Utilising Machine Learning and Image Processing for Object Recognition in Traditional Indian Theatre

Bachelor Thesis

Bachelor of Science
in
Computer Science
by

Sammar Ahmed
(R200240300015)

Under the Guidance of
Prof. Divakar Singh

BARKATULLAH VISHWAVIDYALAYA, BHOPAL
(FORMERLY BHOPAL UNIVERSITY, BHOPAL)

Department of Computer Science
Barkatullah University
Bhopal, Madhya Pradesh

July 2024

CERTIFICATE

This is to certify that the work contained in the thesis entitled ***"Utilising Machine Learning and Image Processing for Object Recognition in Traditional Indian Theatre"*** *is a Bonafide work of* ***SAMMAR AHMED*** *(R200240300015), carried out in the department of Computer Science, Barkatullah University, under my supervision and that is has not been submitted elsewhere for a degree.*

Prof. Divakar Singh
Head of Department
Department of Computer Science
Barkatullah University, Bhopal

Head of Department
Computer Science & Engineering
University Institute of Technology,
Barkatullah University,
BHOPAL

ABSTRACT

Yakshagana is an immensely admired folk theatre and dance form of Karnataka. It is famous for its music, colourful makeup and costumes. Currently there doesn't exist any database or model that will help the people who are new to this art form easily identify which character is which *Vesha,* our project thus focuses on creating a diverse and extensive repository of dance form images along with a CNN model leveraging existing architectures.

Using Convolutional Neural Networks (CNNs), an effective method for picture categorization, our goal is to create a reliable model that can reliably discern between Yakshagana' s complex personalities. This technology technique serves as a model for comparable efforts in the fields of traditional performing arts and cultural heritage, in addition to aiding in the preservation of Yakshagana.

Using various sources such as YouTube, Facebook, Google Images we created a dataset of around 1500 images across the different classes. We tried various pre-existing models along with a custom model through which the best accuracy came around 93%. Along with it we created a website in which people can upload images and it will tell them which *Vesha* is in the image.

Contents

List of Tables

List of Figures

ABBREVIATIONS

CNN: Convolutional Neural Network

VGG: Visual Geometry Group

ResNet: Residual Network

DenseNet: Densely Connected Convolutional Networks

PIL: Python Image Library

ML: Machine Learning

API: Application Programming Interface

YOLOv8: You Only Look Once version 8

TL: Transfer Learning

NOTATIONS

Acc: Accuracy

L(y, $\hat{y}$): Loss

TP: True Positive

TN: True Negative

FP: False positive

FN: False Negative

y: Actual Label

$\hat{y}$: Predicted Label

N: Number of samples in dataset

Chapter 1

Introduction

1.1 Area of Work

Yakshagana is one of the most popular folk theatre forms of Karnataka. It is noted for its music, colourful costumes, vigorous dance movements, subtle expressions and extempore dialogues. The tenkuthittu, or southern style, is predominant in the region centred on Mangalore, which includes Mangalore, Sullia, Puttur, Sampaaje, Bantwala, Belthangady, Karkala, Kasaragod (Kerala), Kodagu, etc. In the Udupi district, as well as in sections of the Uttara Kannada district and Kundapura, the badaguthittu, or northern style, is predominant. [Fig 1.1]

1.2 Motivation

For people who our new to this art form and our mesmerized by the costumes and different characters present in it, there doesn't exist a way for them to be able to easily decipher and identify which Vesha is which and that can be a little overwhelming. Our CNN model helps the beginners or people who are interested in Yakshagana to be able to easily identify which character is what Vesha or part of the dance. The model is trained on a custom dataset consisting of a diverse and extensive range of dance forms.

This project strives to bridge the gap between traditional art forms and modern technology, ensuring that the essence of Yakshagana transcends generations through accurate and efficient identification and classification methodologies.

1.3 Objective

- Create a diverse and extensive repository of dance form images.
- Preprocess and label acquired images through image processing techniques.
- Train a CNN model for classification, leveraging existing architectures.
- Evaluate model efficiency and accuracy by employing relevant metrics

Figure 1.1 Some characters of the dance form (Veshas)

Chapter 2

Literature Review

Indian classical dance combines everybody component in a motion. It can take many different forms, but it usually combines leg alignment, hip movement, eye movement, leg posture, and single or double hand mudras. Every dancing style has a distinct gesture that may be used to categorise it. Dancers' attires are distinctive as well. **[1]** In this study, a Deep Learning CNN is proposed for the identification and categorization of photographs of Indian classical dance. The dataset used in this study includes five dance courses: Bharatanatyam, Odissi, Kathak, Kathakali, and Yakshagana. Google Crawler was used to gather the photos for these classes from the internet. Anyone may use this method to determine how well-versed they are in the diversity of dance forms in India given its wide range of postures and styles. The final accuracy of the resnet34 was 78.8%.

Facial expressions and hand gestures are crucial in communicating the tale of the accompanied music to the audience in any type of dance, whether it be folk or classical. Due in part to a severe lack of skilled and committed instructors and gurus, Bharatanatyam, a traditional dance style with roots in southern India, is on the approach of being fully mechanised. **[2]** They chose to detect and classify the single hand gestures, mudra, and hasta using two variations of CNNs, which demonstrates effectiveness of transfer learning regardless of the domain difference between the pre-training and the training dataset. This is an honest attempt to expedite this automation process while maintaining the cultural heritage. Building a new dataset of single hand gestures that consists of 27 classes is the main goal of this work. Other sources of data included Gogle images, dynamic YouTube videos with, professional artists working in staged environments with plain backgrounds, and more. They were able to get an accuracy of 98.25% through DTL technique.

Artificial Intelligence (AI) has revolutionized decision-making through mimicking human intelligence, employing ML and, specifically, DL with Artificial Neural Networks (A.N.Ns). DL, a key ML subset, utilizes ANNs for tasks like computer vision, speech recognition, and automated driving. ANNs, evolving through training and learning algorithms, enhance intelligent systems by minimizing human intervention. However, for image classification, CNNs outperform ANNs. CNNs utilize filters to map image features, making them suitable for tasks requiring thousands of features. **[3]** In this study in the comparison between different CNNs ResNet50 emerges as the most accurate CNN model, surpassing VGG19 and VGG16, with optimal performance observed between 15 to 20 epochs and an accuracy of 97.33% on the test data. The dataset of 6000 images was used that needed to be classified into 5 categories.

[4] This study uses CNN with transfer learning to detect 7 basic human states, that are Anger, Disdain, Disgust, Fear, Jovial and Sad. The paper compares three pre-trained networks VGG16, ResNet50 and a SE-ResNet50. Modified networks of the mentioned architectures are trained on images present in the dataset. The study was able to get val-accuracies of 96.8%, 99.47%, and 97.34% for the three networks, respectively. In the end ResNet50 was the most accurate out of all three

[6] One of the fatal diseases that requires extremely precise medical surgery is a brain tumour. MRI can be used to detect brain tumours. The purpose of image segmentation is to clearly define the tumor's boundaries and to distinguish the tumour area (region of interest, or ROI) from the healthy brain. This study uses a fully CNN with a unique design, the UNet & VGG16, to classify ROI and non-ROI. With the use of pre trained to make the U-Net architecture simpler, this new model is a hybrid of the UNet and VGG16. In the learning dataset, this approach has a high accuracy of around 96.1%. The CCR score indicates that UNet-VGG16 was able to identify the region affected by brain tumours, with a mean C.C.R value of almost 95.69%.

Recent research in computer vision has witnessed a surge in image classification studies, with a focus on both classical ML and DL methodologies. While deep learning, particularly utilizing CNNs like VGG19, has shown impressive results, challenges persist in fully extracting crucial image information, impacting overall accuracy. **[6]** This study proposes enhancing image classification by amalgamating deep features from VGG19 with handmade features like SIFT, SURF, ORB, and Shi-Tomasi corner detection. Classification using Gaussian Naïve Bayes, Decision Tree, Random Forest, and XGBoost classifiers on the Caltech-101 dataset demonstrates Random Forest's superior accuracy at 93.73%. The findings emphasize the effectiveness of combining deep and traditional features for robust image classification.

[7] This study introduces a brand classification method for multiple types of coal and multiple class image sorting using TL and learning from VGG Net, Inception Net, and Res Net. The results demonstrate the effectiveness of CNNs in coal image classification, with the Res Net 50 model achieving the highest among all the coal datasets with accuracy rates of 90.19%, 83.55%, 79.29%, and 83.42%, respectively for anthracite, gas, coking and mixed.

Chapter 3

Problem Definition

The project's research topic centers on the lack of a specific dataset and effective technical techniques for recognising and categorising the important elements of Yakshagana, an Indian traditional dance play performed in Karnataka. Although Yakshagana represents a rich cultural legacy, utilising cutting-edge technology like Convolutional Neural Networks (CNNs) to automate the identification procedure is severely hampered by the absence of a dedicated dataset. Looking at the complexity and intricacy of Yakshagana entities, our work attempts to create a novel dataset that encompasses a lot of their many visual components. To provide precise and proper categorisation, the problem also involves developing a CNN model specifically suited to the intricacies and minute details of Yakshagana. By addressing these gaps, the project aims to contribute to the preservation of Yakshagana while also offering a technological framework for people to interact with and thus increasing the influence.

Chapter 4

Tools Used

4.1 Dataset Creation

In order to gather a large and engaging dataset for our model, the images were obtained from a wide range of platforms such as snap shots from **YouTube videos**, members of various Facebook groups and **Google Images**. The use of images also helps the dancer to keep and capture the delicacy of various objects present in the dance. The dataset was then annotated and pre-processed before the model was trained on it.

4.2 Platform - Google Colab

Google Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education. The collaborative nature of Google Colab enabled seamless collaboration among team members by facilitating real-time sharing and editing of notebooks.

4.3 Python

The project's major programming language was Python. It was the best option for creating and deploying machine learning and deep learning models because of its adaptability, large library, and simplicity of use. Python was used for model building, experimentation, data preparation, and analysis of results.

4.4 Convolutional Neural Network (CNN)

These are a subclass of DL neural networks designed to work with structured grid data, such as images. CNNs are really effective for problems such as image recognition, object detection, and computer vision because they can automatically and adaptively learn spatial information of features from input images.

4.5 Libraries

4.5.1 TensorFlow

The Google Brain team created the open-source machine learning framework TensorFlow. Its purpose is to make machine learning and deep learning model building, training, and deployment easier. Researchers and developers use TensorFlow extensively because of its rich ecosystem, scalability, and adaptability.

4.5.2 Keras

Keras is an open-source software library that provides a Python interface for making artificial neural networks. It acts as an abstraction layer, simplifying the building and training of neural network models. It was initially developed as a user-friendly API for deep learning and has since become an important part of the TensorFlow library.

4.5.3 Python Image Library

PIL is a free and open-source library in Python that helps us open, perform different functions and save the images in different file formats. It is also used for pre-processing the dataset before feeding it to the model.

4.5.4 Matplotlib

Matplotlib is an extensive library for creating static, animated, and interactive visualizations in Python. Because of its simplicity and adaptability, it is frequently utilised in the scientific and data analytic sectors. It provides a robust framework for creating a variety of plots, from simple line graphs to complex multi-plot figures.

4.5.5 Sci-Kit Learn (sklearn)

It is a free machine learning package in Python that makes data mining and analysis easier and faster. It is based on NumPy, SciPy and Matplotlib libraries.

4.6 Framework

4.6.1 Flask

It is a python web framework that is used for the crawling of web applications. It is particularly easy to use with plenty of flexibility, which makes it a popular platform for web service and API creations as well as for designing small-to-medium web applications.

4.7 Models Used

4.7.1 VGG19

It is a deep convolutional neural network architecture composed of 19 layers, that uses very small (3x3) convolution filters. This makes it distinctive for its simplicity and uniformity in design. It is developed by the Visual Geometry Group at Oxford. VGG19 achieved high performance on the ImageNet dataset, which demonstrated the effectiveness of deeper network for image recognition.

4.7.2 ResNet50

It is a 50-layer deep residual network that solves the vanishing gradient issue that sometimes plagues deep networks by using skip connections, or shortcuts, to jump over some levels. It guarantees that layers will learn residual mappings, and thus it enables the training of much deeper networks, improving convergence and performance, particularly in image recognition tasks.

4.7.3 DenseNet121

It is a densely connected CNN that introduces direct connections between any two layers with the same feature-map size, thus improving

information flow and gradients throughout the network. This 121-layer design is useful for picture classification and other vision applications because it reduces the vanishing-gradient issue, promotes feature reuse, and boosts computing efficiency.

4.7.4 YOLOv8

It is an advanced real-time object detection model known for its speed and accuracy. It divides images into a grid and predicts bounding boxes and class probabilities directly from the full images in one go. Because of its high efficiency, it used for detecting objects in various applications such as video surveillance, autonomous driving, and robotics.

4.7.5 Custom Model

A custom CNN model was also developed. It begins with a **Conv2D layer** with 32 filters of size 3x3, uses the **ReLU** activation function, and takes input images of size 256x256 with three color channels (RGB). This is followed by a **MaxPooling2D layer** with a pool size of 2x2 to reduce the noise of the feature maps and to take the strongest features forward, this in turn aid in reducing computation and control overfitting. The pattern of a convolutional layer followed by a pooling layer is repeated, first with 64 filters and then with 128 filters, each time increasing the depth of the feature maps to learn more complex features.

The outputs of the convolutional and pooling layers are flattened into a one-dimensional vector, and then fed into a densely connected (fully connected) layer with 512 neurons, using ReLU activation for non- linearity. The final layer is a dense layer with 5 neurons and a softmax function, for the 5 classes.

Chapter 5

Methodology

5.1 For single character in an image

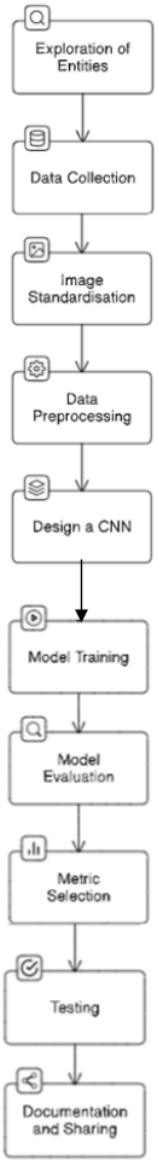


Figure 5.1 Methodology for single image classification

5.1.1 Exploration of Entities

Various sources of literature were referred to identify the important and recurring characters in the dance. After much research and review, the final entities that would be used in the dataset and for the model to recognize were chosen to be [Fig 5.2] *Raja Vesha, Stree Vesha, Bannada Vesha, Hasyagara and Bhagvata.* Other important entities that weren't included are Pundu Vesha and Hanumantha among others. This decision was made because, in a small dataset, similar-looking characters might be easily mistaken for one another, thereby reducing the efficiency of the model.

(L to R) Raja Vesha, Stree Vesha, Hasyagara

(Top) Bhagavata , (Down) Bannada Vesha

Figure 5.2 5 Key Characters of Yakshagana

5.1.2 Data Collection and Preprocessing

To create a diverse and extensive dataset for our model, images from a variety of platforms that include screengrabs from **YouTube videos**, images from various **Facebook group**s and **Google Images** were taken. The collection of images ensures that it maintains and captures the intricacy of different entities present in the dance. A total of **1300** images were collected.

The images are first converted to **RGB format** then **padding [Fig 5.3]** is applied to the pictures to preserve spatial information i.e. **maintain their**

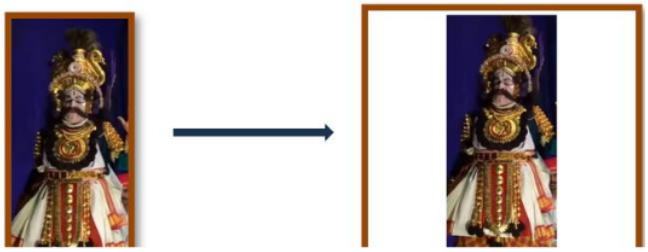


Figure 5.3 Picture border highlighted to show before and after padding.

aspect ratio and prevent loss during **resizing** to the target size of the model of 256x256 pixels. The dataset was divided in a 80-20 ratio into Training and Validation sets manually such that no two same images are common in both.

5.1.3 Data Augmentation

The dataset was enhanced by through techniques such as **zooming** by 50% in or out, **rotation** by 20 degrees in either direction and **flipping** the image horizontally. All of these were done to create variation and prevent the model from overfitting. [Fig 5.4]

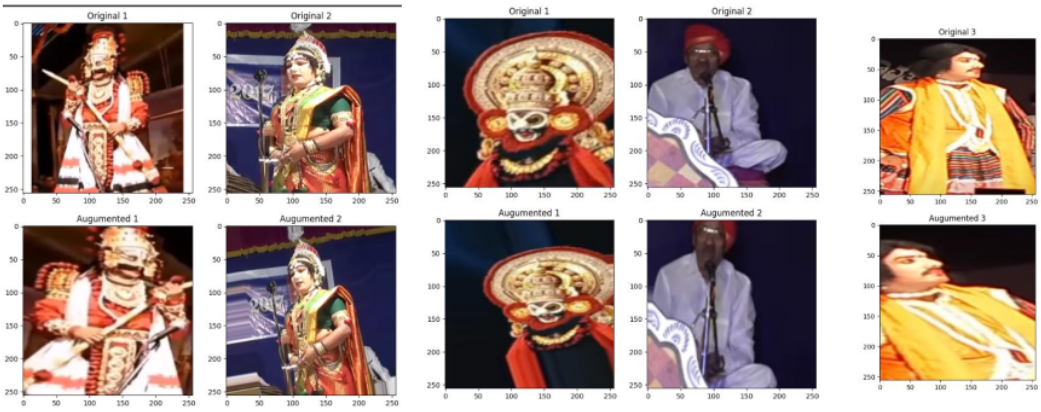


Figure 5.4 Before and After Augmentation Images

5.1.4 CNN Architecture

The project mainly used and compared the performance of 3 pre-trained model **VGG19[Fig 5.5]**, **ResNet50** and **DenseNet121**. The models were pre-trained on **ImageNet database**, that contains over 14,000,000 images with over 20000 categories, this provided are models to have robust feature extraction.

The models were then customised, the **top layers are removed**, and a new **dense layer** including a **SoftMax function** is added to classify the images into five specific classes: Bannada Vesha, Bhagvata, Hasyagara, Raja Vesha, and Stree Vesha.

The **lower layers of models are frozen** to retain the learned features from

ImageNet, ensuring that the model can make use of the general patterns while learning new class-specific features. A custom model was also used to be run on the dataset.

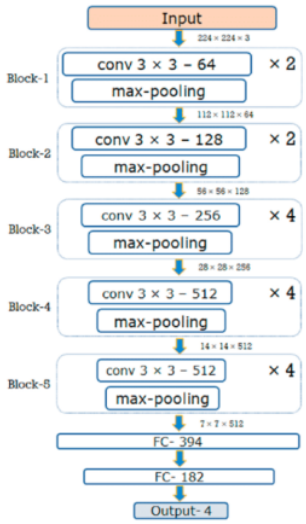


Figure 5.5 VGG19 Architecture

5.1.5 Model Training

Transfer learning is utilized by leveraging the pre-trained models as the base architecture. The fully connected layers of the models are replaced with a custom classifier consisting of a flatten layer followed by a dense layer including SoftMax for multiclass classification of the 5 entities. The model is then **compiled using the Adam as optimizer and categorical cross-entropy is the loss function**. To mitigate class imbalance, class weights are computed and utilized during training. [Table 5.1]

The model was trained over **10 epochs** and both **training loss and accuracy along with validation loss and accuracy** were monitored**. Early stopping** is used to prevent overfitting, it has **patience of 3** i.e., if the validation accuracy doesn't improve over 3 epochs, the training is stopped. Also, **Model Checkpoint** callback ensures that only the best performing model is saved.

PARAMETER	VALUE
OPTIMIZER	Adam
BATCH SIZE	16
EPOCHS	10
LOSS FUNCTION	Categorical Cross-Entropy
OUTPUT ACTIVATION	SoftMax
METRIC	Accuracy

Table 5.1 Hyperparameters for Training Dataset

5.1.6 Metric Selection

Accuracy [Fig 6.2, Fig 6.4] and Loss[Fig 6.3,Fig6.5] were the main metrics used to measure the model's performance. TPR [Fig 6.6] and FPR of the classes were also measured along with confusion matrix generation [Fig 6.1].

$$Acc = \frac{TP+TN}{TP+TN+FP+FN}$$

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

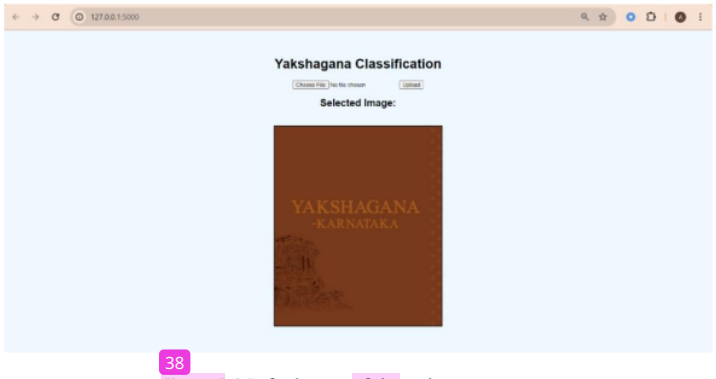
5.1.7 Testing

The model was evaluated on a **separate set of validation images** (around **300**) that are not used during training to gauge the ability of the model to generalise to new images. The dataset was divided manually such that no two pictures from the same video of the same character were present in both. Different pictures of the same person were either wholly in the training dataset or the validation dataset. The performance of all the four models was compared to each other and the best model was saved and deployed as a web app. The confidence of the prediction is depicted along with the actual as well as the predicted class. **[Fig 6.7 , Fig 6.8 , Fig 6.9 , Fig 6.10 , Fig 6.11]**

5.1.8 Comprehensive Documentation and Deployment of the model

The complete steps of model making, data collection, data preprocessing, data augmentation, analysis and display of the results is thoroughly documented for future reference and improvements.

The best model out of all was deployed as a web application. The frontend was made using **HTML, CSS and JavaScript**, the backend was made using **Flask** framework and deployed on a local server. A person who is new to Yakshagana might have difficulty identifying all the characters, to make it easier for beginners they can input an image of a single character and the model will classify the character and give the result. [Fig 5.6, Fig 5.7, Fig 5.8, Fig 5.9, Fig 5.10, Fig 5.11, Fig 5.12]



38

Figure 5.6 Default page of the web app

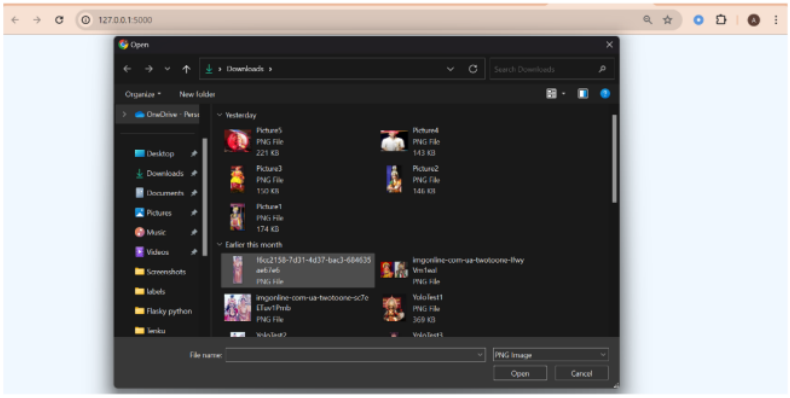


Figure 5.7 Picture Selection

Figure 5.8 Selected image "Stree Vesha"

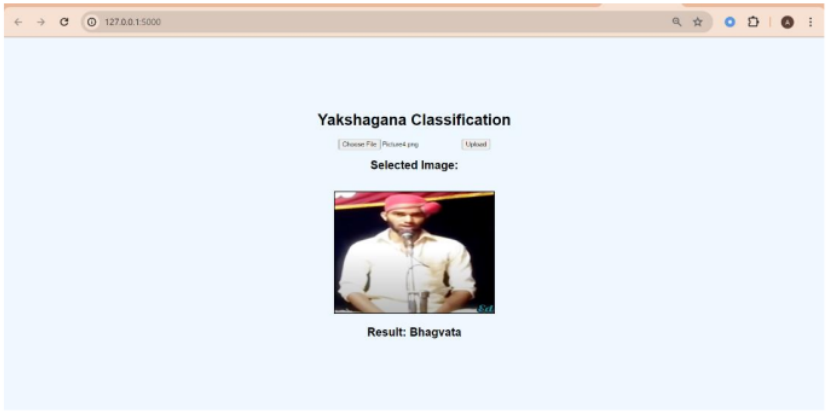


Figure 5.9 Selected Image "Bhagvata"

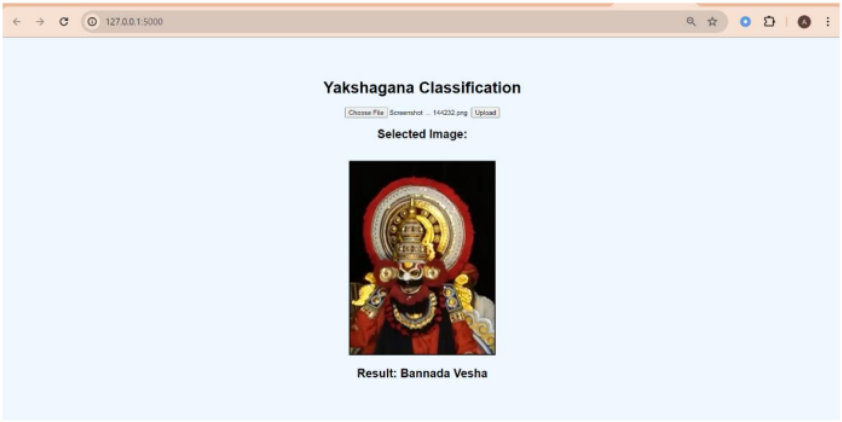


Figure 5.10 Selected Image "Bannada Vesha"

Figure 5.11 Selected Image "Hasyagara"

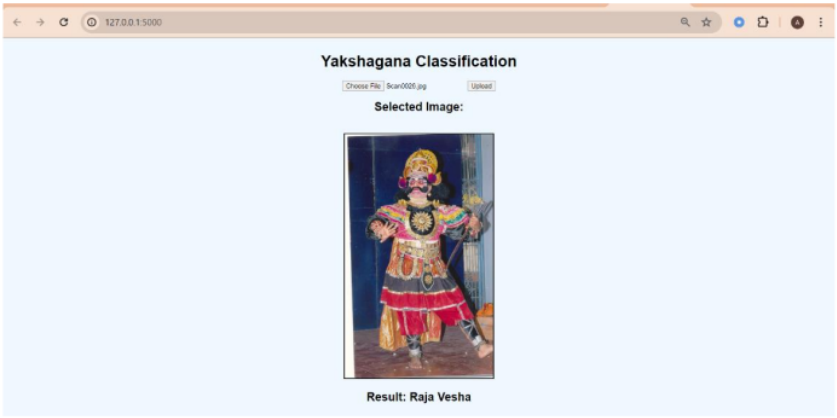


Figure 5.12 Selected Image "Raja Vesha"

5.2 For multiple characters in an image

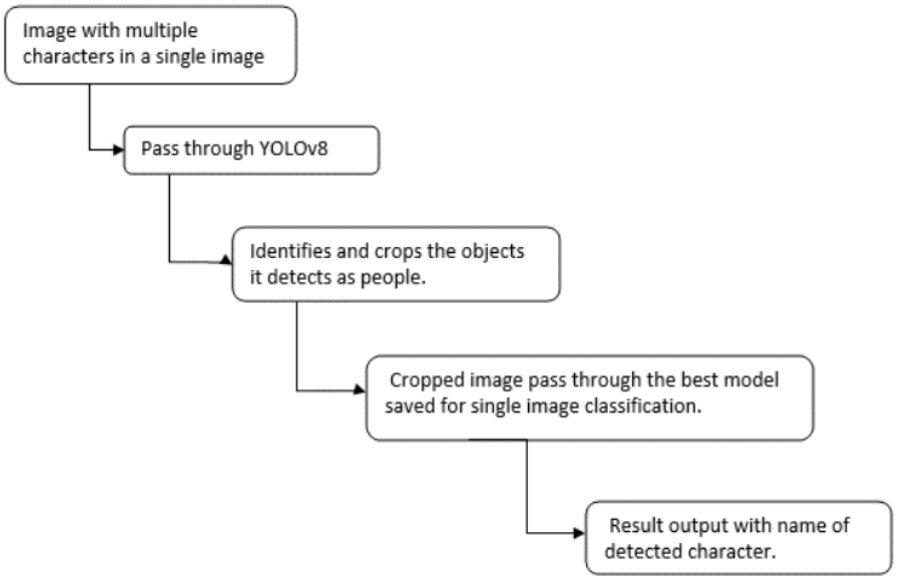


Figure 5.13 Methodology for multiple characters in a single image

Firstly, the **YOLOv8** model is loaded and used to detect objects in the image. For each image, the objects detected as person are outlined with bounding boxes. These **bounding boxes** are used to crop the specific parts of the image, which are saved in a destination folder. The original and cropped images are displayed using Matplotlib. A total of **40 images** having multiple characters in a frame were used to check the model.

Secondly, the new cropped images are **pre-processed** before being passed to the classifier. The images are converted to RGB, padded to maintain the aspect ratio before being resized to the target image size of the classifier model.

Then we loop through the cropped images which each have a single character present in them and classify using the best model that we saved during our training of single character in a single image classification.[Fig 6.12, Fig 6.13, Fig 6.14]

Chapter 6

Results

6.1 For single character in an image

After running the dataset over 10 epochs for all the models, the best result for given by **VGG19**, with an accuracy of approximately **94.2%** on the validation set[Table 6.1]. The run time was 4m 52s.

	VGG19	ResNet50	DenseNet121	Custom
Epoch 1	0.860	0.533	0.817	0.634
Epoch 2	0.884	0.533	0.875	0.697
Epoch 3	0.899	0.480	0.937	0.754
Epoch 4	0.884	0.557	0.903	0.855
Epoch 5	0.875	0.533	0.913	0.810
Epoch 6	0.909	0.601	0.764	0.759
Epoch 7	0.918	0.545	0.856	0.781
Epoch 8	0.798	0.562	0.903	0.851
Epoch 9	0.903	0.634	0.911	0.822
Epoch 10	0.942	0.663	0.872	0.795

Table 6.1 Validation Accuracy comparison over 10 epochs for different models

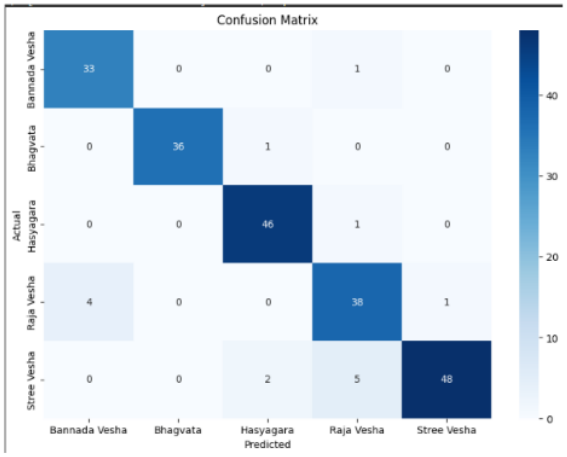


Figure 6.1 Confusion Matrix generated for the VGG19 model

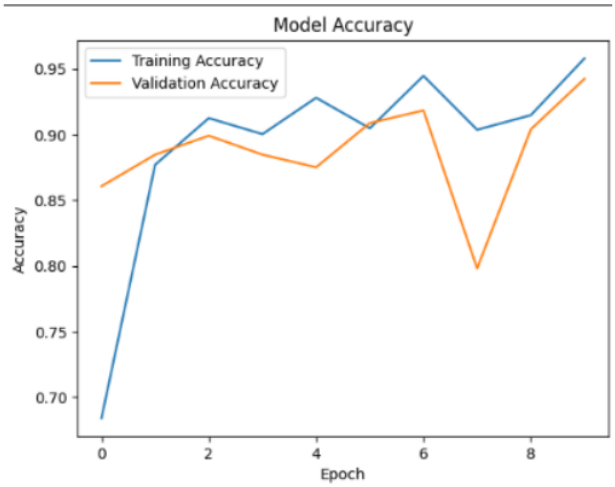


Figure 6.2 Model Accuracy VGG19

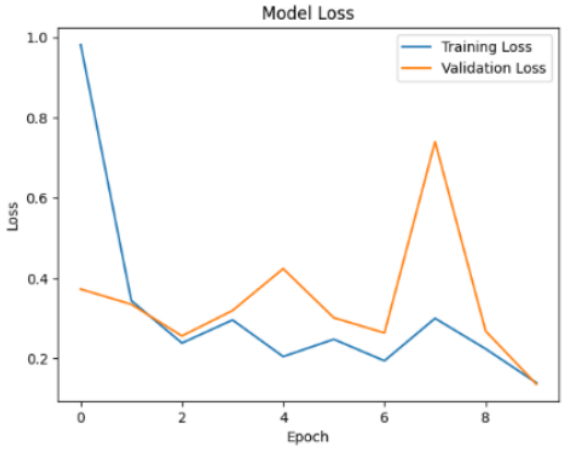


Figure 6.3 Model Loss VGG19

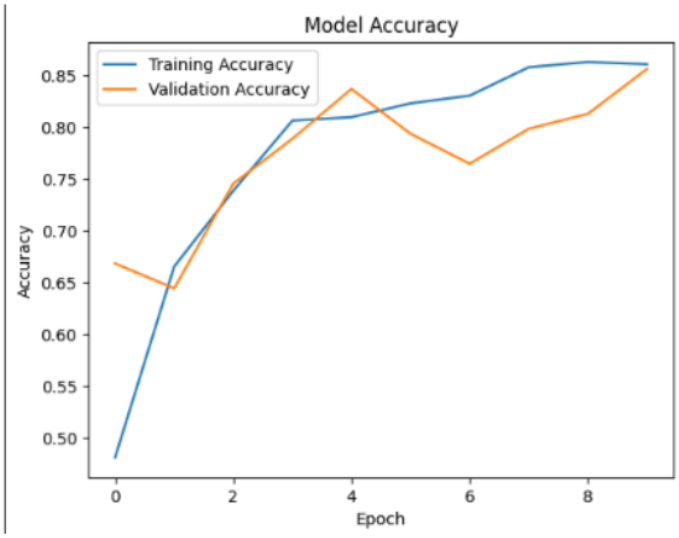


Figure 6.4 Model accuracy Custom model

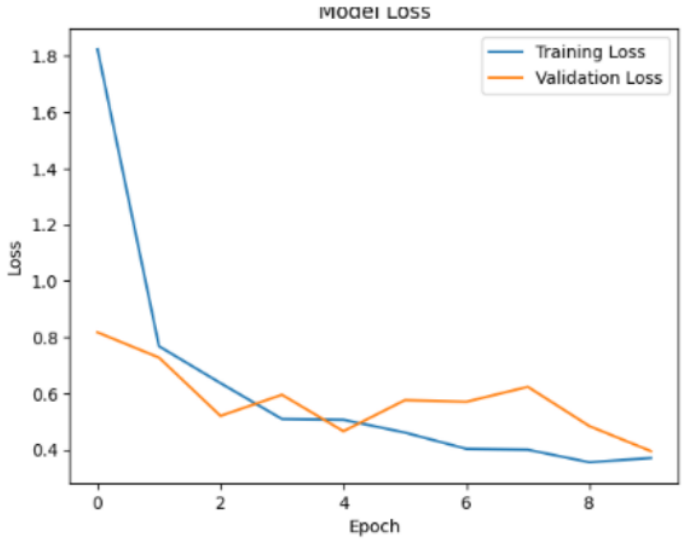


Figure 6.5 Model Loss Custom Model

```
Class: Bannada Vesha
  True Positive Rate (TPR): 1.00
  False Positive Rate (FPR): 0.03
Class: Bhagvata
  True Positive Rate (TPR): 0.98
  False Positive Rate (FPR): 0.00
Class: Hasyagara
  True Positive Rate (TPR): 1.00
  False Positive Rate (FPR): 0.02
Class: Raja Vesha
  True Positive Rate (TPR): 0.82
  False Positive Rate (FPR): 0.00
Class: Stree Vesha
  True Positive Rate (TPR): 0.96
```

Figure 6.6 TPR and FPR of different classes

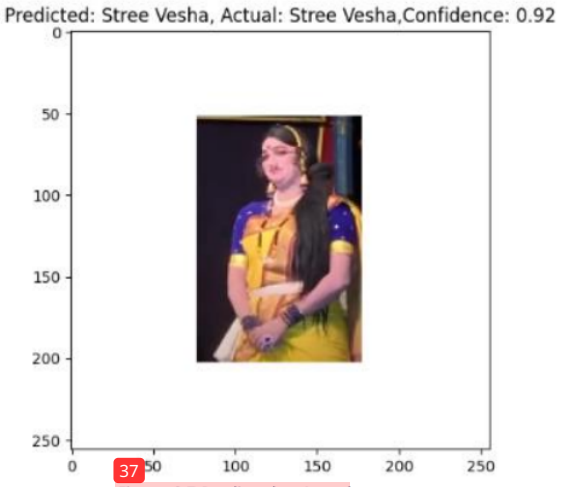


Figure 6.7 Predicted vs Actual

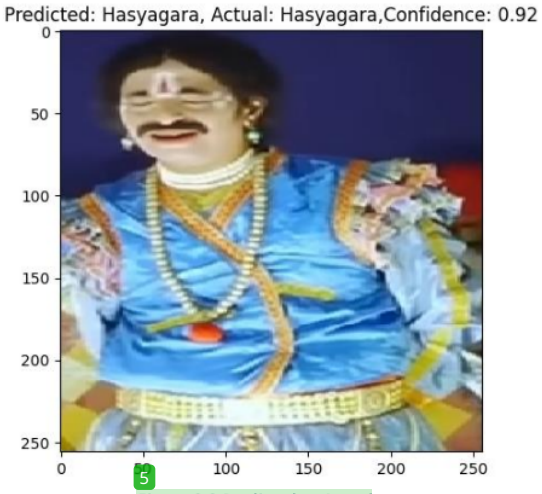


Figure 6.8 Predicted vs Actual

Figure 6.9 Predicted vs Actual

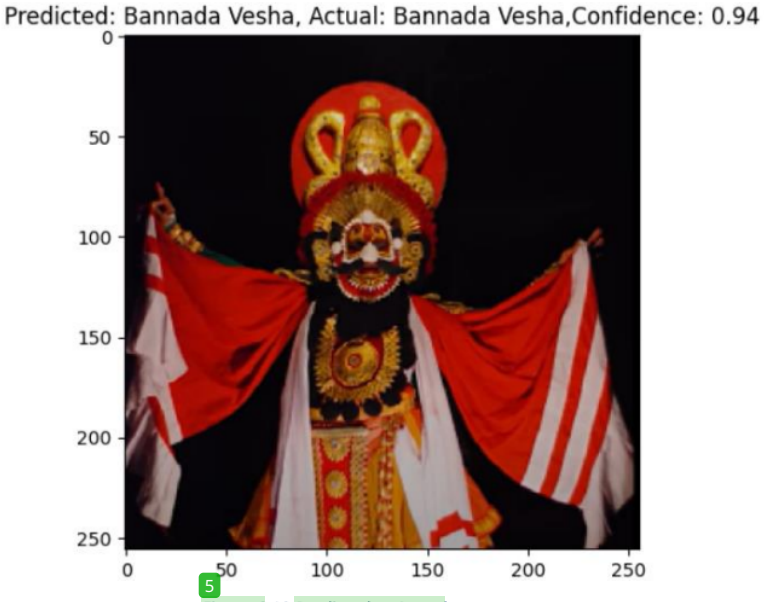


Figure 6.10 Predicted vs Actual

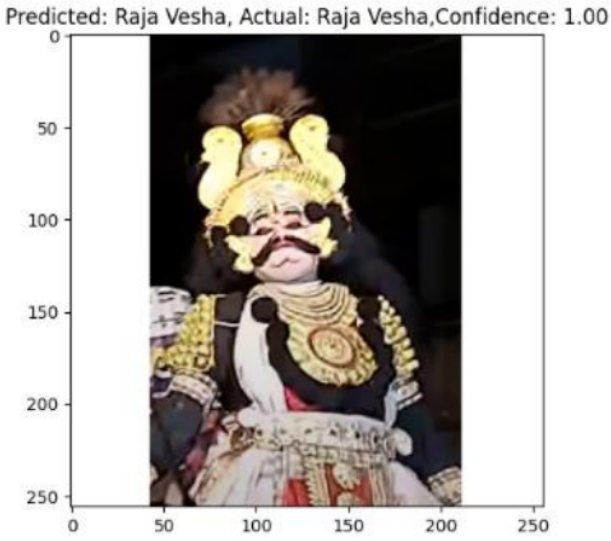


Figure 6.11 Predicted vs Actual

6.2 For multiple characters in an image

Of the total present 90 characters in 40 images 76 were detected giving it an accuracy of 84.4%.

Figure 6.12 Image followed by person detected followed by their classification

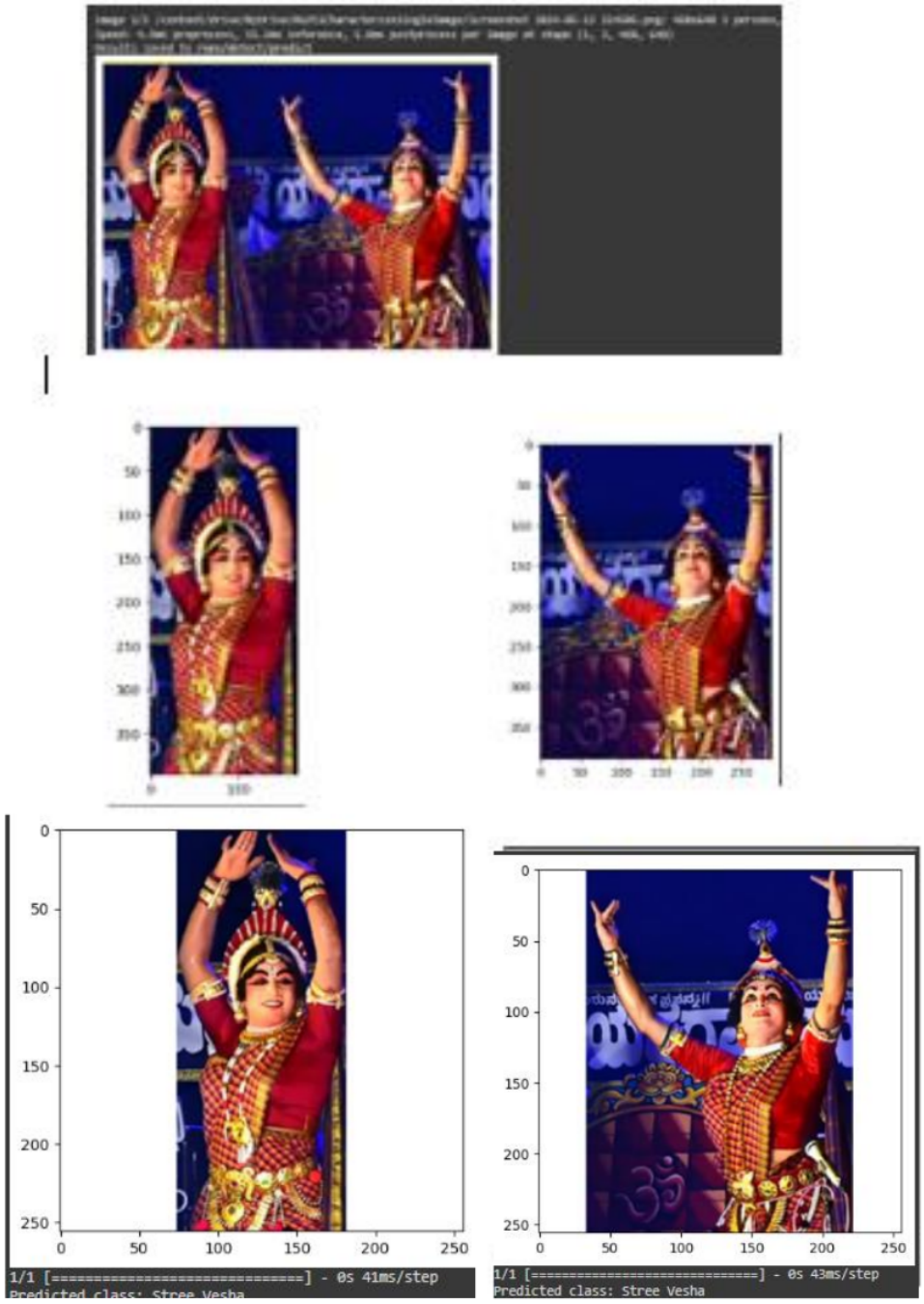


Figure 6.13 Image followed by person detected followed by their classification

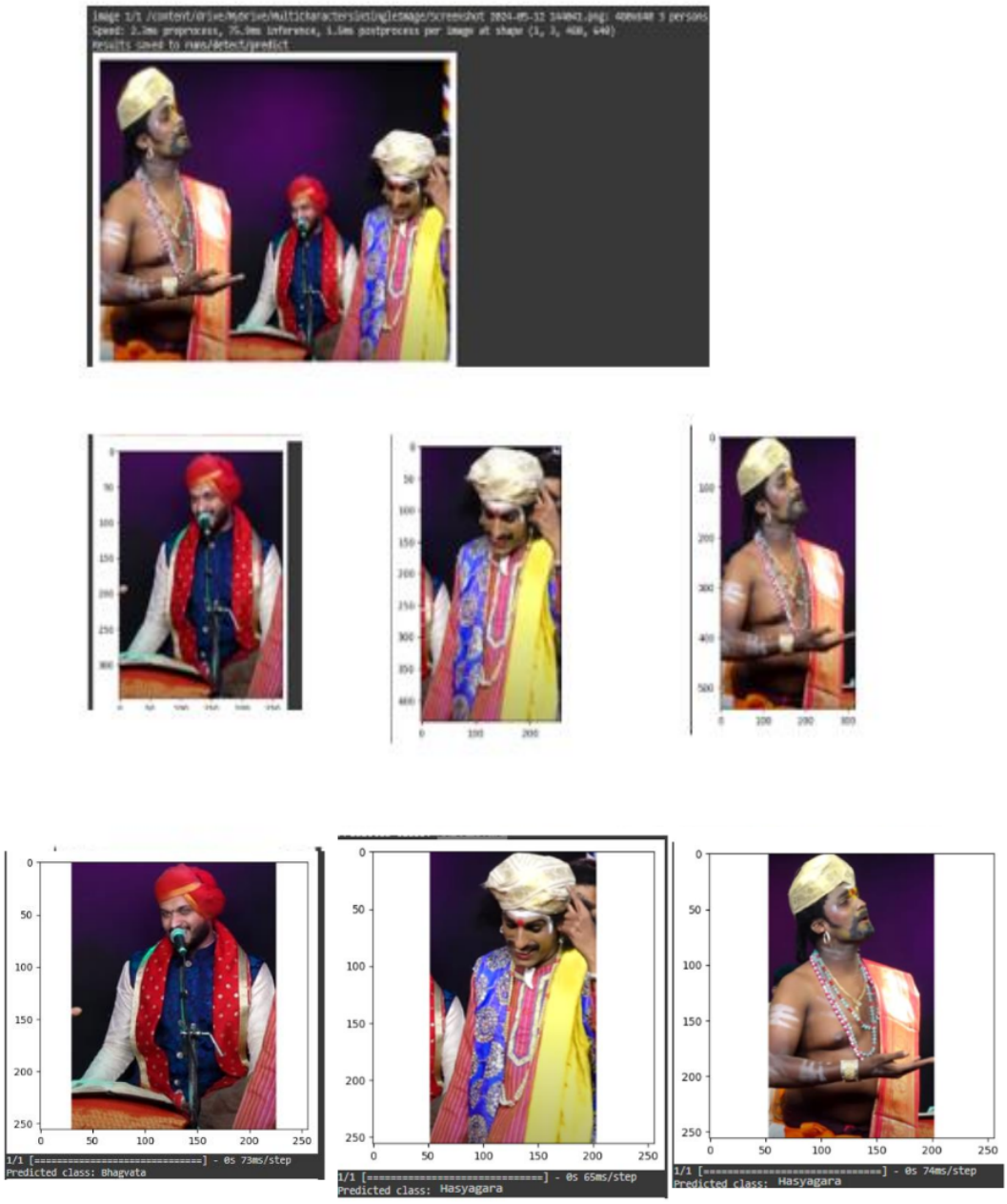


Figure 6.14 Image followed by person detected followed by their classification

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The project successfully connects the traditional art form of Yakshagana with modern technology by developing a robust Convolutional Neural Network (CNN) model for classifying various characters in the performances and making it easier for beginners to learn about it. The model, particularly the VGG19, demonstrated an accuracy of 94.2% on the validation set, effectively identifying characters based on their visual features.

This achievement demonstrates that the potential of deep learning can be used to help in advancing cultural heritage, popularizing it as long as this field is of interest for newbies. The creation of the dataset, its preprocessing and augmenting, and the creation of a web application for the sake of presenting the information in an accessible manner allows people to learn about Yakshagana characters easily if they wish to learn or just try something new.

7.2 Future Scope

The future directions could encompass the significant improvement of the custom models' quality and accuracy and the re-tuning of the existing models. The dataset can be made more visually rich and representative with more unique images in more detail which in turn will make the model more robust and accurate. A significant focus of improvement should also be put on the ability of the AI model to recognize multiple characters in one frame and then classify each of them.

Secondly, introducing of classification into more people via live streaming or through classification that the user can access through an online website.

Users will be able to upload or stream videos and get immediate character identification; this in turn will enhance the potential of the application more in promoting Yakshagana and possibly other hand: This would enable users to upload or stream videos and receive instant character identification which in turn will greatly increase the potential of the project in the promotion of Yakshagana and possibly other cultural heritage forms.

Appendices

8.1 Appendix A (Code)

8.1.1 Data Augmentation

```
batchSize = 16
trainDatagen = ImageDataGenerator(

    zoomRange=0.5,
    rotationRange=20,
    horizontalFlip=True,
    fillMode='nearest',

)
```

8.1.2 Pretrained Model

```
base_model = DenseNet121( include_top=False, input_shape=(256, 256, 3))


for layer in base_model.layers:
  layer.trainable= False

x=Flatten()(base_model.output)

x=Dense(units= 5, activation='softmax')(x)

model= Model(base_model.input,x)
```

8.1.3 Custom model

```
model = models.Sequential()


model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256,256, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
```

8.1.4 Model Compilation along with ModelCheckpoint and Early Stopping

```
checkpoint = "/content/drive/MyDrive/model/best_model.h5"
mc = callbacks.ModelCheckpoint(filepath=checkpoint,
                                            monitor='val_accuracy',
                                            mode='max',
                                            save_best_only=True,
                                            verbose=1)


es= callbacks.EarlyStopping(
    monitor='val_accuracy',
    patience=3,  # Number of epochs with no improvement after which training will be stopped
    restore_best_weights=True
)

cb=[mc,es]

# Train the model with validation data
history = model.fit(train_generator,
                    steps_per_epoch=train_generator.samples // batch_size,
                    epochs=10,
                    validation_data=validation_generator,
                    validation_steps=validation_generator.samples // batch_size,class_weight=class_weights,
                    callbacks=cb)
```

8.1.5 Result Display

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

8.1.6 YOLOv8 model

```
# Load a pretrained YOLOv8n model
model = YOLO('/content/yolov8n.pt')

# Define paths to the source folder and the destination folder for cropped images
source_folder = '/content/drive/MyDrive/MultiCharactersinSingleImage'
destination_folder = '/content/drive/MyDrive/Cropped_images'

# Ensure the destination folder exists, create it if necessary
if not os.path.exists(destination_folder):
    os.makedirs(destination_folder)
```

```
for filename in os.listdir(source_folder):
    if filename.endswith('.png') or filename.endswith('.jpg') or filename.endswith('.jpeg'):
        # Construct the full path to the image file
        source = os.path.join(source_folder, filename)

        # Run inference on the source image
        results = model(source, save=True)
        boxes = results[0].boxes.xyxy.tolist()

        # Open the image
        image = Image.open(source)
        plt.imshow(image)
        plt.axis('off')  # Turn off axis
        plt.show()

        # Crop the image according to each bounding box
        for box_index, box in enumerate(boxes):
            x_min, y_min, x_max, y_max = box

            x_min, y_min, x_max, y_max = map(int, [x_min, y_min, x_max, y_max])
            # Crop the image
            cropped_image = image.crop((x_min, y_min, x_max, y_max))
            # Save the cropped image
            cropped_filename = os.path.splitext(filename)[0] + f'_cropped_{box_index}.png'
            cropped_path = os.path.join(destination_folder, cropped_filename)
            cropped_image.save(cropped_path)
            plt.imshow(cropped_image)
            plt.show()
```

References

[1] A. D. Naik and M. Supriya, "Classification of Indian Classical Dance Images using Convolution Neural Network," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 1245-1249, doi: 10.1109/ICCSP48568.2020.9182365. keywords: {Feature extraction;Convolution;Machine learning;Videos;Training;Neural networks;Google;CNN;fastai;Google Crawler;Indian dance forms}

[2] Parameshwaran, Anuja P., et al. "Transfer learning for classifying single hand gestures on comprehensive Bharatanatyam Mudra dataset." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2019.

[3] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 96-99, doi: 10.1109/CENTCON52345.2021.9687944.

[4] Theckedath, D. and Sedamkar, R.R., 2020. Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. SN Computer Science, 1, pp.1-7.

[5] Pravitasari, A.A., Iriawan, N., Almuhayar, M., Azmi, T., Irhamah, I., Fithriasari, K., Purnami, S.W. and Ferriastuti, W., 2020. UNet-VGG16 with transfer learning for MRI-based brain tumor segmentation. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(3), pp.1310-1318.

[6] Bansal, M., Kumar, M., Sachdeva, M. and Mittal, A., 2021. Transfer learning for image classification using VGG19: Caltech-101 image data set. Journal of ambient intelligence and humanized computing, pp.1-12.

[7] Liu, Y., Zhang, Z., Liu, X., Wang, L. and Xia, X., 2021. Deep learning-based image classification for online multi-coal and multi-class sorting. Computers & Geosciences, 157, p.104922.

ORIGINALITY REPORT

12%	9%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	mafiadoc.com Internet Source	2%
2	www.coursehero.com Internet Source	2%
3	www.researchgate.net Internet Source	<1%
4	T Richie Emmanuel Armstrong, P. Manimegalai, A. Abinath, D. Pamela. "Brain tumor image segmentation using Deep learning", 2022 6th International Conference on Devices, Circuits and Systems (ICDCS), 2022 Publication	<1%
5	Agarwal, Saurav. "An Approach of SLA Violation Prediction and QoS Optimization Using Regression Machine Learning Techniques.", University of Windsor (Canada), 2020 Publication	<1%

6	Yang Liu, Zelin Zhang, Xiang Liu, Lei Wang, Xuhui Xia. "Deep learning-based image classification for online multi-coal and multi-class sorting", Computers & Geosciences, 2021 Publication	<1 %
7	dokumen.pub Internet Source	<1 %
8	ijritcc.org Internet Source	<1 %
9	openaccess.thecvf.com Internet Source	<1 %
10	pdfcoffee.com Internet Source	<1 %
11	Ashwini Dayanand Naik, M Supriya. "Classification of Indian Classical Dance Images using Convolution Neural Network", 2020 International Conference on Communication and Signal Processing (ICCSP), 2020 Publication	<1 %
12	devpost.com Internet Source	<1 %
13	scholarworks.gsu.edu Internet Source	<1 %
	online-journals.org	

14	Internet Source	<1%
15	Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens van der Maaten, Kilian Q. Weinberger. "Convolutional Networks with Dense Connectivity", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022 Publication	<1%
16	www.ir.juit.ac.in:8080 Internet Source	<1%
17	dspace.christcollegeijk.edu.in:8080 Internet Source	<1%
18	Sheldon Mascarenhas, Mukul Agarwal. "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification", 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), 2021 Publication	<1%
19	dspace.univ-guelma.dz Internet Source	<1%
20	eandv.biomedcentral.com Internet Source	<1%
21	www.amrita.edu Internet Source	<1%

22	www.iieta.org Internet Source	<1%
23	"Data Engineering and Intelligent Computing", Springer Science and Business Media LLC, 2022 Publication	<1%
24	"Mobile Radio Communications and 5G Networks", Springer Science and Business Media LLC, 2023 Publication	<1%
25	research.sabanciuniv.edu Internet Source	<1%
26	wikimili.com Internet Source	<1%
27	"Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries", Springer Science and Business Media LLC, 2018 Publication	<1%
28	Bunny Saini, Divya Venkatesh, Nikita Chaudhari, Tanaya Shelake, Shilpa Gite, Biswajeet Pradhan. "A comparative analysis of Indian sign language recognition using deep learning models", Forum for Linguistic Studies, 2023 Publication	<1%

29	V. Amrutha Raj, G. Malu. "GAMNet: A deep learning approach for precise gesture identification", Journal of Intelligent & Fuzzy Systems, 2024 Publication	<1 %
30	ebin.pub Internet Source	<1 %
31	link.springer.com Internet Source	<1 %
32	porto.polito.it Internet Source	<1 %
33	semarakilmu.com.my Internet Source	<1 %
34	www.grin.com Internet Source	<1 %
35	Dhananjay Theckedath, R. R. Sedamkar. "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks", SN Computer Science, 2020 Publication	<1 %
36	"Computational Intelligence and Data Analytics", Springer Science and Business Media LLC, 2023 Publication	<1 %
37	Visser, Hendrik Marthinus. "The Neural Modelling of a Direct Reduction Process.",	<1 %

University of Johannesburg (South Africa), 2021
Publication

38 www.dgp.utoronto.ca
Internet Source
<1%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		