National College of Engineering

(Affiliated to Tribhuvan University)

A

MINOR PROJECT PROPOSAL

ON

"MONUMENT RECOGNITION USING MOBILE APPLICATION"

Submitted By:		Submitted To:
Aayush Shrestha	077BCT002	DEPARTMENT OF COMPUTER AND
Ishita Chalise	077BCT012	ELECTRONICS ENGINEERING
Pradish Tamrakar	077BCT019	

TALCHIKHEL, LALITPUR

SHRAWAN,2080

ACKNOWLEDGEMENT

This project is being prepared in partial fulfillment of the requirements for the Bachelor's degree in Computer Engineering. We extend our sincere gratitude to **Er. Subash Panday**, our esteemed project supervisor, for his unwavering guidance, inspiring lectures, and invaluable encouragement. His constant support and insightful suggestions have been instrumental in making this journey a rewarding one. Without his expertise, this project would have been a far more challenging.

We would like to express our gratitude to the Department of Computer and Electronics Engineering at National College of Engineering for providing us with the opportunity to embark on this collaborative project. This experience has allowed us to apply the knowledge acquired over the years as a minor project for the third year's second part, fostering both personal and professional growth. The chance to develop our own minor project has been a significant milestone, deepening our understanding and enhancing our teamwork skills.

Finally, we would like to acknowledge everyone who played a role directly or indirectly for their invaluable help.

We welcome any suggestions or criticisms for further improvement.

Abstract

The Monument Recognition System Using Mobile Application is a mobile app created to address the need for information about Nepal's diverse monuments. Preserving the historical information associated with heritages and monuments presents a major challenge in modern Nepal. Affected by decay and loss, original writings and papers are deteriorating, and the loss of priceless historical information is made worse by a lack of focused preservation measures. This mobile application aims to bridge this historical gap by providing a platform for exploring and engaging with information about Nepal's various monuments. The dataset comprises of 2000 images, covering 100 photos for each side of five distinct monuments sourced from Patan Durbar Square. To enhance diversity and make the dataset' robustness, augmentation techniques such as zooming, flipping, rotation, & shearing was used expanding the dataset to 12,000 images. Each image was resized to 128 x 128 pixels, and split into training set and validation set in the ratio of 7:3 respectively. Preprocessing involved normalizing pixel values, introducing shading, adding noise, and grayscale conversion. The Convolutional Neural Network (CNN) architecture, tailored specifically for this application, attained training accuracy of 96.69% and validation accuracy of 98.79%, with a F1 score of 0.9875. The Monument Recognition System Using Mobile Application addresses historical gaps in Nepal, by providing a platform to explore information about various monuments.

Keywords: *Convolutional Neural Network, Deep Learning, Machine Learning, Mobile App, Monument Recognition*

Table of Contents

List of Figures

List of Tables

List of Abbreviations

APF	Adaptive Project Framework
API	Application Programming Interface
BN	Batch Normalization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
CVPR	Computer Vision and Pattern Recognition Conference
DCNN	Deep Convolutional Neural Network
GPS	Global Positioning System
GPU	Graphical Processing Unit
HAM10000	Human Against Machine with 10000 training images
HOG	Histogram of Oriented Gradients
IoT	Internet of Things
JSON	JavaScript Object Notation
LBP	Local Binary Pattern
NiN	Network in Network
ORB	Oriented FAST and Rotated BRIEF
ReLU	Rectified Linear Unit
SDLC	Software Development Life Cycle
SVM	Support Vector Machine
VGG	Visual Geometry Group
XP	Extreme Programming

1. Introduction

1.1. Background

The Monument Recognition System addresses the pressing need for preserving historical information about Nepal's monuments. In modern Nepal, the original writings and papers related to these monuments are deteriorating, leading to the loss of invaluable historical insights. The lack of focused preservation measures exacerbates this issue. To overcome these challenges, this mobile app provides a platform for users to explore and engage with information about Nepal's diverse monuments. The initiative is crucial for bridging historical gaps and ensuring the accessibility of heritage details that are at risk of decay and loss.

The foundation of the app lies in a carefully curated dataset comprising 2000 images, representing five distinct monuments from Patan Durbar Square. To enhance the diversity and robustness of the dataset, advanced augmentation techniques such as zooming, flipping, rotation, and shearing were employed, resulting in an expanded dataset of 12,000 images. These images were resized to 128 x 128 pixels and further underwent preprocessing steps, including normalization of pixel values, shading, noise addition, and grayscale conversion. The meticulous attention to data quality and diversity contributes to the effectiveness of the Convolutional Neural Network (CNN) architecture, which was tailored specifically for this application.

The CNN model demonstrated remarkable performance, achieving a training accuracy of 96.69% and a validation accuracy of 98.79%. Notably, the F1 score, a metric encompassing both precision and recall, reached a high value of 0.9875. These results affirm the model's efficacy in recognizing and classifying monuments. Beyond the technical aspects, the Monument Recognition System Using Mobile Application serves a broader purpose. By providing a user-friendly platform, it successfully addresses historical gaps in Nepal, enabling users to access and appreciate the historical significance of various monuments. This academic paper highlights the significance of the app in the preservation and dissemination of cultural heritage, emphasizing its potential impact on historical awareness and education.

1.2. Problem Statement

In present scenario, preserving historical insights into Nepal's heritage and monuments faces significant challenges due to decay and a lack of focused preservation measures. Original writings are deteriorating, leading to a loss of priceless historical information. The current availability of such accounts is limited to print resources, making them prone to deterioration and inaccessible in local areas. This solution aims to make historical information accessible to both locals and tourists, exceeding geographical boundaries and bridging the chronological gap between the past and present. This transformative initiative not only protects and disseminates Nepal's rich cultural heritage but also fosters a broader appreciation across generations and borders.

1.3. Aims and Objectives

The aim of project is to develop a Monument Recognition System Using Mobile Application.

The objectives are:

- To collect and create a diverse dataset of images of monuments.
- To design and train a CNN model using the collected dataset.
- To design and develop mobile applications using flutter.
- To recognize the monuments using the trained model.

1.4. Scope

The Monument Recognition System Using Mobile Application has a broad scope that combines technology and cultural preservation. The program is designed to help people easily learn about and explore information on Nepal's monuments. It addresses the challenges of losing historical records due to decay by using advanced technologies. With a large dataset of images from Patan Durbar Square, the program goes beyond basic image recognition. It uses Convolutional Neural Network (CNN) to accurately identify monuments. The program's goal is not only to excel in technology but also to make a real impact by preserving and sharing cultural heritage. It aims to raise awareness and contribute to education about Nepal's diverse monuments, acting as a modern solution to document and appreciate the country's rich history.

2. Literature Review:

2.1. Related Theory:

2.1.1. CNN

An exceptionally effective neural network architecture for processing and analyzing visual data is called a convolutional neural network (CNN). It does this by utilizing specialized layers that automatically recognize and learn visual elements from the input photos, such as edges, textures, and shapes. The network can identify local patterns and combine them into higher-level representations as to the convolutional layers' use of filters that scan throughout the input. In tasks like image classification, where CNNs can precisely identify objects within images, this hierarchical method has led to spectacular accomplishments, making CNNs a pillar technology in contemporary computer vision applications.

2.1.2. Data Augmentation

By making controlled alterations to the original data, such as cropping, resizing, or rotating photos, or subtly changing text, data augmentation creates fresh training instances. This method aids the model in learning from a wider range of possible outcomes, ultimately enhancing its accuracy and resilience. Data augmentation reduces overfitting and improves the model's capacity to generate precise predictions on unknown or varied inputs by artificially extending the dataset.

2.2. Related Works

"Landmark Recognition Using Machine Learning" by Andrew Crudge et al. [1] presents detection of buildings in image, the image is cropped into multiple overlapping cells with identical aspect ratios and features are extracted from the cells using the HOG descriptor and classified using the SVM algorithm. The proposed model used dataset consisting of 193 images of various buildings collected from Google Images. To improve performance of model, each classifier was run 20 times varying the training and test sets by randomly permuting the dataset. The accuracy of the SVM classifier approaches 90%. "Monument Recognition using Deep Neural Networks" by Siddhant Gada et al. [2] proposes the concept of Transfer learning which has been used to prune the computational load, dataset of about 400 images per monument were used to retrain the final layer of the Inception model. The model is tested on a few arbitrary images

and results with a training accuracy of 99.4% and corresponding testing accuracy ranging from 96-99%.

"Cross Platform Web-based Smart Tourism Using Deep Monument Mining" by Mehdi Etaati et al. proposed platform uses a deep neural network (VGGNet model; transfer learning technique is used) for feature extraction from the images captured on a mobile phone and a classification algorithm: SVM and Random Forest are used for identification of monuments in the image. image uploaded to the web server detects the monument in the image, extracts its information from the database, and sends the results to the device [3]. "Towards Deep Learning for Architecture: A Monument Recognition Mobile App" by Valerio Palma et al. [4] uses data augmentation techniques on the dataset of roughly 50–100 images per monument to generate 500 training images per monument which is trained using CNN algorithm based on the MobileNet model implemented in Python using the open-source libraries TensorFlow and Keras. A commercial iOS app was developed with overall accuracy of the trained models estimated to be over 95%.

"Mobile Travel Guide using Image Recognition and GPS/Geo Tagging A Smart Way to Travel" by Ramsha Fatima et al. project uses Open-CV for image recognition and GPS navigation system with Google Maps API to point the location of that monument on the map. The information associated with monument is available in the JSON database. To identify the monument, the camera's captured image is compared to images stored in the database using Open-CV. The application compares two images using histogram comparison and feature detection using the ORB method [5].

"Image based Indian Monument Recognition using Convoluted Neural Networks" by Aradhya Saini et al. experiments on different method of extraction of features using hand crafted features like HOG, LBP and GIST and then CNN. The manually acquired dataset comprises of 100 different monuments with 50 images per monument. HOG, LBP and GIST features extracted on training dataset classified by classification algorithms like SVM obtained very low accuracy. Finally, DCNN model obtained accuracy of 92.7% using fc6 layer [6].

Table 1: Study of research works

S. N	Title	Summary

1	Landmark Recognition Using Machine Learning	In [1], feature extractor; HOG descriptor and classifier; SVM algorithm is used on 193 training images with model accuracy of 90%
2	Monument Recognition using Deep Neural Networks.	In [2], Transfer learning has been used only retraining the final layer of the Inception model with 400 images per monument. The model is tested on a few arbitrary images and results with a training accuracy and testing accuracy of 99.4% and 96-99% respectively.
3	Cross Platform Web-based Smart Tourism Using Deep Monument Mining	In [3], feature extraction: (VGGNet model; transfer learning technique is used) and classification algorithm: SVM and Random Forest are used for identification of monuments in the image.
4	Towards Deep Learning for Architecture:	[4] uses CNN algorithm based on the MobileNet model trained on 500 images per monument to develop a commercial iOS app with overall accuracy ofthe trained models estimated to be over 95%.
5	Mobile Travel Guide using Image Recognition and GPS/Geo Tagging a Smart Way to Travel	[5] employs Open-CV for image recognition and a GPS navigation system with Google Maps API to pinpoint the location of the monument on a map. The application compares two images (the camera's captured image and images stored in the database) using histogram comparison and feature detection using the ORB method.
6	Image based Indian Monument Recognition using Convoluted Neural Networks	[6] experiments on different method of extraction of features using hand crafted features and CNN. The models trained on manually obtained datasetof 50 images per monument accuracy of 92.7% on DCNN using fc6 layer.

3.Methodology

3.1 Block diagram

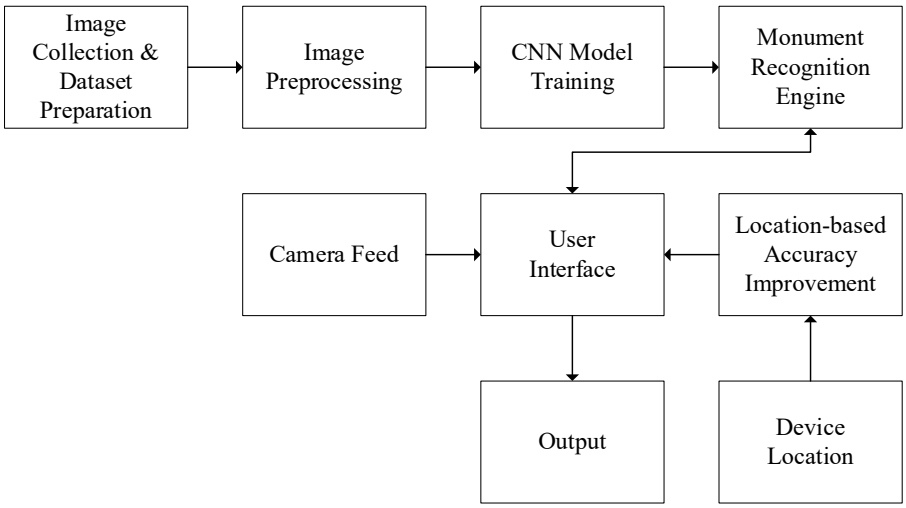


Figure 1: Block Diagram of Monument Recognition System

Description:

1. **Image Collection and Dataset Preparation:**

- This component is responsible for gathering a diverse dataset of images of monuments and heritage sites.
- Images can be collected from various sources, such as the web, public image repositories, and on-site photography.

2. **Image Preprocessing:**

- Before feeding images into the model, preprocessing techniques are applied to enhance the quality and consistency of the dataset.
- Preprocessing may include resizing images to a uniform size, normalizing pixel values, and handling color variations.

3. **CNN Model Training:**

- The image dataset is used to train the Convolutional Neural Network (CNN) model for monument recognition.
- During training, the model learns to identify unique features and patterns in monument images.

4. **Monument Recognition Engine:**

- This is the core engine of the application that performs real-time monument recognition.
- The engine takes the camera feed as input and applies the trained CNN model to identify the monument in the captured frame.

5. **User Interface (UI):**

- The user interface is the visual representation of the application that users interact with.
- It includes the camera viewfinder for real-time capturing and the output display to show the recognized monument and historical information.

6. **Location-based Accuracy Improvement:**

- This module compares the device's live location coordinates (obtained through GPS or other location services) with the known coordinates of monuments.
- This comparison enhances the accuracy of the identified monument, ensuring that the closest monument is displayed.

7. **Device Location Service:**

- This component captures the real-time location of the user's device using GPS or other location services.
- The device's location is used for comparing with the monument coordinates and for improving the accuracy of the recognition process.

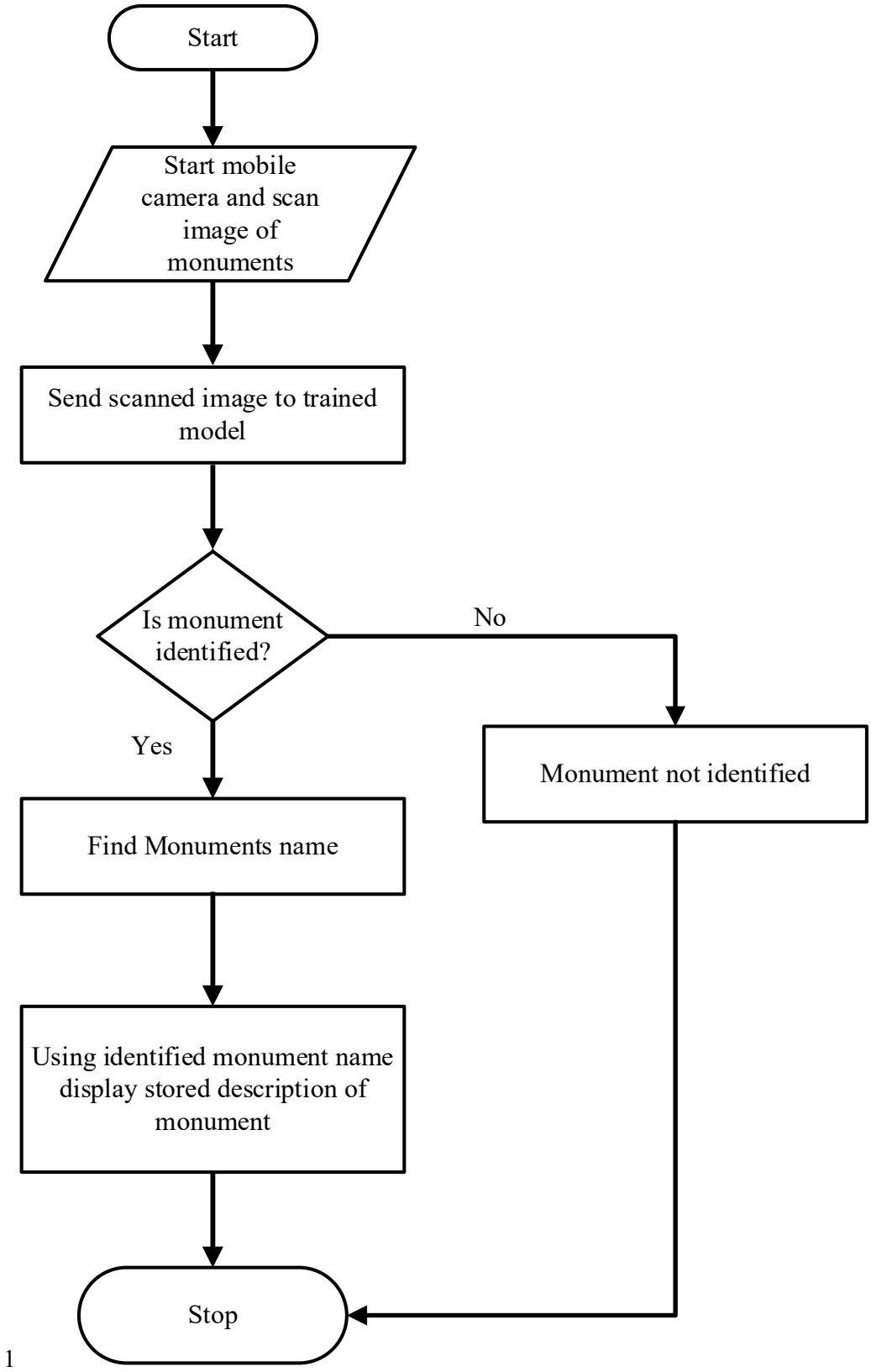
8. **Camera Feed and Real-time Image Processing:**

- The camera feed is the input stream of images captured by the user's device camera.
- Real-time image processing is performed to prepare the images for monument recognition.

9. **Output and Historical Information Display:**

- The output component displays the results of the monument recognition process.
- Once a monument is identified, the relevant historical information is displayed on the user interface.

3.2 Working principle of Monument Recognition System:



1

Figure 2: Flow Diagram of Monument Recognition System

The provided flowchart outlines the process of monument recognition and historical information display in a mobile application. Here's a brief description of each step:

1. **Start**

2. **Start Camera and Scan Image of Monuments:**

- The process begins by initiating the device's camera to capture live frames.
- Users point the camera at a monument they want to identify.

3. **Send Scanned Image to Trained Model:**

- The captured image is sent to the trained Convolutional Neural Network (CNN) model for monument recognition.
- The model processes the image to identify the monument.

4. **Is Monument Identified?**

- The application checks whether the model successfully recognized the monument in the image.

4.1. **If No - Monument Not Identified:**

- If the model fails to recognize the monument, the application informs the user that the monument could not be identified.
- The user may try again or seek alternative ways to obtain information.

4.2. **If Yes - Find Monument Name:**

- If the model successfully identifies the monument, the application proceeds to find the name of the recognized monument.
- The name is determined based on the model's classification result.

5. **Using Identified Monument Name and Display Stored Description:**

- With the identified monument's name, the application retrieves the corresponding historical information stored in the database.
- The historical description, along with relevant details, is displayed on the user interface.
- Users can access historical information about the recognized monument.

6.**Stop**

3.3 Dataset

For our project, we have carefully curated a dataset showcasing the monuments of Patan Durbar Square. We captured a total of 100 photos for each side (East, West, North, and South) at each of the five monuments: "Chaar Narayan Mandir," "Chyasin Dega," "Krishna Mandir," "Shree Bhimsen Mandir," and "Vishwanath Mandir."

All these photos were taken on-site with the goal of creating a diverse dataset that highlights the unique features of each location from various angles. Having 100 photos for each side provides us with a robust dataset, serving as the foundation of our project.

3.4 Preprocessing

In the preprocessing phase, we implemented several key steps to enhance the quality of our dataset and prepare it for effective model training. Data augmentation techniques were applied to diversify the dataset, incorporating random rotations, flips, shear, and zoom. To facilitate model training and evaluation, we split the dataset into training and validation sets, with the latter serving as a measure of the model's performance on previously unseen data.

To ensure consistency in input dimensions, all images were resized to a standardized dimension of 64x64 pixels. Subsequently, normalization of pixel values was performed, scaling the data to a range of [0, 1]. To simulate real-world lighting conditions, shading effects were introduced. Additionally, a controlled amount of noise was added to a subset of images, replicating imperfections commonly found in photographs. This step aids in training models to handle noisy input, contributing to improved overall performance.

As a further simplification, the dataset underwent grayscale conversion, transforming images into black and white. This allows models to focus on essential features and patterns during training. These preprocessing steps collectively contribute to the creation of a robust dataset, marking a crucial phase in the development of our models.

3.5 System Requirements

3.5.1. Hardware Requirements

A computer system with the following specifications

- CPU: Intel Core i5 (10th generation or newer) or AMD Ryzen 5 (4000 series or newer)
- Installed memory (RAM): 8 GB or above

- Storage Device: SSD (500 GB or above)
- GPU: CUDA enabled GPU (NVIDIA GTX 3050)

An Android OS based mobile phone with the:

- Octa core CPU or above
- RAM: 4GB or above
- ROM: 64GB or above
- Camera: 12 Megapixels or above
- Android Version 12.0 or newer

3.5.2. Software Requirements

- Python
- Android Studio
- Flutter
- Dart
- TensorFlow
- TensorFlow Lite
- Keras
- sqflite

3.6 Algorithm

3.6.1 Convolutional Neutral Network

A class of artificial neural network called a convolutional neural network, sometimes referred to as CNN or ConvNet, is mostly used for image processing. By utilizing convolutional layers, it can recognize patterns in images, aiding in image analysis. Input and output layers, as well as one or more hidden layers, are present in CNN, like most neural networks. In CNN, the hidden layers read the inputs from the input layer then perform convolution on the input values. Convolution in this context denotes a dot product or matrix multiplication. The Rectified Linear Unit (ReLU), a nonlinearity activation function, is used by CNN after matrix multiplication, followed by additional convolutions like pooling layers. In order to minimize the dimensionality of the data, pooling layers compute the outputs using functions like maximum pooling or average pooling. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

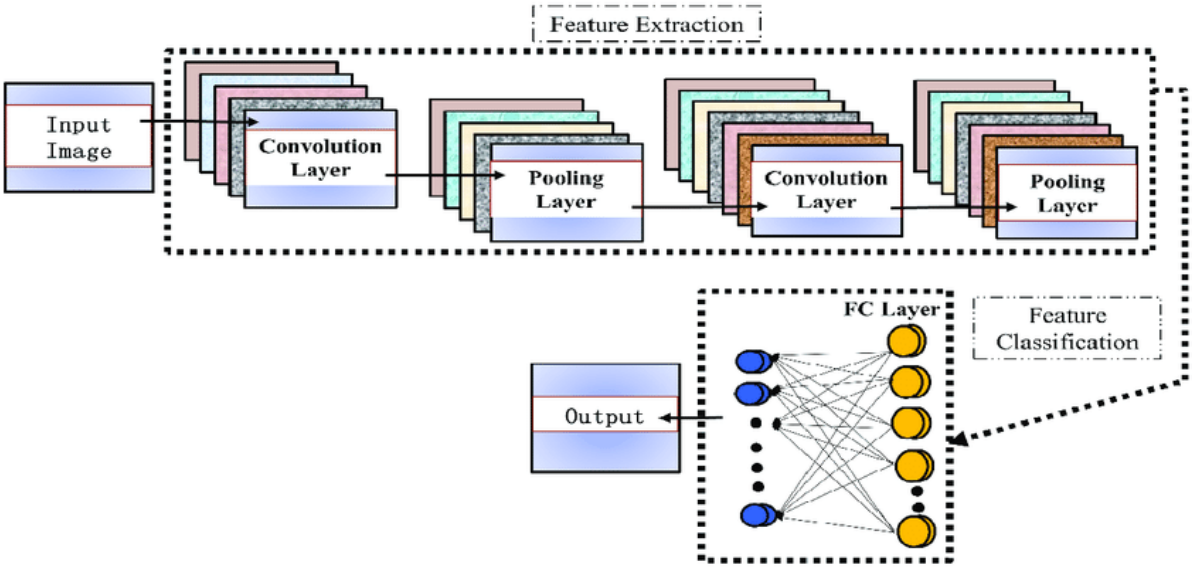


Figure 3: Architecture of CNN (Source: https://www.researchgate.net/figure/Structure-of-CNN-There-are-some-special-structural-features-in-the-CNN-architecture_fig1_344807764)

A convolutional layer is the main part of CNN model that performs dot product between two matrices where one matrix is the filter/kernel which is set of learnable parameters and the other matrix is the matrix representation of image. The feature detector, also known as a kernel or a filter, which moves across the receptive fields of the image, to extract the feature present in image. This process is known as a convolution. This helps for pattern recognition. During the convolution operation the filter slides across the height and width of the image two-dimensional representation of the image known as activation map is generated which gives response of the kernel at corresponding spatial position of the image. The sliding size of the kernel is called a stride. Pooling is down-sampling in order to reduce the complexity for further layers. It can be compared to reducing the resolution when it comes to image processing. Pooling has no effect on the number of filters. One of the most popular kinds of pooling techniques is max-pooling. It partitions the image to sub-region rectangles, and it only returns the maximum value of the inside of that sub-region. Down-sampling does not preserve the position of the information. Therefore, it should be applied only when the presence of information is important. Moreover, pooling can be used with non-equal filters and strides to improve the efficiency. The fully-connected layer is a similar to the way that neurons are arranged in 16 a traditional neural network. Therefore, each node in a fully-connected layer is directly connected to every node in both the previous and in the next

layer. It is a part of classification layer which process the final classification output. There can be no convolutional layers after a fully connected layer.

3.7 Model Architecture

The architecture of model based on Convolutional Neural Network (CNN) employed for the training phase. The CNN consists of multiple layers designed to capture intricate patterns within the dataset. The architecture includes:

1. **Convolutional Layers:**

- The convolutional layer has 32 filters with a kernel size of 3 and uses the ReLU activation function. This layer processes input images with a shape of [128, 128, 3].
- Max pooling layer with a pool size of 2 and strides of 2 follow each convolutional layer, helping reduce spatial dimensions and extract prominent features.
- There are 3 convolutional layers each followed by a max pooling layer.

2. **Flatten Layer:**

- After the convolutional layers, a flattening layer transforms the output into a one-dimensional array, preparing it for input into the dense layers.

3. **Dense Layers:**

- Two dense layers with 32 and 16 units, respectively, use the ReLU activation function. These layers facilitate the learning of complex hierarchical features.
- Dropout layers with a rate of 0.25 are inserted after each dense layer, aiding in regularization to prevent overfitting.

4. **Output Layer:**

- The output layer consists of five units, corresponding to the number of classes in our multi-class classification task. The SoftMax activation function is applied to generate probabilities for each class.

5. **Compilation:**

- The model is compiled using the 'adam' optimizer and 'binary_crossentropy' loss function. Binary Cross entropy is suitable for multi-class classification tasks.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 32)	0
conv2d_2 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 32)	200736
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dropout_1 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 5)	85

Total params: 220,741
Trainable params: 220,741
Non-trainable params: 0

Figure 4: CNN Architecture

3.8 Software Development Model

The Agile technique is a way of managing projects that emphasizes rapid product releases and modifying the project in accordance with requirements. Planning, requirement analysis, designing, development, and testing are all phases of the agile method cycle. The term "agile software development" refers to a collection of software development approaches built on an incremental and iterative strategy for evolution and adaptation that welcomes modifications at any stage of the SDLC process. Some of the development frameworks included in the Agile project management technique are Scrum, Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF). We will adopt the Scrum framework for this project since it promotes teamwork and high-quality software creation. Scrum projects are divided into brief and regular time-boxed SDLC iterations.

Figure 5: Agile Methodology, Scrum Model for Software Development

4. Epilogue:

4.1 Tasks Accomplished:

We have accomplished following tasks:

4.1.1 Data Collection:

We've compiled a dataset featuring monuments from Patan Durbar Square, totaling 2000 photos. This involved capturing 100 images for each side (East, West, North, and South) at five distinct monuments: "Chaar Narayan Mandir," "Chyasin Dega," "Krishna Mandir," "Shree Bhimsen Mandir," and "Vishwanath Mandir." This dataset forms the foundation of our project, serving as the primary material for training our model.

4.1.2 Data Augmentation:

We applied data augmentation techniques to enhance the diversity of our dataset, incorporating random rotations, flips, shear, and zoom. Each image underwent augmentation five times, resulting in a total of six images, including the original. Following augmentation, the dataset expanded to 12,000 images. Additionally, we resized all images to 128x128 pixels.

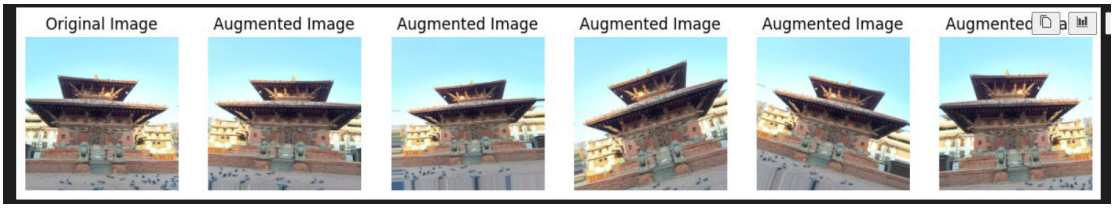


Figure 6: Sample of augmented images

4.1.3 Data Preprocessing:

To prepare the dataset for training we normalized the pixel values, ensuring they were scaled between 0 and 1. We also introduced shading effects to mimic various lighting conditions. Some images received a bit of controlled noise to make them resemble real-world photos. Additionally, we converted our images to black and white using grayscale conversion. All these actions combined contribute to the formation of a robust dataset. The dataset was split into training and test set in the ratio of 7:3.

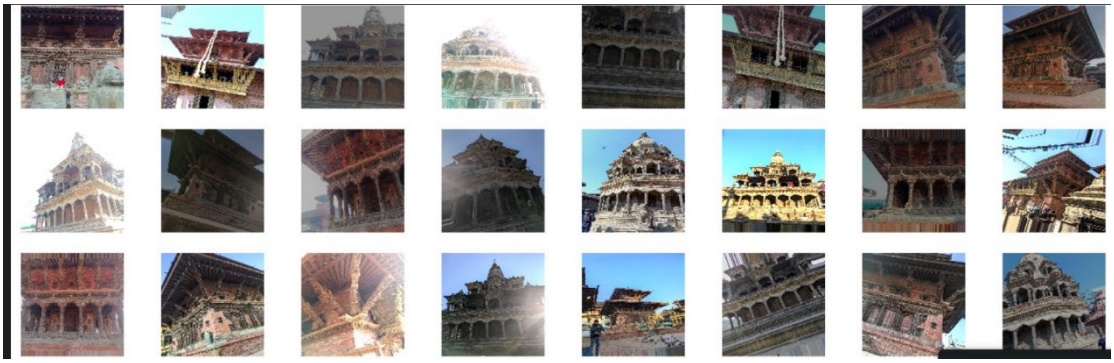


Figure 7: Sample images after shading

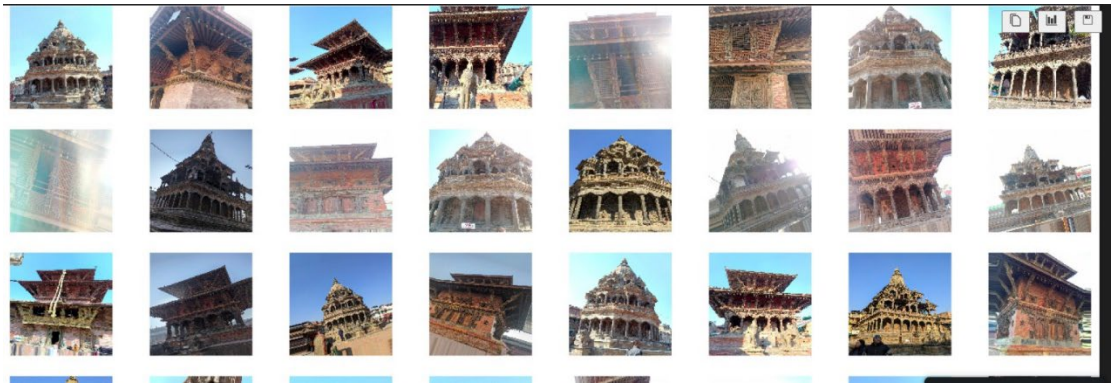


Figure 8: Sample images after adding noise

Figure 9: Sample images after grayscale conversion

4.1.4 Model Training:

In the model training phase, the prepared dataset takes center stage for training the model to comprehend and recognize patterns within the dataset. The images, normalized and augmented with shading effects and controlled noise, serve as inputs for the model, with grayscale conversion ensuring simplicity and focus on essential features. The CNN architecture was carefully chosen to facilitate effective learning. The hyperparameters, such as learning rate and batch size, were tuned to enhance the model's performance. Over multiple epochs, the model iteratively refines its understanding of the dataset, progressively improving its ability to recognize complex patterns. The validation set, initially split from the dataset, plays a critical role in

assessing the model's ability to generalize to unseen data. The model achieved training accuracy of 96.69% with a validation accuracy of 98.79%. In addition to accuracy metrics, the model's performance is further evaluated using the F1 score, which stands at an impressive 0.9875. This metric provides a measure of the model's precision and recall, reflecting its ability to correctly classify and capture relevant patterns within the dataset. The model consistently achieves high accuracy on both the training and validation sets, indicating its capability to generalize well to unseen data. To delve deeper into the model's performance, the confusion matrix offers a comprehensive view of its classification results. This matrix displays the number of true positive, true negative, false positive, and false negative predictions, providing insights into the model's strengths and areas for improvement.

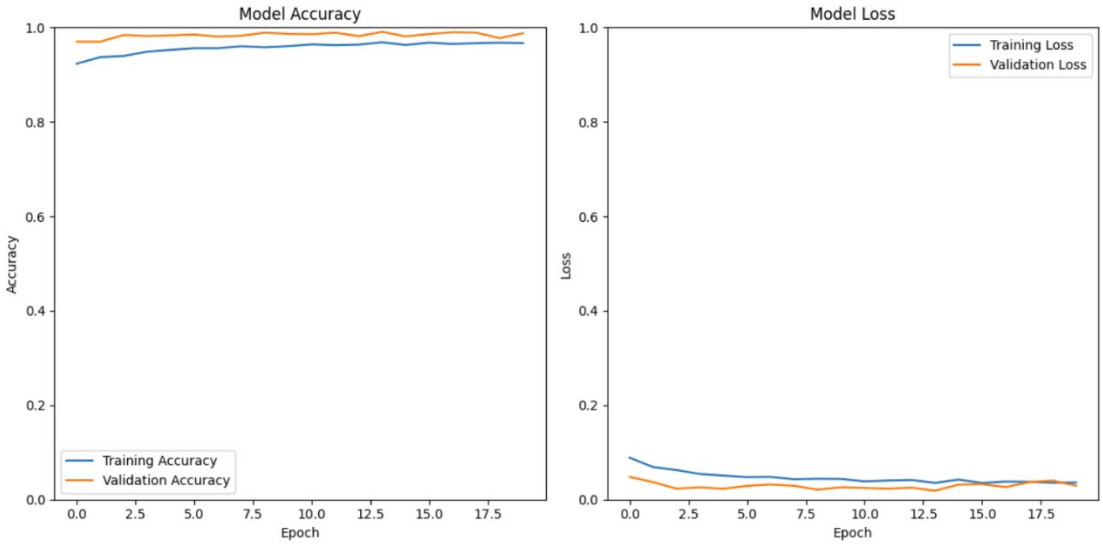


Figure 10: Loss and accuracy graph

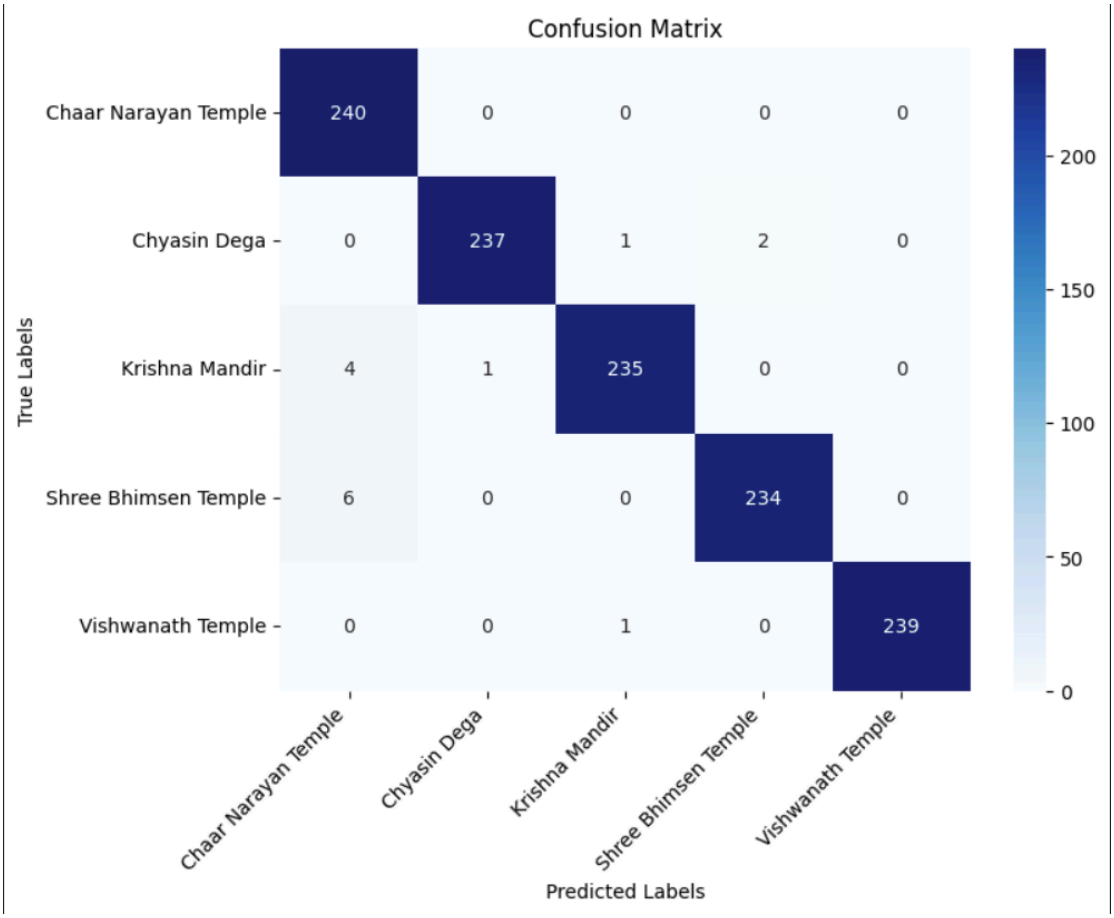


Figure 11: Confusion matrix

4.2 Tasks Remaining

4.2.2 Building Application

4.2.3 Model Integration

4.3. Gantt chart (Time Schedule)

Task	Mangsir				Poush				Magh				Falgun			
Feasibility Study																
Requirement Analysis																
Design																
Coding																
Review and Update																
Implementation																
Documentation																

Feasibility Study	
Requirement Analysis	
Design	
Coding	
Review and Update	
Implementation	
Documentation	

References

[1] Crudge, A., Thomas, W., & Zhu, K. (2014). Landmark recognition using machine learning. *CS229, Project*.

[2] Gada, S., Mehta, V., Kanchan, K., Jain, C., & Raut, P. (2017, December). Monument recognition using deep neural networks. In *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-6). IEEE.

[3] Etaati, M., Majidi, B., & Manzuri, M. T. (2019, March). Cross platform web-based smart tourism using deep monument mining. In *2019 4th International conference on pattern recognition and image analysis (IPRIA)* (pp. 190-194). IEEE.

[4] Palma, V. (2019). Towards deep learning for architecture: A monument recognition mobile app. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *42*, 551-556.

[5] Fatima, R., Zarrin, I., Qadeer, M. A., & Umar, M. S. (2016, July). Mobile travel guide using image recognition and GPS/Geo tagging: A smart way to travel. In *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)* (pp. 1-5). IEEE.

[6] Saini, A., Gupta, T., Kumar, R., Gupta, A. K., Panwar, M., & Mittal, A. (2017, December). Image based Indian monument recognition using convoluted neural networks. In *2017 International conference on big data, IoT and data science (BID)* (pp. 138-142). IEEE.

[7] Wikipedia contributors. (2023). Convolutional neural network. *Wikipedia*. https://en.wikipedia.org/wiki/Convolutional_neural_network

[8] *Python 3.11.4 Documentation*. (n.d.). https://docs.python.org/3/

[9] *Flutter documentation*. (n.d.). Flutter. https://docs.flutter.dev/

[10] TensorFlow. (n.d.). *TensorFlow*. https://www.tensorflow.org/learn

Appendix

Figure 12: Input Image (source: [Lalitpur Patan Durbar Square 2023 01 - Patan Durbar Square - Wikipedia])

```
Chaar Narayan Temple: 0.00%
Chyasin Dega: 0.00%
Krishna Mandir: 0.00%
Shree Bhimsen Temple: 100.00%
Vishwanath Temple: 0.00%
```

Figure 13: Predicted Probabilities

Figure 14: Input Image of Chyasin Dega (Chyasin Dega Krishna Temple at Patan Durbar Square in Lalitpur or Patan city near in Kathmandu in Nepal Stock Photo - Alamy)

```
Chaar Narayan Temple: 0.01%
Chyasin Dega: 88.49%
Krishna Mandir: 9.57%
Shree Bhimsen Temple: 1.82%
Vishwanath Temple: 0.11%
```

Figure 15: Predicted Probabilities

Figure 16: Input Image of Vishwanath Mandir (source: [Vishwanath Mandir Temple, Patan (Lalitpur) - TripAdvisor])

Figure 17: Predicted Probabilities

```
Chaar Narayan Temple: 31.12%
Chyasin Dega: 0.05%
Krishna Mandir: 0.11%
Shree Bhimsen Temple: 0.31%
Vishwanath Temple: 68.41%
```