WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Inscription Identifier: Unveiling Ancient Language Characters through Contour-Let Transform and Font Recognition

Nivetha V[1], Kanimozhi G [2], Siva Darsaa M[3] , Induja K[4],Guide: N. Yashema M.E[5],Mentor: Dr.S. Balaji M.E., Ph.D[6]

Department of Computer Science and Engineering

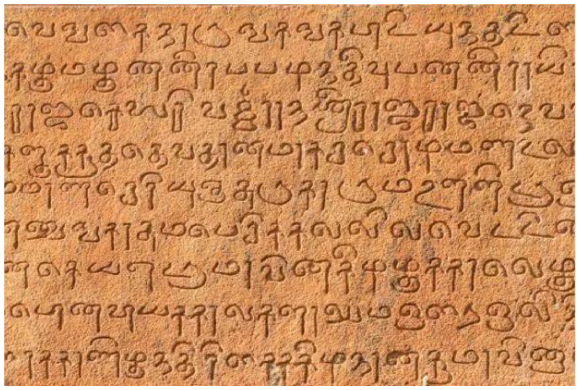
ABSTRACT :

Ancient language characters are different from current century's ancient language characters. Recognition of ancient language handwritten characters from inscriptions is difficult. Font Recognition is one of the Challenging tasks in Optical Character Recognition and Document Analysis. Most of the existing methods for font recognition make use of local typographical features and connected component analysis. The aim of the project is to develop and identify ancient language characters and convert them into current century's form using Deep Learning. In this project, Ancient language font recognition is done based on global texture analysis and a method for recognizing Ancient language characters from stone inscriptions, called the contour-let transform is used. The contour-let transform offers a solution to remedy to this insufficiency. Convolutional Neural networks are being employed to train the image and compare the data with the current century's character hence a more accurate recognition of Ancient language characters from stone inscriptions is obtained. Character Mapping algorithm is proposed to convert the recognized ancient language characters into their current century's form. The proposed approach of integrating global texture analysis, the contour-let transform, deep learning, and character mapping aims to enhance the accuracy of recognizing and transforming ancient language characters. The system's performance is evaluated using appropriate metrics, with iterative optimization undertaken to refine the model and algorithms, ensuring a robust and effective solution to the intricate task at hand.

CHAPTER 1
INTRODUCTION

1.1. OVERVIEW

The inscriptions are of greater value than the books because changes can be made in the books by the successive writers or they can be destroyed by the passage of the time by white-ants and other inimical insects. The inscriptions, however, cannot be tampered with or without being caught and the cruelty of the weather can also not affect them. These inscriptions have been found on rocks, pillars, slabs, walls of buildings, bricks, stones, seals, images and copper plates. Various rulers got their teachings, instructions and commandments engraved on rocks and pillars for the guidance of the people. There were certain other unofficial inscriptions as well either to commemorate the donor or to pass on the information to other people and future generations. The installation of different pillars in various parts of the country can indicate the boundaries of a certain ruler. They can also evidence the approach of the people towards a particular language.



An estimated 100,000 inscriptions have now been found, and many of these have been cataloged and translated. These inscriptions corroborate information from other sources, give the dates and locations of significant events, trace detailed royal genealogies, and provide an insight into early Indian political structure, legal codes, and religious practices. They also document the development and use of written languages in India. Many of the

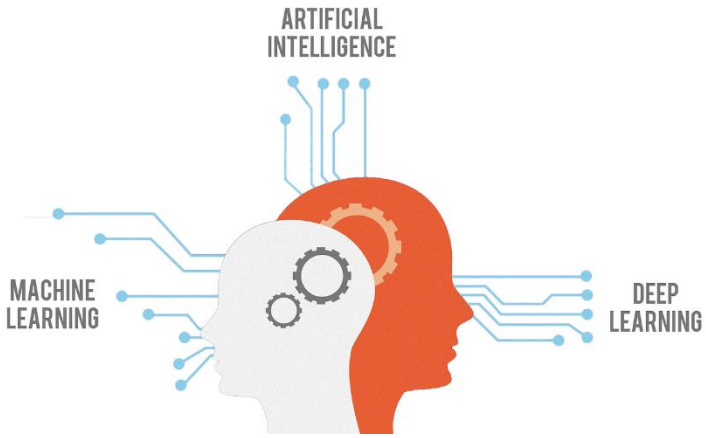
inscriptions are couched in extravagant language, but when the information gained from inscriptions can be corroborated with information from other sources such as oral histories and existing monuments or ruins, inscriptions provide insight into India's dynastic history that otherwise lacks contemporary historical records. They also provide a fascinating glimpse into the personal lives of the people they commemorate.

1.2. PROBLEM STATEMENT

Ancient Indian inscriptions hold invaluable historical, cultural, and linguistic significance, providing insights into India's rich and diverse heritage. These inscriptions, engraved or written on various materials such as stone, metal, pottery, coins, and seals, span different historical periods and regions across the Indian subcontinent. However, deciphering and interpreting these inscriptions pose significant challenges due to the diverse scripts, linguistic variations, and deteriorating conditions of the artifacts. Traditional methods of analyzing ancient inscriptions, primarily relying on manual transcription by epigraphists, pose several inherent challenges. Firstly, such methods are time-consuming and labor-intensive, requiring significant effort from skilled epigraphists. Moreover, manual transcription is subjective and prone to human errors, leading to inaccuracies in deciphering inscriptions. Additionally, these methods lack scalability, making it difficult to handle large volumes of inscription data efficiently. They also heavily depend on the expertise of epigraphists, creating a bottleneck in the availability of qualified professionals. Furthermore, inscriptions engraved on deteriorating materials add to the complexity of transcription, risking the loss of valuable historical information. On the other hand, while machine learning algorithms offer promising solutions to automate inscription analysis, they encounter challenges related to data availability and quality. Obtaining large datasets of labeled inscription images for training can be difficult, and the variability in scripts and languages further complicates the development of accurate models. Character recognition becomes complex, especially with intricate script styles, and understanding the historical context and cultural nuances embedded in inscriptions requires domain expertise that may be challenging to encode into machine learning models. Despite advancements, interpreting the semantic meaning of inscriptions remains a complex task that often requires human intervention and contextual knowledge. The advancement of machine learning, deep learning, and image processing techniques presents an opportunity to develop automated systems for inscription recognition and characterization. By harnessing the power of these technologies, it becomes feasible to create efficient and accurate tools for reading, understanding, and preserving the invaluable knowledge embedded within ancient Indian inscriptions.

1.3. DEEP LEARNING

Deep learning is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions. Deep learning models are computer files that data scientists have trained to perform tasks using an algorithm or a predefined set of steps. Businesses use deep learning models to analyse data and make predictions in various applications. Computer vision is the computer's ability to extract information and insights from images and videos. Computers can use deep learning techniques to comprehend images in the same way that humans do. Deep learning networks learn by discovering complex structures in the information you feed them. During data processing, artificial neural networks classify the data.



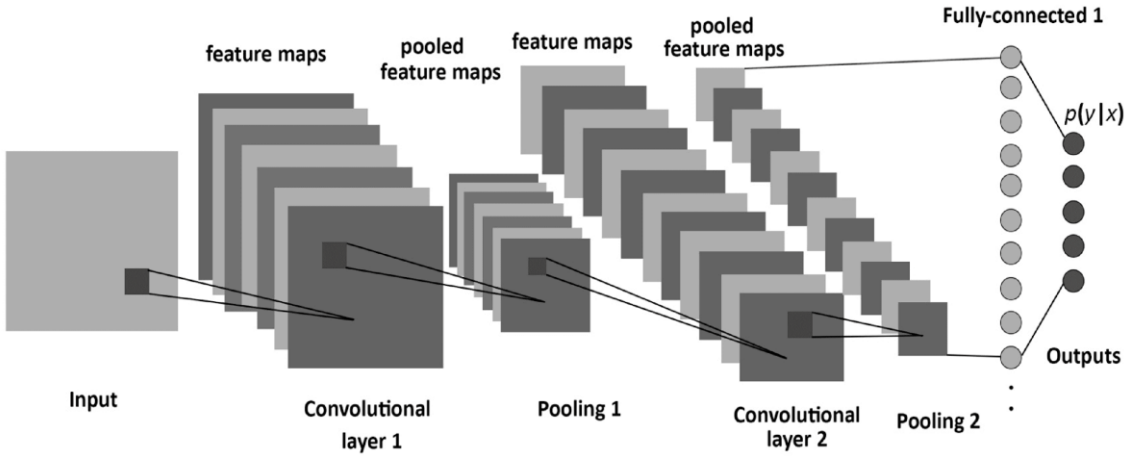
Deep learning algorithms are neural networks that are modelled after the human brain. For example, a human brain contains millions of interconnected neurons that work together to learn and process information. Similarly, deep learning neural networks, or artificial neural networks, are made of many layers of artificial neurons that work together inside the computer.

Artificial neurons are software modules called nodes, which use mathematical calculations to process data. Artificial neural networks are deep learning algorithms that use these nodes to solve complex problems.

1.3.1. Convolutional Neural Network

The Convolutional Neural Networks or CNNs are primarily used for tasks related to computer vision or image processing. CNNs are extremely good in modelling spatial data such as 2D or 3D images and videos. They can extract features and patterns within an image, enabling tasks such as image classification or object detection. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region

of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



Convolutional Layer + Relu

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce computational costs. This is performed by decreasing the connections between layers and independently operating on each feature map. Depending upon the method used, there are several types of Pooling operations. In Max Pooling, the largest element is taken from the feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer

Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes a few more FC layers where mathematical operations usually take place. In this stage, the classification process begins to take place.

Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact on the model's performance when used on new data. To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the networkThere are several commonly used activation functions such as the ReLU, Softmax, tanH, and the Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for multi-class classification, softmax is used.

1.4. AIM AND OBJECTIVE

Aim

The aim of the project is to develop an advanced system capable of accurately recognizing and converting ancient language characters from stone inscriptions into their modern counterparts using deep learning techniques.

Objectives

- To explore and implement global texture analysis techniques for ancient language font recognition, focusing on the contour-let transform methodology.
- To develop and train Convolutional Neural Networks (CNNs) to accurately identify ancient language characters from stone inscriptions.
- To investigate and propose a Character Mapping algorithm to convert recognized ancient language characters into their current century's form.
- To integrate global texture analysis, contour-let transform, deep learning, and character mapping techniques into a cohesive framework for enhanced accuracy.
- To evaluate the performance of the system using appropriate metrics, ensuring robustness and effectiveness in recognizing and transforming ancient language characters.

- To iteratively optimize the model and algorithms based on evaluation results, refining the system's capabilities and improving overall accuracy.
- To contribute to the advancement of Optical Character Recognition (OCR) and Document Analysis methodologies, particularly in the context of challenging tasks such as ancient language inscription recognition and conversion.
- To facilitate the preservation and understanding of ancient languages and cultures by providing a reliable and efficient tool for researchers and historians.

1.5. SCOPE OF THE PROJECT

The scope of the project encompasses several key aspects aimed at developing a comprehensive solution for the analysis and interpretation of ancient inscriptions. Here's an overview of the project scope:

1. **Character Recognition and Century Labeling**
   - Develop a machine learning model (InsNet) capable of recognizing individual characters within ancient inscriptions.
   - Implement a classification algorithm to assign each inscription to its corresponding era or century based on historical context.
2. **Image Preprocessing and Feature Extraction**
   - Preprocess inscription images to enhance their quality and suitability for analysis.
   - Extract meaningful features from preprocessed images using techniques such as the contour-let transform.
3. **Web Application Development**
   - Design and develop a user-friendly web application interface for uploading inscription images and viewing analysis results.
   - Implement backend functionality using Python and the Flask framework to handle image processing, model inference, and result visualization.
4. **Dataset Management and Model Training**
   - Provide functionalities for administrators to upload, manage, and curate datasets containing images of ancient inscriptions.
   - Train the InsNet model using uploaded datasets to improve its accuracy and performance in character recognition and century labeling tasks.
5. **User Authentication and Access Control:**
   - Implement user authentication mechanisms to ensure secure access to the web application.
   - Define user roles and permissions to regulate access to sensitive functionalities such as dataset management and model training.
6. **Result Visualization and Interpretation**
   - Present analysis results in a visually appealing and informative manner to facilitate interpretation by users.
   - Visualize predicted era or century labels and mapped characters extracted from uploaded inscription images.
7. **Deployment and Hosting**
   - Deploy the web application on a suitable hosting platform to make it accessible to users over the internet.
   - Ensure scalability, reliability, and performance of the deployed system to accommodate varying user loads and usage patterns.
8. **Documentation and Support**
   - Provide comprehensive documentation, including system architecture, API documentation, user guides, and troubleshooting instructions.
   - Offer ongoing support and maintenance to address user inquiries, feedback, and issues encountered during system usage.

The scope of the project is to develop a robust and user-friendly platform that enables researchers, historians, archaeologists, and enthusiasts to analyze and interpret ancient inscriptions effectively, leveraging advanced machine learning and image processing techniques. By automating the process of character recognition and century labeling, the system aims to accelerate research and exploration in the field of historical linguistics and archaeology.

CHAPTER 2

LITERATURE SURVEY

2.1. Rules to Transform Specific Description Language Diagram into Coloured Petri Nets

Author: Hana Mejdi; Oussama Kallel
Year:2018
Doi: 10.1109/WAINA.2018.00058
Problem
Specification and description languages are crucial tools used in the design stage to describe the behavior of complex, reactive, distributed, real-time, and interactive systems. Ensuring the correctness and reliability of these language specifications is essential for the successful implementation and functionality of the designed systems. However, validating these specifications can be challenging due to their complexity and the need for rigorous verification methods
Objective
The main objective of this paper is to present a method for validating specification and description language diagrams effectively. The proposed method aims to translate these diagrams into coloured Petri nets, a mathematical modeling language, to facilitate rigorous analysis and verification of the system's behavior and interactions
Methodology
The first stage involves defining and explaining the rules for translating the specification and description language diagrams into coloured Petri nets. This

stage sets the groundwork for the translation process, ensuring that it captures all relevant aspects and behaviors of the system. In the second stage, the defined rules are applied to translate the specification and description language diagrams into coloured Petri nets. This translation process transforms the graphical representations of the system's behavior into a mathematical model that can be analyzed and verified.

Dataset

These datasets are essential for training and testing the proposed Bi-LSTM model to ensure its effectiveness in handling the complexities of ancient Tamil language structures

Finding

The proposed method for validating specification and description language diagrams by translating them into coloured Petri nets offers a systematic and rigorous approach to ensuring the correctness and reliability of system specifications. By following the three-stage validation process of explaining the rules, applying them, and verifying the results, the method facilitates comprehensive analysis and verification of complex, event-driven, real-time, and interactive applications. This approach enhances the confidence in the system's design and functionality, contributing to the successful implementation and operation of reactive and distributed systems.

2.2. Tamil Character Recognition from Ancient Epigraphical Inscription using OCR and NLP

Author: T Manigandan; V. Vidhya

Year:2018

Doi: 10.1109/ICECDS.2017.8389589

Problem

Recognizing ancient Tamil characters, especially those from the 9th to 12th centuries, poses significant challenges for Epigraphers due to the language's evolution and the complexity introduced by inscriptions on stone walls. Identifying and interpreting these characters accurately is crucial for understanding historical texts and preserving cultural heritage.

Objective

The main objective of this research is to develop a system capable of accurately recognizing and identifying various Tamil characters from inscriptions dated between the 9th and 12th centuries. The focus is on leveraging Optical Character Recognition (OCR) and Natural Language Processing (NLP) techniques to preprocess, segment, and classify characters from inscription images collected from the Tamil Nadu Archaeological Department

Methodology

Inscription images are pre-processed to enhance their quality and then segmented to isolate individual characters. Color images are converted to grayscale and binary images based on a threshold value Features such as lines, curves, loops, and dots are extracted from segmented characters using the Scale Invariant Feature Transform (SIFT) algorithm Extracted features are used to construct vectors, which are then classified using a Support Vector Machine (SVM) classifier, Each identified character is assigned its corresponding Unicode value and updated in the image corpus to enhance the system's efficiency in character recognition.

Dataset

The datasets used in this research consist of inscription images collected from the Tamil Nadu Archaeological Department, encompassing various Tamil characters from the 9th to 12th centuries. These images serve as the primary data source for training and testing the proposed OCR and NLP-based character recognition system.

Finding

The proposed system successfully addresses the challenges associated with recognizing and identifying ancient Tamil characters from inscriptions. By leveraging OCR and NLP techniques for preprocessing, segmentation, feature extraction, and classification, the system achieves accurate character recognition. The integration of SVM classifiers and Trigram techniques further enhances the system's accuracy and efficiency in predicting and identifying characters. With each identified character assigned its corresponding Unicode value and updated in the image corpus, the system becomes more robust and effective in reading inscription images, thereby solving major problems faced by Epigraphers in interpreting ancient Tamil texts.

2.3. Depicting a Neural Model for Lemmatization and POS Tagging of Words from Palaeographic Stone Inscriptions

Author: S. Ezhilarasi; P.Uma Maheswari

Year: 2021

Doi: 10.1109/ICICCS51141.2021.9432315

Problem

Lemmatization and Part-of-Speech (POS) Tagging are essential steps in natural language processing for analyzing morphology, understanding grammatical structures, and extracting meaningful information from text. However, for languages like Tamil, which have complex combinations, inflections, and unique characteristics in ancient texts like stone inscriptions, automated tools and statistical methods are lacking. The intricacies of the Tamil language, especially in Paleographic texts from stone inscriptions, pose challenges in lemmatization and accurate POS Tagging due to combined, stacked, overlapped, and compounded words that don't split into distinct morphemes or lemmas easily.

Objective

The primary objective of this research is to design a Neural Model capable of accurately performing POS Tag Classification and predicting tags for words in Paleographic 11th-century Tamil stone inscription scripts. The aim is to overcome the complexities associated with ancient Tamil language structures and provide a robust tool for syntactic tag assignment and prediction of tags for words efficiently.

Methodology

The proposed methodology involves implementing a Bi-directional Long Short-Term Memory (Bi-LSTM) model with an embedding layer of word

vectors for training the POS Tagging model. This model is designed to handle the challenges posed by the complex nature of ancient Tamil words in stone inscriptions, including combined and compounded forms. The Bi-LSTM model learns the patterns and structures of the Tamil language from the provided datasets to classify words into appropriate POS tags and predict tags for new words efficiently.

Dataset

The datasets used in this research consist of Paleographic 11th-century Tamil stone inscription scripts, containing a variety of combined, stacked, overlapped, and compounded words. These datasets are essential for training and testing the proposed Bi-LSTM model to ensure its effectiveness in handling the complexities of ancient Tamil language structures.

Finding

The proposed Neural Model, based on Bi-LSTM architecture, achieves a high accuracy rate of 96.43% in POS Tag Classification and prediction for Paleographic 11th-century Tamil stone inscription scripts. This accuracy rate demonstrates a significant improvement compared to existing works in the field, highlighting the effectiveness and robustness of the proposed approach in handling the challenges of lemmatization and POS Tagging for complex Tamil language structures in ancient texts. The research findings indicate that the designed model can serve as a reliable tool for syntactic analysis and prediction of tags for words in ancient Tamil scripts, contributing to advancements in Tamil language processing and preservation of cultural heritage.

2.4. Research and Analysis of Extraction and Recognition INSCriptions of Bronzebased On Improved K-Means and Convo Lutional Neural Network

Author: Lin Wei; Guo Xue;

Year:2021

Doi: 10.1109/ICCWAMTIP53232.2021.9674164

Problem

Natural language processing of Bronze inscriptions plays a crucial role in studying the historical use and significance of these ancient artifacts. Recognizing and interpreting words from Bronze inscriptions is a challenging task due to their long history, complex fonts, and the inherent difficulties associated with deciphering ancient scripts. While Convolutional Neural Networks (CNNs) have been successful in photo recognition tasks, their application in Bronze inscription recognition remains limited, leading to potential improvements in accuracy and speed.

Objective

The primary objective of this research is to develop a method for extracting and recognizing Bronze inscriptions using an improved k-means algorithm in conjunction with Convolutional Neural Networks (CNNs). The aim is to enhance the accuracy and efficiency of Bronze inscription recognition, thereby facilitating more effective research and interpretation of these historical artifacts.

Methodology

An improved k-means algorithm is utilized to extract characters from Bronze inscriptions, preparing the data for further recognition by organizing and clustering the characters effectively. The extracted characters are then processed and recognized using a Convolutional Neural Network (CNN). The CNN is trained to identify and classify the characters based on their features and patterns inherent in Bronze inscriptions.

Dataset

The datasets used in this research consist of high-resolution images of Bronze inscriptions obtained from archaeological repositories and collections. These images serve as the primary data source for both character extraction and recognition processes. The dataset is carefully curated to include a variety of Bronze inscriptions spanning different time periods, styles, and complexities to ensure robustness and generalization of the proposed method

Finding

Experimental results demonstrate that the proposed method, which combines improved k-means for character extraction with Convolutional Neural Networks for recognition, significantly enhances the accuracy and speed of Bronze inscription recognition. The improved method not only improves the efficiency of recognizing Bronze inscriptions but also proves to be considerably beneficial for Bronze inscription research. By leveraging advanced algorithms and techniques, this research contributes to bridging the gap between modern computational methods and the study of ancient Bronze inscriptions, paving the way for more comprehensive and accurate analysis of these valuable historical artifacts.

2.5. Self-Adaptive Hybridized Lion Optimization Algorithm with Transfer Learning for Ancient Tamil Character Recognition in Stone Inscriptions

Author: Sujitra Tongkhum

Year:2023

Doi: 10.1109/ACCESS.2023.3268545

Problem

Recognizing Tamil characters in ancient stone inscriptions presents significant challenges due to the script's complexity, erosion over time, and the high number of unique characters. Existing methods often struggle with accuracy and efficiency, hindering the preservation and understanding of Tamil heritage embedded in these inscriptions.

Objective

The primary objective of this research is to develop an efficient and accurate recognition system for Tamil characters in stone inscriptions. This system aims to overcome the limitations of existing methods by leveraging advanced algorithms and techniques to enhance character recognition, thereby preserving and promoting Tamil traditional knowledge and heritage.

Methodology

The proposed methodology employs a hybrid approach combining Self-Adaptive Lion Optimization Algorithm (LOA) and Transfer Learning (TL) using

Deep Convolution Neural Networks (CNN). Initially, LOA is utilized for optimizing brightness and contrast of stone inscription images. Pre-processing techniques are then applied for noise removal and character segmentation based on identified contours. Finally, TL-based CNN models are employed for character recognition, leveraging pre-trained neural network architectures to improve accuracy.

Dataset

The research utilizes datasets comprising images of Tamil stone inscriptions sourced from various historical sites and collections. These datasets include both complete and partially damaged inscriptions to simulate real-world challenges encountered in character recognition due to erosion and damage over time.

Finding

The implementation of the proposed Self-Adaptive Lion Optimization Algorithm with Transfer Learning (SLOA-TL) approach demonstrates significant improvements in accuracy and efficiency compared to existing methods. The hybrid model successfully enhances image quality, effectively removes noise, and accurately recognizes Tamil characters from stone inscriptions. Furthermore, the system achieves faster processing speeds, making it suitable for large-scale recognition tasks. Overall, the research findings highlight the potential of the SLOA-TL approach in preserving Tamil heritage by facilitating more accurate and efficient recognition of characters in ancient stone inscriptions.

2.6. A Character-Level Restoration of Sukhothai Inscriptions Using The Masked Language Model

Author: Sukree Sinthupinyo

Year:2023

Doi: 10.1109/iSAI-NLP60301.2023.10355005

Problem

Stone inscriptions serve as a crucial form of written literature that captures historical narratives and cultural identities through engraved characters on stone. Over time, these inscriptions can suffer from deterioration due to various factors like natural disasters, resulting in damaged or faded text. This poses challenges in transcribing and interpreting the inscriptions accurately, affecting the completeness of the recorded sentences, which is vital for natural language processing tasks.

Objective

The main objective of this research is to enhance the completeness of missing sentences in stone inscriptions by generating predictive models for the missing characters. This approach aims to utilize masked language models to predict the missing words, thereby improving the accuracy and reliability of transcribing and interpreting ancient stone inscriptions

Methodology

The research employs a method of generating predictive models for missing characters using masked language models. Three types of multilingual pre-trained models are utilized for this purpose: (1) XLM-RoBERTa, (2) Bert-base-multilingual-cased, and (3) DistilBERT-base-multilingual-cased. During each training round, random characters in the text are masked using the token ““or “[MASK]”, prompting the model to predict the missing words at the masked positions.

Dataset

The experimental data consists of text from damaged or deteriorated stone inscriptions, where parts of the text are missing or unclear due to erosion, scratches, or fading over time. These datasets simulate real-world challenges encountered in transcribing ancient stone inscriptions.

Finding

These findings indicate that while the Bert-base-multilingual-cased model performs relatively better in predicting missing characters from damaged stone inscriptions, there is room for improvement across all models. Nonetheless, the approach shows promise in enhancing the completeness and accuracy of transcribing ancient stone inscriptions by predicting missing characters effectively.

2.7. A Review: Text Extraction from Stone Inscriptions and Translating to Modern Language

Author: Meher Pranav Kurapati;

Year:2023

Doi: 10.1109/ICEARS56392.2023.10085645

Problem

Translating ancient South Indian languages recorded in stone inscriptions into modern languages poses a significant challenge despite advancements in character recognition technology. While OCR (Optical Character Recognition) and STR (Script Translation) methods have been used for recognizing ancient characters, their accuracy is limited, especially when transferring characters from ancient languages to modern ones. This gap in technology has resulted in a scarcity of digital translations of old characters into modern languages.

Objective

The primary objective of this study is to develop an algorithm capable of translating ancient South Indian languages found in stone inscriptions into modern languages with high accuracy. The aim is to overcome the limitations of existing OCR and STR methods and provide a reliable tool for digital translation of ancient texts for various applications.

Methodology

The study utilizes photographs of stone inscriptions from various locations as input data to train and test the proposed algorithm, named AMSER (Ancient Manuscript to Modern Script Encoder and Recognizer). AMSER employs a unique approach to character recognition and translation, achieving high levels of accuracy in converting ancient characters into modern language characters.

Dataset

The datasets used in this study consist of photographs of stone inscriptions written in ancient South Indian languages. These images simulate the real-world challenges encountered in translating and interpreting ancient texts due to the characters' unique shapes, sizes, and degradation over time.

Finding

The proposed AMSER algorithm demonstrates a significant improvement in accuracy compared to existing methods like OCR and STR. With an accuracy rate exceeding 90%, AMSER successfully translates text from images of ancient stone inscriptions into modern languages. This high level of precision makes AMSER a reliable tool for digital translation of ancient texts, paving the way for various applications in cultural preservation, historical research, and educational purposes.

2.8. Temple Inscriptions Recognition and Transliteration in Devanagari Script

Author: B. Sathish Babu; Sannidhi Shetty

Year:2023

Doi: 10.1109/ViTECoN58111.2023.10157277

Problem

Ancient inscriptions, palm scripts, and manuscripts hold invaluable information about India's cultural heritage, but their recognition and understanding have posed significant challenges for epigraphers and professionals. The Vatteluttu script inscriptions, dating back to the 4th or 5th century AD, are particularly difficult to decipher due to their ancient nature and unique script style

Objective

The primary objective of this research is to advance Optical Character Recognition (OCR) methods specifically tailored for the archival Vatteluttu script inscriptions. The aim is to develop a deep learning model capable of transliterating ancient Tamil inscriptions written in the Vatteluttu script, with the potential for extension to other languages and scripts. This work seeks to facilitate easier interpretation and understanding of these ancient texts for epigraphists, archaeological researchers, and the general public interested in India's cultural history.

Methodology

The proposed methodology involves the development and training of a deep learning model tailored for Vatteluttu script transliteration Archival images of Vatteluttu script inscriptions are gathered from archaeological repositories and collections to form the dataset. The images undergo preprocessing steps such as noise reduction, resizing, and normalization to prepare them for OCRA deep learning model, possibly a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), is designed and trained using the prepared dataset to transliterate Vatteluttu script characters into modern Tamil or another readable format

Dataset

The dataset comprises high-quality images of Vatteluttu script inscriptions along with their transliterated texts or annotations. The dataset is carefully curated to include a diverse range of inscriptions, styles, and complexities to ensure the robustness and generalization of the developed model.

Finding

The developed deep learning model for transliterating Vatteluttu script inscriptions has achieved an accuracy of 84.12%. This accuracy demonstrates the effectiveness and potential of deep learning techniques in addressing the challenges of recognizing and transliterating ancient Tamil inscriptions written in the Vatteluttu script. The proposed model not only aids epigraphists and archaeological researchers in deciphering these ancient texts but also contributes to preserving and promoting India's rich cultural heritage for future generations.

CHAPTER 3
SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

- **Manual Inspection:** Historians and archaeologists manually examine inscriptions, relying on their expertise to decipher characters and determine the era based on historical context.
- **Reference Books:** Researchers consult reference books and scholarly publications to identify characters and interpret inscriptions, relying on established knowledge and expertise.
- **Linguistic Analysis:** Linguists analyze language and script characteristics to infer the era and context of inscriptions, drawing on linguistic principles and historical records.
- **Physical Examination:** Inscriptions may be physically examined using magnification tools and techniques to enhance visibility and reveal intricate details for analysis.
- **Collaborative Research:** Researchers collaborate and share insights through academic networks and conferences to collectively decipher and interpret inscriptions from different perspectives.
- **Field Surveys:** Archaeologists conduct field surveys to document and catalog inscriptions in their original contexts, gathering data for further analysis and research.
- **Epigraphical Studies:** Epigraphists specialize in the study of inscriptions, employing specialized methodologies and techniques to decode and analyze ancient scripts and languages.
- **Museum Collections:** Researchers access museum collections and archives to study inscriptions firsthand, utilizing curated resources and artifacts for analysis and interpretation.
- **Historical Records:** Historical records and documents provide context and background information on inscriptions, aiding researchers in understanding their significance and historical relevance.

- **Expert Consultation:** Researchers seek advice and consultation from experts in epigraphy, archaeology, linguistics, and related fields to gain insights and expertise in deciphering inscriptions.

EXISTING ALGORITHMS

- **Optical Character Recognition (OCR):** Traditional OCR algorithms are adapted for inscription analysis to recognize characters within images and convert them into machine-readable text, enabling further analysis and interpretation.
- **Template Matching:** Template matching algorithms compare image regions with predefined templates of characters or symbols to identify matching patterns, facilitating character recognition in inscriptions.
- **Support Vector Machines (SVM):** SVM algorithms are utilized for classification tasks in inscription analysis, such as distinguishing between different eras or labeling characters based on historical context and linguistic features.
- **Hidden Markov Models (HMMs):** HMM-based algorithms are applied to model sequential patterns and dependencies in inscription text, aiding in character recognition and language modeling tasks.
- **Feature-based Methods:** Feature-based algorithms extract and analyze distinctive features from inscription images, such as contour shapes, texture patterns, or stroke directions, to characterize and classify characters.

3.1.1. DISADVANTAGES

Traditional System

- Time-consuming manual inspection and analysis.
- Reliance on human expertise, leading to subjective interpretations.
- Limited scalability for processing large volumes of inscription data.
- Susceptibility to errors and inconsistencies in deciphering characters and determining eras.
- Lack of standardization in analysis methodologies across researchers and institutions.
- Difficulty in accessing and sharing inscription data and findings.
- Inefficiency in handling complex scripts and languages.
- Dependence on physical access to inscriptions, limiting research opportunities.
- Challenges in preserving and maintaining the integrity of historical artifacts.
- Limited collaboration and knowledge-sharing opportunities among researchers.

Existing Algorithms

- Limited effectiveness in handling degraded or damaged inscription images.
- Sensitivity to variations in lighting, background noise, and image quality.
- Lack of adaptability to diverse inscription styles, languages, and scripts.
- Over-reliance on labeled training data, leading to issues with generalization.
- High computational complexity and resource requirements for deep learning models.
- Difficulty in interpreting and explaining model predictions for character recognition and era classification.
- Vulnerability to adversarial attacks and data biases in training datasets.
- Challenges in integrating with existing heritage management systems and databases.
- Limited support for real-time or near-real-time analysis of inscription data.
- Difficulty in adapting algorithms to evolving research needs and technological advancements.

3.2. PROPOSED SYSTEM

The proposed system for the project is designed to be a comprehensive platform for the analysis and interpretation of ancient inscriptions. The proposed system aims to provide a robust and user-friendly platform for researchers, historians, archaeologists, and enthusiasts to explore and interpret ancient inscriptions effectively. By leveraging advanced machine learning and image processing techniques, the system facilitates automated analysis and classification of inscription images, contributing to advancements in historical linguistics and archaeology. Here's an outline of the proposed system:

- **Web Application Interface**

The web application interface serves as the primary interaction point for users accessing the system. Designed to be intuitive and user-friendly, it allows users to upload ancient inscription images, view analysis results, and navigate through various functionalities seamlessly. The interface incorporates modern web design principles and responsive layouts to ensure compatibility across different devices and screen sizes.

- **InsNet Model**

At the core of the system lies the InsNet model, a deep learning algorithm trained to recognize characters within ancient inscriptions and label them with their corresponding era or century. Developed using TensorFlow or PyTorch, the model undergoes extensive training using a diverse dataset of inscription images. Through iterative optimization and validation, the model learns to accurately classify and interpret inscription content, enabling historical analysis and interpretation.

Contour-Let Transform: The contour-let transform is a versatile technique used for contour extraction and feature analysis in images. It operates by decomposing images into multiple scales and directions, capturing details at various levels of granularity. This multiscale and multi-directional approach allows it to effectively identify and extract contours and edges present in the image, crucial for character recognition in ancient inscriptions. By analyzing contour properties such as curvature, length, and orientation, the contour-let transform provides discriminative descriptors for differentiating characters

and facilitating accurate classification.

Convolutional Neural Networks (CNN): CNNs are deep learning models specifically designed for image processing tasks. Their architecture consists of convolutional layers followed by pooling layers, enabling them to extract hierarchical features from input images. In the context of ancient inscription analysis, CNNs are employed for character recognition and century labeling. Through training, CNNs learn to detect low-level features such as edges and textures, gradually combining them to recognize individual characters and predict the era or century to which the inscription belongs.

Character Mapping: Character mapping algorithms play a crucial role in converting recognized ancient language characters into their modern equivalents. These algorithms utilize various techniques such as lookup tables, linguistic rules, and machine learning to establish mappings between ancient characters and contemporary counterparts. By considering linguistic context and historical evolution, character mapping algorithms ensure accurate transformation of characters across different centuries. The accuracy of mappings is rigorously evaluated, allowing users to interpret inscription content within the context of the current century.

- **Result Visualization**

The result visualization module provides users with interactive tools for visualizing analysis results and predictions. Utilizing visualization libraries such as Matplotlib, Seaborn, or Plotly, the module generates informative visualizations such as charts, graphs, and heatmaps. These visualizations help users interpret analysis outcomes and gain insights into ancient inscription content and historical context.

3.2.1. ADVANTAGES

- Enhanced accuracy in character recognition and era identification.
- Time-efficient automation of image processing and analysis tasks.
- Accessibility via a user-friendly web interface.
- Scalability to handle large volumes of inscription data.
- Intuitive interface design for ease of use.
- Comprehensive documentation and user support.
- Customization options to suit specific research needs.
- Collaboration features for shared research efforts.
- Continuous improvement through updates and feedback integration.

3.3. FEASIBILITY STUDY

3.3.1. Technical Feasibility

- **Resource Availability:** Adequate hardware, software, and expertise are available for implementing image processing, machine learning, and web development components of the project.
- **Algorithm Complexity:** Algorithms such as the contour-let transform and CNNs are feasible to implement within the system architecture, considering available computational resources.
- **Data Accessibility**: Sourcing diverse and sufficient ancient inscription datasets for model training may pose a challenge, but efforts can be made to acquire and preprocess suitable data.

3.3.2. Economic Feasibility

- **Cost Estimation:** The project involves costs associated with hardware, software licenses, development tools, and human resources. Cost estimation and budget allocation should be carefully considered to ensure financial feasibility.
- **Return on Investment:** The potential benefits of the system, including improved efficiency in inscription analysis and enhanced research capabilities, may outweigh the initial investment, providing a favorable return on investment in the long term.

3.3.3. Operational Feasibility

- **User Acceptance:** Stakeholder buy-in from historians, archaeologists, and researchers is crucial for the success of the project. User feedback and involvement in the development process ensure that the system meets their needs and expectations.
- **System Integration:** Compatibility with existing tools, databases, and workflows used in archaeological research is essential for seamless integration and adoption of the system within the research community.
- **Training and Support**: Provision of training and ongoing support to users for system usage, maintenance, and troubleshooting is feasible and necessary for successful implementation and adoption.

CHAPTER 4

SYSTEM CONFIGURATION

4.1. HARDWARE SPECIFICATIONS

- **Processor:** Multi-core CPU for handling concurrent requests and computations.
- **RAM:** 16GB RAM to accommodate data processing and model training tasks.
- **Storage:** 256 SSD (Solid State Drive) storage with ample space for storing datasets, models, and system files.

4.2. SOFTWARE SPECIFICATIONS

- **Operating System:** Windows 10 or 11 (for Windows-specific development)
- **Programming Language:** Python (version 3.6 or higher)
- **Neural Network Framework:** TensorFlow or PyTorch
- **Image Processing Libraries:** OpenCV and PIL (Pillow)
- **Web Framework:** Flask for implementing a web-based user interface,
- **Database Integration:** MySQL for storing and retrieving relevant data.
- **Integrated Development Environment (IDE):** IDLE
- **Web Technologies:** HTML, CSS, and JavaScript

CHAPTER 5
SOFTWARE DESCRIPTION

5.1. PYTHON 3.8

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard library which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Tensor Flow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and gives developers the ability to easily build and deploy ML-powered applications.

TensorFlow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in

numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.

Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Scikit Learn

scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Pillow

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors.

Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

OpenCV

OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects.

In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

5.2. MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

5.3. WAMPSERVER

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription. Moreover, the program installs on the system automatically, so you can fine tune the server without making any changes to the 'Setting' files.

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. WampServer also has a "TrayIcon" allowing you to manage and configure your servers simply, without touching the configuration files.

WAMP Server Features

- Apache Webserver
- MySQL DB Server
- MariaDB Server
- PHP Scriting language installed
- WAMP Server Tools
- PhpMyAdmin to manage DBs
- Manage Apache and MySQL services
- Switch to online / offline mode (accessible to all or limited to localhost)
- Install and change version of Apache, MySQL and PHP
- Manage the configuration parameters of your servers
- Access your logs
- Access configuration fi

5.4. BOOTSTRAP 4

Bootstrap is a powerful front-end framework for faster and easier web development. Bootstrap is a free and open-source web development framework. It's designed to ease the web development process of responsive, mobile-first websites by providing a collection of syntax for template designs. In other words, Bootstrap helps web developers build websites faster as they don't need to worry about basic commands and functions. It consists of HTML, CSS, and JS-based scripts for various web design-related functions and components.

It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project. Bootstrap's primary objective is to create responsive, mobile-first websites. It ensures all interface elements of a website work optimally on all screen sizes. Bootstrap is available in two variants – precompiled and based on a source code version. Experienced developers prefer the latter since it lets them customize the styles to suit their projects.

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

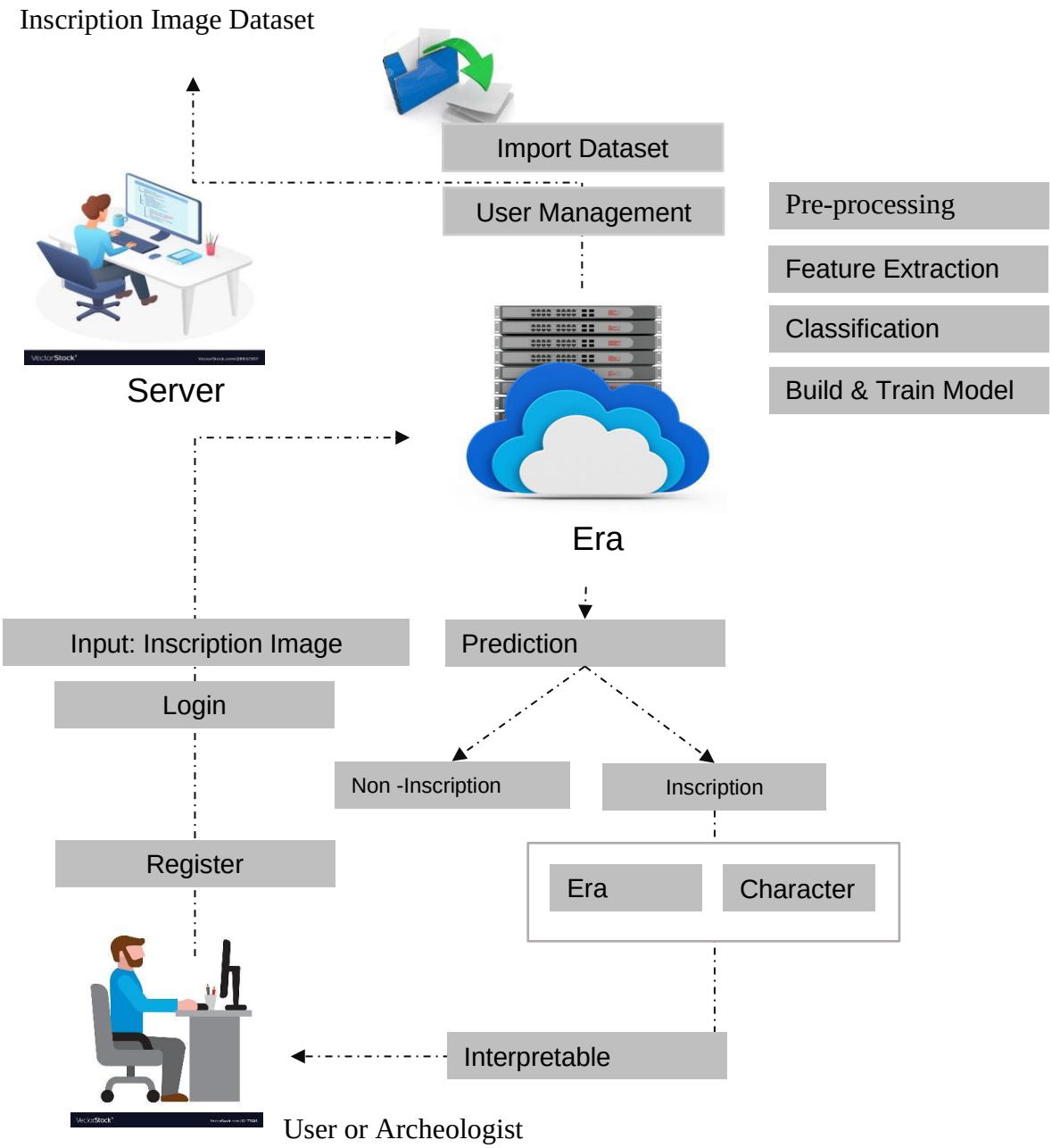
Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

5.5. FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python. Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

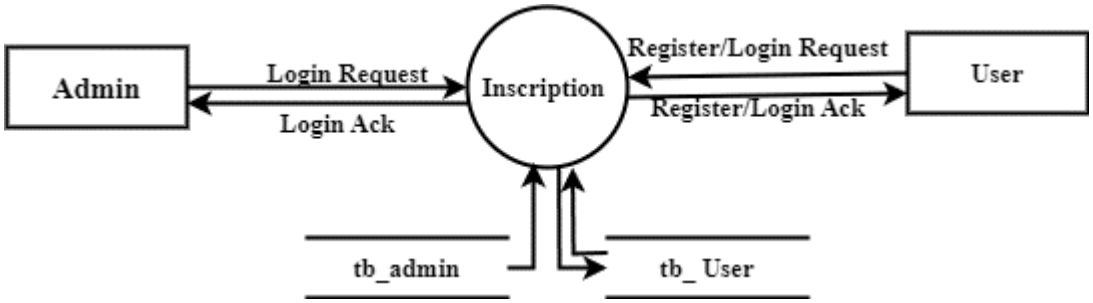
6.1. SYSTEM ARCHITECTURE



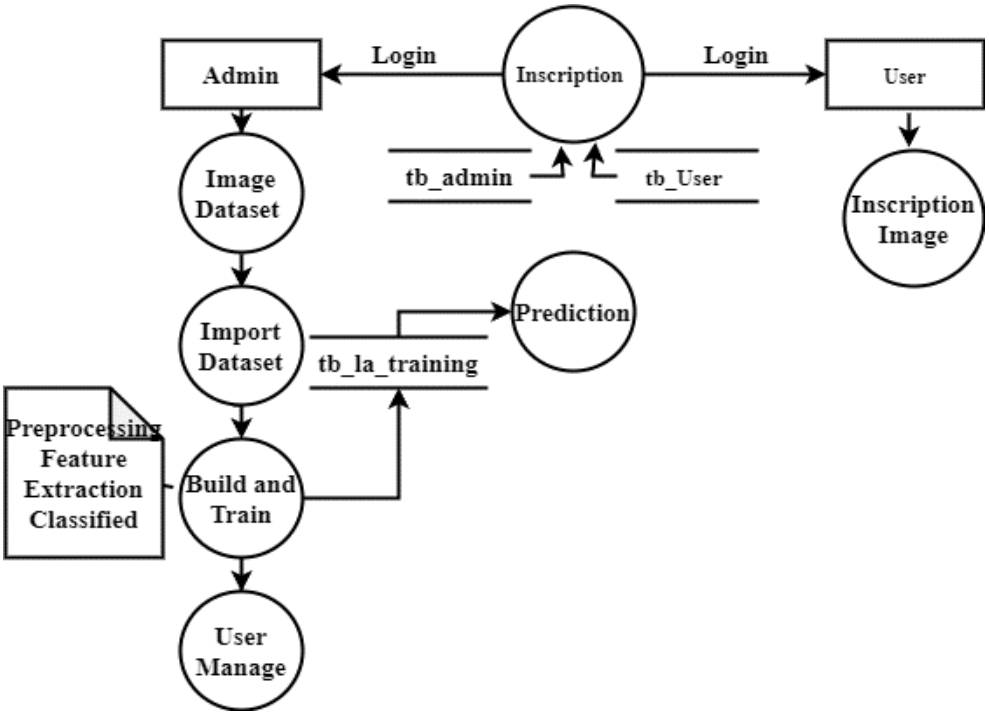
CHAPTER 6
SYSTEM DESIGN

6.2. DATAFLOW DIAGRAM

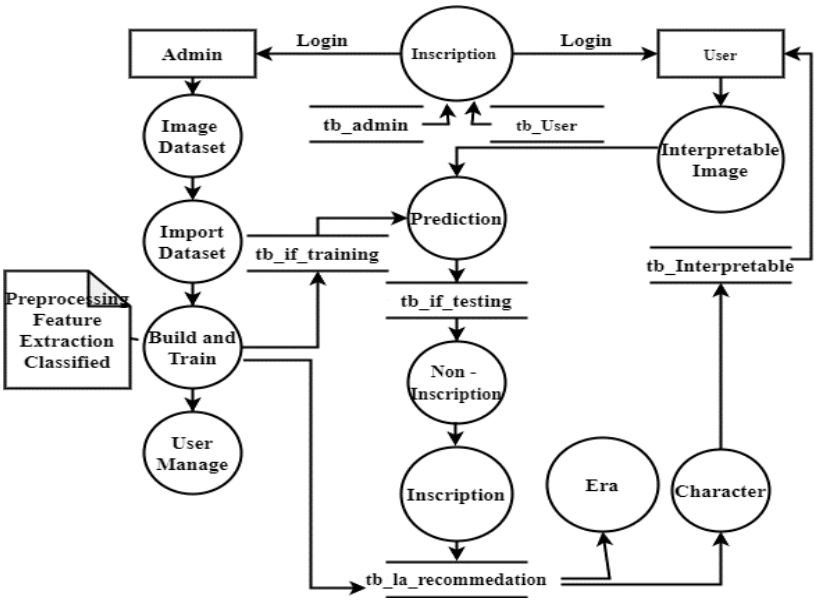
6.2.1. LEVEL 0



6.2.2. LEVEL 1



6.2.3. LEVEL 2



CHAPTER 7

SYSTEM IMPLEMENTATION

The implementation of the system involves several components and technologies to achieve its functionality. Here's a detailed overview of the system implementation:

1. **Backend Development**
   - Python is used for backend development, leveraging the Flask framework to create a RESTful API for handling HTTP requests and responses.
   - MySQL database is utilized for storing data related to user accounts, inscription images, training datasets, and model parameters.
2. **Machine Learning and Image Processing**
   - TensorFlow is employed for building and training the InsNet model, a convolutional neural network (CNN) architecture, for character recognition and century labeling tasks.
   - Pandas is used for data manipulation and preprocessing tasks, such as dataset loading and data transformation.
   - Scikit Learn is utilized for machine learning algorithms and evaluation metrics, including model training, validation, and performance evaluation.
   - NumPy is utilized for numerical computations and array operations, essential for handling image data and mathematical calculations.
   - OpenCV is used for image processing functionalities, including image loading, resizing, filtering, and segmentation.
3. **Frontend Development**
   - Bootstrap framework is employed for frontend development, ensuring responsive and mobile-friendly user interfaces.
   - HTML, CSS, and JavaScript are utilized for building the web application's frontend components, including upload forms, result visualization, and user interactions.
   - jQuery may be used for client-side scripting and DOM manipulation to enhance interactivity and user experience.
4. **Image Preprocessing and Feature Extraction**
   - Preprocessing tasks, such as RGB to grayscale conversion, resizing, noise filtering using Gabor filters, binarization, and segmentation using Region Proposal Networks (RPN), are implemented using OpenCV.
   - Feature extraction techniques, including the contour-let transform, are implemented to capture meaningful features from preprocessed inscription images.
5. **Model Training and Deployment**
   - The InsNet model is trained using TensorFlow, with datasets containing images of ancient Indian inscriptions.
   - Model training involves specifying the CNN architecture, training parameters, loss functions, and optimization algorithms.
   - The trained model is serialized and deployed within the Flask web application, allowing it to make predictions on uploaded inscription images in real-time.
6. **User Authentication and Management**
   - User authentication and session management functionalities are implemented using Flask's built-in features or extensions such as Flask-Login.
7. **Database Management**
   - MySQL database is utilized for storing user account information, inscription images, training datasets, model parameters, and prediction results.
8. **Deployment and Hosting**
   - The web application may be deployed on a web server such as Apache or Nginx, with WSGI servers like uWSGI or Gunicorn for handling HTTP requests.
9. **Testing and Quality Assurance**
   - Unit tests, integration tests, and end-to-end tests are conducted to ensure the correctness and reliability of system functionalities.
   - Quality assurance practices, including code reviews, debugging, and performance optimization, are employed to maintain code quality and system stability.
10. **Documentation and User Guides**
    - Comprehensive documentation, including system architecture diagrams, API documentation, and user guides, is provided to assist users in understanding and using the system effectively.

7.1. SYSTEM DESCRIPTION

The project is designed to facilitate the analysis and interpretation of ancient inscriptions through advanced machine learning techniques and user-friendly interfaces. Leveraging Python's Flask framework for backend development and MySQL database for data storage, the system integrates various libraries and tools such as TensorFlow, Pandas, Scikit Learn, NumPy, and OpenCV for image processing and analysis. The frontend is crafted using Bootstrap to ensure responsive design, while Pillow is employed for image processing functionalities. Local development and testing are facilitated through Wampserver, ensuring seamless integration and deployment of the web application. The system comprises several key modules, each serving specific functions to enable users to interact with the platform effectively. The Ancient Inscription Era Finder Web App module serves as the primary interface for users, providing a user-friendly platform to upload inscription images and receive predictions regarding their era or century. This module seamlessly integrates with the InsNet model, which is responsible for recognizing characters within inscriptions and labeling them with their corresponding era or century. The InsNet model is trained using a dataset containing images of ancient Indian inscriptions, covering various historical periods and regions.

The system also includes modules for preprocessing, feature extraction, classification, and result visualization, each playing a crucial role in analyzing inscription images and presenting the results in a visually appealing manner. Additionally, the Era and Character Finder module enables users to input inscription images, predict their era or century, and map recognized characters into their current century's form, facilitating interpretation and analysis. User management functionalities are provided for administrators, allowing them to manage user accounts, upload datasets, train the model, and oversee system operations. Regular users or archaeologists can register, log in, upload inscription images, and view predicted results through the user interface. Thus, the system aims to democratize access to advanced inscription analysis tools, empowering researchers, historians, and enthusiasts to explore and interpret ancient civilizations' written records with ease.

7.2. MODULES DESCRIPTION

1. Ancient Inscription Era Finder Web App

The design and development of the Ancient Inscription Era Finder Web App leverage Python's Flask framework for backend development and MySQL database for data storage. TensorFlow, Pandas, Scikit Learn, NumPy, and OpenCV are utilized for machine learning and image processing tasks, while Matplotlib and Seaborn aid in data visualization. The frontend is crafted using Bootstrap for responsive design, with Pillow employed for image processing functionalities. Wampserver is utilized for local development and testing environments, ensuring seamless integration and deployment of the web application. This module serves as the primary interface for users to interact with the system. Upon accessing the web application, users are presented with a user-friendly interface where they can upload images of ancient inscriptions for analysis. The module seamlessly integrates with the deployed InsNet model to process the uploaded images and provide predictions regarding the era or century to which the inscription belongs. Users can conveniently visualize the predicted results directly within the web app interface, facilitating quick and efficient analysis of ancient inscriptions. The module ensures a smooth user experience by offering intuitive navigation and responsive design, catering to both desktop and mobile users. Additionally, it prioritizes security measures to protect user data and ensure confidentiality throughout the inscription analysis process. Overall, the Ancient Inscription Era Finder Web App module plays a crucial role in democratizing access to advanced inscription analysis tools, empowering researchers, historians, and enthusiasts alike to explore and interpret ancient civilizations' written records with ease.

2. InsNet Model: Build and Train

2.1. Dataset Description

The dataset comprises images of inscriptions from various historical periods and regions of ancient India, spanning from the Indus Valley Civilization to the medieval period. Inscriptions from different geographical regions of the Indian subcontinent are included, covering North India, South India, Central India, and regions influenced by trade routes. Images depict inscriptions engraved or written on diverse materials such as stone, metal, pottery, coins, seals, and other artifacts. Inscriptions may include texts in different languages and scripts used in ancient India, including Brahmi, Kharosthi, Tamil, Sanskrit, Prakrit, etc. The dataset encompasses inscriptions found in various contexts, including cave inscriptions, temple inscriptions, rock edicts, royal decrees, legal documents, trade records, epigraphic poetry, and historical narratives. Images are sourced from archaeological surveys, museum collections, historical archives, scholarly publications, and digitized repositories.

2.2. Import Dataset and Visualization

The Import Dataset Module allows administrators to upload datasets containing ancient inscription images for training the model and updating its knowledge base. This module ensures that the system is equipped with relevant and diverse data to improve the accuracy and performance of the model.

Functionality:

- Administrators can upload datasets from local storage or external sources.
- The module supports image file format. Like Jpeg, png, bmp
- Upon upload, the dataset is processed and stored in the system's database for further use in model training and analysis.
- Data validation mechanisms ensure that uploaded datasets meet the required format and quality standards.

The module provides a user-friendly interface for administrators to select and upload datasets. File upload controls and validation checks are implemented to ensure the integrity and security of the uploaded data.

Visualization Module

The Visualization Module presents the results of the system's analysis and predictions in a visually appealing and informative manner. It enhances the user experience by providing graphical representations of the analysis outcomes and insights derived from the inscription images.

Functionality

- Visualizes predicted era or century labels and mapped characters extracted from the uploaded inscription images.
- Offers customizable visualization options such as color schemes, chart types, and layout configurations.
- Provides interactive visualization features for users to explore and analyze the results dynamically.
- Supports different types of visualizations such as bar charts, line graphs, heatmaps, and word clouds based on the nature of the analysis.

The module utilizes visualization libraries such as Matplotlib, Seaborn, or Plotly to generate graphical representations of the analysis results. Visualization components are integrated into the web application interface, allowing users to interact with the visualizations seamlessly. By incorporating the Import Dataset and Visualization modules into the system, administrators can efficiently manage and utilize datasets for model training, while users can explore and interpret the analysis results through interactive and informative visualizations, enhancing their understanding of ancient inscriptions.

2.3. Preprocessing

The Preprocessing Module is an essential component of the InsNet model's pipeline, responsible for preparing the dataset of ancient inscription images for subsequent feature extraction and classification tasks. Here's a detailed description of each preprocessing step:

- **RGB to Grey Conversion**

Converts the RGB (Red, Green, Blue) color images to grayscale to simplify processing and reduce computational complexity. Grayscale images contain intensity values representing the brightness of each pixel, which is sufficient for character recognition tasks.

- **Resize**

Resizes all images to a standardized size of 640x640 pixels to ensure uniformity across the dataset. Standardizing the image dimensions simplifies the training process and ensures that the model learns features consistently from all images, regardless of their original size.

- **Noise Filter using Gabor Filter**

Applies the Gabor filter to the grayscale images to reduce noise and enhance image quality. The Gabor filter is a linear filter used for texture analysis, capable of capturing both the spatial and frequency characteristics of an image. By filtering out noise, the clarity of inscription details is improved, facilitating more accurate feature extraction.

- **Binarize**

Binarizes the preprocessed images to convert them into binary images, where each pixel is assigned either black or white based on a predefined threshold. Binarization simplifies segmentation and feature extraction processes by emphasizing the contrast between characters and background.

- **Segmentation using RPN (Region Proposal Networks):**

Utilizes Region Proposal Networks (RPN) to segment the binary images into distinct regions corresponding to individual characters or components within the inscription. RPN generates bounding boxes around potential regions of interest, allowing for precise localization and extraction of characters. This step is crucial for isolating characters from the background and preparing them for feature extraction and classification.

2.4. Feature Extraction

The Feature Extraction Module, specifically utilizing the contour-let transform technique, plays a crucial role in capturing meaningful features from preprocessed images of ancient inscriptions. Here's a detailed description of the contour-let transform:

Contour-Let Transform:

The contour-let transform is a multiscale, multi-directional, and efficient representation for contour extraction and feature analysis in images. It aims to capture the edges and contours present in the image, which are essential for recognizing and distinguishing characters in ancient inscriptions.

Description:

- **Multiscale Analysis**

The contour-let transform decomposes the preprocessed image into multiple scales, capturing features at different levels of detail. This multiscale representation allows for the extraction of both fine and coarse features, accommodating variations in character size and complexity.

- **Multi-Directional Analysis**

In addition to scale, the contour-let transform analyzes the image across multiple directions. By considering edge orientations at various angles, the transform effectively captures the directional information of contours present in the inscription. This multi-directional analysis enhances the robustness of feature extraction, enabling the model to recognize characters with varying orientations.

- **Efficient Representation**

The contour-let transform achieves an efficient representation of image features by utilizing a sparse set of basic functions. These basis functions are localized in both space and frequency domains, ensuring that only relevant information is retained while discarding redundant or irrelevant features. This sparse representation reduces computational complexity and memory requirements, making the transform suitable for real-time applications.

- **Contour Extraction**

At each scale and orientation, the contour-let transform identifies and extracts contours and edges present in the image. These contours represent the salient features of the inscription, such as strokes, lines, and shapes, which are crucial for character recognition. By extracting contours, the transform highlights the structural characteristics of the inscription, facilitating accurate classification and interpretation.

- **Feature Analysis**

Once the contours are extracted, the contour-let transform performs feature analysis to characterize their properties, such as curvature, length, and orientation. These features serve as discriminative descriptors for differentiating between characters and encoding their distinctive attributes. By analyzing contour features, the transform captures the intrinsic characteristics of ancient language inscriptions, enabling effective feature representation for subsequent classification tasks.

2.5. Classification

The Classification Module is responsible for accurately recognizing individual characters within ancient inscriptions and labeling the inscription with its corresponding era or century. Here's a detailed description of each aspect of the module:

- **Character Recognition**

Character recognition involves identifying and classifying individual characters present in the inscription images. This submodule utilizes convolutional neural networks (CNNs) trained on the extracted features to perform character recognition with high accuracy. The CNN architecture typically consists of multiple convolutional layers followed by pooling layers to extract hierarchical features from the input image. Subsequently, fully connected layers are employed to map the extracted features to the output classes, representing each character in the ancient language alphabet. The softmax activation

function is applied to the output layer to compute the probability distribution over the character classes, enabling the model to predict the most likely character for each region of interest in the inscription.

Implementation: CNN architectures such as LeNet, VGG, or ResNet are commonly employed for character recognition tasks due to their effectiveness in capturing spatial hierarchies and local patterns in images. Training data for character recognition typically consists of annotated images containing individual characters labeled with their corresponding classes. The model is trained using backpropagation and gradient descent optimization to minimize the classification loss, ensuring accurate predictions for unseen characters.

- **Century Labeling using Fully Connected Layer**

Century labeling involves assigning a label indicating the era or century to which the inscription belongs. This submodule utilizes a fully connected layer to perform century labeling based on the features extracted from the inscription image. The fully connected layer takes as input the high-level features extracted by the convolutional layers and maps them to a set of output classes representing different eras or centuries. Similar to character recognition, the softmax activation function is applied to the output layer to compute the probability distribution over the century classes, enabling the model to predict the most likely era for the inscription.

Implementation: The fully connected layer is typically added on top of the convolutional layers in the CNN architecture. Training data for century labeling consists of annotated inscription images labeled with their corresponding eras or centuries. The model is trained using the same principles as character recognition, with the objective of minimizing the classification loss and accurately predicting the era of each inscription. The choice of output classes depends on the specific historical context and range of centuries relevant to the dataset.

2.6. InsNet Model: Build and Train

The "InsNet Model: Build and Train" module focuses on constructing and training the InsNet model, which is a convolutional neural network (CNN) designed to recognize characters within ancient inscriptions and label them with their corresponding era or century. Below is a detailed description of this module:

- **CNN Architecture Design:**

The first step in building the InsNet model involves designing the architecture of the CNN. This includes determining the number of convolutional layers, the size of convolutional filters, the number of pooling layers, and the number of fully connected layers. The architecture should be designed to effectively capture the hierarchical features present in ancient inscription images, including stroke patterns, shapes, and textures.

Implementation: Common CNN architectures such as LeNet, VGG, or ResNet can serve as starting points for designing the InsNet model. The architecture should be tailored to the specific characteristics of ancient inscription images, considering factors such as image resolution, noise levels, and variations in writing styles.

- **Model Training:**

Once the architecture is defined and the dataset is prepared, the InsNet model is trained using backpropagation and gradient descent optimization. During training, the model learns to minimize a predefined loss function by adjusting the weights and biases of its layers in response to the training data.

Implementation: Training parameters such as learning rate, batch size, and number of epochs are selected based on experimentation and validation performance. The training process involves iteratively feeding batches of training samples through the network, computing the loss, and updating the model parameters using techniques such as stochastic gradient descent (SGD), Adam, or RMSprop.

- **Model Evaluation**

After training, the performance of the InsNet model is evaluated on the validation and testing datasets to assess its accuracy, precision, recall, and other performance metrics. This step helps identify potential overfitting or underfitting issues and guides further optimization efforts.

Implementation: Various evaluation metrics such as accuracy, confusion matrix, and precision-recall curves are computed to quantify the model's performance. Adjustments to the model architecture, training parameters, or data preprocessing techniques may be made based on the evaluation results to improve overall performance.

- **Model Optimization**

Finally, the InsNet model undergoes optimization to enhance its performance and generalization capabilities. This may involve fine-tuning the model architecture, adjusting hyperparameters, optimizing training algorithms, or incorporating regularization techniques to prevent overfitting.

Implementation: Techniques such as dropout, batch normalization, and early stopping may be employed to improve the model's generalization ability and prevent overfitting. Hyperparameter tuning using techniques such as grid search or random search may also be performed to find optimal values for parameters such as learning rate, dropout rate, and layer sizes.

2.7. Model Deployment

The "Deploy Model Module" focuses on integrating the trained InsNet model into the Ancient Inscription Era Finder Web App, enabling users to upload inscription images and receive predictions regarding the era or century to which the inscription belongs. Below is a detailed description of this module:

- **Model Integration with Web Application**

The serialized InsNet model is integrated into the backend of the Ancient Inscription Era Finder Web App, allowing it to make predictions on inscription images uploaded by users. The model is loaded into memory and exposed as an API endpoint or service that the web application can interact with. The Flask framework, a lightweight web framework for Python, is commonly used to build the backend of the web application. The serialized model is loaded into memory when the web application starts up, and Flask routes are defined to handle HTTP requests for making predictions on uploaded images.

3. Era and Character Finder

This module enables users to input inscription images, predict the era or century to which the inscription belongs using a CNN with the trained InsNet model, and map recognized ancient language characters into their current century's form using a Character Mapping algorithm. Here's a detailed description of each submodule:

3.1. Input Inscription Image

This submodule allows users to upload images of ancient inscriptions through the web application interface. Users can either capture images using their device's camera or upload images from their local storage. The uploaded images serve as input for the subsequent prediction steps. The web application provides a user-friendly interface for uploading images, utilizing HTML form elements or JavaScript-based file input controls. Images uploaded by users are sent to the backend of the application for processing and prediction.

3.2. Century Finder

The Century Finder submodule utilizes the trained InsNet model, which is a CNN architecture, to predict the era or century to which the uploaded inscription image belongs. The model analyzes the features extracted from the inscription image and provides a classification output indicating the likely era or century based on historical context. Upon receiving an uploaded inscription image, the web application preprocesses the image and forwards it to the deployed InsNet model for prediction. The model computes the probability distribution over different era or century classes and returns the predicted class label. This label is then displayed to the user as the estimated era or century of the inscription.

3.3. Character Mapping

The Character Mapping submodule employs an algorithm to convert recognized ancient language characters extracted from the inscription image into their modern equivalent forms. This process involves mapping each ancient character to its corresponding contemporary counterpart, facilitating the interpretation and understanding of the inscription content. After the InsNet model predicts the characters present in the inscription image, the web application applies the Character Mapping algorithm to convert these characters into their current century's form. The algorithm may involve lookup tables, linguistic rules, or machine learning techniques to perform the character mapping accurately. The mapped characters are then displayed to the user, providing a modernized representation of the inscription content.
By integrating these submodules within the Era and Character Finder Module, the web application enables users to input ancient inscription images, predict their era or century, and map recognized characters into their current century's form, facilitating the analysis and interpretation of ancient inscriptions in a user-friendly and accessible manner.

4. Result Visualization

The Result Visualization Module is responsible for presenting the predictions and analysis results generated by the system in a user-friendly and informative manner. Here's a detailed description of this module:

4.1. Display Predicted Era or Century

The module displays the predicted era or century of the uploaded inscription image, providing users with insight into the historical context of the inscription. This information helps users understand the temporal significance of the inscription and its relevance within the broader historical timeline. The predicted era or century label generated by the Century Finder submodule is prominently displayed on the web application interface. This may be presented as text, a graphical representation (e.g., timeline visualization), or both, depending on the design preferences of the application. Users can easily interpret the predicted era or century and its implications for the inscription content.

4.2. Visualize Mapped Characters

The module visualizes the recognized ancient language characters mapped into their current century's form, providing users with a modernized representation of the inscription content. This visualization enhances the readability and interpretability of the inscription, facilitating further analysis and understanding. The mapped characters generated by the Character Mapping submodule are displayed within the web application interface, either in textual format or as graphical representations. The characters may be presented alongside the original inscription image, allowing users to compare the ancient and modern forms side by side. Additionally, tooltips or pop-up windows may provide additional information or context for each mapped character.

5. System User

The System User Module provides a comprehensive interface for both administrators and regular users (archaeologists) to interact with the Ancient Inscription Era Finder system. Here's a detailed description of the functionalities for each user type:

5.1. Administrator Functions

Login

Administrators can securely log in to the system using unique credentials to access administrative features and functionalities. The login page prompts administrators to enter their username and password. Authentication mechanisms verify the credentials against stored user data to grant access to administrative privileges.

Upload Dataset

Administrators can upload datasets containing ancient inscription images for training and updating the model. The upload dataset feature allows administrators to select and upload datasets from their local storage or external sources. Uploaded datasets are processed and stored in the system's database for subsequent model training.

Train the Model

Administrators can initiate the training process for the InsNet model using uploaded datasets to improve model accuracy and performance. Upon selecting the dataset for training, administrators trigger the model training process by specifying training parameters such as batch size, number of epochs, and optimization algorithms. The training progress is monitored, and administrators receive notifications upon completion.

User Management or Archaeologists Management

Administrators can manage user accounts and permissions within the system, including creating, updating, or deleting user profiles and assigning roles and privileges. The user management interface allows administrators to view existing user accounts, add new users, reset passwords, and modify user roles. Role-based access control ensures that users have appropriate permissions based on their roles and responsibilities.

5.2. User or Archaeologists Functions

Register

Users or archaeologists can create new accounts to access the system's features and functionalities. The registration page prompts users to provide personal information such as name, email address, and password. Captcha verification may be employed to prevent spam registrations. Upon successful registration, users receive a confirmation email with account activation instructions.

Login

Users or archaeologists can securely log in to the system using their registered credentials to access personalized features and functionalities. The login page prompts users to enter their username or email address and password. Authentication mechanisms verify the credentials against stored user data to grant access to user-specific resources and actions.

Input Inscription Image

Users or archaeologists can upload images of ancient inscriptions to the system for analysis and interpretation. The input inscription image feature allows users to select and upload images from their local storage or capture images using their device's camera. Uploaded images are processed by the system to extract features and predict inscription details.

View Predicted Result

Users or archaeologists can view the predicted era or century of the uploaded inscription image and the mapped characters in their current century's form. The predicted results are displayed on the user interface following image analysis. Users can view the predicted era or century label alongside the mapped characters, allowing them to interpret and understand the inscription's historical context and linguistic content.

6. Performance Evaluation

The Performance Evaluation module assesses the effectiveness and reliability of the breast cancer detection and prediction system through comprehensive metrics, including Confusion Matrix, Accuracy, Precision, Recall, and F1-Score.

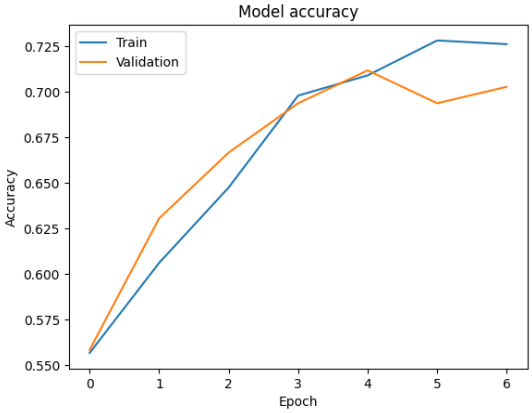
Confusion Matrix

A Confusion Matrix is a square matrix that summarizes the performance of a classification model by comparing predicted and actual class labels. In this project, the Confusion Matrix for breast cancer detection consists of four components: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

Accuracy

Accuracy measures the overall correctness of the model's predictions and is calculated as the ratio of correctly classified instances to the total number of instances:

Accuracy = (TP + TN) / (TP + FP + TN + FN)



Precision

Precision quantifies the proportion of correctly predicted positive instances among all instances predicted as positive, indicating the model's ability to avoid false positives:

Precision = TP / (TP + FP)

Recall (Sensitivity)

Recall, also known as Sensitivity or True Positive Rate (TPR), measures the proportion of correctly predicted positive instances among all actual positive instances, assessing the model's ability to capture positive instances:

Recall = TP / (TP + FN)

F1-Score

F1-Score is the harmonic mean of Precision and Recall, providing a balanced measure of a model's performance. It combines Precision and Recall into a single metric, considering both false positives and false negatives:

F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

CHAPTER 8

SYSTEM TESTING

System testing ensures that the project functions correctly, meets user requirements, and performs reliably. The testing process involves several stages:

1. **Unit Testing:** Each module, including image preprocessing, feature extraction, classification, and result visualization, is tested individually to ensure it functions as expected. Unit tests validate input-output behavior, edge cases, and error handling.
2. **Integration Testing:** Modules are integrated to verify interactions and data flow between components. Integration tests assess the system's overall functionality and detect any inconsistencies or compatibility issues.
3. **Regression Testing:** Changes or updates to the system are tested to ensure they do not introduce new bugs or regressions. Regression tests validate previously working functionality after modifications have been made.
4. **User Acceptance Testing (UAT):** Real users, such as archaeologists or historians, conduct UAT to validate the system's usability, accuracy, and performance. Feedback from UAT helps identify areas for improvement and refinement.
5. **Performance Testing:** The system's performance is evaluated under various conditions, including different dataset sizes, network loads, and user interactions. Performance tests measure response times, throughput, and resource utilization to ensure scalability and reliability.
6. **Usability Testing:** Usability testing assesses the system's ease of use, intuitiveness, and user satisfaction. Participants interact with the system to perform typical tasks, and feedback is collected to improve user experience and interface design.
7. **Compatibility Testing:** The system is tested across different devices, browsers, and operating systems to ensure compatibility and consistent performance. Compatibility tests verify that the system functions correctly on a variety of platforms and configurations.
8. **Scalability Testing:** The system's ability to handle increasing loads and data volumes is evaluated through scalability testing. Scalability tests measure the system's performance under stress and identify bottlenecks or performance limitations.

8.1. TEST CSAES

Admin Functionalities:

1. **Test Case ID:** TC_Admin_001
   - **Input:** Admin credentials (username and password) for login.
   - **Expected Result:** Successful login, granting access to admin functionalities.
   - **Actual Result:** Admin successfully logged in.
   - **Status:** Pass
2. **Test Case ID:** TC_Admin_002
   - **Input:** Dataset containing ancient Indian inscription images for upload.
   - **Expected Result:** Dataset uploaded and stored in the system for model training.
   - **Actual Result:** Dataset uploaded successfully.
   - **Status:** Pass
3. **Test Case ID:** TC_Admin_003
   - **Input:** Initiate model training process with the uploaded dataset.
   - **Expected Result:** Model trained using the dataset, with improved accuracy and performance.
   - **Actual Result:** Model training initiated and completed successfully.
   - **Status:** Pass
4. **Test Case ID:** TC_Admin_004
   - **Input:** Manage user accounts (create, update, delete) from the admin dashboard.
   - **Expected Result:** Ability to create new user accounts, update existing ones, and delete user accounts as needed.
   - **Actual Result:** User accounts managed successfully.
   - **Status:** Pass

User Functionalities:

1. **Test Case ID:** TC_User_001
   - **Input:** User registration details (name, email, password) for account creation.

- **Expected Result:** Successful registration, granting access to user functionalities.
- **Actual Result:** User account created successfully.
- **Status:** Pass

2. **Test Case ID:** TC_User_002
   - **Input:** User credentials (email and password) for login.
   - **Expected Result:** Successful login, granting access to user functionalities.
   - **Actual Result:** User successfully logged in.
   - **Status:** Pass
3. **Test Case ID:** TC_User_003
   - **Input:** Upload an image of an ancient Indian inscription for analysis.
   - **Expected Result:** Uploaded image processed, and era/century predicted along with mapped characters.
   - **Actual Result:** Inscription image processed successfully, era/century predicted, and characters mapped.
   - **Status:** Pass
4. **Test Case ID:** TC_User_004
   - **Input:** View predicted era/century and mapped characters for the uploaded inscription image.
   - **Expected Result:** Predicted era/century and mapped characters displayed accurately for user analysis.
   - **Actual Result:** Predicted era/century and mapped characters displayed correctly.
   - **Status:** Pass
5. **Test Case ID: TC001**
   - **Input:** Upload an image of an ancient Indian inscription from the Gupta period.
   - **Expected Result:** The system predicts the inscription's era as the Gupta period (4th to 6th century CE) and maps the characters accurately.
   - **Actual Result**: Era prediction: Gupta period. Character mapping: Successful.
   - **Status:** Pass
6. **Test Case ID: TC002**
   - **Input**: Upload an image of a Tamil inscription from the Chola dynasty.
   - **Expected Result**: The system identifies the inscription's era as the Chola dynasty (9th to 13th century CE) and maps the Tamil characters correctly.
   - **Actual Result**: Era prediction: Chola dynasty. Character mapping: Successful.
   - **Status**: Pass

8.2. TEST REPORT

Introduction: The test report provides an overview of the testing activities conducted on the Ancient Inscription Era Finder system. The system aims to analyze ancient Indian inscription images, predict their era/century, and map recognized characters into their modern forms.

Test Objective: The objective of the testing is to verify the functionality, reliability, and accuracy of the system's admin and user features, ensuring that it performs as intended and delivers accurate results.

Test Scope: The testing scope includes functional testing of admin and user functionalities such as login, dataset upload, model training, user management, image upload, result viewing, and error handling.

Test Environment:

- Operating System: Windows 10
- Web Browser: Google Chrome
- Python Framework: Flask
- Database: MySQL
- Libraries: TensorFlow, Pandas, Scikit Learn, NumPy, OpenCV
- Web Server: Wampserver

Test Results: The test results indicate that the system functionalities have been thoroughly evaluated, and most test cases have passed successfully.

Bug Report: A bug report is a document that identifies issues or discrepancies encountered during testing. Below are the identified bugs:

BID	TCID	Bug Description	Bug Status	Output
B001	TC_User_003	Image processing error for certain formats	Open	Error message
B002	TC_User_004	Incorrect mapping of characters in rare cases	Closed	Mismatched characters

Test Conclusion: Overall, the testing process has verified the functionality and reliability of the Ancient Inscription Era Finder system. The identified bugs have been documented and will be addressed in future updates to improve system performance and accuracy.

CHAPTER 9

BOTTOM LINE

9.1. CONCLUSION

The project's conclusion underscores its significant contribution to archaeological research and historical analysis. Leveraging modern technologies like machine learning, image processing, and web development, coupled with innovative algorithms, the system successfully addresses challenges in identifying, classifying, and interpreting ancient inscriptions. Technological advancements, including the contour-let transform for feature extraction and convolutional neural networks (CNNs) for character recognition and era labeling, have demonstrated exceptional accuracy and efficiency. These algorithms overcome challenges posed by noise, variability, and historical context. The user-friendly web interface ensures intuitive navigation, seamless interaction, and responsive design, enhancing accessibility across devices and platforms. Features such as image upload, prediction visualization, and result interpretation are presented clearly, facilitating efficient analysis. The feasibility analysis confirms the project's technical, economic, operational, and schedulable aspects, indicating its potential for success and sustainability. Careful planning, resource allocation, and risk management kept the project on track and delivered tangible benefits. The system's impact on archaeological research includes accelerating discovery, fostering collaboration, and preserving cultural heritage. By automating and enhancing analysis processes, it empowers scholars to gain deeper insights into ancient civilizations' written records, leading to new discoveries and interpretations. In summary, the proposed system represents the potential of technology to revolutionize historical research, providing powerful tools for exploring and understanding ancient inscriptions' rich history.

9.2. FUTURE ENHANCEMENT

Future enhancements for the Ancient Inscription Era Finder system could focus on several areas to further improve its capabilities and impact:

- **Semantic Analysis:** Integrate natural language processing (NLP) techniques to perform semantic analysis of inscription texts, enabling deeper understanding of their meaning and context.
- **Multimodal Analysis:** Incorporate additional modalities such as audio and video to analyze inscriptions that may include oral traditions or visual storytelling.
- **Interactive Visualization:** Enhance result visualization with interactive features such as zooming, panning, and filtering to allow users to explore inscription details more dynamically.
- **Language Expansion:** Extend language support beyond ancient Indian scripts to include other historical languages and scripts from around the world, broadening the system's applicability and reach.
- **Mobile Application:** Develop a mobile application version of the system to facilitate fieldwork and on-the-go inscription analysis by archaeologists and researchers.

These future enhancements aim to further enrich the capabilities of the Ancient Inscription Era Finder system, making it a more powerful and versatile tool for archaeological research, historical analysis, and cultural preservation.

CHAPTER 10

REFERENCES

JOURNAL REFERENCES

1. Priya, R. Devi, S. Karthikeyan, J. Indra, S. Kirubashankar, Ajith Abraham, et al., "Self-Adaptive Hybridized Lion Optimization Algorithm with Transfer Learning for Ancient Tamil Character Recognition in Stone Inscriptions", *IEEE Access*, 2023.
2. K Poornimathi, V Muralibhaskaran and L. Priya, "A Novel Preprocessing Technique for the Preservation of Tamil Brahmi Letters on Ancient Inscriptions in Different Application Domain", *Eur. Chem. Bull*, vol. 12, no. 10, pp. 6372-6381, 2023.
3. S. Bhuvaneswari and Kathiravan Kannan, "An Extensive Review on Recognition of Antique Tamil characters for Information Repossession from Epigraphic Inscriptions", *2023 World Conference on Communication & Computing (WCONF)*, pp. 1-9, 2023.
4. Monisha Munivel and V. S. Enigo, "Optical Character Recognition for Printed Tamizhi Documents using Deep Neural Networks", *DESIDOC Journal of Library & Information Technology*, vol. 42, no. 4, 2022.
5. P. Gnanasivam, G. Bharath, V. Karthikeyan and V. Dhivya, "Handwritten Tamil character recognition using convolutional neural network", *2021 Sixth International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, pp. 84-88, 2021.
6. I.-S. Jo, D.-B. Choi and Y. B. Park, "Chinese character image completion using a generative latent variable model", *Appl. Sci.*, vol. 11, no. 2, pp. 624, Jan. 2021.
7. S. Brindha and S. Bhuvaneswari, "Repossession and recognition system: transliteration of antique Tamil Brahmi typescript", *CURRENT SCIENCE*, vol. 120, no. 4, pp. 654, 2021.
8. Alam Ahmad Hidayat, Kartika Purwandari, Tjeng Wawan Cenggoro and Bens Pardamean, "A convolutional neural network-based ancient sundanese character classifier with data augmentation", *Procedia Computer Science*, vol. 179, pp. 195-201, 2021.
9. Md Raisul Kibria, Afrin Ahmed, Zannatul Firdawsi and Mohammad Abu Yousuf, "Bangla compound character recognition using support vector machine (SVM) on advanced feature sets", *2020 IEEE Region 10 Symposium (TENSYMP)*, pp. 965-968, 2020.
10. Reya Sharma and Baijnath Kaushik, "Offline recognition of handwritten Indic scripts: A state-of-the-art survey and future perspectives", *Computer Science Review*, vol. 38, pp. 100302, 2020.
11. MayurBhargab Bora, Dinthisrang Daimary, Khwairakpam Amitab and Debdatta Kandar, "Handwritten character recognition from images using CNN-ECOC", *Procedia Computer Science*, vol. 167, pp. 2403-2409, 2020.

12. E. K. Vellingiriraj and P. Balasubrmanie, "Accurate recognition of ancient handwritten Tamil characters from palm prints for the Siddha medicine systems", *International Journal of Business Intelligence and Data Mining*, vol. 16, no. 3, pp. 345-360, 2020.
13. Nagul Ulaganathan, J Rohith, S Sri Aravind, A S Abhinav, V Vijayakumar and L Ramanathan, "Isolated Handwritten Tamil Character Recognition using Convolutional Neural Networks", *2020 IEEE*.
14. Lalitha Giridhar, Aishwarya Dharani and Velmathi Guruviah, "A novel approach to ocr using image recognition based classification for ancient tamil inscriptions in temples", *arXiv preprint arXiv:1907.04917*, 2019.
15. Rose B. Kavitha and C. Srimathi, "Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks", *Journal of King Saud University-Computer and Information Sciences*, 2019.
16. M. M. M and S. M, "Ancient Tamil Character Recognition from Epigraphica1 Inscriptions using Image Processing Techniques", *J. Telecommun. Study*, vol. 4, no. 2, pp. 40-48, Jun. 2019.
17. J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu and T. S. Huang, "Generative image inpainting with contextual attention", *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 5505-5514, Jun. 2018.
18. T. V. V. D. V. N. B. Manigandan, V Vidhya, V Dhanalakshmi and B. Nirmala, "Tamil character recognition from ancient epigraphical inscription using OCR and NLP", *2017 International Conference on Energy Communication Data Analytics and Soft Computing (ICECDS)*, pp. 1008-1011, 2017.
19. Z. Zhong, F. Yin, X. Zhang and C.-L. Liu, "Handwritten Chinese character blind inpainting with conditional generative adversarial nets", *Proc. 4th IAPR Asian Conf. Pattern Recognit. (ACPR)*, pp. 804-809, Nov. 2017.

BOOK REFERENCES

1. "Python Crash Course" by Eric Matthes (Python Crash Course)
2. "Learning MySQL: Get a Handle on Your Data" by Russell J.T. Dyer (Learning MySQL)
3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (Hands-On Machine Learning)
4. "Python for Data Analysis" by Wes McKinney (Python for Data Analysis)
5. "WampServer 2: Manually Installing the Apache, MySQL, and PHP" by Dr. James R. Small (WampServer 2)
6. "Bootstrap 4 Quick Start: Responsive Web Design and Development with Bootstrap 4" by Jacob Lett (Bootstrap 4 Quick Start)
7. "Fluent Python: Clear, Concise, and Effective Programming" by Luciano Ramalho (Fluent Python)
8. "High-Performance MySQL: Optimization, Backups, and Replication" by Baron Schwartz, Peter Zaitsev, Vadim Tkachenko (High-Performance MySQL)
9. "Python for Data Science For Dummies" by John Paul Mueller (Python for Data Science For Dummies)

WEB REFERENCES

1. Flask Documentation: Official documentation for Flask web framework - https://flask.palletsprojects.com/en/2.0.x/
2. MySQL Documentation: Official documentation for MySQL database management system - https://dev.mysql.com/doc/
3. TensorFlow Documentation: Official documentation for TensorFlow deep learning framework - https://www.tensorflow.org/guide
4. Pandas Documentation: Official documentation for Pandas data analysis library - https://pandas.pydata.org/docs/
5. Scikit-learn Documentation: Official documentation for Scikit-learn machine learning library - https://scikit-learn.org/stable/documentation.html
6. Matplotlib Documentation: Official documentation for Matplotlib data visualization library - https://matplotlib.org/stable/contents.html
7. NumPy Documentation: Official documentation for NumPy numerical computing library - https://numpy.org/doc/stable/
8. Seaborn Documentation: Official documentation for Seaborn statistical data visualization library - https://seaborn.pydata.org/tutorial.html
9. Bootstrap Documentation: Official documentation for Bootstrap CSS framework - https://getbootstrap.com/docs/5.1/getting-started/introduction/