JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SIGHT-SCRIPTER USING DEEP LEARNING

A Virtual Tourist Guide by utilizing YOLOv8

[1]Poonam Narkhede, [2]Shubham Gode, [3]Manish Dait, [4]Shreyas Godbole

[1]Assist. Professor, Shivajirao S. Jondhale College of Engineering, Thane, India
[2-4]Student, Department of Computer Engineering, Shivajirao S. Jondhale College of Engineering, Thane, India,

Abstract : Sight Scripter is an innovative Android application designed to enhance the user experience of exploring Indian monuments by leveraging state-of-the-art machine learning techniques. Utilizing YOLOv8, a cutting-edge object detection model, Sight Scripter can accurately classify monuments in real-time through the device's camera or select images from the gallery. Upon detection, the app provides rich information about the monument, including its name, location, historical significance, and architectural features. This information is presented to the user in both text and audio formats, ensuring accessibility for all users. Additionally, Sight Scripter offers a comprehensive search feature, allowing users to discover monuments based on their state, city, or name. This search functionality enables users to access detailed information about a wide range of Indian monuments, thereby fostering a deeper understanding and appreciation of the country's rich cultural heritage. Through its intuitive interfaceand advanced capabilities, Sight Scripter aims to revolutionize the way users engage with Indian monuments, making cultural exploration more interactive, informative, and enjoyable.

Keywords – YOLOv8, Image Classification, Android Application, Deep Learning.

I. INTRODUCTION

In recent years, tourism has become an essential part of global travel, attracting numerous enthusiasts. This form of tourism, focused on exploring historical sites and heritage monuments, offers travelers unique insights into the rich tapestry of human civilization [12]. India, with its vast culture and rich heritage, has preserved its history through numerous landmarks and monuments. Each of these monuments has a story to tell, providing invaluable insights into India's past, present, and future. However, many visitors and enthusiasts may not have access to comprehensive information about these monuments, limiting their ability to fully appreciate and understand their significance.

During sightseeing, people often hire tour guides. However, many problems arise during these visits, such as scams, limited information about historical sculptures, and incomplete knowledge of guides. To address this gap, we present Sight Scripter—an advanced Android application that leverages machine learning to provide real-time classification and detailed information about Indian monuments. By utilizing the YOLOv8 object detection model, Sight Scripter accurately identifies monuments through the device's camera or gallery images.

Sight Scripter is more than just a tool for exploring Indian monuments; it is a gateway to understanding and appreciating India's rich cultural heritage. Its real-time classification and detailed information, combined with its powerful search feature, make it an indispensable tool for anyone interested in exploring India's historical monuments. Tourists visiting historical sites maynot always be aware of the buildings or statues they are observing. By allowing them to click a picture and upload it to our application, Sight Scripter recognizes the sculpture displayed in the picture and provides users with information on what they are viewing. In addition to real-time classification, Sight Scripter also includes a powerful search feature that allows users to discovermonuments based on their state, city, or name.

The main aim of this paper is to revolutionize the way users engage with Indian monuments, making cultural exploration more interactive, informative, and enjoyable.

II. LITERATURE SURVEY

The study on Monument Recognition using CNN [2] provides a comprehensive overview of previous work in this field, which we have incorporated into our paper. We aimed to improve the accuracy of monument classification by using basic hand-crafted features like HOG, LBP, and GIST. Our study found that CNN's FC6 and FC7 layers outperformed handcrafted features, achieving higher accuracy. We divided our dataset into a 70:30 ratio, using 35 images for training and 15 for testing per monument.

The "Virtual Tourist Guide" [1] is a useful resource for exploring forts in Maharashtra. It relies on a database to provide detailed information about specific forts and uses Google's landmark detection API to identify monument names. The GPS API locates nearby monuments, and the app includes emergency contact numbers for ambulance, fire brigade, police, and disaster management.

In another study, "Monument Recognition using Deep Neural Network" [3], transfer learning is applied using a pre-trained Inception model to classify famous monuments in India's Golden Quadrilateral. The dataset consists of 2600 images divided into 12 classes. The study achieved a 96% validation accuracy after 4000 iterations, tested on 20 unseen images.

However, in [1], the application is limited to providing information about monuments stored in its database. It only provides the name of the monument when identifying monuments through images, but lacks detailed information. The information provided is only in English, which may create a language barrier for some users. Additionally, the application has a limited range of classes stored in its database.

III. SYSTEM ARCHITECTURE

Various methods used to recognize monuments have been explained in this section. By examining the limitations in previous research, we encountered the problem of relying solely on a database as a source of information, which can be inconsistent and not adaptable in certain conditions. The data might be static and limited for the number of monuments. If the information about a particular monument, changes in real-time, the user cannot find valid data. To overcome this problem, we implemented the method of web scraping to obtain real-time, valid information.

Furthermore, we implemented methods of multilingual translation and conversion into a .mp3 audio file for visually impaired individuals. This allows for a more inclusive and accessible experience for users, catering to a wider audience with different language preferences and accessibility needs.

Abbreviations and Acronyms

YOLO	You only look once
CNN	Convolutional Neural Network
SELUs	Scaled Exponential Linear Unit
SPPF	Spatial Pyramid Pooling Fast
REST	Representational State Transfer

3.1 Creation and training of model

Classifying monuments presents a challenge because the images of a single monument can vary significantly in their orientations, as noted in [2]. To address this, we plan to utilize the transfer learning technique for monument recognition. In choosing a pretrained model, we consider several criteria, including task-specific performance, resource availability, and accuracy.

As described in [5], YOLOv8 demonstrates rapid inference capabilities and exceptional accuracy in detecting visual objects, particularly smaller ones. This model has outperformed transformer-based deep learning models and is considered a leading algorithm in its domain. YOLOv8 is a state-of-the-art object detection model published in 2023, developed using PyTorch and trained on the ImageNet dataset. This makes it capable of identifying various objects in an image with a high degree of accuracy.

3.1.1 What is Yolov8?

YOLO is primarily designed for object detection tasks which involve identifying and localizing objects within an image. In January 2023, Ultralytics released YOLOv8 (You Only Look Once), their latest cutting-edge computer vision model. YOLOv8 excels in speed, accuracy, and user-friendliness, making it a powerful tool for various tasks beyond object detection and tracking. These tasks include segmentation, image classification, and even pose estimation.

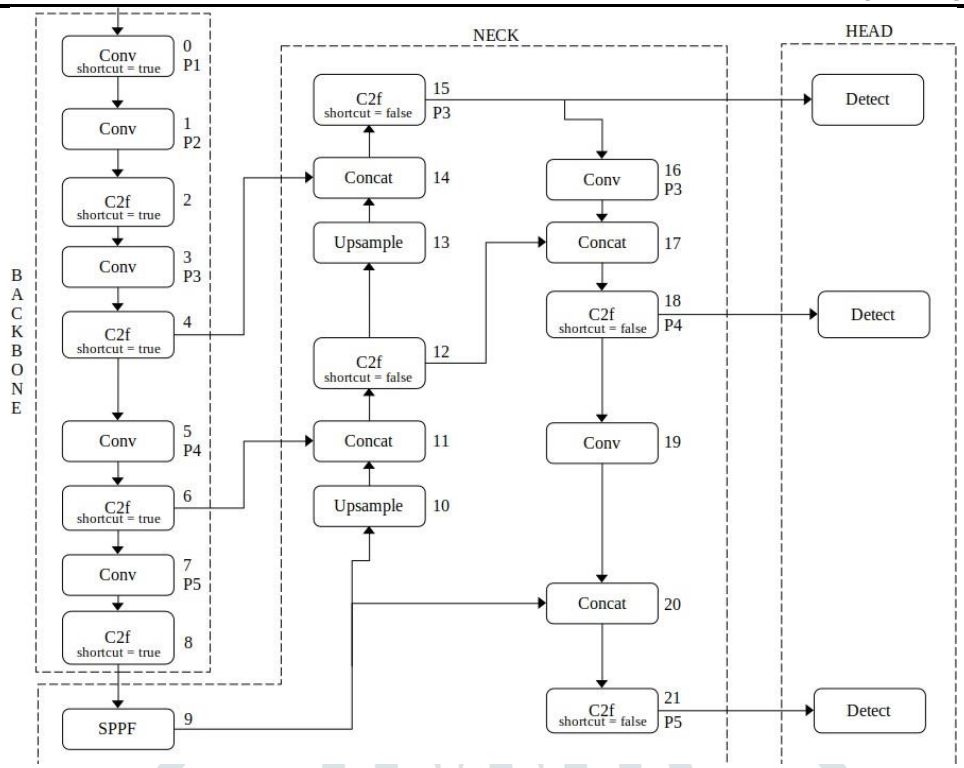

Figure 1. YOLOv8 Architecture

Building upon the strengths of prior YOLO versions, YOLOv8 boasts new features and advancements for improved performance and training efficiency. Notably, it leverages the PyTorch framework. Unlike some object detection models, YOLO belongs to the family of one-stage detectors, processing the entire image in a single pass through a convolutional neural network . YOLOv8 comprises three main components: the backbone, neck, and head, as shown in Figure 1. These components are constructed using a combination of smaller components such as Conv, C2f, Bottlenecks, SPPF, and Detects. The YOLOv8 model has been fine-tuned on a custom dataset specifically for monument classification, with training lasting for 25 epochs. The resulting model has been saved in .pt format for future use.

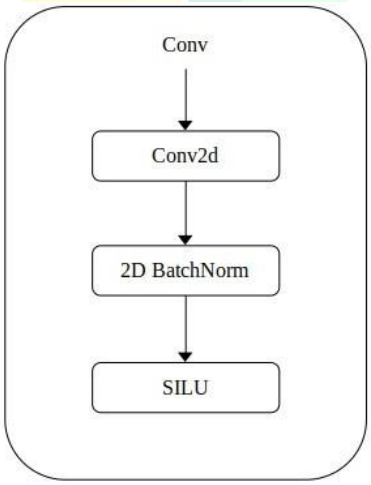

Figure 2a. Conv layer

YOLO leverages a strategy called local feature analysis to achieve real-time object detection. This method breaks down the image into smaller regions and focuses on extracting features from those areas instead of processing the entire image at once. This significantly reduces the computational workload, making real-time processing possible. Convolution, a mathematical operation that combines elements of two arrays to produce a third, forms the backbone of YOLO's feature extraction process. In the context of convolutional neural networks (CNNs) used for image analysis, convolution acts like a filter that highlights specific patterns within the image. By applying convolutions with different kernel sizes, strides, and paddings, YOLO can capture a diverse set of features from various image scales, ultimately leading to accurate object detection.Conv layer mainly consists of 2d convolutional layer, 2d batch normalization layer and SILUs activation function as mentioned in figure 2a.

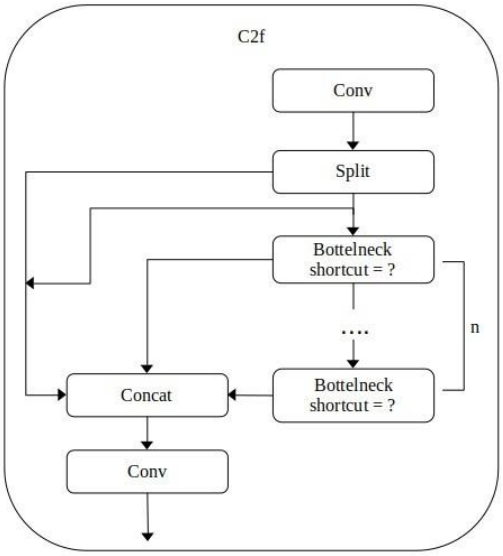

Figure 2b: C2f layer

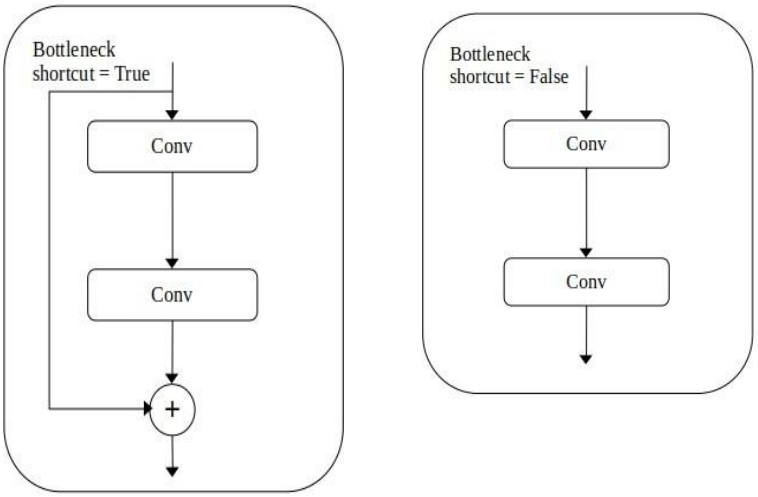

Figure 2c: Bottleneck layer

It is a building block used in the YOLOv8 architecture. The C2f module is a faster implementation of the C2 module. It improves the execution speed of the model while maintaining similar performance. This optimization is achieved by making certain modifications to the original C2 module. C2f layer contains n number of bottleneck layer as shown in figure 2b. Shortcut function of bottleneck layer can be true or false depends upon whether it is in the backbone or bottleneck layer respectively whichis shown in figure 1. The different architectures of bottleneck layer are shown in figure 2c.

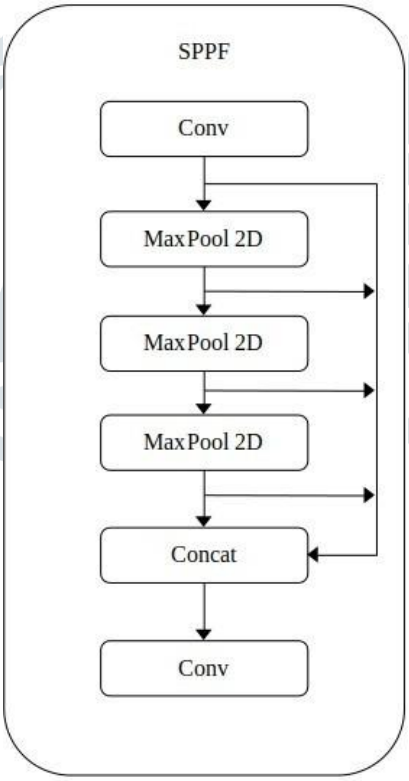

Figure 2d: SPPF layer

Figure 2d. shows internal structure of the SPPF layer in YOLOv8. This layer is designed to speed up the computation of the network by pooling features of different scales into a fixed-size feature map. SPPF structure replaces the parallel maximum pooling operation of three convolutional kernels of different sizes in the original SPPF.

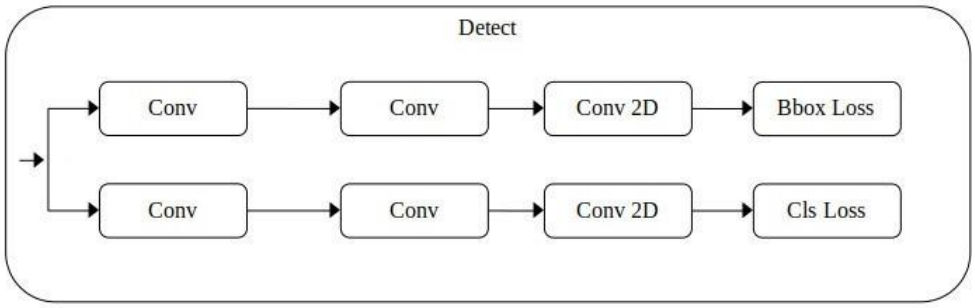

Figure 2e: Detect layer

This is the layer where detection happens. The detect box contains 2 tracks. The first box is for Bonding box prediction. The second is for class prediction as shown in figure 2e. Both tracks have same layer sequence consisting conv layer and one 2d convolutional layer.

3.1.2. Why YOLOv8?

YOLOv8 is known for its speed and efficiency, making it ideal for real-time applications. Despite its speed, YOLOv8 does not compromise on accuracy. The algorithm leverages a deep convolutional neural network (CNN) architecture that is capable of capturing complex features from images. It can give very high accuracy on least amount of data set. As explained in experimental setup section below, we got better accuracy from Yolov8 as compared to other methods.

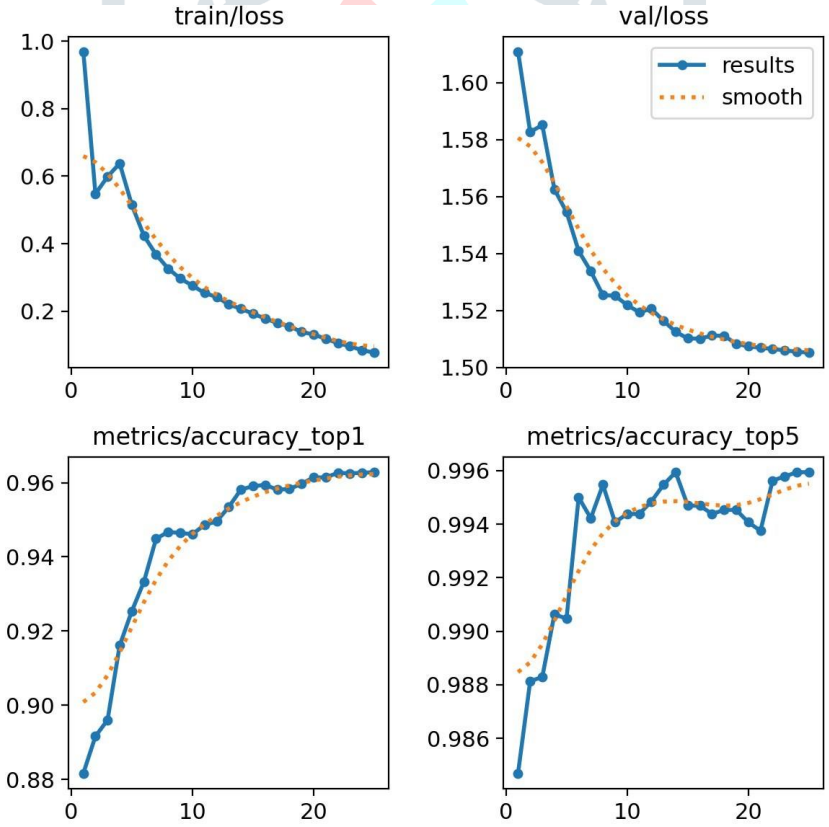

Figure 3. Graphical representation of parameter in YOLOv8.

epoch	train/loss	metrics/accuracy_top1	metrics/accuracy_top5	val/loss
1	0.96919	0.88162	0.98469	1.611
2	0.54777	0.89161	0.98813	1.5827
3	0.5988	0.89599	0.98829	1.5852
4	0.63793	0.91613	0.99063	1.5626
5	0.51527	0.92535	0.99047	1.5546
6	0.4236	0.93331	0.995	1.5409
7	0.36813	0.94487	0.99422	1.5338
8	0.32642	0.94674	0.99547	1.5254
9	0.29697	0.94643	0.99407	1.5252
10	0.27609	0.94612	0.99438	1.5219
11	0.25415	0.94862	0.99438	1.5194
12	0.24239	0.94955	0.99485	1.5206
13	0.22019	0.95346	0.99547	1.5164
14	0.20765	0.95814	0.99594	1.5125
15	0.1927	0.95908	0.99469	1.5102
16	0.17864	0.95939	0.99469	1.5101
17	0.16515	0.95814	0.99438	1.5112
18	0.15563	0.9583	0.99453	1.5111
19	0.13885	0.95971	0.99453	1.5083
20	0.13141	0.96142	0.99407	1.5074
21	0.11841	0.96142	0.99375	1.5069
22	0.10593	0.96267	0.99563	1.5065
23	0.0963	0.96252	0.99578	1.506
24	0.08467	0.96267	0.99594	1.5055
25	0.0776	0.96283	0.99594	1.5052

Table 1: Model Training results (epochs)

Figure 3. shows the graph of valuation accuracy for Monument Classification model of Sightscripter trained on Yolov8. Table shows the result produced during 25 epochs. Figure 4 shows the confusion matrix for the valuation of classes.

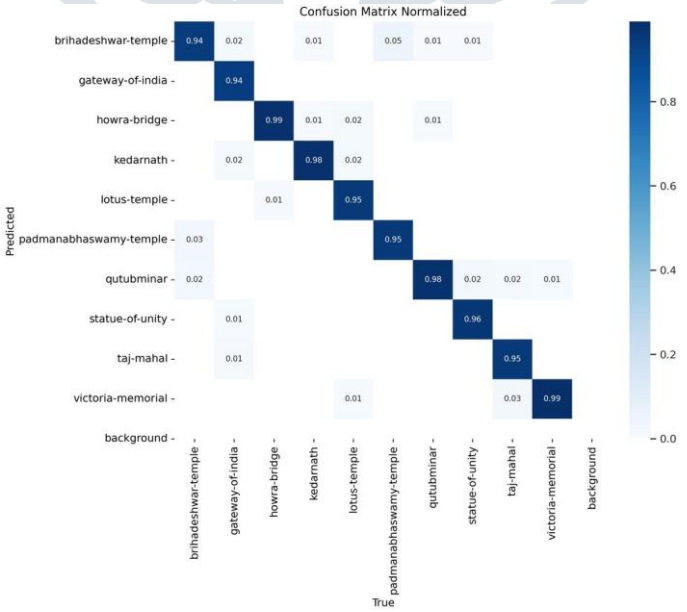

Figure 4: Confusion matrix for validation

3.2. Creation of android application and backend services:

Sight Scripter is based on a client-server architecture, where the Android application acts as the client, responsible for providing the user interface and handling user interaction. The Android application communicates with the server, which hosts theREST API. The architecture of the system is shown in figure 5.

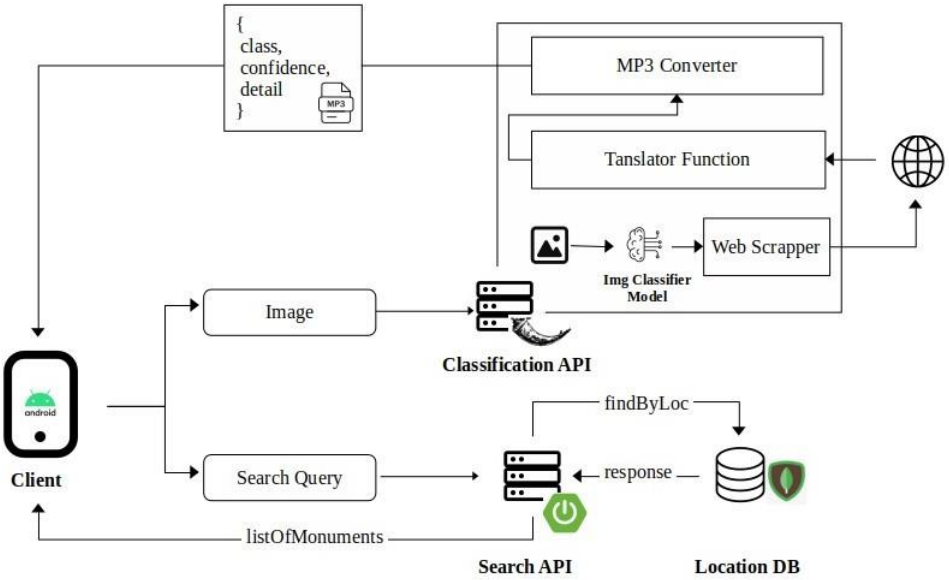

Figure 5: System Architecture

Client (Android Application)

The frontend of Sight Scripter is a mobile application that allows users to interact with the device to capture images or select images for monument recognition, in order to retrieve information about that monument in both textual and audio formats. The application also provides a search function to search for monuments based on state, city, and name.

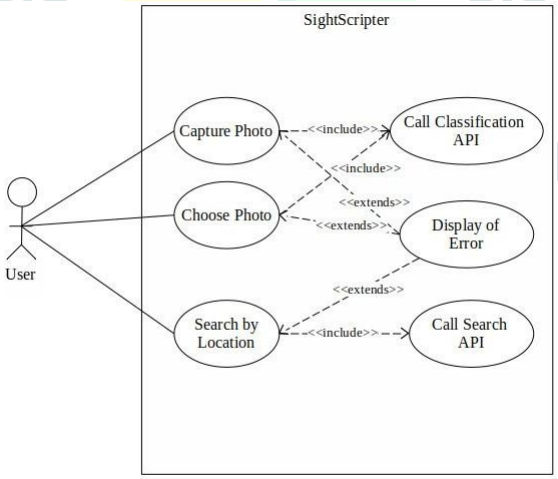

Figure 6: Use case diagram

Figure 6 demonstrates use case diagram for the sight-scripter mobile application. The user has the option to either "Capture Photo" or "Choose Photo". Based on the user's selection, the flowchart diverges into two paths: Capture Photo: If the user captures a photo, the image is sent to a "Call Classification API". Choose Photo: If the user chooses a photo, the process seems to be identical to capturing a photo.

Server (REST API)

The server handles requests from the client and serves as an intermediary between the mobile application and backend services. Sight Scripter consists of two REST APIs: one for classification and the other for search operations. The classification API is implemented using Flask and is responsible for recognizing monuments from images by utilizing YOLOv8. It collects information about the monument, web scrapes, translates it to the required language, and then creates an audio file from it. It sends the response back to the client in both textual and audio formats. On the other hand, the search API is implemented using SpringBoot, which is responsible for retrieving a list of monuments from the database based on the location provided by theclient.

These components work together to provide a seamless user experience for exploring Indian monuments. The Android application provides an intuitive interface for users to capture images, search for monuments, and receive detailed information in both textual and audio formats. The server, with its REST APIs, handles the processing of requests, classification of images, and retrieval of monument information, ensuring a smooth and efficient user experience.

Flow of application :

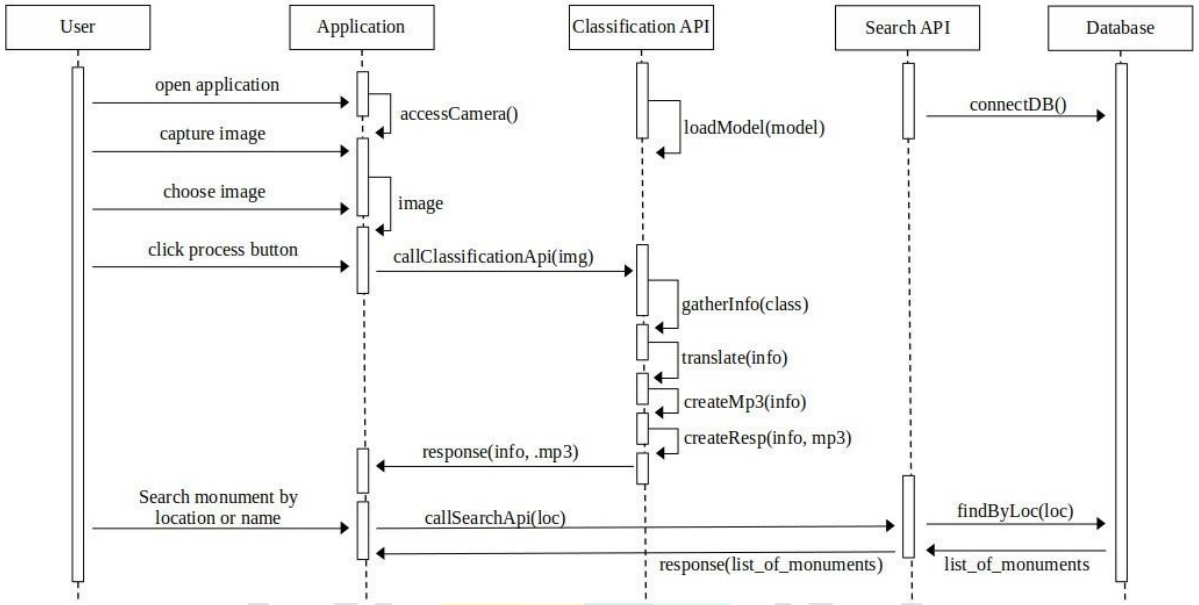

Figure 7: Sequence Diagram

The figure 7. sequence diagram starts with the user opening the application. The user can then either capture a new image using the application's camera or choose an existing image. Once the user has an image, they can click a button to process the image. This triggers the application to call a function called callPredictApi, which presumably sends the image to a server-side API for processing. The API then uses a loaded model, to process the image and identify the landmark in the image. Once the landmark is identified, the API calls another function called gatherInfo, which gathers information about the landmark from a database. The information is then translated and used to create an MP3 file, possibly containing information about the landmarkin an audio format. Finally, the API sends a response back to the application, which includes the information about the landmark and the MP3 file. The user can also search for monuments by location or name. This triggers the application to call another function called callSearchApi, which sends the search query to a server-side API for processing. The search API then uses a function called findByLoc to find monuments based on the user's search criteria. The API then sends a response back to the application, which includes a list of monuments.

IV. Experimental Setup

The following section outlines the methodology used for data collection, augmentation, and the experimental scenarios employed to test the performance of the model.

4.1 Data Collection and Augmentation

The dataset was manually collected from various online sources, comprising 10 classes. To expand the dataset, the Augmenter Python package was utilized due to the initial dataset's small size, containing only 200 images per class. The dataset was then divided into three segments: training (160 images), validation (20 images), and testing (20 images). The testing images were left unaltered.

Post-augmentation, the training dataset was augmented to 6000 images, aiming to enhance the model's generalization during training.

4.2 Experimental Scenario

Various methods and researches were explored to minimize loss and improve accuracy. Initially, a custom CNN model with three convolutional layers was implemented, resulting in a validation accuracy of 80.42%. To address overfitting, Batch Normalization (BN) and Dropout were incorporated.

Subsequently, an ensemble learning approach was adopted, combining two models with varying accuracies to achieve an improved accuracy of 11.30%. The ResNet50 model was then implemented, yielding an accuracy of 85.42%. This was followed by the VGG16 model, which achieved an accuracy of 11.61%. As shown in figure 8 and table 2.

Model	**Accuracy (%)**
CNN	83.92
Ensemble	11.30
RestNET50	85.42
VGG16	11.61
YOLOv8	96.30

Table 2. Comparison of Accuracies

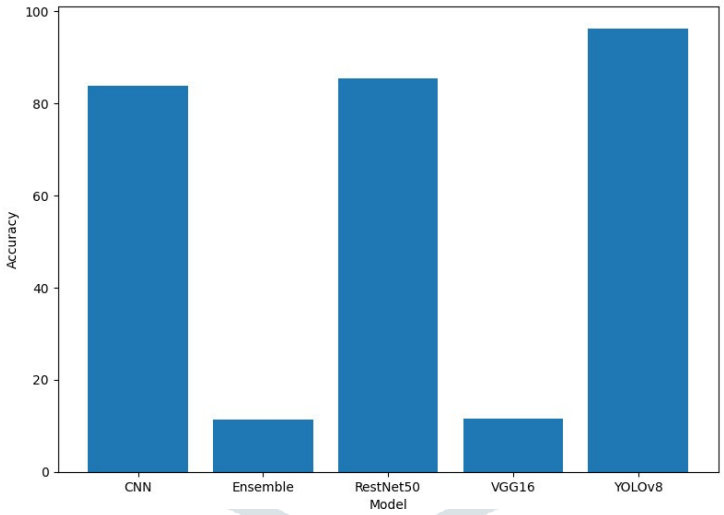

Figure 8. Graphical representation of accuracies

Finally, the YOLOv8 model was implemented, leveraging transfer learning from the ImageNet dataset. This resulted in a validation accuracy of 96.30%. YOLOv8 was identified as the superior model for image classification compared to the other models.

4.3 YOLOv8 arg.yaml Configuration for Classification

Task Configuration:

task: classify - This essential directive specifies that the model's purpose is image classification, not object detection. YOLOv8 can be adapted for both tasks, and this setting ensures the appropriate training regimen.

Model Selection:

model: yolov8n-cls.pt - This line indicates the pre-trained model you'll be using as a foundation. Here's a breakdown, yolov8n: This refers to a specific YOLOv8 model variant, likely a smaller or faster version designed for efficiency. The exact variant (e.g., yolov8s, yolov8m, yolov8l, etc.) depends on your choice and computational resources.

-cls.pt: This suffix signifies that the pre-trained model has been modified to include a classification head. This head is crucial for converting the model's output from bounding boxes (object detection) to class probabilities (classification).

Training Parameters:

epochs: 25 - This setting defines the number of times the entire training dataset will be passed through the model during training. A typical range for epochs is 20-100, but the optimal value depends on your dataset size and complexity.

batch: 16 - This parameter determines the number of images processed by the model simultaneously in each training iteration (batch). A larger batch size can improve training speed but might require more graphics processing unit (GPU) memory. Experiment to find a balance between speed and memory constraints.

imgsz: 64 - This line specifies the size to which images will be resized before being fed into the model. YOLOv8 models are designed to work with specific input image sizes. The provided value of 64 suggests you might be using a smaller YOLOv8 variant like yolov8n that can handle lower resolutions efficiently.

optimizer: auto - This setting lets YOLOv8 automatically select an appropriate optimizer algorithm (e.g., Adam, SGD) based on common practices for this type of task.

learning_rate (not explicitly shown but inferred from lr0 and lrf): The learning rate controls how much the model's weights are updated in each training iteration. It's crucial to set it appropriately to avoid getting stuck in local minima or overfitting. The arg.yamal file likely defines lr0 (initial learning rate) and lrf (final learning rate) to control the learning rate schedule.

augment: true (assumed based on common practices): Data augmentation is a technique where the training data is artificially modified (e.g., random cropping, flipping, color jittering) to create more variations and improve the model's generalization ability.

Other Important Considerations:

Validation (val section): While not explicitly shown in your example, the arg.yamal file likely includes a val section to define how the model's performance is evaluated during training. This typically involves splitting the dataset into training and validation sets, and periodically measuring the model's accuracy on the validation set to monitor progress and avoid overfitting.

Loss function: YOLOv8 employs a loss function tailored for classification tasks (e.g., cross-entropy loss) to calculate the error between the model's predictions and the ground truth labels. The specific loss function is usually defined within the YOLOv8 codebase.

V. RESULTS

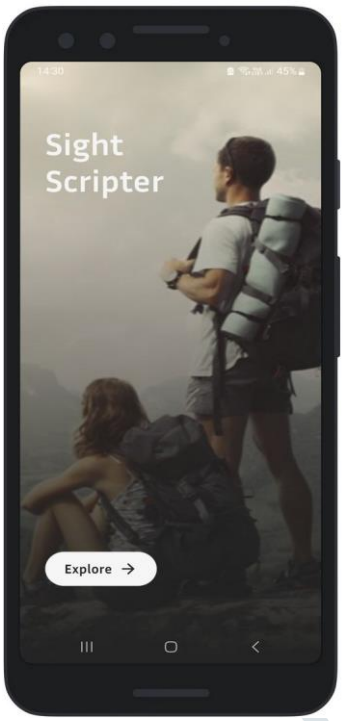

Figure 9.a Layout1

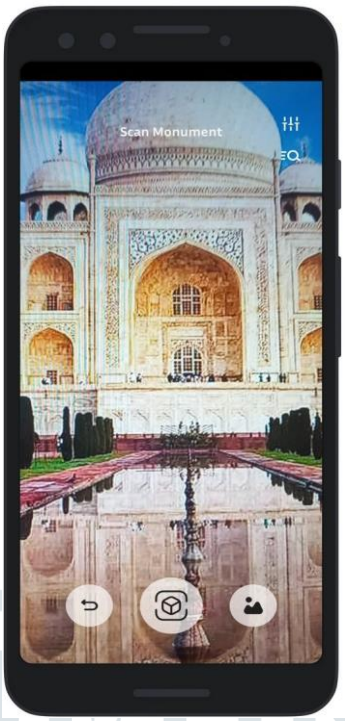

Figure 9.b Layout2

Figure 9.c Layout3

Figure 9.a showcases the homepage of Sight Scripter, depicting a user-friendly interface designed for effortless navigation. In Figure 9.b and 9.c, the application's functionality for capturing and classifying monuments is displayed. Upon capturing an image, the application utilizes the YOLOv8 model for real-time monument classification. The subsequent response from the API is then presented in the form of detailed information on the following page, ensuring users have access to comprehensive knowledge about the identified monument.

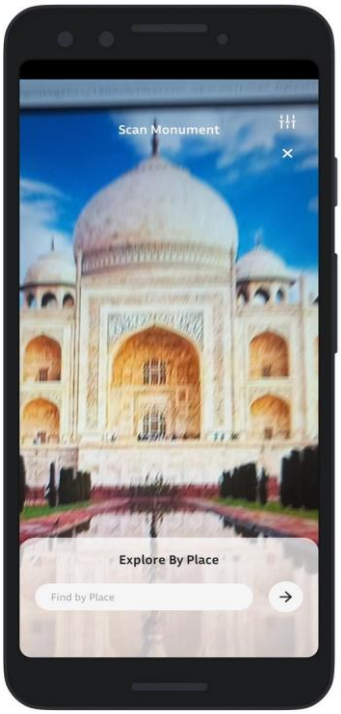

Figure 9.d Layout4

Figure 9.e Layout5

Figure 9.f Layout6

In Figure 9.d, the search feature of Sight Scripter is depicted, enabling users to search for monuments based on their location, city, or name. Figure 9.e illustrates the outcome of the search, displaying a list of monuments in nearby places or cities. Similar to Figure 9.c, Figure 9.f showcases the detailed information of a selected monument from the list, providing users with a comprehensive understanding of the monument's significance.

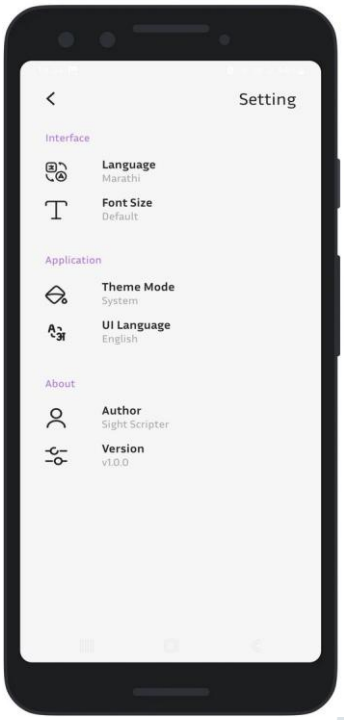

Figure 9.g Layout7

Figure 9.h Layout8

Figure 9.i Layout9

In Figure 9.g, the settings page of Sight Scripter is displayed, offering various functionalities. One of these features is the language translation capability, as depicted in Figure 9.h. This functionality is responsible for translating the information, as shown in Figure 9.i, allowing users to access the content in their preferred language.

VI. CONCLUSION

In conclusion, Sight Scripter is an innovative Android application that leverages state-of-the-art machine learning and natural language processing techniques to enhance the user experience of exploring Indian monuments. The application provides real- time monument classification, detailed information about each monument, and a powerful search feature. Through its intuitive interface and innovative capabilities, Sight Scripter aims to revolutionize the way users engage with Indian monuments, making cultural exploration more interactive, informative, and enjoyable.

REFERENCES

[1] Shila Jawale, Sakshi Jadhav, Priyanka Jaybhaye and Nikita Sonavale. 2022. Virtual Tourist Guide. IJARSCT Volume 2 Issue 3 ISSN (Online) 2581-9429.

[2] Aradhya Saini, Tanu Gupta, Rajat Kumar, Akshay Kumar Gupta, Monika Panwar and Ankush Mittal. 2017. Image based Indian Monument Recognition using Convoluted Neural Networks. IEEE.

[3] Siddhant Gada Viraj Mehta Karan Kanchan Chahat Jain and Prof. Purva Raut. 2017. Monument Recognition using Deep Neural Networks. IEEE.

[4] Rucha Dandavate, Vineet Patodkar. 2020. CNN and Data Augmentation Based Fruit Classification Model. IEEE. 10.1109/I-SMAC49090.2020.9243440.

[5] Zhou, H., Nguyen, M., Yan, W.Q.2024. Computational Analysis of Table Tennis Matches from Real-Time Videos Using Deep Learning. In: Yan, W.Q., Nguyen, M., Nand, P., Li, X. (eds) Image and Video Technology. PSIVT 2023. Lecture Notes in Computer Science, vol 14403. Springer.

[6] Tripathy, S., Singh, R. 2022. Convolutional Neural Network: An Overview and Application in Image Classification. In: Poonia, R.C., Singh, V., Singh Jat, D., Diván, M.J., Khan, M.S. (eds) Proceedings of Third International Conference on Sustainable Computing. Advances in Intelligent Systems and Computing, vol 1404. Springer .

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114.

[8] Haonan Chen, Junmei Guo, Jason Gu, Haiying Liu, Xuehu Duan, and Haitong Lou. 2023. DC-YOLOv8: Small Size Object Detection Algorithm Based on Camera Sensor. 12(10), 2323; https://doi.org/10.3390/electronics12102323.

[9] Abdelkarim Belkhir, Manel Abdellatif, Rafik Tighilt, Naouel Moha, Yann-Gaël Guéhéneuc and Éric Beaudry. 2019. An ObservationalStudy on the State of RESTAPI Uses in Android Mobile Applications. IEEE 10.1109/MOBILESoft.2019.00020.

[10] Dileep Adabala, Sumit Kaushik. 2016. A Virtual Guide for Tourist Assistance. IJRAR Volume 3 Issue 2 ISSN 2349-5138.

[11] Vidhi Singrodia, Anirban Mitra and Subrata Paul. 2019. A Review on Web Scrapping and its Applications. IEEE. 10.1109/ICCCI.2019.8821809.

[12] L. A. E. Gómez, 2018, Realities and problems of a major cultural tourist destination in Spain, Toledo, [Online] Avaliable: https://ojsull.webs.ull.es/index.php/Revista/article/view/1268/pdf .