scientific reports

OPEN

MythicVision: a deep learning powered mobile application for understanding Indian mythological deities using weight centric decision approach

Tauseef Khan✉, Aditya Nitin Patil, Aviral Singh, Gitesh Prashant Bhavsar, Kanakagiri Sujay Ashrith & Sachi Nandan Mohanty

Indian mythology is a treasure trove of divine tales, yet a gap in understanding still exists between foreign tourists and the rich cultural heritage of Indian deities. To address the problem, this paper presents a deep learning-driven mobile application named "*MythicVision*" designed to help foreign tourists better understand India's rich cultural heritage by recognizing and interpreting images of Indian mythological deities. At first, four state-of-the-art deep models have been trained and evaluated on a custom in-house dataset consists of 10,970 images of various Indian deities sourced from both natural scene and web images. Then, model-wise weights have been assigned by estimating the test accuracies obtained from test sets. In weight-centric decision mechanism, buckets of all possible classes of image object are updated by aggregating the corresponding model-weights if, the predicted class of the specific model matches any of the possible class. Finally, any possible output class with highest aggregated value is selected as final class of the image object. The whole framework is seamlessly integrated in an end-to-end web application for the ease of user convenience. Key features of "*MythicVision*" include model-wise weight computation and a weight-centric decision mechanism, which deliver more accurate results compared to traditional majority voting in multi-class image classification. The experimental findings demonstrate that developed framework produces an accuracy of 96% on in-house dataset. The designed *MythicVision* aims to recognize and classify real-time Indian deity images along with providing valuable information to the users about the deity. The developed web application along with source code and user guidelines have been publicly released in https://github.com/Adinp1213/MythicVision for academic, research and other non-commercial purposes.

Keywords Image classification, Deep learning, Mythology images, Weight-centric decision mechanism, MythicVision

Indian mythology is a collection of ancient stories, intricate symbols, and divine tales. It has deeply influenced India's culture and spirituality, evident in its sculptures, artworks, and artifacts. However, this rich heritage has not seamlessly blended with modern technology. While the potential of image classification in deep learning has been harnessed in various domains, its application to mythology has remained conspicuously limited. Image classification involves training algorithms to recognize and categorize objects or patterns within digital images. Such methods have reshaped numerous industries, from healthcare to autonomous vehicles[1], by enabling machines to interpret visual data[2] with remarkable performance. In general, there is a notable lack of research connecting mythological concepts and deep learning. Huang et al.[3] have developed a dataset of Chinese deities and classify images using state-of-the-art deep models marking a valuable advancement in this domain. While there have been some advancements in the field of mythology, however, within Indian culture, there is a noticeable gap in state-of-the-art works leveraging deep learning to promote and classify images related to Indian mythology. This paper has been a significant source of motivation, inspiring to create a classification system that

School of Computer Science and Engineering (SCOPE), VIT-AP University, Amaravati, Andhra Pradesh 522237, India. ✉email: tauseef.khan@vitap.ac.in

not only addresses this gap but also aims to achieve high performance, potentially surpassing contemporary methods.

However, some potential challenges need to be addressed in this research domain before plunging into the development of a robust and user convenient deep framework, especially in real-life applications.

- *Limited availability of mythological dataset* Limited availability of mythological datasets specifically focused on Indian deities poses a challenge for related research and applications.
- *Complexity of deity images* Hinduism mythology includes deities with similar facial features and portrayals, making it difficult to be differentiated. Also, the Indian deity images contain a lot of rich information making it a complex task to extract key features from such images.
- *Hindrances of traditional voting mechanisms in multi-class problem* In a multi-class classification framework, most traditional voting mechanisms struggle to resolve tie situations effectively, leading to random and unconvincing results[4].
- *Lack of end-to-end image classifier with description* Most applications provide only the classification results without offering additional descriptions or detailed information about the classification outcomes.

These intricate issues form the core of the current work's exploration and innovation. In this paper, a deep learning driven mobile application framework, named "*MythicVision*" has been developed and reported to address the challenges associated with Indian mythologies. The primary aim of the developed application is to combine Indian mythology with modern technology to enhance cultural tourism[5]. Therefore, a sequence of works has been implemented to address these issues. At first, a new dataset has been curated comprising of images of different Indian deities from diverse sources. The developed framework has employed popular deep models to achieve state-of-the-art performance[6]. In order to bolster the overall classification accuracy, a novel weighted-aware based decision mechanism has been designed using individual performance of deep models. Thus, the key contributions of the developed framework are discussed below in a nutshell.

- A new dataset has been curated and augmented comprising of various Indian deities from diverse cultural sources and scene images.
- The developed image classification framework has incorporated four cutting-edge deep learning models viz. MobileNet, ResNet, EfficientNet, and GoogleNet, chosen for their relevance to the problem objectives.
- A novel weight-centric decision mechanism assigns performance-based weights to each model. The final prediction is obtained by calculating a weighted sum of the individual model predictions.
- A user-friendly mobile application software is crafted by seamlessly integrating the developed framework. This intuitive application empowers users to access detailed descriptions of input deity images with ease.
- The source code for the developed model along with curated dataset are publicly released on GitHub[7] for academic, research and other non-commercial purposes.

The rest of the paper is organized as follows: "Related work" section reports a comprehensive review of related state-of-the-art methods with identified pros and cons. In "Proposed framework" section, the overall framework has been discussed in brief which includes preparation of the newly developed dataset, layer-wise architecture of employed deep models, and working principle of novel weight-centric mechanism. A series of rigorous experiments have been conducted and obtained results have been reported in "Experimental findings and analysis" section along with insightful analysis. Finally, "Conclusion" section includes the conclusive remarks and potential future scopes.

Related work

In the field of image classification, initially traditional methods have been extensively applied to classify objects in images. These methods typically involve hand-crafted features and trained classifiers. One of the popular feature-driven methods i.e., Connected Component (CC)-based methods first segregate image objects, then design intrinsic feature descriptors from segmented objects, and finally classify them using pattern classifiers[8,9]. Nevertheless, it has been observed that, majority of CC-based methods require extensive pre-processing and leading to compromised performance in unconstrained environments. Among different CC-based methods, texture-based feature descriptors classify image objects by analyzing their textural patterns[10,11]. Region-based methods classify foreground object components based on their geometric properties[12,13]. Maximally Stable Extremal Regions (MSER), a widely used region detector, identifies extremal regions within an image that exhibit maximum stability across a wide range of threshold values. Neumann et al.[14] have applied MSER to extract stable regions, identify characters, and group them to form text lines. Biswas et al.[15] have introduced a novel Deep Fuzzy based MSER model for the classification of diverse document images, encompassing handwritten, printed, and scene text. The model employs a unique combination of fuzzy logic and MSER (Maximally Stable Extremal Regions) to identify candidate components representing dominant information across various types of document images. While region-based methods are effective, they often require substantial pre-processing and post-processing, leading to reduced efficiency. It is worth mentioning that conventional image classification methods like SVM's have certain limitations as discussed by Wang et al.[16] have performed a comparative analysis between traditional machine learning algorithms like SVM and deep learning models CNN on the MNIST dataset and shows its edge over such traditional algorithms. Kovalev et al.[17] have found that deep learning models had higher accuracies than the traditional classification methods when they trained and compared their classification accuracies on chest radiographic images. Traditional methods like SVM often demand extensive feature engineering and may not perform well in complex and dynamic environments and rely on manual parameter tuning. Our developed framework aims to overcome these limitations by embracing deep learning techniques.

In recent years, deep learning has simplified image classification by automating the learning of hierarchical representations from raw data, eliminating the need for extensive manual feature engineering and intricate parameter tuning. This section delves into the use of deep neural networks in image classification and their benefits compared to traditional methods. Deep learning methods excel in automatically extracting intricate image features, adapting to complex situations, and achieving cutting-edge results. Among these, Convolutional Neural Networks (CNNs) stand out as pivotal[18–20]. CNN learns complex hierarchical features, making them highly adept at recognizing patterns and objects in images. Overall, deep learning enhances image classification by automating feature extraction, reducing the need for extensive pre-processing, and handling complex scenarios more efficiently. Affonso et al.[21] have implemented different CNN models and other machine learning techniques for image classification task. Mikolajczyk et al.[22] have applied different data augmentation techniques and showed their importance in image classification tasks carried out using GANs (Generative adversarial networks). Whereas Obaid et al.[23] have implemented different deep learning models for image classification tasks and compared its performance on the CIFAR-10 and CIFAR-100 datasets. Hosseini et al.[24] have discussed how CNN's work poorly on negative images due to its poor shape bias property by conducting experiments on MNIST and CIFAR-10 datasets. Meena et al.[25] have trained an InceptionV3 model on a monkeypox dataset to determine whether a patient is affected by monkeypox. Their approach achieved an impressive accuracy of 98%. Meena et al.[26] have also contributed to categorizing various brain MRI images to detect potential tumors using custom CNNs. Meena et al.[27] also trained CNNs to identify emotions from facial expressions achieving 79% and 95% accuracy on FER-2013 and CK datasets respectively. Meena et al.[28–30] have also contributed to the domain of sentiment analysis using transfer learning, to demonstrate how pre-trained models like VGG-19 and Inceptionv3 can be fine-tuned on domain-specific data to improve accuracy and efficiency. Similarly, this approach can be applied to cultural deity recognition, where pre-trained models can be adapted to classify images associated with diverse deities. Huang et al.[3] have shared similar goals but is tailored specifically to Chinese mythology. It stands as a testament to the potential impact of bridging deep learning with cultural exploration. Table 1 highlights some of the recent state-of-the-art methods in this domain and are represented in a comparative fashion.

In the light of the above discussion some shortcomings have been observed in contemporary methods. To address these shortcomings the developed framework has been designed to address each aspect comprehensively. Our developed framework aims to create custom datasets, choose appropriate deep learning models, and make a well-informed decision regarding accuracy prioritization. By doing so, we envision our goal of enhancing cultural tourism by revealing the hidden stories behind India's art and sculptures.

Proposed framework

This section presents a succinct description of step-wise development of the framework. The four steps of the developed framework are as follows: (i) The development framework commences with the acquisition, selection, and curation of images featuring Indian deities for the creation of the dataset. Subsequently, the developed dataset is partitioned into training and testing sets. (ii) In the next step, four state-of-the art deep models viz. MobileNet, ResNet, EfficientNet, and GoogleNet have been trained using train-set and evaluated on test-set to obtain the model-wise classification accuracy. (iii) The classification accuracies obtained from different models serve as the foundation for a novel weight-centric decision mechanism. The weight-centric approach prioritizes over individual model accuracy, aiming to improve the overall accuracy and reliability of the framework (iv) Finally, the operational modules of (ii) and (iii) have been seamlessly integrated into the *MythicVision* mobile application. Designed application empowers users to acquire concise descriptions of Indian deities from real-time input images using their fingertip. The step-wise development of the framework is depicted in Fig. 1. Moreover, some salient features of the designed applications have been highlighted in this section.

Dataset preparation

The preparation of the dataset for reported work assumed a crucial role given the limitation of existing datasets in Indian mythology. This section provides an in-depth overview of the dataset creation process, including data collection, labeling, and preprocessing steps undertaken to ensure high-quality, representative data for training and evaluation.

References	Key features
Sherly et al.[31]	• Utilized object detection techniques like YOLOv5 and ensemble models • Dataset consisted of Hindu and Christian divine images
Sasithradevi et al.[32]	• Proposed a custom KolamNetv2 architecture comprising of EfficientNet and various attention modules • It aimed to classify various artistic styles of rangoli commonly found in Tamil culture
Sasithradevi et al.[33]	• Proposed MonuNet architecture for classification of cultural heritage in Kolkata • Utilized dense channel attention modules and PSCA modules to extract key architectural details from the images
Babić et al.[34]	• Compares four machine learning algorithms using features from eleven pre-trained deep learning models on a small cultural heritage image dataset • Used DenseNet121, EfficientNetB0 and NASNetMobile architectures for feature extraction
Gao et al.[35]	• Proposed Convolutional Neural Network Attention Retrieval Framework (CNNAR Framework) to classify various Chinese diaspora architectural styles • Tested the proposed architecture on Paris500K and Corel5K datasets to achieve considerable results

Table 1. A brief outline of state-of-the-art methods for mythological and cultural image classification.

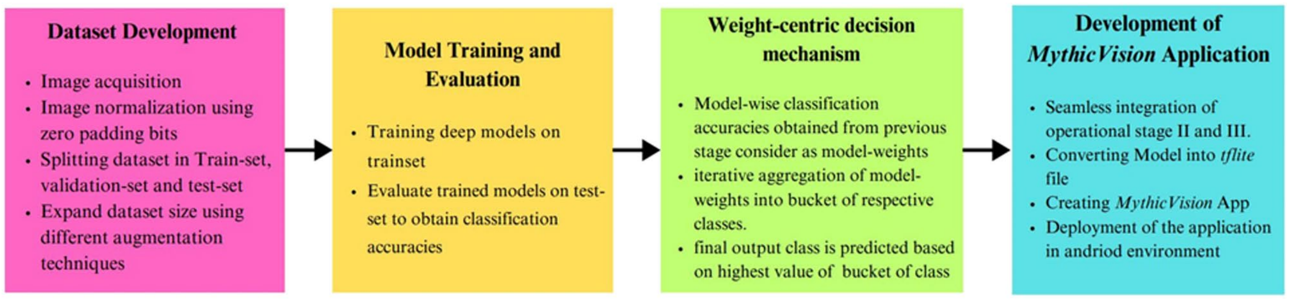

Dataset Development
Image acquisition
Image normalization using zero padding bits
Splitting dataset in Train-set, validation-set and test-set
Expand dataset size using different augmentation techniques
Model Training and Evaluation
Training deep models on trainset
Evaluate trained models on test-set to obtain classification accuracies
Weight-centric decision mechanism
Model-wise classification accuracies obtained from previous stage consider as model-weights
iterative aggregation of model-weights into bucket of respective classes.
final output class is predicted based on highest value of bucket of class
Development of *MythicVision* Application
Seamless integration of operational stage II and III.
Converting Model into *tflite* file
Creating *MythicVision* App
Deployment of the application in andriod environment

Fig. 1. A visual representation illustrates the various stages of the developed application framework. It starts with the creation of the initial dataset, followed by the training and evaluation of deep models using the newly developed dataset. Then, implementation of a weight-centric decision mechanism that bolsters the classification accuracy. Finally, the mobile application is developed, enabling users to obtain detailed descriptions from real-time images of Indian deities.

Class	Scene image	Web source	Total
Training	500	1497	1997
Validation	201	599	800
Testing	120	80	200
Total	821	2176	2997

Table 2. Overview of images collected from scene and web sources.

Dataset acquisition and image normalization
With a set of ten distinct deities, 3000 images (300 per class) are manually collected from various sources viz. scene images (book covers, posters, idols) and web source[36,37]. Attention has been given to ensure that the collected images represent scene images, capturing a range of unique contexts, and are free from duplicates. An overview of the dataset size collected from various sources is provided in Table 2. This approach enhances the dataset's diversity and supports more robust model training. Then these images of different resolutions were all resized to 224 × 224 pixels so that they all are of same resolution thus suitable to be fed into the models. These images were also zero padded to prevent losing their original aspect ratio. The dataset was then split into train, validation, and test set with a ratio of 70:20:10 respectively.

Dataset augmentation
A set of augmentation techniques has been applied to the images to increase the amount of training data to avoid overfitting so that the model can generalize better to unseen data. These augmentation steps include:

- Rotation—Rotates images at different angles, helping the model recognize objects at various orientations.
- Shearing—Skews the image along one axis, making the model better at detecting objects that might appear stretched or tilted in certain perspectives.
- Translation—Moves the image slightly in different directions (up, down, left, or right), training the model to recognize objects even if they are not perfectly centered.
- Flipping—Mirrors the image horizontally or vertically, helping the model generalize to mirrored versions of objects, which is useful for detecting symmetrically structured features.

Within the dataset, 800 images were assigned for validation, while 200 images were designated for testing purposes while the remaining 2000 images (200 per class) underwent a transformative augmentation process. This process included a 20-degree rotation, 180-degree rotation, flipping, sharing and translation effectively generating 5 new images as shown in Fig. 2. Consequently, around 10,000 images were allocated for training purposes for the subsequent image classification task as shown in Table 3. Thus, the in-house dataset encompasses a total of 10,970 images.

Complexity of the dataset
The self-curated dataset presents numerous challenges in its preparation, including significant variability in features, posture, and environmental factors, which require extensive preprocessing and careful dataset curation. These challenges include:

- *Complex background* Some scene deity images often feature intricate, high-contrast backgrounds as seen in Fig. 3a that can negatively impact the framework's performance, highlighting the need for efficient solutions to address this challenge.

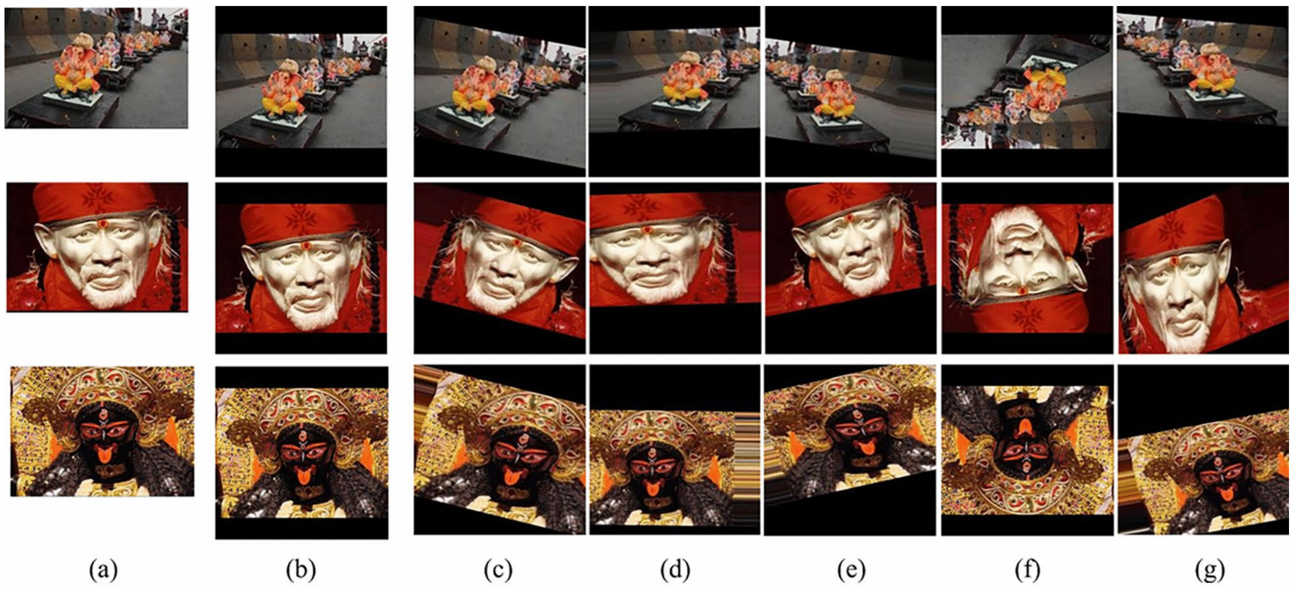


Fig. 2. Development of the dataset and extension of dataset size using different augmentation techniques. (**a**) Original source images, (**b**) corresponding normalized images of 224 × 224 pixels dimensions using zero padding bits, and (**c**–**g**) corresponding replicated image by applying different augmentation techniques like 20-degree rotation, shearing, translation, 180-degree rotation, and flipping respectively.

Class	Train set (before augment.)	Train set (after augment.)	Validation set	Test set
Balaji	199	995	80	20
Durga Maa	200	1000	80	20
Ganesha	200	1000	80	20
Hanuman	199	995	80	20
Kali Maa	200	1000	80	20
Khatu Shyam	200	990	80	20
Krishna	200	1000	80	20
Sai Baba	200	995	80	20
Saraswati	200	1000	80	20
Shiva	199	995	80	20
Total	1997	9970	800	200

Table 3. Brief outline of the experimental dataset along with its particulars (before and after augmentation).

- *Low-resolution and distorted images* Images may be blurred or distorted, especially in cases where an oil lamp is placed in front of the idol, leading to reduced image clarity and potential loss of detail as seen in Fig. 3b.
- *Occluded/noisy images* The presence of garlands and accessories on idols, objects hiding deities, and unnecessary background capture can introduce unnecessary noise as seen in Fig. 3c, potentially affecting the framework's ability to accurately recognize and classify the deity.

These complexities have been carefully considered and addressed during the dataset preparation and preprocessing stages to ensure optimal model performance.

Employed deep models

This section provides a detail insight on employed deep learning models viz. MobileNet, Resnet, EfficientNet, and GoogleNet with their layer-wise architecture and working principle.

Strategy behind model selection

To address the unique challenges of classifying Indian deities, selecting the right models is crucial. Each of the following models are chosen for its specific strengths in image classification, ensuring accuracy, efficiency, and adaptability to various devices. A brief outline of the employed models and their favourable scenarios are mentioned in Table 4.

Each of these models are further explained in detail in the following sections, providing a comprehensive understanding of their unique contributions to the classification of Indian deities.

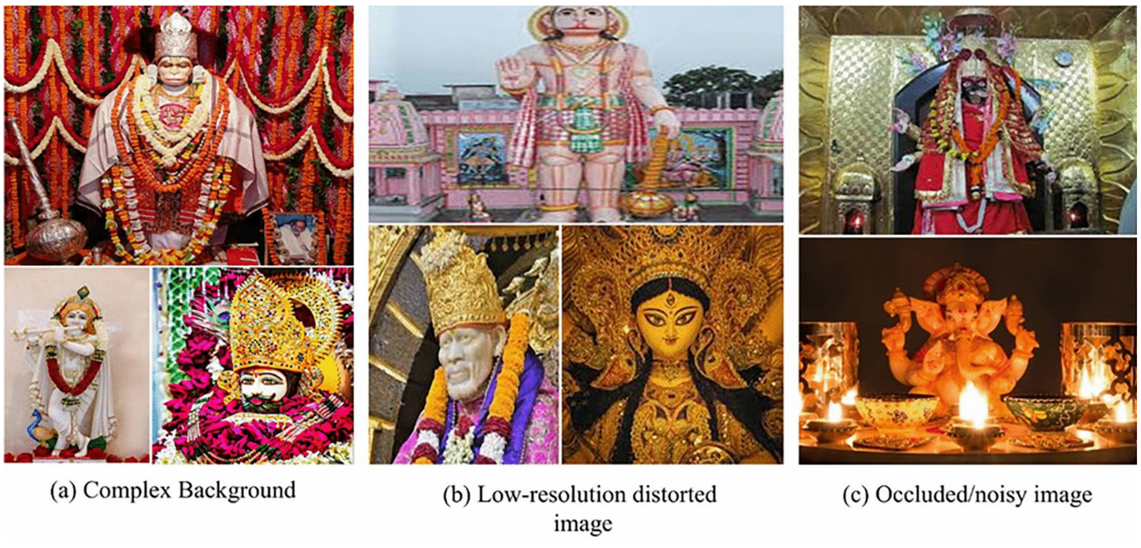


Fig. 3. Some sample images demonstrate the complexity and diversity of the dataset. (**a**) Images with complex background, (**b**) low-resolution and distorted images, (**c**) occluded/noisy images.

Models	Key features	Suitable scenario
MobileNetv2	Lightweight and efficient	Ideal for mobile applications with limited computational resources
ResNet-50	Skip connections effectively address the vanishing gradient problem	Suitable for handling deeper networks and complex image tasks
EfficientNetB0	Scalable design optimizes accuracy	Best for scenarios where computational efficiency and accuracy are priorities
Inceptionv3	Captures spatial hierarchies efficiently in images	Suitable for tasks requiring detailed image analysis with complex patterns

Table 4. Brief outline of the employed models and their key features and suitable use case scenarios.

Architecture and working principle of employed deep models

MobileNetV2 MobileNetV2[38,39] is designed to be lightweight and efficient, making it suitable for various real-time applications on resource-constrained devices. The architecture comprises of Convolution2D, bottleneck and average-pooling layers. The Bottleneck layer consists of 1 × 1 expansion layer, Normalization layer, 3 × 3 Depth-wise convolutions, 1 × 1 projection layers and activation layer consisting of ReLU. The bottleneck layer's function is to add computation modules and force the network to learn more compact data in fewer layers thus saving computational time. Figure 4a[40] illustrates the layer wise architectural view of the MobileNetv2 framework.

EfficientNetB0 The main architecture of EfficientNetB0[41] consists of stacked blocks, each having MobileNetV2-like inverted residual structures (MBConv) as seen in Fig. 4b[42]. The architecture uses a unique compound scaling method, scaling the network's depth, width, and resolution uniformly. The model adjusts depth, width, and resolution using scaling factors (α, β, γ). This optimizes the model for different computational resources. Balancing accuracy and computational cost, EfficientNetB0 is a versatile and scalable choice, suitable for various applications from mobile devices to resource-constrained environments.

ResNet-50 ResNet-50[43] is a deep neural network architecture comprises of 48 convolutional layers, one average pooling layer, and one fully connected layer. It utilizes 3-layered bottleneck blocks to facilitate feature extraction and model training. The resnet-50 architecture is depicted in Fig. 4c[44]. ResNet-50 is a computationally intensive model, with approximately 3.8 billion Floating Point Operations (FLOPs). This high level of computational complexity is one of the reasons behind its remarkable performance in various computer vision tasks, particularly image classification.

GoogleNet (Inception V3) Inception V3[45] model comprises a total of 42 layers. It comprises an initial stem block for feature extraction, followed by a series of diverse Inception modules and Grid Size Reductions. These modules leverage parallel paths with different kernel sizes, including 1 × 1, 3 × 3, and 5 × 5 convolutions, as well as pooling operations, to capture multi-scale features efficiently. Grid Size Reduction blocks are employed to decrease spatial dimensions, and auxiliary classifiers aid in mitigating the vanishing gradient problem during training. The InceptionV3 auxiliary classifier enhances training by offering an extra path for gradient flow, serving as regularization, and promoting the learning of valuable features in intermediate layers to enhance model performance. The architecture is finalized with global average pooling, fully connected layers, and a SoftMax output layer for classification. The architecture of the network is illustrated in Fig. 4d[46].

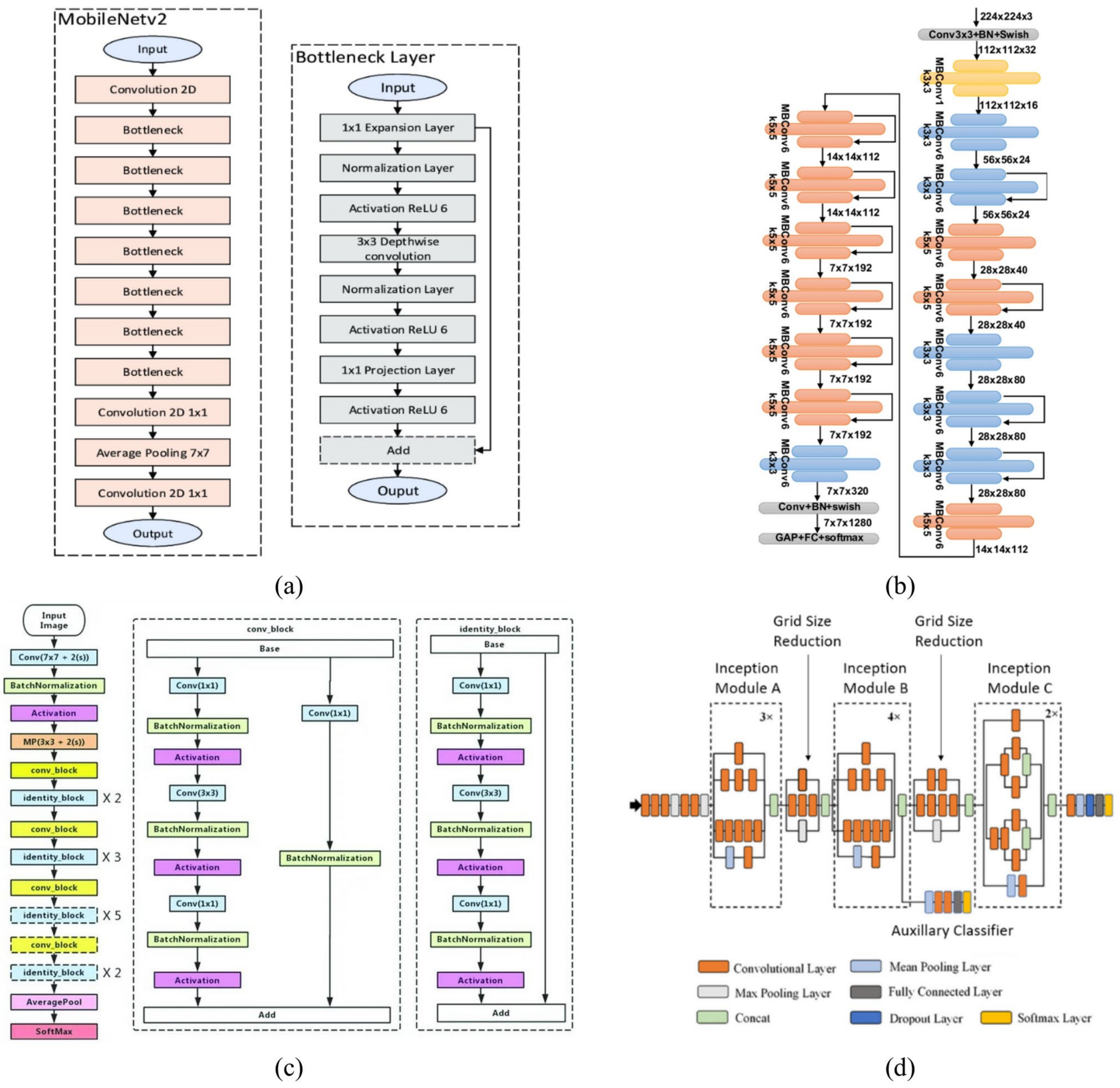


Fig. 4. Layer-wise architecture of employed deep models. (**a**) MobileNetv2, (**b**) EfficientNetB0, (**c**) ResNet-50, and (**d**) Inceptionv3 respectively.

Weight-centric decision mechanism

The Weight-Centric Decision Mechanism is an approach that leverages model-specific test accuracy scores to assign weighted importance to each model in a multi-model framework. By factoring in these accuracy-derived weights, the mechanism aims to increase the likelihood of correct classifications across the framework by favoring models with higher accuracy scores. The selection criteria for the weights, working principle of the mechanism and its significance are reported in the following sub-sections.

Weight computation

Four individual models have been trained on a train dataset and tested on a test dataset. After the models are tested individual test accuracy scores are collected which acts as the weights for the weight-centric mechanism. The model with higher test accuracy has a higher likelihood of classifying the image correctly than the model with less test accuracy. Thus, the model with higher test accuracy gets a higher weightage than models with less test accuracy. The weight for each model w_i is calculated as follows:

$$w_i = multiplier\,(m) \times test\_accuracy_i \tag{1}$$

where $test\_accuracy_i$ represents the test accuracy score of the ith model, and the multiplier m is a scaling factor to adjust the influence of each accuracy score as seen in Eq. 1. This approach ensures that models with higher test accuracy are assigned greater weight, providing a double dependency on both the training and test sets for improved classification reliability.

Weight-centric decision mechanism

In the Weight- centric mechanism, the framework employs a strategic approach to enhance classification accuracy. The working principle of weight-centric decision mechanism is illustrated in Fig. 5. Each step involved in the weight-centric mechanism is mentioned as follows:

(i) Prior to making predictions, four deep models have been evaluated on a carefully selected test set and model-wise test accuracy is obtained. These test accuracies subsequently act as weight values for that model denoted by w_1, w_2, …, w_n, where n is the number of deep models used in the framework (viz. n=4). Moreover, probable classes of an input image are denoted as $\mathcal{C}_1, \mathcal{C}_{2,\ldots,}\mathcal{C}_m$, where m is the number of probable classes of an input image object (viz. $m = 10$).

(ii) Let, class prediction of models denoted as p_1, p_2, …, p_n where n is the number of predictions (which is same as the number of models used).

(iii) Each probable class of an input image object reserves its class-bucket. If the model predicted class as discussed in (ii) matches the probable class, the weight of that model as mentioned in (i) is iteratively added to the corresponding class-bucket. This process is repeated for all the remaining deep models. Finally, the class-bucket of a probable class with highest aggregated value selected as final output class denoted as $\mathbb{Z}$. The weight-centric decision mechanism is illustrated in Eqs. 2–4.

(iv) For instance, in the current study the number of decision classes is ten viz. Balaji, Durga Maa, Ganesha, Hanuman, Kali Maa, Khatu Shyam, Krishna, Sai Baba, Saraswati, and Shiva. Now, if a particular model predicts an image as Ganesha, the weight of that model is put into the bucket of Ganesha. Thus, at the end, the bucket with the maximum aggregated score provides the final output class of the image object.

(v) The idea is to give a higher influence on models with higher test accuracies, assuming they are more reliable or accurate. This approach enhances the reliability and precision of the desired result, further improving the framework's performance.

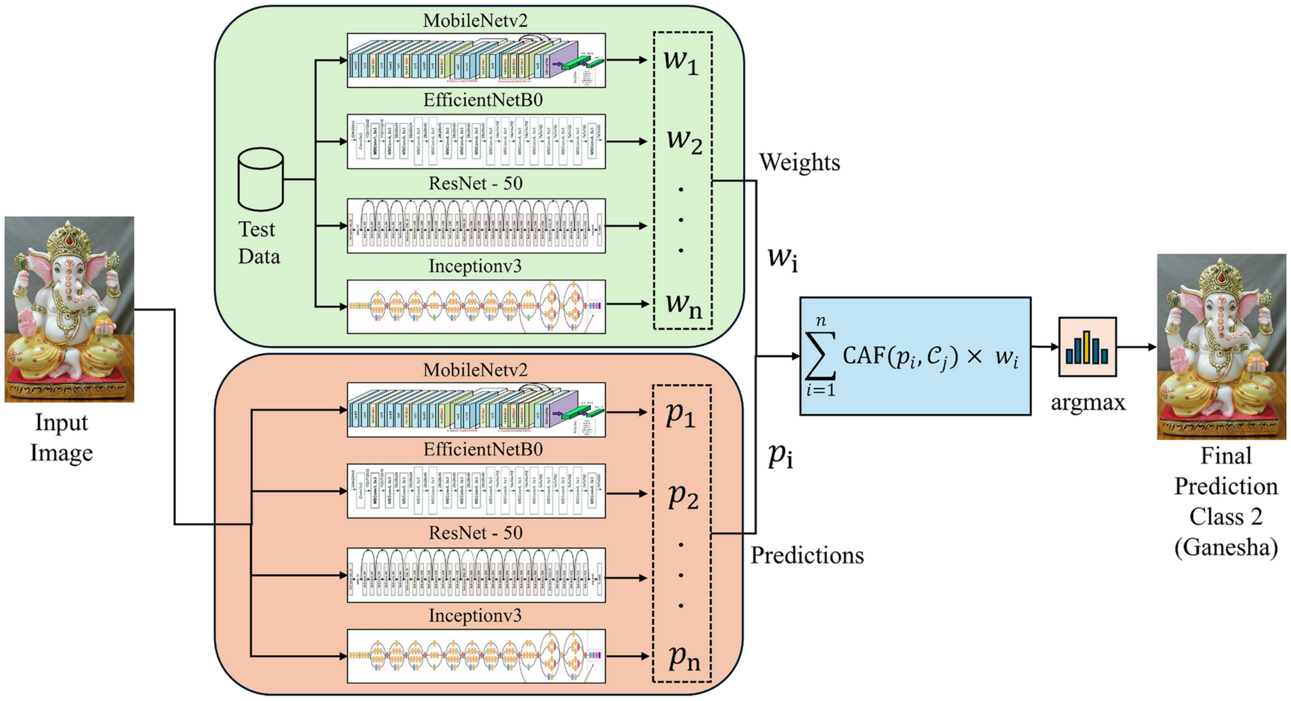


Fig. 5. The weight-centric decision mechanism operates as follows: deep models are initially trained and evaluated on test data, producing classification accuracies. These accuracies serve as weights for each model in the decision mechanism. In real-time, an input image is introduced to the system, and each trained model generates predictions. After model-wise predictions, model-weights are iteratively added into the corresponding buckets of respective predicted classes, finally the prediction of final output class of an image is obtained based on aggregated values accumulated in the bucket of the decision classes.

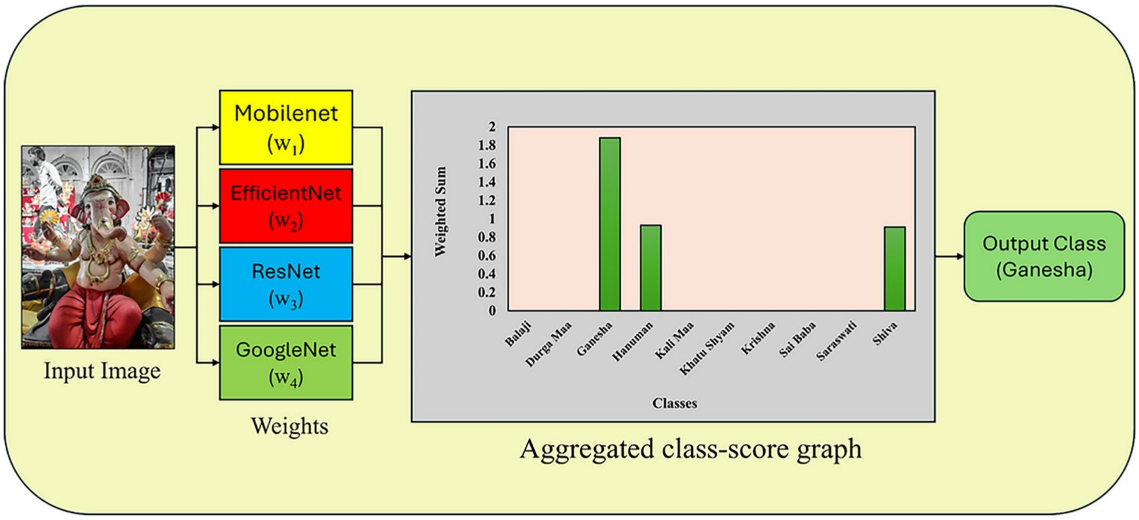


Fig. 6. Pictorial representation of final prediction generated using weight-centric decision mechanism.

Scenario	Conventional majority-based voting	Weight-centric decision mechanism
Case 1 Two classes achieve an equal majority vote, resulting in a tie situation	May make random assumptions in tie situations, leading to potential inaccuracies	If models tie, the final prediction is based on the highest weighted value. Example: EfficientNet and MobileNet predict Ganesha (1.88), Resnet and GoogleNet predict Hanuman (1.84), final prediction: Ganesha
Case 2 All four models provide distinct output, resulting in a four-way tie	May make random assumptions when models predict distinct classes, lacking a systematic resolution	If all models predict distinct classes, the final decision is based on the highest weighted value. Example: EfficientNet (Ganesha), MobileNet (Hanuman), Resnet (Khatu Shyam), GoogleNet (Sai Baba), final prediction: Ganesha
Case 3 The models have different levels of training or are not adequately trained	Varied training levels may hinder majority voting, leading to potential inaccuracies	Ensures consistent predictions despite varied training. Predictions are based on weighted values, so low accuracies do not affect the final decision. Example: EfficientNet (95% accuracy) (predicts Ganesha), MobileNet (20%), Resnet (30%), GoogleNet (10%), final prediction: Ganesha

Table 5. Scenario based comparison between conventional majority-based voting technique and developed weight-centric decision mechanism.

To illustrate this with an example, let us assume: MobileNet predicts Hanuman (Class 3), EfficientNet and ResNet predicts Ganesha (Class 2), and Inceptionv3 predicts Shiva (Class 9). Weight values are then assigned to each class based on their respective model accuracies. A weighted sum is computed of each prediction class. The final prediction becomes the class having maximum weighted sum as shown in Fig. 6.

$$\mathbb{Z} = argmax_m \mathcal{O}\left(\mathcal{C}_j\right) \tag{2}$$

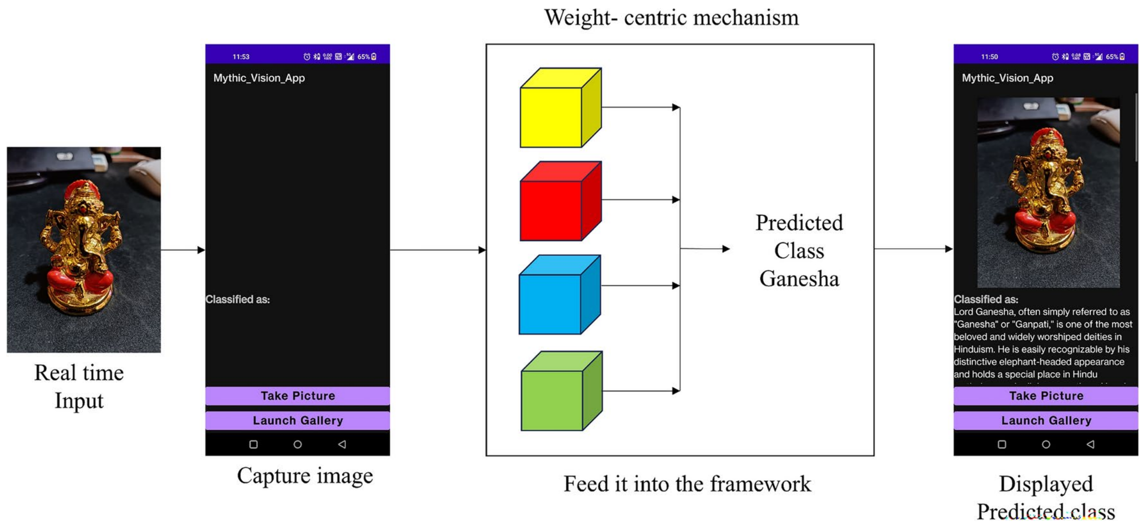
Here, $\mathcal{O}(\mathcal{C}_j)$ is denoted as aggregated class-score of jth probable class, where $1 \leq j \leq m$.

$$\mathcal{O}\left(\mathcal{C}_j\right) = \sum_{i=1}^{n} \text{CAF}\left(p_i, \mathcal{C}_j\right) \times w_i \tag{3}$$

Here, if prediction of ith model denoted as p_i matches with jth probable class (i.e., $\mathcal{C}_j$), then weights of the ith model i.e., w_i, will be aggregated to the class-bucket. The CAF is denoted as *class-affinity-factor*, which is mathematically shown in Eq. 4.

$$\text{CAF}\left(p_i, \mathcal{C}_j\right) = \begin{cases} 1, & if\ p_i == \mathcal{C}_j \\ 0, & else \end{cases} \tag{4}$$

Significance of weight-centric decision mechanism
The weight-centric mechanism stands out as a superior alternative to the conventional majority-based voting approach. In scenarios where the majority-based approach can lead to arbitrary assumptions and inaccurate classifications, the weight-centric decision mechanism excels. By assigning customized weights and giving priority to models showing better performance, it addresses the challenges of intricate data patterns and minority classes. This adaptive approach ensures accurate and informed decisions even in complex situations, ultimately enhancing classification accuracy and result reliability as shown in Table 5.



Lord Ganesha, often simply referred to as "Ganesha" or "Ganpati," is one of the most beloved and widely worshiped deities in Hinduism. He is easily recognizable by his distinctive elephant-headed appearance and holds a special place in Hindu

Fig. 7. Overall working principle of designed MythicVision mobile application for identification of deities from real-time image. The entire developed image classification framework is integrated to the designed application. Initially, an image is captured in real-time by designed MythicVision application integrated with the built-in mobile camera and classified using the developed framework. Subsequently, based on the classified image, a comprehensive description of the specific deity (output class) is furnished.

Package/library	Purpose	Example use case
Android studio	Integrated development environment for developing android applications	Developing the MythicVision mobile app
Java	Programming language for android app development	Writing the app's core logic, UI interactions, and API integrations
TensorFlow lite	Optimized framework for deploying ML models on mobile and embedded devices	Running deity recognition models efficiently on mobile devices
TensorFlow	Framework for building and training machine learning models	Training deep learning models for image classification
Pillow	Python library for image processing tasks	Preprocessing images, such as resizing or augmenting deity images
Pandas	Data manipulation and analysis	Organizing and cleaning the dataset for model training
Matplotlib	Visualization library for plotting data and insights	Visualizing training metrics like accuracy and loss curves

Table 6. Tools and libraries used in MythicVision.

Development of MythicVision application

The main objective of the current research is to develop a convenient and user-friendly image classification application for identification of real-time deity images and provide a detail description on that. Therefore, the whole developed framework discussed in "Dataset preparation", "Employed deep models", "Weight-centric decision mechanism" sections (i.e., dataset creation, model selection, and the weight-centric approach) has been seamlessly integrated into the designed mobile application named *MythicVision*. The operational flow of the developed MythicVision application is illustrated in Fig. 7. The application has been developed through the conversion of the entire framework into a TensorFlow Lite file with the help of TensorFlow library. Then, it is exported to Android Studio, a dynamic platform for application development, where a user-friendly application with a basic User Interface (UI) interface has been developed. All such libraries and frameworks used to develop MythicVision are listed in Table 6. Within this application, users encounter an interface featuring two essential buttons. The "Take Pictures" button activates the device's camera, enabling users to capture real-life images that align with their exploration of Indian mythology. The "Launch Gallery" button provides a convenient portal for users to access their device's image gallery and select a picture they may have captured previously. The user initiates the process by capturing an image through the application. This image is subsequently channeled into the integrated models, where the deep learning models process it, where the weight-centric mechanism is thoroughly applied. The developed application then presents the user with the predicted class corresponding to the captured image, along with comprehensive information related to the recognized deity as shown in Fig. 7.

Salient features of MythicVision

The *MythicVision* app is designed to enhance the experience of users by blending technology with cultural exploration. With a focus on accessibility, interactivity, and cultural preservation, it aims to provide an enriching and educational tool for anyone interested in learning about Indian mythology and deities.

- *Cross-platform Application* The developed application software is compatible with Android-based mobiles, PCs, tablets, and other devices.

- *Temple Location Data* As users scan deities through the app, an enhancement could include providing details about the nearest temples associated with the recognized deities.
- *Multilingual Support* Extend the software to provide information in various languages to cater for the diverse range of tourists.
- *Collaborative Platform* Create a community-driven platform where users can contribute. additional information and stories about detected deities.
- *Expansion to Other Cultures* the developed application may detect and classify deities from different cultures around the world.
- *Language Diversity* Expand the software to offer information in different languages, making it accessible to a wider range of tourists from around the world.
- *Interactive Quizzes* Integrate interactive quizzes and games within the software to make learning about Indian mythology even more engaging and fun.

User feedback and recommendations

Gathering user feedback is a critical step in ensuring the system's accessibility, usability, and overall effectiveness in the field of cultural exploration. Initial feedback from users highlighted the following key areas of improvement:

- *Incorrect upload errors* Users came across errors when uploading corrupt or incorrect format of image. This can be rectified by handling such errors and notifying the user to upload the correct format (jpeg, png, e.t.c) of image.
- *Easy navigation controls* Users appreciated the simple navigation controls of the application, particularly for uploading deity images with a single click of a button.
- *Incorporating a broader range of deities* Some users suggested adding more local deities to the MythicVision framework to enhance its inclusivity and cultural representation.

This feedback is crucial for assessing the robustness of our framework and will be integrated into future updates to enhance the system's accessibility and reliability for all users.

Cultural impact of MythicVision

MythicVision plays an important role in connecting people with the rich cultural heritage of India. By blending modern technology with ancient traditions, the app has the potential to make a positive impact on how we engage with and understand mythology. Some of the significant impact MythicVision have on culture of India is as follows:

- *Preserving cultural heritage* The *MythicVision* app preserves Indian mythology by making information about deities accessible, ensuring this knowledge is passed down to future generations.
- *Increasing cultural awareness* MythicVision helps foreign tourists to learn about Indian culture with a simple click of a button, promoting a better understanding of its mythology and traditions.
- *Promoting interfaith discussion* By recognizing deities from different cultures, the app promotes respect and understanding across religions, encouraging dialogue and shared learning.
- *Supporting local communities* The app provides information about nearby temples and cultural sites, which can boost interest and support for local heritage, benefiting the community.

In these ways, MythicVision not only educates users about various Indian deities but also helps preserve and share the cultural richness of India and the world.

Experimental findings and analysis

A set of experiments has been carried out on a newly developed dataset [7] to assess the performance of employed deep networks. These networks are trained and fine-tuned with a training set and evaluated on test set. The obtained model-wise classification accuracies are reported in this section. It is important to note that these classification accuracies serve as weights for each model in the weight-centric decision mechanism module. In this section, module-wise classification accuracies as well as the final classification accuracy obtained through the weight-centric decision mechanism has been reported. Subsequently, an insightful analysis has been presented based on the results obtained.

Experimental details

To quantify the performance of the employed deep learning models, standard evaluation metrics have been used viz. Recall, Precision, F-Measure, Classification Accuracy, and Error-rate. Precision $(P) = \frac{TP}{TP+FP}$ represents the number of positive class predictions that truly belong to the positive class. Recall $(R) = \frac{TP}{TP+FN}$ indicates the number of positive class predictions made from all positive samples in a dataset. F-Measure $(FM) = 2 \times \frac{P \times R}{P+R}$ is the harmonic mean of Precision and Recall. This metric is particularly useful when there is an uneven class distribution. Classification accuracy $(CA) = \frac{TN+TP}{TN+FP+TP+FN}$ defines the ratio of true-classified samples to the total number of samples in the test set. The Error-rate (ERR), computed as 1-CA, refers to the prediction error of the model concerning the actual class.

Table 7 provides the layer-wise diagram for each employed deep model for the *MythicVision* framework. During the training phase, Adam optimization algorithm has been applied to enhance results, particularly favoring its compatibility with lightweight networks due to its lower memory requirements[47]. The choice of 75 epochs is informed by empirical observations on the convergence of the learning graph. For addressing a

Layer type	MobileNetV2	EfficientNetB0	ResNet-50	Inception V3
Input layer	224 × 224 × 3 RGB image	224 × 224 × 3 RGB image	224 × 224 × 3 RGB image	299 × 299 × 3 RGB image
Base layers (non-trainable)	- Initial Conv2D (3 × 3, 32 filters, stride = 2) - Bottleneck blocks (MobileNetv2)	- Initial Conv2D (3 × 3, 32 filters, stride = 2) - MBConv blocks (EfficientNetB0)	- Initial Conv2D (7 × 7, 64 filters, stride = 2) - Residual blocks (ResNet-50)	- Initial Conv2D (3 × 3, 32 filters, stride = 2) - Inception blocks (Inceptionv3)
Global pooling	Global average pooling	Global average pooling	Global average pooling	Global average pooling
DenseLayer1	512 units, ReLU	1024 units, ReLU	512 units, ReLU	1024 units, ReLU
DropOut	Rate = 0.5	Rate = 0.5	Rate = 0.5	Rate = 0.5
DenseLayer2	256 units, ReLU	256 units, ReLU	256 units, ReLU	256 units, ReLU
DropOut	Rate = 0.5	Rate = 0.5	Rate = 0.5	Rate = 0.5
Output layer	Dense (10 units, softmax)	Dense (10 units, softmax)	Dense (10 units, softmax)	Dense (10 units, softmax)
Total parameters	Total: ~ 3.4 M Trainable: ~ 400 K Non-Trainable: ~ 3 M	Total: ~ 5.3 M Trainable: ~ 300 K Non-Trainable: ~ 5 M	Total: ~ 25 M Trainable: ~ 2 M Non-Trainable: ~ 23 M	Total: ~ 22 M Trainable: ~ 2 M Non-Trainable: ~ 20 M

Table 7. Layer wise architecture details of the employed four models along with total trainable and non-trainable parameters.

Training parameters	Values
Number of epochs	75
Optimizer	Adam
Learning rate	0.0001
Batch size	32
Loss	Categorical cross entropy
Callback_1	Early stopping (patience = 10)
Callback_2	ReduceLRonPlateau (factor = 0.2, patience = 5)
Callback_3	ModelCheckpoint (monitor = 'val_loss')

Table 8. Hyperparameter values set during training procedure.

multiclass problem, categorical cross-entropy serves as the selected loss function. Additionally, normalization of training images involves the utilization of the "same" padding technique, leveraging edge pixels to facilitate network inference with them. A list of all such training hyperparameters have been listed in Table 8. Reported experiments are conducted on a system powered by a 10th Gen Core i5 processor, complemented by an NVIDIA GTX 1650 GPU and 8 GB of memory. The MythicVision framework is implemented in Python framework, utilizing TensorFlow as the backend and running Keras on top of TensorFlow.

Experimental results

Initially, four deep networks have undergone a series of epochs for optimal training with the training dataset. Subsequently, the pretrained networks have been evaluated using the test dataset to determine classification accuracies. The train-test process of deep networks constitutes the foremost activity within operational module (ii), as illustrated in Fig. 1. The empirical findings encompass the graphical representation of epoch-wise training accuracy and loss trends during the training of the models as depicted in Fig. 8.

The confusion matrix provides a thorough analysis of errors, model refinement, and the selection of suitable evaluation metrics. Consequently, the confusion matrix generated from deep models and weight-centric decision mechanism are depicted in Fig. 9. Precision-Recall (PR) and Receiver Operating Characteristic (ROC) curves allow more nuanced evaluation of model performance, especially in situations involving imbalanced datasets, varying error costs, or a need to fine-tune classification thresholds. The PR curve illustrates the trade-off between precision and recall, whereas ROC curve showcases the model's true positive rate against false positive rate. Model-wise PR curve and ROC curve have been depicted in Fig. 10.

Table 9 presents a summary of the different performance metrics obtained from various deep neural networks and the final performance metrics achieved using the novel weight-centric decision mechanism on a newly developed dataset. MobileNetV2, EfficientNet B0, ResNet-50, and GoogleNet have demonstrated high classification accuracies ranging from 0.92 to 0.95. Among them, the highest classification accuracy has been achieved by EfficientNetB0, reaching 95%. However, it is worth mentioning that, after applying the weight-centric decision mechanism, the classification accuracy has reached 96%, resulting in a 1.05% increase in performance. Hence, it can be stated that proposed weight-centric decision mechanism surpasses individual model performance, underscoring its significance in enhancing overall classification accuracy and delivering compelling and reliable results. In addition, a graphical representation of the performance comparison in terms of classification accuracy between individual deep networks and the weight-centric decision mechanism is depicted in Fig. 11.

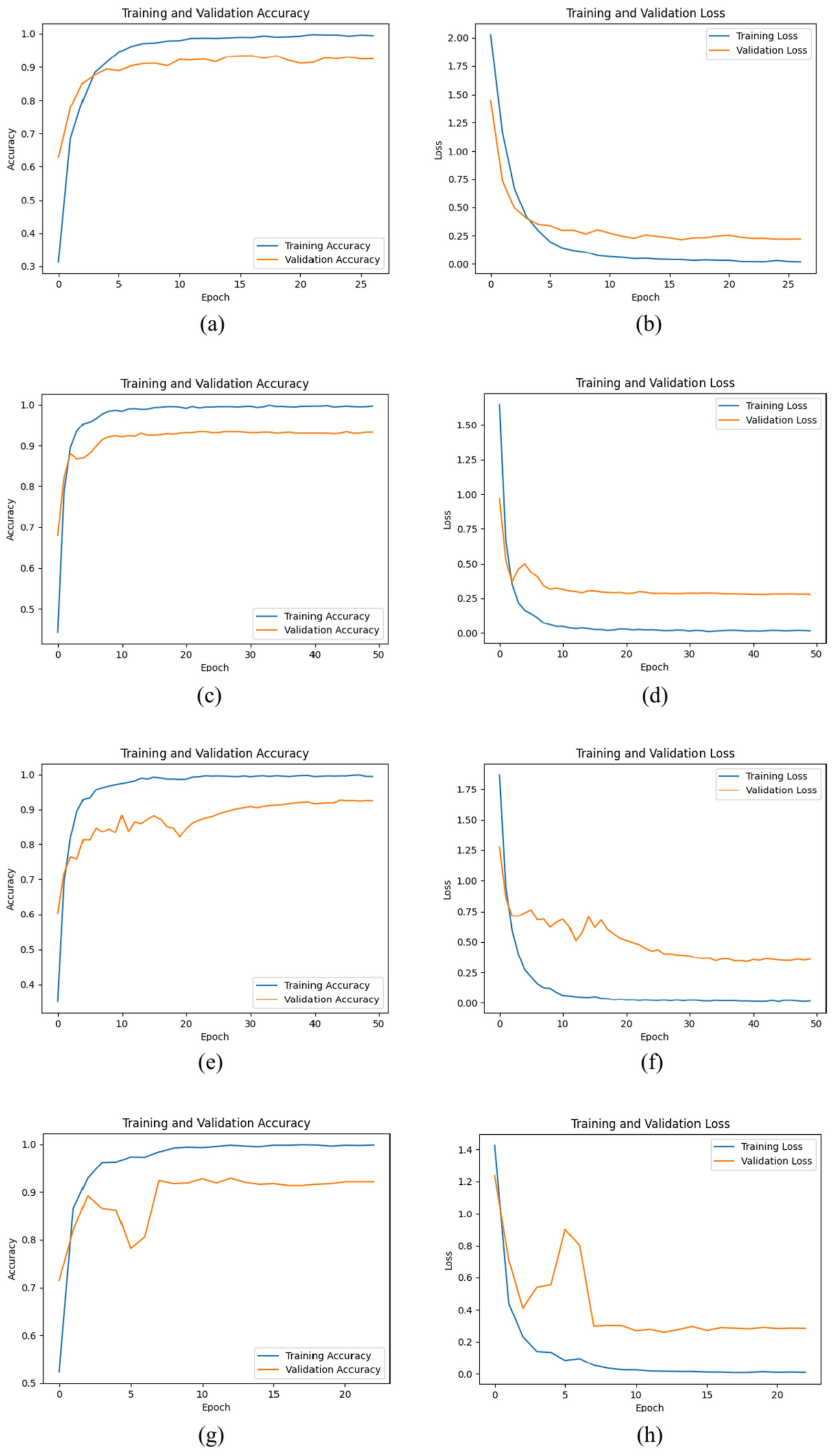
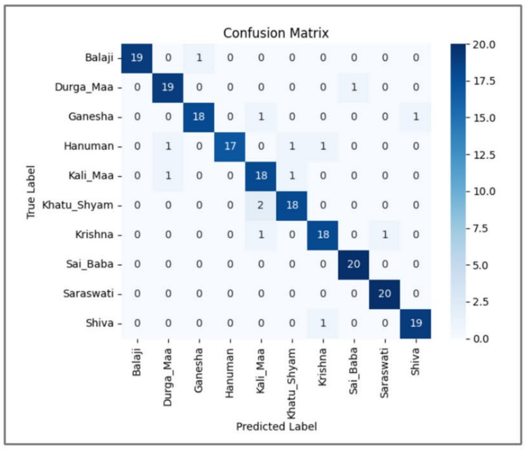


Fig. 8. Graphical representation of epoch-wise training accuracy and loss of different deep models. (**a**–**b**) Corresponding training accuracy and loss plot for MobileNetV2, (**c**–**d**) Training accuracy and loss plot for EfficientNetB0, (**e**–**f**) training accuracy and loss plot for ResNet-50, (**g**–**h**) same plot for GoogleNet (Inception V3).

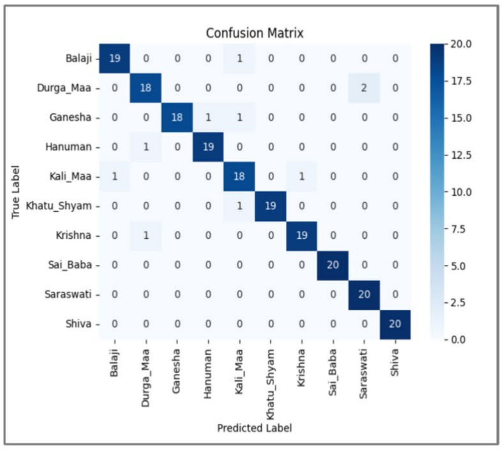
Ablation study

Model computation time

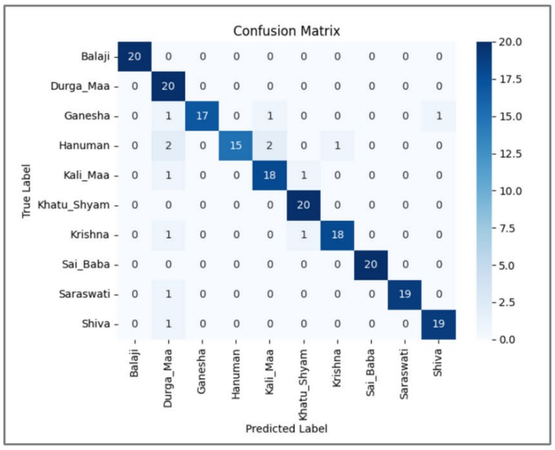
A total of 50 tests were conducted, and the inference times for all four models were calculated and averaged. Figure 12 presents the box plot of these response times, representing the time taken from input image processing



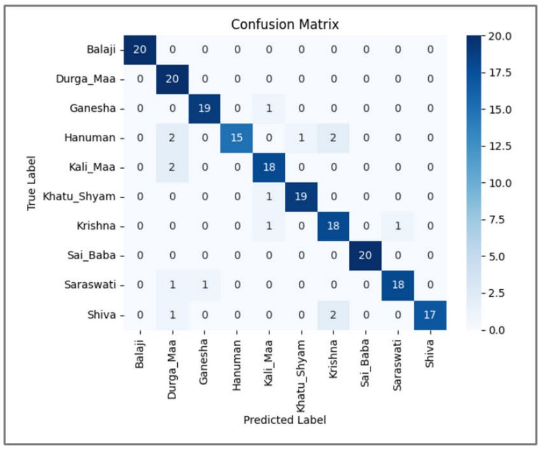
(a)



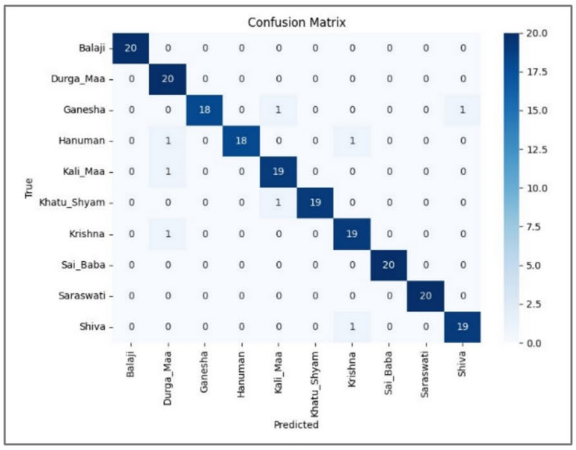
(b)



(c)



(d)



(e)

Fig. 9. Generated confusion matrix obtained from different deep models. (**a**–**d**) Confusion matrix of MobileNetV2, EfficientNetB0, ResNet-50 and GoogleNet, (**e**) generated confusion matrix from weight-centric decision mechanism.

to output prediction for each model. The plot visually illustrates the distribution of inference times, including the median, range, and potential outliers, providing insights into the performance variability across different models. From the plot, we observe MobileNetV2 has the fastest and most consistent inference time with the least variability. EfficientNetB0 shows slightly higher variance but still performs faster than the other models. Table 10 compares the performance of four deep neural networks (MobileNetV2, EfficientNetB0, ResNet-50,

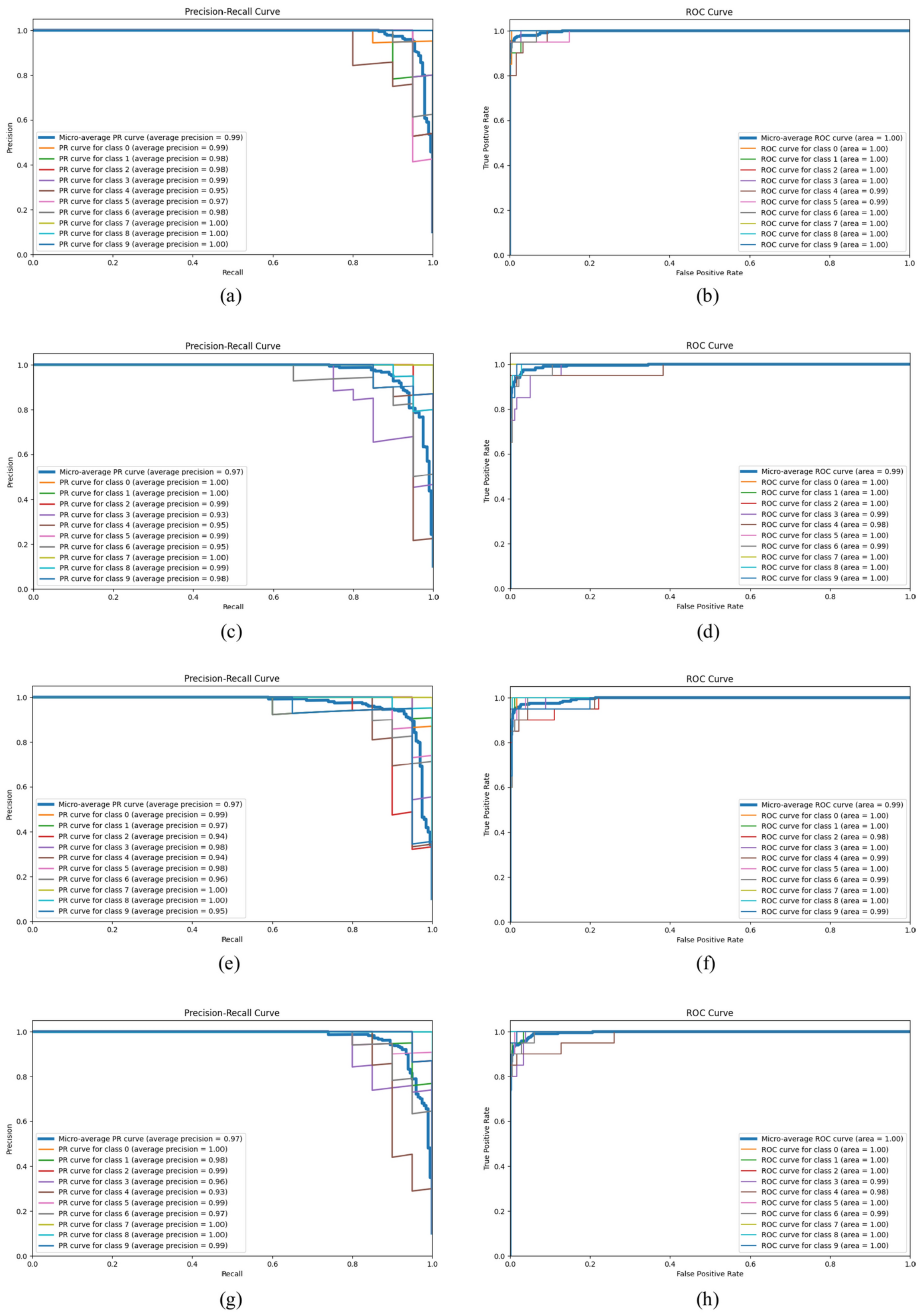


Fig. 10. Precision(P)–Recall(R) curve and ROC curve of different deep models. (**a**–**b**) Corresponding PR and ROC curve for MobileNetV2, (**c**–**d**) PR and ROC curve for EfficientNetB0, (**e**–**f**) PR and ROC curve for ResNet-50, (**g**–**h**) same plot for GoogleNet (Inception V3).

and GoogleNet) along with a weight-centric decision mechanism based on parameters, GFlops, accuracy, and average inference time. MobileNetV2 has the fewest parameters and achieves the fastest inference time with an accuracy of 93%. EfficientNet B0, though slightly slower but delivers the highest accuracy (95%). ResNet-50 and GoogleNet both have larger models with higher parameter counts and longer inference times, but they offer similar accuracy (93%). The weight-centric decision mechanism, which combines the predictions from all four

Employed deep neural networks	Accuracy (A)	F-measure (FM)	Precision (P)	Recall (R)	Error (ERR)
MobileNetV2	0.93	0.9302	0.9325	0.93	0.07
Efficient Net B0	0.95	0.9502	0.9516	0.95	0.05
ResNet-50	0.93	0.9305	0.9404	0.93	0.07
GoogleNet (Inception V3)	0.92	0.9204	0.9292	0.92	0.08
Weight-centric decision mechanism	0.96	0.9602	0.9629	0.96	0.04

Table 9. Performance report of different deep networks and weight-centric decision mechanism.

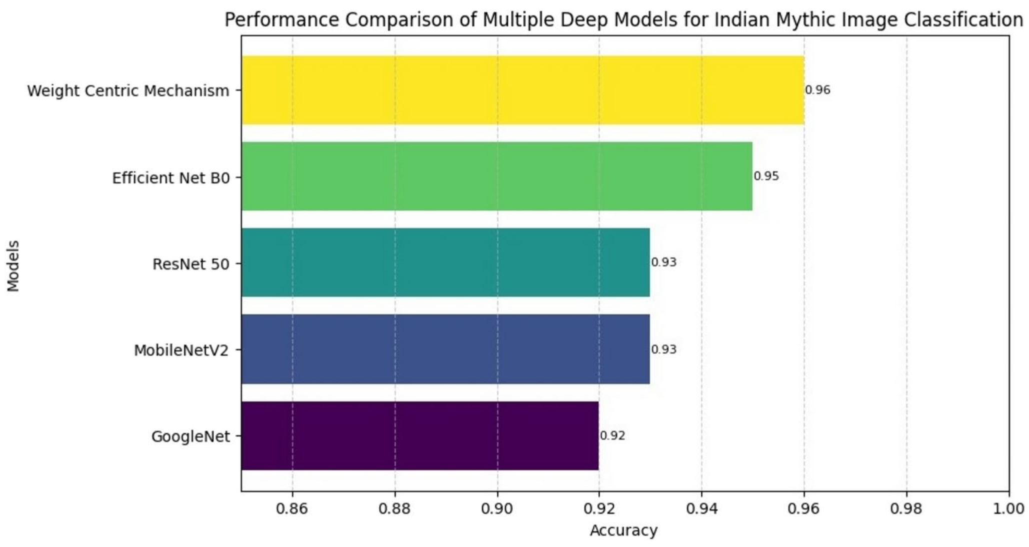


Fig. 11. Performance comparison of MobileNetV2, EfficientNetB0, ResNet-50, GoogleNet (Inception V3), and weight-centric decision mechanism on in-house developed dataset.

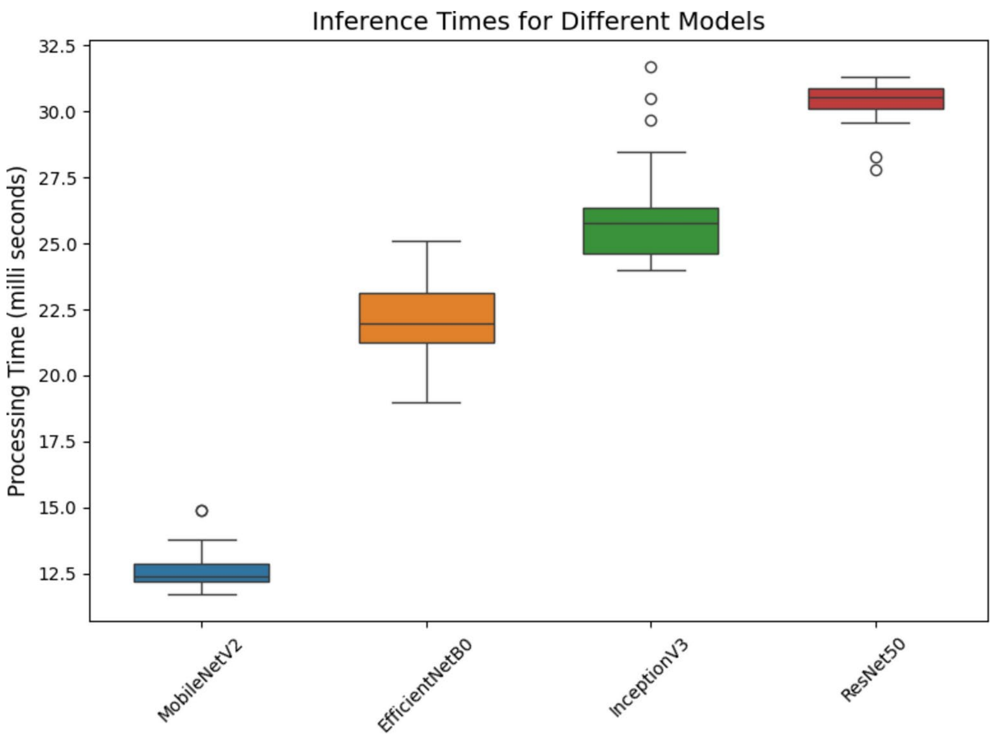


Fig. 12. Graphical representation of processing time of all employed deep models.

Employed deep neural networks	No. of parameters	GFLops	Accuracy (%)	Average Inference time (in milli-seconds)
MobileNetV2	~3 M	0.43	93	12.68
Efficient Net B0	~5 M	0.59	95	22.21
ResNet-50	~25 M	7.72	93	25.96
GoogleNet (Inception V3)	~22 M	5.68	92	30.37
Weight-centric decision mechanism	54 M	14.42	96	55.61

Table 10. Performance comparison of individual models and the weight-centric mechanism based on computational resources.

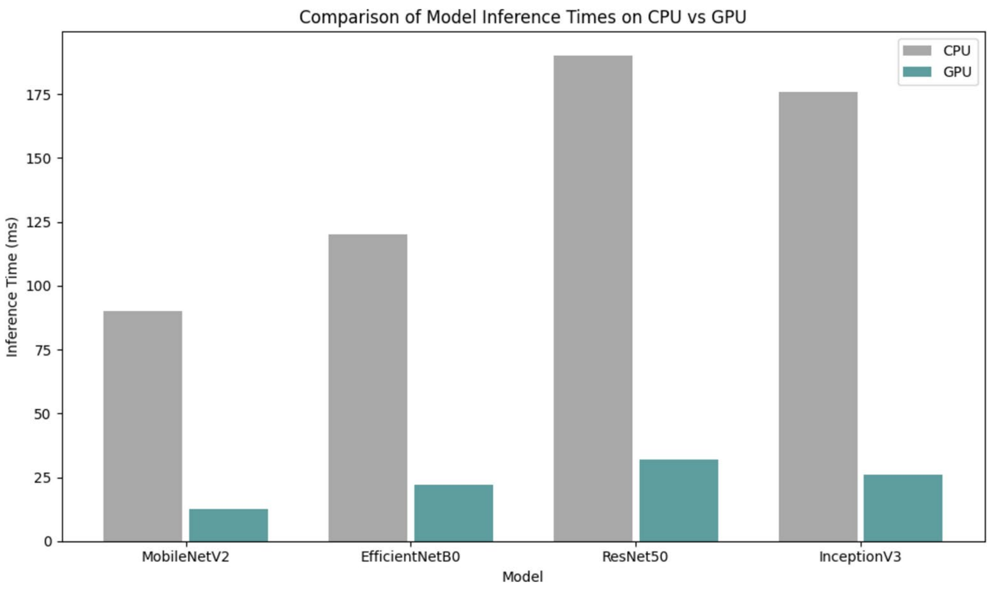


Fig. 13. Comparison of inference times for different models on CPU versus GPU: the bar plot compares the image classification inference times for MobileNetV2, EfficientNetB0, ResNet50, and InceptionV3 models on CPU and GPU.

models, has the most parameters (54 M) and achieves the highest accuracy (96%) but has the longest inference time (55.61 ms). This highlights that using multiple models in parallel for a weight-centric mechanism increases inference time, demonstrating the trade-off between leveraging multiple models for higher accuracy and the computational resources required for faster inference.

Analysis of inference time for CPU versus GPU-powered models
Figure 13 highlights the significant reduction in inference time when using GPU, with MobileNetV2 showing the fastest performance across both hardware setups. Running multiple models in parallel using multi-threading (or multi-processing) is highly beneficial in scenarios where inference time is critical. In a weight-centric mechanism that relies on predictions from all 4 models, leveraging parallel execution significantly reduces latency by utilizing hardware resources more effectively. Additionally, utilizing a GPU over a CPU greatly accelerates inference times due to the GPU's ability to handle parallel computations, making it far more efficient than a CPU for deep learning tasks.

Impact of data augmentation
Augmenting the dataset through various transformations, such as rotation, translation, and others, allowed the model to learn more effectively, reducing overfitting on the training data and improving its ability to generalize to new, unseen data. As shown in Fig. 14, both individual models and the overall framework demonstrate enhanced performance when trained on the augmented dataset.

Performance comparison with state-of-the-art works

The performance of the proposed framework is compared with other state-of-the-art models, as shown in Table 11. In most cases, the framework consistently outperforms other methods with a notably high accuracy score. This demonstrates that a weight-centric approach can significantly enhance the effectiveness of classification tasks, surpassing the results of traditional single-model approaches. Karegowda et al.[48] have utilized Xception architecture- an evolution of the inception architecture for 11 classes resulting in a respectable 94% classification accuracy score. It is to be noted that its dataset size (=2200) is substantially lower compared to

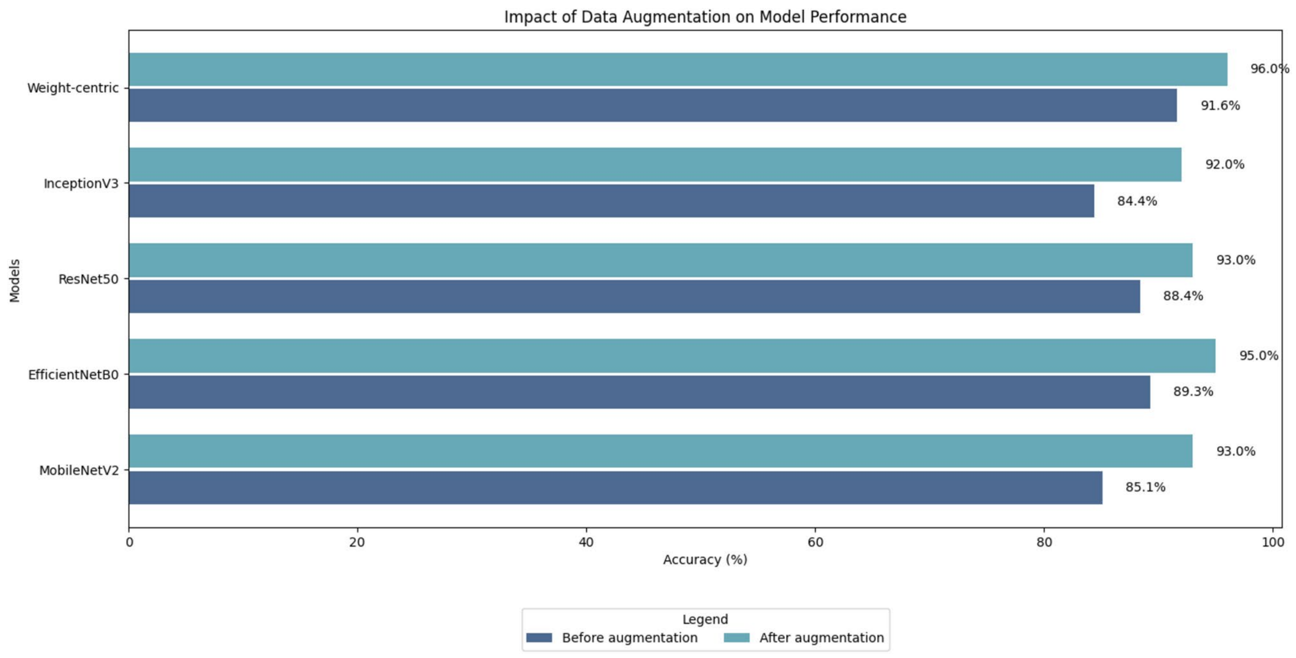


Fig. 14. Impact of data augmentation on performance for different models viz. MobileNetV2, EfficientNetB0, ResNet50, InceptionV3 and weight-centric respectively.

References	Framework	Number of classes	Dataset size	Accuracy (%)
Huang et al.[3]	MobileNetV2	5	10,786	92.31
Karegowda et al.[49]	Xception	11	2200	94.00
Mehta et al.[48]	ResNet50	9	NA	93.75
Proposed framework (ours)	Weight-centric (EfficientNetB0 + MobileNetv2 + ResNet50 + Inceptionv3)	10	10,970	96.00

Table 11. Performance comparison of MythicVision with state-of-the-art works.

our in-house dataset. Huang et al.[3] have classified Chinese God images using MobileNetv2 and obtained 92.31% accuracy. For the same work, EfficientNetB0 attained a higher accuracy of 96.15%. However, this result is based on a classification task involving only five classes, which is fewer than those considered in our work. These results underscore the robustness of our proposed framework in handling complex classification tasks across a larger number of classes and a significantly larger dataset. The weight-centric approach not only enhances accuracy but also positions the proposed framework as a highly reliable solution for large-scale applications.

Analytical discussion and limitations

Figure 15 illustrates a few correctly classified and misclassified samples identified by the MythicVision framework. Durga Maa misclassified as Kali Maa in Fig. 15f may have occurred due to the similar facial features shared by these deities. In Indian mythology, various deities often exhibit similar attributes, possibly influenced by their taken forms or representations. Hanuman misclassified as Khatu Shyam as seen in Fig. 15g was likely due to the presence of multiple floral garlands adorning the statue, obscuring key features that characterize Hanuman. A clearer, zoomed-in image of the face could potentially correct this misclassification. The primary reason for Khatu Shyam being misclassified as Sai Baba as seen in Fig. 15h could be that the input image depicted a figurine of Khatu Shyam, which lacked the distinct and recognizable features typically associated with him.

Despite the significant findings of this study, several limitations should be acknowledged. Firstly, while weight-centric mechanism offers improvements in accuracy, it still requires substantial computational power for both training and inference. Additionally, the lack of comprehensive Indian deity datasets, encompassing a wider range of deities, restricts the model's ability to generalize effectively and limits its cultural representation. Furthermore, the robustness of the proposed model in real-world scenarios, such as varying lighting conditions, angles, and artifacts, remains untested. Future research should focus on addressing these limitations by exploring strategies to reduce computational demands, expanding and diversifying the dataset to include more deities, and rigorously evaluating the model's performance in practical, real-world settings.

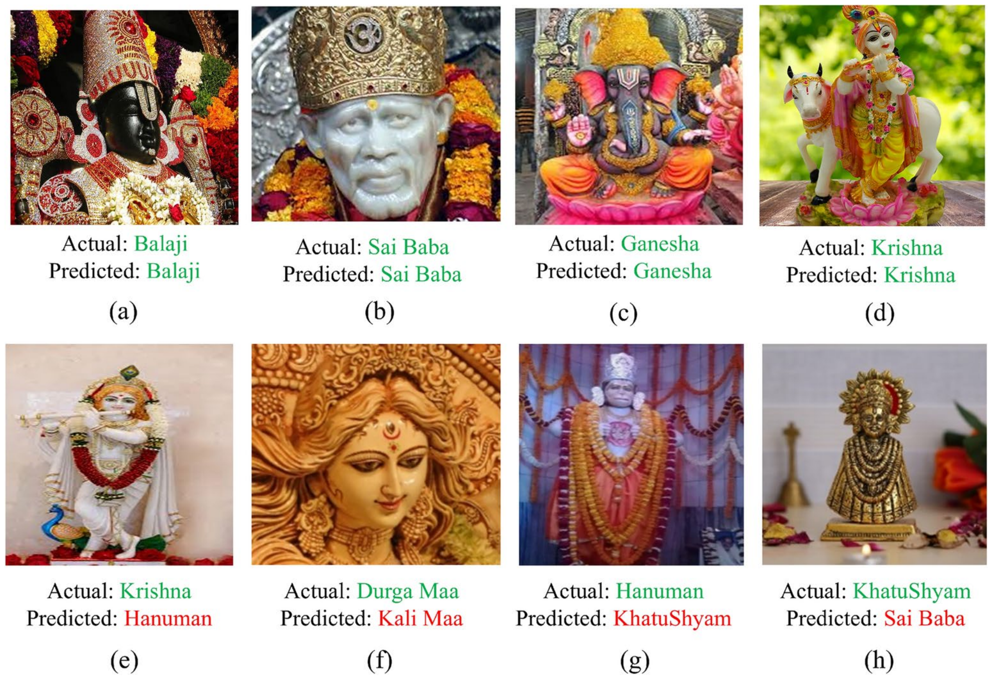


Fig. 15. Some true classified and misclassified samples of the presented MythicVision framework. (**a**–**d**) correctly classified samples, (**e**–**h**) misclassified samples along with their actual and predicted class.

Conclusion

The designed *MythicVision* mobile-driven application represents a significant advancement at the crossroads of modern deep learning and India's rich mythology. The backbone of the application is the developed deep learning-based image classification framework, seamlessly integrated into the MythicVision application. The four employed deep networks have been trained and evaluated with the train-set and test-set of our in-house dataset to achieve accuracies in the range of 90–95%. The obtained test accuracies are considered as weights and are fed to a novel weight-centric decision mechanism for the final prediction of any real-time input image. Employing such a mechanism pushed the test accuracy of the overall framework to be 96%. MythicVision will help tourists engage with Indian mythology, enriching their understanding of the culture. The future of *MythicVision* looks bright with exciting possibilities. Nevertheless, there is still room for improvement. Some of the future work that can be done is:

- Given the wide variety of gods found in Indian mythology, expanding the dataset to encompass more gods and related stories would enhance the system's richness and cultural significance.
- Adding interactive features and social networking functionalities to the mobile application could improve user engagement.
- Including systems for getting direct user input will yield insightful information about user preferences, allowing for incremental changes to better suit their requirements and expectations.
- Extending the use of the weight-centric framework to domains such as tumor classification, fraud detection, or disease diagnosis would demonstrate its versatility and broader applicability.
- Incorporating additional AI domains, such as virtual reality and augmented reality, would further enrich the proposed work.
- Adapting the system to reflect diverse cultural nuances and integrating advancements in AI to remain relevant in a rapidly evolving technological landscape.

Moreover, ensuring the system's flexibility to adapt to diverse cultural nuances and future technological AI advancements is essential. The vision is to develop *Mythicvision* into a comprehensive platform that bridges the gap between Indian mythology and modern digital engagement. In conclusion, *Mythicvision* has the potential to foster cultural awareness and provide an enriching experience for users by seamlessly integrating traditional narratives with innovative features.

Data availability

The datasets generated and/or analyzed during the current study are available in the Mythic_Vision repository, https://github.com/Adinp1213/Mythic_Vision

Received: 25 January 2024; Accepted: 7 January 2025
Published online: 21 January 2025

References

1. Rojas Rueda, D., Nieuwenhuijsen, M. J., Khreis, H. & Frumkin, H. Autonomous vehicles and public health. *Annu. Rev. Public Health* **41**, 329–345 (2020).
2. Zhang, Q. S. & Zhu, S. C. Visual interpretability for deep learning: a survey. *Front. Inf. Technol. Electron. Eng.* **19**(1), 27–39 (2018).
3. Huang, M. L. ., Liao, Y. C., Shiau, K. L. & Tseng, Y. L. Traditional Chinese god image dataset: A glimpse of Chinese culture. *Data Brief* **46**, 108861 (2023).
4. Zhang, Y., Zhang, H., Cai, J. & Yang, B. A weighted voting classifier based on differential evolution. *Abstr. Appl. Anal.* **2014**, 1–6 (2014).
5. Artificial Intelligence and Robotics: A Journey through Ancient Indian Texts, https://rishihood.edu.in/artificial-intelligence-and-robotics-a-journey-through-ancient-indian-texts/ (Accessed on January 9 2024)
6. Wang, P., Fan, E. & Wang, P. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recogn. Lett.* **141**, 61–67 (2021).
7. Mythic_Vision, https://github.com/Adinp1213/Mythic_Vision, (Accessed on January 9 2024)
8. Spagnolo, F., Perri, S., Frustaci, F. and Corsonello, P. Connected component analysis for traffic sign recognition embedded processing systems. In *Proceedings of the 25th IEEE International Conference on Electronics, Circuits and Systems*, 749–752. (2018)
9. Spagnolo, F., Frustaci, F., Perri, S. & Corsonello, P. An efficient connected component labeling architecture for embedded systems. *J. Low Power Electron. Appl.* **8**(7), 1–11 (2018).
10. Armi, L. and Fekri-Ershad, S. Texture image analysis and texture classification methods-A review', arXiv preprint arXiv:1904.06554. (2019)
11. Rastghalam, R. & Pourghassem, H. Breast cancer detection using MRF-based probable texture feature and decision-level fusion-based classification using HMM on thermography images. *Pattern Recogn.* **51**, 176–186 (2016).
12. Lalaoui, L. & Mohamadi, T. A comparative study of image region-based segmentation algorithms. *Int. J. Adv. Comput. Sci. Appl.* **4**(6), 198–206 (2013).
13. Zhang, H., Fritts, J. E. & Goldman, S. A. Image segmentation evaluation: A survey of unsupervised methods. *Comput. Vision Image Underst.* **110**(2), 260–280 (2008).
14. Neumann, L. and Matas, J. A method for text localization and recognition in real-world images. In *Proceedings of the 10th Asian Conference on Computer Vision*, Queenstown, New Zealand, 770–783. (Springer, 2011)
15. Biswas, K., Shivakumara, P., Sivanthi, S., Pal, U., Lu, Y., Liu, C.L. and Ayub, M.N.B. A New Deep Fuzzy Based MSER Model for Multiple Document Images Classification. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, 358–370, (Springer, 2022).
16. Wang, P., Fan, E. & Wang, P. 'Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recogn. Lett.* **141**, 61–67 (2021).
17. Kovalev, V., Liauchuk, V., Kalinovsky, A. & Shukelovich, A. A comparison of conventional and deep learning methods of image classification on a database of chest radiographs. *Int. J. Comput. Assist. Radiol. Surg.* **12**, 139–140 (2017).
18. Sharma, A. and Phonsa, G. Image classification using CNN. In *Proceedings of the International Conference on Innovative Computing & Communication*, 1–5. (2021)
19. Krichen, M. Convolutional neural networks: A survey. *Computers* **12**(8), 1–41 (2023).
20. Guo, T., Dong, J., Li, H. and Gao, Y. Simple convolutional neural network on image classification. In *Proceedings of the IEEE 2nd International Conference on Big Data Analysis*, 721–724, (IEEE, 2017).
21. Affonso, C., Rossi, A. L. D., Vieira, F. H. A. & de Leon Ferreira, A. C. P. Deep learning for biological image classification. *Expert Syst. Appl.* **85**, 114–122 (2017).
22. Mikołajczyk, A. and Grochowski, M. Data augmentation for improving deep learning in image classification problem. In *Proceedings of the International Interdisciplinary PhD Workshop*, 117–122 (IEEE, 2018).
23. Obaid, K. B., Zeebaree, S. & Ahmed, O. M. Deep learning models based on image classification: a review. *Int. J. Sci. Bus.* **4**(11), 75–81 (2020).
24. Hosseini, H., Xiao, B., Jaiswal, M. and Poovendran, R. (2017) 'On the limitation of convolutional neural networks in recognizing negative images', *Proceedings of the IEEE International Conference on Machine Learning and Applications*, pp. 352–358, IEEE.
25. Meena, G., Mohbey, K. K. & Kumar, S. Monkeypox recognition and prediction from visuals using deep transfer learning-based neural networks. *Multimed. Tools Appl.* **83**(7), 71695–71719. https://doi.org/10.1007/s11042-024-18437-z (2024).
26. Meena, G., Mohbey, K. K., Acharya, M. & Lokesh, K. An improved convolutional neural network-based model for detecting brain tumors from augmented MRI images. *J. Autonom. Intell.* https://doi.org/10.32629/jai.v6i1.5611 (2023).
27. Meena, G., Mohbey, K. K., Indian, A., Khan, M. Z. & Kumar, S. Identifying emotions from facial expressions using a deep convolutional neural network-based approach. *Multimed. Tools Appl* **83**(6), 15711–15732 (2024).
28. Meena, G., Mohbey, K. K., Indian, A. & Kumar, S. Sentiment analysis from images using vgg19 based transfer learning approach. *Proc. Comput. Sci.* **204**, 411–418 (2022).
29. Meena, G. & Mohbey, K. K. Sentiment analysis on images using different transfer learning models. *Proc. Comput. Sci.* **218**, 1640–1649 (2023).
30. Meena, G., Mohbey, K. K. & Kumar, S. Sentiment analysis on images using convolutional neural networks based Inception-V3 transfer learning approach. *Int. J. Inf. Manag. Data Insights* **3**(1), 100174 (2023).
31. Sherly, P. S. & Velvizhy, P. "Idol talks!" AI-driven image to text to speech: Illustrated by an application to images of deities. *Herit. Sci.* **12**(1), 1–21 (2024).
32. Sasithradevi, A., Sabarinathan, S., Shoba, S. M. & Mansoor, P. KolamNetV2: Efficient attention-based deep learning network for Tamil heritage art-kolam classification. *Herit. Sci.* **12**(1), 60 (2024).
33. Sasithradevi, A., Nathan, S., Chanthini, B., Subbulakshmi, T. & Prakash, P. MonuNet: a high-performance deep learning network for Kolkata heritage image classification. *Herit. Sci.* **12**(1), 242 (2024).
34. Janković Babić, R. A comparison of methods for image classification of cultural heritage using transfer learning for feature extraction. *Neural Comput. Appl.* **36**, 11699–11709. https://doi.org/10.1007/s00521-023-08764-x (2024).
35. Gao, L. et al. Research on image classification and retrieval using deep learning with attention mechanism on diaspora Chinese architectural heritage in Jiangmen, China. *Buildings* **13**(2), 275. https://doi.org/10.3390/buildings13020275 (2023).
36. Unsplash. (n.d.). *Free high-resolution photos.* https://unsplash.com/ (Accessed on January 9 2024)
37. Pexels. (n.d.). *Free stock photos and videos.* https://www.pexels.com/ (Accessed on January 9 2024)
38. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520. (2018)
39. Tragoudaras, A. et al. Design space exploration of a sparse mobilenetv2 using high-level synthesis and sparse matrix techniques on FPGAs. *Sensors* **22**(12), 4318 (2022).
40. MobileNetv2, https://pytorch.org/hub/pytorch_vision_mobilenet_v2 (Accessed on January 9 2024)
41. Tan, M. and Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning*, 6105–6114. (2019)
42. Alhichri, H., Alsuwayed, A., Bazi, Y., Ammour, N. & Alajlan, N. Classification of remote sensing images using EfficientNet-B3 CNN model with attention. *IEEE Access* **1**, 1–1. https://doi.org/10.1109/ACCESS.2021.3051085 (2021).
43. He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. (2016)

44. Oguntayo, S. Implementing transfer learning from RGB to multi-channel imagery: Semantic segmentation using ResNet50 backbone plus pyramid pooling. *Towards Data Science*. https://towardsdatascience.com/implementing-transfer-learning-from-rgb-to-multi-channel-imagery-f87924679166 (Accessed on January 9 2024) (2021).
45. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2818–2826. (2016)
46. Ye, Z., Li, M., Han, S., Ren, Q. & Shi, J. Intelligent identification for rock-mineral microscopic images using ensemble machine learning algorithms. *Sensors* **19**(18), 1–14. https://doi.org/10.3390/s19183914 (2019).
47. Kingma, D.P. and Ba, J. (2014) Adam: A method for stochastic optimization', arXiv preprint arXiv:1412.6980.
48. Mehta, A., Sindhu, C., Dube, S., Kanneganti, M., & Taneesha, T. K. NICE: Navagraha Iconography Classification Engine. In *International Conference on ICT for Sustainable Development*, Vol. 754, 59–70. (Springer Nature Singapore, 2023).
49. Karegowda, A.G., Pooja, R., Tara, K.N., Leena Rani, A. Identification of Deity Images Using CNN and Transfer Learning Models. In *Intelligent Control, Robotics, and Industrial Automation. RCAAI 2023. Lecture Notes in Electrical Engineering*, Vol. 1220, 153–167 (Springer, Singapore, 2024). https://doi.org/10.1007/978-981-97-4650-7_1.

Acknowledgements
The authors extend their sincere gratitude to the School of Computer Science and Engineering at VIT-AP University for their invaluable support throughout the duration of this research study.

Author contributions
T.K.: Conceptual design and communicating author, manuscript reviewing A.N.P.: Data collection, original manuscript drafting, analysis and framework design, code development A.S.: Data visualization G.P.B.: Data pre-procesing K.S.A.: Code development S.N.M.: Result analysis and manuscript drafting, manuscript reviewing.

Funding
No funding was received for conducting this study.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to T.K.