**ABSTRACT**

The "Mythic Vision" software represents an innovative and socially impactful capstone project with a primary focus on enhancing the cultural experiences of tourists visiting India. The project's core objective is the development of a deep learning-based software solution that possesses the capability to detect and classify representations of Indian deities found within various cultural artifacts, sculptures, and artworks. The software aspires to equip tourists with a deeper comprehension of India's cultural heritage and traditions, thereby elevating their overall tourism experience.

1. **INTRODUCTION**

Indian mythology is a treasure trove 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 hasn't seamlessly blended with modern technology. Our developed model, "Mythic Vision" steps in to bridge this gap. It aims to reveal the hidden stories in India's cultural artifacts, revolutionizing how people interact with Indian mythology during cultural tourism experiences.

At the heart of the developed project lies the field of deep learning, specifically image classification. In the context of deep learning, image classification refers to the process of training algorithms to recognize and categorize objects or patterns within digital images. This technology has reshaped numerous industries, from healthcare to autonomous vehicles, by enabling machines to interpret visual data with remarkable accuracy. In the context of our project, image classification becomes the key in understanding the representations of Indian deities, unraveling their stories, and bringing this knowledge to the fingertips of travelers.

While the potential of image classification in deep learning has been harnessed in various domains, its application to Indian mythology has remained conspicuously absent. An exploration of similar endeavors reveals a notable project, "Traditional Chinese God Image Dataset: A Glimpse of Chinese Culture,". Huang et Al. [1] which sought to achieve a similar objective in the context of Chinese culture. However, the fact that there are very few similar projects in the field of Indian mythology underscores how unique and innovative the "Mythic Vision" project is.

The primary challenges addressed by the developed project explains several aspects. Firstly, the lack of a complete dataset of Indian deities is a major obstacle. Secondly, the project faces the task of selecting the most suitable deep learning model from a range of options. Additionally, the dilemma of whether to prioritize individual model accuracy or employ a weighted decision-based accuracy approach presents a critical decision-making challenge [2]. These intricate issues form the core of the project's exploration and innovation.

The aim of the developed application named *MythicVision* is to combine Indian mythology with modern technology to enhance cultural tourism [3]. To fulfill this vision, the project follows a systematic sequence of work. It begins by making unique datasets, carefully selecting images of Indian deities from various places and sources. Next, the project delves into deep learning by selecting the most appropriate model from a wide range of options [4]. It then solves the problem of accuracy, choosing to prioritize a weighted decision-based approach over individual model accuracy. Finally, the project concludes by developing an intuitive application, which offers travelers a gateway to the enchanting world of Indian mythology. MythicVision is a comprehensive project that can be divided into four key stages.

* First, it involves the meticulous process of sourcing, selecting, and curating images of Indian deities to construct our custom in-house dataset.
* Next, it focuses on the selection of appropriate deep learning models, such as MobileNet, ResNet, EfficientNet, and GoogleNet, based on their suitability for the project's objectives.
* A critical aspect of this project is the adoption of a weight-centric decision approach, prioritizing it over individual model accuracy to enhance the accuracy and reliability of the results.
* Finally, the project culminates in the development of an intuitive and user-friendly application that allows users to interact with and benefit from the project's findings.

The developed dataset and framework have been made publicly available through a GitHub repository for academic, research, and non-commercial purposes, as referenced in this paper [5].

Through this journey, the developed project endeavors to bridge the gap between cultural exploration and technological innovation, uncovering hidden stories of Indian deities and culture.

The paper is organized into 5 sections: In Section1, this research paper introduces the subject matter, emphasizing its significance and relevance. It outlines the purpose and driving force behind undertaking this research and clearly defines the objectives of the paper. Section 2 delves into the existing body of research in this domain, examining the prior work and achievements that have been accomplished thus far. Section 3 covers the dataset creation process, explains the use of four deep learning models, and delves into the incorporation of the weight-centric mechanism in the proposed model, emphasizing its role in improving overall performance. Section 4 of this research paper is dedicated to the presentation of experimental results and their in-depth analysis. In Section 5, the research paper offers a comprehensive conclusion, synthesizing the main discoveries and providing a broader perspective on the implications of the study for the field. Moreover, this research paper delves into the potential future avenues for further exploration in the research work. Additionally, a comprehensive list of references cited within the paper has been provided at the end.

1. **RELATED WORK**

In the field of image classification, traditional methods have been used to categorize objects in images. These methods typically involve hand-crafted features and trained classifiers. They fall into several categories: Connected Component (CC) based methods segment images to distinguish text from non-text using features and classifiers [6], but they have limitations in complex environments and require extensive pre-processing. Texture-based methods classify text based on patterns [7], while region-based methods label [8] object components based on properties and shapes, sometimes using techniques like MSERs [9]. These methods are effective but often involve substantial pre-processing and post-processing, reducing efficiency. It's important to note that conventional image classification methods have certain limitations [10]. They often demand extensive feature engineering, may not perform well in complex and dynamic environments, and rely on manual parameter tuning. In our project, we aim to overcome these limitations by embracing deep learning techniques. This research paper leverages the insights from various studies on conventional methods for chest radiograph classification, citing relevant papers, to provide a foundation for the comparative analysis of conventional and deep learning approaches [11].

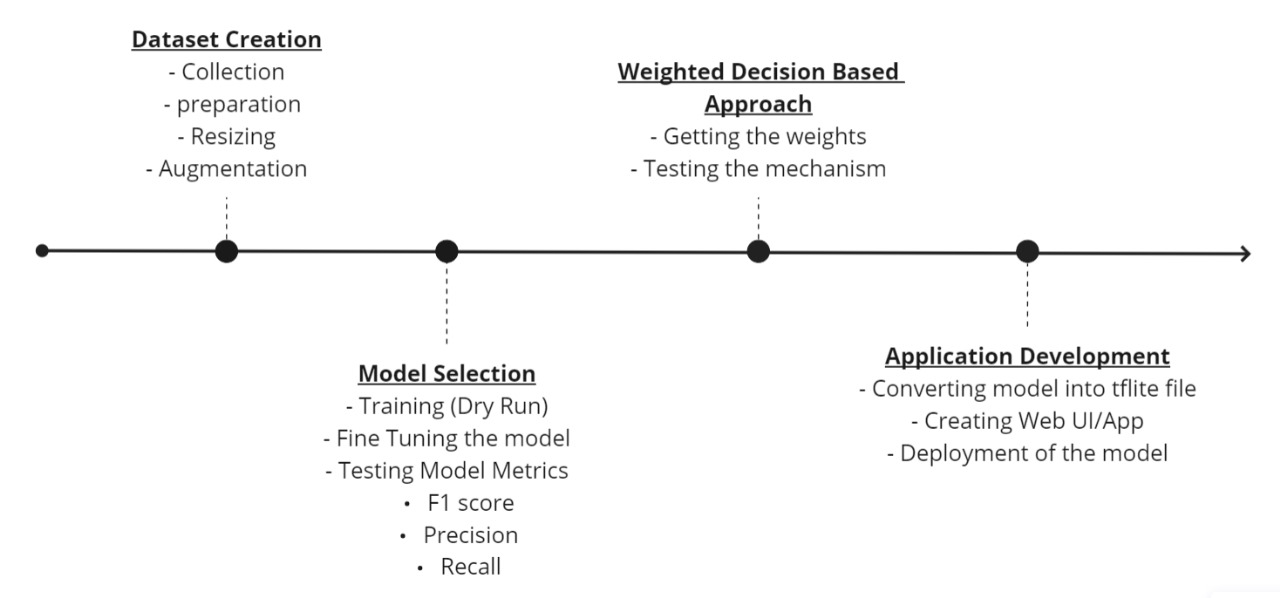
In recent years, deep learning has transformed image classification. This section delves into the use of deep neural networks in image classification and their benefits compared to traditional methods. Deep learning methods excel in automatically extracting intricate image features, adapting to complex situations, and achieving cutting-edge results. Among these, Convolutional Neural Networks (CNNs) stand out as pivotal [12]. They can learn 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.

A project like our venture has been undertaken in the realm of Chinese culture. Huang et Al. [1] this initiative shares similarities with our goals but is tailored to Chinese mythology. It stands as a testament to the potential impact of bridging deep learning with cultural exploration. In our research, we draw upon the valuable insights provided by the paper titled ‘Deep Learning and Transfer Learning Approaches for Image Classification' to inform and support our own exploration of image classification methods, particularly in the context of deep learning and transfer learning techniques [13].

After a careful survey, as outlined above, several challenges have emerged in the context of cultural exploration and technological innovation. These challenges include the absence of comprehensive datasets, the selection of appropriate deep learning models, and the dilemma of accuracy prioritization. Firstly, there's a lack of comprehensive datasets, particularly for Indian mythology, making it tough to create accurate deity and artifact classification models. Secondly, selecting the right deep learning model from a diverse range is tricky, as it needs to understand the vastness of Indian mythology. Lastly, we're dealing with the decision of whether to prioritize individual model accuracy or use a weighted decision-based accuracy approach, which directly affects the output, that is, quality of information our software provides to tourists about Indian deities. In response to these challenges, the developed project has been designed to address each aspect comprehensively. Our project 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.

1. **Developed Framework**

This section provides a succinct overview of the project's four key stages. It begins with the meticulous process of acquiring, selecting, and curating images of Indian deities to create custom datasets. The next stage involves selecting appropriate deep learning models, such as MobileNet, ResNet, EfficientNet, and GoogleNet, tailored to the project's specific objectives. The paper emphasizes the prioritization of a weighted decision-based approach over individual model accuracy to enhance the overall accuracy and reliability of the results. Finally, the project culminates in the development of MythicVision, an intuitive and user-friendly application that allows users to interact with and benefit from the project's findings.

****

**Fig1.** Overall pipeline of the developed framework

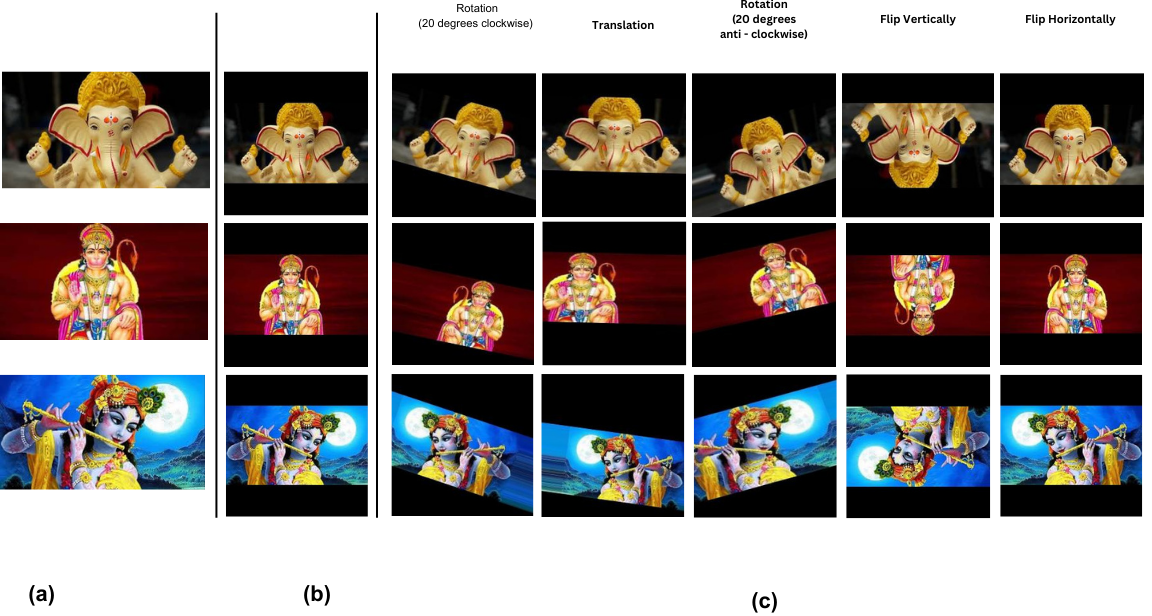
<slight colorful and styling>

* 1. ***Dataset Preparation***

The preparation of the dataset for the project assumed a crucial role given the absence of existing datasets online. With a set of ten distinct deities, each characterized by unique symbolism, a total of 500 images per deity were acquired. From the initial pool, a selection process was initiated to identify images aligning precisely with the project's objectives. The images were filtered, resulting in a refined subset of 300 images per deity. Within this subset, 80 images were earmarked for validation, while an additional 20 were designated for testing purposes. The remaining 200 images underwent a transformative augmentation process. This process included standardizing the resolution of each image through resizing and padding, ensuring a uniform framework for subsequent analysis. Furthermore, each image underwent a 20-degree rotation, effectively generating a new image as shown in Figure [1]. Consequently, a total of 1000 images were allocated for each deity, comprising the training phase for the subsequent image classification task as shown in Table 1.

**Table 1**. Brief outline of the experimental dataset along with its particulars.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Training  Before Augmentation | Training  After Augmentation | Validation | Testing |  |
| 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 |  |



**Fig.1** Dataset Creation, where (a) is the original image from the source, (b) is the resized 224x224 pixel padded image, and (c) is the image that is being augmented in 5 different ways mentioned above.

* 1. ***Employed Deep Models***

Within this sub-section, detailed insights are provided into the approach employed for selecting appropriate deep learning models. A comprehensive discussion ensues regarding the various models under consideration, including MobileNet, Resnet, EfficientNet, and GoogleNet.

* + 1. *MobileNetV2*

MobileNetV2 [14] is designed to be lightweight and efficient, making it suitable for deployment on embedded systems. It balances model size, speed, and accuracy, making it a valuable architecture for various computer vision tasks, including image classification, object detection, and more. MobileNet V2 model has 53 convolution layers and 1 Average pool with nearly 350 GFLOP.

The system comprises two main components. First, it includes the Inverted Residual Block, which plays a pivotal role in its architecture. Additionally, the system incorporates the Bottleneck Residual Block as the second key component.

|  |  |
| --- | --- |
|  |  |

**Fig. 2**. The overall framework and working principle of MobileNetv2.

Each block within the system consists of three distinct layers.

Firstly, it includes a 1x1 Convolution layer with Relu6 activation, which serves as a non-linear transformation applied to the input data.

Following this, the block incorporates a Depth wise Convolution layer, which convolves the input independently for each channel, reducing computational complexity.

Lastly, it features a 1x1 Convolution layer without any linearity, allowing the model to capture linear relationships in the data.

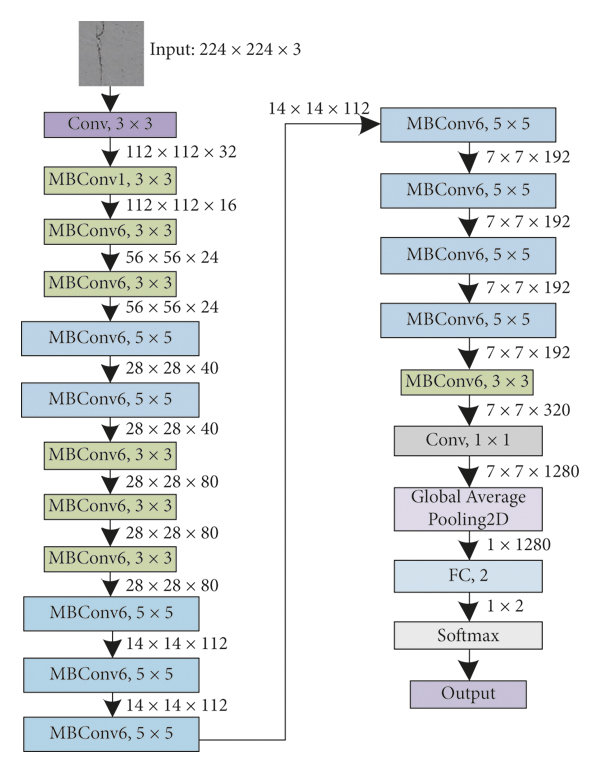
* + 1. *EfficientNetB0*

EfficientNetB0 [15] is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. EfficientNet employs a compound scaling approach, adjusting the width, depth, and resolution of the network with a compounding coefficient (ϕ). The key elements include:

Width Scaling (w): It increases the number of channels in convolutional layers, effectively widening the network.

Depth Scaling (d): This increases the number of layers, making the network deeper.

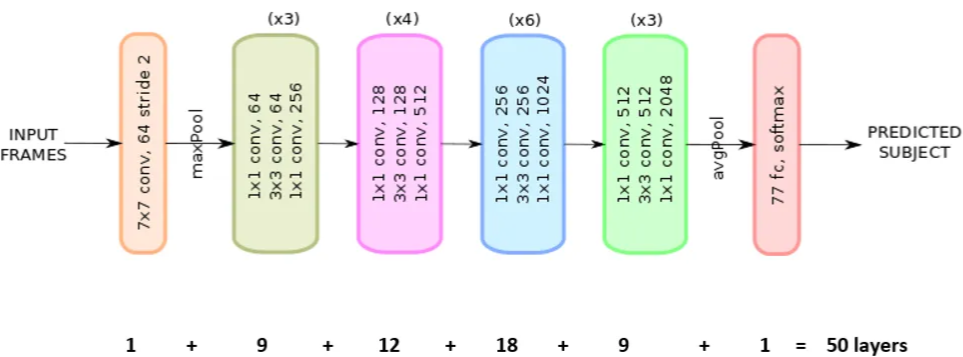
Resolution Scaling (r): It changes the input image's resolution, allowing the network to handle different image sizes.



**Fig 3.** Architecture of EfficientNetB0

* + 1. *ResNet – 50*

ResNet-50 [16] is a deep neural network architecture consisting of 48 convolutional layers, 1 average pooling layer, and 1 fully connected layer. It is a type of deep residual neural network that utilizes 3-layered bottleneck blocks to facilitate feature extraction and model training. In the network, the Rectified Linear Unit (ReLU) activation function is applied to every set of three consecutive convolutional layers. This non-linear activation helps the network capture intricate features within the data. Additionally, batch normalization techniques are incorporated to ensure stable and accurate training. 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. The resnet50 architecture comprised of the below mentioned layers as depicted in Fig. 6.



***Fig.4*** *Architecture of Layers of ResNet*

The network architecture consists of various convolutional layers with different kernel sizes and numbers of channels. The first layer employs 64 kernels with a size of 7 \* 7 and a stride of 2, resulting in one layer. Subsequently, a max-pooling layer follows with a stride of 2.

In the subsequent convolutional layers, there are different kernel sizes, including 1 \* 1 with 64 channels, 3 \* 3 with 64 channels, and 1 \* 1 with 256 channels. This set of layers is repeated three times, totaling nine layers.

Another set of convolutional layers is composed of 1 \* 1 with 128 channels, 3 \* 3 with 128 channels, and 1 \* 1 with 512 channels. This set is repeated four times, giving us 12 layers.

Following that, there are convolutional layers with kernel sizes of 1 \* 1 and channels of 256, 3 \* 3 and channels of 256, and 1 \* 1 with 1024 channels. This group is repeated six times, leading to 18 layers.

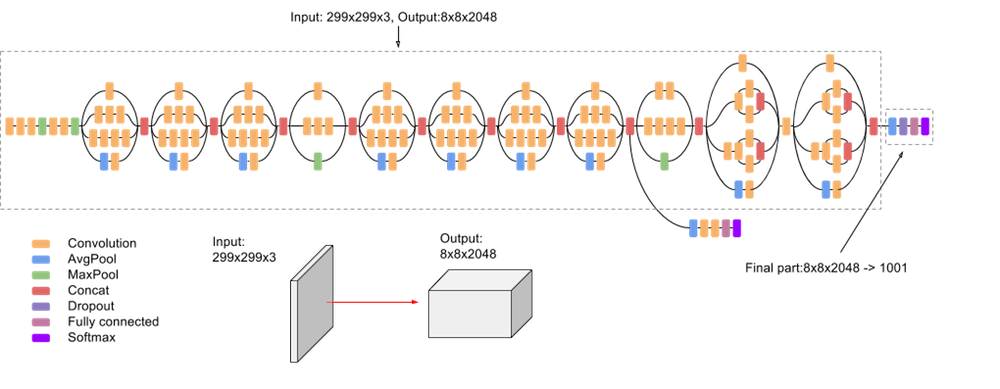
Additionally, there are convolutional layers comprising 1 \* 1 with 512 channels, 3 \* 3 with 512 channels, and 1 \* 1 with 2048 channels. This set is repeated three times, resulting in nine layers.

After these convolutional layers, average pooling is applied, followed by a fully connected layer containing 1000 nodes and a SoftMax function, which amounts to one layer.

It's worth noting that activation functions and pooling layers are not counted in this calculation. Therefore, the total number of layers in this deep convolutional network is determined by adding the layers in each set:

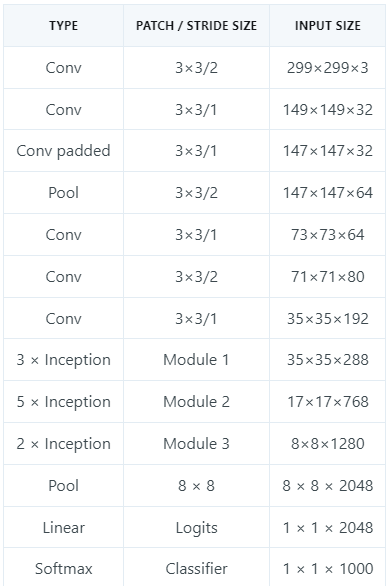
* + 1. *GoogleNet Inception V3*

The Google Inception V3 [17], also known as Inception Net version 3, is a deep convolutional neural network architecture. It builds upon the foundation laid by its predecessors, Inception V1 and Inception V2, by adding more layers to the network. The increased layer count allows Inception V3 to capture and process more complex features and patterns within images, making it highly effective for tasks like image classification and object detection. The Inception V3 model comprises a total of 42 layers.



**Fig.5** Architecture of GoogleNet (Inception V3)

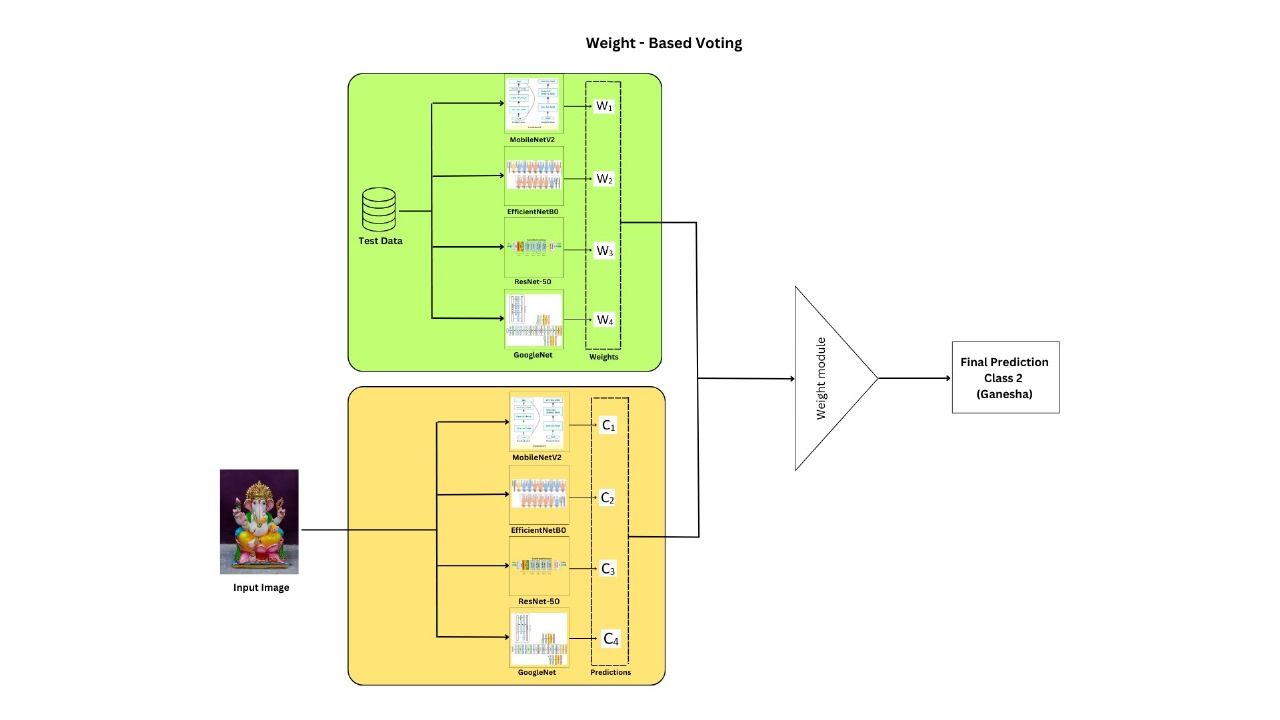
Following are the components of the Inception V3 model:



***Fig.6*** *Components of the Inception V3 model*

* 1. ***Weight-centric decision mechanism.***

To address the complexities of majority-based voting approach and ensure more accurate results, the project integrates a weight-centric decision mechanism, which will be elaborated upon in the subsequent paragraph. This approach enhances the reliability and precision of the desired result, further improving the software's performance.



**Fig. 7** Overall working principle of weight-centric mechanism. The weight-centric mechanism prioritizes and assigns varying weights to features, enhancing decision-making and customization in machine learning models.

In the Weight-Based Voting mechanism, the project employs a strategic approach to enhance classification accuracy. Prior to making predictions, each of the four models undergo thorough testing on a dedicated dataset, allowing for the calculation of individual model accuracies. These accuracies are subsequently used as weight values for the final classification. To illustrate this, if, after testing, the model accuracies are found to be 95%, 93%, 93%, and 92%, and an input image is analyzed by the models, the predictions are as follows: Ganesha – 2, Hanuman – 1, Khatu Shyam – 1. Weight values are then assigned to each class based on their respective model accuracies. For instance, Ganesha is assigned a weight value of 0.95 + 0.93, summing up to 1.88, while Hanuman receives a weight value of 0.93, and Khatu Shyam gets 0.92. With class Ganesha having the highest weighted value, the final prediction becomes Ganesha.

The mechanism can also be put in a formula mathematically, The "" in Eq.1. represents the ultimate class prediction, and it is determined by selecting the class that maximizes the weighted sum of model predictions. This sum is computed by iterating over all available models (indexed from 1 to N). Each model's contribution to the sum is determined by its assigned weight, which is typically based on its test accuracy.

|  |  |
| --- | --- |
|  | (1) |

The function evaluates whether predicts as shown in Eq.2. By summing up these weighted predictions from all models, the equation provides a comprehensive and weighted assessment of each class's likelihood, ultimately leading to the selection of the class with the highest score as the final prediction. In Eq.2, is defined as a piecewise function, returning 1 if predicts and 0 if it doesn't. This function essentially checks if the prediction made by matches , and the value is set accordingly to 1 or 0.

|  |  |
| --- | --- |
|  | (2) |

Now, addressing the intricacies of test cases within the Majority-Based Voting mechanism could lead to inaccurate results.

* + 1. *Significance of weight-centric mechanism*

The weight-centric mechanism stands out as a superior alternative to the conventional majority-based 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 to factors based on their significance, 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 2.

**Table 2**. Brief outline of the experimental dataset along with its particulars.

|  |  |  |
| --- | --- | --- |
| Scenario | Conventional majority-based approach | weight-centric decision mechanism |
| **Case 1.** Two classes achieve an equal majority vote, resulting in a tie situation. | Result: Random assumption | Result: accurate classification |
| ***Case 2.*** All four models provide distinct output, resulting in a four-way tie. | Result: Random assumption among the four predictions | Result: Predictions made based on weights thus, more accurate |
| ***Case 3.*** The models have different levels of training or are not adequately trained. | Result: Low accuracies might hinder majority vote | Result: Low accuracies doesn’t affect weighted vote thereby gives accurate predictions |

*Example Test Cases of all scenarios*

*Case 1.*Two classes achieve an equal majority vote, resulting in a tie situation.

In cases where models tie in their predictions, the mechanism excels. For instance, if EfficientNet and MobileNet predict Ganesha (class 2), while Resnet and GoogleNet predict Hanuman (class 6), each with weighted values as Ganesha (1.88) and Hanuman (1.84), the final prediction, being Ganesha, is chosen based on the highest weighted value.

*Case 2.*All four models provide distinct output, resulting in a four-way tie.

When all four models predict distinct classes, the mechanism smoothly resolves the situation. Suppose EfficientNet predicts Ganesha, MobileNet predicts Hanuman, Resnet predicts Khatu Shyam, and GoogleNet predicts Sai Baba, each with weight values assigned accordingly. Ganesha holds the highest weighted value (0.95), leading to its selection as the final prediction.

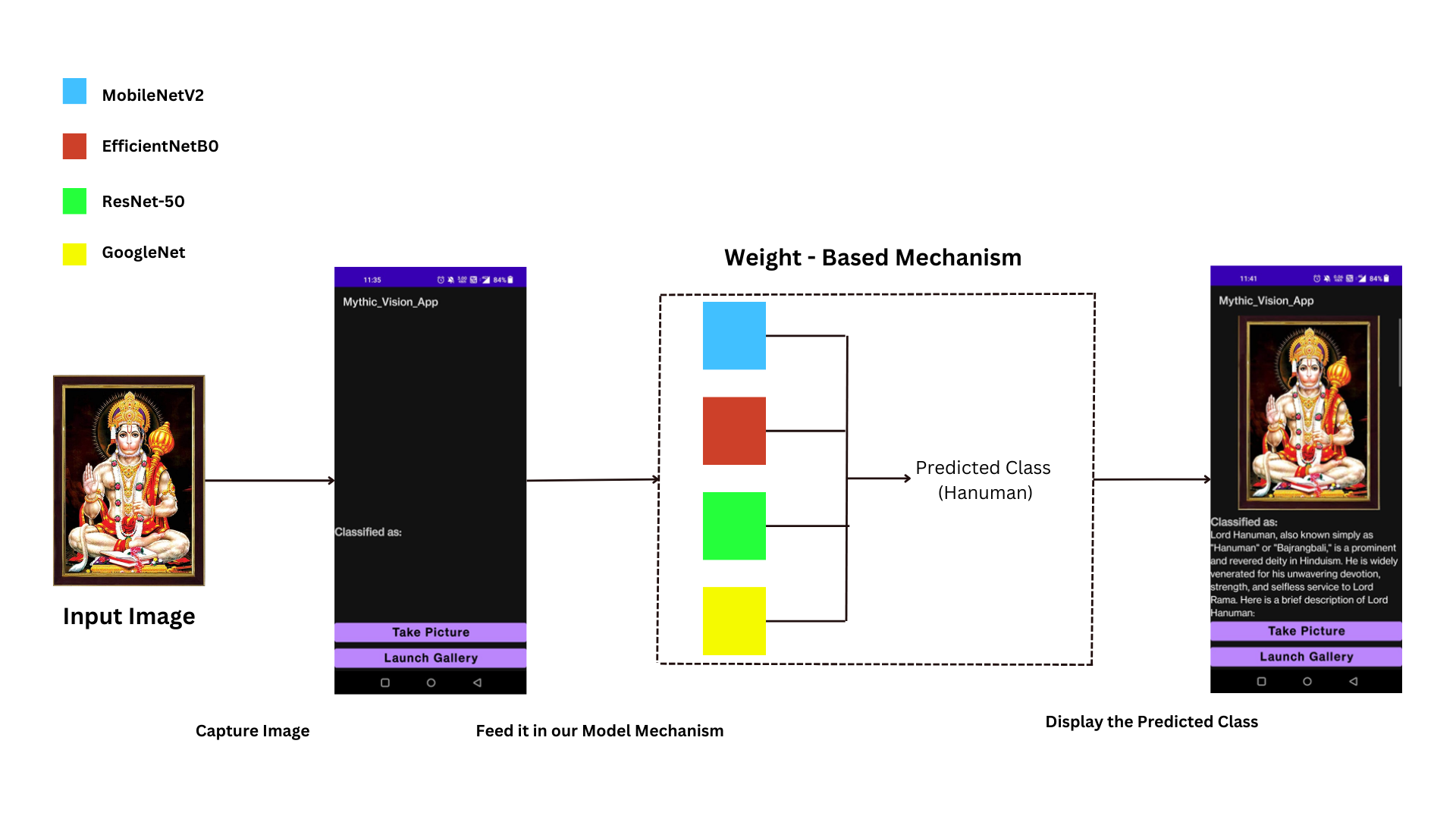
*Case 3.* The models have different levels of training or are not adequately trained.

When the models are not trained uniformly due to variations in accuracy, the mechanism ensures consistent predictions. In an example where EfficientNet achieves 95% accuracy in predicting Ganesha, while MobileNet (20%), Resnet (30%), and GoogleNet (10%) exhibit different accuracies in predicting Khatu Shyam, the final prediction aligns with Ganesha, given its maximum weighted value (0.95) compared to Khatu Shyam's combined weighted value of 0.60.

* 1. ***Development of MythicVision App***

The operational flow of the developed application is illustrated in Fig.8. The final model, which was created after the key stages of dataset creation, model selection, and the weight-based approach, is integrated into an application, the name of the application is Mythic Vision. And this is achieved through the conversion of the model into a TensorFlow Lite file, which is then exported to Android Studio, a dynamic platform for application development, where a user-friendly application with a basic User Interface (UI) interface was developed. 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 app. This image is subsequently channeled into the integrated models, where the deep learning models process it, where the weight-based approach 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.



**Fig.8** Mechanism of Application usage

1. **Experimental Results and Analysis**

A series of experiments was performed on an in-house dataset [5]. These experiments employed various deep learning models, and the resulting outcomes have been documented. Subsequently, an insightful analysis and discussion are presented based on the obtained results. This section commences with an overview of the dataset preparation process.

The research encompasses the presentation of accuracy and loss trends observed during the training of the model. Specifically, a total of four accuracy plots and four loss plots corresponding to four distinct models – MobileNetV2, EfficientNetB0, Resnet -50, GoogleNet are meticulously detailed and analyzed in Fig.9, Fig.10, Fig.11, Fig.12 respectively.

A graph of a graph

Description automatically generatedA graph of a training loss

Description automatically generated

**Fig.9** Accuracy Plot (left) andLoss plot (right) of MobileNetV2.

A graph of a training and validation accuracy

Description automatically generated***A graph of training and validation loss

Description automatically generated***

**Fig.10** Accuracy Plot (left) andLoss plot (right) of EfficientNetB0.

A graph of a graph

Description automatically generatedA graph of a graph

Description automatically generated

**Fig.11** Accuracy Plot (left) andLoss plot (right) of ResNet-50.

A graph of a training and validation accuracy

Description automatically generatedA graph of training loss

Description automatically generated

**Fig.12** Accuracy Plot (left) andLoss plot (right) of GoogleNet (Inception V3)

The confusion matrix is important as it offers a detailed breakdown of a model's predictions, allowing for a deeper analysis of errors, model fine-tuning, and the selection of appropriate evaluation metrics. Fig.13, Fig.14, Fig.15, Fig.16, Fig.17 illustrates the confusion matrix obtained from MobileNetV2, EfficientNetB0, Resnet -50, GoogleNet and combined weight-centric mechanism respectively.

|  |  |
| --- | --- |
| A graph with numbers and a number of names  Description automatically generated with medium confidence  **Fig.13** Confusion Matrix using MobileNetV2. | A graph with numbers and a number of different colored squares  Description automatically generated with medium confidence  **Fig.14** Confusion Matrix using EfficientNetB0. |
| A graph with numbers and a number of different colored squares  Description automatically generated with medium confidence  **Fig.15** Confusion Matrix using ResNet-50. | A graph with blue squares and white text  Description automatically generated  **Fig.16** Confusion Matrix using GoogleNet |
| **Fig.17** Confusion Matrix obtained using combined weight-centric mechanism. | |

Precision-Recall (PR) and ROC curves offer insights beyond traditional accuracy metrics. They allow for a more nuanced evaluation of model performance, especially in situations involving imbalanced datasets, varying error costs, or a need to fine-tune classification thresholds. These curves aid in making informed decisions about model selection, threshold adjustment, and optimizing performance for specific applications. The precision-recall curve illustrates the trade-off between precision and recall. The ROC curve showcases the model's true positive rate against the false positive rate.

A graph with text and numbers

Description automatically generated with medium confidenceA graph of a graph showing the results of a graph

Description automatically generated with medium confidence

**Fig.18** ROC curve (left) andPR curve (right) of MobileNetV2.

A graph with text and numbers

Description automatically generated with medium confidenceA graph with numbers and lines

Description automatically generated with medium confidence

**Fig.19** ROC curve (left) andPR curve (right) of EfficientNetB0.

A graph with text and numbers

Description automatically generated with medium confidenceA graph with numbers and lines

Description automatically generated with medium confidence

**Fig.20** ROC curve (left) andPR curve (right) of ResNet-50.

A graph with text and numbers

Description automatically generated with medium confidenceA graph with numbers and lines

Description automatically generated with medium confidence

**Fig.21** ROC curve (left) andPR curve (right) of GoogleNet (Inception V3).

Metrics like F1 score, precision, and recall are crucial in this paper as they provide a balanced assessment of a model's performance in situations involving imbalanced datasets or varying costs of errors. Unlike accuracy, these metrics help evaluate the model's ability to correctly classify both positive and negative instances and distinguish between types of errors, making them valuable in real-world applications.

***Table 3****. Performance Metrics of Deep Learning Models and Weight-Centric Mechanism*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Accuracy | F1 Score | Precision | Recall |
| MobileNetV2 | 0.93 | 0.9302 | 0.9325 | 0.93 |
| Efficient Net B0 | 0.95 | 0.9502 | 0.9516 | 0.95 |
| ResNet 50 | 0.93 | 0.9305 | 0.9404 | 0.93 |
| GoogleNet (Inception V3) | 0.92 | 0.9204 | 0.9292 | 0.92 |
| Weight centric mechanism | 0.96 | 0.9602 | 0.9629 | 0.96 |

Table.3 provides a concise summary of the performance metrics for various deep learning models and the proposed weight-centric mechanism. MobileNetV2, EfficientNet B0, ResNet-50, and GoogleNet (Inception V3) demonstrate strong performance with high accuracy, F1 scores, precision, and recall, ranging from 0.92 to 0.95. The Weight Centric Mechanism stands out with exceptional performance, achieving the highest accuracy of 0.96 and superior F1 score, precision, and recall, all around 0.96, showcasing its effectiveness in enhancing classification results as shown in Fig.22. In summary, the Weight Centric Mechanism surpasses individual model performance, highlighting its significance in improving accuracy and reliability.

**Fig.22** Comparison of MobileNetV2, EfficientNetB0, ResNet 50, GoogleNet (Inception V3), and Weight centric mechanism

1. **CONCLUSION**

In conclusion, the "Mythic Vision" software represents a significant advancement at the crossroads of modern deep learning and India's rich mythology. Its impact on cultural exploration will be remarkable as tourists will use this tool, they will actively engage with Indian mythology, gaining a deeper understanding of the culture. This infusion of knowledge will help the travelers form meaningful connections with the culture they're exploring. They move beyond surface-level experiences and gain insights into the symbolism, stories, and traditions that shape India's cultural landscape. This deeper engagement ensures that tourists leave not only with memories but with a genuine appreciation of the heritage they've encountered.

The "Mythic Vision" software is more than just a tool; it's a catalyst for an educational tourism experience. It's set to enhance India's global appeal as a tourist destination, offering a richer and more profound journey for all. The fusion of deep learning and cultural exploration promises to elevate India's tourism industry, providing a more enlightening experience for visitors.

**References**

1. Huang., Liao., Shiau., Tseng.: Traditional chinese God image dataset: A glimpse of chinese culture (February 2023)
2. Zhang., Jing., Yang.: A Weighted Voting Classifier Based on Differential Evolution (May 2014)
3. <https://rishihood.edu.in/artificial-intelligence-and-robotics-a-journey-through-ancient-indian-texts/> Accessed on 08th November 2023
4. Aggarwal., Sahoo., Bansal., Sarangi.: Image Classification using Deep Learning: A Comparative Study of VGG-16, InceptionV3 and EfficientNet B7 Models (July 2023)
5. <https://github.com/Adinp1213/Mythic_Vision> Created in September 2023
6. Spagnolo., Perri., Frustaci., Corsonello.: Connected Component Analysis for Traffic Sign Recognition Embedded Processing Systems (December 2018)
7. Armi., Ershad.: TEXTURE IMAGE ANALYSIS AND TEXTURE CLASSIFICATION METHODS - A REVIEW (2019)
8. Lalaoui., Mohamadi.: A comparative study of Image Region-Based Segmentation Algorithms (July 2013)
9. Natarajan V., Kumar M., Tamizhazhagan V.: Text Region Detection and Recognition in Natural Scene Images Using MSER and Convolutional Neural Network (November 2020)
10. Chen., Li., Bai., Yang., Jiang., Miao.: Review of Image Classification Algorithms Based on Convolutional Neural Networks (November 2021)
11. Kovalev., Liauchuk., Kalinovsky.: A comparison of conventional and deep learning methods of image classification on a database of chest radiographs. (June 2017)
12. Sharma., Phonsa.: Image Classification Using CNN (April 2021)
13. Tulasi., Kumar Kalluri.: Deep Learning and Transfer Learning Approaches for Image Classification (February 2019)
14. Sandler., Howard., Zhu., Zhmoginov., Chen.: MobileNetV2: Inverted Residuals and Linear Bottlenecks (Revised March 2019)
15. Tan., Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (Revised September 2020)
16. He., Zhang., Ren., Sun.: Deep Residual Learning for Image Recognition (December 2015)
17. Szegedy., Vanhoucke., Loffe., Shlens., Wolna.: Rethinking the Inception Architecture for Computer Vision (Revised December 2015)