CAPSTONE FINAL REVIEW

MYTHIC VISION

ILLUMINATING GODS THROUGH DEEP LEARNING

20BCE7373: AVIRAL SINGH
20BCE7013: PATIL ADITYA NITIN
20BCR7083: GITESH PRASHANT BHAVSAR
20BCI7111: KANAKAGIRI SUJAY ASHRITH

GUIDED BY PROF.TAUSEEF KHAN

INDEX

STAGES OF PROJECT COMPLETION

- **Dataset Creation:**

Involves the meticulous process of sourcing, selecting, and curating images of Indian deities to construct custom datasets.

- **Model Selection:**

Focuses on the selection of appropriate deep learning models, including , MobileNet, ResNet, EfficientNet, and GoogleNet, based on their suitability for the project's objectives.

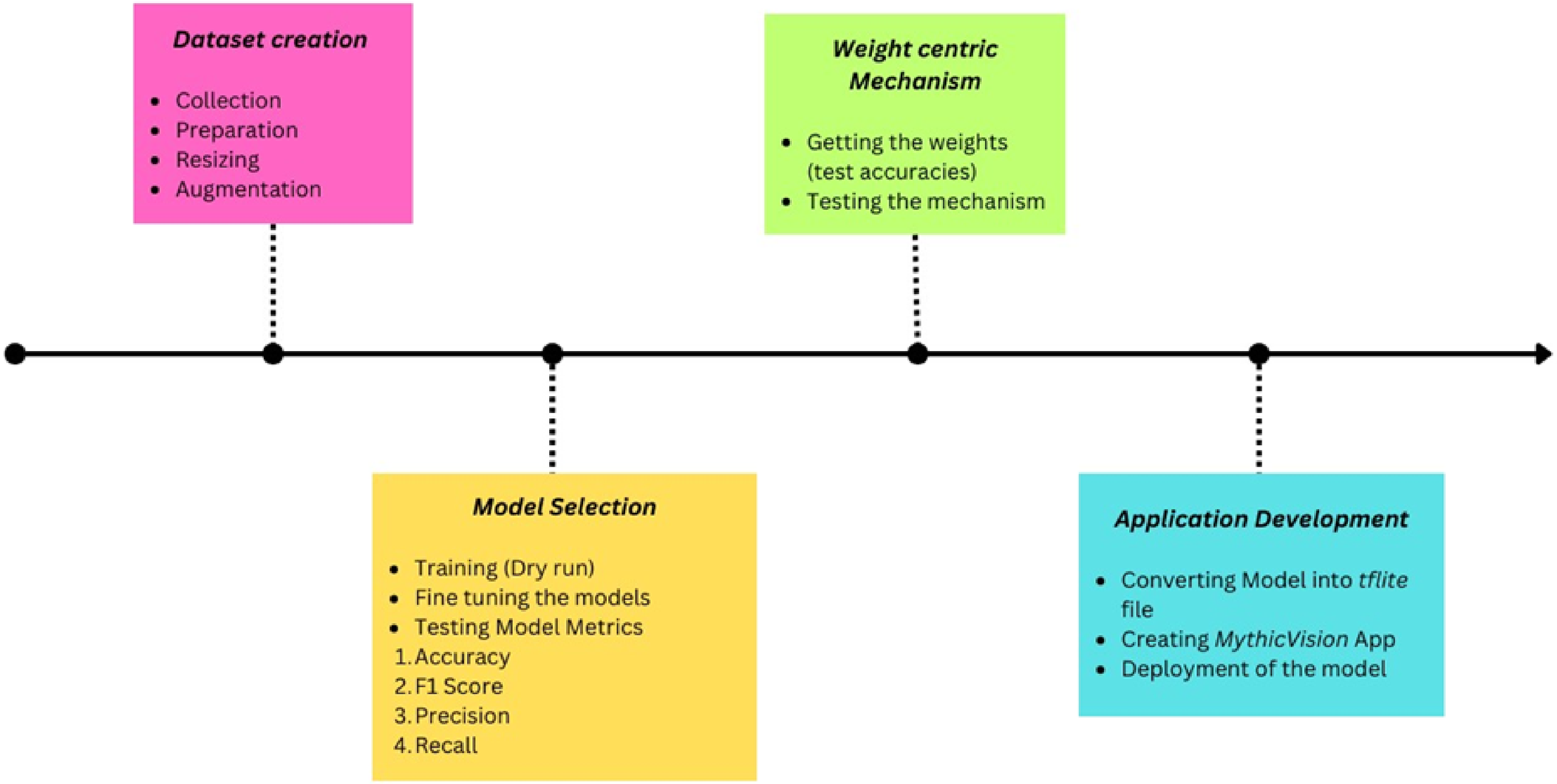
- **Weighted Decision-Based Approach**:

Discusses the prioritization of a weighted decision-based approach over individual model accuracy to enhance the accuracy and reliability of our results.

- **Application Development:**

Encompasses the creation of an intuitive and user-friendly application that allows users to interact with and benefit from the project's findings.

STAGES OF PROJECT COMPLETION



DATASET CREATION

- 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,i.e 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 60-degree rotation, effectively generating a new image. This approach significantly expanded the dataset, with each original image evolving into a collection of five distinct images.
- Consequently, a total of 1000 images were allocated for each deity, comprising the training phase for the subsequent image classification task.

DATASET CREATION

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

Brief outline of the experimental dataset along with its particulars.

DATASET CREATION

Balaji
(500 images)

filtering

Balaji
(300 images)

resizing and padding

Balaji
(300 images)

Splitting the dataset

Training (200 images)

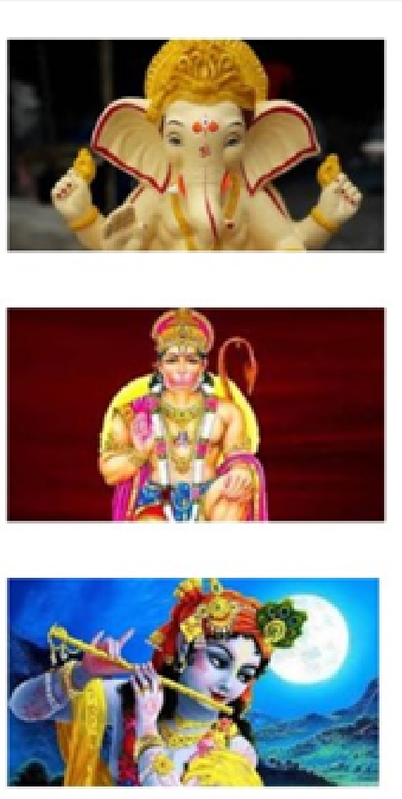
Validation (80 images)

Testing (20 images)

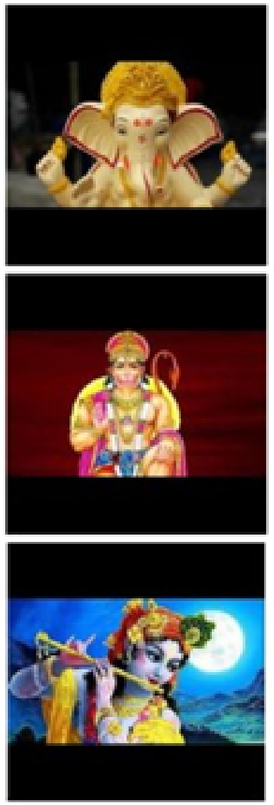
Data Augmentation

Balaji Augmented training
(1000 images)

DATASET CREATION



Original image



224 x 224 pixel padded image

Shearing | translation | Rotation 20 | Rotation 180 | Flipping

5 data augmented images

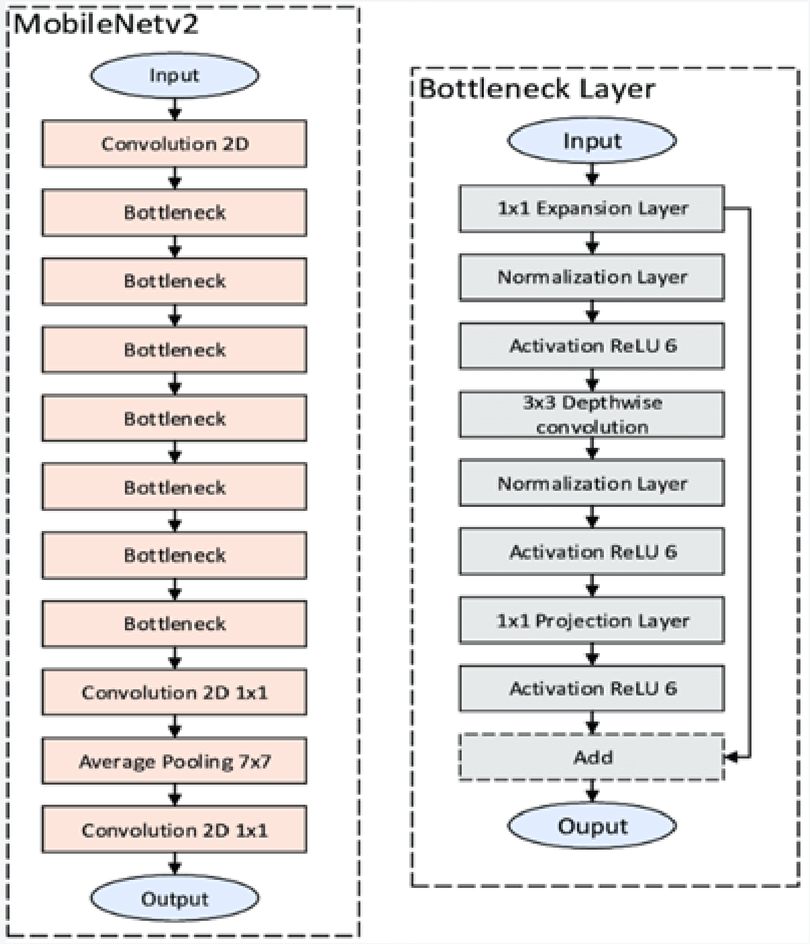
MODELS USED

- (a.) MobileNet
- MobileNet is a convolutional neural network architecture designed for efficient on-device vision applications.
- It is known for its lightweight and fast inference speed, making it suitable for resource-constrained environments like mobile devices and embedded systems.
- MobileNet uses depth wise separable convolutions to reduce model size and computational complexity while maintaining good accuracy. Following is the resultant test accuracy obtained after training and testing the model on our in house dataset.

```
Test Loss: 0.3468
Test Accuracy: 0.9300
Weighted F1 Score: 93.02%
Weighted Precision: 93.25%
Weighted Recall: 93.00%
```

MODELS USED

MobilenetV2



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

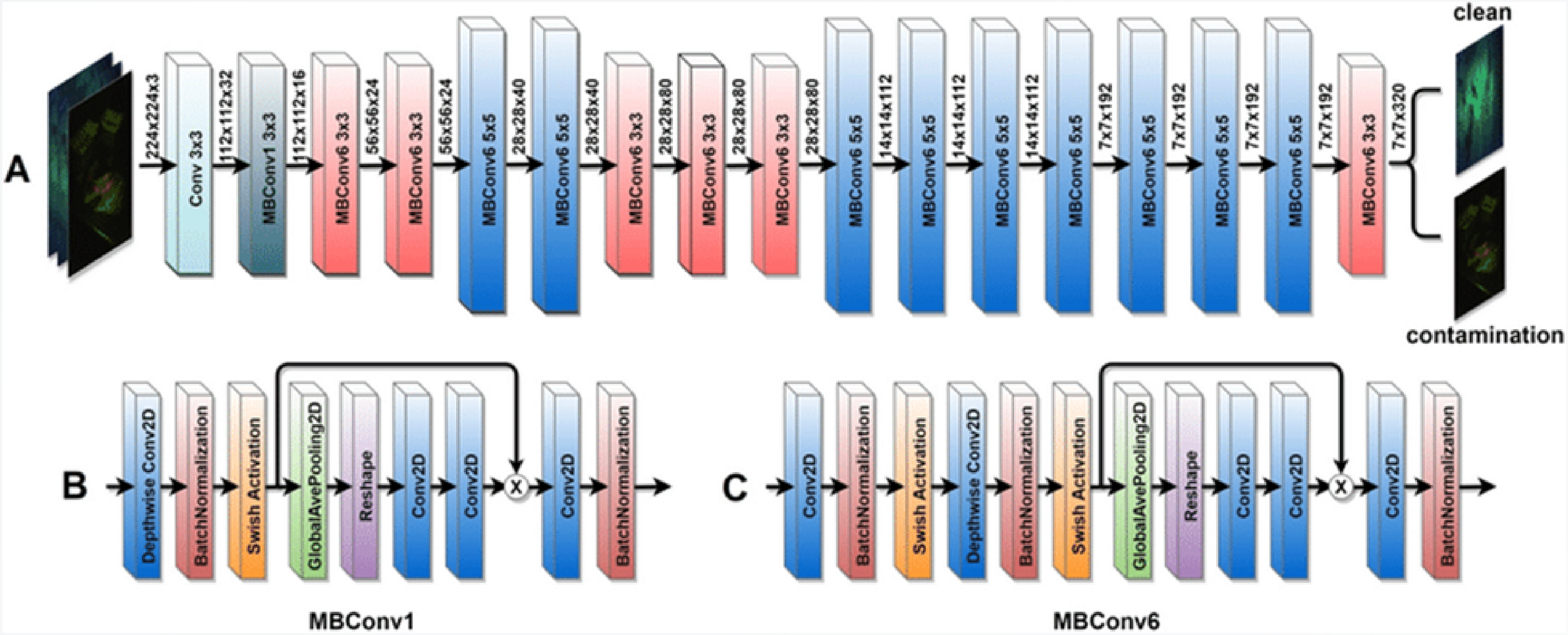
MODELS USED

- (b.) EfficientNetB0
- EfficientNet is a family of neural network architectures that systematically scales the model's depth, width, and resolution to achieve better performance while maintaining efficiency.
- It uses a compound scaling method to determine the optimal model size based on the available resources.
- EfficientNet models have demonstrated strong performance on various image classification tasks with relatively few parameters. Following is the resultant test accuracy obtained after training and testing the model on our in house dataset.

```
Test Loss: 0.2084
Test Accuracy: 0.9500
Weighted F1 Score: 95.02%
Weighted Precision: 95.16%
Weighted Recall: 95.00%
```

MODELS USED

EfficientnetB0

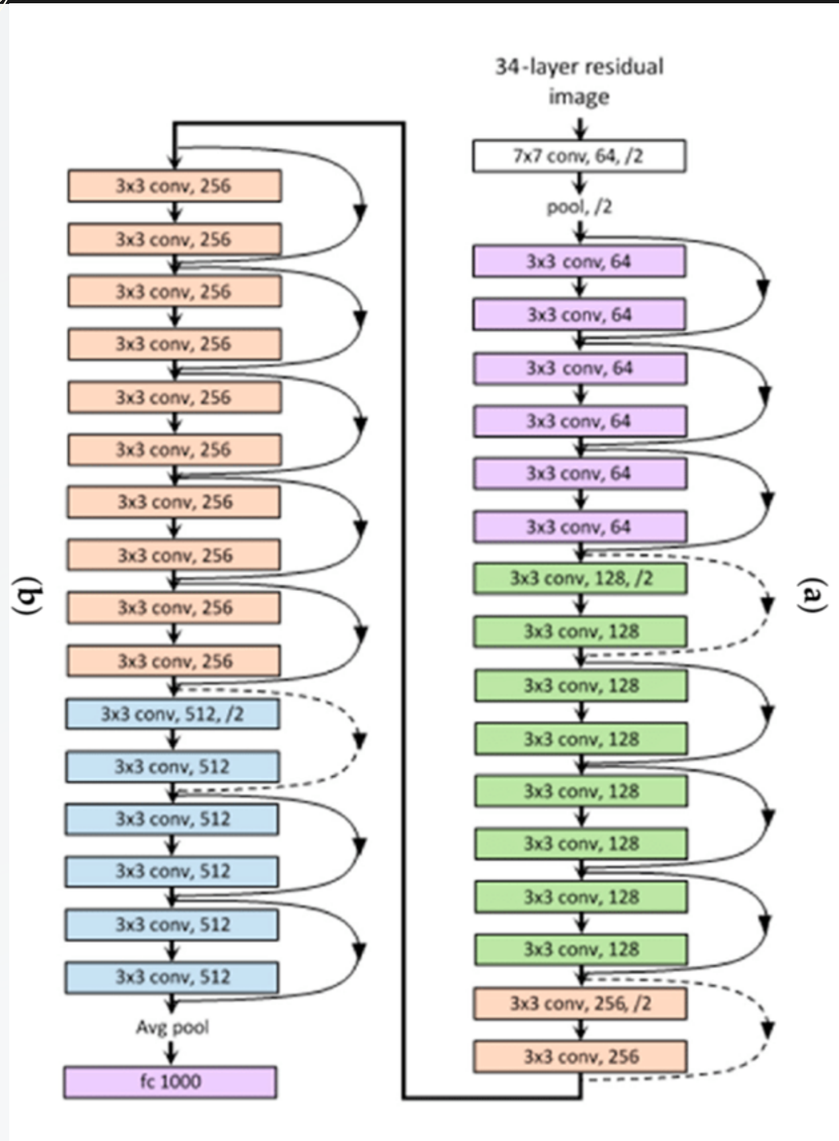


MODELS USED

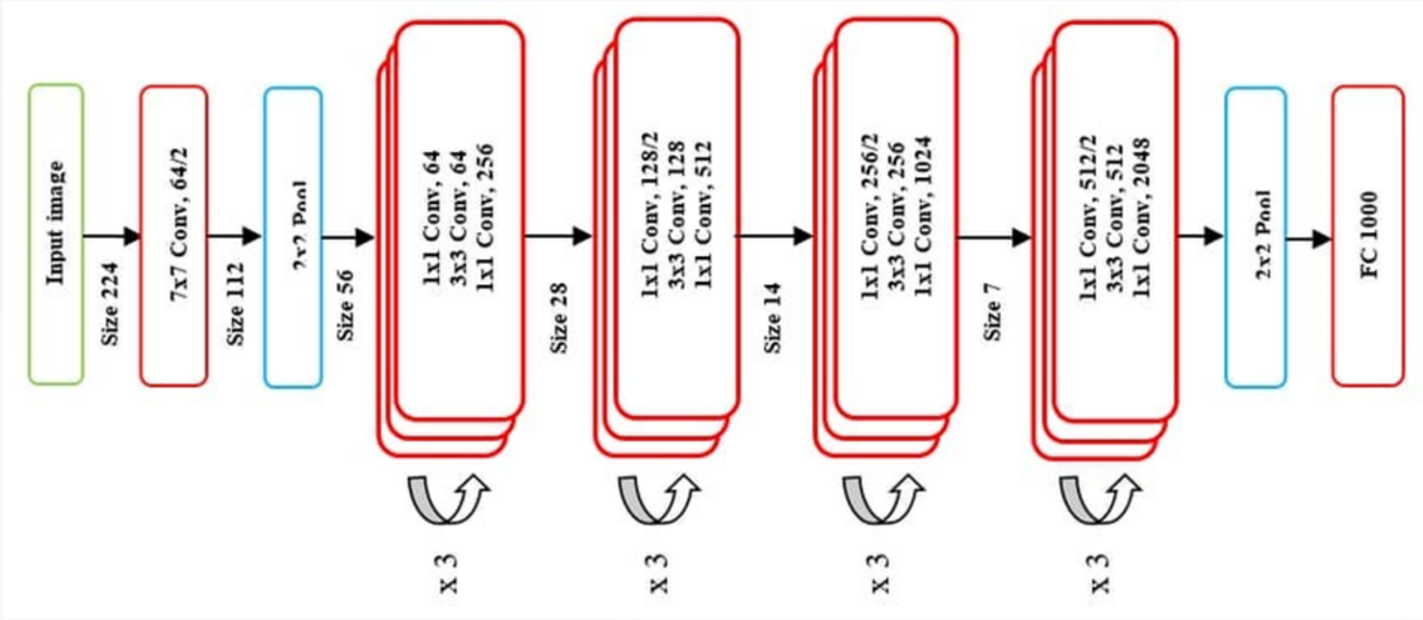
- (c.) ResNet-50
- ResNet is a deep convolutional neural network architecture that introduced the concept of residual blocks.
- It addresses the vanishing gradient problem by allowing the network to skip connections, or shortcuts, between layers. These shortcuts enable the network to learn residual functions, which are added to the input of a layer, making it easier to train very deep networks.
- ResNet has been widely adopted in various computer vision tasks. Following is the resultant test accuracy obtained after training and testing the model on our in house dataset.

```
Test Loss: 0.3468
Test Accuracy: 0.9300
Weighted F1 Score: 93.02%
Weighted Precision: 93.25%
Weighted Recall: 93.00%
```

MODELS USED



ResNet - 50



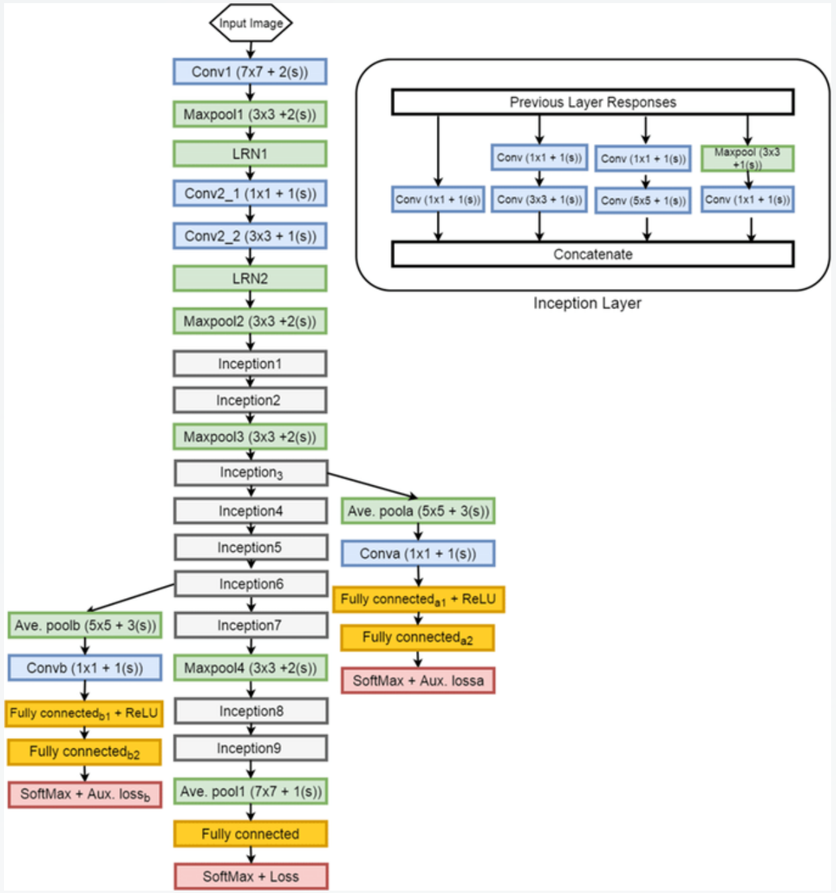
MODELS USED

- (d.) Google Net (Inception V3)
- GoogleNet, also known as Inception-v3, is a deep convolutional neural network architecture developed by Google. It is part of the Inception family of models.
- GoogleNet is known for its utilization of inception modules, which consist of multiple parallel convolutional filters of different sizes.
- These modules capture features at various scales and improve the network's ability to handle diverse features in images. Following is the resultant test accuracy obtained after training and testing the model on our in house dataset.

```
Test Loss: 0.3220
Test Accuracy: 0.9200
Weighted F1 Score: 92.04%
Weighted Precision: 92.92%
Weighted Recall: 92.00%
```

MODELS USED

GoogleNet (InceptionV3)



type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

WEIGHT-CENTRIC MECHANISM

- 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.

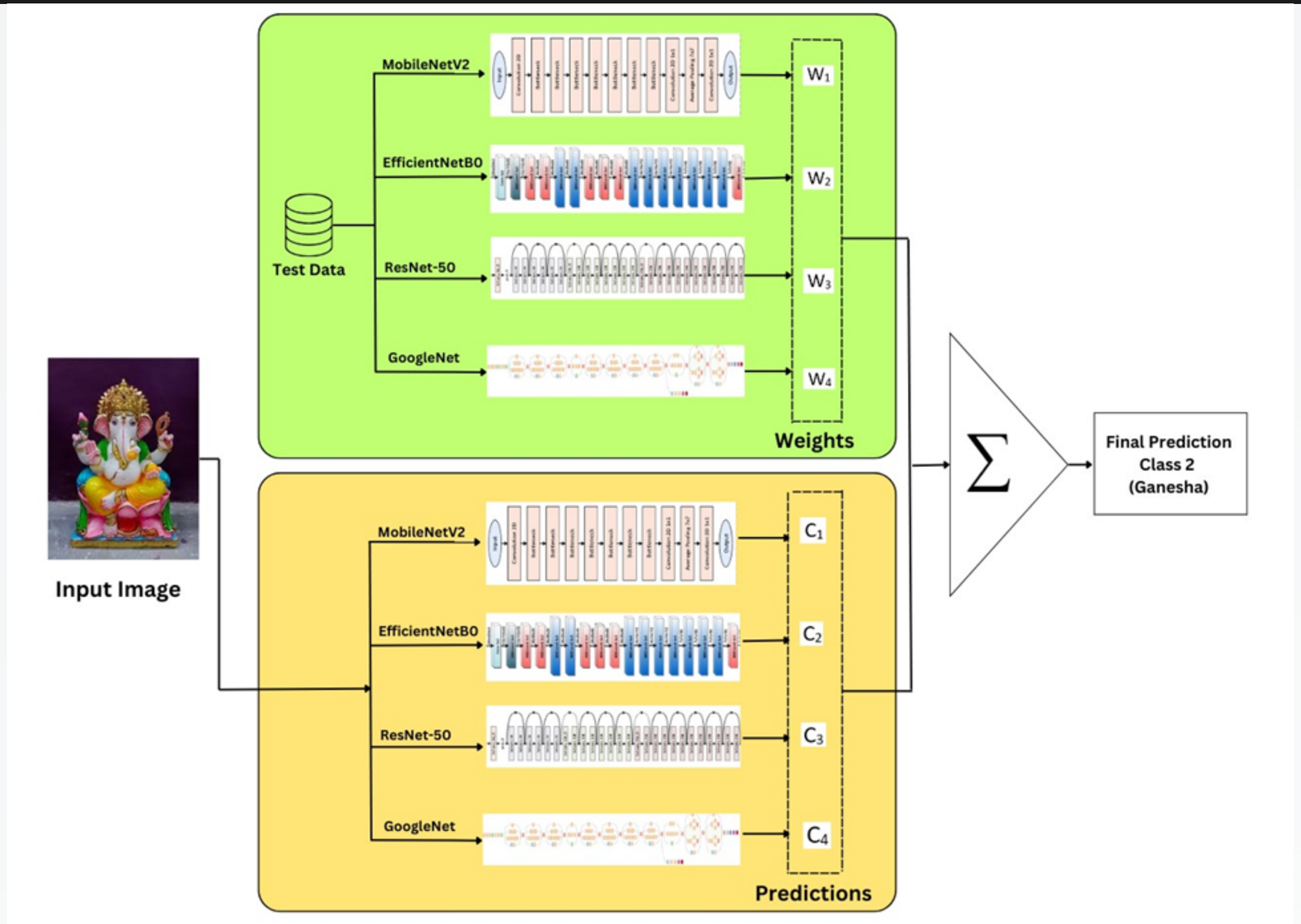
WEIGHT-CENTRIC MECHANISM

- (a.) 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.

- (b.) 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.

- (c.) 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.

WEIGHT-CENTRIC MECHANISM



WEIGHT-CENTRIC MECHANISM

Scenario	Conventional majority-based approach	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 don't affect the final decision. Example: EfficientNet (95% accuracy) (predicts Ganesha), MobileNet (20%), Resnet (30%), GoogleNet (10%), final prediction: Ganesha.

APPLICATION DEVELOPMENT

- In this section, the operational flow of the developed application is being illustrated. The final model, which was created after the key stages of dataset creation, model selection, and the weight-based approach, is seamlessly integrated into an application.
- 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 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 a comprehensive information related to the recognized deity.

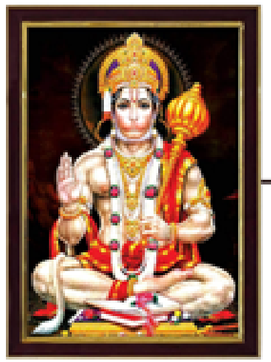
APPLICATION DEVELOPMENT

MobileNetV2

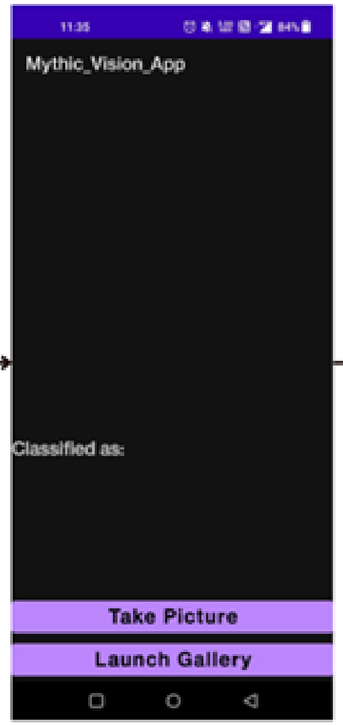
EfficientNetB0

ResNet-50

GoogleNet

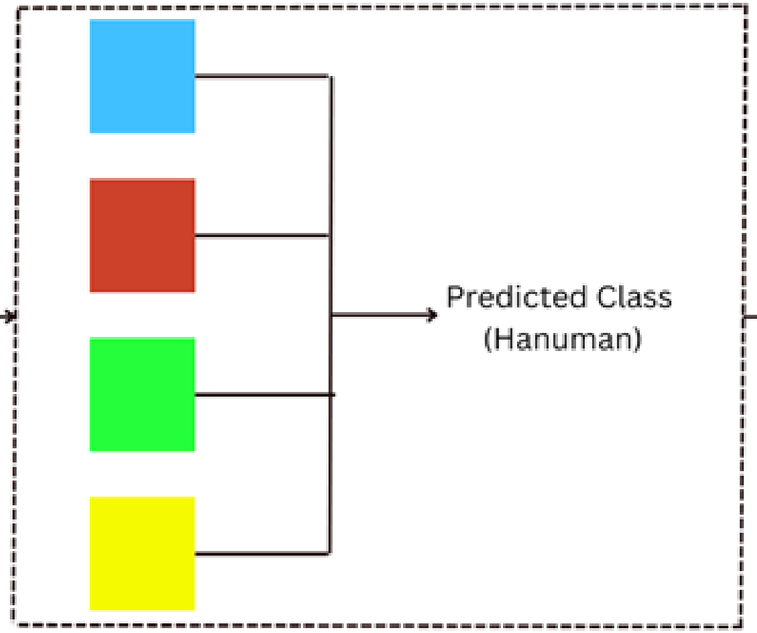


Input Image



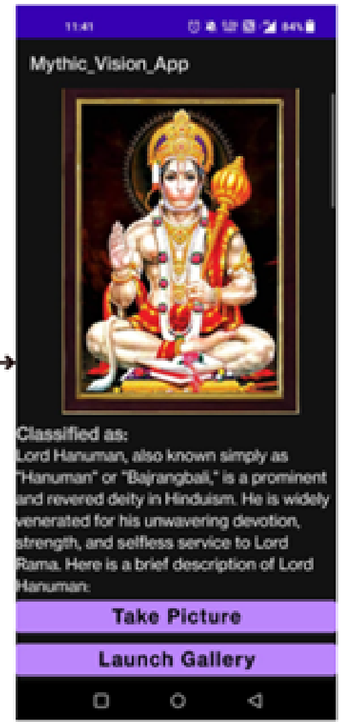
Capture Image

Weight - Based Mechanism



Predicted Class
(Hanuman)

Feed it in our Model Mechanism



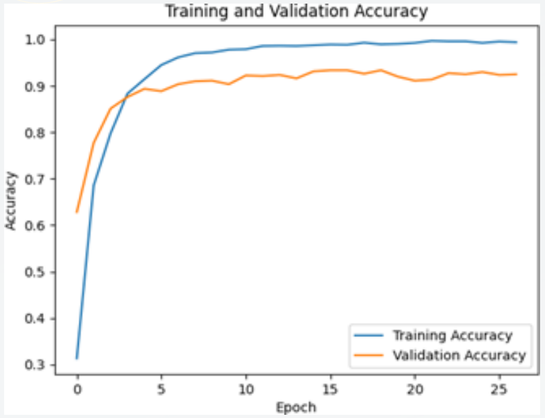
Display the Predicted Class

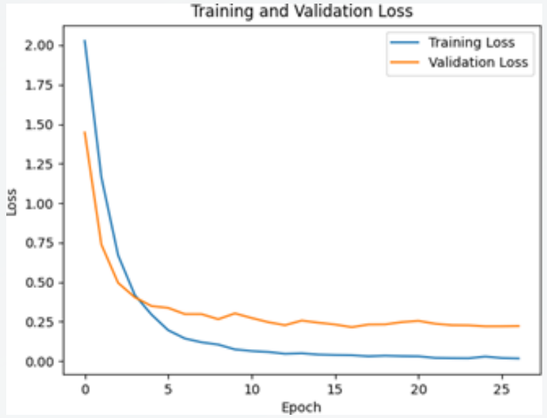
RESULTS AND DISCUSSIONS

Employed Deep Models	CA	F1 Score	P	R	Error
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 mechanism	0.96	0.9602	0.9629	0.96	0.04

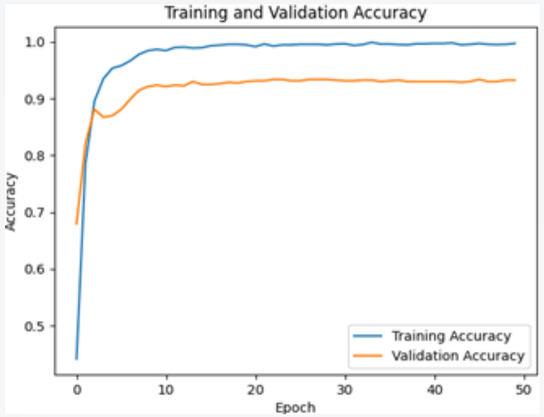
Performance Metrics of Deep Learning Models and Weight-Centric Mechanism

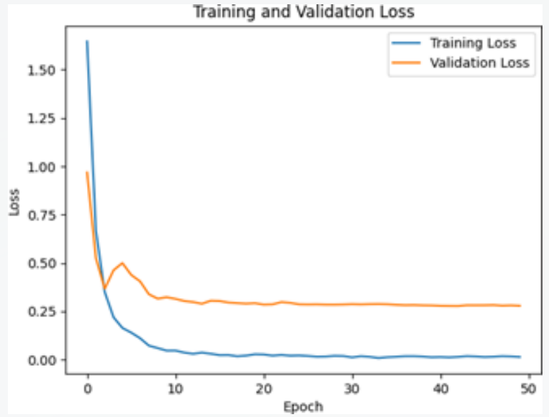
ACCURACY AND LOSS PLOTS



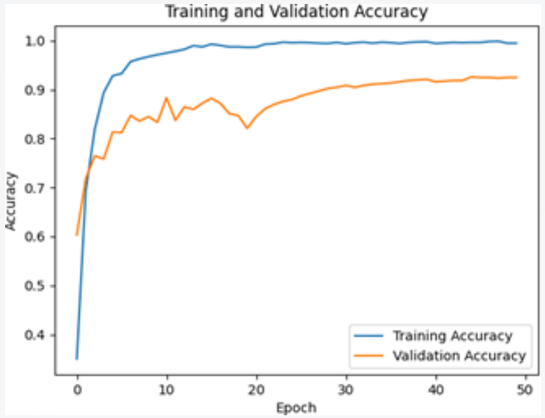


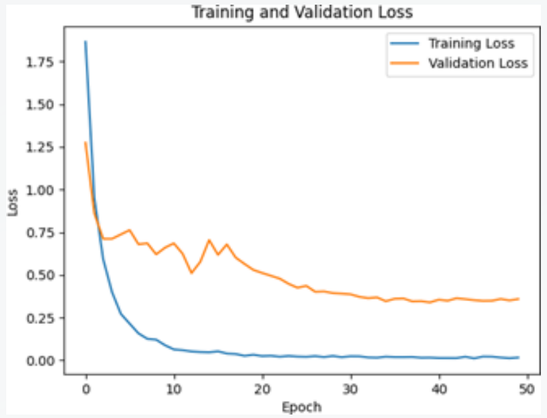
MobilenetV2



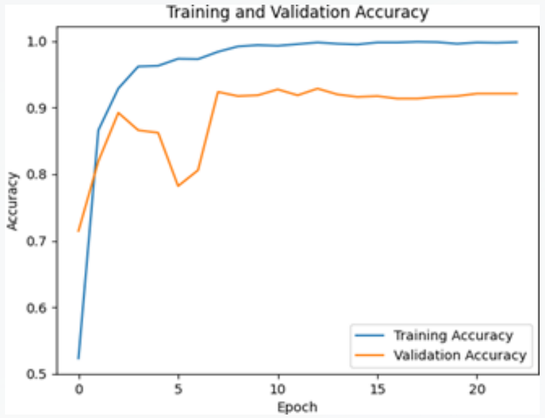


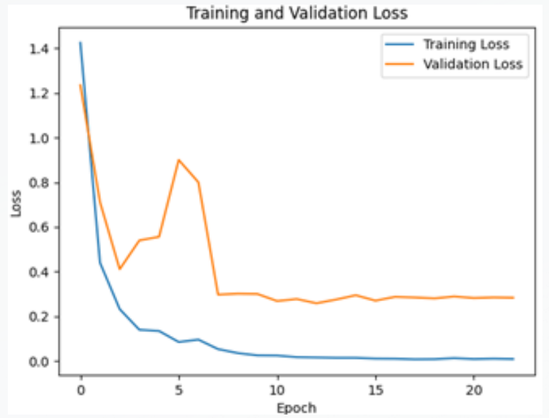
EfficientnetB0





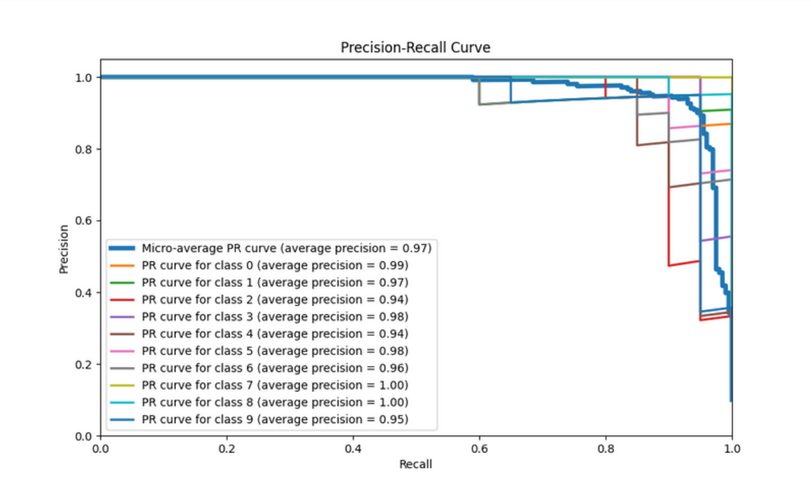
Resnet-50



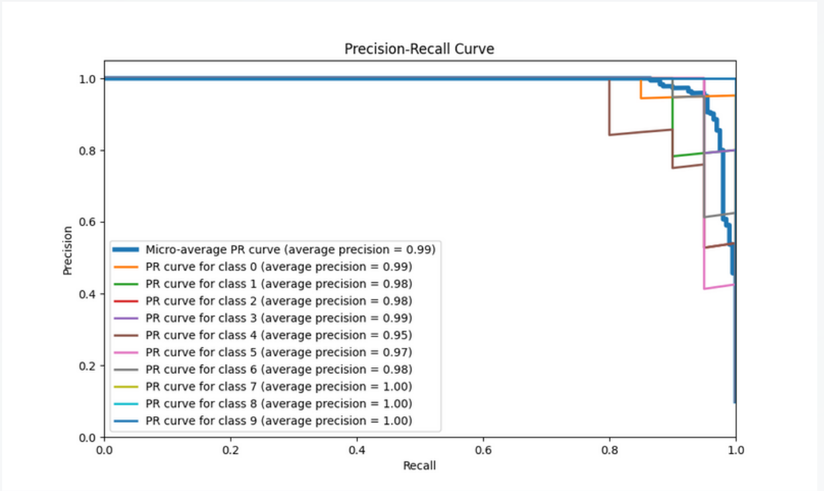


Googlenet (Inception V3)

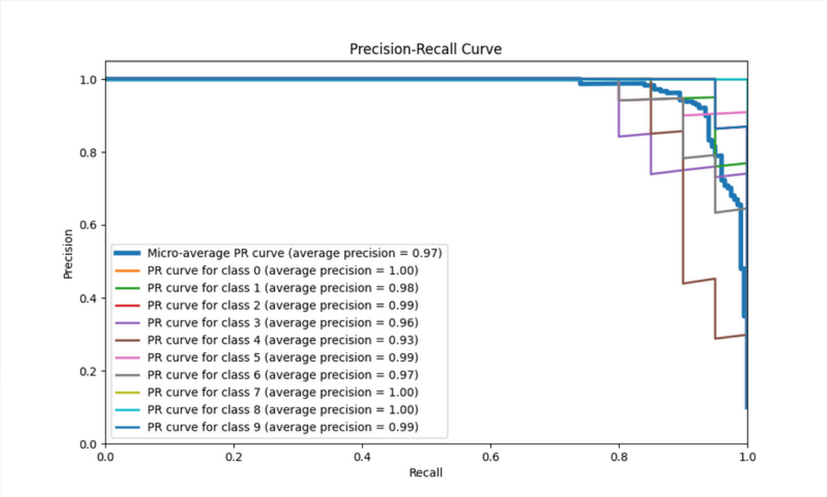
PRECISION-RECALL (PR) CURVES



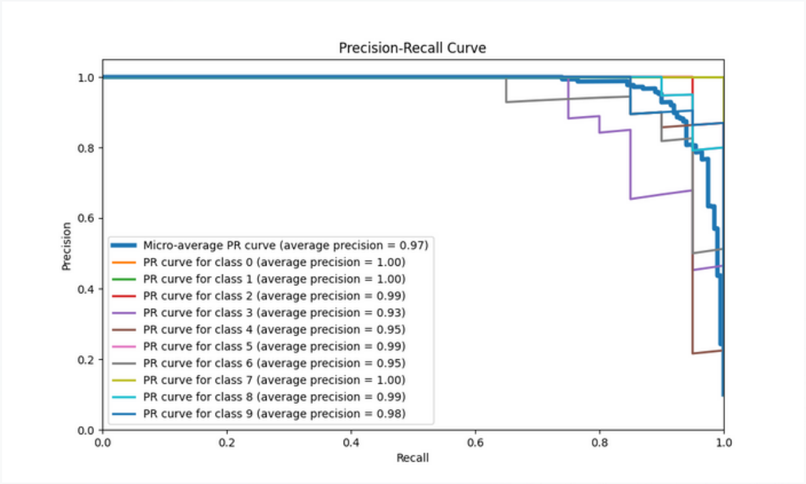
MobilenetV2



EfficientnetB0

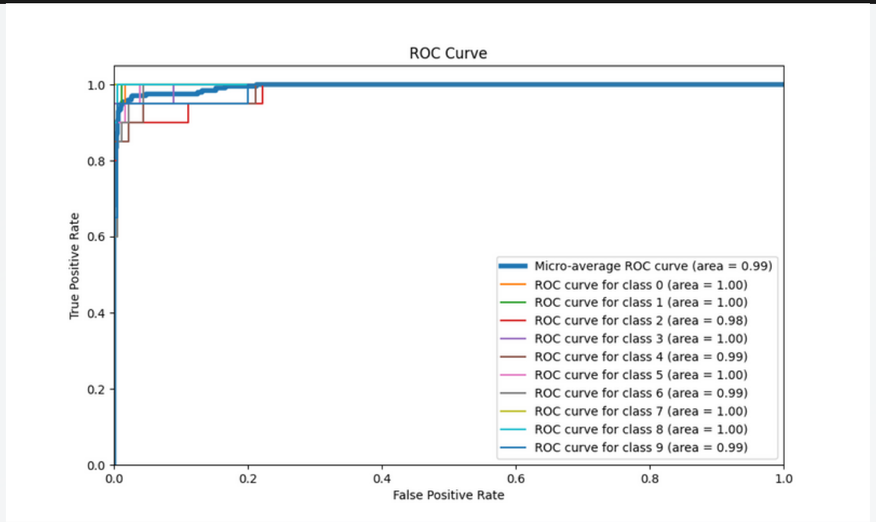


Resnet-50

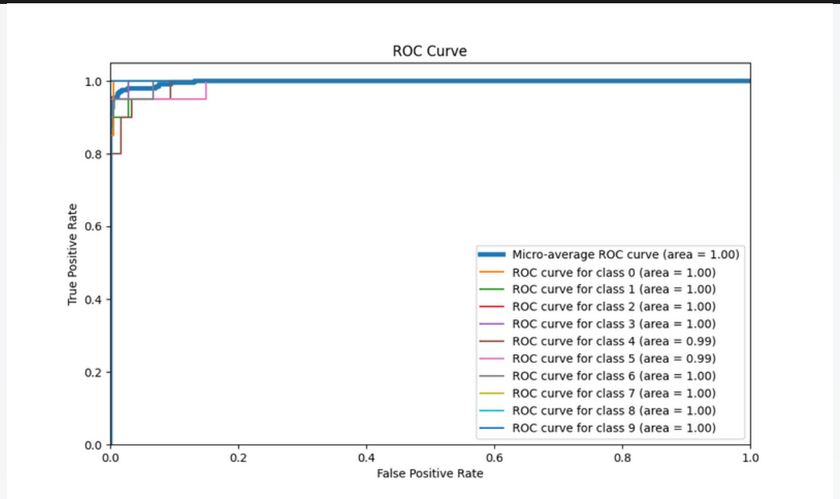


Googlenet (Inception V3)

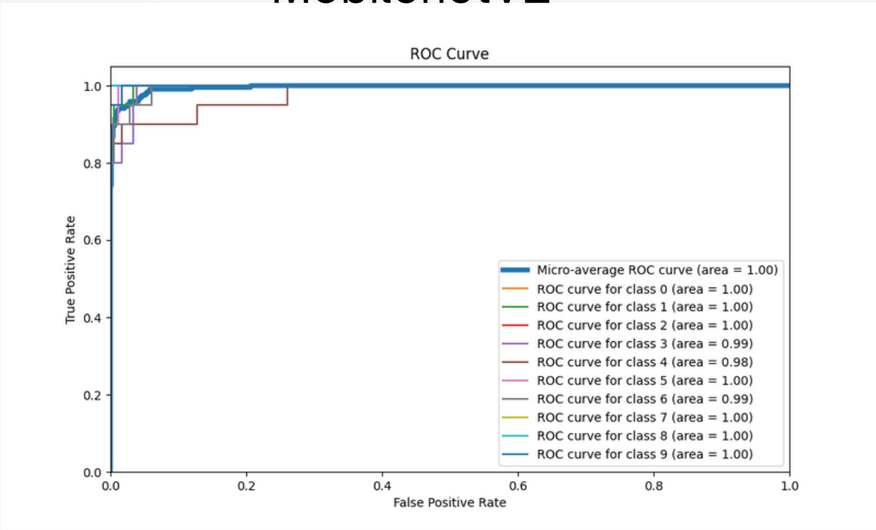
RECEIVER OPERATING CHARACTERISTIC CURVE (ROC) CURVES



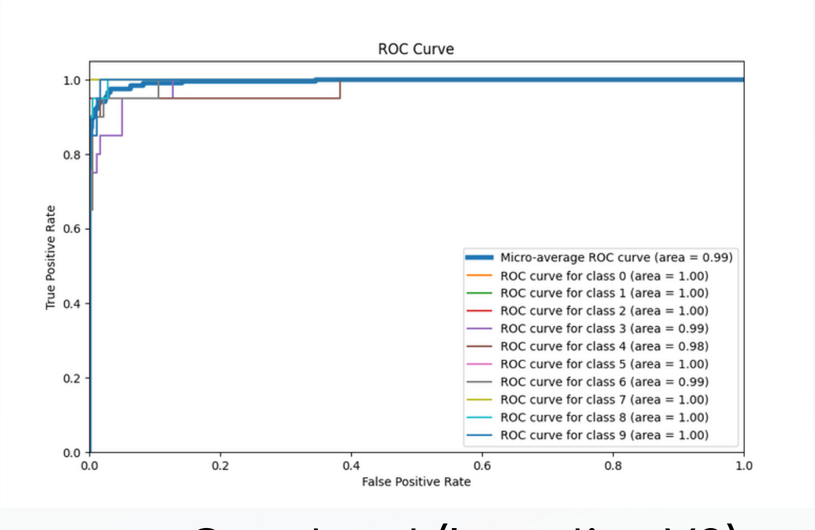
MobilenetV2



EfficientnetB0

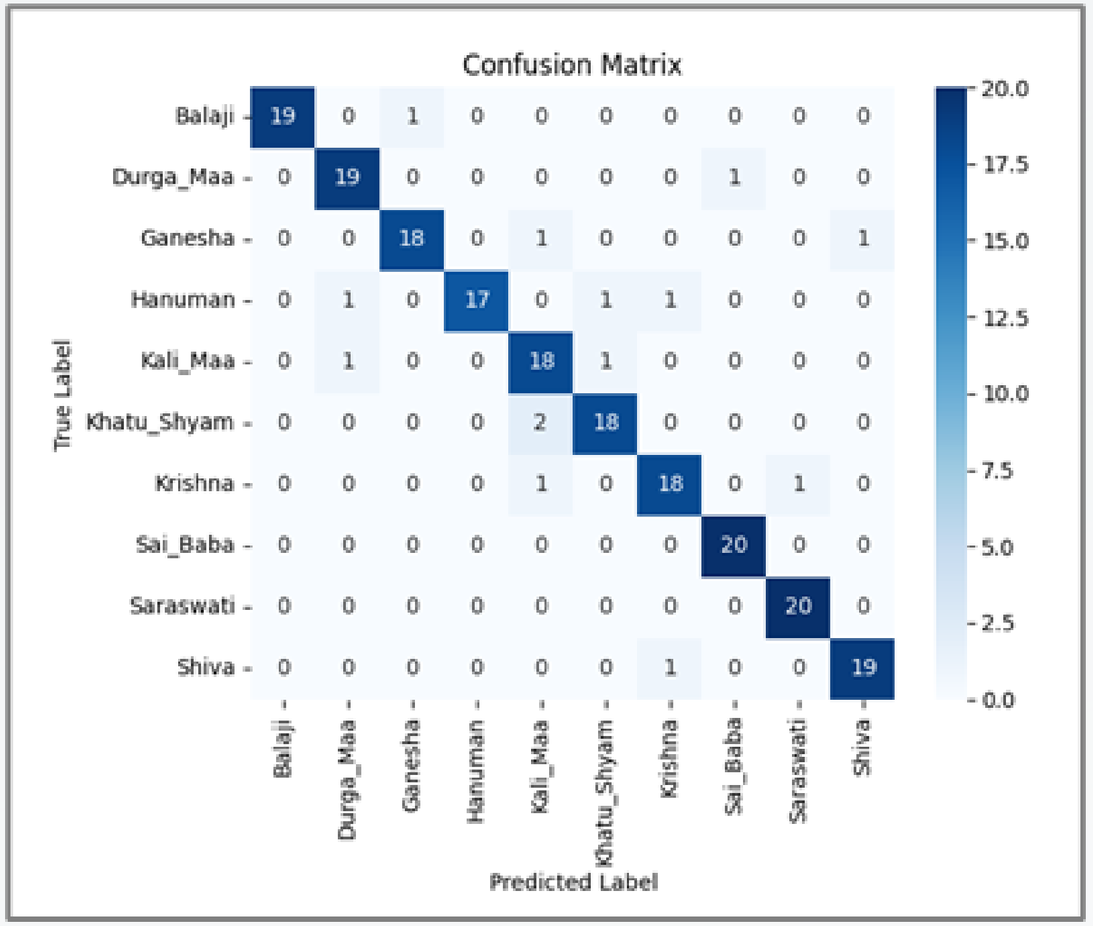


Resnet-50

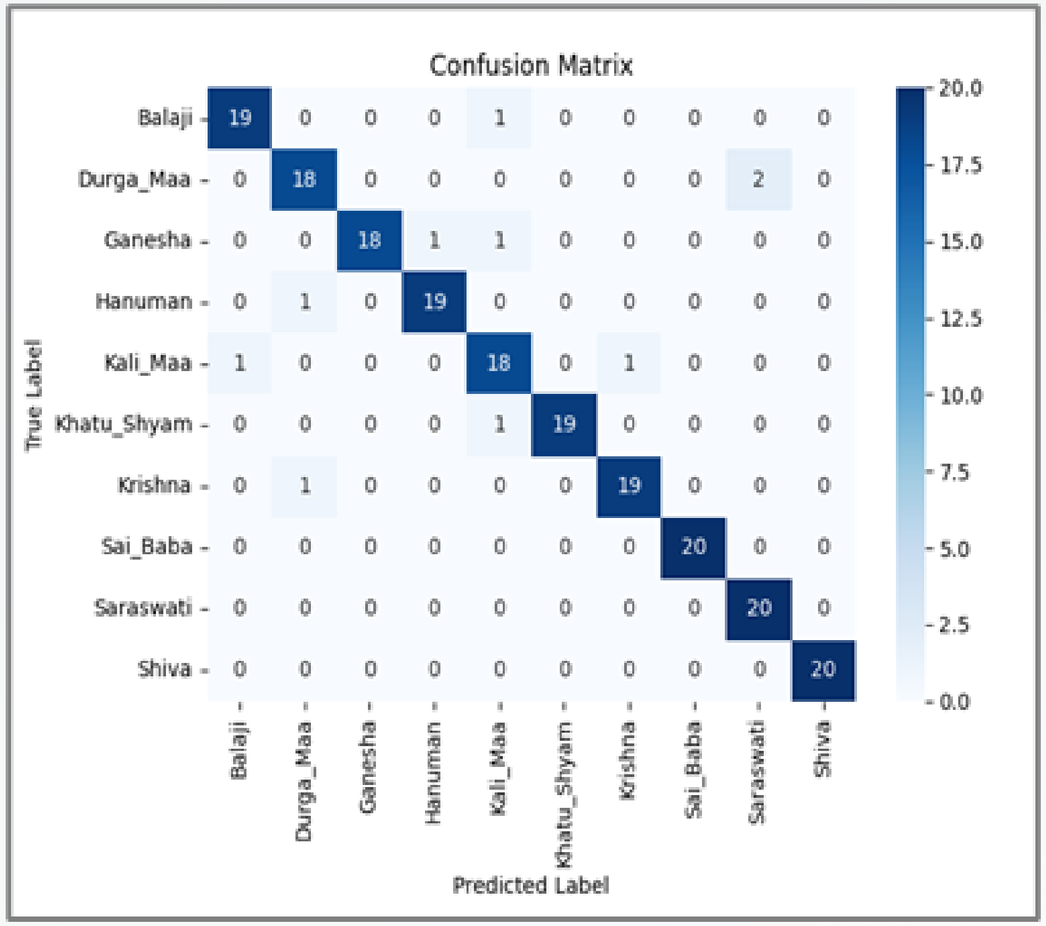


Googlenet (Inception V3)

CONFUSION MATRIX

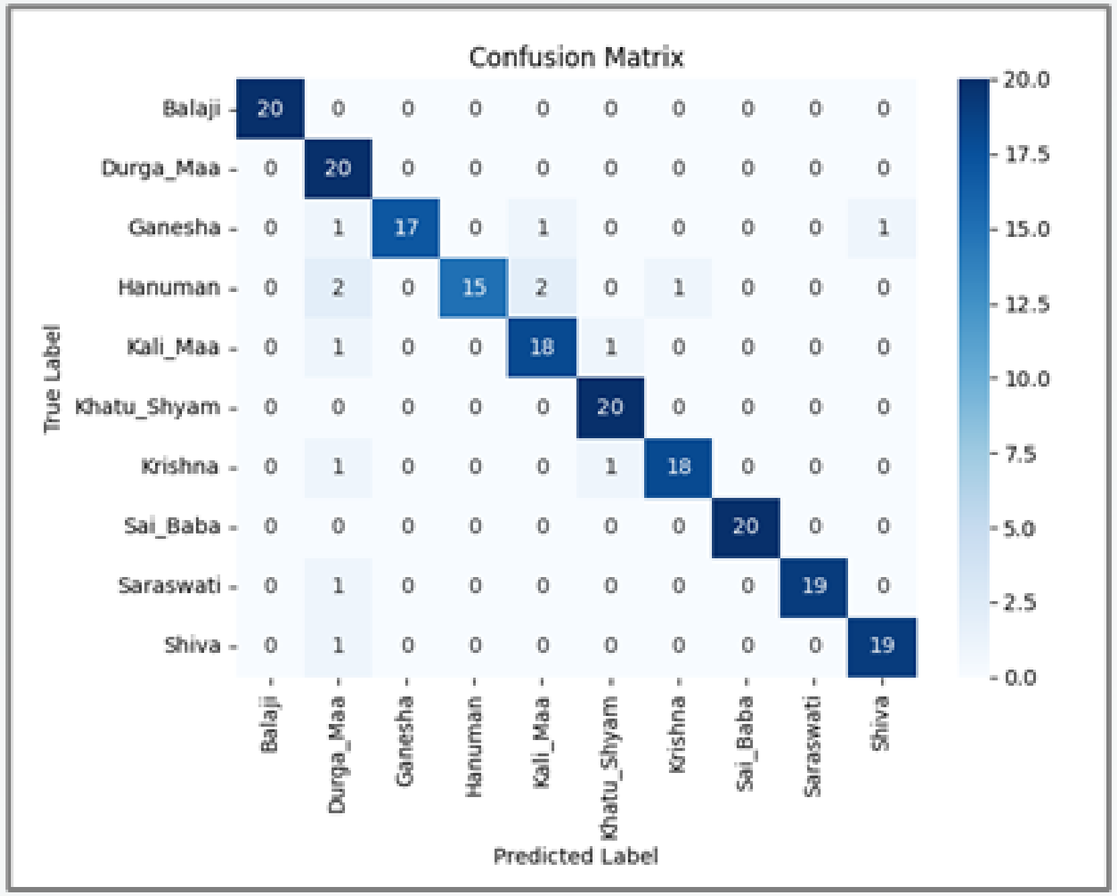


MobilenetV2

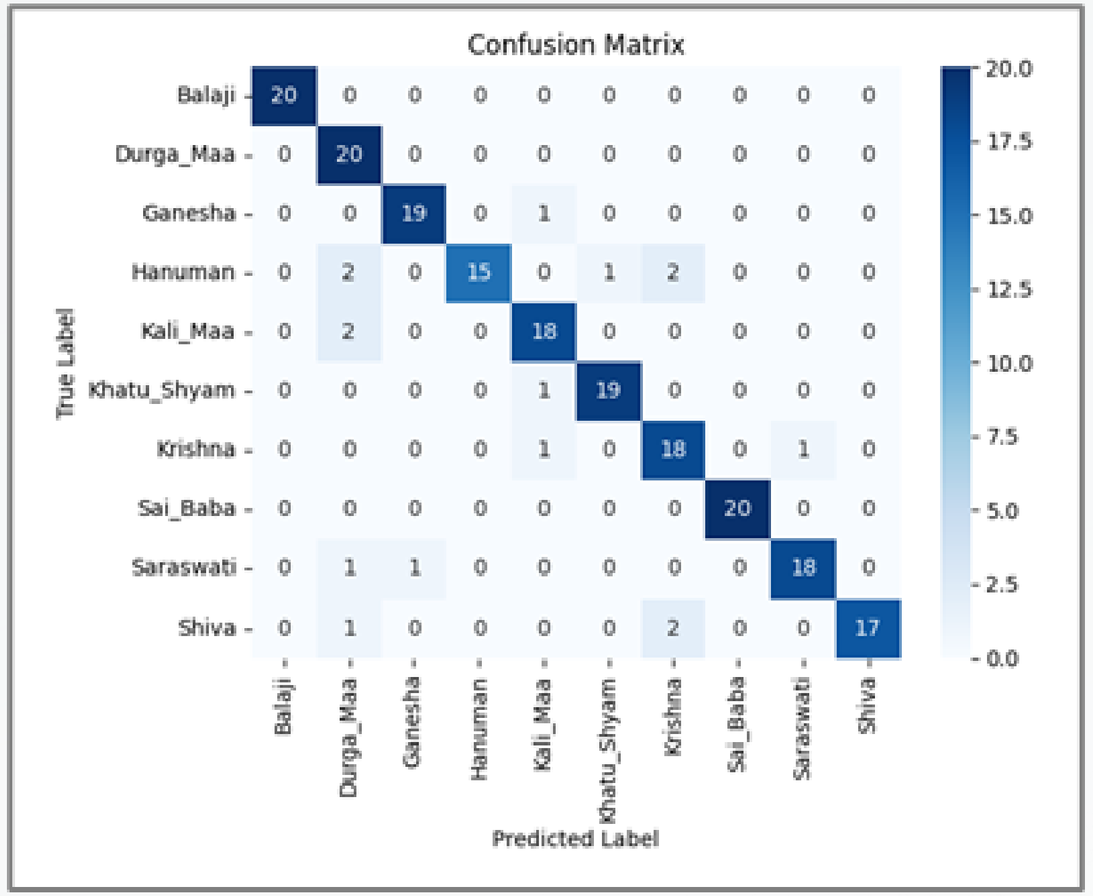


EfficientnetB0

CONFUSION MATRIX

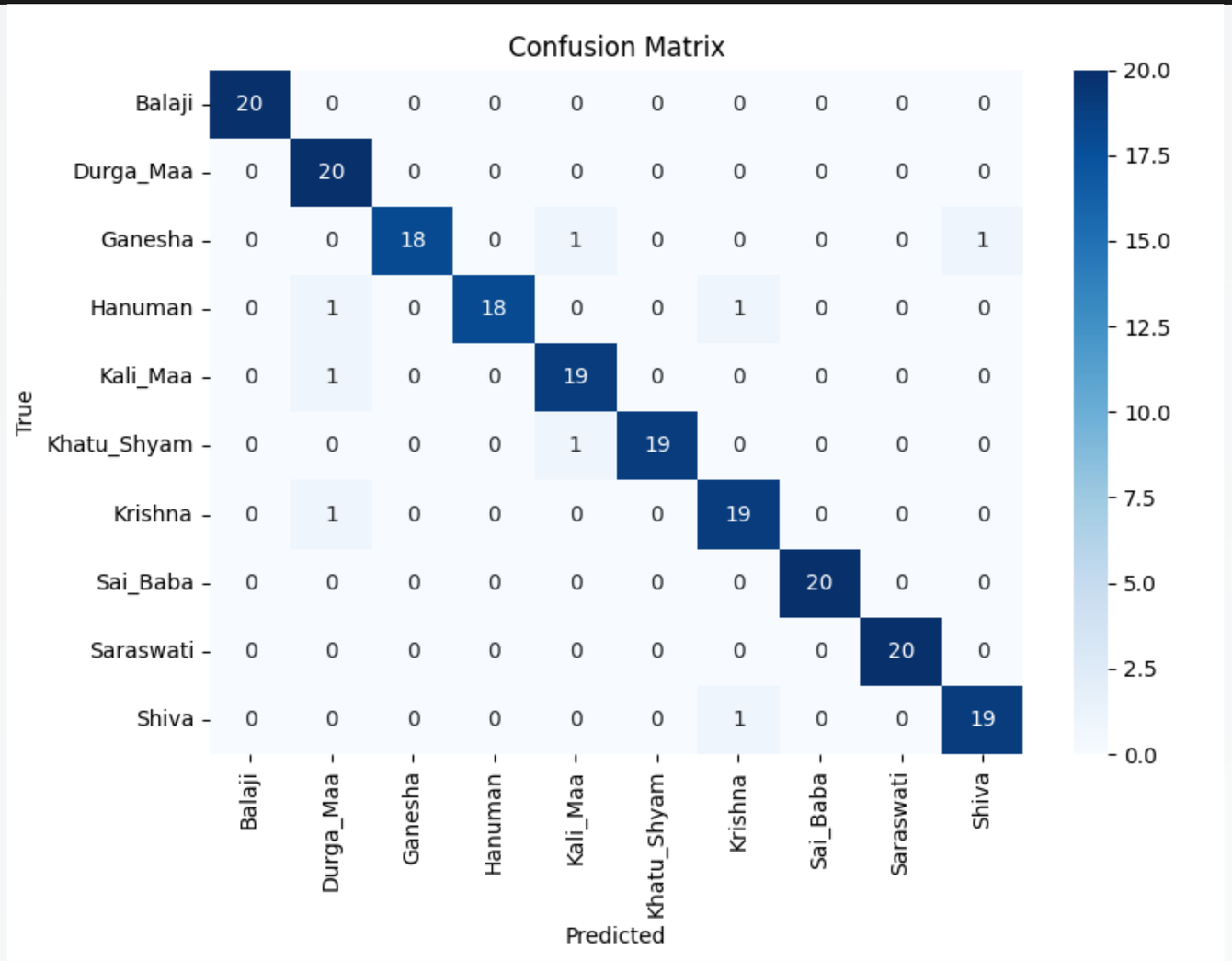


Resnet-50



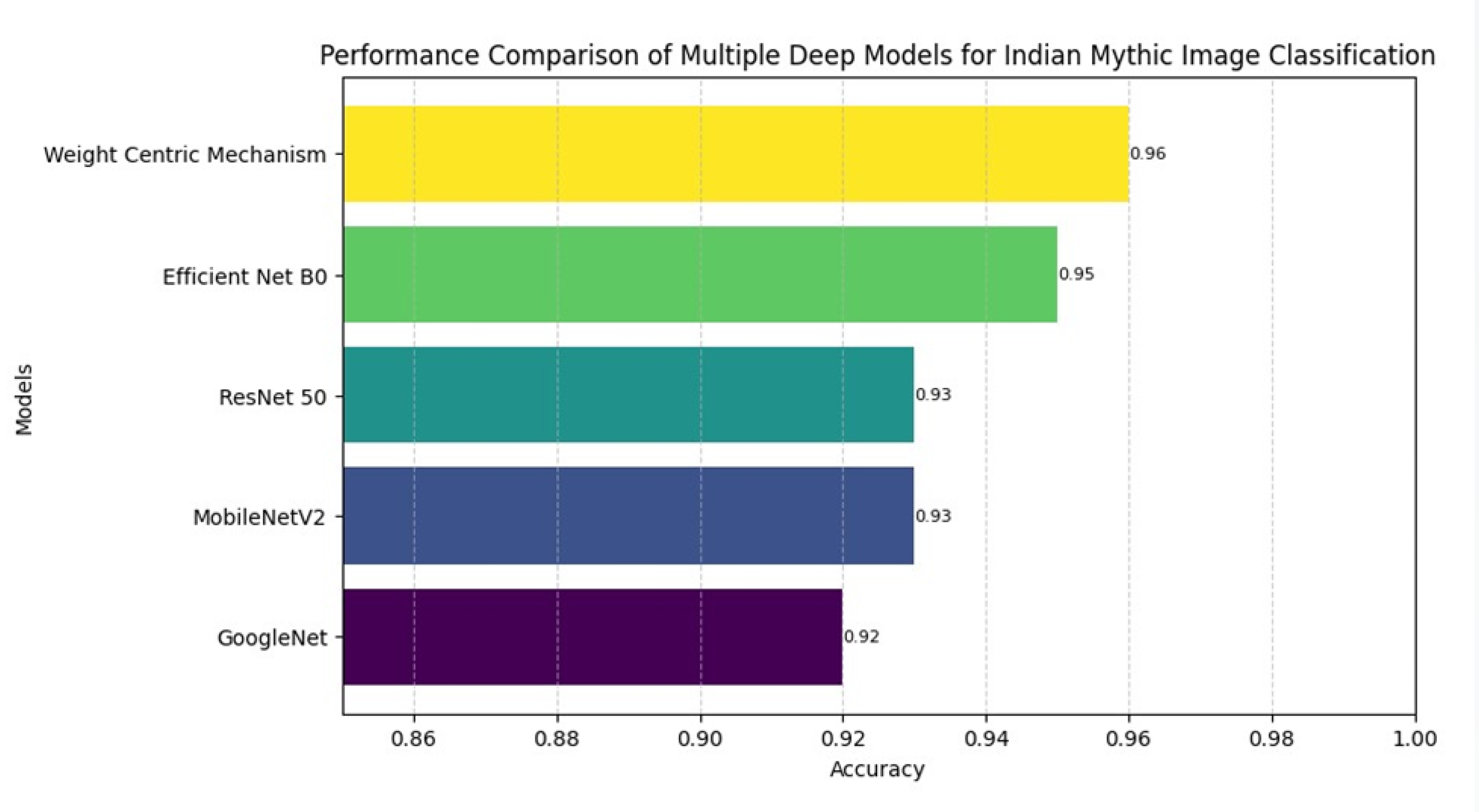
Googlenet (Inception V3)

CONFUSION MATRIX



Weight - centric Mechanism

COMPARISON


Performance Comparison of Multiple Deep Models for Indian Mythic Image Classification
Weight Centric Mechanism
Efficient Net B0
ResNet 50
MobileNetV2
GoogleNet
0.96
0.95
0.93
0.93
0.92
Models
0.86
0.88
0.90
0.92
0.94
0.96
0.98
1.00
Accuracy

CONCLUSION

- The "Mythic Vision" project holds significant potential to revolutionize the way tourists engage with and appreciate the cultural heritage of India. By using deep learning technology, the software offers an accessible and accurate means of learning about Indian mythology, eliminating the need for costly guides. This project aligns with the broader goals of promoting sustainable tourism and preserving cultural heritage, ensuring that future generations can continue to explore and appreciate the richness of India's cultural history.

- In conclusion, the successful completion of this project has the potential to not only enhance the tourism experience in India but also serve as a model for similar initiatives in other parts of the world.