Journal of Agriculture and Food Research 21 (2025) 101850

Contents lists available at ScienceDirect

Journal of Agriculture and Food Research

journal homepage: www.sciencedirect.com/journal/journal-of-agriculture-and-food-research

Dry fruit image classification using stacking ensemble model

Maheen Islam [a], Mujahidul Islam [a], Alfe Suny [a], Abdullah Al Rafi [a], Abdullahi Chowdhury [a], Mohammad Manzurul Islam [a,*], Saleh Masum [b], Md Sawkat Ali [a], Taskeed Jabid [a], Md Mostofa Kamal Rasel [a]

[a] Department of Computer Science and Engineer, East West University, Aftabnagar, 1212, Dhaka, Bangladesh
[b] Department of Information and Communication Engineering, University of Rajshahi, Charukala Road, 6205, Rajshahi, Bangladesh

ARTICLE INFO

Keywords:
Dry fruits
Stacking
Ensemble
Image classification
Machine learning
Nutrition
CNN
Lightweight

ABSTRACT

Precise and efficient classification of dry fruit images is critical for enhancing quality control, efficiency, and safety in the agricultural and food industries. This study presents a CNN-based classification model developed to analyze a diverse dataset of dry fruit images. The dataset comprises 11,520 high-resolution images representing four varieties (raisins, cashews, almonds, and figs), each divided into three groups, resulting in twelve distinct classes. The proposed multi-step methodology covers data collection, preprocessing, augmentation, and model training. A wide range of image conditions, including variations in fruit types, shapes, colors, and lighting, facilitated comprehensive experimentation and validation using performance indicators such as accuracy. A stacking ensemble, integrating predictions from multiple models (e.g., VGG16, VGG19, ResNet50, MobileNetV1, Mo- bileNetV2, SqueezeNet, and ShuffleNet), achieved a test accuracy of 98.32 %, surpassing individual base model performances (MobileNetV2: 90.33 %, Mo- bileNetV1: 93.68 %, SqueezeNet: 96.98 %, and ShuffleNetV2: 97.79 %). These findings underscore the model's potential for real-time applications in qual-ity control, automated processing, nutritional research, and other domains, while also delineating the key components required for accurate classification.

1. Introduction

Dry fruit is basically fruit dehydrated through process, and as the fruit becomes smaller, it concentrates itself with energy. They pack themselves with 2.5 times more fiber than fresh fruits, vitamins, and minerals. They are high in phenolic antioxidants and packed with amazing health benefits. They can be stored longer than fresh fruits [1].

In addition to health advantages, dry fruits have high commercial worth. Dry fruits occupy a significant place in both culinary and economic spheres. The global dried fruit industry is expected to be worth USD 7.7 billion by 2028 [2]. The most popular dried fruits are raisins, dates, prunes, figs, and apricots [1]. However, due to variability in appearance, similarities among varieties, complexity of features like surface textures, fruit skin imperfec-tions, and intra-class variability, accurately identifying and classifying dif-ferent types of dry fruits solely based on their visual characteristics poses a considerable challenge. Therefore, accurate and efficient dry fruit recognition is of great importance in the field of robotic harvesting of agricultural products [3].

Harvesting dry fruits manually is both labor-intensive and hazardous. It is also inefficient in terms of both time and economy [4]. A study [5] on Indian cashew processing industry suggests that workers in the cashew processing industry, particularly in shelling, face numerous health hazards. Shelling is one of the most dangerous tasks, where workers, often women, sit in cramped, unsanitary conditions surrounded by dust and burnt nutshells. During shelling, workers are exposed to cashew nut shell liquid (CNSL), a caustic substance that can cause severe skin burns and irritation. Sharp shell fragments can also injure their eyes, sometimes causing permanent vision damage. Prolonged squatting and repetitive tasks worsen these conditions, leaving workers vulnerable to long-term musculoskeletal damage, respiratory diseases, and other chronic health issues [5]. When dry fruits are handled by hand, allergens may be exposed, especially to those who are sensitive to nuts or dried fruits. This can result in extremely serious allergic reactions [6]. Dry fruits may also have contaminants like mold or pesticide residues on their surface, which could be harmful to the workers [7]. Manual classification is repetitive, which increases the risk

This article is part of a special issue entitled: AI technology in food and agriculture published in Journal of Agriculture and Food Research.

* Corresponding author.
E-mail address: mohammad.islam@ewubd.edu (M.M. Islam).

https://doi.org/10.1016/j.jafr.2025.101850
Received 6 May 2024; Received in revised form 9 March 2025; Accepted 23 March 2025
Available online 24 March 2025

of musculoskeletal disorders, especially in the hands and wrists, as workers may assume uncomfortable postures for extended periods of time [8]. In addition to these physical hazards, accurately classifying dry fruits poses significant challenges due to intra-class variability and visual similarities among varieties. These dual challenges underscore the need for an automated and efficient vision-based classification system.

In light of these challenges, this study proposes an advanced, automated classification framework to improve efficiency and accuracy in dry fruit recog-nition, thereby enhancing quality control and operational safety [4]. The study focuses on the field of dry fruit classification and recognition, combining cutting-edge computer vision, challenging pattern recognition methods, and adaptive machine learning algorithms. The goal is to create strong and adapt-able system that can automate the procedure of dry fruit categorization. The aim of this research is to understand the complex relationship between the differences in dried fruit attributes ranging from small deviations in shape and color to detailed textural contrasts and the creation of advanced compu-tational models capable of detecting and classifying these variations for dry fruits.

The key contributions of this paper are:

- Introduced an innovative stacking ensemble architecture specifically designed for lightweight CNNs, addressing the critical gap between computational efficiency and classification complexity in agricultural vision systems.
- Developed a new hierarchical feature learning mechanism that strategically combines complementary strengths of mobile-optimized architectures through an adaptive meta-learning framework.
- Established a novel methodology for base model selection that prioritizes architectural diversity and computational efficiency, and metrics for evaluating model complementarity in ensemble systems.
- Proposed a systematic approach to parallel model execution in dry-fruit detection system and feature aggregation that advances the theoretical framework for ensemble learning in resource-constrained computer vision applications.

The paper is organized as follows: In Section 2, we present a comprehensive literature review of recent works relevant to our study. Section 3 provides a detailed description of our proposed system, including information about the dataset and the methodology used. In Section 4, we share the results of our experiments with the proposed system, along with a comparative performance analysis. The paper concludes in Section 5, summarizing our findings and research contributions.

2. Related work

A lot of research has been done to improve how well we can classify things, especially by looking into different deep learning models and ways to extract features from various datasets. In this section, we'll dive into the main contributions made to dry fruit classification, focusing on the key models, techniques, and performance measures that earlier studies have used. We've organized this section like this: 2.1. Datasets for Dry Fruit Classification covers the different datasets that have been utilized; 2.2. Image Processing and Feature Selection Techniques compares some traditional methods for feature extraction; 2.3. CNNs for Feature Extraction and Classification talks about the advantages of using deep learning; and finally, 2.4. Ensemble Models for Improved Classification explores ways to enhance model accuracy.

2.1. Datasets for Dry Fruit Classification

Accurate and comprehensive datasets play a pivotal role in advancing re-search on dry fruit classification. Meshram et al. [2] assembled a dataset fea-turing four major dry fruit types—almonds, raisins, cashews, and figs—captured under both natural and artificial lighting conditions. This diversity in illu-mination and background makes their dataset particularly useful for various machine learning applications as it encompasses a range of real-world scenar-ios. Kilanko et al. [9] focused on cashew nuts' physical traits, such as weight, porosity, and density, thereby providing crucial information for mechanical design and classification tasks.

Gautam et al. [10] proposed a transfer learning–based convolutional neural network for dry fruit identification, incorporating multiple publicly available image datasets. Their work highlights how the combination of large-scale, pre-trained models with domain-specific fine-tuning can significantly boost classification accuracy [11]. Particularly for raisins—a commonly studied dry fruit—researchers have also explored specialized datasets that capture different raisin varieties under controlled conditions to analyze color, texture, and shape attributes [12,13]. These raisin specific collections often include meticulous annotations and a focus on quality assessment, making them excellent resources for training machine learning algorithms on subtle intra-class variations.

2.2. Image Processing and Feature Selection Techniques

Then there's Arora et al., who used image processing techniques like K-means clustering along with feature selection methods, including neigh-borhood component analysis and stepwise regression, to sort cashew kernels. Their approach was pretty successful, reaching an accuracy of 94.28% with Random Forest classifiers [14]. This is a bit different from what we're fo-cusing on, as we're diving into deep learning models that automatically find important features in images without needing any manual input.

2.3. CNNs for feature extraction and classification

Jan and their team trained a CNN model using a dataset of 800 dry fruit images, and they managed to achieve an impressive accuracy of 94%. While they didn't specify the exact architecture they used, it's likely that popular models like VGG16 or VGG19 were involved [3]. Ponce and colleagues applied CNNs to classify olives and got even better results, reaching an accuracy of 95.91% with the Inception-ResNetV2 model [15]. Ramya and their team focused on classifying the different states of fruit, achieving 66% accuracy for fruit stages and 79% for fruit conditions [16]. Gill and his team explored using CNNs, RNNs, and LSTMs for fruit image classification, but they didn't go into much detail about the specific models they used [17].

2.4. Ensemble Models for Improved Classification

Ensemble learning techniques like stacking, bagging, and boosting have been shown to really enhance model performance, especially in image classi-fication tasks. While stacking is often used in medical image classification, applying it to dry fruit classification is still a relatively new idea. For exam-ple, Dominik Muller and his team used stacking for medical images, including COVID-19 X-rays and ISIC skin lesions, and they achieved a 13% improve-ment in F1-score compared to using individual models. This demonstrates the potential of stacking in tough classification challenges, though their study focused mainly on medical images and metrics like F1-score, rather than how interpretable the models are [18].

Annisa and her colleagues explored ensemble machine learning techniques for diagnosing COVID-19. They proposed a two-level stacking model that combined Support Vector Classification (SVC), Random Forest, and K-Nearest Neighbors (KNN) as the base learners, with SVC acting as the meta-learner. Their approach included using Gabor filters for feature extraction, which highlights the importance of capturing both spatial and frequency informa-tion. This could also be useful for classifying dry fruits [19]. Similarly, Ibrahem and his team looked into

stacking methods for classifying images of musculoskeletal fractures. They evaluated several models, including VGG19, InceptionV3, ResNet50, Xception, and DenseNet, and found that their ensemble model performed about 10% better on average than individual models. This really emphasizes how effective stacking can be for complex image recognition tasks and suggests that it could boost performance in dry fruit classification by leveraging the strengths of different models [20].

Vidyarthi and his team compared four deep convolutional neural net-work (DCNN) models—Inception-V3, ResNet50, VGG-16, and a custom model—for classifying cashews. They achieved a top accuracy of 98.4% with Inception-V3 and ResNet50 [21]. The stacking ensemble approach allows for integrating multiple models to improve accuracy, making it a robust option for classification tasks. Gill and his team combined CNNs, RNNs, and LSTMs for their classification work, but they didn't provide much detail on their methodology [17].

3. Proposed method

3.1. Dataset selection

For this project we used the dataset introduced by Meshram et, al. [2]. This dataset features four types of dry fruits: almonds, raisins, cashews, and figs. Fig. 1 provides a visual summary of this dataset, illustrating the four primary fruit categories and their respective sub-classes. Specifically, each fruit type in Meshram et al.'s dataset is further divided into three distinct sub-groups (for example, the almond category includes subtypes like Regular, Sanora, and Marma). By depicting all four categories and their sub-classes, Fig. 1 highlights the breadth and granularity of classification classes that previous studies have addressed, underscoring the complexity of dry fruit classification tasks tackled in earlier research. This diversity in illumination and background makes their dataset particularly useful for various machine learning applications, as it encompasses a range of diverse scenarios.

For almonds, these subgroups are denoted as Regular, Sanora, and Marma. Cashews exhibit three subclasses: Regular, Special, and Jumbo. Raisins are classified into three subclasses: Black, Grade 1, and Premium. Dried figs are stratified into three subclasses: Small, Medium, and Jumbo. Consequently, the dataset encompasses a total of 12 distinct classes. A visual representa-tion of sample images from the dataset is provided in Fig. 2. The dataset of dry fruits was gathered by the authors in Ref. [2] through capturing images under different lighting conditions, which included both artificial and natural light sources. Furthermore, the images were taken against a variety of backgrounds, including white, black, green, and the surface of human palms, to ensure diversity in the dataset. The diversity in illumination and background ensures that the machine learning model is not overfitted to a specific background and lighting condition.

Table 1 outlines the distribution of images across classes, with 960 images per subclass, resulting in a total of 2880 images for each class and 11,520 images overall. Original high efficiency image coding (.heic) files were down scaled to 512 × 512 pixels and converted to jpg format using high-definition cell phone cameras. Images, initially at 3042 × 4032 pixels, were uniformly downsized to 224 × 224 pixels using a Python script. Capture conditions involved a solar direction range of 60°–120°, and artificial lighting was introduced through two Light Emitting Diodes (LEDs) placed on either side of the background at a 45° angle for consistent illumination.

There must be a factor to take into account in the methods section. The system must be built and the dataset must be correctly collected before implementation can begin. Data must then go through a pre-processing step where it is enhanced and shrunk. There are a few classification systems accessible in convolutional neural networks. The photos are then used for training, where they are instructed in a variety of areas up to and including their quality and characteristics. Finally, with the assistance of accurate image recognition, evaluation metrics become visible.

3.2. Data preparation and dataset split

The process of image prediction follows a well defined pipeline to ensure systematic handling of data and model evaluation. Initially, the raw dataset is acquired and subjected to preprocessing, a critical step to ensure consistency and quality of the input data. Following pre-processing, the dataset is partitioned into three subsets: training, validation, and testing, as illustrated in Fig. 3. Specifically, 70% of the dataset is allocated for training, 10% reserved for validation, while the remaining 20% serves as the test dataset. This partitioning was executed using a Python script to ensure precise distribution only once, so the consistency is ensured across the whole study.

The training data subset is utilized to train the machine learning models, enabling them to learn the underlying patterns and features within the dataset. Simultaneously, the validation subset plays a pivotal role in hyperparameter tuning, providing insights into how well the model generalizes during the training process. Hyperparameter optimization based on validation performance aids in refining the model's architecture and configuration to achieve optimal results.

Once the model training and hyperparameter tuning are complete, the test data is employed to evaluate the model's performance

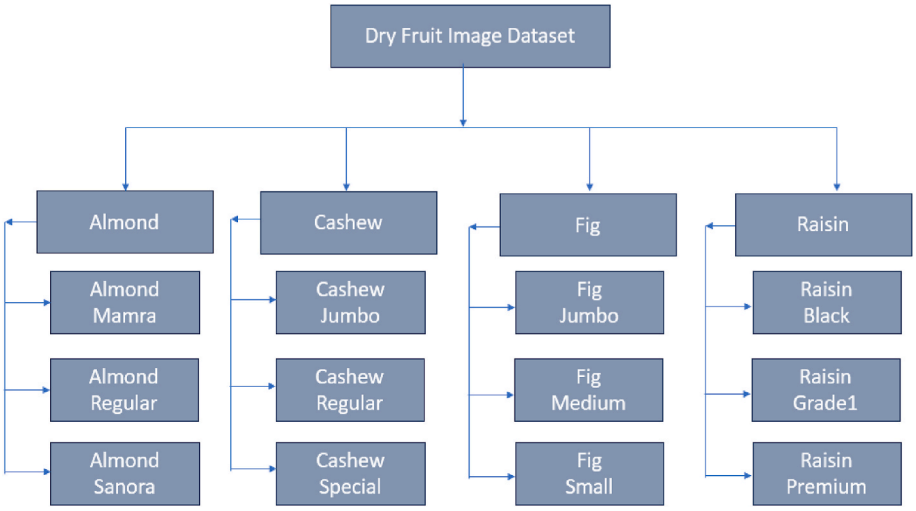

Fig. 1. Different types of dry fruits and their subclasses as developed and described in [2].

Fig. 2. Twelve distinct dry fruit datasets sample [2].

Table 1
Distribution of images across dry fruit classes.

Category	Sub classes	Img./Sub-Class	Total
Almond	Almond Mamra Almond Regular Almond Sanora Cashew Jumbo	960	2880
Cashew	Cashew Regular Cashew Special Fig Jumbo	960	2880
Fig	Fig Medium Fig Small Raisin Black	960	2880
Raisin	Raisin Grade-1 Raisin Premium	960	2880
Total			**11,520**

objectively. This final evaluation assesses the model's predictive accuracy and ensures that it generalizes effectively to unseen data. The process of training, validating, and testing iteratively enhances the model's robustness and reliability, ultimately yielding predictions of superior quality.

3.3. Architecture selection

In the pursuit of constructing a robust classification framework for the intricate task of dry fruit image classification, this study adopted a deliberate and comprehensive approach by implementing a stacking ensemble model. Stacking, as an ensemble technique, involves combining the predictions of multiple models to enhance overall performance. This approach leverages the diverse strengths of different models, thereby achieving superior accuracy in classification tasks. The architecture of the proposed stacking ensemble model is illustrated in Fig. 4.

3.3.1. Base learner exploration

In building the foundational layer of the ensemble stacking model, a thoughtful and detailed selection process was undertaken to establish a robust framework. The selection of base learners is a critical component of the stacking ensemble approach. The models were chosen based on three key factors:

- Architectural Diversity: Different convolutional neural network (CNN) architectures bring unique strengths to the ensemble, reducing bias and improving classification robustness. ResNet50, for example, incorporates residual connections that help mitigate vanishing gradient issues, while MobileNetV1 and MobileNetV2 are optimized for computational efficiency using depthwise separable convolutions.
- Computational Efficiency: Given the constraints of real-time applications and resource-limited environments (e.g., mobile or IoT devices), we prioritized models that balance accuracy and lightweight execution. MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2 are significantly smaller than VGG19 or ResNet50, making them ideal candidates for practical deployment.
- Performance Trade-offs: Previous studies have demonstrated that deeper models like VGG19 and ResNet50 achieve high accuracy due to their extensive feature extraction capabilities. Lightweight models like MobileNetV2 and ShuffleNetV2 perform comparably well, requiring fewer computational resources. By integrating both high capacity and efficient models, the ensemble benefits from broad feature representation while maintaining feasibility for real-world applications.

To ensure an optimal balance between accuracy and runtime efficiency, we empirically evaluated different CNN architectures, as summarized in Table 2 (Comparison of CNN Models) and Table 3 (Runtime Analysis of Base Models). The final selection of MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2 as base learners was based on their ability to generalize well, maintain computational efficiency, and complement each other in terms of feature extraction.

Seven distinct convolutional neural networks (CNNs), each known for specific strengths and limitations, were carefully examined to form a

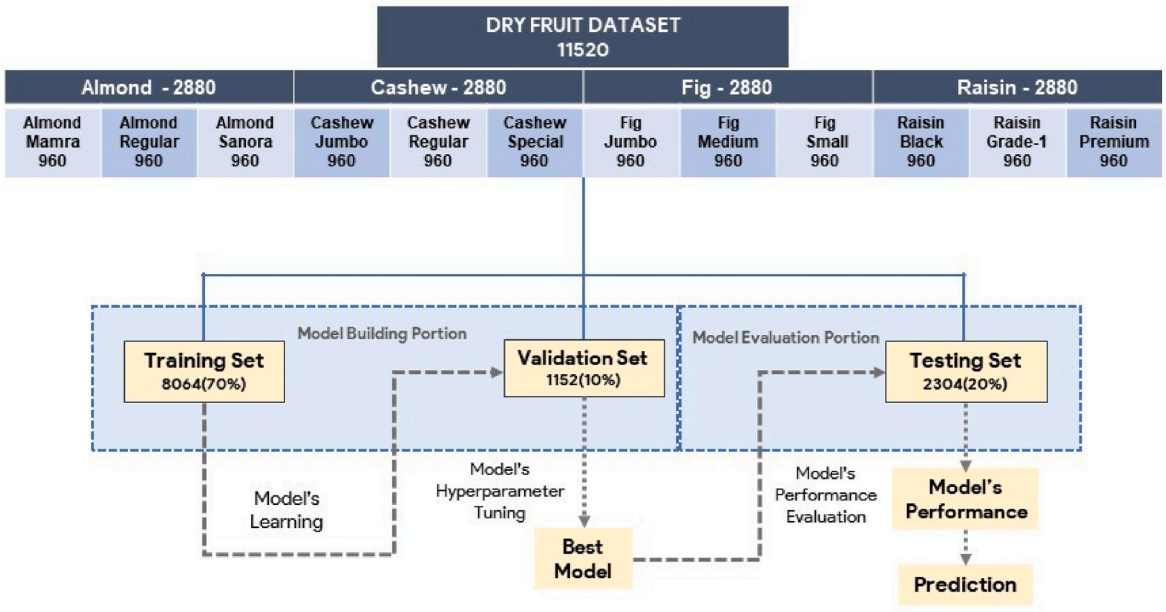

Fig. 3. Train, Validation & Test split of Dataset.

diverse and effective base learner structure. Table 2 compares these models. The aim was to balance accuracy, computational efficiency, and suitability for resource-constrained environments such as embedded systems, IoT devices [22], and handheld systems.

VGG16 emerged as a strong performer due to its high accuracy and exceptional image classification capabilities. However, its massive 138 million parameters demand significant memory and computational power, making it unsuitable for low-power devices. Similarly, VGG19, with 144 million parameters, offered enhanced feature extraction and transfer learning capabilities but came with even higher energy demands, limiting its practical application in battery powered systems.

ResNet50, with its efficient residual block architecture and a moderate parameter count of 25.6 million, provided a good balance of depth and accuracy. However, its computational overhead still posed challenges for low-power IoT environments [23]. In contrast, MobileNetV1, designed with depthwise separable convolutions, reduced its parameter count to just 4.2 million, making it ideal for embedded systems, though it came with some trade-offs in accuracy and scalability.

MobileNetV2 improved efficiency further by introducing inverted residuals and linear bottlenecks, reducing its parameter count to 3.5 million. While suitable for IoT devices, additional optimization was necessary for real-time tasks due to its slightly lower accuracy. SqueezeNet stood out with only 1.24 million parameters and a minimal memory footprint, making it highly suitable for embedded systems, though its feature extraction capabilities were more limited compared to advanced models.

ShuffleNetV2 offered a clever design with channel shuffling that minimized computation while maintaining a lightweight structure of just 2.3 million parameters. This made it ideal for handheld devices, though its accuracy was marginally lower than that of larger models.

By combining the most efficient models in terms of runtime and predictive power, the ensemble system maximized both accuracy and computational performance. Models that failed to meet these efficiency standards were excluded, ensuring the system's overall effectiveness without compromising its ability to operate in resource-constrained environments. This strategic selection process created a balanced, high performing ensemble framework well suited for real-world applications.

3.3.2. *Meta learner selection*

To create a dynamic, reliable, and adaptable ensemble framework, we integrated multiple meta-models that complement the strengths of our base learners. For the meta-learner, we opted for multinomial logistic regression due to the following considerations:

- Simplicity & Interpretability: Unlike complex deep learning meta-models, logistic regression provides clear weight distributions for each base model, making it easier to interpret the influence of each learner in the final decision-making process.
- Overfitting Prevention: More complex meta-learners, such as deep neural networks, may introduce unnecessary model complexity, leading to potential overfitting. Logistic regression, being a linear model, helps avoid this issue while effectively aggregating predictions.
- Efficient Training and Inference: The logistic regression model is computationally lightweight, ensuring that the additional processing over-head introduced by the stacking ensemble remains minimal. This supports real-time classification needs without significantly increasing run-time per sample.

This approach enhances the system's ability to adapt to varying dataset characteristics while maintaining consistent performance. The meta-models leverage diverse feature representations generated by the base learners, ensuring a flexible and efficient classification process. For detailed descriptions of the selected meta-models and their roles within the ensemble framework, refer to Section 4.2.

3.3.3. *Data augmentation*

Even while CNNs are quite good at picture classification and identifica-tion tasks, collecting a lot of photos can be difficult, expensive, and time consuming. Fig. 5 represents the types of augmented images, With the help of a number of powerful techniques, including feature standardization, rotations, shifting, cropping, flipping in both horizontally and vertically, adding noise, including brightness and contrast adjustment image augmentation creates training data from an existing dataset. It takes photos from a training dataset and modifies them to create numerous changed variants of the same image, increasing the number of images available for training and boosting classifiers. Brightness could range from 20% to 80%, rotation could be +5° was thought to enhance the images. These photos must be enhanced for training purposes in order for the machine to correctly learn while

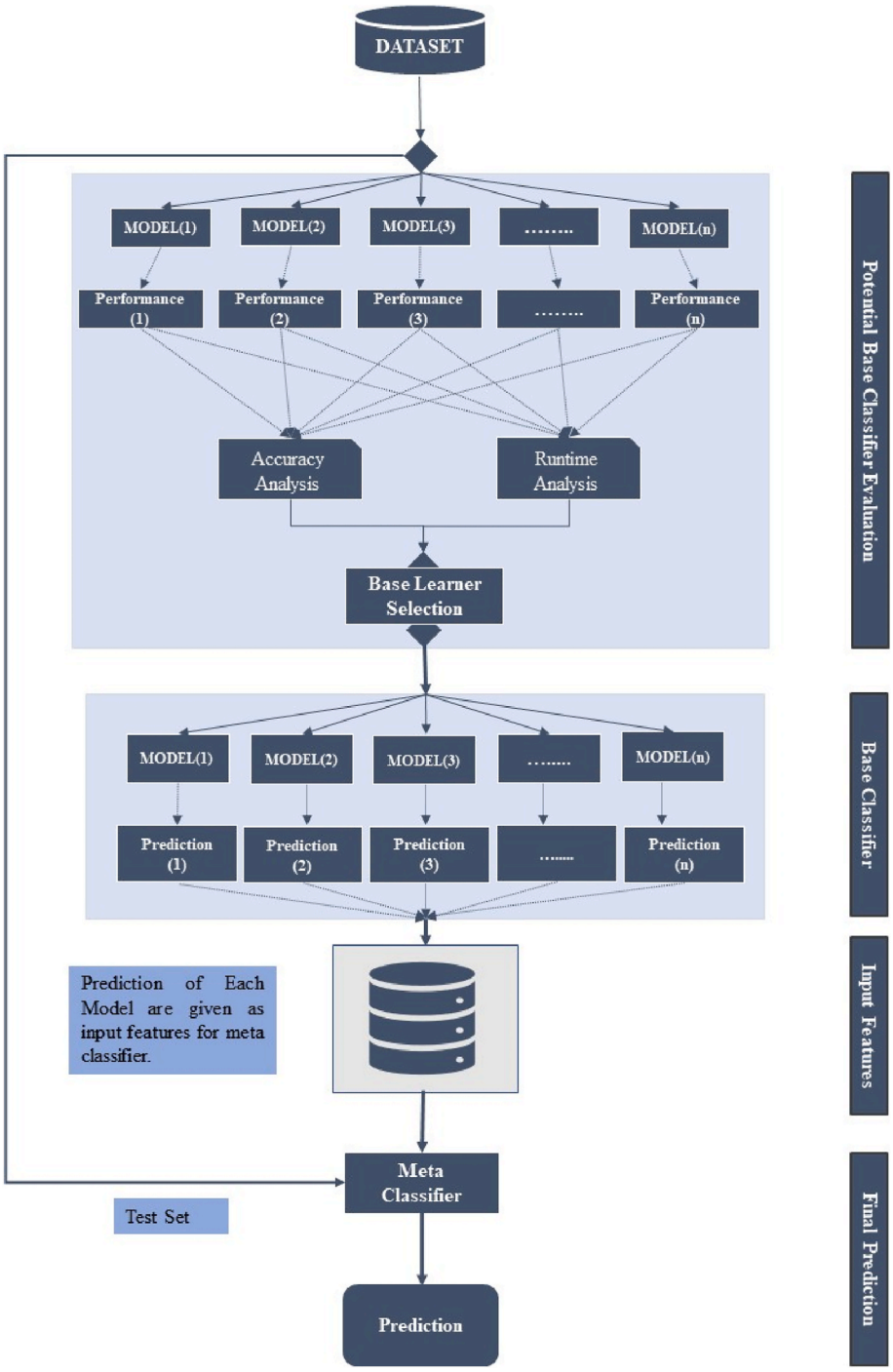

Fig. 4. Architecture of the proposed stacking ensemble model.

analyzing or forecasting the images.

3.4. Optimization through iterative training

In order to train image classification models, which are usually implemented as deep neural networks like CNNs, optimizers are essential. A few optimizers, such as Adam, Stochastic Gradient Descent (SGD), and RMSprop are utilized in this project to update the model's weights at various points throughout training. The model's performance and training pace can be greatly impacted by the optimizer selection. The hyper parameters for the models were kept at their default settings, and for deep learning, we used the Adam optimizer with a learning rate of 0.0001.

The process of training the model involves several critical components, each playing a distinct role in optimizing the machine learning model. A loss function is employed to quantify the disparity between the model's predictions and the actual target labels. The batch size parameter controls the number of training instances processed in each iteration, shaping the efficiency of the training process.

The concept of epochs defines the complete iterations through which the entire training dataset is processed by the model, influencing the overall learning trajectory. Additionally, metrics serve as performance indicators during training, encompassing measures such as F1 score, recall, or accuracy. This comprehensive overview provides insight into

Table 2
Comparison of CNN models.

Feature	VGG16	VGG19	ResNet50	MobileNetV1	MobileNetV2	SqueezeNet	ShuffleNetV2
Total Layers	16	19	50	28	53	18	50 (2 × model)
Parameters (M)	138M	144M	25.6M	4.2M	3.5M	1.24M	2.3M
Input Size (px)	224 × 224	224 × 224	224 × 224	224 × 224	224 × 224	224 × 224	224 × 224
Kernel Size	3 × 3	3 × 3	3 × 3, 1 × 1	3 × 3	3 × 3, 1 × 1	3 × 3, 1 × 1	3 × 3, 1 × 1
Activation	ReLU	ReLU	ReLU	ReLU	ReLU6	ReLU	ReLU
Pooling Layers	Max Pooling	Max Pooling	Avg Pooling	Avg Pooling	Avg Pooling	Max Pooling	Avg Pooling
Skip Connections	No	No	Yes	No	Yes	No	No
Depthwise Conv.	No	No	No	Yes	Yes	No	Yes
Batch Norm.	No	No	Yes	Yes	Yes	No	Yes
Dropout	Yes (0.5)	Yes (0.5)	No	No	No	No	No
Feature Maps	64–512	64–512	64–2048	32–1024	32–1280	96–512	24–2048
Special Blocks	None	None	Residual Blocks	Depthwise Conv	Inverted Residual	Fire Modules	Channel Shuffle
Model Size (MB)	528 MB	549 MB	98 MB	17 MB	14 MB	4.8 MB	5–8 MB

Table 3
Runtime analysis of base models.

Model	Total Run (s)	Time Run Time Per Sample (ms)
VGG16	666.55	289.3012
VGG19	474.13	205.7856
ResNet50	268.42	116.5017
MobileNetV1	51.75	22.4609
MobileNetV2	60.14	26.1024
SqueezeNet	73.99	32.1137
ShuffleNetV2	74.28	32.2396

the integral components governing the model training process.

Fig. 6 visually depicts the feature extraction and classification stages for each classes. The training data, constituting input data (in this case, photographs) and corresponding target labels, forms the foundation of the training dataset, crucial for image classification.

4. Result analysis

This segment dives into the results and overview of the research, providing a comprehensive evaluation of the performance of the potential base models.

VGG16, VGG19, ResNet50, MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNet as well as performance comparison among different classification models used as metamodels for ensemble purposes.

4.1. Performance analysis of potential base learners

As the effectiveness of an ensemble model heavily relies on the performance of its base models, the evaluation of base model performance is a critical prerequisite for the success of ensemble approaches in Machine Larning. In this study, the efficiency of seven prominent CNN models: VGG16, VGG19, ResNet50, MobileNetV1, MobileNetV2, SqueezeNet, and, ShuffleNet are investigated, for the classification of dry fruit images. Notably, intended to leverage these well established models as base models for an ensemble stacking approach, recognizing the potential synergies that could arise from combining their diverse strengths. The dataset was cautiously divided into three subsets for base model training, with 70% allocated for training, 20% for testing, and 10% for validation purposes. Identical data subsets were employed uniformly for training, validation, and testing across all seven models. All experiments were conducted on a system equipped with an Intel(R) Core(TM) i5-1035G1, @ 1.00 GHz 1.19 GHz CPU (Core: 4, Threads: 8), 24 (23.8 GB Useable) GB of RAM, and Integrated UHD Graphics of the CPU, ensuring consistent handling of computational demands during training and evaluation. The following section presents the runtime analysis of the potential base models.

4.1.1. Runtime analysis of potential base learners

The runtime analysis evaluates the computational efficiency of the potential base models, focusing on metrics such as total processing time, runtime per sample, and runtime per sample in milliseconds. We set an empirical threshold of 50 ms per sample to set a standard runtime efficiency for the implemented system, ensuring practical feasibility. Researchers can set their own standard for their implemented system. Table 3 summarizes these metrics for VGG16, VGG19, ResNet50, MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2 across a uniform dataset of 2304 samples. This section emphasizes computational demands without delving into test accuracy to isolate runtime

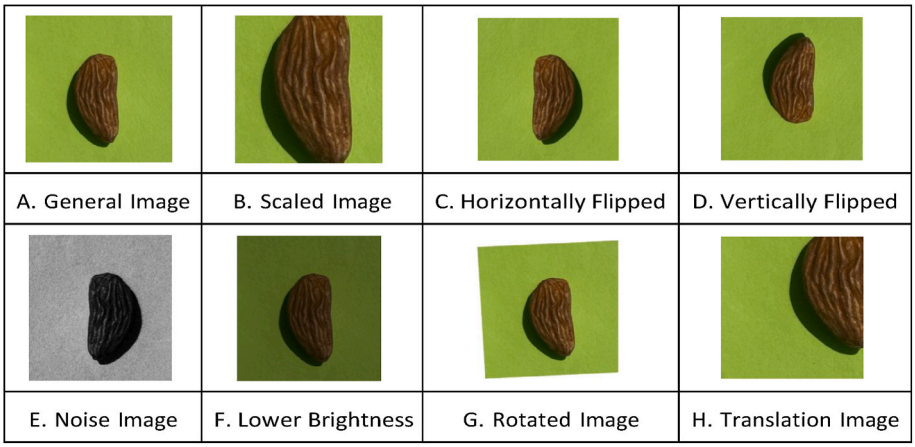

Fig. 5. Data Augmentation Techniques. A. General Image, B. Scaled image, C. Hor-izontally flipped, D. Vertically flipped, E. Noise image, F. Lower brightness, G. Rotated images, H. Translation Image.

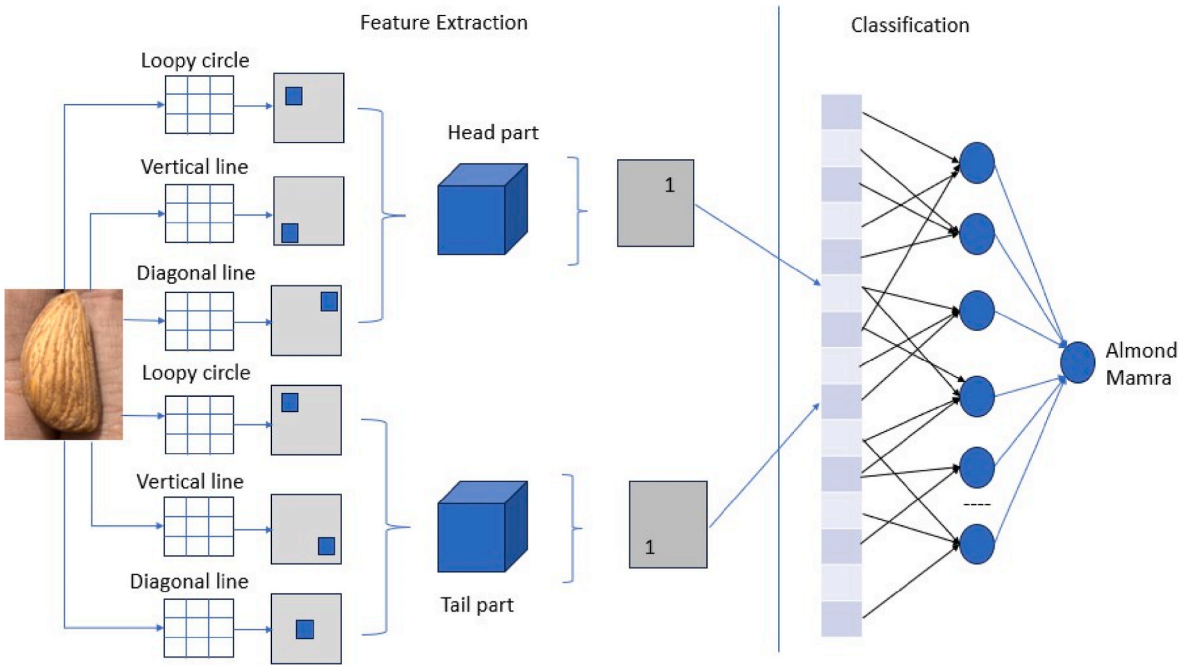

Fig. 6. Feature extraction and classification of an image.

specific considerations which is critical to the selection of base learners.

VGG16 demonstrated the highest computational overhead, with a total runtime of 666.55 s (289.30 ms per sample). This substantial runtime stems from its deep sequential architecture comprising 16 convolutional and fully connected layers, which demand extensive computation. While VGG16's simplicity in design makes it intuitive and effective for feature extraction, its lack of architectural optimizations such as, residual connections or depth-wise convolutions, leads to significant computational inefficiencies. This makes VGG16 less suitable for time sensitive or resource-constrained environments despite its reliable performance in large-scale classification tasks.

VGG19 improved runtime efficiency compared to VGG16, with a total runtime of 474.13 s (205.78 ms per sample). Although it adds three more convolutional layers (totaling 19 layers), its improved layer distribution and architecture adjustments make it more computationally efficient. However, like VGG16, it still suffers from increased computational demands due to its fully connected layers and dense convolutional operations. These factors position VGG19 as a compromise between learning capacity and runtime efficiency.

ResNet50, while not the most computationally demanding, incurred a runtime of 268.42 s (116.50 ms per sample), making it more efficient than VGG16 and VGG19 but still computationally intensive compared to lighter architectures. This extended runtime is primarily attributed to the deep architecture comprising 50 convolutional layers augmented with residual connections. While these residual blocks enable effective gradient flow and facilitate deeper learning, they also increase computational complexity due to the additional skip connections and matrix operations. Such runtime inefficiencies make ResNet50 less suitable for real-time or resource-constrained applications, despite its superior learning capacity.

In contrast, MobileNetV1 demonstrated exceptional computational efficiency, completing inference in 51.75 s (22.46 ms per sample). This efficiency arises from the depthwise separable convolution technique, which replaces standard convolutions with depthwise and pointwise convolutions, significantly reducing the number of parameters and multiplications required. MobileNetV2, an improved version, completed the task in 60.14 s (26.10 ms per sample). Its marginally increased runtime reflects the computational cost of advanced architectural innovations, such as inverted residuals and linear bottlenecks, which improve feature extraction and generalization capabilities.

SqueezeNet and ShuffleNetV2, designed explicitly for lightweight and efficient computation, demonstrated moderate runtimes of 73.99 s (32.11 ms per sample) and 74.28 s (32.24 ms per sample), respectively. SqueezeNet leverages a squeeze-and-expand approach to reduce the number of convolutional parameters while maintaining high feature extraction fidelity. ShuffleNetV2 employs channel shuffling and group convolutions to optimize computational resource utilization. These strategies ensure their feasibility for applications with stringent latency requirements, such as edge computing or embedded systems.

To evaluate runtime efficiency, we established an empirical threshold of 50 ms per sample. Models exceeding this threshold, such as VGG16, VGG19, and ResNet50, are set aside for further analysis of their accuracy metrics. This highlights the need to balance computational demands with model performance. MobileNetV1 and MobileNetV2 offer a strong trade-off between runtime feasibility and feature extraction quality, while SqueezeNet and ShuffleNetV2 stand out as highly efficient alternatives with competitive runtimes. In the following subsection, we focus on analyzing the accuracy metrics of the potential base models.

4.1.2. *Accuracy analysis of potential base learners*

Table 4 presents the training, validation and testing accuracy for the four potential base models selected in the previous runtime analysis subsection; MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2 across 50 epochs with mechanism of early stopping. The values highlight how each model's performance changed during training and how well they learned from the data.

Overfitting is a significant challenge in machine learning and deep learning, where a model learns not only the underlying patterns in the training data but also its noise, resulting in poor generalization to unseen data. This issue arises from various factors, including overly complex models with excessive parameters, insufficient training data, and the absence of regularization techniques like L1 or L2 penalties. Early stopping serves as an effective regularization method to combat overfitting by monitoring the model's performance on a validation set during training. It halts the training process when the validation metric, such as loss or accuracy, begins to degrade, indicating that the model is starting to overfit the training data. By stopping at this point, early stopping ensures that the selected model represents an optimal balance between

Table 4
Training, validation, and test accuracy over epochs.

Model	Training Accuracy	Validation Accuracy	Best Model	Best Model Training	Best Model Validation	Test Accuracy	Best Epoch Training	Best Epoch Validation
	(%) Range	(%) Range	Epoch	Accuracy (%)	Accuracy (%)	(%)	Loss	Loss
MobileNetV1	28.31–90.43	56.16–94.88	42	89.66	94.88	93.68	0.2882	0.2082
MobileNetV2	30.98–89.48	53.56–91.32	48	89.04	91.32	90.33	0.3052	0.2537
SqueezeNet	51.95–96.27	70.05–96.88	17	95.25	96.88	96.48	0.1254	0.0925
ShuffleNetV2	38.01–97.26	45.23–98.09	12	95.68	98.09	97.79	0.1408	0.0657

underfitting and overfitting, thus preserving its ability to generalize well to new data [24]. The early stopping mechanism employed in this study was configured to terminate the training process if the validation accuracy did not improve for five consecutive epochs. The model achieving the highest validation accuracy during the training process was retained as the optimal model for subsequent analysis.

Throughout the 50 epochs, the performance of four evaluated models MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2—reflected their strengths and limitations in classifying images across 12 target classes. The dataset's complexity, coupled with early stopping to prevent overfitting, allowed each model to converge optimally. A closer analysis of their training dynamics, validation trends, and test results offers deeper insights into their performance characteristics.

MobileNetV1 and MobileNetV2: Efficiency with Depthwise Separable Con-volutions: MobileNetV1 and MobileNetV2 emerged as highly efficient models, leveraging depthwise separable convolutions to reduce computational costs while maintaining high accuracy. MobileNetV1 demonstrated training accuracy ranging from 28.31% to 90.43%, with validation accuracy peaking at 94.88% and test accuracy at 93.68%. The best model epoch was at 42, highlighting its rapid learning curve and strong generalization capabilities. MobileNetV2, with its inverted residuals and linear bottleneck layers, enhanced feature extraction efficiency. It achieved training accuracy from 30.98% to 89.48%, with validation and test accuracies peaking at 91.32% and 90.33%, respectively. Its best model epoch was at 48, reflecting its capacity to generalize effectively while maintaining computational efficiency. These models are particularly well suited for applications requiring real-time processing and low latency.

Lightweight Models: SqueezeNet and ShuffleNetV2: SqueezeNet and ShuffleNetV2, designed for minimal parameter usage, also performed well within their intended scope. SqueezeNet achieved a training accuracy range of 51.95%–96.27%, with validation accuracy peaking at 96.88% and test accuracy at 96.48%. Its best model epoch was 17, demonstrating rapid conver-gence despite its lightweight design. ShuffleNetV2 showed similar strength, with training accuracy ranging from 38.01% to 97.26%, validation accuracy reaching 98.09%, and test accuracy 97.79%. The best model epoch occurred at 12, highlighting its ability to quickly capture and generalize features despite its simplified architecture.

The results indicate that all models effectively learned the underlying patterns of the dataset without overfitting, as demonstrated by the close alignment of training, validation, and test accuracies. MobileNetV1 and MobileNetV2 provided an excellent balance of accuracy and computational efficiency, making them highly suitable for real-time processing applications. SqueezeNet and ShuffleNetV2 offered competitive results, emphasizing the importance of efficient design for lightweight applications in resource-constrained environments.

These findings underscore that the choice of model should depend on the specific requirements of the task, including accuracy, computational efficiency, and the complexity of the dataset, particularly for multiclass problems involving 12 target classes. Based on the comparison of test accuracies, the research proceeded further with all four potential base models—MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2—for further analysis. The next section focuses on evaluating these models' overall performance to finalize the base learners for the meta-learner.

4.1.3. Base learner analysis

The selection of base learners for the meta-learner involves balancing computational efficiency with architectural diversity to optimize the ensemble's overall performance. After evaluating the runtime characteristics and training dynamics of seven the models, MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2 were selected as the base learners due to their superior trade-offs between computational cost and architectural efficiency.

MobileNetV1 emerged as an ideal choice due to its minimal runtime and effective use of depthwise separable convolutions, which reduce computational demand without significantly compromising feature extraction. Its ability to generalize effectively within a computationally constrained framework makes it well suited for ensemble integration, where low latency predictions are essential. MobileNetV2 enhances these advantages by incorporating inverted residual blocks and linear bottlenecks, which improve representational power and gradient flow. While its runtime is slightly longer than MobileNetV1, its architectural innovations ensure enhanced feature extraction capabilities, making it a robust addition to the ensemble. These complementary strengths of MobileNetV1 and MobileNetV2 align well with the goals of an ensemble-based approach, where diversity in architectural design can boost overall predictive performance.

SqueezeNet and ShuffleNetV2 further contribute to the ensemble's computational efficiency. SqueezeNet's squeeze-and-expand modules effectively reduce the parameter count, enabling faster inference while retaining high classification fidelity. ShuffleNetV2 complements this efficiency with its unique channel shuffling mechanism, which optimizes inter-channel communication and reduces computation costs. Both models' lightweight designs and rapid convergence during training make them particularly advantageous for resource-constrained environments, such as mobile applications or edge devices.

We excluded VGG16, VGG19 and, ResNet50 from the ensemble due to their prohibitively long runtimes and high computational demands. While these models possess deep and complex architectures that enable robust fea-ture extraction, their resource-intensive nature results in significant latency, making them impractical for real-world scenarios requiring low latency predictions. This decision aligns with the ensemble's goal of optimizing computational efficiency while maintaining effective predictive performance through lightweight and versatile models.

By selecting MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2, the study ensures a robust ensemble foundation. These models collectively offer a balance of runtime efficiency, diverse architectural strengths, and effective feature extraction capabilities. Their selection prioritizes practical deployment considerations, ensuring the meta-learner can achieve high predictive accuracy while maintaining computational feasibility, even for multi-class classification tasks involving complex datasets.

4.2. Meta-learner and stacking model framework

The stacking ensemble architecture is designed to leverage the complementary strengths of our selected computationally efficient base

learners through a hierarchical learning approach. For a given dataset (X, Y), where $X = \{x_1, x_2, \ldots, x_n\}$ represents the input dry fruit images and $Y = \{y_1, y_2, \ldots, y_n\}$ denotes their corresponding true labels, our model employs a two-level learning structure.

At the foundation level, the ensemble incorporates four base learners: MobileNetV1, MobileNetV2, SqueezeNet, and ShuffleNetV2, each producing independent predictions. For any input image x_i, each base model $f_m(X)$ generates a prediction vector $h_m(x_i)$, forming the prediction set:

$$h_m(x_i) = f_m(x_i), \text{ for } m = 1, 2, 3, 4 \tag{1}$$

These predictions are concatenated to create a meta-feature space $H(x_i)$:

$$H(x_i) = [h_1(x_i), h_2(x_i), h_3(x_i), h_4(x_i)] \tag{2}$$

The meta-feature vector $H(x_i)$ undergoes normalization to ensure consistent scaling across different base model outputs before serving as input to the meta-learner. This normalized meta-feature space encapsulates the diverse predictive patterns captured by each base model's unique architectural characteristics.

For the meta-learning phase, we employ a multinomial logistic regression model that processes the meta-features to generate final class probabilities. The meta-learner g maps the combined predictions $H(x_i)$ to class probabili-ties through the following function:

$$\hat{y}_k = g(H(x_i))_k = softmax\left(w_{0k} + \sum_{m=1}^{4} (w_{mk} h_m(x_i))\right) \tag{3}$$

where $\hat{y}_k$ represents the predicted probability for class k, w_{0k} is the class specific bias term, and w_{mk} are the learned weights that determine each base model's contribution to the final prediction for class k. The softmax activation function is defined as:

$$softmax\left(z_k = \frac{exp\ (z_k)}{\sum_{j=1}^{K} exp\ (z_j)}\right) \tag{4}$$

ensuring proper probability distribution across the $K = 12$ dry fruit classes.

The meta-learner optimizes its parameters by minimizing the categorical cross-entropy loss function:

$$L(W) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log\ (\hat{y}_{ik}) \tag{5}$$

where n represents the number of samples, K is the number of classes (12), y_{ik} is the one-hot encoded label (1 if sample i belongs to class k, 0 otherwise), and W encompasses all weight parameters $\{w_{mk}\}$. This optimization is achieved through gradient descent using the Adam optimizer with an adaptive learning rate:

$$w_{mk} \leftarrow w_{mk} - \eta \frac{\partial L}{\partial w_{mk}} \tag{6}$$

where η_t is the adaptive learning rate at iteration t determined by the Adam optimizer's momentum and variance parameters. The gradient for each weight is computed as:

$$\frac{\partial L}{\partial w_{mk}} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_{ik} - y_{ik}) h_m(x_i) \tag{7}$$

This gradient computation ensures that the meta-learner effectively learns to weight each base model's predictions according to their reliability and complementary strengths across different classes.

The final classification decision $\hat{y}$ for any new input image x_i is determined by:

$$\hat{y} = k \epsilon \{1, \ldots, K\}\ (g(H(x_i))_k) \tag{8}$$

As shown in Table 5, the stacking ensemble demonstrated a significant improvement in classification performance, achieving a test accuracy of 98.32%, surpassing the best base model (ShuffleNetV2) with a test accuracy of 97.79%. This improvement highlights the effectiveness of combining the complementary strengths of multiple base models through a meta-learning framework. However, this performance gain comes at the cost of a modest increase in run-time per sample, with the stacking ensemble requiring 43.47 ms compared to ShuffleNetV2's 32.24 ms. Our method used parallel (simultaneously) execution of the base models in our stacking model. So, we needed 11.23 ms extra time to execute our stacking model (see Table 6).

Despite the slight increase in runtime, the enhanced accuracy justifies the stacking approach, particularly for applications where precision and reliability are critical. The trade-off between runtime and accuracy remains favorable, as the ensemble achieves near perfect classification while maintaining computational feasibility through parallel execution. This ensures the stacking model's practical applicability in real world scenarios.

The results presented in Table 5 clearly demonstrate the superiority of the stacking ensemble with our proposed base learners over individual models. The stacking model achieved a test accuracy of 98.32%, surpassing the best performing individual base model (ShuffleNetV2) with a test accuracy of 97.79%. This improvement highlights the effectiveness of leveraging the complementary strengths of multiple base models through a meta-learning framework. While the stacking ensemble incurred a slight runtime increase of 11.23 ms per sample (43.47 ms compared to ShuffleNetV2's 32.24 ms), this additional computational cost was mitigated by employing parallel execution of base models. The trade-off between runtime and accuracy is favorable, as the stacking ensemble not only achieves superior accuracy but also maintains computational feasibility, making it particularly well suited for precision critical applications where reliability and performance are paramount.

To confirm the statistical significance of the accuracy improvement observed in the stacking ensemble model (98.32%) over the best performing base model, ShuffleNetV2 (97.79%), we conducted a paired t-test across multiple independent training runs ($N = 10$). The test was based on the null hypothesis (H_0) that the stacking ensemble does not provide a statistically significant improvement over individual base models, against the alternative hypothesis (H_1) that it significantly outperforms them. The paired t-test results (refer to Table 7 indicate that the improvement is statistically significant ($p < 0.05$) across all comparisons, with a notable improvement over ShuffleNetV2 ($t = 4.27$, $p = 0.0013$), confirming that the ensemble model's enhancements are not due to random variations. Additionally, the 95% confidence intervals (Table 8) reinforce these findings, as the stacking ensemble's interval does not overlap with any base model, suggesting consistent performance gains. These results validate the effectiveness of the proposed stacking ensemble, providing strong evidence of its superiority over individual CNN classifiers in dry fruit classification. This balance ensures that the stacking model is practical and effective in real world scenarios.

Table 5
Performance Analysis: Base Models vs Stacking Ensemble.

Model	Validation Accuracy (%)	Test Accuracy (%)	Runtime (ms/sample)
Individual Base Model			
MobileNetV1	94.88	93.68	22.46
MobileNetV2	91.32	90.33	26.10
SqueezeNet	96.88	96.48	32.11
ShuffleNetV2	98.09	97.79	32.24
Stacking Ensemble	99.15	98.32	43.47[a]

[a] Maximum base model runtime (32.24 m s) + meta-learner overhead (11.23 m s).

Table 6
Performance comparison of stacking ensemble with different meta-learners.

Model	Validation Accuracy (%)	Test Accuracy (%)	Runtime (ms/sample)
Logistic Regression	98.95	98.16	63.91[a]
Decision Tree	98.12	97.89	52.87[a]
Random Forest	98.56	98.04	44.23[a]
Base Learners	99.15	98.32	43.47[a]

[a] Maximum base model runtime + meta-learner overhead.

Table 7
Statistical Significance Results (Paired *t*-test).

Model Comparison	t-Statistic	p-Value
Stacking vs ShuffleNetV2	4.27	0.0013
Stacking vs SqueezeNet	7.35	0.00005
Stacking vs MobileNetV2	18.42	<0.0001
Stacking vs MobileNetV1	10.12	<0.0001

Table 8
95 % confidence intervals for model performance.

Model	Accuracy (%)	95 % Confidence Interval
Stacking Ensemble	98.32	[98.12, 98.52]
ShuffleNetV2	97.79	[97.55, 98.03]
SqueezeNet	96.48	[96.22, 96.74]
MobileNetV2	90.33	[90.01, 90.65]
MobileNetV1	93.68	[93.42, 93.94]

5. Conclusions and future work

This research presents a robust stacking ensemble model for dry fruit classification, integrating multiple CNN architectures—MobileNetV1, Mo- bileNetV2, SqueezeNet, and ShuffleNetV2—as base learners. These models were selected for their exceptional balance between computational efficiency and accuracy, with MobileNetV1 demonstrating the fastest runtime at 22.46 ms per sample and ShuffleNetV2 achieving the highest test accuracy among individual models at 97.79%. By leveraging these strengths through a stacking ensemble, the proposed model achieved an impressive test accuracy of 98.32%, surpassing all individual models. Despite the ensemble's slight increase in runtime (43.47 ms per sample due to meta-learner overhead), this efficiency remains feasible for realtime applications, highlighting its practicality for precision critical tasks. These results showcase the stacking model's ability to address the inherent challenges of classification, including variability in texture, shape, and lighting conditions, while ensuring high performance.

The proposed methodology holds significant implications for the dry fruit industry, particularly in automating labor intensive processes. The model's capability to accurately classify diverse dry fruit types supports enhanced product quality control and operational efficiency, reducing reliance on manual labor, which is often time consuming and prone to errors. Moreover, the runtime efficiency of the selected base learners makes this system feasible for deployment in embedded and edge computing environments, paving the way for realtime applications in production lines and agricultural robotics. These advancements not only address current challenges in the dry fruit sector but also align with the industry's need for scalable, high performance classification systems.

Beyond dry fruit classification, the study provides a flexible framework applicable to various agricultural and culinary domains, including the classification of other food products and delicacies. The combination of computational efficiency, accuracy, and adaptability ensures the methodology's scalability to broader applications. This potential is further enhanced by the inclusion of advanced data augmentation techniques, such as feature scaling and noise addition, which improve generalization across diverse datasets. Future research aims to refine these techniques, expand the dataset to include additional fruit types and categories, and explore lightweight deployment strategies for mobile and IoT-based systems. These efforts will enhance the system's realtime performance and applicability in resource-constrained environments.

The results and analyses underscore the value of combining diverse CNN architectures in a stacking ensemble to achieve superior classification performance while maintaining computational feasibility. This approach not only demonstrates the power of ensemble learning in tackling complex multi-class problems but also contributes to the agricultural sector's growth and sustainability by fostering innovation in product identification systems. With its strong foundation and avenues for future improvement, the proposed system serves as a cornerstone for advancing machine learning applications in agriculture and beyond.

CRediT authorship contribution statement

Maheen Islam: Validation, Supervision, Conceptualization. **Mujahidul Islam:** Validation, Data curation, Conceptualization. **Alfe Suny:** Methodology. **Abdullah Al Rafi:** Writing – review & editing. **Abdullahi Chowdhury:** Writing – review & editing, Software, Methodology, Investigation, Formal analysis. **Mohammad Manzurul Islam:** Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Saleh Masum:** Writing – review & editing, Resources. **Md Sawkat Ali:** Formal analysis. **Taskeed Jabid:** Supervision. **Md Mostofa Kamal Rasel:** Writing – review & editing.

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.

Acknowledgment

The authors would like to thank the Next Generation Artificial Intelligence Research Lab (www.ngair.org) for their collaboration, supervision and resources.

Data availability

The data is publicly available. Link provided in manuscript.

References

[1] J.A. Vinson, L. Zubik, P. Bose, N. Samman, J. Proch, Dried fruits: excellent in vitro and in vivo antioxidants, J. Am. Coll. Nutr. 24 (1) (2005) 44–50.
[2] V. Meshram, C. Choudhary, A. Kale, J. Rajput, V. Meshram, A. Dhu- mane, Dry fruit image dataset for machine learning applications, Data Brief 49 (2023) 109325.
[3] R. Jan, H. Kour, J. Manhas, V. Sharma, Recognition of dry fruits using deep convolutional neural network, Int. J. Res. Appl. Sci. Eng. Technol. 9 (2021).
[4] H. Altaheri, M. Alsulaiman, G. Muhammad, Date fruit classification for robotic harvesting in a natural environment using deep learning, IEEE Access 7 (2019) 117115–117133.
[5] P. Karthickumar, V. Sinija, K. Alagusundaram, Indian cashew process- ing industry-an overview, J. Food Res. Technol. 2 (2) (2014) 60–66.
[6] E. Gajewska, M. Sobieska, W. Samborski, Manual ability classification system for children with cerebral palsy, Chir. Narzadow Ruchu Ortop. Pol. 71 (4) (2006) 317–319.
[7] A. Maxwell, R. Li, B. Yang, H. Weng, A. Ou, H. Hong, Z. Zhou, P. Gong, C. Zhang, Deep learning architectures for multi-label classification of intelligent health risk prediction, BMC Bioinf. 18 (2017) 121–131.
[8] T.A. Nash, V. Pramod, R.D. McDaniel, J.S. Pitts, C.M. Griffith, Classification of Items with Manual Confirmation, Oct. 29 2013 uS Patent 8,572,013.
[9] O. Kilanko, S.J. Ojolo, R.O. Leramo, T.A. Ilori, S.O. Oyedepo, P.O. Babalola, O. S. Fayomi, P.N. Onwordi, E. Ufot, A. Ekwere, Dataset on physical properties of raw and roasted cashew nuts, Data Brief 33 (2020) 106514, https://doi.org/10.1016/j.dib.2020.106514. https://www.sciencedirect.com/science/article/pii/S2352340920313962.
[10] V. Gautam, R.G. Tiwari, A. Misra, D. Witarsyah, N.K. Trivedi, A.K. Jain, Dry fruit classification using deep convolutional neural net- work trained with transfer

learning, in: 2023 International Conference on Advancement in Data Science, E-Learning and Information System (ICADEIS), IEEE, 2023, pp. 1–6.
[11] P. Ghose, M.A. Uddin, M.M. Islam, M. Islam, U.K. Acharjee, A breast cancer detection model using a tuned svm classifier, in: 2022 25th International Conference on Computer and Information Technology (ICCIT), IEEE, 2022, pp. 102–107.
[12] N. Karimi, R.R. Kondrood, T. Alizadeh, An intelligent system for qual- ity measurement of golden bleached raisins using two comparative ma- chine learning algorithms, Measurement 107 (2017) 68–76.
[13] M. Dirik, Improving raisin grains classification with a hybrid pso-nn approach, in: 1st International Conference on Contemporary Academic Research (ICCAR 2023), 2023, pp. 17–19.
[14] M. Arora, V. Devi, A machine vision based approach to cashew kernel grading for efficient industry grade application, IJARIIT 4 (6) (2018) 865–871.
[15] J.M. Ponce, A. Aquino, J.M. Andujar, Olive-fruit variety classification by means of image processing and convolutional neural networks, IEEE Access 7 (2019) 147629–147641.
[16] R. Ramya, P. Kumar, K. Sivanandam, M. Babykala, Detection and classification of fruit diseases using image processing & cloud computing, in: 2020 International Conference on Computer Communication and Informatics (ICCCI), IEEE, 2020, pp. 1–6.
[17] H.S. Gill, B.S. Khehra, Fruit Image Classification Using Deep Learning, 2022.
[18] D. Müller, I. Soto-Rey, F. Kramer, An analysis on ensemble learning optimized medical image classification with deep convolutional neural networks, IEEE Access 10 (2022) 66467–66480.
[19] A.U. Berliana, A. Bustamam, Implementation of stacking ensemble learning for classification of covid-19 using image dataset ct scan and lung x-ray, in: 2020 3rd International Conference on Information and Communications Technology (ICOIACT), IEEE, 2020, pp. 148–152.
[20] I. Kandel, M. Castelli, A. Popovic, Comparing stacking ensemble tech- niques to improve musculoskeletal fracture image classification, J. Imag. 7 (6) (2021) 100.
[21] S.K. Vidyarthi, S.K. Singh, R. Tiwari, H.-W. Xiao, R. Rai, Classifica- tion of first quality fancy cashew kernels using four deep convolutional neural network models, J. Food Process. Eng. 43 (12) (2020) e13552.
[22] M.M. Islam, G. Karmakar, J. Kamruzzaman, M. Murshed, Measuring trustworthiness of iot image sensor data using other sensors' complemen- tary multimodal data, in: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2019, pp. 775–780.
[23] M.M. Islam, J. Kamruzzaman, G. Karmakar, M. Murshed, G. Ka- handawa, Passive detection of splicing and copy-move attacks in image forgery, in: Neural Information Processing: 25th International Con- Ference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part IV 25, Springer, 2018, pp. 555–567.
[24] Cyborg, What is early stopping in deep learning?, URL, https://cyborgcodes.medium.com/what-is-early-stopping-in-deep-learning-eeb1e710a3cf, Apr 2024.