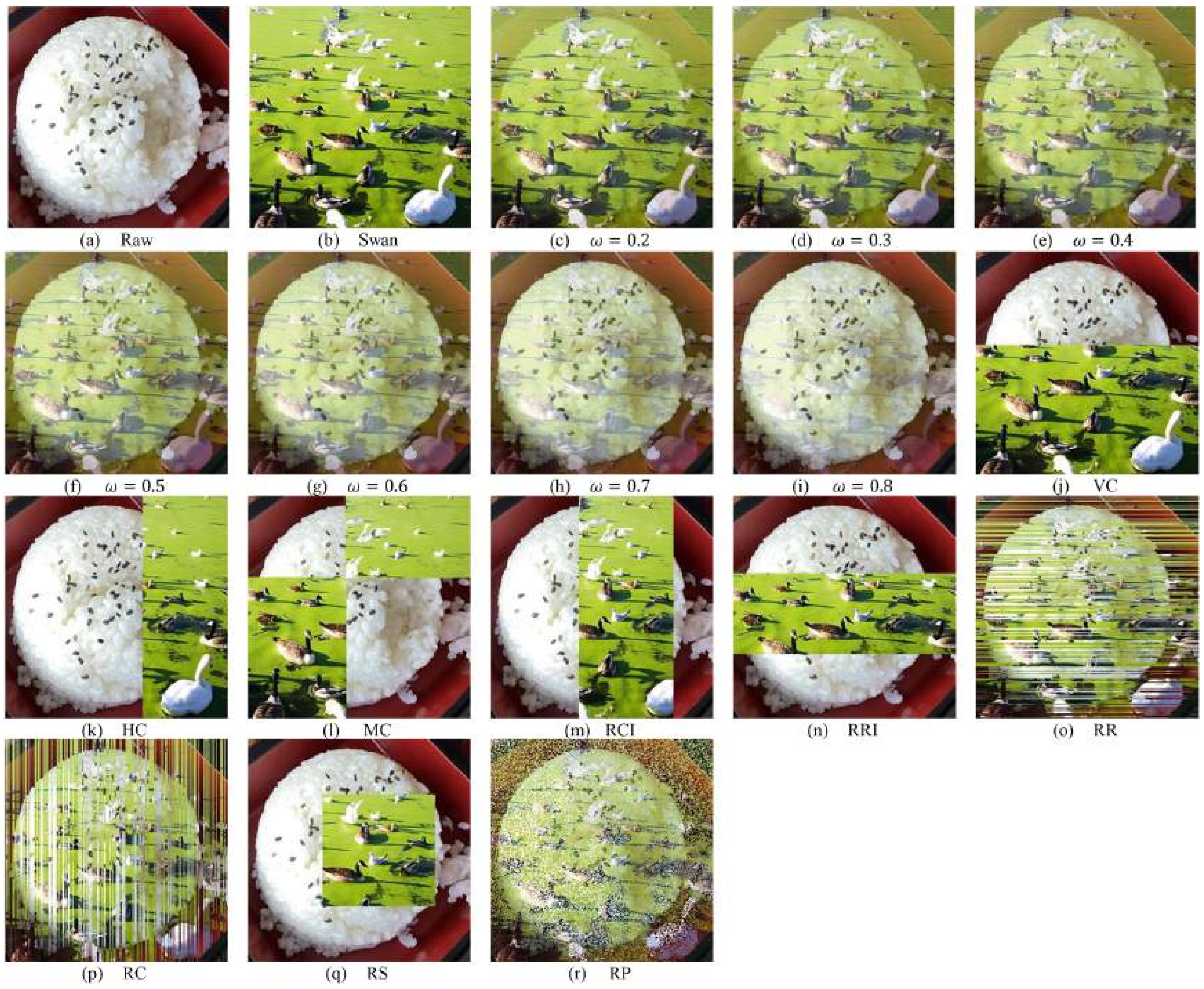


Fig. 10. Illustration of mixup and nonlinear mixing. (a and b) two raw images, (c-i) mixup results, (j-r) nonlinear mixing results.

$$p(w,h) = ^{\mathrm{PF}} \begin{cases} 0 \\ 255 \\ p_m \\ r(w,h) \end{cases}, \tag{25}$$

where 0 and 255 stand for black and white. Two parameters of RE are (i) the selection of PF and (ii) the size of the patches. The best PF method has been verified to be RVs. Fig. 11(a) displays the rice image. Fig. 11(b-c) display the two RE examples, which the observer can recognize it is a rice image.

RE is not permanently “safe”. In digit recognition jobs, if the top patch is erased, then figure “7” looks like “1”. RE is not safe in more fine-grained jobs. For example, (i) tumor grade classification, the RE could clog the tumor. (ii) vehicle identification, the AI model’s performance is impaired if RE blocks the brand part of vehicles. Therefore, some guarantee-goal schemes are implemented to pledge the “safety” of the RE-augmented dataset.

4.7. Deep learning-based DA: adversarial training and generative adversarial network

Adversarial training (AT) endeavors to trick AI models with deceitful data [179,180]. The adversarial attack contains a rival network, which gains knowledge of deceitful DA images, which triggers wrong classifications in its rival classification network [181,182]. Supposing there is an image A of class C_1, the user adds a minor amount of noise to it $\varepsilon \times N$, where ε is a minor value, usually $\varepsilon < 0.01$. The noise is fabricated intentionally. The summation result B falls into another class C_2.

$$\underbrace{A}_{C_1} + \varepsilon \times N = \underbrace{B}_{C_2}. \tag{26}$$

Suppose s_c stands for the confidence score, Fig. 12(a) displays an image A marked as “bread” with $s_c = 80.1\%$. After the perturbation noise, shown in Fig. 12(b), by DeepFool [183] is added, the noised image B is labelled by AI models as “Steak” with $s_c = 99.9\%$, as shown in Fig. 12(c).

The AT is effective in solving weak points in conventional AI models. Therefore, the trained AI models are more reliable and resilient to attackers. AT cannot improve the test performance. However, it improves the performance of AI models working on adversarial examples, viz., enhancing the safety and sturdiness of trained AI models.

The generative adversarial network (GAN) [184,185] entails two neural networks (NNs) competing in a zero-sum contest, in which one NN’s gain is the other NN’s loss. There are numerous generative models that presently work; however, GAN provides prominent functioning in terms of both quality and computation promptness [186,187]. Two natural examples of GAN are (i) a predator and prey and (ii) a rivalry between police (Discriminator **D**) and a counterfeiter (Generator **G**).

In the second example, both parties are advancing their performances; hence, the counterfeiter knows how to create travel documents that are arduous to identify by the police. This procedure is shown in Fig. 12(d). The triumph of the **G** renders it potent for generative modeling. Thus, GANs are verified to be effectual in DA. Goodfellow et al. (2014) [188] presented the earliest GAN based on multilayer perceptron to manage MNIST handwritten digit-image recognition. The width of its image is $W = 28$, so it contains $W \times W \times 1 = W^2 = 784$ pixels. At present, the pictures in food datasets are of finer resolution and more intricate than those in the MNIST dataset. Therefore, several variants of GANs are nowadays universally operated in the food category classification field.

5. Hand-crafted features

5.1. Color histogram

Color, texture, and shape are the most widely used visual features in image recognition tasks. Color is one of the most important features of food recognition [189]. Especially in specific situations like pre-processed foods with the same shape (such as shredded Potatoes and Carrots). It is almost impossible to distinguish foods without color information in such scenarios.

The color histogram is a common representation of color information in images. Digital colorful images usually consist of three channels: R, G, and B [190]. Pixel values of each channel represent the levels of one of three colors: red, green, or blue. Based on that, color histograms are three separate histograms representing the corresponding channels’ brightness distribution [191].

The advantage of the color histogram is its robustness to image rotation and translation, and after normalization, the color histograms are also unaffected by changes in image scale [192]. However, the color histogram also has some drawbacks. Firstly, the color histogram of an image describes the statistical characteristics of the color of the image and ignores information about the spatial distribution of the pixels [193]. Two images with a small difference in color histogram but a large difference in the spatial distribution of colors may describe very different things. In addition, the image color quantization process may quantize visually different colors into the same color interval, or visually similar colors may be quantized into different intervals [194]. Moreover, the feature vector obtained from the color histogram usually has high dimensionality [195]. In response to these problems, many improved methods of histogram algorithms have been proposed in recent years.

- By using the main colors of the image as samples to construct a color histogram of the image and ignoring those color intervals with small values, the improved histogram is less sensitive to image noise.
- The local cumulative histogram method and the method of constructing fuzzy histograms are used to improve the efficiency of constructing color histograms.

Due to the specific scenarios that may exist for food classification tasks and the advantages of the color histogram as a color feature representation, many studies on food classification have used color histograms as features to represent the color information of food images. Fig. 13(b-d) illustrate an example of univariate RGB color histograms

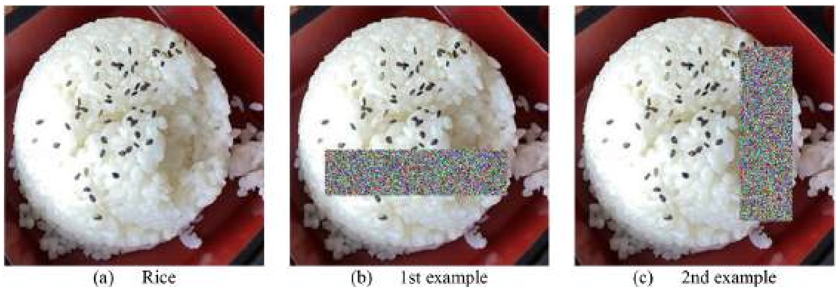


Fig. 11. Examples of RE.

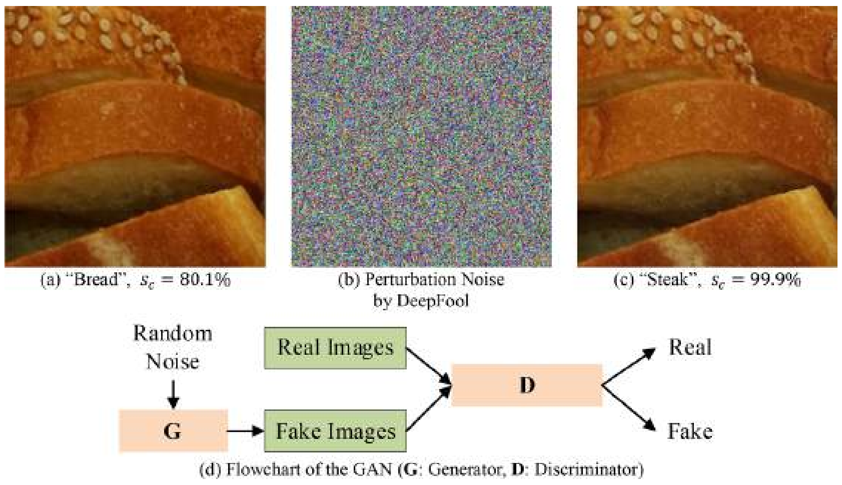

(a) "Bread", s_c = 80.1%
(b) Perturbation Noise by DeepFool
(c) "Steak", s_c = 99.9%
Random Noise
G
Real Images
Fake Images
D
Real
Fake
(d) Flowchart of the GAN (G: Generator, D: Discriminator)

Fig. 12. Illustration of adversarial training and GAN.

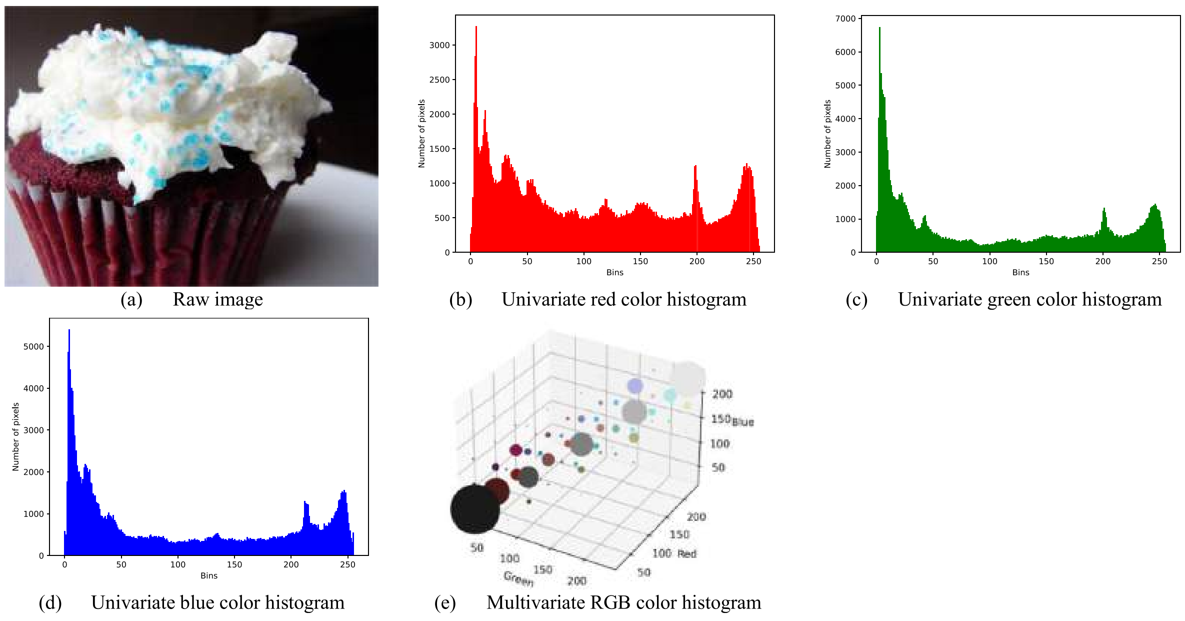


Fig. 13. Univariate and multivariate RGB color histograms.

extracted from a cupcake image sample, Fig. 13(a), from the Food101 dataset.

Although univariate RGB color histograms can indicate the distribution of colors in the image by counting red, green, and blue colors separately, this approach cannot capture the correlation among the red, green, and blue colors. To be specific, when these three colors are mixed additively at different proportions, more colors are generated. For example, when blue is zero, equal proportions of red and green give a pure yellow, more red than green gives an orange-yellow color, and so on.

Univariate RGB color histograms can only reflect the distribution of red, green, and blue colors but cannot reflect the distribution of colors generated by mixing these three colors. Therefore, there is an alternative type of color histogram called the multivariate color histogram. It counts the distribution of different colors in the image when the red, green, and blue colors are mixed to give a complete picture of an image's color distribution. Fig. 13(e) illustrates an example of the multivariate RGB color histogram with 6 $\times$ 6 $\times$ 6 bins. Each bin is a 3D cube, visualized as a circular marker corresponding to a color generated by mixing different proportions of red, green, and blue colors. The radius of each marker is proportional to the histogram count of the corresponding color.

5.2. *Histogram of oriented gradient*

The directional density distribution of gradients or edges can well describe the appearance and shape of a local target in an image [196]. The Histogram of Oriented Gradients (HOG) is a feature descriptor widely used in vision-based tasks, which consists of a histogram of the local gradient directions of the image [197]. HOG operates locally on the image and is more robust to image geometry and optics changes [198]. For better robustness against illumination changes and shadows, the local histogram can be normalized by contrast over a larger image area (called a range or block). Specifically, the density of each histogram in the corresponding block is calculated, and then all cells in the block are normalized according to this density. There are five main steps in the process of extracting HOG from an image, as shown below:

- In order to reduce the effect of illumination factors, the global image needs to be normalized and equalized. It is usually done using gamma correction or calculating each pixel value's square root or logarithm for each color channel. Gamma correction can effectively reduce local shadows and lighting variations in the image. As the color information is of little use to the HOG calculation process, the image is usually first converted to a grayscale image. The gamma correction of the raw image I_R is defined as: $I(x,y) = I_R(x,y)^\gamma$, where (x,y) are coordinates of a pixel, and γ is a customizable number. When $\gamma > 1$, the contrast of the high grey areas of the image is enhanced, and the visual effect is a darkening of the image. When $\gamma < 1$, the contrast is enhanced in the lower grey areas of the image, and the visual effect is brightening.

- The gradients in the horizontal and vertical directions of the image are calculated, and the gradient direction value is calculated for each pixel position to capture the contour information and further attenuate the effect of light intensity. The gradients calculation process for the pixel $I(x,y)$ of input image I is shown in Eq. (27).

$$\begin{aligned} g_h(x,y) &= I(x+1,y) - I(x-1,y), \\ g_v(x,y) &= I(x,y+1) - I(x,y-1), \end{aligned} \tag{27}$$

where $g_h(x,y)$ represents the horizontal gradient and $g_v(x,y)$ is the vertical gradient of the pixel $I(x,y)$ in input image I, respectively. On this basis, the gradient amplitude ∇g and the gradient direction value α of the pixel (x,y) can be calculated by Eq. (28).

$$\begin{aligned} \nabla g(x,y) &= \sqrt{g_h(x,y)^2 + g_v(x,y)^2}, \\ \alpha(x,y) &= \tan^{-1}\left(\frac{g_v(x,y)}{g_h(x,y)}\right). \end{aligned} \tag{28}$$

- The image is divided into small, non-overlapping spatial regions (cells). Then a gradient direction histogram is computed in each cell to produce an encoding sensitive to local image content while remaining weakly sensitive to small changes in position and appearance. Also, the gradient angle range is divided into a fixed number of bins. Based on these bins, the gradient direction histogram is constructed for each cell using the gradient size as weights.
- Calculate the normalization. This step combines several cells into a larger block, and the HOG of the block is obtained by concatenating the histograms of all the cells in the block. Here, the blocks may overlap, and the local domain information can be used effectively. Due to local illumination variations, the gradient intensity varies over a very large range. Calculating normalization over a larger spatial area (block) allows further compression of illumination, shadows, and edges to provide better invariance. Common normalization methods are L1 normalization (L1-norm), square root of L1 normalization (L1-sqrt), and L2 normalization (L2-norm), which can be defined as Eq. (29).

$$\begin{aligned} v_{\text{L1-norm}} &= \frac{v}{||v||_1 + \varepsilon}, \\ v_{\text{L1-sqrt}} &= \sqrt{\frac{v}{||v||_1 + \varepsilon}}, \\ v_{\text{L2-norm}} &= \frac{v}{\sqrt{||v||_2^2 + \varepsilon^2}}, \end{aligned} \tag{29}$$

where ε is a small constant.

- Collect the HOGs from all blocks and combine these HOGs to form feature vectors.

HOG is often used as a local texture feature of an image, which can effectively display information about the edges, contours, and shapes of objects in a local area of the image. Fig. 14 shows an example of HOG.

5.3. Gabor features

In image processing, the Gabor features can describe the texture information of images. It uses a Gabor filter to superimpose a window function on the frequency domain signal to describe the signal's local frequency information. Since the frequency and direction of the Gabor filter are similar to that of the human visual system [199], the Gabor feature is widely considered suitable for texture representation and discrimination. In the spatial domain, a two-dimensional Gabor filter is a two-dimensional Gaussian function modulated by the complex sinusoid, which can be defined as the product of the complex sinusoid and a two-dimensional Gaussian function.

The prerequisite for extracting Gabor features is converting a two-dimensional image signal from the spatial domain to the frequency domain utilizing the Fourier transform (FT). The essence of the FT is to transform any function into a combination of sine waves of different frequencies [200]. In this way, signals in the spatial domain, such as image signals, can be converted to the frequency domain. Furthermore, the superposition of multiple waves in the spatial domain is represented in the frequency domain as a number of scattered points. Thus, problems that are complex in the spatial domain become relatively simple in the frequency domain. It makes the FT an important tool that can be used in image processing. The two-dimensional FT can be defined as Eq. (30).

$$\widehat{f}(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-i2\pi(ux+vy)}\mathrm{d}x\mathrm{d}y, \tag{30}$$

where (x,y) refers to the spatial domain coordinates and (u,v) to the corresponding frequency domain coordinates. The complex sinusoid can be defined as $s(x,y) = e^{i(2\pi(ux+vy)+P)}$, where P represents the phase of the sinusoid.

The two-dimensional Gaussian function with the consideration of rotation is given by Eq. (31).

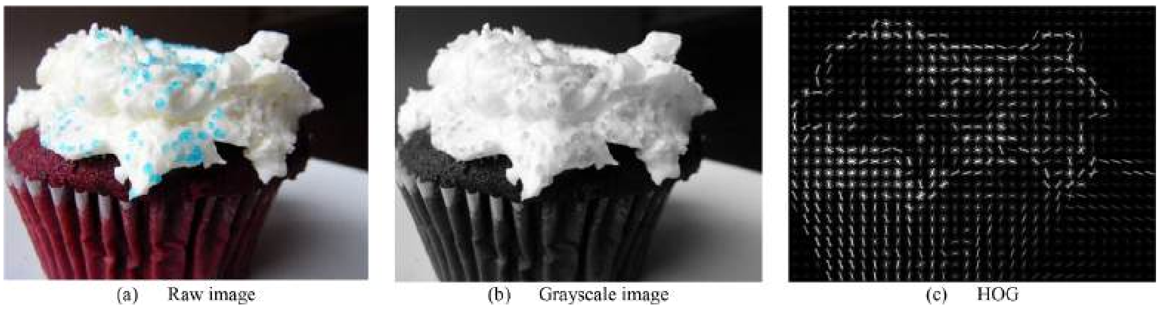


Fig. 14. An example of a histogram of oriented gradient (HOG). As HOG considers gradient information representing edge and shape characteristics, the color in the image is not important for calculating HOG. So, we first convert the raw image (a) to a grey-scale (b) image and then calculate the HOG (c).

$$\omega_r(x,y) = Ke^{-\pi\left(\frac{(x-x_0)_r^2}{\sigma_x^2}+\frac{(y-y_0)_r^2}{\sigma_y^2}\right)}, \tag{31}$$

where σ_x and σ_y are the scaling parameters in the two directions. $(x_0,\ y_0)$ is the centroid of the Gaussian function (peak). K is a constant, representing the scale of the magnitude of the Gaussian function. r represents the rotation operation. $(x-x_0)_r$ and $(y-y_0)_r$ are calculated as shown in Eq.s (32).

$$\begin{aligned}(x-x_0)_r &= (x-x_0)cos\theta + (y-y_0)sin\theta,\\ (y-y_0)_r &= -(x-x_0)sin\theta + (y-y_0)cos\theta,\end{aligned} \tag{32}$$

where θ is the rotation degree of the Gaussian function. The two-dimensional Gabor filter is calculated by multiplying the Gaussian function with the complex sinusoid, as shown in Eq. (33).

$$G(x,y) = \omega_r(x,y)s(x,y) = Ke^{-\pi\left(\frac{(x-x_0)_r^2}{\sigma_x^2}+\frac{(y-y_0)_r^2}{\sigma_y^2}\right)} e^{i(2\pi(ux+vy)+P)}. \tag{33}$$

The frequency distribution of each pixel in the image and its vicinity can be obtained by convolving the image with a Gabor filter of different frequencies. Since texture features are usually frequency-dependent, the Gabor filter is often used to extract texture features.

Gabor filters are often used to extract texture features in images. Fig. 15 shows examples of applying different Gabor filters to the cup cake image sample.

5.4. Scale-invariant feature transform

A central problem in image recognition tasks is recognizing the same target at different resolutions, under different illumination, in different orientations, etc. The usual solutions to this problem are based on corner or edge recognition, but these methods are often less resilient to environmental changes. Scale Invariant Feature Transform (SIFT) can extract local features from images.

SIFT was first introduced by Lowe, D. G. (1999) [201] and refined by Lowe, D. G. (2004) [202]. It extracts the location, scale, and orientation invariants of key points detected at a spatial scale. SIFT features are based on corners, edges, bright spots in dark regions, dark spots in light regions, etc. Therefore, they are, to some extent, robust to illumination changes or noise and can remain undistorted by scale, orientation, and light changes. They are also robust to changes in noise [203]. The SIFT feature extraction process can be divided into four steps: (1) scale-space extrema detection, (2) key point localization, (3) orientation assignment, and (4) key point description.

SIFT searches for key points on different scale spaces obtained by 2D Gaussian functions and identifies potential interest points invariant to scale and rotation based on the difference-of-Gaussian (DoG). Eq. (34) defines the scale space, $L(x,y,\sigma_x,\ \sigma_y)$, which is calculated by convolving a simple 2D Gaussian function ω with the input image I_R.

$$L(x,y,\sigma) = \omega(x,y,\sigma) * I(x,y), \tag{34}$$

where $*$ stands for the convolution operator. ω can be defined as Eq. (35).

$$\omega(x,y,\ \sigma) = \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{35}$$

A series of scale-space images can be obtained by repeatedly convolving the input image with a two-dimensional Gaussian variable scale function. The adjacent scale-space images can then be subtracted to calculate the DoG using Eq. (36).

$$D(x,y,\sigma) = (\omega(x,y,k\sigma) - \omega(x,y,\sigma)) * I(x,y), \tag{36}$$

where k is a constant multiplicative factor. A pixel in a DoG image is compared to its 26 neighboring pixels (8 neighboring pixels in the same DoG image and 18 neighboring pixels in two DoG images at the adjacent scale). If the current pixel is the largest or smallest of all neighboring pixels, the pixel is selected as a candidate key point.

Once a candidate key point is identified, the location and scale of the key point can be determined by fitting to its nearby data, and the key point can be selected based on its contrast and edge response (key points with low contrast or not well localized along an edge can be rejected).

The assignment of orientations is primarily based on the image's local gradient direction, with one or more directions assigned to each key point location. Eqs. (37) and (38) show how the gradient magnitude and orientation are calculated, respectively.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{37}$$

where $L(x,y)$ is the scale of the image sample, and $m(x,y)$ is the gradient magnitude of an image sample at the $L(x,y)$ scale.

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right), \tag{38}$$

where $\theta(x,y)$ is the gradient orientation of an image sample at the $L(x,y)$ scale.

Ultimately, the image's local gradients are measured at chosen scales around each key point in the neighborhood. The scale, orientation, and location are transferred to a representation that is robust to illumination or viewpoint change to some extent by assigning them to key points. Then Feature description for local image regions can be calculated by sampling the local image intensities around key points at an appropriate scale and then using the normalized correlation to match these regions.

The SIFT features rely on local key points on the target item. Therefore, they are independent of the direction and size of the influence (noise, illumination change, etc.). Fig. 16 shows the key points extracted from the raw and transformed images and matches the key points in the raw image and key points in the transformed image. It can be seen that the key points can be easily matched even after the complex changes.

5.5. Wavelet

Fourier transform (FT) is a transform that can convert a signal from the time domain to the frequency domain. As the domain to which a signal belongs changes, so does the perspective of how things are presented. Some complex signals in the time domain become simpler when transformed into the frequency domain, so FT is widely used in signal

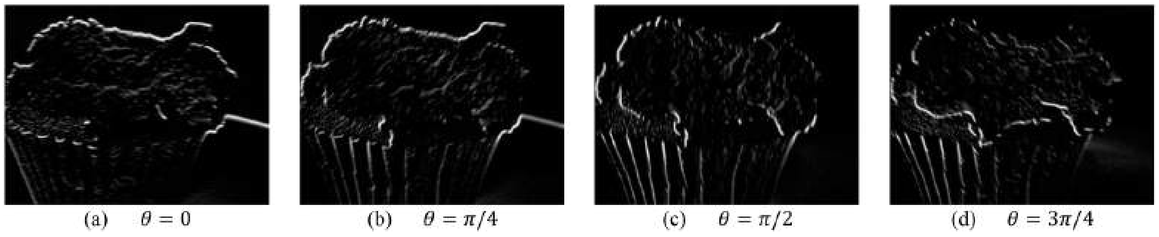


Fig. 15. Examples of extracting image texture features by Gabor kernels. Different kernel rotation degrees (θ) are used for each of the four images.

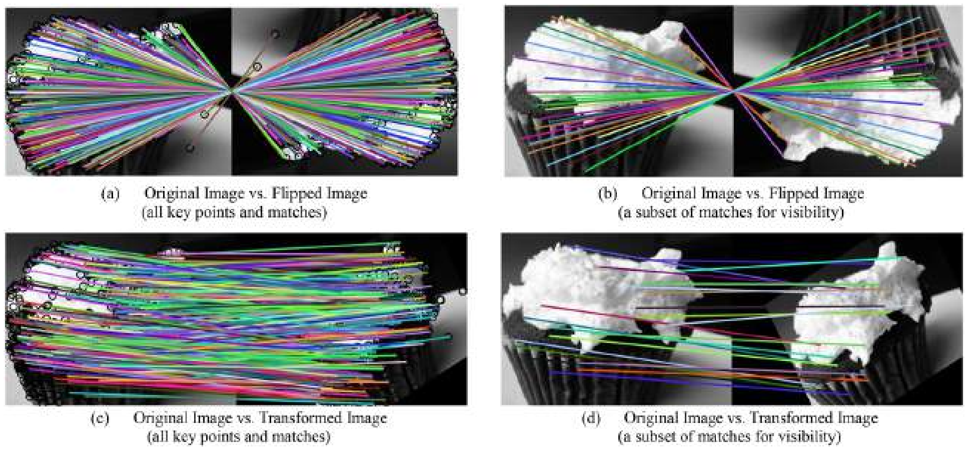

(a) Original Image vs. Flipped Image (all key points and matches)
(b) Original Image vs. Flipped Image (a subset of matches for visibility)
(c) Original Image vs. Transformed Image (all key points and matches)
(d) Original Image vs. Transformed Image (a subset of matches for visibility)

Fig. 16. An example of SIFT feature detection and description. The colorful lines in the images are to match the key points between grayscale images and flipped or transformed images. Image pairs (a) and (c) illustrates all key points and matches; (b) and (d) illustrates the subset of matches for visibility. The transformation applied in (c) and (d) is affine transform, including scale with scale factors 1.3 for the x-axis and 1.1 for the y-axis, counter-clockwise rotation of 90 degrees, and translation along the x-axis and y-axis.

analysis [204]. The one-dimensional FT can be defined as $\mathscr{F}(\mathrm{w}) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi \mathrm{w}t}\mathrm{d}t$, where the variable w represents frequency, t represents time and e^{-iwt} is the complex function.

The FT assumes that a periodic signal contains multiple frequency components and that any signal $f(t)$ can be synthesized by summing multiple basis functions. It uses a set of trigonometric functions as orthogonal bases for the linear transformation of the original function. Its physical meaning is, therefore, the projection of the original function onto each set of basis functions. Although the FT can simplify complex problems by converting time domain signals to the frequency domain, it also has drawbacks. The FT usually only captures the frequency composition of a signal but not the time of occurrence and spatial information of these frequencies.

As a result, the Fourier-transformed frequency information cannot be located at different times or in different spaces [205]. It makes it possible for different non-stationary signals in the time domain to overlap in the frequency domain [206]. One solution to this problem is the short-time FT (STFT) [207], which decomposes the time domain signal into multiple short-time signals along the time axis and assumes that the frequency does not vary with time, thus segmenting the signal to locate it. However, the window's width in STFT is fixed [208], with large windows resulting in low time resolution and difficulty obtaining details in the time domain and small windows resulting in low-frequency resolution [209]. As a result, it makes signal analysis inflexible and difficult to combine frequency and time resolution.

The wavelet transform obtains frequencies while localizing the FT by replacing the basis function from an infinitely long trigonometric function with a finite and decaying wavelet basis function [210]. In addition, the wavelet transform is used to analyze multiscale signals using variable-size windows [208]. The continuous wavelet transform of a function $f(t)$ can be defined as

$$W(a,\tau) = \int_{-\infty}^{\infty} f(t)\psi^{*}_{a,\tau}(t)\mathrm{d}t, \tag{39}$$

where $\psi_{a,\ \tau}(t)$ is a scaled and translated version of a mother wavelet $\psi(t)$ with scale a and translation τ ($a > 0, \tau \in \mathbf{R}$), and $\psi^{*}_{a,\ \tau}(t)$ is the conjugate complex of $\psi_{a,\ \tau}(t)$. If $\psi_{a,\tau}(t)$ is not a complex function, then the conjugate complex operation has no effect. The scaled and translated wavelet $\psi_{a,\tau}(t)$ can be defined as Eq. (40).

$$\psi_{a,\tau}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-\tau}{a}\right). \tag{40}$$

The discrete wavelet function can be obtained by defining a wavelet function with discretized scale $a = c^m$ and discretized translation $\tau = nbc^m$, where $m,n \in Z$, $b \neq 0$, and $c > 1$. A scaled and translated wavelet function can be defined as Eq. (41).

$$\begin{aligned}\psi_{m,n}(t) &= \frac{1}{\sqrt{c^m}}\psi\left(\frac{t-nbc^m}{c^m}\right)\\ &= c^{-\frac{m}{2}}\psi(c^{-m}t - nb),\end{aligned} \tag{41}$$

where $c^{-\frac{m}{2}}$ is used to regularise the function. By taking a_0 to be 2 and b to be 1, the function can be further reduced to a binary wavelet to obtain an orthogonal basis function [211]. The simplified formula is shown in Eq. (42).

$$\psi_{m,n}(t) = 2^{-\frac{m}{2}}\psi(2^{-m}t - n). \tag{42}$$

The scaling function (a.k.a., the father wavelet) can also produce wavelets. It is interrelated with the mother wavelet [212]. Eq. (43) presents the general and normalized form of the scaling function.

$$\varphi(t) = \sum_n h_n 2^{-\frac{m}{2}}\varphi(2^{-m}t - n). \tag{43}$$

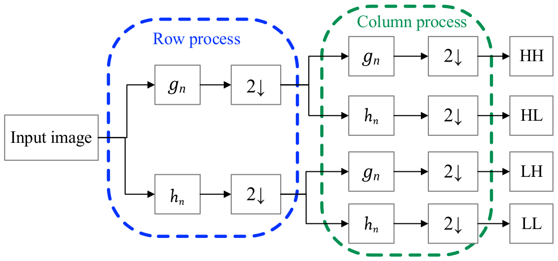
The mother wavelet can be described as Eq. (44) according to the relationship between itself and the scaling function.

$$\psi(t) = \sum_n g_n 2^{-\frac{m}{2}}\varphi(2^{-m}t - n), \tag{44}$$

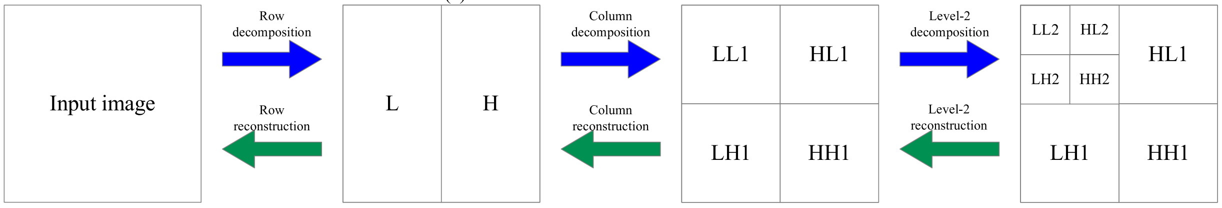
where h_n and g_n in Eqs. (43) and (44) are refinement coefficients known as quadrature mirror filters (QDFs). Different families of wavelets have different QDFs, which can be categorized into high-pass filters (g_n) and low-pass filters (h_n). The relationship between g_n and h_n can be described as $g_n = (-1)^n h_{1-n}$, which is derived through Fourier analysis using orthogonality [213]. More details can be found in [214] and [215].

The human visual system is adaptive to the varying sizes of objects in an image. Depending on the distance between the observer and the object, it can acquire different representations of the object. At greater distances, the observer sees the overall outline of the object, while at closer distances, the observer sees the details of the object. It is equivalent to decomposing an image at different scales. A similar decomposition process can be achieved by applying a wavelet transform to the image, called two-dimensional discrete wavelet decomposition. Fig. 17 (a) illustrates the wavelet transform process.

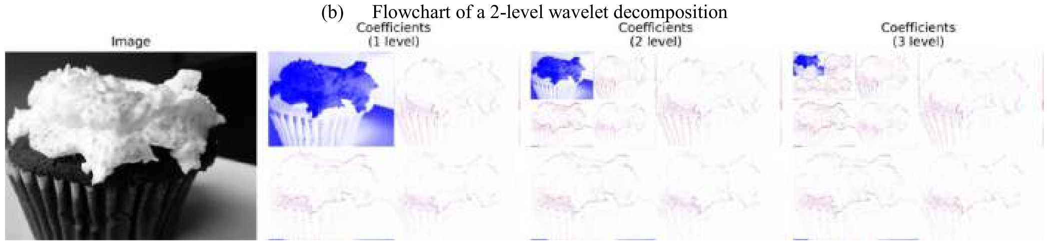
The two-dimensional discrete wavelet decomposition starts with a wavelet transform of every row of the raw image to gain the L (low-frequency component) and H (the high-frequency component) of the raw image in the horizontal direction. And then, a wavelet transform is applied to every column of the transformed data to obtain LL (low-frequency components of the raw image in both the horizontal and vertical directions), LH (the low-frequency component in the horizontal



(a) Two-dimensional wavelet transform



(b) Flowchart of a 2-level wavelet decomposition



(c) A 3-level wavelet decomposition on the cupcake image. A blue-white-red color map is applied for visibility.

Fig. 17. Illustration of wavelet transform and wavelet decomposition.

direction and the high-frequency component in the vertical direction of the raw image), HL (the high-frequency component in the horizontal direction and the low-frequency component in the vertical direction of the raw image), and HH (high-frequency components in both the horizontal and vertical directions).

The decomposition process results in LL, HL, LH, and HH, representing the approximate image, the vertical edge features, the horizontal edge features, and the diagonal features, respectively. The decomposed image is obtained by inverting each column of the transformed data with a discrete wavelet and then each row with a one-dimensional discrete wavelet. Fig. 17(b) illustrates the two-dimensional wavelet decomposition and reconstruction process. Fig. 17(c) illustrates an example of a 3-level wavelet decomposition on the cupcake image.

6. Traditional machine learning

Traditional machine learning algorithms were predominant in food category classification a decade ago. However, these algorithms are still worth investigating because they are better for small datasets than deep models. Meanwhile, traditional machine learning approaches are faster to converge and more interpretable than deep learning methods as they are mainly developed on statistical analysis and probability estimation.

Moreover, traditional machine learning models are still evolving to improve their generalization performance continuously. Traditional machine learning models are usually built upon handcrafted features, but they are more likely to be adapted with automated features from deep models nowadays. All the machine learning approaches fall into two types: supervised methods or unsupervised ones, depending on the training supervision conditions. In this section, we will briefly review the famous machine learning algorithms.

6.1. Support vector machine and logistic regression

Support vector machine (SVM) can be the most well-known machine learning method for binary-class classification problems in supervised learning, which was proposed by CORTES, C. et al. (1995) [216]. The core idea behind the vanilla SVM is simple: it aims to find the separation hyperplane with the maximum margin between both classes, as is shown in Fig. 18(a). For a training set $\mathrm{T}=\{(x_1,y_1),\ (x_2,y_2),\ (x_3,y_3),\ldots,\ (x_n,y_n)\}$ where $y_i\in\{+1,\ -1\}$, its classification hyperplane in the feature space can be expressed as $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+\ b\ =\ 0$, where $\boldsymbol{w}=(w_1,w_2,w_3,\ldots,w_d)^{\mathrm{T}}$ decides the direction of the hyperplane, and b is the offset. Therefore, if the hyperplane can classify all the training samples correctly, that is:

$$\begin{cases} w^{\mathrm{T}}x_i+b\geq+1, y_i=+1 \\ w^{\mathrm{T}}x_i+b\leq-1, y_i=-1 \end{cases}. \tag{45}$$

Then, the sum of the distances between the hyperplane and the support vectors from both classes can be computed using $\mathrm{D}\ =\frac{2}{\|w\|}$. The training objective of an SVM is to find the hyperplane with the maximum D, which can be expressed as

$$\max_{w,b}\frac{2}{\|\ w\ \|}$$
$$s.t.\,y_i\left(w^{\mathrm{T}}x_i+b\right)\geq1, i=1,2,\ldots,n. \tag{46}$$

This is the basic SVM, which can be solved by its dual problem and the sequential minimal optimization efficiently. Then, SVM was improved to solve linear-inseparable problems with kernel functions. The samples can be mapped into higher dimension spaces where they can be separated linearly if the original feature dimension is finite. SVM can also be error-tolerated with the soft margin.

Logistic regression (LR) is a linear regression algorithm for binary-class classification in supervised learning [217]. In Logistic regression, all the samples are mapped into the interval [0,1] so that the samples with output values over 0.5 are classified as one class, and all the others can be classified as the other class. For a training set $\mathrm{T}=\{(x_1,y_1),\ (x_2,y_2),\ (x_3,y_3),\ldots,\ (x_n,y_n)\}$ where $y_i\in\{0,1\}$, Logistic regression using sigmoid function can be written as $\mathrm{LR}(x)\ =\frac{1}{1+e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$.

LR training is to obtain the best $\boldsymbol{w}$ and b. There are many methods to solve this problem, such as maximum likelihood estimation and cross entropy-based loss function.

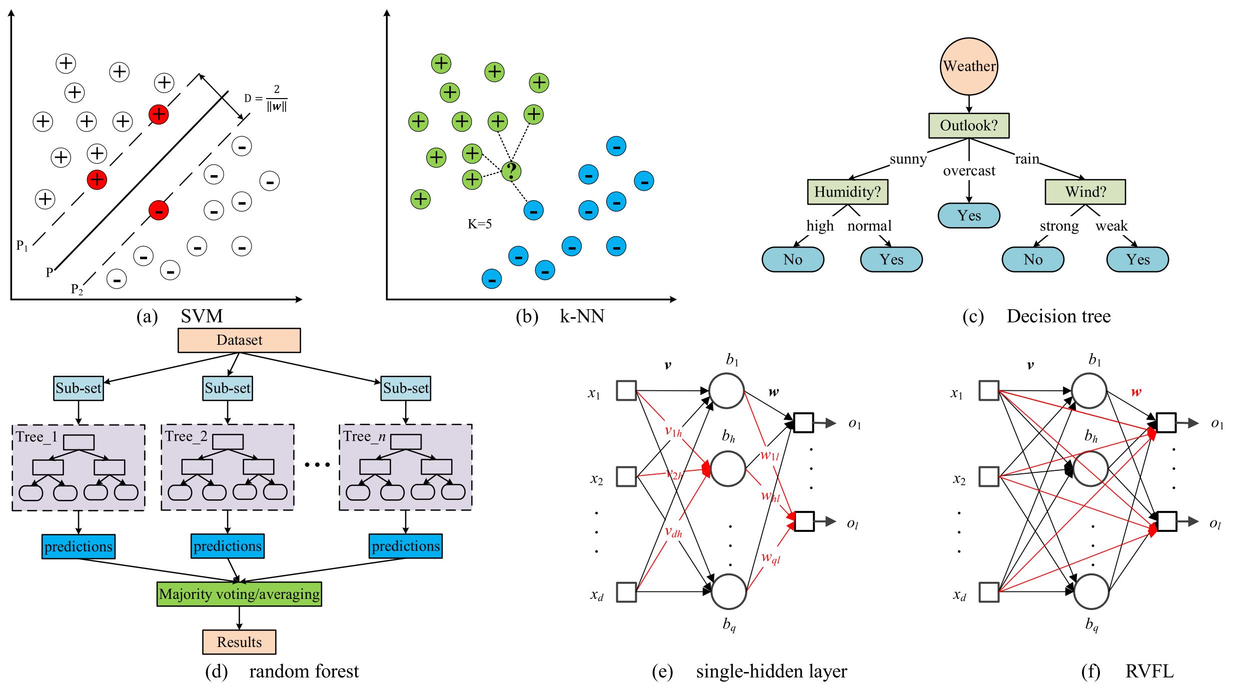


Fig. 18. Illustration of traditional machine learning models. (a) support vectors and classification hyperplanes, (b) an example of k-NN, (c) an example of a decision tree, (d) An example of a random forest, (e) A single-hidden layer feedforward structure, (f) the structure of an RVFL.

6.2. k-nearest neighbors, K-means, and fuzzy C-means

The k-nearest neighbors (k-NN) is a supervised-learning algorithm that is easy to understand and implement [218]. Unlike other algorithms, k-NN does not come up with a classification model. Instead, it just stores the entire training set during the training stage.

In the testing stage, for a testing sample, its k nearest neighbors in the training set will be obtained based on their distances, such as Euclidean distance. Then, the label of the testing sample is generated based on the majority voting on the labels of its k nearest neighbors. A toy instance is demonstrated in Fig. 18(b). k-NN is fast to train, but its testing may require more time and memory when the dataset gets larger or the dataset is of higher dimension.

K-means is a basic clustering algorithm in unsupervised learning [219]. Without the labels of the samples, K-means works with merely the feature vectors in the latent space. Initially, K clustering center points are randomly generated. Then, every training sample is grouped into one center point based on the distances between the training sample and the center points. Afterward, the center points will be updated using the training samples in their groups. This grouping and updating will be repeated until the maximum iteration times or the center points do not change anymore. K-means is effective in finding the distribution patterns in the latent feature space. However, it requires manual intervention to define the value of K, which is vital for the algorithm because different values of K can generate different results.

K-means is a hard clustering method, which means every sample can only be classified into one group. However, there are many fuzzy definitions in our lives without strict borders, so there can be overlapping areas among different groups. To this end, fuzzy C-means (FCM) are proposed, which can be seen as the generalized form of K-means [220]. A degree vector for each training sample is introduced to estimate the probability that it belongs to each group. Therefore, FCM can be defined as

$$\begin{aligned} &\min_{u,v} \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} \|x_i - v_j \text{AptCommand2016;}^2 \\ &\qquad s.t. \sum_{j=1}^{C} u_{ij} = 1, u_{ij} \geq 0. \end{aligned} \tag{47}$$

in which N is the number of training samples, C is the number of clusters, x_i denotes the i^{th} sample vector, v_j means the j^{th} center point, u_{ij} denotes the membership degree that the i^{th} sample belongs to the j^{th} center point, and $m>1$ stands for the fuzzy degree. The above equation can be solved using quadratic programming conveniently. However, FCM may fail when there are too many training samples, or the dimension of samples is high.

6.3. Bayesian network, decision tree, and random forest

The Bayesian network, also called the belief network, is a directed acyclic graphical model based on probabilities [221]. The directed acyclic graph in a Bayesian network is used to describe the dependence relationship among the attributes. Once the structure of the graph is determined, training becomes easy for the Bayesian network. It just scans all the training samples and calculates the conditional probabilities for every node in the network.

Unfortunately, the structure of the graphic model is unknown in real-world applications. The score-searching algorithm is often employed to get the structure. Specifically, a score function is introduced to estimate the matching degree of the graph structure and the training samples.

A decision tree is a famous classification model which works with a tree architecture [222]. There is a root, several internal nodes, and several leaves in a decision tree. In this model, the final decision can be made using a series of sub-decisions. On every internal node, it makes tests on the attributes to divide the samples into sub-groups. The final results can be found on the leaves. To estimate the purity of the samples in one sub-group, scholar leverage information entropy which can be expressed as

$$\mathrm{E(D)} = -\sum_{i=1}^{C} p_i \log_2 p_i, \tag{48}$$

in which D is the current sub-group of samples, C is the total number of classes in D, and p_i denotes the ratio of the i^{th} class of samples in D. The training of a decision tree is to minimize the entropies of the samples in the leaves. A toy example of a decision tree is shown in Fig. 18(c).

The random forest is an ensemble learning model developed based on decision trees [223]. In a random forest, a set of decision trees are trained independently. For instance, the entire training set is divided into n groups of data with the same size to train n decision trees independently and individually. Then, in the testing phase, the predicted labels of testing samples can be obtained by majority voting or averaging on the output labels from the decision trees.

A graph of the random forest is presented in Fig. 18(d). The training set can also be divided by the attributes. That is, every decision tree is trained with a sub-set of all the attributes. Parallel training can be implemented for the random forest to accelerate the training. However, the performance of the random forest is dependent on the decision trees. If the decision trees are poorly designed and trained, the random forest cannot produce satisfactory results.

6.4. BP neural network, extreme learning machine, and random vector functional link

Inspired by the activation of neurons in human brains, artificial neural networks are invented [224]. There are various shallow neural networks available, but the back propagation neural network (BPNN) can be the most important model [225]. In the early time, the perceptron was proposed, but it can only be used to solve linear-separable problems because a perceptron only consists of two layers (the input layer and output layer).

To overcome this problem, BPNN is presented typically with a single-hidden layer feedforward structure, as is shown in Fig. 18(e). The activation function in networks is used to provide non-linearity. The sigmoid function is often used in BPNNs, which is expressed as $f(x) = \frac{1}{1+e^{-x}}$.

Given a training set $S = \{(x_1,y_1), (x_2,y_2), (x_3,y_3), \ldots, (x_n,y_n)\}$, the output of the BPNN is denoted as $\mathbf{O} = (o_1, o_2, \ldots, o_n)^{\mathrm{T}}$. Then, the training mean squared error can be written as $\text{err} = \frac{1}{n}\sum_{i=1}^{n} \|y_i - o_i\text{AptCommand2016;}^2$.

Then, all the parameters in the BPNN can be updated using the gradient descent strategy. With a pre-defined learning rate η, the updating of w_{hl} for sample (x_k, y_k) can be computed using the chain rule as

$$\Delta w_{hl} = -\eta \frac{\partial \text{err}}{\partial w_{hl}} = \eta o_l^k \left(1 - o_l^k\right)\left(y_l^k - o_l^k\right) b_h. \tag{49}$$

The input weights v_{dh} can be updated with

$$\Delta v_{dh} = \eta x_d b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} o_j^k \left(1 - o_j^k\right)\left(y_j^k - o_j^k\right). \tag{50}$$

The biases can be updated similarly. The updating of the parameters will repeat until the termination condition is met. Back propagation is the most widely used method to train artificial neural networks, and it is also applied in training deep networks.

Extreme learning machine (ELM) offers a different way to train the structure in Fig. 18(e) [226]. Instead of the tedious iterations to update the parameters in BPNNs, ELM presented a close-formed solution to determine the parameters. Initially, the input weights $\boldsymbol{v}$ and hidden biases $b = (b_1, b_2, \ldots, b_q)^{\mathrm{T}}$ are assigned with random values, which will remain fixed during the training procedure. Then, the output matrix of the hidden layer $\mathbf{H}$ can be calculated with the input vectors from the training set. Finally, the output weights $\boldsymbol{w}$ can be determined by pseudo-inverse: $w = \mathbf{H}^{\dagger}\mathbf{Y}$, where $\mathbf{Y} = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ is the label matrix and $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{H}$. The ELM can converge faster than BPNN, and the input features are randomly mapped into the hidden layer without training, which is beneficial for the generalization of the network [227].

The structure of a random vector functional link (RVFL) is different from conventional feedforward neural networks [228], as shown in Fig. 18(f). Besides the traditional links between layers, an extra connection links the input layer with the output layer directly. See the connections in red in Fig. 18(f). The training algorithm of RVFL is similar to ELM, as both of them belong to random neural networks. The only difference is that the output matrix of the hidden layer in the RVFL should be concatenated with the original input features before computing the pseudo-inverse. The short-cut connections in an RVFL can be very useful if the input features are already discriminant.

7. Convolutional neural networks

With the continuous progress of computer technology, convolutional neural networks (CNNs) are currently paving new avenues for image processing. They have been proven to have abilities to achieve great success in image processing [229]. CNNs attract a sea of attention from various fields, such as medical science, agriculture production, etc. In recent years, many scholars applied CNNs to food category recognition [230].

In contrast to other food category recognition methods used for many years, CNNs are characterized by a significantly increased number of successive layers. CNNs can reveal higher-level features and hierarchical relationships with the added layers. The increasing layers of neural networks could require more training time and computational costs. Many studies have demonstrated that CNNs with increased depth could enhance image processing for food category recognition. A typical feature map-based framework of the CNN (VGG16) is displayed in Fig. 19.

7.1. Convolution, padding, and pooling

CNNs are composed of different layers. The input layer is the initial layer, such as food images. The end layer is the output, e.g., the predicted food category. The hidden layers are after the input and output layers [231]. The convolutional layer is one of the most significant hidden layers. There are multiple optimizable filters in the convolutional layers. Convolution is essentially the sliding of the filter on the layer and calculating the filter dot product and the values of layers [232].

The results calculated by dot product are named feature maps [233]. The convolution operation is shown in Fig. 20(a). The feature maps would be the input for the next layer, such as another convolutional layer, pooling layer, and so on. The feature maps extracted from deeper convolution layers usually have higher concepts and more abstract patterns [234]. Therefore, the successive convolutional layers could contribute to the CNNs learning subtle features.

The calculation step of convolution is shown as follows $O = \left\lfloor \frac{I-R+2q}{e} \right\rfloor + 1$, where the input size and output size are $I \times I$ and $O \times O$, respectively. The filter size is $R \times R$, and the stride and padding are represented as e and q. $\lfloor\rfloor$ is the floor function.

There are some limitations to the convolutional layers. First, the image size would be reduced after the convolution operations [235]. The input and output sizes of the convolution operation are shown in Fig. 20(a). The image size could be very small by several convolution operations. Second, after the input image is convolved with the convolution filter, some values are lost, only some pixels are convolved at the edges, and much information at the edges of the image is lost.

To settle these problems, some researchers first add padding to the original matrix, that is, add a few values on the matrix boundary to

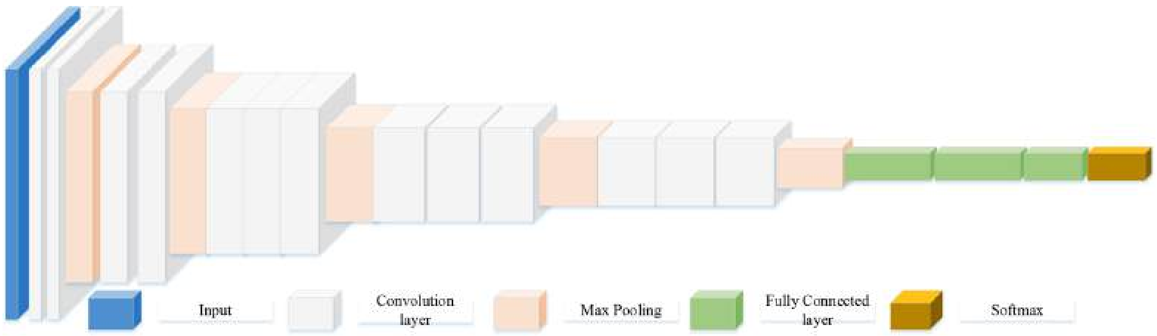


Fig. 19. The feature map-based framework of VGG16.

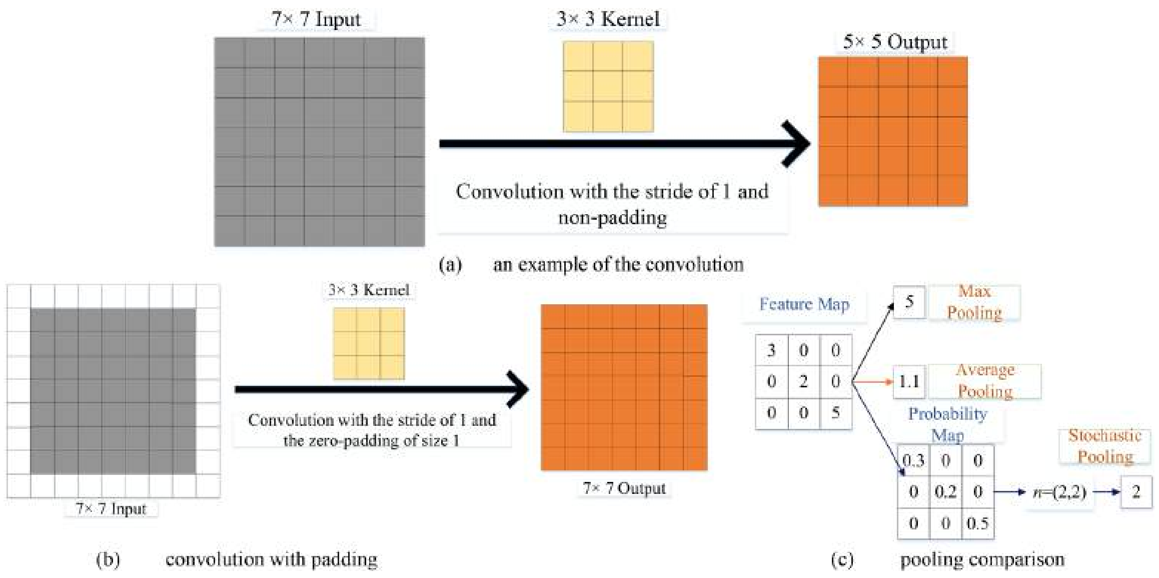

7× 7 Input
3× 3 Kernel
5× 5 Output
Convolution with the stride of 1 and non-padding
(a) an example of the convolution
3× 3 Kernel
Convolution with the stride of 1 and the zero-padding of size 1
7× 7 Input
7× 7 Output
(b) convolution with padding
Feature Map
Max Pooling
Average Pooling
Probability Map
Stochastic Pooling
n=(2,2)
(c) pooling comparison

Fig. 20. Illustration of convolution and pooling.

increase the size of the original matrix. An example of padding is presented in Fig. 20(b). The input size is the same as the output size by the convolution with one-pixel thick zero-padding.

After several convolutional layers, the feature maps are commonly down-sampled using the pooling operations. There are some advantages to pooling operations: The feature map size is reduced while keeping discriminant information, and it can detect more abstract information and thereby condense semantic features. Like the convolutional operation, the filter size and stride are assigned before pooling operations. Several different pooling operations are available and popular in recent research. In this paper, three popular pooling operations are shown in Fig. 20(c), which are max pooling [236], average pooling [237], and stochastic pooling [238].

P_R is pooling region in the feature map, and l is the index of each element. Max pooling obtains the maximum value of the feature map by step. The formula of max pooling ($M_{pooling}$) is given as follows:

$$M_{pooling} = \max(b_l),\ l \in P_R. \tag{51}$$

Average pooling averages the values of the feature map by step. The formula of average pooling ($A_{pooling}$) is shown as follows:

$$A_{pooling} = \frac{\sum_{l\in P_R} b_l}{|P_R|}, \tag{52}$$

where $|P_R|$ represents the number of elements for the $|P_R|$.

Stochastic pooling ($S_{pooling}$) selects the map based on the probability map $G = (g_1, g_2 \dots g_l, \dots)$. The formula of g_l is

$$g_l = \frac{b_l}{\sum_{l\in P_R} b_l}. \tag{53}$$

The outputs are obtained from the multinomial distribution. The formula of stochastic pooling is given as follows:

$$S_{pooling} = b_n,\ n \sim (g_1, g_2 \dots g_l, \dots). \tag{54}$$

7.2. Activation functions

The nonlinearity of CNNs can be increased by adding the activation function [239]. The output signal could be a simple linear function without the activation function. The complexity and mapping ability of the results learned by linear equations from data are limited, far less than those obtained by nonlinear equations. Some common activation functions are selected in CNNs.

The formula of the sigmoid function [240] is $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. The formula of the tanh function [241] is presented below:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{55}$$

The formula of the ReLU function [242] is written as below:

$$\text{ReLU}(x) = \begin{cases} x, x > 0 \\ 0, x \le 0 \end{cases}. \tag{56}$$

The equation of the leaky ReLU (LReLU) function [243] is defined below:

$$\mathrm{LReLU}(x) = \begin{cases} x, x > 0 \\ 0.01x, x \leq 0 \end{cases}. \tag{57}$$

The formula of the PReLU function [244] is given below:

$$\mathrm{PReLU}(x) = \begin{cases} x, x > 0 \\ ax, x \leq 0 \end{cases}, \tag{58}$$

where a is very small. These common types of activation functions and figures are concluded in Table 4 and Fig. 21. Some other variants of activation functions are discussed in Table 5.

7.3. Batch normalization and dropout

The distribution may change as the depth of the network increases. The distribution may be close to two extremes close to the upper or lower limits of the interval. One of the reasons for this phenomenon is the slow convergence speed during training. In this case, the gradient may disappear in the backpropagation, which could cause a slow convergence problem during CNN model training.

Batch normalization (BN) was proposed to address this problem [252]. The method to distribute the output value of each layer to a normal distribution with a mean value of 0 and variance of 1 is BN. This not only can avoid this problem but also accelerate the training speed. The equations of BN are presented as follows:

First, the data is defined as $X = [x_1, x_2, \ldots, x_n]$. Second, the mean value of this data is calculated as $\varphi_B = \frac{1}{n}\sum_{i=1}^{n} x_i$. Third, the variance is shown as $\vartheta_B = \frac{1}{n}\sum_{i=1}^{n}(x_i - \varphi_B)^2$. Then, the normalization is performed as follows:

$$x_i' = \frac{x_i - \varphi_B}{\sqrt{\vartheta_B{}^2 + \in}}, \tag{59}$$

where $\in$ is greater than 0 to guarantee that the denominator is greater than 0.

Finally, the network nonlinearity is increased by introducing two parameters: $y_i = \alpha x_i' + \beta$, where α and β are the scale parameter and shift parameter, respectively.

Dropout means randomly 'discarding' some neurons in the network. This is to reduce the problem of overfitting [253]. Dropout is to randomly select some neurons in the network and set the output of these neurons to 0. Based on this method, the overfitting problem could be inhibited. The standard neural network and the dropout-applied network are presented in Fig. 22.

8. Transfer learning

Deep learning has gained much attention from researchers in recent years and has become the master algorithm in a wide range of applications, such as image recognition, machine translation, speech recognition, and medical image analysis. However, in some domains, the amount of labeled data is limited because of the high cost of data annotation and difficulties in data acquisition. This can affect the model performance as the deep learning models can suffer from the overfitting problem. Deep learning models require sufficient data to learn of extracting meaningful features from the input. In recent years, transfer learning has been widely used to overcome the problem caused by insufficient data. This technique allows deep learning models to transfer knowledge learnt from one domain to the other domain, as shown in Fig. 23.

The initial domain for gaining prior knowledge is referred to as a source domain. The domain where the models are adapted is referred to as a target domain. Knowledge learnt from the source domain represents the weights of the model. Then the pre-trained mode is adapted to the target domain to perform a new learning task.

This section reviews current popular pre-trained convolutional neural networks (CNNs) widely used in transfer learning. These networks are pre-trained on a large-scale dataset, such as *ImageNet* before their weights are transferred to a target domain. The architectures of these networks show the design trend from designing deeper and more complex architectures to focusing on striking a balance between accuracy and complexity, from the manual design by human experts to an automatic pattern.

8.1. AlexNet, VGG and SqueezeNet

CNNs gradually became dominant across various computer vision tasks after AlexNet considerably outperformed the previous state-of-the-art approaches on the ImageNet LSVRC-2010 dataset and won the ILSVRC-2012 competition in 2012. AlexNet contains eight learnable layers, including five convolutional layers and three fully-connected layers [15]. Max-pooling layers are applied after some of the convolutional layers to reduce the dimensionality of feature maps. A final softmax function following the last fully-connected layer generates a prediction distribution for classes.

Adopting the non-saturating nonlinearity, rectified linear units (ReLUs), in AlexNet makes the network training several times faster than its alternatives using tanh units. AlexNet contains 60 million parameters [254,255]. To deal with the overfitting problem, the authors introduce data augmentation to enlarge the dataset artificially. In addition, dropout is applied in the network during training. Other techniques applied to AlexNet leading to performance improvement include local response normalization and overlapping pooling. The ground-breaking results on ILSVRC achieved by AlexNet have opened a new paradigm for computer vision tasks and spurred more and more research in the field of deep learning.

Simonyan, K. et al. (2014) [256] further pushed the depth of convolutional networks by introducing the VGG architecture, addressing an important aspect of CNN architecture design: the depth. The VGG architecture consists of a stack of convolutional layers with 3 × 3 filters and three fully-connected layers at the network's end. Moreover, 1 × 1 convolutional filters are also adopted to transform the input channels linearly. The non-linear ReLUs are used to introduce nonlinearity to the network. Different from AlexNet, the VGG architecture does not contain local response normalization, which could increase memory requirements and computational costs [257,258].

There are a few convolutional layers followed by max-pooling layers. The VGG16 and VGG19 are two representatives of the VGG architecture that consist of 13 and 16 convolutional layers, respectively. The architecture of VGG16 is illustrated in Fig. 19. Compared to AlexNet, the VGG16 is a huge network with many parameters which takes more time to train. Nevertheless, it highly outperforms previous methods in the ILSVRC-2012 competition and competes for the classification task winner in the ILSVRC-2014 competition.

Since the success of AlexNet, CNNs have gained more and more interest from researchers. Various CNN architectures are proposed to handle image recognition and classification. Given an accuracy level, there can be seen multiple CNNs of different architectures can be able to achieve that requirement. However, most research focuses on improving the model performance, while the compact design of CNN architecture is rarely explored.

A small CNN architecture with fewer learnable parameters has several advantages over its cumbersome equivalents: more efficient

Table 4
The comparison of activation functions.

Activation function	Speed of convergence	Output range
sigmoid	low	(0, 1)
tanh	low	(−1, 1)
ReLU	Fast	[0, +∞)
LReLU	Fast	(−∞, +∞)
PReLU	Fast	(−∞, +∞)

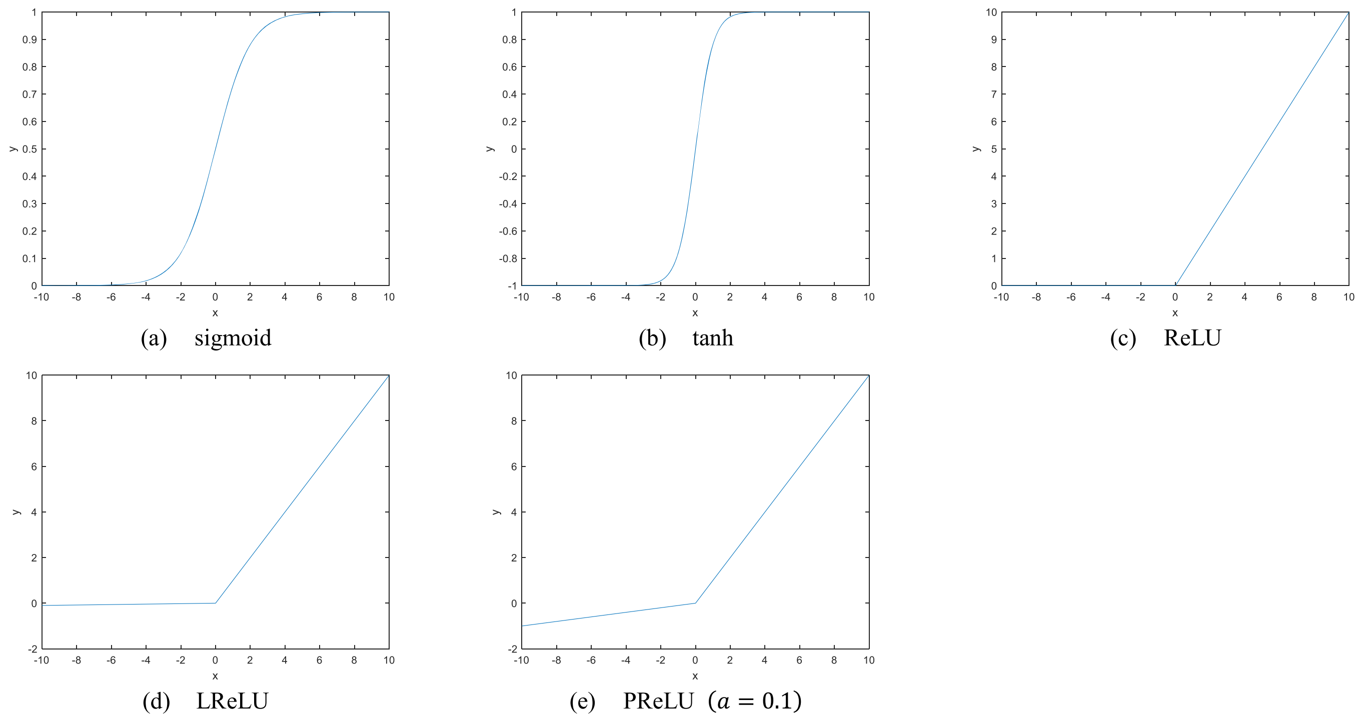


Fig. 21. Comparison of five common activation functions.

Table 5
Other variants of activation functions.

Name	Equation	Output range
SReLU [245]	$\mathrm{SReLU}(x) = \begin{cases} t^u + b^u(x - t^u), x \geq t^u \\ x, t^u > x > t^l \\ t^l + b^l(x - t^l), x \leq t^l \end{cases}$	(-∞, + ∞)
ELU [246]	$\mathrm{ELU}(x) = \begin{cases} x,\ \mathrm{if} x \geq 0 \\ \beta(e^x - 1),\ \mathrm{if}\ x < 0 \end{cases}$ $\beta = 1$	(-∞, ∞)
SELU [247]	$\mathrm{SELU}(x) = \begin{cases} \gamma \mathrm{x},\ \mathrm{if}\ x > 0 \\ \gamma\beta(e^x - 1),\ \mathrm{if}\ x \leq 0 \end{cases}$ $\gamma \approx 1.0507, \beta \approx 1.6733$	$(\approx -1.76. + \infty)$
Softplus [248]	$\mathrm{Softplus}(x) = \log(1 + e^x)$	(0, +∞)
Softsign [249]	$\mathrm{Softsign}(x) = \frac{x}{1 + \lvert x \rvert}$	(-1,1)
SLU [250]	$\mathrm{SLU}(x) = \begin{cases} x,\ \mathrm{if}\ x \geq 0 \\ 2\log\left[\frac{(e^x + 1)}{2}\right],\ \mathrm{if}\ x < 0 \end{cases}$	(-2log2, +∞)
EliSH [251]	$\mathrm{ELiSH}(x) = \begin{cases} \frac{x}{1 + e^{-x}}, x \geq 0 \\ \frac{e^x - 1}{e^{-x} + 1}, x < 0 \end{cases}$	(-1, +∞)

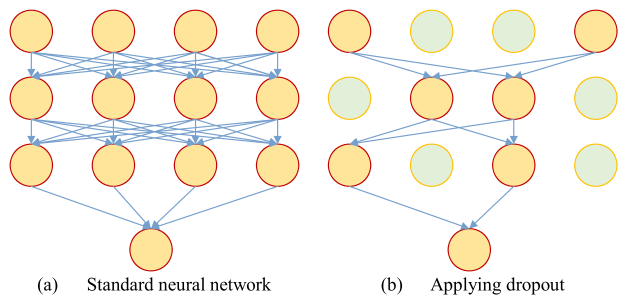


Fig. 22. Illustration of dropout.

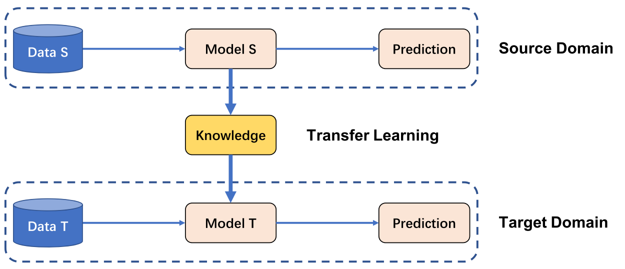


Fig. 23. Knowledge transferred from a source domain to a target domain (S means source, and T means target).

communication in distributed training, less bandwidth required to export models for over-the-air updates, and more feasible deployment on resource-limited hardware. Iandola, F. N. et al. (2016) [259] investigated the impact of CNN architectural designs on model size and performance and proposed SqueezeNet with 50 × fewer parameters than AlexNet and performs on par with AlexNet on ImageNet.

To equip SqueezeNet with a small CNN architecture and competitive accuracy, Iandola, F. N. et al. (2016) [259] designed three main strategies: 1) use 1 × 1 filters to replace 3 × 3 filters to reduce the parameters; 2) reduce the channels of features maps to 3 × 3 filters, thereby reducing the channels of filters; 3) delay the down-sampling

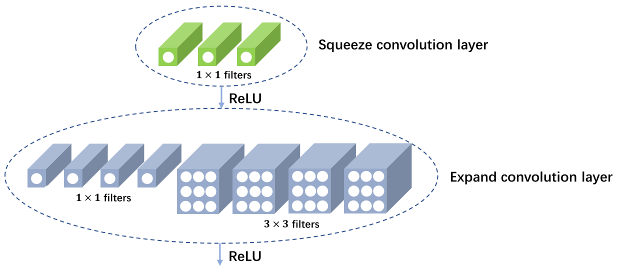


Fig. 24. Organisation of convolution filters in the Fire module.

operations to maintain large feature maps for most convolutional layers that help achieve higher classification accuracy. These strategies are implemented in the Fire module, as shown in Fig. 24.

A Fire module contains a squeeze convolution layer and an expand convolution layer. All filters in the squeeze layer are of the size 1 × 1, which implements the strategy of replacing 3 × 3 filters. The expand layer contains a mix of 1 × 1 and 3 × 3 filters. To limit the number of input channels corresponding to the second strategy, Iandola, F. N. et al. (2016) [259] set the squeeze layer as fewer filters than the expand layer.

The SqueezeNet is constructed by stacking a standalone convolutional layer, 8 Fire modules, and a final convolutional layer [260,261]. The number of filters in each Fire module increases along the network. Dropout is applied after the last Fire module to solve the overfitting problem. The total quantity of parameters in SqueezeNet is 1.2 million, 50 × fewer than AlexNet. The model size of SqueezeNet can be further compressed to 510 × smaller than AlexNet, while it achieves the equivalent accuracy.

8.2. *Inception*

Although directly increasing neural networks' depth and width can potentially improve their model performance, it can inevitably lead to a dramatic increase in computation and memory requirements. In addition, with a larger number of parameters, networks tend to suffer from the overfitting problem, especially when the amount of training data is limited. These issues can be solved by replacing the fully connected architecture with a sparsely connected architecture, even inside the convolution operations. To find an optimal local sparse structure that can be repeated to construct a network, Szegedy, C. et al. (2015) [262] proposed the Inception module to approximate the optimal local sparse structure in CNNs. In contrast to the conventional convolutional layer, where filters of the same size are applied, the Inception module introduces multiple filter sizes within a layer, as depicted in Fig. 25(a). Additionally, a parallel max-pooling path is added to the module.

To reduce computations, the computationally intensive 3 × 3 and 5 × 5 convolutions are followed by 1 × 1 convolutions. Feature maps produced from multiple paths are finally concatenated [263,264]. An Inception network is a network constructed by stacking Inception modules upon each other with max-pooling layers occasionally inserted. Typically, GoogLeNet is an incarnation of the Inception network, a 22-layer deep network consisting of nine Inception modules. GoogLeNet is equipped with auxiliary classifiers during network training to encourage discrimination in the low-level layers and attain extra gradient signals for better model training.

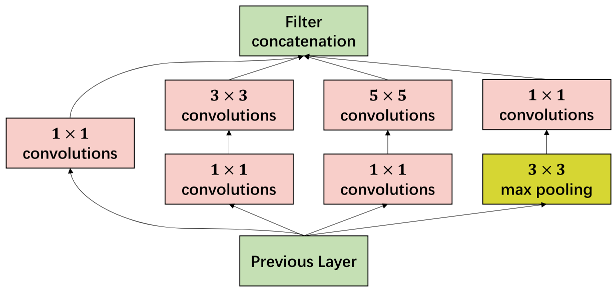
To scale up CNNs in efficient ways while maintaining high-quality network architectures, Szegedy, C. et al. (2016) [265] further revised the original Inception module under four design principles, as shown in Table 6.

Three different improved Inception modules are designed with the above design principles imposed, as shown in Fig. 25(b-d). The version on the left is obtained using two 3 × 3 convolutions to replace each 5 × 5 convolution in the original Inception module, as suggested by the third design principle. The version in the middle represents Inception modules after the spatial factorization into asymmetric convolutions. The version on the right represents Inception modules with expanded activation dimensions of filters.

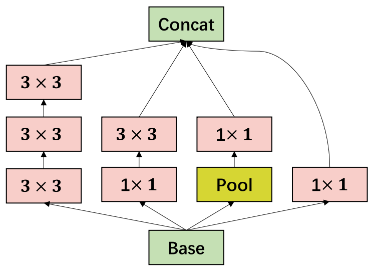
Based on the three types of improved Inception modules, the authors of [265] proposed a new architecture, termed Inception-v3, that consists of multiple convolutional layers followed by improved Inception modules, a batch-normalized auxiliary classifier as a side branch, and a final classifier regularised via label smoothing. The depth of Inception-v3 reaches 42 layers, while its computational cost is still much less than that of VGG architecture.

Table 6
Four design principles of the improved Inception module.

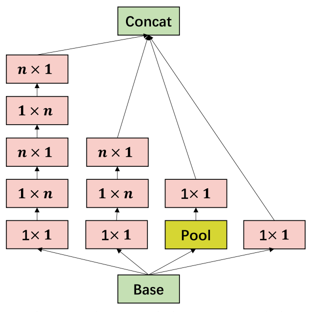
Design Principle	Content
I	avoid representational bottlenecks
II	promote high-dimensional representations
III	perform a spatial aggregation over low dimensional embeddings without loss of information
IV	balance the width and the depth of the network in terms of the computational budgets



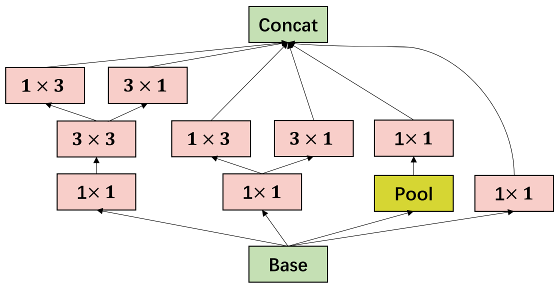
(a) overview of the Inception module



(b) an improved Inception module where each 5 × 5 convolution in the original module is replaced by two 3 × 3 convolutions



(c) an improved Inception module after spatial factorization into asymmetric convolutions



(d) an improved Inception module with expanded filter bank outputs

Fig. 25. Illustration of Inception module. (a) The overview, (b-d) Improved Inception modules introduced in Inception-v3.

8.3. Xception, ResNet, and DenseNet

Chollet, F. et al. (2017) [266] reformulated the Inception module in Inception-v3 as a large 1×1 convolution followed by a set of 3×3 convolutions that perform on non-overlapping proportions of the output channels, as illustrated in Fig. 26(a). This reformulation is based on the Inception hypothesis that separately looking at cross-channel correlations via 1×1 convolutions and spatial correlations via $n \times n$ convolutions could make the learning process more efficient. An extreme form of an Inception module uses a 3×3 convolution to separately map the spatial correlations in each channel, as shown in Fig. 26(b).

With a stronger hypothesis that the mapping of cross-channel correlations and spatial correlations can be completed separately, Chollet, F. et al. (2017) [266] proposed an architecture, termed Xception, that is constructed by a linear stack of Inception modules of the extreme form with residual connections around them. This architecture outperforms Inception-v3 on both the ImageNet and JFT datasets.

CNNs can be used to extract features of different levels from the input images. A network with more layers is expected to extract richer features. However, with the increase in the network depth, deeper models become difficult to train. Training deeper neural networks suffers from gradient vanishing and gradient exploding. In addition, the problem of performance degradation is observed on CIFAR-10 and ImageNet as the network depth of a “plain” network increases [267]. These problems have limited the design of deeper models expected to gain accuracy from the increased depth.

He, K. M. et al. (2016) [267] addressed the degradation problem by making layers fit a residual mapping. This can be realized by introducing shortcut connections between layers called identity mapping, as shown in Fig. 27(a). Formally, a building block based on the residual mapping is defined as

$$y = \mathscr{F}(x, \{W_i\}) + x, \tag{60}$$

where x represents the input of the building block, y represents the output, and the function $\mathscr{F}(\cdot)$ denotes the residual mapping to be learnt. The introduction of identity mapping does not require any parameters. Therefore, it does not increase the computation complexity. Instead, it solves the problems of vanishing and exploding gradients, allowing the design of greatly increased depth [268,269]. With the introduction of shortcut connections, the gradients can be directly backpropagated from the loss function to the preceding layers [270,271]. They make the deeper networks more easily optimized than those constructed by simply stacking layers without skip connections.

A family of Residual Networks (ResNets) can be created by varying the number of residual building blocks. The shortcut connections allow a ResNet to reach a depth of over 100 layers without optimization difficulty. In addition, the greatly increased depth result in accuracy gains. In terms of the model complexity, a 152-layer ResNet still has $2 \times$ lower complexity than VGGNets, while achieving better performance.

The shortcut connections introduced in ResNet help alleviate the gradient vanishing problem. These short paths can also preserve information about the input through many layers. To ensure maximum information flow along the network, Huang, G. et al. (2017) [272] adopted a dense connectivity pattern to connect layers within a building block. Specifically, all layers are connected directly in a way that each layer obtains the output feature maps learnt from preceding layers, as illustrated in Fig. 27(b).

In other words, the input for previous layers is also fed into the layers ahead. This can help to preserve the feed-forward nature and alleviate the information-vanishing problem. On the other hand, with a dense connectivity pattern, the gradients can be more easily back-propagated from the loss function throughout the network. The dense connectivity pattern allows the re-use of features learnt from preceding layers, thereby reducing the number of parameters used to re-learn redundant feature maps.

Different from ResNets, feature maps from preceding layers are concatenated instead of summation before they are passed into a layer [273,274]. For the l-th layer, it receives l sets of feature-map from preceding layers. The feature maps learnt by the l-th are passed on to subsequent layers. Within a building block that consists of L layers, there are N_c connections, where $N_c = \frac{L\times(L+1)}{2}$.

A DenseNet is constructed by multiple building blocks linked by transition layers and a classifier, as illustrated in Fig. 27(c). Transition layers are introduced to down-sample the size of feature maps and reduce the dimensionality, improving model compactness. The dense connectivity pattern allows a deep DenseNet to yield better performance with the increasing number of parameters without the problem of performance degradation.

8.4. MobileNet and ResNeXt

There can be seen that much effort has been made to achieve higher classification accuracy by making networks deeper and more complex. However, these advances may not efficiently balance the trade-off between performance and model complexity. In real-world scenarios, recognition tasks are often performed on platforms with limited computational resources, such as mobile devices and robotics. Mobile and embedded vision applications require the network architecture to be small and computationally efficient. Howard, A. G. et al. (2017) [275] presented a family of efficient network architectures called MobileNets for mobile applications that runs on a computationally limited platform.

MobileNet is primarily built on depthwise separable convolutions [276], where a standard convolution is factorized into a depthwise convolution and a pointwise convolution, separately learning spatial correlations and cross-channel correlations [277]. As illustrated in Fig. 28, instead of using a single filter of size $D_k \times D_k \times M$, M filters of size $D_k \times D_k \times 1$ are applied to the input, with each filter convolving with one channel of the input.

The depthwise convolution generates M feature maps with the same depth as the input. The pointwise convolution applies 1×1 convolution to the feature maps a depthwise convolution produces to learn cross-channel correlations. By replacing a standard convolution with a depthwise separable convolution, the computation can be reduced by R_c, where

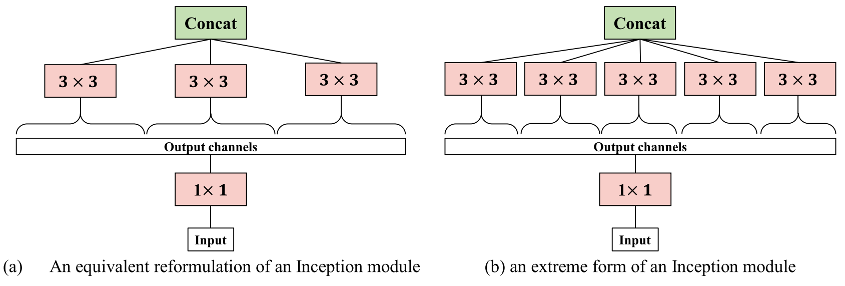


Fig. 26. Reformulations of the simplified Inception module.

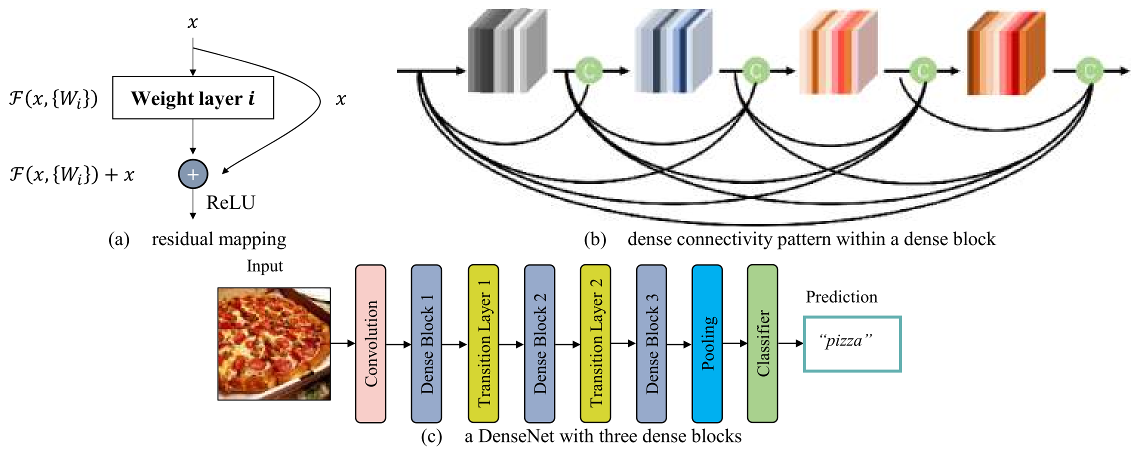


Fig. 27. Illustration of ResNet and DenseNet.

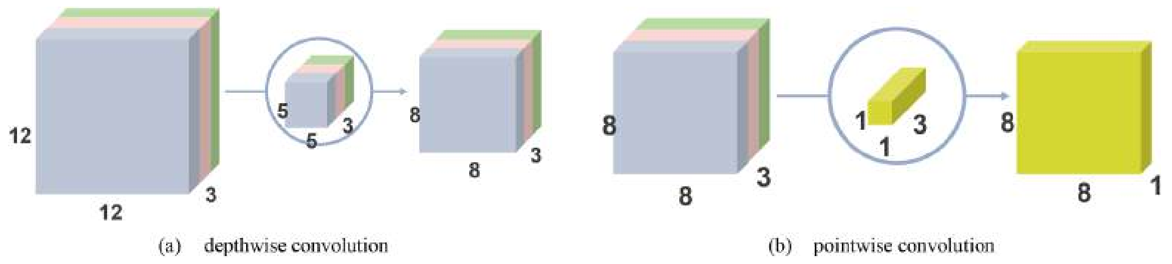


Fig. 28. The depthwise convolution and the pointwise convolution within a depthwise separable convolution.

$$R_c = \frac{1}{N} + \frac{1}{D_K^2}, \tag{61}$$

where N denotes the number of channels in the output maps. For example, 3×3 depthwise separable convolutions use around nine times less computation than standard 3×3 convolutions.

The architecture of MobileNet comprises one standard convolution at the beginning, a series of consecutive depthwise separable convolutions as the network backbone, followed by an average pooling layer and a fully-connected layer. To further reduce the model size and the latency of MobileNet, two hyperparameters, width and resolution multiplier, are introduced to thin a network at each layer. All these techniques allow customizing a MobileNet for efficiently trading off accuracy and model size.

Increasing the network depth and width are two common directions for improving model performance. With a deeper architecture, a model can extract richer information from the input. Extending the width of the network improves the expressiveness of neural networks. Xie, S. N. et al. (2017) [278] presented an architecture dubbed ResNeXt constructed by repeating multi-branch building blocks of the same topology. This architecture explores a new network dimension called cardinality, which refers to the total number of paths within a building block, as shown in the left diagram in Fig. 29. Experiments in [278] showed that the model capacity could be increased more effectively by increasing the cardinality of ResNeXt than the network depth or width.

Similar to the Inception modules, the building block of ResNeXt adopts a split-transform-merge strategy. The main difference is that each path representing a transformation in a building block of ResNeXt is of the same topology, as illustrated in Fig. 29(a). A set of transformations of the same topology are first aggregated within the building block. A skip connection is added to summate aggregated and input feature maps.

Fig. 29(b-c) shows two equivalent formulations of the building block. ResNeXt comprises a convolution layer followed by a max-pooling layer at the beginning, a stack of multi-branch building blocks, and a global average pooling layer followed by a fully-connected layer. Compared to a ResNet-50, a ResNeXt-50, shown in Fig. 29(c), has slightly fewer parameters while achieving better model performance. Compared to the Inception modules, it only needs to set a few hyper-parameters about the

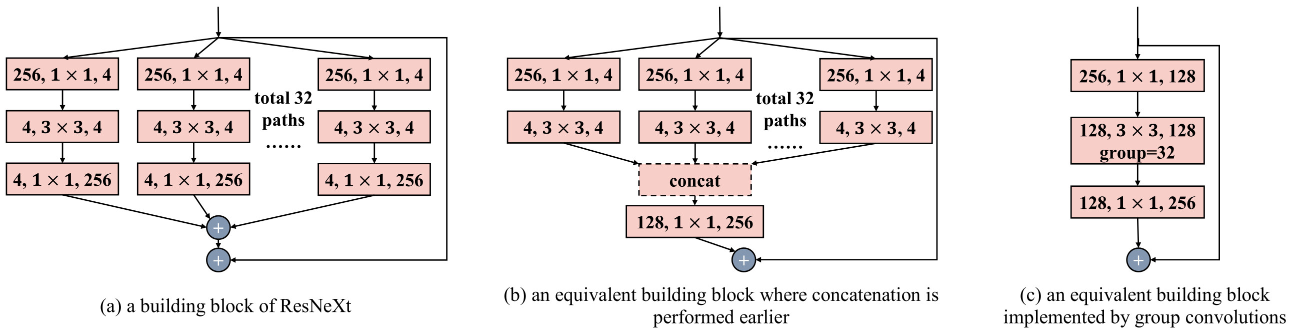


Fig. 29. Equivalent implementations of building blocks of ResNeXt with cardinality = 32. Each layer is represented in a box as (input channels, filter size, and output channels).

building blocks as they are of the same topology.

8.5. *ShuffleNet and NASNet*

Despite the tremendous success achieved by CNNs, most models are huge in size and of a large number of parameters, making it unfeasible to deploy them on mobile devices with very limited computational resources. ShuffleNet is a computation-efficient CNN architecture designed to pursue the best accuracy with limited computational budgets [279]. This architecture greatly reduces computational costs while maintaining model performance by utilizing two operations: group-pointwise convolution and channel shuffle.

Although depthwise separable convolutions and group convolutions have been widely used to replace standard convolutions in state-of-the-art extremely small networks such as ResNeXt to trade off model performance and computational cost, the expensive 1 × 1 convolutions are not taken into account, which introduces considerable computation complexity. In ShuffleNet, pointwise group convolutions are performed on 1 × 1 layers to reduce the computation complexity [280,281].

Fig. 30(a-c) illustrates a channel shuffle operation between two stacked group convolutions. Fig. 30(a) demonstrates two stacked group convolutions where the outputs of a group only relate to that certain group of channels in the input. As the outputs of different groups are independent, the side effect is that information flow is blocked among channels. This is solved by shuffling channels between two group convolutions, as shown in the middle and the right diagram of Fig. 30(b and c).

The shuffle channel operations allow information to flow across feature channels of different groups. The ShuffleNet unit is designed based on the channel shuffle and group convolution, as shown in Fig. 30 (d). This building unit has two branches. One branch is a shortcut path equipped with an average-polling layer. The second branch composes of a 1 × 1 pointwise group convolution whose output channels are shuffled, a 3 × 3 depthwise convolution followed by a 1 × 1 pointwise group convolution.

Feature maps of two branches are finally concatenated to form the output of a ShuffleNet unit. Pointwise group convolutions with channel shuffle allow the units to be computed efficiently and encode more information. The ShuffleNet architecture is constructed with a standard convolutional layer followed by a max-pooling layer, a stack of ShuffleNet units, and a global pooling layer followed by a classifier at the end of the network.

NASNet is an architecture with high transferability found on the CIFAR-10 dataset using neural architecture search methods [282]. Specifically, reinforcement learning is utilized to search for the best convolutional cells within a pre-defined search space that can be repeated to construct a NASNet network. Experiments in [282] demonstrated that a NASNet with the best architecture of convolutional cells found on the CIFAR-10 dataset can outperform contemporary published works on ImageNet.

NASNet composes of two types of generic convolutional cells that are repeated many times along the network: (1) Normal Cell that returns a feature map of the same dimension as the input; and (2) Reduction Cell that returns a feature map with the height and the width reduced by a factor of two. The architecture of the best Normal Cell and Reduction Cell found on CIFAR-10 are illustrated in Fig. 31(a and b).

The input of a convolutional cell comes from the outputs of two preceding cells, denoted as h_i and h_{i-1}, respectively. Each cell contains five blocks, each composed of two primitive operations (green) and a combination operation (blue). A primitive operation takes as input a hidden state from h_i, h_{i-1} or the hidden states generated from other blocks. The number of convolutional cells and the number of initial convolutional filters in a NASNet network are manually predetermined according to the scale of a classification problem and the computational demands. This property allows for generating a set of models that meet different computational budgets.

8.6. *MnasNet and EfficientNet*

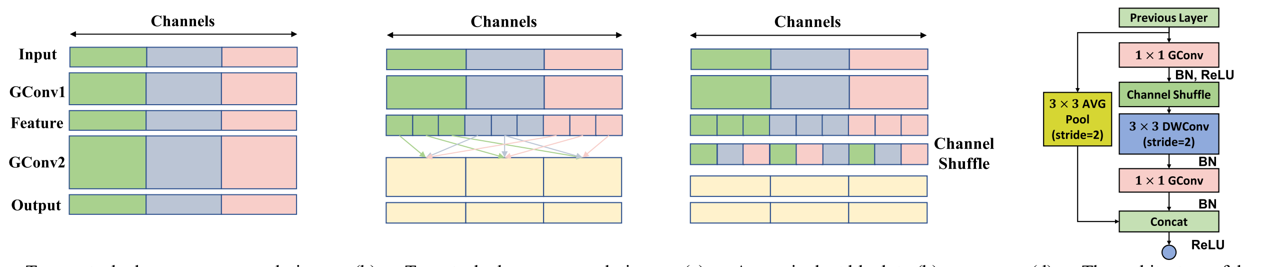
Significant efforts can be seen to balance the trade-offs between accuracy and latency for mobile models either by manually designing the network architecture or by neural architecture search methods [275, 278,279,282]. In contrast to these works, Tan, M. X. et al. (2019) [283] presented an automated mobile neural architecture search (MNAS) approach that directly incorporates real-word inference latency measured on mobile devices into the search objective to identify an optimal model striking a proper balance between accuracy and latency.

MNAS defines a factorized hierarchical search space where a CNN is partitioned into a series of unique pre-defined blocks, and operations and connections are searched for per block based on a set of choices. This encourages layer diversity which is beneficial to achieving high accuracy and less latency. Reinforcement learning searches for optimal solutions in the hierarchical search space. Especially a recurrent neural network (RNN) acts as a controller in the search framework that samples a set of models from the search space. Accuracy and latency for each sampled model are obtained by training and then running it on mobile phones. These two metrics are used to calculate the multi-objective reward for updating the controller's parameters, finding an optimal model maximizing the expected reward.

A representative architecture MnasNet-A1 found by the automated approach is illustrated in Fig. 32(a). Its components, MBConv3, MBConv6, and SepConv, are shown in Fig. 32(b-d). MnasNet-A1 consists of a standard convolutional layer, a sequence of blocks of three different structures, and a classifier. Layer diversity in MnasNet leads to better trade-offs between accuracy and latency than other models constructed by stacking a single type of layer throughout the network [283].

The conventional practice of scaling CNNs for performance improvements involves manually scaling the network depth, width, and resolutions. However, this process needs tedious manual tuning to strike a good balance between accuracy and efficiency.

Tan, M. X. et al. (2019) [284] first empirically quantified the relationship exiting among network depth, width, and resolution and proposed an effective compound model scaling method that uniformly scales these three dimensions of a given network, leading to accuracy



(a) Two stacked group convolutions without cross-group information flow (b) Two stacked group convolutions with cross-group information flow (c) An equivalent block to (b) (d) The architecture of the ShuffleNet unit

Fig. 30. Illustration of ShuffleNet. (a-c) Channel shuffle for group convolutions, (d) Architecture of the ShuffleNet unit.

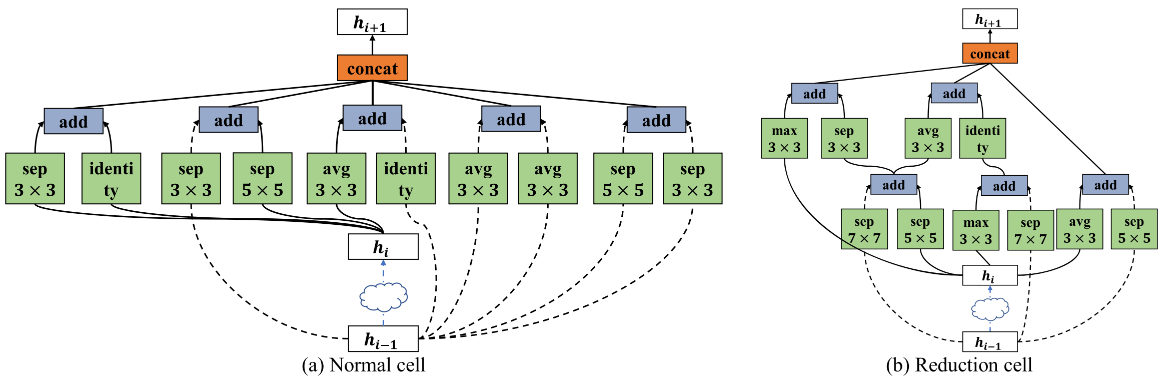


Fig. 31. Architecture of the best convolutional cells found on CIFAR-10.

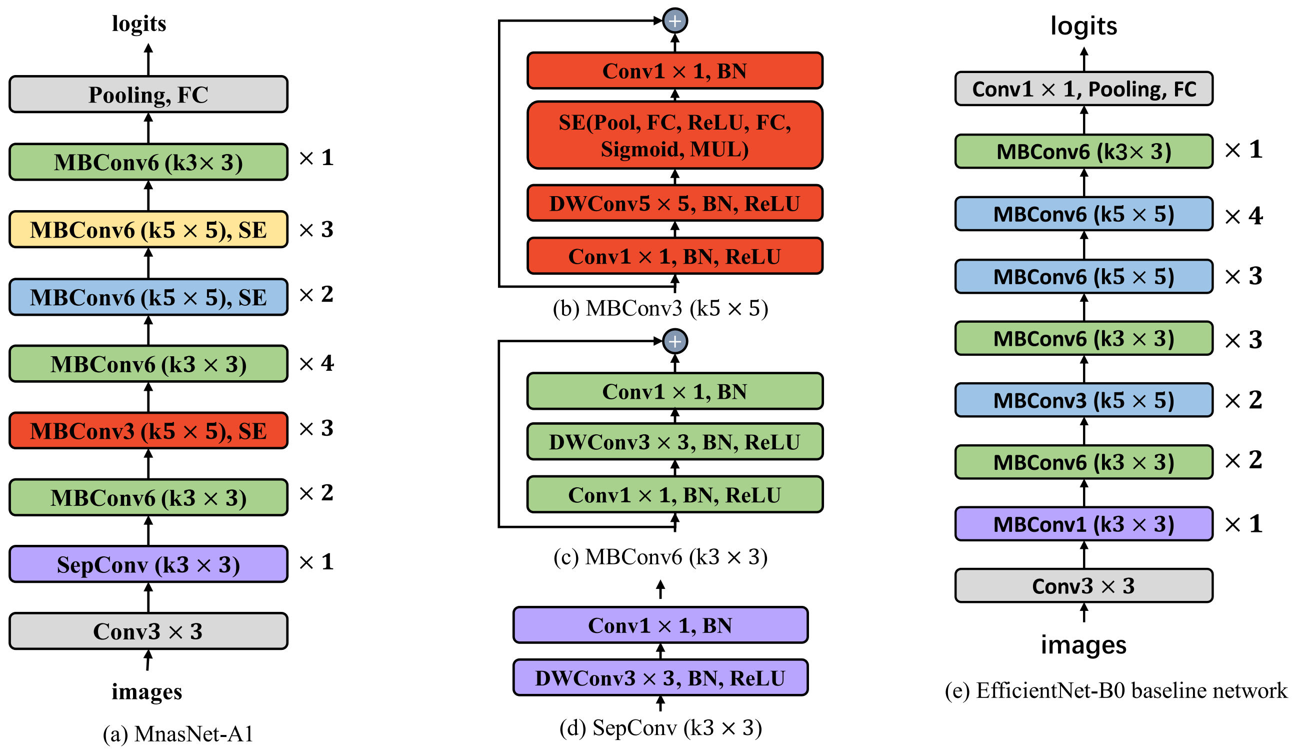


Fig. 32. Illustration of (a) MnasNet-A1 architecture, (b-d) MBConv3, MBConv6, and SepConv, (e) EfficientNet-B0 baseline network.

gains. The compound model scaling method can be formulated as $d = \alpha^{\phi}$, $w = \beta^{\phi}$, $r = \gamma^{\phi}$, $\alpha \geq 1$, $\beta \geq 1$, $\gamma \geq 1$, where d, w, r are depth, width, and resolutions, α, β, γ are constants determined by grid search, and ϕ denotes the compound coefficient to uniformly scale three dimensions. EfficientNet is a mobile-size baseline network on which the effectiveness of the compound model scaling method is evaluated. This architecture is found by a multi-objective neural architecture search within the same search space in Ref. [283]. The architecture of the baseline network EfficientNet-B0 is shown in Fig. 32(e).

Similar to MnasNet, EfficientNet-B0 composes of a convolutional layer at the beginning, a sequence of blocks, and a final convolutional layer followed by a pooling and a classifier [285,286]. Based on the amount of additional available resources, the baseline network is scaled up with different values of coefficient ϕ, generating a family of networks called EfficientNets. The scaled network EfficientNet-B7 performed on par with GPipe [287] on ImageNet with 8.4x fewer parameters and 6.1x lower inference latency.

9. Semi-supervised learning

Over the last decade, deep learning-based techniques have been developed in a wide variety of domains, and it is common sense that deep learning models heavily rely on large-size datasets with ground-truth labels [288]. However, assigning labels for all the datasets is laborious and costly, especially in some domains which need sufficient domain knowledge to annotate the labels for the unlabeled data [289,290]. For example, in medical image analysis, the annotation process needs lots of annotators to label the image and annotators with confident domain knowledge to ensure the correctness of assigned labels [291,292]. Many efforts try to address this problem with semi-supervised learning (SSL)-based methods.

SSL is a paradigm that can retrieve semantic knowledge from limited labeled data and unlabeled instances. For all the SSL-based methods, all the data should contain underlying semantic knowledge or features [293,294]. Then the SSL-based methods can utilize all the labeled or unlabeled data points to extract meaningful data features. If some data instances cannot provide underlying semantic knowledge, these data points are useless for the SSL-based methods [295,296].

Based on this obligatory condition, the SSL-based methods can easily obtain related algorithms and model architectures from existing supervised learning and unsupervised methods. Recently, the SSL-based method has attracted lots of attention [297]. It provides a promising way to explore semantic knowledge from unlabeled examples and alleviates the situation that some domains only have limited labeled images [298,299].

In the problem setting of semi-supervised learning, we assume that there are two group datasets: labeled and unlabeled datasets denoted as $D = \{D_L, D_U\}$. Here, the D represents the entire dataset, $D_L = \{x_i\}_{i=1}^{L}$ denotes limited labeled data with ground-truth labels $Y_L = \{y_i\}_{i=1}^{L}$, and $D_U = \{x_i\}_{i=1}^{U}$ is unlabeled data. By comparing the quantities of labeled and unlabeled instances, we assume $L \ll U$ that the quantity L of labeled data is far less than U of the unlabeled data. During the training process of SSL, we aim to minimize the loss function as Eq. (62) shows,

$$\mathscr{L}_{ssl} = \sum_{x \in D_L, y \in Y_L} \mathscr{L}_s(x, y, \theta) + \alpha \sum_{x \in D_U} \mathscr{L}_u(x, \theta) + \beta \sum_{x \in D} \mathscr{R}(x, \theta). \tag{62}$$

It consists of three parts: supervised loss $\mathscr{L}_s$ for each example, unsupervised loss $\mathscr{L}_u$ and a regularization term $\mathscr{R}$. θ represents the trainable parameters, and $\alpha, \beta \in \mathbb{R}$ are larger than zero.

There are two evaluation criteria for SSL-based methods. The first one requires the SSL-based method should outperform the supervised learning methods with limited labeled data. Secondly, the performance improvement corresponds to the increased portion of labeled training instances. In conclusion, both improved gaps should show the effectiveness of the proposed SSL-based approach [300].

SSL mainly includes smoothness assumption and manifold assumption (Also known as cluster assumption, structure assumption, or low-density separation assumption). In the last decade, SSL-based methods have succeeded dramatically in many domains. By analyzing most SSL-based methods, we divided existing works into four categories, as Fig. 33 (a) shows: generative models, consistency regularization methods, pseudo-labeling methods, and graph-based methods. We will explain the concepts of these categories in the remaining subsections and illustrate the related works of the corresponding method.

9.1. *Generative models*

Generative models belong to unsupervised learning, which aims to extract the underlying data distributions from unlabeled data [301]. The outputs of the generative model are representations of underlying data distribution (The representation can be a data feature, label, or new instances), then integrating these outputs with a supervised learning-based method to build an SSL framework. With the development of deep learning techniques, we summarize all the generative models into four types, as Fig. 33(b) shows: generative adversarial networks (GANs), variational auto-encoders (VAEs), diffusion models, and flow-based models. This chapter mainly introduces the concept of GANs, VAEs, and diffusion models. These methods are commonly used in many applications.

9.1.1. *Generative adversarial networks (GANs)*

GANs is first proposed by Goodfellow, I. et al. (2020) [302], aiming to map the data into high-dimensional distributions implicitly. It can be used in both semi-supervised and unsupervised learning-based methods. Since it was proposed in 2014, it has been applied in many different domains. Moreover, it has already developed in many different tasks, such as image restoration [303], image-to-image translation [304], image generation [305], image editing [306], domain transfer [307], and so on.

The classic generative adversarial network contains two parts named as discriminator and generator. The generator denotes by G, which aims to generate fake instances from gaussian noise $\mathbf{z}$. The discriminator denotes by D, and the inputs of the discriminator are original or synthetic images. The goal of the discriminator is to distinguish whether the input is a real or fake image. When the images generated by the generator cannot distinguish by the discriminator, we define that the generator is well-trained and has enough ability to generate synthetic images. The entire training process of the generative adversarial network can be regarded as a min-max two-person zero-sum game. The generator attempts to minimize the value V, and the discriminator attempts to maximize the value V. The objective of the game can be described as Eq. (63),

$$\min_{\theta} \max_{\varphi} V(G_\theta, D_\varphi) = \mathbb{E}_{\mathbf{x} \sim \mathrm{P_{data}}} [\log D_\varphi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim \mathrm{p(z)}} [\log(1 - D_\varphi(G_\theta(\mathbf{z})))], \tag{63}$$

where the generator tries to decrease the probability of the discriminator correctly classifying the fake images. This probability represents by the $\log[1 - D_\varphi(G_\theta(\mathbf{z}))]$ with respect to its parameters θ. During the training process, the discriminator will try to improve the ability to get the correct predictions denoted by the $\log D_\varphi(\mathbf{x})$ with respect to its parameters φ, and $\mathbf{x}$ represents the real data instance. The general architecture of the generative adversarial network depicts in Fig. 34(a).

Fig. 34(b) shows some food examples generated by ChefGAN [308]. These generated images can be regarded as an additional training dataset to improve the performance of the SSL-based method. Another work presented by Fei, Z. H. et al. (2021) [309] uses CycleGAN to predict the position and geometry of fruits. Their method can boost the domain adaption process and reduce labeling costs.

9.1.2. *Variational auto-encoders (VAEs)*

VAE is one type of generative model that integrates autoencoders with generative latent-variable models to reconstruct the input data instances [310–312]. The typical VAEs include two parts: the encoder and the decoder. The encoder will transform the input data x to the latent feature representation z, then the decoder will reconstruct the original input x to the x'. Fig. 35(a) depicts the standard architecture of VAEs. It trains the model by maximizing the variational lower bound.

Here, we denote the encoder $q_\theta(zx)$ with parameters θ, and decoder $p_\phi(xz)$ with parameters ϕ. Then the loss function of VAEs can be

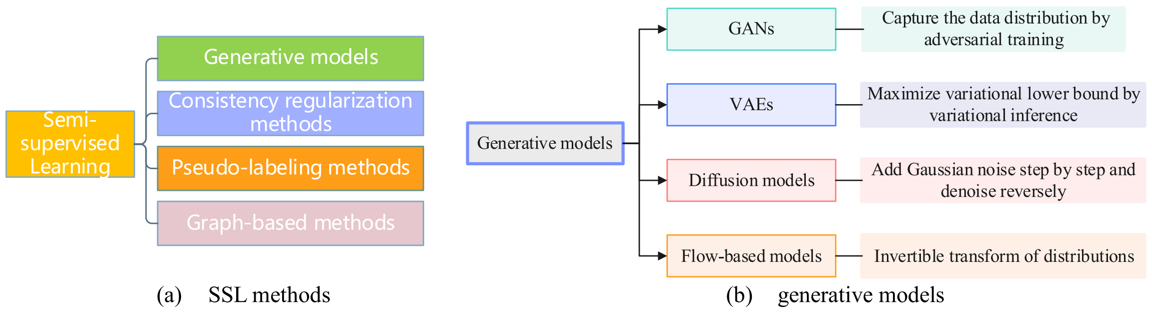
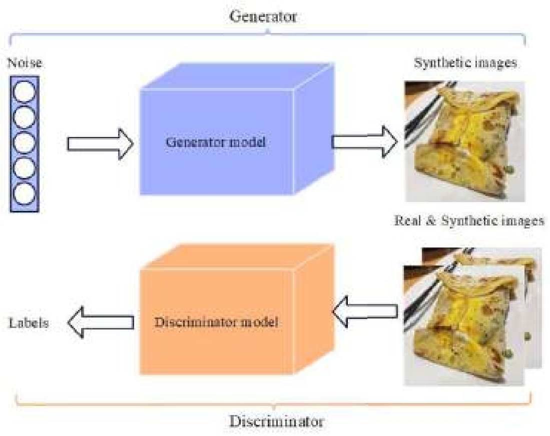

Semi-supervised Learning
Generative models
Consistency regularization methods
Pseudo-labeling methods
Graph-based methods
(a) SSL methods
Generative models
GANs
Capture the data distribution by adversarial training
VAEs
Maximize variational lower bound by variational inference
Diffusion models
Add Gaussian noise step by step and denoise reversely
Flow-based models
Invertible transform of distributions
(b) generative models

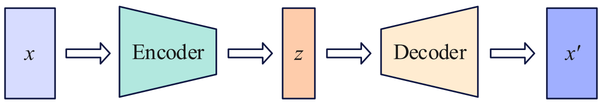
Fig. 33. The taxonomy of (a) SSL methods and (b) generative models.



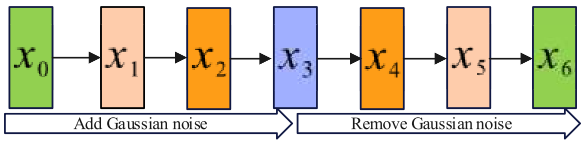
(a) The architecture of GAN

(b) Examples generated from ChefGAN [308]

Fig. 34. Illustration of GAN.



(a) VAE



(b) diffusion model

Fig. 35. The architectures of VAE and diffusion model.

described as Eq. (64), which represents a negative log-likelihood with a regularization term. In Eq. (64), $-\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)]$ is a negative log-likelihood for the data point x_i, it aims to force the decoder to reconstruct the input data. The $\mathbb{KL}(q_\theta(z|x_i)\|\,p(z))$ is a Kullback-Leibler divergence which evaluates the loss between the encoder q and the decoder p.

$$l_i(\theta,\phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] + \mathbb{KL}(q_\theta(z|x_i)\|\,p(z)) \tag{64}$$

9.1.3. Diffusion models

The diffusion model formulates a new paradigm in generative models and shows an impressive performance in computer vision tasks. The diffusion model consists of two parts, as Fig. 35(b) shows: forward and reverse diffusion. In the forward diffusion part, the original data x_0 will gradually be perturbed by adding gaussian noise within different steps. In the reverse diffusion part, the model will try to gradually recover the data feature by removing the gaussian noise from the original data. Since introducing the diffusion model, it has been widely used for generating various high-quality data samples. However, the inference time of the diffusion model is very slow due to the huge number of steps during the sampling process.

The diffusion model has three categories shown in Table 7. The diffusion models have been applied in different computer vision tasks, such as image generation [313,314], image-to-image translation [315], and image editing [316]. Moreover, the latent data features generated by the diffusion model can also be used in discriminative tasks like classification, segmentation, and object detection.

Table 7
Three categories in diffusion models.

Abbreviation	Term	Role
DDPMs	Denoising diffusion probabilistic models	Use latent variables to predict the probability distribution.
NCSNs	Noise-conditioned score networks	Use score matching to train a shared model to evaluate the score function for the perturbed data distribution with different noise levels.
SDEs	Stochastic differential equations	A generalization over denoising diffusion probabilistic model and noise-conditioned score network

9.2. Consistency regularization methods

The manifold and smoothness assumption are vital foundations of consistency regularization methods. The loss function of the consistency regularization includes a consistency regularization term which indicates the pre-defined constraints. The basic rationale behind the consistency regularization term is that the model outputs are consistent when the noise or perturbation is added to the same data instances or models [317]. There mainly includes two categories of consistency regularization methods: input variations and model variations. An intuitive illustration of these methods is shown in Fig. 36(a and b).

Input variations focus on generating different augmented data based on the same input instance. Then apply the consistency regularization with different versions of the same inputs. Here, we mainly introduce five different types of input variations: basic image manipulations, random augmentation, adversarial perturbation, MixUp, and automated augmentation [300]. The most basic strategy is applying different image manipulations, such as flipping, color augmentation, random cropping, translation, and noise injection.

Apart from these basic data manipulations, random augmentation is also a widely used data augmentation technique. It adds the gaussian noise first, then applies jittering to the input data. The representative work using the random augmentation is Π-model [318], which uses the random data augmentation on the input data and then decreases the consistency regularization of the outputs from the two models. Adversarial perturbation aims to change the perditions of the model by adding adversarial noises to the input data, and adversarial noise can help SSL methods to explore the underlying semantic knowledge of unlabeled data. Adding adversarial noise can change the predicted labels or the predictive confidence [319]. MixUp [176] produces a weighted combination of different image pairs. Given two data instances and their labels denoted by (x_n, y_n) and (x_m, y_m), a weighted combination of image

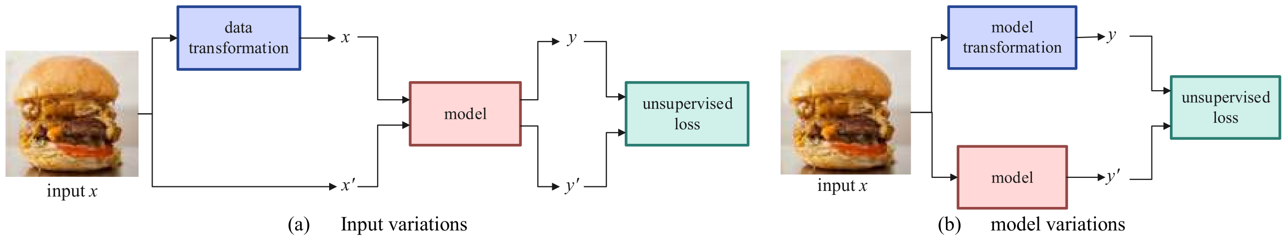


Fig. 36. Consistency regularization methods based on (a) the input variations and (b) model variations.

pairs is described as Eqn. (65)

$$\begin{cases} \hat{x} = \theta x_n + (1-\theta)x_m \\ \hat{y} = \theta y_n + (1-\theta)y_m \end{cases}, \tag{65}$$

where the $(\hat{x}, \hat{y})$ denotes the new synthetic training example. Automated augmentation can explore the data itself to generate potential data augmentation strategies and produce strong-augmented data instances. It can efficiently reduce the cost of designing domain-specific data augmentation [320–322].

Model variants apply consistency regularization by different model designs, such as parameter settings, model architectures, and model weights. Here we introduce two types of methods for the model variants: stochastic perturbation and ensembling.

Stochastic perturbation modifies the model's parameters by adding perturbed noise or dropout layers [323]. For instance, Ladder Network will add gaussian noises in the training process [324], Pseudo-Ensemble [323], and Virtual Adversarial Dropout [325] use dropout to minimize the consistency regularization term.

Ensembling methods try to get consistent predictions from different models and data augmentations. For example, Temporal Ensembling [326] and Mean Teacher [327] compute the exponential moving average (EMA) by concatenating different models during the training process.

Consistency regularization methods can retrieve useful information from unlabeled data by maintaining the prediction variants under the different input variants and model variants. During training, the model can learn more robust feature representations by optimizing the consistency regularization loss. It only needs variations added to the training data instead of cost on the annotations.

9.3. *Pseudo-labeling methods and graph-based methods*

Pseudo-labeling methods try to generate pseudo-labels for the unlabeled data, which can be regarded as additional training data with labels. These methods differ from consistency regularization methods, which aim to get consistent predictions based on various constraints, but the pseudo-labeling methods highly rely on our confidence in pseudo-labels. There are two types of pseudo-labeling models: disagreement-based models and self-training models.

Disagreement-based models will train several learners to exploit the unlabeled data and try to maintain the max disagreement among the different learners [328]. Some works are based on the disagreement-based mechanism, such as Co-training [329] and Tri-training [330]. Co-training [329] assumes that each data instance has two complementary views, and these views for all the data instances are enough to train two classifiers. Then, two classifiers are used to predict unlabeled data on both views and give the pseudo-labels to the most confident instance until all the unlabeled data are iterated. Tri-training trains three classifiers with three training datasets generated by the bootstrap sampling. In detail, Tri-training [330] uses labeled data with output smearing [331] to generate three training datasets, then train three initial classifiers with different training datasets. Pseudo-labels were obtained by making predictions on these classifiers with unlabeled data. The pseudo-label is regarded as confident when the consistency of two models checks it, and the confident instance adds to the training dataset of the third model. The third model uses an augmented dataset to predict the final outputs.

Self-training models generate pseudo-labels by training an unsupervised learning-based classifier and then assigning the pseudo-labels for unlabeled instances. These models can be regarded as a data augmentation process to expand existing limited labeled images. One of the self-training models is Noise Student [332], which contains a teacher and student model based on knowledge distillation. Firstly, the teacher model trains on the labeled data and produces pseudo-labels for all the unlabeled data. Then a larger student model will combine the labeled and pseudo-labeled data for training.

During the training process, multiple data augmentations are implemented to improve the robustness of the model. Another work of self-training models is EnAET [333], which ensembles different Auto-Encoding Transformations to improve the model's performance. The well-trained EnAET model can generate meaningful feature representations, which can generate pseudo-labels for unlabeled data. Apart from the methods mentioned above, contrastive learning-based methods can also generate pseudo-labels to utilize unlabeled data, such as SimCLR [334], SimSiam [335], and DINO [336].

Graph-based methods assume that data instances can be mapped into a weighted graph where the graph nodes represent the data instance, and the edge describes the similarity between the nodes. In detail, if all the labeled and unlabeled samples are mapped into the weighted graph, the labels of unlabeled data can predict by the positions of labeled data because the neighboring data points are more likely to have the same label based on the smoothness assumption.

Graph-based methods can be divided into two types: transductive learning and inductive learning [337]. Transductive learning methods propagate the labels for unlabeled data by exploring the feature representation on the labeled data. Related works for transductive learning are Gaussian Fields and Harmonic Functions (GFHF) method [338], the Local and Global Consistency (LGC) algorithm [339], sparse representation-based methods [340], low-rank model [341], and so on.

Inductive learning methods using labeled and unlabeled data to train the classifier also work for the out-of-samples, i.e., it initially builds a graph based on the labeled and unlabeled data, then propagates the label of the out-of-sample (unseen data) based on the existing graph. Related works of inductive learning methods are BGDH [342], SSDH [343], and AELP-WL [344].

10. Deep-learning applications in food category classification

Food is intimately connected to human health; however, food recognition can often be time-consuming and expensive. Applying deep learning to food recognition can effectively alleviate this issue, making food recognition more accessible and less costly. This section reviews research on food image recognition based on deep learning from three perspectives: food category classification, food quality identification, and food ingredient detection.

10.1. Food category classification and quality identification

Accurate and rapid classification of food products can effectively promote an automated food industry, encourage healthy eating habits, and assist patients with illnesses requiring specific food contraindications or individuals with particular food allergies in identifying foods quickly. With the rapid advancement of technology, electronic devices capable of displaying images are present in numerous aspects of human life and production, and the acquisition of food images has become fast and affordable. The abundant information in image data can reflect color, shape, texture, volume, and various other characteristics that aid in classifying food products. It makes images an important form of data for food classification and is widely employed in research on food classification tasks. Compared with traditional image classification methods, deep learning-based food image classification techniques can automatically extract complex features from food images, significantly reducing the dependence of image-based food classification on food experts. Consequently, numerous deep learning-based food image classification methods have been proposed recently.

McAllister, P. et al. (2018) [345] used two different pre-trained CNN models (ResNet-152 and GoogleNet Inception) to extract features from food images and five common classifiers (Naive Bayes, SVM polynomial, SVM RBF, ANN, and Random Forest) to classify the extracted features for different kinds of food. The datasets used for the experiments are Food-5k, Food-11, RawFooT-DB, Food101, UNICT-Caltech, where UNICT-Caltech is only used for model evaluation, and the other datasets are also used for the training of the model. By mixing and matching CNN models for feature extraction with machine learning classifiers, training and evaluating these models on different datasets, several sets of experiments were constructed to comprehensively assess the feasibility of such methods for food classification tasks. Among all the experiments conducted, the combination of ResNet-152 for feature extraction and ANN for feature classification achieved the highest performance with a 98.8% accuracy in the binary classification (Food class and Non-Food class) on the Food-5k dataset. This combination also achieved accuracies higher than 90% in most other experiments. The study demonstrates that combining pre-trained models with machine learning classifiers is practical in food image classification tasks.

Based on cognitive uncertainty analysis, Aguilar, E. et al. (2022) [26] proposed a food image classification method called Forward Step-wise Uncertainty-Aware Model Selection (FS_UAMS). This approach selects the optimal combination of CNN models for optimal performance while avoiding the high computational resource requirements associated with traditional model selection methods such as random or exhaustive search. They replaced the outputs of two classical CNN models, ResNet 50 and InceptionV3, with a missing layer with a probability of 0.5 and an output layer with softmax activation. Moreover, the integrated learning models constructed based on the two modified models were trained and evaluated using three food image datasets (MAFood121, UECFood256, and VIREO172). The research experiments compared the proposed method with traditional model selection methods (validation set-based and random selection methods). The experimental results showed that the proposed method performs similarly or better than the traditional method on all three datasets, achieving 73.01% accuracy on UECFood256, 89.26% on VIREO172, and 88.95% accuracy on MAFood121. The validation set-based model selection method obtained a slightly higher performance on only UECFood256 (73.10%).

Yu, Z. Y. et al. (2021) [346] proposed a novel method for identifying hybrid okra seeds by combining deep learning techniques with hyperspectral imaging. Their study is divided into two aspects, one is used to classify mixed okra seeds, and the other is to explore the effect of different okra varieties on the model's performance. The experiment used a database from the Zhengjiang Institute of Agricultural Science in China to collect 18 species of okra seeds with 18,931 stable states in 2017. Their experiments compared two deep learning neural network models, CNN and stacked sparse auto-encoder (SSAE), and two traditional machine learning, extreme learning machine (ELM) and back propagation neural network (BPNN). The experiments showed that both deep learning methods exhibit higher precision performance than traditional machine learning methods. More specifically, a CNN model achieves the best performance among all experiments in the research with a test accuracy of 93.79%.

Zhang, W. D. et al. (2020) [347] proposed a framework called a wide hierarchical subnetwork-based neural network (WI-HSN) to classify food types based on images based on a supervised subnetwork model of feature encoding and pattern classification. The framework consists of two subnetwork neural nodes (SNN), an entry SNN and an exit SNN. With one entry SNN and one exit SNN as the initialization structure, new entry SNNs and exit SNNs are added gradually with iterations to ensure the model is suitable for tasks of different complexity. In addition, the authors also proposed a batch-by-batch scheme with a parallelism learning strategy to reduce the learning cost. The proposed method is trained and evaluated on five different food image datasets (UEC-Food100, Food101, Food251m, Food251a, and UECFood256). Compared with other common classifiers, WI-HSN achieved the best performance in classification tasks on several datasets, with the highest performance on the Food101 dataset with an accuracy of 90.8% and an average accuracy of 78.2% across the five datasets.

Food quality identification is a crucial task that significantly impacts human dietary health. However, traditional food quality testing methods often have drawbacks, the most important of which is the high cost. It makes food quality detection a challenge in the food industry when the number of food samples has increased dramatically. The automatic feature extraction nature of deep learning techniques makes it possible to obtain accurate food quality inspection results at a relatively low cost. Therefore, many studies have proposed various deep learning-based food quality inspection methods.

Zhou, J. et al. (2023) [5] proposed a deep learning framework based on double-layer rough-refinement optimization to achieve standardized and consistent food quality identification. The proposed model introduces the idea of multi-objective optimization to optimize the topology of the deep neural network and uses a meta-heuristic algorithm to optimize the global weight parameters of the neural network. They trained and evaluated the model on a wine quality dataset from the UCI machine learning database and a rice cake dataset. The wine quality dataset covers two types of wine (white wine and red wine). Each sample in the wine quality dataset has 11 physicochemical properties as features and sensory data based on the sommelier's rating on a ten-point scale (0 for very poor and 10 for very good) as the label. The rice cake dataset includes six raw material indicators and six product indicators. The proposed method has achieved accuracies of 93.73% for the red wine quality identification task, 89.77% for the white wine quality identification task, and an $\overline{\mathrm{MSE}}(10^{-2})$ (the average mean squared error divided by 10^{-2}) of 0.010 for the task of rice cake quality identification. The experimental results showed that the proposed method is competitive with the traditional food quality evaluation methods and has general applicability for the quality evaluation of different food products.

Kazi, A. et al. (2022) [6] used two traditional structures of CNN (AlexNet and VGG-16) and a 50-layer residual CNN (Resnet-50) to classify a six-class (fresh apples, fresh bananas, fresh oranges, rotten apples, rotten bananas, and rotten oranges) fruit image dataset consisting of three different species (bananas, apples, and oranges) and two different states (fresh and rotten). The research experiments showed that the Resnet-50 model performs best in classifying fruit images with an accuracy of 99.7%, compared to the AlexNet model (97.74%) and the VGG-16 model (99.3%). The authors suggested that this is because the shortcut design of the residual network allows deeper network layers, which can effectively avoid the negative impact of a large number of network layers on the overall performance while learning more and deeper features.

Pradana-Lopez, S. et al. (2021) [7] used a pre-trained ResNet-34 based on transfer learning techniques to perform the food quality identification task on a custom coffee image dataset. The objective is to detect the presence of an impurity in these coffees and the ability of the deep learning model to discriminate between adulterants of different particle sizes to control the coffee quality. The proposed method can effectively identify adulterations of ground Arabica and Robusta coffee with chicory and barley, with an accuracy of 98.6%.

10.2. Food ingredients detection

Processed foods often contain a wide range of ingredients, and even the same dish may have different combinations. Identifying food ingredients is important for human health, not only to ensure a balanced diet but also to consider the sensitivity of specific populations to particular foods. Ideally, food ingredients can be quickly identified by acquiring food images from electronic devices such as cell phones or computers. Deep learning, a technique that automatically extracts features and quickly outputs detection results, is widely used for food ingredient detection.

Tahir, G. A. et al. (2021) [348] proposed a food image analysis method based on transfer learning by combining MobileNetV3 and snapshot ensembles. The framework of the model is divided into two parts, first identifying whether the input image is a food image and then performing food ingredients detection on the identified food image. This framework uses transfer learning to reduce the computational cost and the chance of generating generalization errors. The distributed structure of the framework utilizes the association between two sub-problems to effectively solve the drawback that transfer learning can only solve a single problem but not an associated problem. Their research used five image datasets, Food/Non-Food, Food101, UECFood100, UECFood256, and Malaysian food, in which the Malaysian food dataset was extended to a new dataset (MF-145) for ingredient detection. The experimental results showed that the ensemble model outperformed or was similar to the three separate models on all five data sets. The proposed ensemble model achieves up to 99.12% accuracy on the Food/Non-Food classification task. It achieves a Hamming loss of only 0.0070 on the food ingredients detection task, which is only 0.0004 higher than MobileNetV3 but achieves a precision of 83.81%, 3.16% higher than MobileNetV3.

Chen, J. J. et al. (2021) [349] comprehensively analyzed two food composition recognition methods, (1) a multi-task learning-based method that uses global features in food images to identify both food types and ingredients, (2) a method that uses regional features in food images to identify food ingredients, firstly by identifying ingredients in local images and then by pooling the recognition results from different regions as the final recognition results. The results showed that using multi-scale image processing to compensate for the loss of image context is not very effective. In addition, the recognition of food types is based on image-level features. In contrast, the recognition of food ingredients is mostly based on local-level features, and these two different levels of features conflict with each other regarding learning objectives. The multi-task learning-based approach requires complex network structures to optimize both problems to prevent performance degradation. In addition, the paper extended the Chinese cuisine dataset VIREO172 with 172 categories into a larger food data set with 251 categories and 406 food ingredient labels, named VIREO251.

Wang, Z. L. et al. (2022) [350] proposed a multi-task learning framework that simultaneously identifies food categories and predicts their composition. This framework uses a food composition dictionary to acquire visual regions in an image related to food composition and an attention mechanism to enhance the features of these visual regions. In addition, a graph of semantic and visual representations of food ingredients was constructed to model the relationship between food ingredients. The nodes are the visual components, and the edges are the semantic similarities between the component words. A graph convolutional neural network was used to learn the constructed graph and to aggregate semantic features with visual features. Three food datasets were used: the Western food dataset Food101, the Chinese food dataset Vireo Food-172, and the mixed food dataset ISIA Food-200. The effectiveness of the proposed approach was further demonstrated using component assignment graphs and attention graphs.

Table 8 shows recent research on applying deep learning techniques to various food image recognition tasks. The best evaluation performance column records only the best performance for the corresponding

Table 8
A survey on deep learning-based food recognition methods.

Authors	Year	Task Type	Dataset	Best Evaluation Performance
McAllister, P. et al. (2018) [345]	2018	Food Category Classification	Food-5k	ACC: 98.8% (Binary)
			Food-11	ACC: 91.34% (11-class)
			RawFooT-DB	ACC: 99.28% (68-class)
			Food101	ACC: 64.98% (101-class)
			UNICT-Caltech	ACC: 97.50% (Binary)
Aguilar, E. et al. (2022) [26]	2022	Food Category Classification	MAFood121	ACC: 88.95% (121-class)
			UECFood256	ACC: 73.10% (256-class)
			VIREO172	ACC: 89.26% (172-class)
Yu, Z. Y. et al. (2021) [346]	2021	Food Category Classification	Custom okra seeds	ACC: 93.79% (18-class)
Zhang, W. D. et al. (2020) [347]	2020	Food Category Classification	UECFood100	ACC: 87.7% (100-class)
			Food101	ACC: 90.8% (101-class)
			Food251m	ACC: 61.6% (251-class)
			Food251a	ACC: 66.4% (251-class)
			UECFood256	ACC: 83.1% (256-class)
Zhou, J. et al. (2023) [5]	2022	Food Quality Identification	Wine Quality (Red Wine)	(Error Tolerance = 1.0) ACC: 93.73%
			Wine Quality (White Wine)	ACC: 89.77%
			Glutinous Rice Cake	$\overline{\text{MSE}}(10^{-2})$: 0.010
Kazi, A. et al. (2022) [6]	2022	Food Quality Identification	Custom fruit dataset	ACC: 99.7%
Pradana-Lopez, S. et al. (2021) [7]	2021	Food Quality Identification	Custom coffee dataset	ACC: 98.6%
Tahir, G. A. et al. (2021) [348]	2021	Food Ingredients Detection	MF-145	F1-score: 86.03% Hamming Loss: 0.0070
Chen, J. J. et al. (2021) [349]	2021	Food Ingredients Detection	VIREO251	Micro-F1: 83.06% Macro-F1: 72.00%
			UECFood100	Micro-F1: 68.95% Macro-F1: 49.36%
Wang, Z. L. et al. (2022) [350]	2022	Food Ingredients Detection	ISIA Food-200	Micro-F1: 64.74% Macro-F1: 62.61%
			Food101	Micro-F1: 91.51% Macro-F1: 90.82%
			VIREO172	Micro-F1: 74.34% Macro-F1: 59.56%

task of each research listed in the task type column. For some research articles that report performance on multiple subsets of datasets, only the performance on the evaluation dataset is listed.

11. Conclusion

Developing and deploying deep learning and AI systems to task food category recognition is a large field of potential innovation and impact in the 21st century. In this survey, we provided a survey of the current techniques and methods used for various applications in food category recognition, including food quality and ingredients detection and food type and quantity detection. The techniques and methods cover each stage of constructing an AI system, including data collection, data augmentation, hand-crafting features, and methods for recognition. Recognition, which is most formulated in classification tasks, can be achieved through both traditional machine learning and deep learning approaches. The survey lies in the intersection between deep learning, computer vision, and food science, providing an overview of existing methods and insights for developing new recognition systems.

We place a particular focus on the most commonly applied approach of transfer learning and the approach that will allow the utilization of potentially abundant unlabeled data: semi-supervised learning. For the open challenges detailed by Zhu, L. L. et al. (2021) [10] in a prior survey that focuses on food processing, our survey primarily focuses on the development of ever higher-performing computer vision algorithms for the more general task of food category recognition in all stages of the food "life-cycle."

The remaining open challenges include integrating sensory information, e.g., smell [10]. Sensory information may provide more direct information regarding food ingredients and quality than images, while this survey focuses on computer vision methods. AI systems' efficiency and size are other challenges we do not address. The survey focused on deep learning with deep neural networks, which may be difficult to integrate into mobile applications. The final challenge that remains to be addressed is the integration with robotics [10,24], the natural step forward when a reliable recognition system is available. Future development in the direction of these challenges has the potential to revolutionize the relationship between humankind and food.

CRediT authorship contribution statement

Yudong Zhang: Conceptualization, Methodology, Formal analysis, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Lijia Deng:** Conceptualization, Formal analysis, Resources, Data curation, Writing – original draft, Writing – review & editing. **Hengde Zhu:** Conceptualization, Investigation, Writing – original draft, Writing – review & editing. **Wei Wang:** Methodology, Software, Resources, Writing – original draft, Writing – review & editing. **Zeyu Ren:** Methodology, Software, Data curation, Writing – original draft, Writing – review & editing. **Qinghua Zhou:** Methodology, Validation, Writing – original draft, Writing – review & editing. **Siyuan Lu:** Methodology, Validation, Data curation, Writing – original draft, Writing – review & editing. **Shiting Sun:** Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Ziquan Zhu:** Methodology, Formal analysis, Data curation, Writing – original draft, Visualization. **Juan Manuel Gorriz:** Conceptualization, Investigation, Writing – review & editing, Visualization, Supervision. **Shuihua Wang:** Conceptualization, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgment

This paper is partially supported by MRC (MC_PC_17171), Royal Society (RP202G0230); BHF (AA/18/3/34220); Hope Foundation for Cancer Research (RM60G0680); GCRF (P202PF11); Sino-UK Industrial Fund (RP202G0289); LIAS (P202ED10, P202RE969); Data Science Enhancement Fund (P202RE237); Fight for Sight (24NN201); Sino-UK Education Fund (OP202006); BBSRC (RM32G0178B8).

References

[1] E.J.T. Collins, Food adulteration and food safety in britain in the 19th and early 20th centuries, Food Policy 18 (2) (1993) 95–109.
[2] R.W. Welch, et al., Food processing: A century of change, Br. Med. Bull. 56 (1) (2000) 1–17.
[3] S.B. Eaton, et al., Paleolithic vs. Modern diets - selected pathophysiological implications, Eur. J. Nutr. 39 (2) (2000) 67–70.
[4] J.W. Crandall, et al., Cooperating with machines, Nat. Commun. 9 (2018) 233.
[5] J. Zhou, et al., Deep learning networks with rough-refinement optimization for food quality assessment, Nat. Comput. 22 (2023) 195–223.
[6] A. Kazi, et al., Determining the freshness of fruits in the food industry by image classification using transfer learning, Multimed. Tool. Applic. 81 (6) (2022) 7611–7624.
[7] S. Pradana-Lopez, et al., Deep transfer learning to verify quality and safety of ground coffee, Food Control 122 (2021), 107801.
[8] M. Mishra, et al., Allergen30: Detecting food items with possible allergens using deep learning-based computer vision, Food Anal. Method. 15 (11) (2022) 3045–3078.
[9] G.A. Tahir, et al., A comprehensive survey of image-based food recognition and volume estimation methods for dietary assessment, Healthcare 9 (12) (2021) 1676.
[10] L.L. Zhu, et al., Deep learning and machine vision for food processing: a survey, Curr. Res. Food Sci. 4 (2021) 233–249.
[11] S. Rossner, Obesity: The disease of the twenty-first century, Int. J. Obes. 26 (2002) S2–S4.
[12] M. Di Cesare, et al., Trends in adult body-mass index in 200 countries from 1975 to 2014: A pooled analysis of 1698 population-based measurement studies with 19.2 million participants, Lancet 387 (10026) (2016) 1377–1396.
[13] A. Gulati, et al., Conformer: Convolution-augmented transformer for speech recognition, in: Interspeech Conference, Shanghai, PEOPLES R CHINA, 2020, pp. 5036–5040.
[14] T. Brown, et al., Language models are few-shot learners, Adv. Neur. Inform. Process. Syst. 33 (2020) 1877–1901.
[15] A. Krizhevsky, et al., Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[16] C. Holmberg, et al., Adolescents' presentation of food in social media: An explorative study, Appetite 99 (2016) 121–129.
[17] F. Pernollet, et al., Methods to simplify diet and food life cycle inventories: accuracy versus data-collection resources, J. Clean. Prod. 140 (2017) 410–420.
[18] X. Chen, et al., Chinesefoodnet: A Large-Scale Image Dataset for Chinese Food Recognition, arXiv e-prints, 2017, https://doi.org/10.48550/arXiv.1705.02743 arXiv:1705.02743.
[19] Y. He, et al., Analysis of food images: features and classification, in: IEEE International Conference on Image Processing (ICIP), Paris, FRANCE, 2014, pp. 2744–2748.
[20] C. Shorten, et al., A survey on image data augmentation for deep learning, J. Big Data 6 (1) (2019) 60.
[21] A. Loddo, et al., On the efficacy of handcrafted and deep features for seed image classification, J. Imaging 7 (9) (2021) 171.
[22] M.I.H. Khan, et al., Machine learning-based modeling in food processing applications: state of the art, Compreh. Rev. Food Sci. Food Saf. 21 (2) (2022) 1409–1438.
[23] S. Sonoda, et al., Neural network with unbounded activation functions is universal approximator, Appl. Comput. Harmon. Anal. 43 (2) (2017) 233–268.
[24] F. Rosenblatt, The perceptron - a probabilistic model for information-storage and organization in the brain, Psychol. Rev. 65 (6) (1958) 386–408.
[25] D.E. Rumelhart, et al., Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[26] E. Aguilar, et al., Uncertainty-aware selecting for an ensemble of deep food recognition models, Comput. Biol. Med. 146 (2022), 105645.
[27] J.N. Teng, et al., Recognition of chinese food using convolutional neural network, Multimed. Tool. Applic. 78 (9) (2019) 11155–11172.
[28] I. Goodfellow, et al., Deep Learning, MIT press, 2016.
[29] A.S. Lundervold, et al., An overview of deep learning in medical imaging focusing on mri, Z. Med. Phys. 29 (2) (2019) 102–127.

[30] F.Z. Zhuang, et al., A comprehensive survey on transfer learning, in: Proceedings of the Ieee 109, 2021, pp. 43–76.
[31] J. Deng, et al., Imagenet: a large-scale hierarchical image database, in: IEEE-Computer-Society Conference on Computer Vision and Pattern Recognition Workshops, Miami Beach, FL, 2009, pp. 248–255.
[32] J. Yosinski, et al., How transferable are features in deep neural networks ?, in: 28th Conference on Neural Information Processing Systems (NIPS), Montreal, CANADA, 2014, pp. 1–9.
[33] X. Yang, et al., A survey on deep semi-supervised learning, IEEE Trans. Knowl. Data Eng. (2022), https://doi.org/10.1109/TKDE.2022.3220219.
[34] P. Dhiman, et al., Citrus fruits classification and evaluation using deep convolution neural networks: an input layer resizing approach, in: 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2021, pp. 1–4.
[35] M.-Y. Chen, et al., Automatic chinese food identification and quantity estimation, in: SIGGRAPH Asia 2012 Technical Briefs, Association for Computing Machinery, Singapore, Singapore, 2012, pp. 1–4.
[36] J. Chen, et al., Deep-based ingredient recognition for cooking recipe retrieval, in: Proceedings of the 24th ACM international conference on Multimedia, Amsterdam, The Netherlands, Association for Computing Machinery, 2016, pp. 32–41.
[37] Tatsuya Miyazaki, et al., Image-based calorie content estimation for dietary assessment, in: 2011 IEEE International Symposium on Multimedia, Dana Point, CA, USA, 2011, pp. 363–368.
[38] Y. Matsuda, et al., Recognition of multiple-food images by detecting candidate regions, in: 2012 IEEE International Conference on Multimedia and Expo, Melbourne, VIC, Australia, 2012, pp. 25–30.
[39] W. Min, et al., Ingredient-guided cascaded multi-attention network for food recognition, in: Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 2019, pp. 1331–1339.
[40] W. Min, et al., 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, Seattle, WA, USA, 2020, pp. 393–401.
[41] F. Zhou, et al., Fine-grained image classification by exploring bipartite-graph labels, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 1124–1133.
[42] A. Mariappan, et al., Personal dietary assessment using mobile devices, in: Proceedings Computational Imaging VII 7246, 2009, pp. 1–11.
[43] X. Gao, et al., Hierarchical attention network for visually-aware food recommendation, IEEE Trans. Multimed. 22 (6) (2020) 1647–1659.
[44] M. Bosch, et al., Integrated database system for mobile dietary assessment and analysis, in: 2011 IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 2011, pp. 1–6.
[45] L. Bossard, et al., Food-101 – mining discriminative components with random forests, in: D. Fleet, et al. (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 446–461. Editors.
[46] A. Myers, et al., Im2calories: towards an automated mobile vision food diary, in: 2015 IEEE International Conference on Computer Vision (ICCV), Boston Massachusetts, 2015, pp. 1233–1241.
[47] P. Ma, et al., Application of deep learning for image-based chinese market food nutrients estimation, Food Chem. 373 (2022), 130994.
[48] M. Chen, et al., Pfid: Pittsburgh fast-food image dataset, in: 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 2009, pp. 289–292.
[49] J. Taichi, et al., A food image recognition system with multiple kernel learning, in: 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 2009, pp. 285–288.
[50] H. Hoashi, et al., Image recognition of 85 food categories by feature fusion, in: 2010 IEEE International Symposium on Multimedia, Taichung, Taiwan, 2010, pp. 296–301.
[51] Y. Kawano, et al., Foodcam-256: a large-scale real-time mobile food recognitionsystem employing high-dimensional features and compression of classifier weights, in: Proceedings of the 22nd ACM international conference on Multimedia, Orlando, Florida, USA, 2014, pp. 761–762.
[52] Q. Yu, et al., Food image recognition by personalized classifier, in: 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 2018, pp. 171–175.
[53] J. Qiu, et al., Mining discriminative food regions for accurate food recognition, in: British Machine Vision Conference, BMVC, Cardiff, UK, 2019, pp. 1–11.
[54] P. Pandey, et al., Foodnet: recognizing foods using ensemble of deep networks, IEEE Signal Process. Lett. 24 (12) (2017) 1758–1762.
[55] G.A. Tahir, et al., An open-ended continual learning for food recognition using class incremental extreme learning machines, IEEE Access 8 (2020) 82328–82346.
[56] T. Stütz, et al., Can mobile augmented reality systems assist in portion estimation? A user study, in: 2014 IEEE International Symposium on Mixed and Augmented Reality - Media, Art, Social Science, Humanities and Design (ISMAR-MASH'D), Munich, Germany, 2014, pp. 51–57.
[57] C. Termritthikun, et al., Nu-innet: Thai food image recognition using convolutional neural networks on smartphone, J. Telecommun. Electron. Comput. Eng. (JTEC) 9 (2-6) (2017) 63–67.
[58] C. Güngör, et al., Turkish cuisine: a benchmark dataset with turkish meals for food recognition, in: 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 2017, pp. 1–4.
[59] H. Mureşan, et al., Fruit recognition from images using deep learning, Acta Universitatis Sapientiae, Informatica 10 (1) (2018) 26–42.
[60] S. Hou, et al., Vegfru: a domain-specific dataset for fine-grained visual categorization, in: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 541–549.
[61] G. Waltner, et al., Personalized dietary self-management using mobile vision-based assistance, in: New Trends in Image Analysis and Processing – ICIAP 2017, Springer International Publishing, Cham, 2017, pp. 385–393.
[62] J. Steinbrener, et al., Hyperspectral fruit and vegetable classification using convolutional neural networks, Comput. Electron. Agric. 162 (2019) 364–372.
[63] V. Meshram, et al., Fruitnet: Indian fruits image dataset with quality for machine learning applications, Data in Brief 40 (2022), 107686.
[64] S.K. Behera, et al., Fruits yield estimation using faster r-cnn with miou, Multimed. Tool. Applic. 80 (12) (2021) 19043–19056.
[65] J. Zhou, et al., A vegetable disease recognition model for complex background based on region proposal and progressive learning, Comput. Electron. Agric. 184 (2021), 106101.
[66] G.M. Farinella, et al., A benchmark dataset to study the representation of food images, in: Computer Vision - ECCV 2014 Workshops, Springer International Publishing, Cham, 2015, pp. 584–599.
[67] W. Xin, et al., Recipe recognition with large multimodal food dataset, in: 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Turin, 2015, pp. 1–6.
[68] G. Ciocca, et al., Food recognition and leftover estimation for daily diet monitoring, in: New Trends in Image Analysis and Processing – ICIAP 2015 Workshops, Springer International Publishing, Cham, 2015, pp. 334–341.
[69] L. Herranz, et al., A probabilistic model for food image recognition in restaurants, in: 2015 IEEE International Conference on Multimedia and Expo (ICME), Turin, Italy, 2015, pp. 1–6.
[70] O. Beijbom, et al., Menu-match: restaurant-specific food logging from images, in: 2015 IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 2015, pp. 844–851.
[71] G. Ciocca, et al., Food recognition: a new dataset, experiments, and results, IEEE J. Biomed. Health Informat. 21 (3) (2017) 588–598.
[72] M. Merler, et al., Snap, eat, repeat: a food recognition engine for dietary logging, in: Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, Amsterdam, The Netherlands, Association for Computing Machinery, 2016, pp. 31–40.
[73] A. Singla, et al., 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, Amsterdam, The Netherlands, 2016, pp. 3–11.
[74] G.M. Farinella, et al., Retrieval and classification of food images, Comput. Biol. Med. 77 (2016) 23–39.
[75] J. Rich, et al., Towards bottom-up analysis of social food, in: Proceedings of the 6th International Conference on Digital Health Conference, Montréal, Québec, Canada, 2016, pp. 111–120.
[76] M. Bolaños, et al., Simultaneous food localization and recognition, in: 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 2016, pp. 3140–3145.
[77] G. Ciocca, et al., Learning cnn-based features for retrieval of food images, in: Proceedings of the New Trends in Image Analysis and Processing—ICIAP 2017, Springer International Publishing, Cham, Italy, 2017, pp. 426–434.
[78] P. Kaur, et al., Foodx-251: A Dataset for Fine-Grained Food Classification, arXiv e-prints, 2019, https://doi.org/10.48550/arXiv.1907.06167 arXiv:1907.06167.
[79] D. Sahoo, et al., 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, Anchorage, AK, USA, 2019, pp. 2260–2268.
[80] W. Min, et al., Large Scale Visual Food Recognition, arXiv e-prints, 2021, https://doi.org/10.48550/arXiv.2103.16107 arXiv:2103.16107.
[81] E. Aguilar, et al., Regularized uncertainty-based multi-task learning model for food analysis, J. Visual Commun. Image Represent. 60 (2019) 360–370.
[82] G. Popovski, et al., Foodbase corpus: a new resource of annotated food entities, Database 2019 (2019) baz121.
[83] S.Y. Alaba, et al., Deep learning-based image 3-d object detection for autonomous driving: review, IEEE Sens. J. 23 (4) (2023) 3378–3394.
[84] L.C.D. Vergara, et al., Analysis of the accuracy potential of a stereo high-speed camera system in 3d measurements in highly dynamic experiments, Sensors 23 (4) (2023) 2158.
[85] D.S. Kaputa, et al., Model based design of a real time fpga-based lens undistortion and image rectification algorithm for stereo imaging, IEEE Access 11 (2023) 18972–18982.
[86] S.E.H. Piacenza, et al., Design and fabrication of a stereo-video camera equipped unoccupied aerial vehicle for measuring sea turtles, sharks, and other marine fauna, PLoS One 17 (10) (2022).
[87] A.M. Al-juboori, et al., A hybrid cracked tiers detection system based on adaptive correlation features selection and deep belief neural networks, Symmet.-Base. 15 (2) (2023) 358.
[88] O. Duran, et al., Vehicle-to-vehicle distance estimation using artificial neural network and a toe-in-style stereo camera, Measurement 190 (2022), 110732.
[89] B. Pipping, et al., Three-dimensional quasi-static displacement of human middle-ear ossicles under static pressure loads: measurement using a stereo camera system, Hear. Res. 427 (2022), 108651.
[90] E.H. Williams, et al., Estimating length composition of fish observed with stereo-video cameras: a simulation study with application to red snapper (lutjanus campechanus), Fish. Res. 254 (2022), 106424.

[91] O. Saadi, et al., Application of remote sensing data and geographic information system for identifying potential areas of groundwater storage in middle moulouya basin of morocco, Groundwater Sustain. Develop. 14 (2021), 100639.
[92] V. Pancholi, et al., Estimation of groundwater potential zones using remote sensing and geographical information system technique- waghai taluka, dang district, gujarat, western india, Environ. Challeng. 9 (2022), 100615.
[93] N. Bachagha, et al., Remote sensing and gis techniques for reconstructing the military fort system on the roman boundary (tunisian section) and identifying archaeological sites, Remote Sens. Environ. 236 (2020), 111418.
[94] M.-D. Cheng, et al., A study of extractive and remote-sensing sampling and measurement of emissions from military aircraft engines, Atmos. Environ. 44 (38) (2010) 4867–4878.
[95] T. Wellmann, et al., Remote sensing in urban planning: contributions towards ecologically sound policies? Landsc. Urban Plan. 204 (2020), 103921.
[96] H. Li, Pattern planning and design of tiger hazelnut shrub in urban ecosystem based on remote sensing technology, Environ. Technol. Innov. 21 (2021), 101330.
[97] S. Eilola, et al., Lessons learned from participatory land use planning with high-resolution remote sensing images in tanzania: practitioners' and participants' perspectives, Land Use Policy 109 (2021), 105649.
[98] H. Miura, et al., Empirical estimation based on remote sensing images of insured typhoon-induced economic losses from building damage, Int. J. Disast. Risk Reduct. 82 (2022), 103334.
[99] T.S. Aung, et al., The environmental burdens of special economic zones on the coastal and marine environment: a remote sensing assessment in myanmar, Remote Sens. Applic.: Soc. Environ. 28 (2022), 100809.
[100] L. Maldonado, Lighting-up the economic activity of oil-producing regions: a remote sensing application, Remote Sens. Applic.: Soc. Environ. 26 (2022), 100722.
[101] F. Tupinambá-Simões, et al., Assessment of drought effects on survival and growth dynamics in eucalypt commercial forestry using remote sensing photogrammetry. A showcase in Mato Grosso, Brazil, Forest Ecol. Manag. 505 (2022), 119930.
[102] R.J. Birk, et al., Government programs for research and operational uses of commercial remote sensing data, Remote Sens. Environ. 88 (1) (2003) 3–16.
[103] S. Huang, et al., Laser powder bed fusion of titanium-tantalum alloys: compositions and designs for biomedical applications, J. Mech. Behav. Biomed. Mater. 108 (2020), 103775.
[104] H. Li, et al., Eagnet: a method for automatic extraction of agricultural greenhouses from high spatial resolution remote sensing images based on hybrid multi-attention, Comput. Electron. Agric. 202 (2022), 107431.
[105] S.A.E.-S. Zahran, et al., Remote sensing based water resources and agriculture spatial indicators system, Egypt. J. Remot. Sens. Space Sci. 25 (2) (2022) 515–527.
[106] N. Jiang, et al., Remote sensing of swidden agriculture in the tropics: a review, Int. J. Appl. Earth Obs. Geoinf. 112 (2022), 102876.
[107] J. Wieme, et al., Application of hyperspectral imaging systems and artificial intelligence for quality assessment of fruit, vegetables and mushrooms: a review, Biosyst. Eng. 222 (2022) 156–176.
[108] A. Porebski, et al., Comparison of color imaging vs. Hyperspectral imaging for texture classification, Pattern Recognit. Lett. 161 (2022) 115–121.
[109] K. Kniha, et al., Results of thermal osteonecrosis for implant removal on electron microscopy, implant stability, and radiographic parameters - a rat study, Head Face Med. 19 (1) (2023) 4.
[110] T.A. Grunewald, et al., Bone mineral properties and 3d orientation of human lamellar bone around cement lines and the haversian system, IUCRJ 10 (2023) 189–198.
[111] A.A. Boitor, et al., The impact of simulated bruxism forces and surface aging treatments on two dental nano-biocomposites-a radiographic and tomographic analysis, Medicina-Lithuania 59 (2) (2023) 360.
[112] F.J.M. Shamrat, et al., High-precision multiclass classification of lung disease through customized mobilenetv2 from chest x-ray images, Comput. Biol. Med. 155 (2023), 106646.
[113] S.M. Abdullah, et al., Clinical, hematobiochemical and radiographical studies of caprine pneumonia, Slovenian Veterin. Res. 60 (2023) 65–74.
[114] J.J. Wu, et al., Lncrna dgcr5 silencing enhances the radio-sensitivity of human esophageal squamous cell carcinoma via negatively regulating the warburg effect, Radiat. Res. 199 (3) (2023) 264–272.
[115] H. Daly, et al., Mature teratoma of the anterior mediastinum revealed by supravalvular pulmonary stenosis: a case report, Pan Afr. Med. J. 43 (2022) 109.
[116] G. Roque, et al., Sub-pixel energy-weighting techniques for metallic contaminant highlighting in a pharmaceutical hard capsule using a timepix3 cdznte hybrid pixel detector, J. Instrum. 17 (10) (2022) P10030.
[117] V.T. Oanh, et al., Instant facile method for the in situ growth of ni(oh)(2) nanohives on nickel foam for non-enzymatic electrochemical glucose sensor, J. Electrochem. Soc. 169 (11) (2022), 117506.
[118] T. Matsui, et al., Development of automatic detection model for stem-end rots of 'hass' avocado fruit using x-ray imaging and image processing, Postharvest Biol. Technol. 192 (2022), 111996.
[119] V. Schatz, Measuring timing properties of thermal infrared cameras, Meas. Sci. Technol. 34 (5) (2023), 055407.
[120] A.D.K. Ali, et al., Cutting parameter optimization based on online temperature measurements, Eng. Technol. Appl. Sci. Res. 13 (1) (2023) 9861–9866.
[121] A.N. Wilson, et al., Recent advances in thermal imaging and its applications using machine learning: a review, IEEE Sens. J. 23 (4) (2023) 3395–3407.
[122] L.J. Gu, et al., Low-cost assistive body temperature screening system to combat communicable infectious diseases leveraging edge computing and long-range and low-power wireless networks, IEEE Internet Thing. J. 10 (5) (2023) 4174–4183.
[123] C.L. McGinnis, et al., Enhanced thermal imaging to detect microvasculature during surgery: real-time image acquisition, Infrared Phys. Technol. 127 (2022), 104410.
[124] A. Sarhadi, et al., Machine learning based thermal imaging damage detection in glass-epoxy composite materials, Compos. Struct. 295 (2022), 115786.
[125] V. Vasdev, et al., Thermal imaging in rheumatoid arthritis knee joints and its correlation with power doppler ultrasound, Med. J. Arm. Force. India (2022), https://doi.org/10.1016/j.mjafi.2022.05.011.
[126] S. Civilibal, et al., A deep learning approach for automatic detection, segmentation and classification of breast lesions from thermal images, Exp. Syst. Appl. 212 (2023), 118774.
[127] A.A. Gowen, et al., Applications of thermal imaging in food quality and safety assessment, Trend. Food Sci. Technol. 21 (4) (2010) 190–200.
[128] C. Wang, et al., An association between large optic cupping and total and regional brain volume: the women's health initiative, Am. J. Ophthalmol. 249 (2023) 21–28.
[129] S.D. Kikano, et al., Association of cardiovascular magnetic resonance diastolic indices with arrhythmia in repaired tetralogy of fallot, J. Cardiovasc. Magn. Reson. 25 (1) (2023) 17.
[130] A.R. Adams, et al., Peripheral and central iron measures in alcohol use disorder and aging: a quantitative susceptibility mapping pilot study, Int. J. Mol. Sci. 24 (5) (2023) 4461.
[131] J. Bomyea, et al., Randomized controlled trial of computerized approach/avoidance training in social anxiety disorder: neural and symptom outcomes, J. Affect. Disord. 324 (2023) 36–45.
[132] K. Maki, et al., Ckd, brain atrophy, and white matter lesion volume: the Japan prospective studies collaboration for aging and dementia, Kidn. Med. 5 (3) (2023), 100593.
[133] P. Hnilicova, et al., Imaging methods applicable in the diagnostics of Alzheimer's disease, considering the involvement of insulin resistance, Int. J. Mol. Sci. 24 (4) (2023) 3325.
[134] R.P. Lee, et al., First experience with postoperative transcranial ultrasound through sonolucent burr hole covers in adult hydrocephalus patients, Neurosurgery 92 (2) (2023) 382–390.
[135] A. Nagata, et al., Development of an outdoor mri system for measuring flow in a living tree, J. Magn. Reson. 265 (2016) 129–138.
[136] G. Collewet, et al., Multi-exponential mri t2 maps: a tool to classify and characterize fruit tissues, Magn. Reson. Imaging 87 (2022) 119–132.
[137] G. Winisdorffer, et al., Mri investigation of subcellular water compartmentalization and gas distribution in apples, Magn. Reson. Imaging 33 (5) (2015) 671–680.
[138] J. Cai, et al., Characterization and recognition of citrus fruit spoilage fungi using raman scattering spectroscopic imaging, Vib. Spectrosc. 124 (2023), 103474.
[139] Y. Zou, et al., Mass spectrometry imaging and its potential in food microbiology, Int. J. Food Microbiol. 371 (2022), 109675.
[140] S. Verdú, et al., Laser scattering imaging combined with cnns to model the textural variability in a vegetable food tissue, J. Food Eng. 336 (2023), 111199.
[141] K.M. Shaw, et al., Predicting volatile fatty acid synthesis from palm oil mill effluent on an industrial scale, Biochem. Eng. J. 187 (2022), 108671.
[142] A.M.R. Bulbul, et al., In-depth analysis of cement-based material incorporating metakaolin using individual and ensemble machine learning approaches, Materials 15 (21) (2022) 7764.
[143] H.S. Kim, et al., Gru-based buzzer ensemble for abnormal detection in industrial control systems, Cmc-Comput. Mater. Continua 74 (1) (2023) 1749–1763.
[144] I. Alnazer, et al., Usefulness of computed tomography textural analysis in renal cell carcinoma nuclear grading, J. Med. Imaging 9 (5) (2022), 054501.
[145] E. Ocran, et al., Estimation of the tail index of pareto-type distributions using regularisation, J. Math. Tokushima Univ. 2022 (2022), 5064875.
[146] D.F. Lian, et al., Ranking-based implicit regularization for one-class collaborative filtering, IEEE Trans. Knowl. Data Eng. 34 (12) (2022) 5951–5963.
[147] H. Lim, et al., Prediction of polyreactive and nonspecific single-chain fragment variables through structural biochemical features and protein language-based descriptors, BMC Bioinf. 23 (1) (2022) 520.
[148] G.H. Yue, et al., Gpr data augmentation methods by incorporating domain knowledge, Appl. Sci.-Base. 12 (21) (2022) 10896.
[149] M. Morita, et al., The width underestimation of 3d objects with image rotation, I-Perception 10 (2019) 43. -43.
[150] M. George, et al., Abnormal activity detection using shear transformed spatio-temporal regions at the surveillance network edge, Multimed. Tool. Applic. 79 (2020) 27511–27532, https://doi.org/10.1007/s11042-020-09277-8.
[151] D.H. Wang, et al., Automatic defect recognition and localization for aeroengine turbine blades based on deep learning, Aerosp. 10 (2) (2023) 178.
[152] Q.M. Ilyas, et al., Automated estimation of crop yield using artificial intelligence and remote sensing technologies, Bioeng.-Base. 10 (2) (2023) 125.
[153] L.G. Divyanth, et al., Image-to-image translation-based data augmentation for improving crop/weed classification models for precision agriculture applications, Algorithms 15 (11) (2022) 401.
[154] K.W. Lee, et al., Diverse covid-19 ct image-to-image translation with stacked residual dropout, Bioeng.-Base. 9 (11) (2022) 698.
[155] A. Jahanpour, Buckling analysis of functionally graded plates subjected to combined in-plane loads, J. Eng. Math. 138 (1) (2023) 2.

[156] L.C. Mesquita, et al., Uncertainties consideration in elastically heterogeneous fluid-saturate d media using first-order second moment stochastic method and green?S function approach, Appl. Math. Modell. 115 (2023) 819–852.
[157] A. Ahmad, et al., Deepdemod: Bpsk demodulation using deep learning over software-defined radio, IEEE Access 10 (2022) 115833–115848.
[158] W. Zhao, et al., Nonlinear volterra filter method for mixed noise reduction in rotate-speed with kernel parameters optimized based on bat algorithm, J. Nonlinear Conv. Analy. 23 (9) (2022) 1895–1911.
[159] A. Svec, et al., Amplitude-modulation forward masking for listeners with and without hearing loss, JASA Express Lett. 2 (12) (2022), 124401.
[160] M. Shajahan, et al., Denoising of impulse noise using partition- supported median, interpolation and dwt in dental X-ray images, Int. Jo. Adv. Comput. Sci. Applic. 13 (9) (2022) 274–280.
[161] M. Momeny, et al., Detection of citrus black spot disease and ripeness level in orange fruit using learning-to-augment incorporated deep networks, Ecolog. Inform. 71 (2022), 101829.
[162] G. Karthiha, et al., Speckle noise suppression in ultrasound images using modular neural networks, Intell. Autom. Soft Comput. 35 (2) (2023) 1753–1765.
[163] W.D. Zhou, et al., Compressed sensing in the presence of speckle noise, IEEE Trans. Inf. Theory 68 (10) (2022) 6964–6980.
[164] K. Matsumoto, et al., Development of poisson noise reduction method using wavelet based bayesshrink technique in brain spect, Eur. J. Nucl. Med. Mol. Imaging 49 (1) (2022) S607. SUPPL-S607.
[165] S.W. Hasinoff, Photon, Poisson noise, in: K. Ikeuchi (Ed.), Computer Vision: A Reference Guide, Springer US, Boston, MA, 2014, pp. 608–610. Editor.
[166] A. Kroshnin, et al., On the properties of some low-parameter models for color reproduction in terms of spectrum transformations and coverage of a color triangle, J. Opt. Soc. Am. a-Opti. Image Sci. Visi. 39 (3) (2022) 452–458.
[167] V. Tanos, et al., Computer-aided diagnosis by tissue image analysis as an optical biopsy in hysteroscopy, Int. J. Mol. Sci. 23 (21) (2022) 12782.
[168] H.G. Kim, et al., Underwater optical-sonar image fusion systems, Sensors 22 (21) (2022) 8445.
[169] X.R. Dai, et al., Data augmentation for thermal infrared object detection with cascade pyramid generative adversarial network, Appl. Intell. 52 (1) (2022) 967–981.
[170] E. Solovyeva, et al., Dual autoencoder network with separable convolutional layers for denoising and deblurring images, J. Imaging 8 (9) (2022) 250.
[171] R. Rajagopal, et al., Deep convolutional spiking neural network optimized with arithmetic optimization algorithm for lung disease detection using chest x-ray images, Biomed. Signal Process. Control 79 (2023), 104197.
[172] G. Awathalei, et al., The role of digital radiography in musculoskeletal imaging: a review article, J. Pharmaceut. Negat. Result. 13 (2022) 48–53.
[173] Kang, G., et al., Patchshuffle regularization. arXiv preprint arXiv:1707.07103, 2017.
[174] Z. Zhu, Pscnn: Patchshuffle convolutional neural network for covid-19 explainable diagnosis, Front. Public Health 9 (2021), 768278.
[175] Inoue, H., Data augmentation by pairing samples for images classification. arXiv preprint arXiv:1801.02929, 2018.
[176] H. Zhang, et al., Mixup: beyond empirical risk minimization, in: 6th International Conference on Learning Representations (ICLR), Vancouver CANADA, 2018, pp. 1–13.
[177] C. Summers, et al., Improved mixed-example data augmentation, in: IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, USA, IEEE, 2019, pp. 1262–1270.
[178] Z. Zhong, et al., Random erasing data augmentation, in: Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, USA, 2020, pp. 13001–13008.
[179] J.J. Zhang, et al., Toward adversarial robustness in unlabeled target domains, IEEE Trans. Image Process. 32 (2023) 1272–1284.
[180] M. Lechner, et al., Revisiting the adversarial robustness-accuracy tradeoff in robot learning, IEEE Robot. Automat. Lett. 8 (3) (2023) 1595–1602.
[181] F.J. Rendon-Segador, et al., Crimenet: neural structured learning using vision transformer for violence detection, Neur. Netw. 161 (2023) 318–329.
[182] K. Tian, et al., Domain adaptive object detection with model-agnostic knowledge transferring, Neur. Netw. 161 (2023) 213–227.
[183] O. Kohannim, et al., Predicting temporal lobe volume on mri from genotypes using l 1-l 2 regularized regression, in: 2012 9th IEEE International Symposium on Biomedical Imaging (ISBI), IEEE, 2012, pp. 1160–1163.
[184] S. Thakre, et al., Quantification of similarity and physical awareness of microstructures generated via generative models, Comput. Mater. Sci. 221 (2023), 112074.
[185] F. Abbas, et al., Approximating global illumination with ambient occlusion and environment light via generative adversarial networks, Pattern Recognit. Lett. 166 (2023) 209–217.
[186] H.A. Madni, et al., Blockchain-based swarm learning for the mitigation of gradient leakage in federated learning, IEEE Access 11 (2023) 16549–16556.
[187] K. Psychogyios, et al., Missing value imputation methods for electronic health records, IEEE Access 11 (2023) 21562–21574.
[188] I. Goodfellow, et al., Generative adversarial nets, in: Adv. Neur. Inform. Process. Syst., 2014, pp. 2672–2680.
[189] A. Taheri-Garavand, et al., Meat quality evaluation based on computer vision technique: a review, Meat Sci. 156 (2019) 183–195.
[190] A. Giraudo, et al., Development of an automated method for the identification of defective hazelnuts based on rgb image analysis and colourgrams, Food Control 94 (2018) 233–240.
[191] R. Javed, et al., An intelligent saliency segmentation technique and classification of low contrast skin lesion dermoscopic images based on histogram decision, in: 12th International Conference on the Developments in eSystems Engineering (DeSE), Kazan, RUSSIA, 2019, pp. 164–169.
[192] A. Humeau-Heurtier, Color texture analysis: a survey, IEEE Access 10 (2022) 107993–108003.
[193] Y.M. Fang, et al., Perceptual quality assessment for screen content images by spatial continuity, IEEE Trans. Circuit. Syst. Video Technol. 30 (11) (2020) 4050–4063.
[194] C. Tan, et al., Automatic extraction of color features from landscape images based on image processing, Traitement Du Signal 38 (3) (2021) 747–755.
[195] A. Raza, et al., Correlated primary visual texton histogram features for content base image retrieval, IEEE Access 6 (2018) 46595–46616.
[196] C.L. Wang, et al., Histogram of oriented gradient based plantar pressure image feature extraction and classification employing fuzzy support vector machine, J. Med. Imaging Health Inform. 8 (4) (2018) 842–854.
[197] H. Zoubir, et al., Concrete bridge crack image classification using histograms of oriented gradients, uniform local binary patterns, and kernel principal component analysis, Electronics 11 (20) (2022) 3357.
[198] A. Humeau-Heurtier, Texture feature extraction methods: a survey, IEEE Access 7 (2019) 8975–9000.
[199] V. Ranga, et al., American sign language fingerspelling using hybrid discrete wavelet transform-gabor filter and convolutional neural network, J. Eng. Sci. Technol. 13 (9) (2018) 2655–2669.
[200] R. Wang, et al., Multimedia image data compression based on wavelet analysis, Wirel. Commun. Mob. Comput. 2022 (2022), 2773868.
[201] D.G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, IEEE, Kerkyra, Greece, 1999, pp. 1150–1157.
[202] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision 60 (2) (2004) 91–110.
[203] J. Wu, et al., A comparative study of sift and its variants, Measur. Sci. Rev. 13 (3) (2013) 122–131.
[204] C.B. Singh, et al., Wavelet analysis of signals in agriculture and food quality inspection, Food Bioprocess Technol. 3 (1) (2010) 2–12.
[205] R. Polikar, et al., Frequency invariant classification of ultrasonic weld inspection signals, IEEE Transact. Ultrason. Ferroelectr. Freq. Control 45 (3) (1998) 614–625.
[206] S.Q. Chen, et al., Separation of overlapped non-stationary signals by ridge path regrouping and intrinsic chirp component decomposition, IEEE Sens. J. 17 (18) (2017) 5994–6005.
[207] L. Durak, et al., Short-time fourier transform: two fundamental properties and an optimal implementation, IEEE Trans. Signal Process. 51 (5) (2003) 1231–1242.
[208] O. Rioul, et al., Wavelets and signal processing, IEEE Signal Process. Mag. 8 (4) (1991) 14–38.
[209] C. Mateo, et al., Short-time fourier transform with the window size fixed in the frequency domain, Digital Signal Process. 77 (2018) 13–21.
[210] A. Graps, An introduction to wavelets, IEEE Comput. Sci. Eng. 2 (2) (1995) 50–61.
[211] L. Wang, et al., Drag reduction characterizations of turbulent channel flow with surfactant additive by proper orthogonal decomposition and wavelet transform, Korea-Austr. Rheol. J. 32 (1) (2020) 1–14.
[212] A. Agarwal, et al., Wavelet-based multiscale similarity measure for complex networks, Eur. Phys. J. B 91 (11) (2018) 296.
[213] E. Aboufadel, et al., Discovering wavelets, John Wiley & Sons, New York, USA, 1999.
[214] I. Daubechies, Ten Lectures on Wavelets. Cbms-nsf Series in Applied Mathematics, 61, Society for industrial and applied mathematics, Philadelphia, PA, USA, 1992.
[215] R.M. Rao, et al., Wavelet transforms: introduction to theory and applications, J. Electron. Imaging 8 (4) (1999) 478.
[216] C. CORTES, et al., Support-vector networks, Mach. Learn. 20 (20) (1995) 273–297.
[217] T.G. Nick, et al., Logistic regression, Top. Biostatist. (2007) 273–301.
[218] K. Fukunaga, et al., A branch and bound algorithm for computing k-nearest neighbors, IEEE Trans. Comput. 100 (7) (1975) 750–753.
[219] A. Likas, et al., The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[220] J.C. Bezdek, et al., Fcm: the fuzzy c-means clustering algorithm, Comput. Geosci. 10 (2-3) (1984) 191–203.
[221] N. Friedman, et al., Bayesian network classifiers, Machine Learn. 29 (2) (1997) 131–163.
[222] S.R. Safavian, et al., A survey of decision tree classifier methodology, IEEE Trans. Syst. Man Cybern. 21 (3) (1991) 660–674.
[223] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[224] W.S. McCulloch, et al., A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[225] A.T. Goh, Back-propagation neural networks for modeling complex systems, Artificial Intell. Eng. 9 (3) (1995) 143–151.
[226] H. Guang-Bin, et al., Extreme learning machine: theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.
[227] S. Baek, et al., Face detection in untrained deep neural networks, Nat. Commun. 12 (1) (2021) 7328.
[228] Y.H. Pao, et al., Learning and generalization characteristics of random vector functional-link net, Neurocomputing 6 (1994) 163–180.
[229] H.Y. Jiang, et al., Deep learning techniques for tumor segmentation: a review, J. Supercomput. 78 (2) (2022) 1807–1851.
[230] L. Zhou, et al., Application of deep learning in food: a review, Compreh. Rev. Food Sci. Food Saf. 18 (6) (2019) 1793–1811.

[231] B. Benuwa, et al., A review of deep machine learning, Int. J. Eng. Res. Afr. 24 (2016) 124–136.
[232] X.Q. Yan, et al., Deep multi-view learning methods: a review, Neurocomputing 448 (2021) 106–129.
[233] Y.B. Fu, et al., Deep learning in medical image registration: a review, Phys. Med. Biol. 65 (20) (2020) 20tr01.
[234] A. Yadav, et al., Sentiment analysis using deep learning architectures: a review, Artif. Intell. Review 53 (6) (2020) 4335–4385.
[235] Z.Q. Zhao, et al., Object detection with deep learning: a review, IEEE Transact. Neur. Netw. Learn. Syst. 30 (11) (2019) 3212–3232.
[236] S.T. Hang, et al., Bi-linearly weighted fractional max pooling an extension to conventional max pooling for deep convolutional neural network, Multimed. Tool. Applic. 76 (21) (2017) 22095–22117.
[237] T.Y. Hsiao, et al., Filter-based deep-compression with global average pooling for convolutional networks, J. Syst. Archit. 95 (2019) 9–18.
[238] S.H. Wang, et al., Alcoholism detection by data augmentation and convolutional neural network with stochastic pooling, J. Med. Syst. 42 (1) (2018) 2.
[239] K. Kakuda, et al., Nonlinear activation functions in cnn based on fluid dynamics and its applications, Cmes-Comput. Model. Eng. Sci. 118 (1) (2019) 1–14.
[240] M. Vlcek, Chebyshev polynomial approximation for activation sigmoid function, Neural Netw. World 22 (4) (2012) 387–393.
[241] H.A. Abdusalam, On an improved complex tanh-function method, Int. J. Nonlin. Sci. Numer. Simulat. 6 (2) (2005) 99–106.
[242] J.L. Cui, et al., Text classification based on relu activation function of SAE algorithm, in: 14th International Symposium on Neural Networks (ISNN), Japan, 2017, pp. 44–50.
[243] Y.S. Liu, et al., A modified leaky relu scheme (mlrs) for topology optimization with multiple materials, Appl. Math. Comput. 352 (2019) 188–204.
[244] J. Crnjanski, et al., Adaptive sigmoid-like and prelu activation functions for all-optical perceptron, Opt. Lett. 46 (9) (2021) 2003–2006.
[245] X. Jin, et al., Deep learning with s-shaped rectified linear activation units, in: Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, Arizona, 2016, pp. 1737–1743.
[246] D.-A. Clevert, et al., Fast and accurate deep network learning by exponential linear units (elus), in: International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016, pp. 1–14.
[247] G. Klambauer, et al., Self-normalizing neural networks, Adv. Neur. Inform. Process. Syst. 30 (2017) 1–10.
[248] H. Zheng, et al., Improving deep neural networks using softplus units, in: 2015 International joint conference on neural networks (IJCNN), IEEE, 2015, pp. 1–4.
[249] X. Glorot, et al., Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.
[250] H. Zhao, et al., A novel softplus linear unit for deep convolutional neural networks, Appl. Intell. 48 (2018) 1707–1720.
[251] M. Basirat, et al., The quest for the golden activation function, in: Proceedings of the ARW & OAGM Workshop, Steyr, Austria, 2019, p. 190.
[252] Y.H. Li, et al., Adaptive batch normalization for practical domain adaptation, Pattern Recognit. 80 (2018) 109–117.
[253] X. Shen, et al., Continuous dropout, IEEE Transact. Neur. Netw. Learn. Syst. 29 (9) (2018) 3926–3937.
[254] Q.K. Abood, et al., Predicting age and gender using alexnet, Tem J.-Technol. Educ. Manag. Inform. 12 (1) (2023) 512–518.
[255] M.A. Kumar, et al., Classification of ecg signal using fft based improved alexnet classifier, PLoS One 17 (9) (2022), e0274225.
[256] Simonyan, K., et al., Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[257] W.Q. Fan, et al., Fusion of visible and infrared images using ge-wa model and vgg-19 network, Sci. Rep. 13 (1) (2023) 190.
[258] J. Thomkaew, et al., Improvement classification approach in tomato leaf disease using modified visual geometry group (vgg)-inceptionv3, Int. J. Adv. Comput. Sci. Applic. 13 (12) (2022) 362–370.
[259] Iandola, F.N., et al., Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.
[260] M. Rasool, et al., A novel approach for classifying brain tumours combining a squeezenet model with svm and fine-tuning, Electronics 12 (1) (2023) 149.
[261] L.S. Bernardo, et al., Modified squeezenet architecture for parkinson's disease detection based on keypress data, Biomedicines 10 (11) (2022) 2746.
[262] C. Szegedy, et al., Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1–9.
[263] Y.C. Han, et al., Ultra-short-term wind power interval prediction based on hybrid temporal inception convolutional network model, Electr. Power Syst. Res. 217 (2023), 109159.
[264] Muhammad, W., et al., Irmirs: Inception-Resnet-Based Network for MRI Image Super-Resolution. CMES-Computer Modeling in Engineering & Sciences,.
[265] C. Szegedy, et al., Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, 2016, pp. 2818–2826.
[266] F. Chollet, et al., Xception: deep learning with depthwise separable convolutions, in: 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR, Honolulu, HI, 2017, pp. 1800–1807.
[267] K.M. He, et al., Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, 2016, pp. 770–778.
[268] M.P. Paing, et al., Adenoma dysplasia grading of colorectal polyps using fast fourier convolutional resnet (ffc-resnet), IEEE Access 11 (2023) 16644–16656.
[269] M.K.U. Ahamed, et al., Dtlcx: an improved resnet architecture to classify normal and conventional pneumonia cases from covid-19 instances with grad-cam-based superimposed visualization utilizing chest X-ray images, Diagnostics 13 (3) (2023) 551.
[270] Y.P. Singh, et al., Automatic prediction of epileptic seizure using hybrid deep resnet-lstm model, AI Commun. 36 (1) (2023) 57–72.
[271] S.P. Praveen, et al., Resnet-32 and fastai for diagnoses of ductal carcinoma from 2d tissue slides, Sci. Rep. 12 (1) (2022) 20804.
[272] G. Huang, et al., Densely connected convolutional networks, in: 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR, Honolulu, HI, 2017, pp. 2261–2269.
[273] R. Angeline, et al., Multimodal human facial emotion recognition using densenet-161 and image feature stabilization algorithm, Traitement Du Signal 39 (6) (2022) 2165–2172.
[274] S. Nizarudeen, et al., Multi-layer resnet-densenet architecture in consort with the xgboost classifier for intracranial hemorrhage (ich) subtype detection and classification, J. Intell. Fuzzy Syst. 44 (2) (2023) 2351–2366.
[275] Howard, A.G., et al., Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.
[276] A. Namburu, et al., Forest fire identification in uav imagery using x-mobilenet, Electronics 12 (3) (2023) 733.
[277] R.O. Ogundokun, et al., Mobilenet-svm: a lightweight deep transfer learning model to diagnose bch scans for iomt-based imaging sensors, Sensors 23 (2) (2023) 656.
[278] S.N. Xie, et al., Aggregated residual transformations for deep neural networks, in: 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR, Honolulu, HI, 2017, pp. 5987–5995.
[279] X. Zhang, et al., Shufflenet: an extremely efficient convolutional neural network for mobile devices, in: 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, 2018, pp. 6848–6856.
[280] S.S. Ying, et al., Shufflenet v2.3-stackedbilstm-based tool wear recognition model for turbine disc fir-tree slot broaching, Machines 11 (1) (2023) 92.
[281] Y.X. Fu, et al., Chinese lip-reading research based on shufflenet and cbam, Appl. Sci.-Basel 13 (2) (2023) 1106.
[282] B. Zoph, et al., Learning transferable architectures for scalable image recognition, in: 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, 2018, pp. 8697–8710.
[283] M.X. Tan, et al., Mnasnet: platform-aware neural architecture search for mobile, in: 32nd IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, 2019, pp. 2815–2823.
[284] M.X. Tan, et al., Efficientnet: rethinking model scaling for convolutional neural networks, in: 36th International Conference on Machine Learning (ICML), Long Beach, CA, 2019, pp. 6105–6114.
[285] S. Abd El-Ghany, et al., Computer-aided diagnosis system for blood diseases using efficientnet-b3 based on a dynamic learning algorithm, Diagnostics 13 (3) (2023) 404.
[286] N. Siddique, et al., Fractal, recurrent, and dense u-net architectures with efficientnet encoder for medical image segmentation, J. Med. Imaging 9 (6) (2022), 064004.
[287] Y.P. Huang, et al., Gpipe: efficient training of giant neural networks using pipeline parallelism, in: 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, CANADA, 2019, pp. 103–112.
[288] Y.F. Li, et al., Semi-supervised learning for abus tumor detection using deep learning method, IET Image Proc. (2023), https://doi.org/10.1049/ipr2.12777.
[289] Z. Guo, et al., Car emotion labeling based on color-ssl semi-supervised learning algorithm by color augmentation, Int. J. Intell. Syst. 2023 (2023), 4331838.
[290] F. Naeem, et al., Federated-learning-empowered semi-supervised active learning framework for intrusion detection in zsm, IEEE Commun. Mag. 61 (2) (2023) 88–94.
[291] S.J. Tang, et al., Multi-scale recursive semi-supervised deep learning fault diagnosis method with attention gate, Machines 11 (2) (2023) 153.
[292] Z. Liu, et al., Dual-feature-embeddings-based semi-supervised learning for cognitive engagement classification in online course discussions, Knowl.-Based. Syst. 259 (2023), 110053.
[293] F.A.L. Laskowski, et al., Identification of potential solid-state li-ion conductors with semi-supervised learning, Energy Environ. Sci. 16 (3) (2023) 1264–1276.
[294] Y. Wang, et al., Dual semi-supervised learning for classification of alzheimer's disease and mild cognitive impairment based on neuropsychological data, Brain Sci. 13 (2) (2023) 306.
[295] M.G. Albayati, et al., Semi-supervised machine learning for fault detection and diagnosis of a rooftop unit, Big Data Min. Analyt. 6 (2) (2023) 170–184.
[296] A. Kumar, et al., Tlspg: transfer learning-based semi-supervised pseudo-corpus generation approach for zero-shot translation, J. King Saud Univ.-Comput. Inform. Sci. 34 (9) (2022) 6552–6563.
[297] M. Chen, et al., Duplicate image representation based on semi-supervised learning, Int. J. Grid High Perform. Comput. 14 (1) (2022).
[298] J. Calder, et al., Rates of convergence for laplacian semi-supervised learning with low labeling rates, Res. Math. Sci. 10 (1) (2023) 10.
[299] W.J. Sang, et al., Porosity prediction using semi-supervised learning with biased well log data for improving estimation accuracy and reducing prediction uncertainty, Geophys. J. Int. 232 (2) (2023) 940–957.
[300] Y. Chen, et al., Semi-supervised and unsupervised deep visual learning: a survey, in: IEEE transactions on pattern analysis and machine intelligence, 2022, https://doi.org/10.1109/tpami.2022.3201576.
[301] J.R. Cheng, et al., Generative adversarial networks: a literature review, KSII Transact. Internet Inform. Syst. 14 (12) (2020) 4625–4647.

[302] I. Goodfellow, et al., Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[303] J.M. Yang, et al., Progressive image restoration with multi-stage optimization, in: 31st International Conference on Artificial Neural Networks (ICANN), Univ W England, Bristol, ENGLAND, 2022, pp. 445–457.
[304] J. Ye, et al., Research on image-to-image translation with capsule network, in: 28th International Conference on Artificial Neural Networks (ICANN), Klinikum Rechts Isar, Munich, GERMANY, Tech Univ Munchen, 2019, pp. 141–151.
[305] H. Zhang, et al., Stackgan plus plus: realistic image synthesis with stacked generative adversarial networks, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2019) 1947–1962.
[306] Y. Cheng, et al., Sequential attention gan for interactive image editing, in: 28th ACM International Conference on Multimedia (MM), Electr Network, 2021, pp. 4383–4391.
[307] D. Yoo, et al., Pixel-level domain transfer, in: 14th European Conference on Computer Vision (ECCV), Amsterdam, NETHERLANDS, 2016, pp. 517–532.
[308] S.Y. Pan, et al., Chefgan: food image generation from recipes, in: 28th ACM International Conference on Multimedia (MM), Electr Network, 2021, pp. 4244–4252.
[309] Z.H. Fei, et al., Enlisting 3d crop models and gans for more data efficient and generalizable fruit detection, in: IEEE/CVF International Conference on Computer Vision (ICCVW), Electr Network, 2021, pp. 1269–1277.
[310] W.P. Lei, et al., Multiworking conditions anomaly detection of mechanical system based on conditional variational auto-encoder, Shock Vibra. 2023 (2023), 2332669.
[311] A. Driessen, et al., Single-cell map of childhood acute myeloid leukaemia using variational auto-encoders, Blood 140 (2022) 2265–2266.
[312] Y. Fu, et al., Association prediction of circrnas and diseases using multi-homogeneous graphs and variational graph auto-encoder, Comput. Biol. Med. 151 (2022), 106289.
[313] Y. Song, et al., Generative modeling by estimating gradients of the data distribution, in: 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, CANADA, 2019, pp. 1–13.
[314] A. Nichol, et al., Improved denoising diffusion probabilistic models, in: International Conference on Machine Learning (ICML), Virtual Only, 2021, pp. 1–17.
[315] J. Choi, et al., Ilvr: conditioning method for denoising diffusion probabilistic models, in: 18th IEEE/CVF International Conference on Computer Vision (ICCV), Electr Network, 2021, pp. 14347–14356.
[316] I.J. Sreelakshmy, et al., A hybrid inpainting model combining diffusion and enhanced exemplar methods, ACM J. Data Inform. Qual. 13 (3) (2021) 14.
[317] A. Oliver, et al., Realistic evaluation of deep semi-supervised learning algorithms, in: 32nd Conference on Neural Information Processing Systems (NIPS), Montreal, CANADA, 2018, pp. 1–19.
[318] S. Laine, et al., Temporal ensembling for semi-supervised learning, in: International Conference on Learning Representations (ICLR), Toulon, France, 2017, pp. 1–13.
[319] T. Miyato, et al., Distributional Smoothing with Virtual Adversarial Training, Arxiv, 2016 arXiv:1507.0067.
[320] E.D. Cubuk, et al., Randaugment: practical automated data augmentation with a reduced search space, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, 2020, pp. 3008–3017.
[321] S. Lim, et al., Fast autoaugment, in: 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, CANADA, 2019, pp. 1–10.
[322] D. Ho, et al., Population based augmentation: efficient learning of augmentation policy schedules, in: 36th International Conference on Machine Learning (ICML), Long Beach, CA, 2019, pp. 1–11.
[323] P. Bachman, et al., Learning with pseudo-ensembles, in: 28th Conference on Neural Information Processing Systems (NIPS), Montreal, CANADA, 2014, pp. 1–9.
[324] J.H. Tao, et al., Semi-supervised ladder networks for speech emotion recognition, Int. J. Autom. Comput. 16 (4) (2019) 437–448.
[325] S. Park, et al., Adversarial dropout for supervised and semi-supervised learning, in: 32nd AAAI Conference on Artificial Intelligence /30th Innovative Applications of Artificial Intelligence Conference / 8th AAAI Symposium on Educational Advances in Artificial Intelligence, New Orleans, LA, 2018, pp. 3917–3924.
[326] X.Y. Cao, et al., Uncertainty aware temporal-ensembling model for semi-supervised abus mass segmentation, IEEE Trans. Med. Imaging 40 (1) (2021) 431–443.
[327] A. Tarvainen, et al., Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results, in: 31st Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, 2017, pp. 1–16.
[328] Z.H. Zhou, et al., Semi-supervised learning by disagreement, Knowl. Inform. Syst. 24 (3) (2010) 415–439.
[329] T.Y. Qian, et al., Co-training on authorship attribution with very few labeled examples: methods vs. views, in: 37th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval, Gold Coast, AUSTRALIA, 2014, pp. 903–906.
[330] Z.H. Zhou, et al., Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.
[331] L. Breiman, Randomizing outputs to increase prediction accuracy, Mach. Learn. 40 (3) (2000) 229–242.
[332] Q. Xie, et al., Self-training with noisy student improves imagenet classification, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 10687–10698.
[333] X. Wang, et al., Enaet: a self-trained framework for semi-supervised and supervised learning with ensemble transformations, IEEE Trans. Image Process. 30 (2021) 1639–1647.
[334] T. Chen, et al., A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning (ICML), Electr Network, 2021, pp. 1–20.
[335] X.L. Chen, et al., Exploring simple siamese representation learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, 2021, pp. 15745–15753.
[336] M. Caron, et al., Emerging properties in self-supervised vision transformers, in: 18th IEEE/CVF International Conference on Computer Vision (ICCV), Electr Network, 2021, pp. 9630–9640.
[337] K. Avrachenkov, et al., Semi-supervised learning with regularized laplacian, Optim. Method. Softw. 32 (2) (2017) 222–236.
[338] C.A.R. de Sousa, et al., An overview on the gaussian fields and harmonic functions method for semi-supervised learning, in: International Joint Conference on Neural Networks (IJCNN), IEEE, Killarney, IRELAND, 2015, pp. 1–8.
[339] D. Zhou, et al., Learning with local and global consistency, Adv. Neur. Inform. Process. Syst. 16 (2003) 1–8.
[340] B. Wohlberg, Efficient algorithms for convolutional sparse representations, IEEE Trans. Image Process. 25 (1) (2016) 301–315.
[341] L.S. Zhuang, et al., Non-negative low rank and sparse graph for semi-supervised learning, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, 2012, pp. 2328–2335.
[342] X.Y. Yan, et al., Semi-supervised deep hashing with a bipartite graph, in: 26th International Joint Conference on Artificial Intelligence (IJCAI), Melbourne, AUSTRALIA, 2017, pp. 3238–3244.
[343] J. Zhang, et al., Ssdh: semi-supervised deep hashing for large scale image retrieval, IEEE Trans. Circuit. Syst. Video Technol. 29 (1) (2019) 212–225.
[344] Z. Zhang, et al., Robust adaptive embedded label propagation with weight learning for inductive classification, IEEE Transact. Neur. Netw. Learn. Syst. 29 (8) (2018) 3388–3403.
[345] P. McAllister, et al., Combining deep residual neural network features with supervised machine learning algorithms to classify diverse food image datasets, Comput. Biol. Med. 95 (2018) 217–233.
[346] Z.Y. Yu, et al., Hyperspectral imaging technology combined with deep learning for hybrid okra seed identification, Biosyst. Eng. 212 (2021) 46–61.
[347] W.D. Zhang, et al., Wi-hsnn: a subnetwork-based encoding structure for dimension reduction and food classification via harnessing multi-cnn model high-level features, Neurocomputing 414 (2020) 57–66.
[348] G.A. Tahir, et al., Explainable deep learning ensemble for food image analysis on edge devices, Comput. Biol. Med. 139 (2021), 104972.
[349] J.J. Chen, et al., A study of multi-task and region-wise deep learning for food ingredient recognition, IEEE Trans. Image Process. 30 (2021) 1514–1526.
[350] Z.L. Wang, et al., Ingredient-guided region discovery and relationship modeling for food category-ingredient prediction, IEEE Trans. Image Process. 31 (2022) 5214–5226.