

# Robust Classification of Black-Eyed Peas Based on Segment Anything Model and Transfer Learning

Sachin Sonawane<sup>1</sup>, Suresh Kurumbanshi<sup>2</sup>

<sup>1</sup>SVKM's NMIMS, Mukesh Patel School of Technology Management and Engineering, Shirpur, Maharashtra, India – 425405.  
sachin.sonawane@nmims.edu

<sup>2</sup>SVKM's NMIMS, Mukesh Patel School of Technology Management and Engineering, Shirpur, Maharashtra, India – 425405.  
suresh.kurumbanshi@nmims.edu

## ARTICLE INFO

## ABSTRACT

Received: 21 Dec 2024

Revised: 01 Feb 2025

Accepted: 14 Feb 2025

Evaluating the physical quality of harvested black-eyed peas is essential to ensure their products meet high standards. Carefully designed and optimized machine learning models can provide better quality evaluation. A hybrid neural network integrating EfficientNetV2B1 and Vision Transformer (ViT) to classify black-eyed peas is introduced in this work. One of the main challenges in accurate classification was segmenting objects in a clustered view. Inconsistent lighting, variations in sample size, random placement of the objects, and neighboring objects touching each other make the task difficult. We utilized the Segment Anything Model (SAM) to address the issue. SAM detected individual objects for our samples of weights up to 30 grams with 100% accuracy. We incorporated SAM with our custom object retrieval block to separate the segmented objects into images of size  $224 \times 224 \times 3$  pixels for classification purposes. We also used image augmentation with a stable diffusion method to balance the dataset. Stable diffusion generates high-quality and diverse images while preserving the original distribution. Subsequently, we experimented with five hybrid architectures, EfficientNetV2B1+ViT, MobileNetV2+ShuffleNetV2, ResNet50+DenseNet121, VGG16+ResNet18, and InceptionV3+MobileNetV2 with feature fusion addressed by the convolutional block attention module (CBAM). Our experimentation showed that the EfficientNetV2B1+ViT model outperformed other models. EfficientNetV2B1+ViT exploited depth-wise separable convolution and transformer-based models utilizing multi-head self-attention mechanisms. With hyperparameter optimization, EfficientNetV2B1+ViT achieved an impressive accuracy of 95.80% and a loss of 0.1256 across eight classes of sound-quality seeds, defects, and foreign contamination, highlighting its efficiency and robustness.

**Keywords:** Black-eyed peas classification, Segment anything model, Object detection, Hybrid architecture, Attention mechanism

## 1. INTRODUCTION

Black-eyed peas are a valuable crop for food industries. Their nutritional benefits and applications make them vital in food product manufacturing. The products include animal feed, frozen food, canned and dried products, soups and stews, etc. However, anomalies like discoloration, damage, insect infestation, and contamination reduce the nutritional value of black-eyed peas and may convert them into less-valued bio-products. Therefore, food industries must conduct quality evaluations of harvested black-eyed peas before manufacturing products [1-2]. In quality evaluation, assessing the physical properties of harvested black-eyed peas is essential. Conventionally, food industries and commercial markets conduct physical quality evaluations through a non-invasive manual inspection. In this process, an evaluator draws random samples of specific weight from harvested black-eyed peas lot and inspects them according to the domestic or international grade standards [1-2]. Then, evaluators decide on the quality of the lot based on the results from the assessment of samples and fix its rate. The manual evaluation process is erroneous and leads to financial losses. Therefore, in the past few years, researchers have shifted their focus toward developing computer vision and machine learning-based automatic systems [3]. Image acquisition, image processing, object segmentation, feature extraction, and object classification are the sequential blocks of such systems [4, 5]. Here, we briefly discuss these developments.

Ansari et al. [6] focused on agricultural product classification by extracting features such as color, texture, and area for input into support vector machines (SVM). Zhao et al. [7] enhanced MobileNetV2 to detect surface defects in soybean seeds, experimenting with different intensities of light illumination. Huang et al. [8] segmented soybeans using R-CNN and developed SNet, achieving an impressive accuracy of 96.20%. Chatterjee et al. [9] utilized an ensemble machine learning approach to classify three wheat varieties based on seven physical features: perimeter, area, major-axis length, minor-axis length, asymmetry coefficient, compactness, and groove length of the wheat kernel. The ensemble classifier achieved an accuracy of 95%. Lin et al. [10] introduced an approach to classify soybeans under varying illumination conditions using online deep-learning models, claiming a classification accuracy of 95.63%. Song et al. [11] improved an Inception-ResNet network using depth-wise separable convolution, an attention mechanism, and feature fusion methods to assess the appearance of maize seeds, achieving an average accuracy of 96.03%. Zhang et al. [12] utilized hyperspectral imaging to acquire images of nine maize seed varieties with 3D image features and employed a five-layer convolutional neural network (CNN) to classify the maize seeds into their respective varieties, achieving an accuracy of 96.65% on the test images. Lin et al. [13] developed an algorithm to separate soybean seeds in physical contact with each other. They used a multi-scale retinex method with a color restoration technique to improve the image contrast. They applied Otsu's adaptive thresholding method to segment objects. The minimum bounding rectangle method was employed to locate individual objects. Ghimire et al. [14] attempted to assess the physical quality of soybean seeds by comparing the physical parameters obtained manually with those derived using software tools SmartGrain and WinDIAS. They used contour detection methods to segment individual objects. Wang et al. [15] implemented hyperspectral imaging with machine learning algorithms to classify sound-quality and insect-infested maize seeds. For this purpose, they applied 1D-CNN-BiLSTM and SVM on the spectral and texture features of the seeds. The authors achieved an overall accuracy of 96% with reduced complexity in the process. Sable et al. [16] designed a lightweight neural network to identify defects in soybean seeds. The model was customized using depth-wise convolution and squeeze-and-excitation attention mechanisms to enhance the classification accuracy.

These developments describe the importance of efficient and accurate instance segmentation and classification methods. Instance segmentation methods face challenges due to the random spatial distribution of instances, such as seeds and anomalies, occurring in clustered formations of physically touching neighboring objects. Moreover, factors like changes in illumination, sample sizes, and random placement of the objects add complexity to the segmentation process. Also, instance segmentation algorithms must be fast and highly accurate.

Similarly, in object classification, CNNs have consistently outperformed traditional machine learning methods, classifying segmented instances into sound-quality seeds and their anomalies. However, the accuracy of CNNs largely depends upon the weights assigned to their neurons during training. When constructing CNNs from scratch, extensive training on a large image dataset is required to achieve a higher classification accuracy. For this reason, a transfer learning approach is often preferred over deploying an untrained CNN. Despite these advancements, there remains significant scope for improving the classification performance. Utilizing customized pre-trained neural networks can enhance the precision and accuracy of classification, ultimately resulting in a more efficient process with better outcomes.


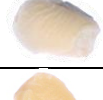




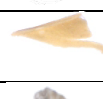

In this work, we proposed an enhanced hybrid architecture, combining EfficientNetV2B1 and ViT, to classify harvested black-eyed peas into sound-quality kernels and their anomalies. An accurate and robust instance segmentation was realized through a state-of-the-art instance segmentation algorithm, SAM. SAM effectively managed varying illumination conditions and sample sizes, accurately segmenting objects of different colors, shapes, and sizes. Its integration with our customized object retrieval block enabled the extraction of individual objects along with their constituent RGB colors into separate images of size  $224 \times 224 \times 3$  pixels, fulfilling the necessary image size requirement for pre-trained classification models. We used a stable diffuser for image augmentation to increase the sample size to 1000 images per class. This strategy helped to balance the input dataset and avoided problems of overfitting and underfitting. We developed a hybrid model of EfficientNetV2B1 and ViT, gaining the advantages of convolutional and transformer-based architectures. The model used depth-wise separable convolutions, multi-head self-attention mechanisms, and feature fusion techniques inspired by CBAM. As a result, the model demonstrated an efficient and reliable approach, achieving higher classification accuracy while maintaining an optimal computational cost. The dataset consisted of eight classes with sound-quality seeds, defects, and foreign contamination.

## 2. MATERIALS AND METHODS

### 2.1. Black-eyed peas sample

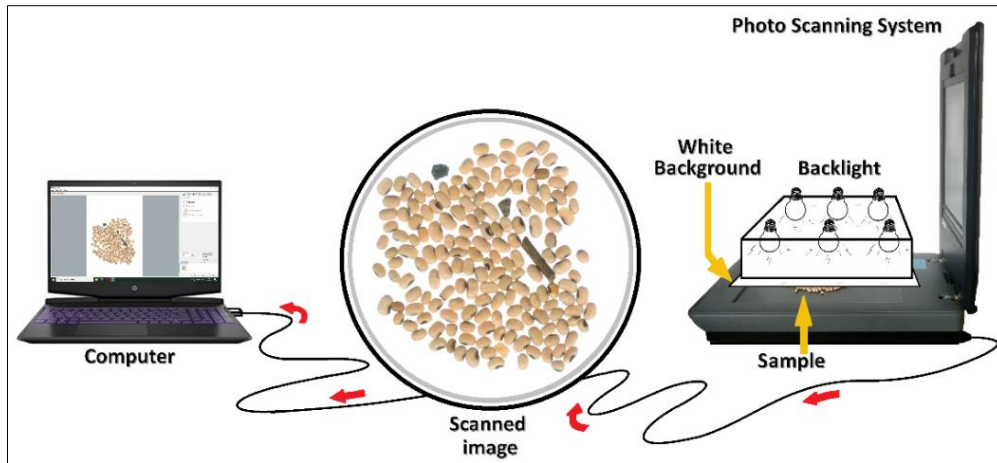
A random sample of black-eyed peas was collected from local farmers in Dhule, Maharashtra, India. The sample weighed 600 grams with a moisture content of around 13%. It was a harvest from February 2024 and comprised sound-quality black-eyed peas alongside various anomalies characterized by their physical properties. The sample was manually sorted into eight distinct categories under the supervision of experts from the black-eyed pea industry. Table 1 summarizes the physical properties and object counts in the sample of harvested black-eyed peas [1,2].

**Table 1.** Physical properties and count of objects in the 600-gram training sample [1,2]

Object category	Image	Physical properties	Count
Sound-quality black-eyed peas		It is a wholesome, fully mature, dry, and clean black-eyed peas kernel with a yellow-brownish color and an oval shape.	1851
Broken and cracked black-eyed peas		The unbroken part is oval, and the broken is random in shape. It has a cracked outer seed coat, and it is yellow-brownish in color.	425
Split black-eyed peas		It is a broken black-eyed pea without an outer coat on the split side. It may be one of the two halves formed due to the breakage of black-eyed peas along the grain. It is lighter yellow-brownish in color. The shape may be random.	482
Damaged and discolored black-eyed peas		Peas or pieces of peas diseased, discolored, mouldy, or materially damaged due to heat, bad weather, moisture or microbial action, or sprouted.	579
Immature and shrivelled black-eyed peas		This category includes greenish, non-fully developed, or shrunk black-eyed peas.	421
Insect infested black-eyed peas		Includes peas that are bored by insects. Usually, the non-infested part is yellow-brownish in color, and its shape is oval. The infested or bored part looks in a different color than the non-infested part.	376
Organic foreign materials		This category includes husks, stems, chaffs, and straws of random shape and color.	531
Inorganic foreign materials		This category includes lumps of earth, sand, and stones of random shape and color.	495

## 2.2. Image acquisition

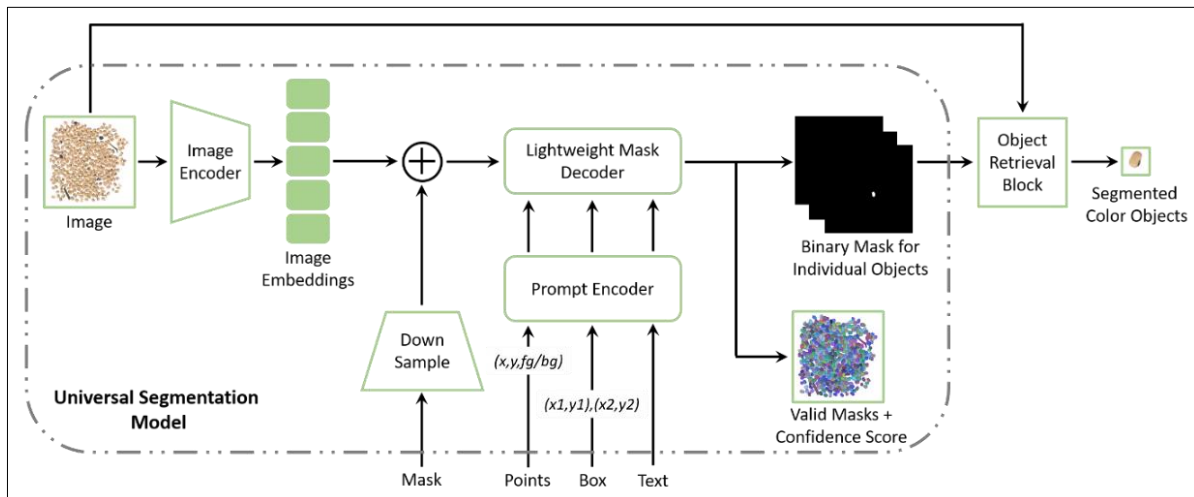
Fig. 1 illustrates the setup we used for imaging the sample of harvested black-eyed peas. The setup comprised an HP Scanjet G4050 scanner equipped with a charge-coupled device (CCD) and a computer to store and process the scanned images. The setup was configured to acquire sample images at a resolution of 300 pixels per inch and produce 2D color images sized  $3300 \times 3300$  pixels. We divided each category of the collected sample into smaller portions, each weighing up to 30 grams maximum, to accommodate the sample on the scanning surface. A plain white paper background was selected for higher contrast with various objects within the sample. A bank of flicker-free 14-watt bulbs was utilized for backlighting, which helped to mitigate shadows during image acquisition. For each scan, the distance between the bulbs and the plain white paper background was randomly selected between 6 cm and 12 cm to perform imaging at different intensities of light illumination. Accordingly, images were captured for each sample of weight up to 30 grams. The sample objects were arranged on the scanning surface in non-overlapping positions to avoid information loss about any hidden parts of the objects.



**Fig. 1** Setup for photo-scanning harvested black-eyed peas

### 2.3. Detection of individual objects from the scanned images

Each scanned image obtained during the image acquisition stage contained numerous objects. It was essential to detect each object from the scanned images to train and test machine learning-based classifiers. To fulfill this requirement, the scanned images were processed for noise removal using Otsu's threshold technique and subsequently underwent instance segmentation to identify individual objects. We employed a state-of-the-art algorithm, SAM, with our customized object retrieval block to segment every instance from scanned images (refer to Fig. 2) [17].



**Fig. 2** SAM integrated with the object retrieval block

SAM is a lightweight CNN architecture featuring a U-Net structure for efficient feature extraction and up-sampling. The architecture includes skip connections between the encoder and decoder layers to preserve spatial information and combines features from multiple scales to capture local and global contexts. SAM employs a transformer-based architecture to embed prompt tokens or image grid points into fixed-size embedding outputs. Its vision transformer converts image patches into a sequence of feature embeddings. Additionally, SAM incorporates cross-attention between the prompts or image grid points and image encoders to emphasize relevant regions within the image. It utilizes a pixel-wise classification approach to predict instance masks, computing a probability map (confidence score) for each instance. SAM is pre-trained on a combination of image classification, masked image modeling, and instance segmentation objectives while using fewer parameters, contributing to its high accuracy in instance segmentation [17]. In this work, SAM is configured to segment every instance in the image, producing a binary mask for each detected instance. The mask size is the same as the input image. Subsequently, the object retrieval block utilizes this binary mask to extract the object associated with that instance from the original image in color format, resulting in separate small-sized images of the required dimensions. In this work, the small-sized images were set to a size of  $224 \times 224 \times 3$  pixels. Table 2 presents the pseudo-algorithm for the object retrieval algorithm.

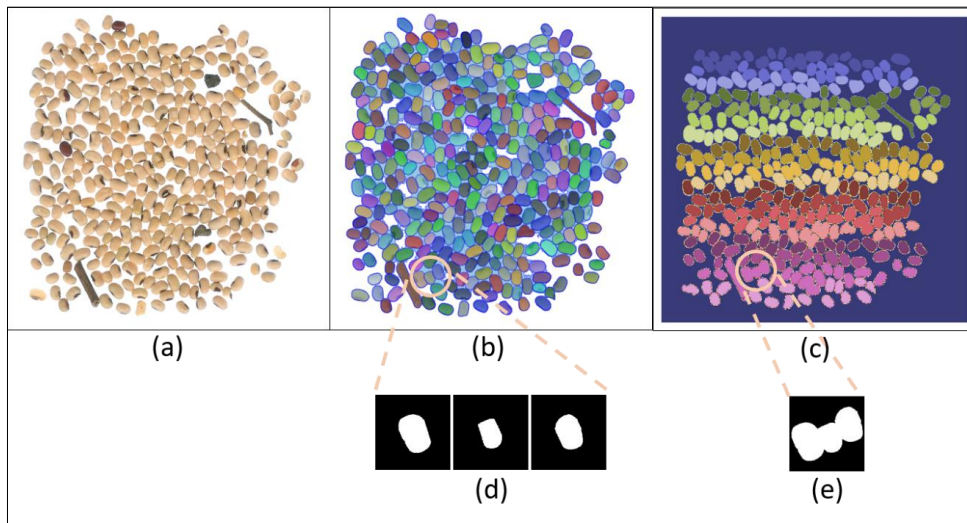


The pseudo-algorithm begins by extracting the  $H$ ,  $S$ , and  $V$  color channels from the scanned image  $I$ , resulting in images  $HI$ ,  $SI$ , and  $VI$  of the same size ( $row \times col$  pixels) as the original images  $I$  and  $B$ . The algorithm then visits each binary mask  $B$  one by one. During this process, the centroid of an object within the binary mask is computed by utilizing the properties of the white area of that object. The row index 's' and the column index 't' denotes the centroid. Next, the algorithm operates within a window of size  $224 \times 224$ , defined as  $(s - 111 : \Delta: s + 111, t - 111 : \Delta: t + 111)$ , to visit all the white pixels of the object in image  $B$ . The window size of  $224 \times 224$  is selected as it accommodates various objects in the sample and meets the requirements of the pre-trained models implemented in this work. While iterating through the window, whenever the algorithm encounters a white pixel in image  $B$ , it assigns the pixel values from the corresponding location in images  $HI$ ,  $SI$ , and  $VI$  to the pixels at the locations computed for the blank images  $H$ ,  $S$ , and  $V$ , which are also of size  $224 \times 224$  pixels. The algorithm calculates the pixel locations in images  $H$ ,  $S$ , and  $V$  based on the size differences (row, col) between  $HI$ ,  $SI$ , and  $VI$  and the  $224 \times 224$  pixels of  $H$ ,  $S$ , and  $V$ . Finally, the extracted  $H$ ,  $S$ , and  $V$  color channels are merged to reconstruct the object in a separate color image of size  $224 \times 224 \times 3$  pixels.

**Table 2.** Pseudo algorithm for the object retrieval block

- 
1. Initialize:  $I_{row \times col} \leftarrow \text{Scanned image after noise removal},$   
 $B_{row \times col} \leftarrow \text{Binary mask for individual objects},$   
 $H_{224 \times 224} \leftarrow 0, S_{224 \times 224} \leftarrow 0, V_{224 \times 224} \leftarrow 0, Object_{224 \times 224} \leftarrow 0, \Delta \leftarrow 1$
  2.  $HI_{row \times col}, SI_{row \times col}, VI_{row \times col} \leftarrow HSV(I_{row \times col})$
  3.  $N \leftarrow \text{count}(\text{binary mask for individual objects})$
  4. For ( $n \leftarrow 1 : \Delta: N$ ):
  5.    $(s, t) \leftarrow \text{centroid\_of\_white\_area}(B_{row \times col})$
  6.   For ( $a \leftarrow s - 111 : \Delta: s + 111, b \leftarrow t - 111 : \Delta: t + 111$ ):
  7.     If  $B_{row \times col}(a, b) == 1$  then:
  8.        $H_{224 \times 224}(111 - (s - a - 1), 111 - (t - b - 1)) \leftarrow HI_{row \times col}(a, b)$
  9.        $S_{224 \times 224}(111 - (s - a - 1), 111 - (t - b - 1)) \leftarrow SI_{row \times col}(a, b)$
  10.        $V_{224 \times 224}(111 - (s - a - 1), 111 - (t - b - 1)) \leftarrow VI_{row \times col}(a, b)$
  11.    $Object_{224 \times 224} \leftarrow \text{merge}(H_{224 \times 224}, S_{224 \times 224}, V_{224 \times 224})$
- 

To evaluate the performance of SAM integrated with the object retrieval block, we implemented an additional, widely recognized instance segmentation algorithm, the watershed algorithm [18]. Fig. 3 illustrates the effectiveness of SAM compared to the watershed algorithm. SAM demonstrated a remarkable ability to accurately segment all objects within a given sample image, achieving 100% accuracy even in the presence of objects exhibiting minor or significant variations in color, shape, and size, as well as those positioned randomly and in physical contact.

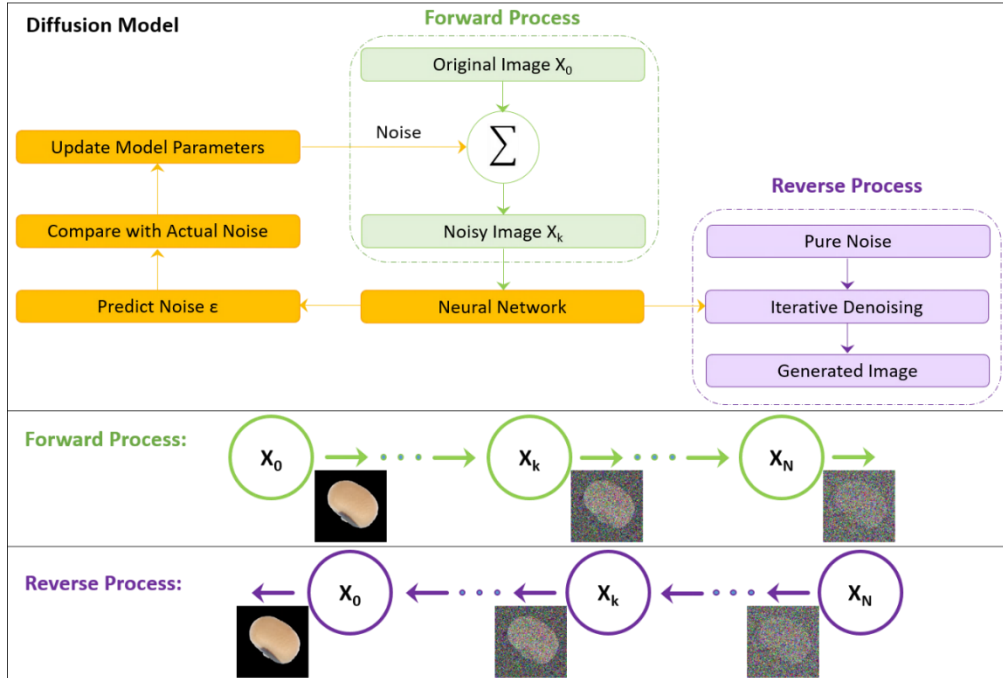


**Fig. 3** (a) Scanned image of a 30-gram sample after noise removal, (b) Instance segmentation using SAM, (c) Instance segmentation using the Watershed algorithm, (d) SAM accurately produced individual binary masks for the three objects highlighted, (e) The Watershed algorithm inaccurately produced only one binary mask for the three objects

#### 2.4. Image Augmentation

Usually, a sample of harvested black-eyed peas contains anomalies in a smaller proportion than sound-quality kernels. It leads to an imbalance within the training dataset, resulting in data overfitting and an inherent bias in machine learning models toward the majority class. In such a case, although a machine learning model effectively learns from the training data, it inadequately performs on unseen data during the validation and testing phases. Image augmentation is a solution to the problem. This work implemented a generative model-based image augmentation technique known as stable diffusion to obtain synthetic images for the sample categories with an object count of less than 1000. Traditional image augmentation methods, like rotation, crop, and flip, have limitations in enhancing image diversity effectively. The stable diffusion model was trained on color images of individual objects from the respective classes [19-21].

Stable diffusion can generate high-quality and diverse images. This method reduces the risk of mode collapse while preserving the original image distribution. The generated images appear realistic. Moreover, diffusion models produce more coherent images with better contrast than variational autoencoders (VAEs) and generative adversarial networks (GANs). These high-quality images contribute to more stable training of neural networks. The stable diffusion model primarily operates through three key processes: forward, reverse, and noise prediction (refer to Fig. 4) [19-21].



**Fig. 4** Forward, reverse, and noise prediction processes in the stable diffusion model

In the forward process, Gaussian noise is gradually added to the input image over a series of steps. Therefore, the process is also known as the diffusion process. Equations (1) and (2) describe the forward process. Let  $X_0$  denote the input image, and  $X_1, X_2, X_3, \dots, X_k$  the noise versions of  $X_0$ . 'k' is the number of noise steps.

$$D(X_k|X_{k-1}) = G(X_k; \sqrt{(1 - \beta_k)} X_{k-1}, \beta_k I) \quad (1)$$

$$D(X_{1:N}|X_0) = \prod_{k=1}^N D(X_k|X_{k-1}) \quad \{\beta_k \in (0,1)\}_{k=1}^N \quad (2)$$

Where,

- $D$  is the forward diffusion process
- $G$  represents a Gaussian distribution
- $\beta_k$  is the noise schedule (typically increases with  $k$ )
- $I$  is the identity matrix

The noise prediction process minimizes the difference between the predicted and actual noise. Equation (3) describes the objective function of the noise prediction process.

$$L = E[\|\epsilon - \epsilon(X_k, k)\|^2] \quad (3)$$

Where,

- $\epsilon(X_k, k)$  is the model's prediction of the added noise

Learning the reverse process is the main objective of the diffusion model. In the reverse process, the iterative denoising of the image takes place, which undoes the forward process. Equation (4) describes the reverse process.

$$R(X_{k-1}|X_k) = G(X_{k-1}; \mu(X_k, k), \Sigma(X_k, k)) \quad (4)$$

Where,

- $R$  is the learned reverse process
- $\mu$  is the predicted mean
- $\Sigma$  is the predicted covariance

Thereafter, starting with the pure noise  $X_k$ , the learned reverse process is iteratively applied to generate new images. Equation (5) describes the process for generating new images.

$$X_{k-1} = \frac{1}{\sqrt{\alpha_k}} \left( X_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} \varepsilon(X_k, k) \right) + \sigma_k z \quad (5)$$

Where:

- $\alpha_k = 1 - \beta_k$
- $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$
- $\sigma_k$  is a small amount of stochastic noise
- $z$  is standard Gaussian noise

Following image augmentation, the count of images or objects for each category is 1000 or more. Only 1000 images per category were considered for building the classification model. Specifically, 1000 images were randomly selected for sound-quality soybeans, as the count exceeds this value. The images were then divided into datasets in the proportion of 70:20:10, resulting in 700 images for the training dataset, 200 for the validation dataset, and 100 for the test dataset.

### 3. CLASSIFICATION MODEL

Pre-trained CNNs are deep-learning models that develop their weight matrices through training based on large datasets, such as ImageNet [22]. The performance of the pre-trained CNNs can be further improved using techniques like hybrid models, feature fusion, and attention mechanisms. These strategies enable CNN architectures to combine the strengths of different models. In this work, we implemented an improved hybrid model that combines the strengths of EfficientNetV2B1 and ViT, integrating CBAM for feature fusion.

#### 3.1. Improved architecture of EfficientNetV2B1+ViT

Fig. 5 illustrates an improved architecture of the hybrid model of EfficientNetV2B1 and ViT. The model consists of two parallel branches of EfficientNetV2B1 and ViT, each fed with input images of dimensions  $224 \times 224 \times 3$  pixels for feature extraction. Features obtained from both branches are combined using the feature fusion block and then used by the classification head to classify an object in the input image into one of the eight classes.

##### 3.1.1. EfficientNetV2B1

In the EfficientNetV2B1 branch, the initial feature extractor is the stem Conv2d layer, which employs a  $3 \times 3$  convolution to transform the input from 3 channels into 32 feature maps. As described in equation (6), the operation  $Y_1(i, j)$  performs a convolution between the input image  $X$  with patch  $[i+k, j+l]$  and the kernel weights  $W$ . Convolution detects specific patterns in the image, such as edges, corners, etc. by applying a kernel to the image. The output ( $Y_1$ ) is then normalized using the mean ( $\mu$ ), standard deviation ( $\sigma$ ), offset ( $\beta$ ), and scaling factor ( $\gamma$ ) as described in equation (7). The term  $\varepsilon$  denotes a small constant that ensures numerical stability. As expressed in equation (8), the Swish activation function introduces non-linearity in normalized output ( $Y_2$ ) while maintaining smooth gradients [23].

$$Y_1(i, j) = \sum \sum X(i+k, j+l) \times W(k, l) \quad (6)$$

$$Y_2 = \gamma \times \frac{Y_1 - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \quad (7)$$

$$Y_3 = Y_2 \times \text{sigmoid}(Y_2) \quad (8)$$

Where,

- $i, j$  are the spatial indices of the image

Further, the MBConv block implements depth-wise separable convolutions, which process each channel followed by a point-wise convolution that mixes channel information. Equation (9) presents the mathematical formulation for depth-wise separable convolution, while equation (10) describes the mathematical representation for point-wise

convolution. The output from the depth-wise separable convolutions undergoes batch normalization and is subsequently processed using the activation functions. In the same way, the outcome from point-wise convolutions is also subjected to batch normalization.

$$Y_4(i, j, k) = \sum \sum Y_3(i + p, j + q, k) \times W_d(p, q, k) \quad (9)$$

$$Y_5(i, j, n) = \sum Y_4(i, j, k) \times W_p(k, n) \quad (10)$$

Where,

- $k$  is input channel index
- $p, q$  are kernel spatial indices with patch  $[i+p, j+q]$ .
- $n$  is output channel index

The Squeeze-Excitation (SE) mechanism calculates global context ( $z$ ) and channel-specific scaling factors ( $s$ ) using learned weights. The SE block averages each channel separately. Equations (11) and (12) describe the mathematical formulations relevant to the SE block.

$$Z(k) = \frac{1}{H \times W} \times \sum \sum Y_5(i, j) \quad (11)$$

$$S = \sigma(W_2 \delta(W_1 Z)) \quad (12)$$

Where,

- $H$  is height and  $W$  is width of the feature map from the stem Conv2d layer
- $\sigma$  is sigmoid and  $\delta$  is ReLU
- $W_1$  and  $W_2$  are the learnable weights specific to the SE block's fully connected layers.

The global average pooling layer calculates the average value across each 2D feature map and converts it into a single value. The projection layer performs feature transformation and reduces dimensions to match dimensions for fusion.

### 3.1.2. ViT

In the ViT branch, the patch embedding block divides an input image into non-overlapping patches, linearly embedding each patch into a token. As illustrated in Fig. 5, a total of 196 patches are created, each of size  $16 \times 16 \times 3$ , yielding a dimension of 768. Each patch is linearly projected into a 768-dimensional space utilizing a learned embedding matrix. Further, the class token prepend block leverages the ViT's capacity to model global image features for the classification tasks. The class token represents the entire image, enabling the model to aggregate information from all the patches. By inserting the class token at the beginning of the patch embedding sequence, the resulting output shape becomes  $(197 \times 768)$ . Positional embeddings are incorporated to maintain spatial information. The Transformer Encoder Block comprises two sub-layers, Multi-Head Self-Attention (MSA) and Multi-Layer Perceptron (MLP). The MSA mechanism attends to various parts of the input sequence, while the MLP transforms activations through two linear layers followed by a ReLU activation function. The ReLU activation introduces non-linearity into the model. Moreover, the layer normalization (LayerNorm) layer calculates the mean ( $\mu$ ) and variance ( $\sigma^2$ ) across the three channels for normalizing activations and conducting training smoothly [24].

### 3.1.3. Feature fusion

The feature fusion block concatenates the features from the EfficientNetV2B1 and ViT branches. It employs an attention mechanism emphasizing relevant context within the input data to make predictions. It allocates greater attention to vital information and lower attention to less important information. Commonly used attention mechanisms are channel attention, self-attention, and spatial attention. The channel attention mechanism emphasizes the color channels of images relevant to the task. The self-attention mechanism captures complex relations between various regions in the image to determine remote connections efficiently. Similarly, the spatial attention mechanism is helpful when an image region is more important for classification.

This work considered the attention mechanism CBAM for focusing only on the vital features to discriminate harvested black-eyed peas. CBAM is suitable as the relevant information in black-eyed peas images is distributed spatially across the image rather than localized in a fixed region/s. Spatial locations provide features to classify objects based on shape, size, texture patterns, etc. CBAM combines spatial attention with a channel attention mechanism to selectively emphasize relevant channels and spatial regions. Being a spatial attention mechanism, CBAM is lightweight and requires less computation. The following equations explain the process of CBAM [25].



Here,  $X \in \mathbb{R}^{(H \times W \times C)}$  is the input tensor of features belonging to space  $R$  of height ( $H=28$ ), weight ( $W=28$ ), and number of channels ( $C=1024$ ). The global average pooling computes the average, and global max pooling finds the maximum value across each feature map, converting each 2D feature map into a single value as described in equations (13) and (14).

$$\text{Global Average Pooling: } X_{avg} = \frac{1}{H \times W} \sum_{h=1}^H \sum_{w=1}^W X(H, W, C) \quad (13)$$

$$\text{Global Max Pooling: } X_{max} = \max\{X(H, W, C) \mid h \in [1, H], w \in [1, W]\} \quad (14)$$

MLP transformation occurs with a reduction ratio of  $r = 16$ , causing the hidden layer dimensions of  $l = C/r = 64$ . Equation (15) expresses MLP transformations that computes a 1D channel attention vector  $\alpha_c \in \mathbb{R}^{(1 \times 1 \times C)}$  after global average pooling and global max pooling.

$$\alpha_c = \text{Sigmoid}\{[WC_2 \times \text{ReLU}(WC_1 \times X_{avg} + b_1) + b_2] + [WC_2 \times \text{ReLU}(WC_1 \times X_{max} + b_1) + b_2]\} \quad (15)$$

Where,  $WC_1 \in \mathbb{R}^{(l \times C)}$  and  $WC_2 \in \mathbb{R}^{(C \times l)}$  are the weights, and  $b_1 \in \mathbb{R}^{(l)}$  and  $b_2 \in \mathbb{R}^{(C)}$  are the biases of fully connected layers with MLP transformations for global average pooling and global max pooling.

Then, CBAM performs an element-wise multiplication between  $X$  and  $\alpha_c$  to refine the features per the importance of each color channel, producing  $X' \in \mathbb{R}^{(H \times W \times C)}$  as expressed in equation (16).

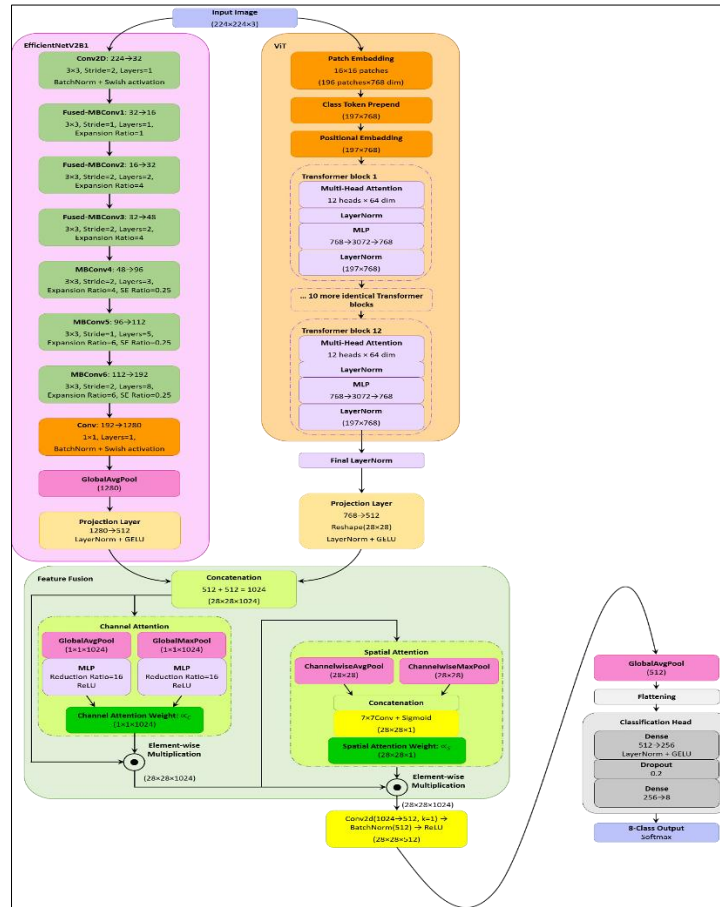
$$X' = \alpha_c \odot X \quad (16)$$

Further, the tensor  $X'$  is fed to the spatial attention mechanism, which performs channel-wise average pooling and max pooling, producing  $X'_{avg} \in \mathbb{R}^{(H \times W)}$  and  $X'_{max} \in \mathbb{R}^{(H \times W)}$ . The concatenation block combines  $X'_{avg}$  and  $X'_{max}$ , producing  $X'_{concat} \in \mathbb{R}^{(H \times W \times 2)}$ .  $7 \times 7$  Convolution reduces concatenated spatial features from 2 channels to 1 channel with Sigmoid activation to compress values between 0 and 1. Equation (17) expresses the complete operation.

$$\alpha_s = \text{Sigmoid}\left(\text{Conv}_{7 \times 7}(X'_{avg}; X'_{max})\right) \quad (17)$$

Then, CBAM performs an element-wise multiplication between  $\alpha_s \in \mathbb{R}^{(H \times W \times 1)}$  and  $X' \in \mathbb{R}^{(H \times W \times C)}$  to combine the channel and spatial attention mechanisms as expressed in equation (18). The operation produces  $X'' \in \mathbb{R}^{(H \times W \times C)}$ .

$$X'' = \alpha_s \odot X' \quad (18)$$



**Fig. 5** Improved hybrid architecture of EfficientNetV2B1+ViT

Subsequently, the convolutional block downsizes the feature map from the size of  $(28 \times 28 \times 1024)$  to  $(28 \times 28 \times 512)$ . The global average pooling block reduces spatial dimensions  $(28 \times 28)$  to a single value per channel. The flatten layer converts a multidimensional input tensor into a one-dimensional output tensor. Then, the dropout layer performs regularization by erratically dropping a percentage of neurons in the pooling layer during training to prevent overfitting. The dense layer performs a linear transformation by connecting all its neurons to each neuron in the previous layer. Equation (19) expresses output  $A$  of the dense layer with activation function  $f$  for input vector  $x$ .

$$A = f(Wx + b) \quad (19)$$

In the classification head, the dense layers reduce feature dimensionality, and the dropout layer performs regularization by erratically dropping a percentage of neurons in the pooling layer during training to prevent overfitting. Finally, the output layer produces a probability distribution of all possible classes to indicate the model's confidence in each classification. The count of neurons in the output layer equals the count of classes. Typically, multi-class classification tasks utilize a softmax activation function. Equation (20) represents the mathematical form [26].

$$Probability(class_i) = \frac{e^{A_i}}{\sum_j e^{A_j}} \quad (20)$$

Where,  $A_i$  is output of the dense layer for class  $i$ .

#### 3.1.4. Fine-tuning hyperparameters

A classification model with optimized hyperparameters performs better with minimal computations than a non-optimized model. Therefore, it was essential to optimize the hyperparameters of the EfficientNetV2B1+ViT model. The hyperparameters include kernel sizes, activation functions, batch sizes, learning rate, dropout, pooling layer, optimizer, etc. A pooling layer requires a pool size, while a dropout layer necessitates specifying the percentage of neurons to drop. A dense layer requires several kernels, and the output layer requires an activation function, among other considerations. A hyperparameter optimizer minimizes the error (or loss) function during model training. Numerous optimizer algorithms exist, each operating in a specific manner. Stochastic gradient descent (SGD) and adaptive moment estimation (Adam) are widely used optimizers. SGD adjusts the model parameters by moving them in a direction opposite to the gradient of the loss function concerning those parameters. The Adam optimizer maintains moving averages of gradients and squared gradients of the parameters, combining benefits from the root mean square propagation (RMSProp) and adaptive gradient algorithm (Adagrad) methods. RMSProp implements adaptive learning for model parameters by considering the previous mean of the squares of the gradients, thereby mitigating issues such as vanishing or exploding gradients. Adagrad, on the other hand, adjusts the learning rates for individual parameters by accounting for the cumulative sum of squared gradients from the past. This approach effectively addresses sparse data and automatically reduces the learning rates for parameters that receive frequent updates. The learning rate acts as a step size during optimization. It controls the magnitude of adjustments made to the model weights during each training iteration. The appropriate selection of learning rates across a continuum of high and low values is crucial to avoid divergence caused by overly high values and to ensure adequate convergence when faced with lower values [27, 28].

### 4. PERFORMANCE METRICS

We used accuracy, precision, recall, and F1-score as quantitative metrics to evaluate classifiers. The input image dataset was used to compute these metrics. Generally, the classification accuracy on the validation dataset is considered a comprehensive indicator of a model's performance. As expressed in equation (21), accuracy is the ratio of correctly predicted cases to the total cases [9-16].

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative} \quad (21)$$

Precision measures how many predicted positive cases are correctly positive. Equation (22) shows the mathematical representations.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (22)$$

Recall is the relation between correctly predicted positive cases and the total of correctly predicted positive and incorrectly predicted negative cases, as described in equation (23).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (23)$$

The F1-score represents the balanced average of precision and recall, calculated as the harmonic mean of these two metrics. Equation (24) describes the mathematical representation [9-16].

$$F1\_score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (24)$$

## 5. RESULTS AND ANALYSIS

### 5.1. Computer environments

The computer system used for experimentation was equipped with 13<sup>th</sup> generation Intel(R) Core(TM) i7-13700HX, 2.10 GHz, x64-based central processing unit, processor, NVIDIA GeForce RTX 4060 graphic processing unit, and 32 GB RAM. It was operated using Windows 11, a 64-bit operating system. The system configuration included the latest versions of NVIDIA drivers, CUDA toolkit, cuDNN, Python, TensorFlow, PyTorch, and Keras.

### 5.2. Model selection and optimization

In this work, we experimented with five hybrid architectures constructed using some of the most popular pre-trained deep learning models to design a robust model that would not impose heavy computational demands while enhancing its accuracy. These hybrid models included combinations such as EfficientNetV2B1 with ViT, MobileNetV2 paired with ShuffleNetV2, ResNet50 with DenseNet121, VGG16 integrated with ResNet18, and InceptionV3 with MobileNetV2. These models were implemented on our test dataset images in Python without fine-tuning. Table 3 presents the overall performance of all five architectures across eight classes. The results show that the EfficientNetV2B1+ViT architecture achieved better classification accuracy than the others. The hybrid model approach and advanced feature fusion method of CBAM improved the model's performance.

**Table 3.** Performance comparison for all five architectures

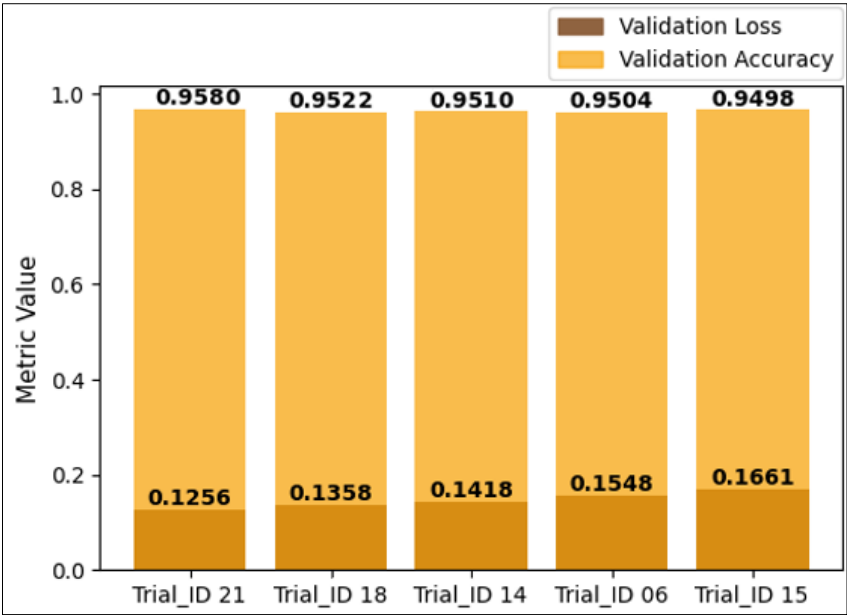
Model	Parameters (M)	Inference Time (ms)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EfficientNetV2B1+ViT	56.3	52	94.60	94.61	94.57	94.54
MobileNetV2 + ShuffleNetV2	13.9	22	91.65	91.37	90.98	91.04
ResNet50 + DenseNet121	50.1	48	93.57	93.22	92.98	92.95
VGG16 + ResNet18	149.8	73	92.94	92.64	92.38	92.37
InceptionV3 + MobileNetV2	29.2	35	92.46	92.43	92.03	92.01

The hyperparameters of EfficientNetV2B1+ViT were optimized to fine-tune their performance on our image dataset. Table 4 lists the hyperparameters and their values considered for fine-tuning. The RandomSearch tuner from the Python library KerasTuner was used for fine-tuning hyperparameters. This tuner employed a random sampling approach for hyperparameter values rather than exhaustively searching through all possible options. Such a method enabled the tuner to efficiently explore a diverse range of hyperparameter combinations [29, 30]. The RandomSearch tuner was configured for 50 trials of various hyperparameter combinations, each running for 40 epochs.

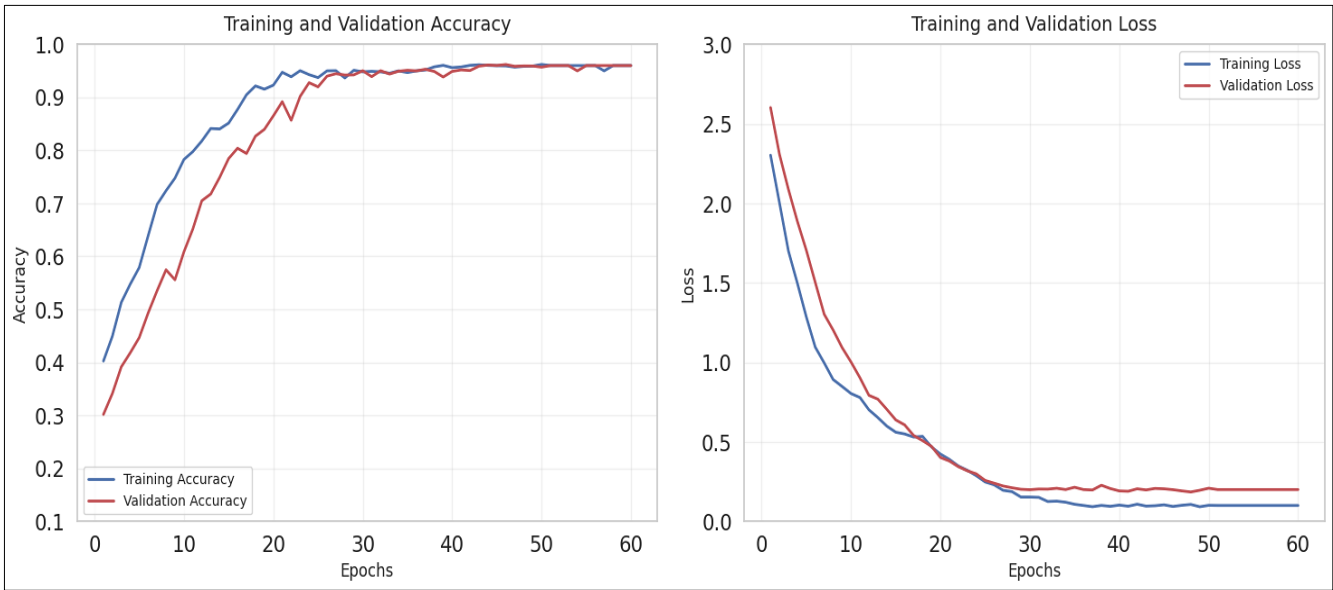
Fig. 6 illustrates the first five trials with the best validation loss and accuracy after hyperparameter tuning of EfficientNetV2B1+ViT. The EfficientNetV2B1+ViT architecture achieved the highest validation accuracy of 95.80% and the lowest validation loss of 0.1256 (trial ID 21) with the following hyperparameter values: Pooling2D (Height = 3, Width = 3), Dropout: 0.2, Optimizer: Adam, Batch Size: 32, and Learning Rate: 0.01. The height and width of Convolutional2D kernels for respective layers are shown in Fig. 5.

**Table 4.** Hyperparameters and the values for fine-tuning [9-16]

Sr. no.	Hyperparameters	Values for tuning
1.	Pooling2D – height of a pool	[1, 2, 3]
2.	Pooling2D – width of a pool	[1, 2, 3]
3.	Dropout	[0.2, 0.3]
4.	Optimizer	[Adam, SGD]
5.	Batch size	[16, 32, 64]
6.	Learning rate	[0.001, 0.01, 0.1]

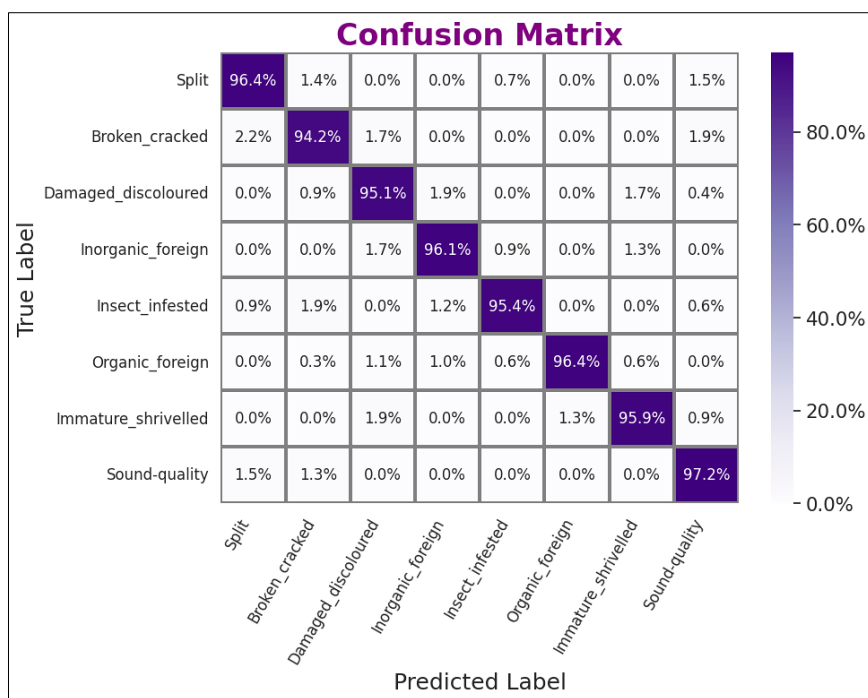


**Fig. 6** Validation loss and accuracy for the best five trials after hyperparameter tuning of EfficientNetV2B1+ViT. Furthermore, the best-performing fine-tuned model was trained on the training dataset, and its performance was evaluated on the validation dataset. As illustrated in Fig. 7, the model initialized with improved accuracy values, likely due to effective optimization techniques. Additionally, it achieved better performance in under 30 epochs, with classification accuracy nearing its peak. Similarly, the loss showed a significant decrease within the same timeframe. Notably, there was no evidence of substantial overfitting or underfitting in the curves representing the training and validation datasets.



**Fig. 7** Accuracy and loss curves for the best-performing fine-tuned EfficientNetV2B1+ViT model for the training and validation datasets

Subsequently, the model was tested on an unseen dataset to evaluate its performance. As depicted in the confusion matrix in Fig. 8, the classification accuracy exceeded 94.0% across all classes. Specifically, the highest accuracy of 97.2% was observed for sound-quality black-eyed peas, and the lowest accuracy of 94.2% was noted for broken and cracked black-eyed peas, highlighting the model's discriminative ability. Overall, the model demonstrated consistent performance across all classes.



**Fig. 8** Confusion matrix for the best-performing fine-tuned EfficientNetV2B1+ViT model for the test dataset

## 6. CONCLUSION

This paper presents a comprehensive framework systematically designed to classify harvested black-eyed peas with enhanced accuracy. We identified and addressed factors that influence or deteriorate the performance of the classification model. Segmenting individual objects was crucial for the object detection block. The object detection block encounters challenges when segmenting individual objects from a clustered view of random samples of black-eyed peas, compounded by varying illumination, sample sizes, random object positioning, and neighboring objects in contact with each other. We employed a fast and accurate instance segmentation algorithm called SAM to tackle the issues. SAM facilitated effective object detection, segmenting objects from the sample images with 100% accuracy. The integration of SAM with our customized object retrieval block provided the extraction of segmented objects into separate pictures of the size necessary for classification.

Furthermore, we implemented image augmentation using stable diffusion to address the problems of overfitting and under-fitting arising from an imbalanced dataset. Stable diffusion generated high-quality, diverse images while reducing the risk of mode collapse and preserving the original image distribution.

For designing a robust and highly accurate classifier, we experimented with five hybrid models constructed from some of the most popular pre-trained neural networks for classifying harvested black-eyed peas into eight classes comprising sound-quality seeds, defects, and foreign contamination. We compared the performance of the hybrid models, including EfficientNetV2B1+ViT, MobileNetV2+ShuffleNetV2, ResNet50+DenseNet121, VGG16+ResNet18, and InceptionV3+MobileNetV2. Our experimentation revealed that the EfficientNetV2B1+ViT model outperformed the other models on our image dataset. This hybrid model benefitted from the convolutional and vision transformer-based methods, capturing local and global features. It used depth-wise separable convolutions and multi-head self-attention mechanisms.

Feature fusion was another concern affecting the accuracy of the hybrid model. We integrated a CBAM that combines and processes features from EfficientNetV2B1 and ViT. CBAM utilized spatial and channel attention mechanisms to emphasize relevant channels and spatial regions. Hyperparameter optimization enhanced the performance of EfficientNetV2B1+ViT, achieving an overall classification accuracy of 95.80% and an overall loss of 0.1256 across all eight classes featuring sound-quality seeds, defects, and foreign contamination. Therefore, the model presented a promising and efficient approach, achieving higher classification accuracy with optimal computational cost.

## Declarations

**Conflict of interest:** The authors confirm that there are no conflicts of interest.



## REFERENCES

- [1] Post-Harvest Profile of cowpeas. <https://www.ams.usda.gov/sites/default/files/media/WholeDryPeas.pdf>. Accessed in September 2024.
- [2] Post-Harvest Profile of cowpeas. [https://igfri.icar.gov.in/wp-content/uploads/2022/09/Folder\\_Fodder\\_Cowpea.pdf](https://igfri.icar.gov.in/wp-content/uploads/2022/09/Folder_Fodder_Cowpea.pdf). Accessed in September 2024.
- [3] Z. A. Sahili and M. Awad, "The power of transfer learning in agricultural applications: AgriNet," *Frontiers in Plant Science*, vol. 13, Dec. 2022, <https://doi.org/10.3389/fpls.2022.992700>.
- [4] Y. S. Taspinar, M. Dogan, I. Cinar, R. Kursun, I. A. Ozkan, and M. Koklu, "Computer vision classification of dry beans (*Phaseolus vulgaris* L.) based on deep transfer learning techniques," *European Food Research and Technology*, vol. 248, no. 11, pp. 2707–2725, Aug. 2022, <https://doi.org/10.1007/s00217-022-04080-1>.
- [5] H. O. Velesaca, P. L. Suárez, R. Mira, and A. D. Sappa, "Computer vision based food grain classification: A comprehensive survey," *Computers and Electronics in Agriculture*, vol. 187, p. 106287, Jul. 2021, <https://doi.org/10.1016/j.compag.2021.106287>.
- [6] N. Ansari, S. S. Ratri, A. Jahan, M. Ashik-E-Rabbani, and A. Rahman, "Inspection of paddy seed varietal purity using machine vision and multivariate analysis," *Journal of Agriculture and Food Research*, vol. 3, p. 100109, Jan. 2021, <https://doi.org/10.1016/j.jafr.2021.100109>.
- [7] G. Zhao *et al.*, "Real-time recognition system of soybean seed full-surface defects based on deep learning," *Computers and Electronics in Agriculture*, vol. 187, p. 106230, Jun. 2021, <https://doi.org/10.1016/j.compag.2021.106230>.
- [8] Z. Huang *et al.*, "Deep learning based soybean seed classification," *Computers and Electronics in Agriculture*, vol. 202, p. 107393, Oct. 2022, <https://doi.org/10.1016/j.compag.2022.107393>.
- [9] A. Khatri, S. Agrawal, and J. M. Chatterjee, "Wheat seed Classification: Utilizing Ensemble Machine Learning approach," *Scientific Programming*, vol. 2022, pp. 1–9, Feb. 2022, <https://doi.org/10.1155/2022/2626868>.
- [10] W. Lin, L. Shu, W. Zhong, W. Lu, D. Ma, and Y. Meng, "Online classification of soybean seeds based on deep learning," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106434, May 2023, <https://doi.org/10.1016/j.engappai.2023.106434>.
- [11] C. Song *et al.*, "Maize seed appearance quality assessment based on improved Inception-ResNet," *Frontiers in Plant Science*, vol. 14, Aug. 2023, <https://doi.org/10.3389/fpls.2023.1249989>.
- [12] F. Zhang *et al.*, "Hyperspectral imaging combined with CNN for maize variety identification," *Frontiers in Plant Science*, vol. 14, Sep. 2023, <https://doi.org/10.3389/fpls.2023.1254548>.
- [13] W. Lin *et al.*, "Image segmentation method for physically touching soybean seeds," *Software Impacts*, vol. 18, p. 100591, Oct. 2023, <https://doi.org/10.1016/j.simpa.2023.100591>.
- [14] A. Ghimire *et al.*, "Automatic evaluation of soybean seed traits using RGB image data and a Python algorithm," *Plants*, vol. 12, no. 17, p. 3078, Aug. 2023, <https://doi.org/10.3390/plants12173078>.
- [15] Z. Wang, S. Fan, T. An, C. Zhang, L. Chen, and W. Huang, "Detection of Insect-Damaged maize seed using hyperspectral imaging and hybrid 1D-CNN-BILSTM model," *Infrared Physics & Technology*, vol. 137, p. 105208, Feb. 2024, <https://doi.org/10.1016/j.infrared.2024.105208>.
- [16] A. Sable, P. Singh, A. Kaur, M. Driss, and W. Boulila, "Quantifying Soybean Defects: A computational approach to seed classification using deep learning techniques," *Agronomy*, vol. 14, no. 6, p. 1098, May 2024, <https://doi.org/10.3390/agronomy14061098>.
- [17] A. Kirillov *et al.*, "Segment anything," *arXiv (Cornell University)*, Jan. 2023, <https://doi.org/10.48550/arXiv.2304.02643>.
- [18] H. Liu *et al.*, "Application of an improved watershed algorithm based on distance map reconstruction in bean image segmentation," *Heliyon*, vol. 9, no. 4, p. e15097, Apr. 2023, <https://doi.org/10.1016/j.heliyon.2023.e15097>.
- [19] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," *arXiv (Cornell University)*, Jan. 2021, <https://doi.org/10.48550/arXiv.2105.05233>.
- [20] R. Montoya-Del-Angel, K. Sam-Millan, J. C. Vilanova, and R. Martí, "MAM-E: Mammographic Synthetic Image Generation with Diffusion Models," *Sensors*, vol. 24, no. 7, p. 2076, Mar. 2024, <https://doi.org/10.3390/s24072076>.
- [21] Z. Zhang, Q. Gao, L. Liu, and Y. He, "A High-Quality Rice Leaf Disease Image data augmentation method based on a dual GAN," *IEEE Access*, vol. 11, pp. 21176–21191, Jan. 2023, <https://doi.org/10.1109/ACCESS.2023.3251098>.

- 
- [22] H. Pinckaers, B. Van Ginneken, and G. Litjens, "Streaming convolutional neural networks for End-to-End learning with Multi-Megapixel images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1581–1590, Aug. 2020, <https://doi.org/10.1109/TPAMI.2020.3019563>.
- [23] M. Tan and Q. V. Le, "EfficientNetV2: Smaller models and faster training," *arXiv (Cornell University)*, Jan. 2021, <https://doi.org/10.48550/arXiv.2104.00298>.
- [24] D. Zhu and D. Wang, "Transformers and their application to medical image processing: A review," *Journal of Radiation Research and Applied Sciences*, vol. 16, no. 4, p. 100680, Oct. 2023, <https://doi.org/10.1016/j.jrras.2023.100680>.
- [25] S. Agac and O. D. Incel, "On the Use of a Convolutional Block Attention Module in Deep Learning-Based Human Activity Recognition with Motion Sensors," *Diagnostics*, vol. 13, no. 11, p. 1861, May 2023, <https://doi.org/10.3390/diagnostics13111861>.
- [26] D. J. Hemanth, J. Anitha, A. Naaji, O. Geman, D. E. Popescu, and L. H. Son, "A modified deep convolutional neural network for abnormal brain image classification," *IEEE Access*, vol. 7, pp. 4275–4283, Dec. 2018, <https://doi.org/10.1109/ACCESS.2018.2885639>.
- [27] A. Srivastava, B. S. Rawat, G. Singh, V. Bhatnagar, P. K. Saini and S. A. Dhondiyal, "A Review of Optimization Algorithms for Training Neural Networks," *2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET)*, Ghaziabad, India, 2023, pp. 886-890, <https://doi.org/10.1109/ICSEIET58677.2023.10303287>.
- [28] S. Simon, N. Kolyada, C. Akiki, M. Potthast, B. Stein and N. Siegmund, "Exploring Hyperparameter Usage and Tuning in Machine Learning Research," *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, Melbourne, Australia, 2023, pp. 68-79, <https://doi.org/10.1109/CAIN58948.2023.00016>.
- [29] Keras tuner. [https://keras.io/keras\\_tuner/](https://keras.io/keras_tuner/). Accessed in October 2024.
- [30] D. Dibyanshu and R. K. S. Rajput, "Exploring agricultural image data through deep learning for visual information analysis: A study on Cabbage (*Brassica Oleracea* var. *Capitata*) farming," *Journal of Statistics and Management Systems*, vol. 27, no. 2, pp. 501–513, Jan. 2024, <https://doi.org/10.47974/JSMS-1291>.