HERITAGE IDENTIFICATION OF MONUMENTS USING DEEP LEARNING TECHNIQUES

Siva Srinivas M[1], Rohit Balaji R[2], Sangeethaa S N[3]

[1] Student, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India
[2]Student, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India
[3]Faculty, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India

ABSTRACT

A monument is a physical structure built with the purpose of commemorating a specific individual, event, or achievement. These structures hold significance within particular communities due to their aesthetic, historical, political, technological, or architectural importance, becoming integral parts of their culture and heritage. It is crucial to preserve and document these assets and monuments, as they are closely tied to the regions they belong to. To gather information about a specific subject, one common approach is to use the subject as a search query. The field of image recognition has seen significant advancements, thanks to developments in machine learning and deep learning, particularly in computer vision. With the increasing attention given to international landmarks and monuments, there is a growing need to establish a connection between the physical structures and their digital representations, making automated monument recognition essential. Recognizing monuments from images presents challenges because photographs can vary greatly based on the angle and viewpoint from which they were taken. The two-dimensional representation of monuments adds complexity to the task, as structural similarities among monuments can make identification more challenging. While various classifiers have been used for landmark recognition, the specific task of monument identification has received limited attention, leaving room for improvement. This study's primary goal is to classify images of monuments accurately, utilizing various deep learning architectures to achieve the highest possible accuracy. Several deep learning models, including Canny, Median Filter, and R-CNN, were found to be effective in recognizing monuments. Additionally, preprocessing plays a vital role in converting the dataset into a usable format and improving system performance. Monuments are three-dimensional structures, and differences in image perception can arise due to their varying perspectives. To address this, data augmentation techniques, such as random flip, random rotation, random translation, random zoom, random contrast, random hue, random brightness, and random saturation, were employed to reduce perspective variability. The study evaluated the performance of different design strategies to identify the approach that achieves the highest accuracy in monument recognition.

Keyword : - *Indian Monuments, Deep Learning, Monument recognition, Canny, Median Filter.*

1. INTRODUCTION

Computer vision is a multidisciplinary field that focuses on enabling computers to understand and interpret the visual world, much like how humans perceive and categorize objects from images or videos. It involves using mathematical methods to construct partial 3D models of objects, aiming to replicate our three-dimensional comprehension of the world in computers. Computer vision has found applications across various industries, including optical character recognition (OCR), quality control in manufacturing, retail, automotive safety, surveillance, 3D modeling, facial recognition, and more. Recent advancements in deep learning and machine learning have played a pivotal role in enabling powerful machine learning models for image and object detection, regardless of an object's characteristics.

At the core of these advancements lies the Convolutional Neural Network (CNN), a deep learning algorithm that excels at processing images. CNNs analyze images, assign significance (weights) to distinctive features within them, and distinguish one object from another. Given these capabilities, Convolutional Neural Networks hold promise for improving landmark and monument recognition.

Recognition as a landmark or monument carries significance, not only for tourists exploring different regions but also for locals interested in their cultural heritage. Imagine an individual using a smartphone app that employs this technology – they

can simply point their phone at a landmark or monument, open the Landmark/Monument identification app, and instantly access information about what they're looking at without any guidance. This interactive learning approach can make the educational experience engaging and enjoyable.

The rapid growth and revolutionary advancements in technology, particularly in mobile devices, have enabled on-device object and image recognition. With smartphones being an integral part of our lives and featuring increasingly powerful hardware, machine learning algorithms for image recognition and object detection have become accessible on these devices. However, it's important to optimize these technologies due to mobile devices' processing and hardware limitations.

Tensorflow, an open-source machine learning software package, offers a range of neural network models and includes Tensorflow Mobile, which helps bridge the gap between machine learning methods and mobile device constraints. Researchers and developers can choose a suitable model, deploy it even on lower-end devices, and further enhance application performance through optimization strategies.

This thesis aims to implement and deploy a Deep Convolutional model for real-time landmark and monument recognition. The goal is to create a deep learning application capable of recognizing landmarks and monuments from various angles, lighting conditions, and potentially partial obstructions. This is a challenging task, as these objects often share similar geometric shapes with other elements, and slight changes in angles can yield vastly different information.

Several key considerations emerge in this endeavor:

- Computational efficiency is crucial when dealing with image processing and recognition. Balancing the time required for model training and inference against the limitations of mobile devices is a critical task.
- Gathering and preparing training data is a time-consuming but essential step, as the quality of data and preprocessing techniques significantly impact the model's accuracy.
- Choosing the right model is another challenging yet pivotal decision. There is often a trade-off between model performance and speed, and certain models may excel in specific industries or applications.
- Model optimization is essential to ensure the program runs smoothly and efficiently, addressing potential bottlenecks and improving performance.

2. CODING PLATFORMS

Our research involves a thorough examination of the methodology and reasoning guiding the conducted tests, in conjunction with the selection of a specific procedure. Our aim is to implement and evaluate the suitability and effectiveness of CNN models in recognizing and identifying tourist spots and landmarks under various angles and lighting conditions. Ultimately, our goal is to provide highly precise predictions for the bounding boxes while maintaining computational efficiency.

2.1 PYTHON

Python is high level programming language. It is interpreted and object-oriented programming language. It have many libraries compared to other language. It is easy to learn. The syntax is readable so the anyone can understand. It is used in many project and mainly in Artificial intelligent or in Machin learning or in deep learning models. It can also to connect database and can access data from backend. It can also to create API application to interact with user. It have many built-in function so that user no need to write code for many basic functions. There are high level build-in function which are using in data structures with is combined with dynamic typing and binding. The source are available for all major platforms, which can be used without any change.

2.2 IDLE

An Integrated Development Environment (IDE) is a software tool designed to assist programmers in efficiently creating software code. It offers various advantages, such as facilitating software editing, building, testing, and packaging within a user-friendly interface. An IDE streamlines the development process, enabling developers to launch programs or applications more swiftly without the need for manual integration and configuration of multiple software components. By providing a unified interface for developer tools, IDEs enhance the effectiveness of software development. Examples of popular IDEs include NetBeans, Microsoft Visual Studio, Adobe Flex Builder, and Eclipse. IDEs typically support a wide range of programming languages, including but not limited to C, C++, Python, Perl, PHP, Java, Ruby, Tcl, JavaScript, and more. These versatile tools empower developers to work with various languages and scripting languages efficiently.

.

2.3 FEATURES OF IDLE

Let's explore some of the features of Python's Integrated Development Environment (IDE). Python's IDE primarily consists of Python code and a graphical user interface (GUI) toolkit. One of its key strengths lies in its robust debugging tools, which make it a valuable asset for developers. The IDE includes an easy-to-use editing window and an alternate text edit window.

- The Python IDE also provides a Python shell window, where input and output are displayed in color. This feature supports various content types, such as images, maps, plots, and error messages. Additionally, Python boasts a large standard library, offering a wide range of libraries catering to various professions, including artificial intelligence, web development, and scripting. Notable libraries for artificial intelligence include Tensorflow, Pandas, NumPy, Keras, and PyTorch. For web development, popular Python frameworks like Pyramid, Flask, and Django are available.
- Python is known for its versatility as a cross-platform language, functioning seamlessly on multiple operating systems like Windows, Linux, UNIX, Mac, and others. This portability allows software engineers to write a single program that can be deployed across different platforms.
- Furthermore, Python is free and open source, granting access to everyone, including developers. A vibrant global community actively contributes to creating new Python modules and enhancing its capabilities. This open nature of Python encourages collaboration and growth within the Python community.
- Python is also an object-oriented language, supporting concepts such as classes and objects. It embraces principles like polymorphism, encapsulation, and inheritance. This object-oriented approach enables developers to write more efficient code with less redundancy and promotes the creation of reusable code modules.

2.4 ADVANTAGES OF PYTHON IDLE:

- Python IDLE allows users to quickly test and run small chunks of Python code without having to develop an entire program.
- Python IDLE offers tools like syntax highlighting and code completion that make writing Python programs easier and quicker.
- A built-in debugger in Python IDLE makes it possible for programmers to walk through their code and identify errors and issues.
- Python IDLE's cross-platform nature allows it to be used on Linux, macOS, and Windows.
- Users don't need to install any additional programs in order to start writing Python code because Python IDLE is already included with the Python installation.
- Python IDLE is open-source, free software, thus users are free to use it for both commercial and non-commercial purposes without any restrictions.

2.5 JUPYTER NOTEBOOK

Jupyter Notebook is a notable application within the Project Jupyter framework, designed for authoring digital notebooks. Leveraging the capabilities of computational notebook formats, Jupyter Notebook introduces efficient and interactive approaches to code prototyping, code explanation, data exploration and visualization, as well as collaborative idea sharing.
These notebooks represent a significant departure from traditional console-based interactive computing. Instead, they provide a web-based platform that excels at capturing the entire computational journey. This encompasses tasks such as code creation, comprehensive documentation, code execution, and effective communication of results. In essence, Jupyter Notebook empowers users to document, execute, and share their computational work seamlessly.
Two elements are combined in the Jupyter notebook:

- A web application:An interactive authoring tool for computational notebooks that runs in a browser and offers a quick interactive environment for examining and explaining code, prototyping, and sharing ideas with others.
- Computational Notebook documents: A document that can be shared and includes computer code, data, descriptions in simple language, complex visuals like 3D models, charts, and graphs, and interactive controls.

2.6 FUNCTIONS IN JUPYTER NOTEBOOK

1. Varieties in Language Selection

You can choose the language you choose because the notebook supports more than 40 programming languages, including Python, R, Julia, and Scala.

2. Share Your Notebooks

This tool allows you to share your notebooks with others via email, Dropbox, GitHub, and the Jupyter Notebook Viewer.

3. Generating an Interactive Output
This feature enables your code to produce complex and interactive output, such as HTML, images, videos, LaTeX, and many MIME types. This includes figures created by the library that are suitable for publication and can be used inline. MathJax can be used to create mathematical notations, which are also easy to incorporate thanks to LateX.

4. Ease of Removing
Before publishing your book on the web, you may now remove any material you wish. In addition, you can remove simply the code so that the picture and other outputs continue to work.

5. Code Execution
Additionally, this has the capability of running browser-based scripts, with calculation results tied to the original code that created them.

6. Markdown
Rich text can be edited in-browser using the Markdown markup language, which is not just restricted to plain text but also gives commentary for the code. To give your postings cool effects, you may also embed photos, HTML, and other outputs inside them.

3. ALGORITHMNS AND METHODS

3.1 PREPARING THE DATASET

Undoubtedly, the parts of the experiment that took the longest to complete were data gathering and pre-processing. For each deep learning training model, the acquisition of the required dataset is an essential and unavoidable task.

3.1.1 Data Gathering

We gathered pictures of the 18 various UNESCO World Heritage Monuments in the city of Thessaloniki for the aim of this thesis. Since there was no existing dataset relevant to our objectives, the data collection was made up of manually acquired web images combined with owner-taken photos of the relevant monuments/landmarks. We make an effort to retain variation in our photographs because lighting and various monument angles are two aspects that directly affect a monument's coloration and possibly its design. With this approach, a higher likelihood of a correct and accurate prediction is ensured, regardless of the situation. To achieve our objective, we gathered a total of 4708 photos total, with 18 classes, are included. The sought-after classes are shown in Table 1 lists the quantity of photos for each monument or site.

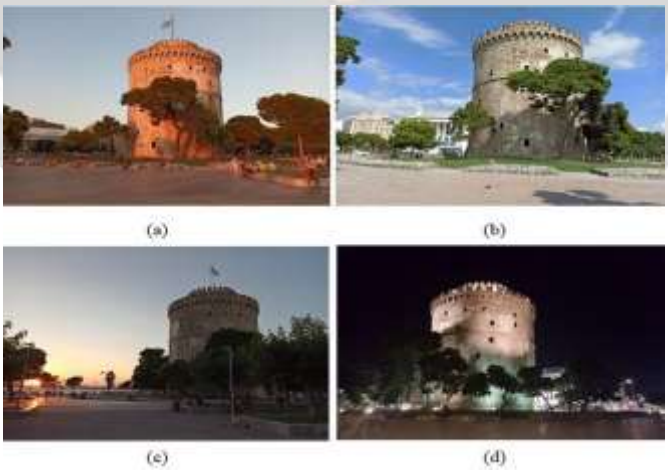

Figure 3.1: Examples of the White Tower in various lighting situations and perspectives.

3.1.2 Data Pre-Processing

Pre-processing is a crucial stage in machine learning. In this portion, we make sure that the data collection maintains a consistent quality while also taking the necessary steps to get the data ready for the training section.

Table 1: The number of images provided for each of the UNESCO Monuments of Thessaloniki.

UNESCO Monuments of Thessaloniki	Monument Image	Number of Images per Monument	UNESCO Monuments of Thessaloniki	Monument Image	Number of Images per Monument
Acheiropoietos		293	Saint Aikaterini		213
Aghios Demetrius		302	Saint Apostoles		249
Byzantine Baths		95	Saint Nikolaos Orphanos		222
Heptapyrgion		382	Saint Panteleimon		197
Metamorphosis Sotiros		163	Saint Sophia		329
Osios David		100	The Walls		292
Panayia Chalkeon		319	Trigonion Tower		274
Profitis Elias		285	Vlatadon Monastery		229
Rotunda		371	White Tower		393

3.1.3 Checking the Quality

First and foremost, it's crucial to keep only photographs that adhere to a particular resolution. Very low resolution images are useless for our purposes because they provide scant information and, as a result, degrade performance overall.

3.1.4 Renaming the Images

To prevent potential issues associated with lengthy image names containing punctuation marks and spaces, we opted to rename each image with more meaningful and manageable names. This decision has not only made it easier to identify and locate the desired images but has also streamlined the process of making necessary modifications when needed.

```
from PIL import Image
import os, sys

path = "C:/data/images/"
dirs = os.listdir(path)
final_size = 300;

def resize_an_image_keeping_aspect_ratio():

    for item in dirs:
        if item == '.DS_Store':
            continue
        if os.path.isfile(path+item):
            im = Image.open(path+item)
            f, e = os.path.splitext(path+item)
            size = im.size
            ratio = float(final_size) / max(size)
            new_image_size = tuple([int(x*ratio) for x in size])
            im = im.resize(new_image_size, Image.ANTIALIAS)
            new_im = Image.new("RGB", (final_size, final_size))
            new_im.paste(im, ((final_size-new_image_size[0])//2, (final_size-new_image_size[1])//2))
            new_im.save(f + '.jpg', 'JPEG', quality=90)
```

Figure 3.2: Example of the code utilized to resize the images on a specific path.

3.1.5 Annotating The Images

Image annotation represents a vital yet often time-intensive preparatory stage. To accomplish this task efficiently, we employ the LabelImg application, which is accessible on multiple platforms. This tool simplifies the process of creating essential bounding boxes and adding annotations to each of them, as demonstrated in (Figure 4.3). Furthermore, LabelImg offers the flexibility to save annotations in either YOLO or PascalVOC format, enhancing the versatility of the labeling process.

Figure 3.3: Example of the Labelimg environment. Creating a bounding box and provide an annotation

```
<annotation>
    <folder>Desktop</folder>
    <filename>Aghios_Demetrius24.jpg</filename>
    <path>C:/data/images/Aghios_Demetrius24.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>4032</width>
        <height>2268</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Aghios Demetrius</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>568</xmin>
            <ymin>247</ymin>
            <xmax>2741</xmax>
            <ymax>1916</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.4: The (.xml) file created after the annotation.

3.1.6 Splitting The Dataset Into Train And Test Set

Before proceeding to the next stage, we split our photos and their corresponding annotation files into two distinct sets: a training set and a testing set. This division is done using a 70:30 ratio to ensure a balanced representation. The separation is performed randomly, resulting in the creation of two separate folders—one for the training data and another for the testing data—aiming to maintain data homogeneity throughout the process.

4. RESULTS AND DISCUSSION:

When performing a work of categorising monuments, a collection of photographs including various forms of historic buildings was used. A deep learning model was used to train and accurately categorise monuments based on their respective types. The street-level monument categorization has an accuracy rate of 97%, while the spatial perspective of monuments achieves a 95%

accuracy rate.

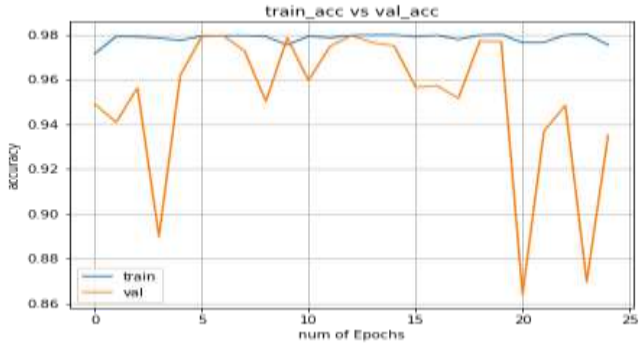

Figure 4.1: Visualization of metrics in tensor board

5. CONCLUSION:

We would be better able to identify the changes that will occur to monuments in the next days with the creation and deployment of a real-time inference framework and continuous monitoring.Additionally, it would aid in the monument's prevention, aiding in its identification in the future.Anomaly detection systems must be able to adapt to changing data patterns in order to continue to work effectively over time.

It is beyond a doubt that collecting more images in a variety of settings and from various angles, particularly for those with lower metrics, would enrich the work presented in this thesis and better the performance of the models. The frequency of inaccurate detections would be significantly decreased by combining this activity with the application's integration of the GPS feature, which would improve the end result. Giving users the option to use a built-in tool to find out crucial information about each monument is unquestionably a move that will be beneficial.

Although object identification models used to be either accuracy- or speed-oriented a few years ago, it is now possible to satisfy both requirements without losing any of these factors, and in fact without requiring much in the way of processing power. Researchers have given computer vision and, to a lesser extent, object identification a great deal of attention and research. Given the rate of technological advancement and the continual improvement of object identification models, object recognition has a bright future.

6. REFERENCES:

[1] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[2] Szeliski, R., 2010. Computer vision: algorithms and applications. Springer Science & Business Media.

[3] Umbaugh, S.E., 2010. Digital image processing and analysis: human and computer vision applications with CVIPtools. CRC press.

[4] Alsing, O., 2018. Mobile object detection using tensorflow lite and transfer learning.

[5] Gada, S., Mehta, V., Kanchan, K., Jain, C. and 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.

[6] Palma, Valerio. (2019). TOWARDS DEEP LEARNING FOR ARCHITECTURE: A MONUMENT RECOGNITION MOBILE APP. ISPRS

- International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-2/W9. 551-556. 10.5194/isprs-archives-XLII-2- W9-551-2019.

[7] Termritthikun, C., Kanprachar, S. and Muneesawang, P., 2018. NU-LiteNet: Mobile Landmark Recognition using Convolutional Neural Networks. arXiv preprint arXiv:1810.01074.

[8] Shi, H., Xu, M. and Li, R., 2017. Deep learning for household load forecasting—A novel pooling deep RNN. IEEE Transactions on Smart Grid, 9(5), pp.5271-5280.