BRAIN STROKE IMAGE PROCESSING USING OPENCV

SUBMITTED BY

Group Number 1

Anupam Kumar Ram

Enroll: 2011200001011 Regn: 200010117494

Kuntal Mondal

Enroll: 2011200001029 Regn: 200010298425

Shivam Basak

Enroll: 2011200001054 Regn: 200010549823

Riddhith Banerjee

Enroll: 2011200001039 Regn: 200010398194

Under the Supervision of

Prof Pratik Bhattacharya

ASSISTANT PROFESSOR - COMPUTER SCIENCE & ENGINEERING

SISTER NIVEDITA UNIVERSITY

Content **Page No**

Introduction:

Brain stroke, a medical emergency that occurs when blood supply to the brain is disrupted, requires swift and accurate diagnosis for effective treatment. Recent advancements in computer vision and image processing technologies, particularly using OpenCV (Open Source Computer Vision Library), have opened up new avenues for improving the speed and accuracy of stroke detection through image recognition.

Motivation:

The motivation behind developing a brain stroke image recognition and processing system lies in the critical nature of timely stroke diagnosis. Prompt identification of stroke symptoms and immediate medical intervention significantly enhance the chances of successful treatment and recovery. By leveraging computer vision techniques, we aim to create a tool that assists healthcare professionals in rapidly and accurately diagnosing brain strokes based on medical imaging.

Problem Description:

The challenge in detecting brain strokes from medical images lies in the complexity of the human brain's anatomy and the subtle visual cues indicative of stroke damage. Manual analysis of medical images is time-consuming and prone to human error, especially in urgent situations. Therefore, the development of an automated system capable of recognizing and processing brain stroke images becomes imperative for expediting the diagnostic process.

Reason for Choosing this Topic:

The choice of developing a brain stroke image recognition system using OpenCV is rooted in the need for efficient and reliable tools in the healthcare sector. OpenCV, with its robust set of computer vision algorithms and image processing capabilities, provides a solid foundation for automating the analysis of medical images. The versatility and open-source nature of OpenCV make it an ideal choice for creating a solution that can be easily adopted and adapted in various healthcare settings.

Industrial Application:

The industrial application of brain stroke image recognition and processing extends to medical institutions, hospitals, and emergency care units. Implementing an automated system using OpenCV can significantly reduce the time required for diagnosis, enabling healthcare professionals to make faster and more informed decisions. This can lead to improved patient outcomes, reduced treatment costs, and enhanced overall efficiency in the healthcare sector. Additionally, the technology has the potential to be integrated into telemedicine platforms, enabling remote diagnosis and consultations, especially in regions where access to specialized medical expertise is limited.

The development of a brain stroke image recognition and processing system using OpenCV addresses a critical need in the healthcare sector by providing a tool that aids in the rapid and accurate diagnosis of strokes. The application of computer vision in this context holds the potential to revolutionize stroke care, offering benefits in terms of time, accuracy, and accessibility to medical expertise.

Methodology:

1. *Problem Specification:*
- Image Dataset Collection: Begin by acquiring a diverse and comprehensive dataset of brain stroke images. This dataset should include images from various modalities, such as CT scans and MRI, to capture the different aspects and stages of stroke.
- Data Preprocessing: Conduct preprocessing tasks on the acquired dataset, including image normalization, resizing, and noise reduction. Ensuring that the dataset is appropriately labeled, indicating the presence or absence of strokes, as well as the type and severity.
- Algorithm Selection: Choosing suitable computer vision and image processing algorithms for brain stroke detection. OpenCV provides a wide range of algorithms, such as edge detection, contour analysis, and feature extraction, which can be employed for this purpose.
- Model Training: Implementing a machine learning model using OpenCV's machine learning capabilities or integrate OpenCV with popular frameworks like TensorFlow or PyTorch. Training the model on the preprocessed dataset to recognize patterns and features indicative of brain strokes.

2. Why the Problem is Significant:
- Medical Emergency Response: Timely identification of brain strokes is crucial for initiating prompt medical intervention. Automating the detection process using computer vision can significantly reduce the time required for diagnosis, leading to faster medical responses and potentially improving patient outcomes.
- Accuracy Enhancement: Human analysis of medical images is prone to errors and can be time-consuming. An automated system can provide consistent and accurate results, reducing the likelihood of misdiagnosis and improving overall diagnostic reliability.
- Resource Optimization: The implementation of an image recognition system can optimize the utilization of healthcare resources by streamlining the diagnostic process. This efficiency is particularly important in emergency situations where time plays a critical role.

3. *Need for the Study :*
- Advancements in Medical Technology: With the increasing availability of medical imaging technologies, there is a growing need for advanced tools that can effectively analyze the vast amount of image data generated. OpenCV, as a powerful and versatile library, is well-suited for addressing this need.
- Integration of Computer Vision in Healthcare: The integration of computer vision techniques in healthcare has the potential to revolutionize the field, making diagnostics more efficient and accessible. Studying brain stroke image recognition using OpenCV contributes to this ongoing paradigm shift in medical technology.
- Public Health Impact: Brain strokes are a leading cause of disability and mortality worldwide. By developing an automated system for stroke detection, this study aims to make a significant contribution to public health by improving the speed and accuracy of stroke diagnoses, particularly in emergency scenarios.

The methodology outlined above encompasses data collection and preprocessing, algorithm selection, model training, and highlights the significance and need for studying brain stroke image recognition and processing using OpenCV. This approach aims to leverage the capabilities of computer vision to address the critical challenges in timely and accurate stroke diagnosis, ultimately impacting patient care and healthcare resource management.

Related Research:

1. Moeskops et al. (2016):
   - Study: Moeskops et al. conducted a study on brain tumor segmentation using deep learning techniques. Although not focused specifically on stroke, the study explores the application of advanced image processing methods for neurological conditions.
   - Relevance: While the primary focus is on tumors, the methodologies and techniques employed in this study could offer insights into image segmentation and feature extraction relevant to brain stroke recognition.

2. Lopes et al. (2017):
   - Study: Lopes et al. explored the application of machine learning algorithms for brain tumor detection and segmentation in magnetic resonance images (MRI). The study emphasizes the importance of accurate segmentation for effective diagnosis.
   - Relevance: While the emphasis is on tumors, the study's findings on segmentation methodologies can be relevant to brain stroke image processing, where precise delineation of affected areas is crucial.

3. Sarikaya et al. (2019):
   - Study: Sarikaya et al. investigated the use of convolutional neural networks (CNNs) for the automated detection of acute ischemic stroke in non-contrast CT images. The study focuses on the application of deep learning in stroke detection.
   - Relevance: This study is particularly relevant as it directly addresses the automated detection of stroke using advanced neural network architectures, providing insights into the potential application of similar methods in the context of OpenCV.

4. Bhanu Prakash et al. (2020):
   - Study: Bhanu Prakash et al. proposed a stroke detection system using deep learning algorithms applied to MRI images. The study aimed to enhance the accuracy and efficiency of stroke diagnosis through automated methods.
   - Relevance: This study contributes to the exploration of deep learning techniques for stroke detection and may offer insights into methodologies applicable to OpenCV-based solutions.

5. Bhagyashree Rajendra Gaidhani,Dr. R.Rajamenakshi,Samadhan Sonavane(2019):
- Study: They used two types of cnn - LeNet and SegNet.The experimental result show that the classification model achieved accuracy between 96- 97% and the segmentation model achieved accuracy between 85- 87%. Through experimental results, we found that deep learning models not only used in non-medical images but also give accurate results on medical image diagnosis, especially in brain stroke detection.
- Relevance: The two models of classification using LeNet and segmentation using SegNet can be used in our project as they gave such good accuracy results.

For the latest studies and advancements beyond January 2022, I recommend checking academic databases such as PubMed, IEEE Xplore, or other relevant sources to find the most recent research on brain stroke image recognition and processing using OpenCV or similar technologies.

6. In 2021, Abdullah Alamoudi and Yousif Abdallah published a chapter titled "Characterization of Brain Stroke Using Image and Signal Processing Techniques" in the book "Biomedical Signal and Image Processing" 1. The chapter presents the application of both image and signal processing techniques in the characterization of Brain Stroke field. It also summarizes how to characterize the brain stroke using different image processing algorithms such as ROI based segmentation and watershed methods

7. In 2022, a study was conducted by researchers from the University of Tabuk, Saudi Arabia, and published in Springer's "Advances in Intelligent Systems and Computing" 2. The study aimed to monitor the changes in the brain stroke images using image processing techniques. The authors used OpenCV to segment the brain stroke images and extract the features.

8. In 2022, a study was conducted by researchers from the University of Technology Sydney, Australia, and published in the journal "Diagnostics" 3. The study applied machine learning techniques for the characterization of ischemic stroke using CT images. The authors used OpenCV to preprocess the images and extract the features.

9. In 2020, Manoj Kumar Sharma,Vijaypal Singh Dhaka published a study named 'Innovations in Computational Intelligence and Computer Vision' where three pre trained CNN model learning technique was applied to classify brain tumor MRI images into benign and malignant.By using the transfer learning and data augmentation technique, it was possible to achieve such high accuracy even on such small datasets.

Detailed Methodology with a block diagram, flowchart, and algorithm:

Methodology:

1.Data Acquisition:
- Collecting a diverse dataset of brain images including CT scans and MRI slices.
- Ensuring the dataset contains labeled images indicating the presence or absence of strokes, along with relevant information on stroke type and severity.

2.Data Preprocessing:
- Image Normalization:
  - Normalizing pixel values to a common scale for consistency.
- Resize Images:
  - Resizing images to a standard resolution to ensure uniformity.
- Noise Reduction:
  - Applying filters to reduce noise and enhance image clarity.

3.Feature Extraction:
- Region of Interest (ROI) Selection:

- Identifing and extract the region of interest in the brain images, focusing on areas relevant to stroke detection.
- Edge Detection:
    - Applying edge detection algorithms (e.g., Canny edge detector) to highlight boundaries in the images.
- Texture Analysis:
    - Using texture analysis methods (e.g., GLCM - Gray-Level Co-occurrence Matrix) to capture textural features indicative of strokes.

4. Algorithm Selection:
    - OpenCV Integration:
        - Leverage OpenCV's image processing functions for tasks like contour analysis, morphology, and image filtering.
    - Machine Learning Model:
        - Implementing a machine learning model using OpenCV's machine learning module or integrate with external frameworks like TensorFlow or PyTorch.

5. Model Training:
    - Train-Test Split:
        - Spliting the dataset into training and testing sets for model evaluation.
    - Feature Vector Creation:
        - Converting the extracted features into a feature vector for each image.
    - Model Training:
        - Training the machine learning model on the training dataset using a suitable algorithm (e.g., Support Vector Machine, Random Forest) with cross-validation.

6. Model Evaluation:
    - Testing Set Evaluation:
        - Evaluating the trained model on the testing set to assess its performance.
    - Performance Metrics:
        - Using metrics such as accuracy, precision, recall, and F1-score for quantitative evaluation.

7. Integration and Deployment:
    - OpenCV Implementation:
        - Integrating the trained model with OpenCV for real-time image recognition.
    - Deployment:
        - Deploy the system in a healthcare environment, ensuring compatibility with existing infrastructure.

Block Diagram:

```
+----------------------+      +------------------------+      +--------------------------------+
|    Input Image   |          |    Preprocessing |        |  Feature Extraction     |
|                      |  →   | (Normalization,  |  →  |  (ROI Selection,           |
|                      |        |  Resize, Noise    |       |  Edge Detection,          |
|                      |        |  Reduction)        |       |   Texture Analysis)      |
+----------------------+      +------------------------+      +--------------------------------+
```

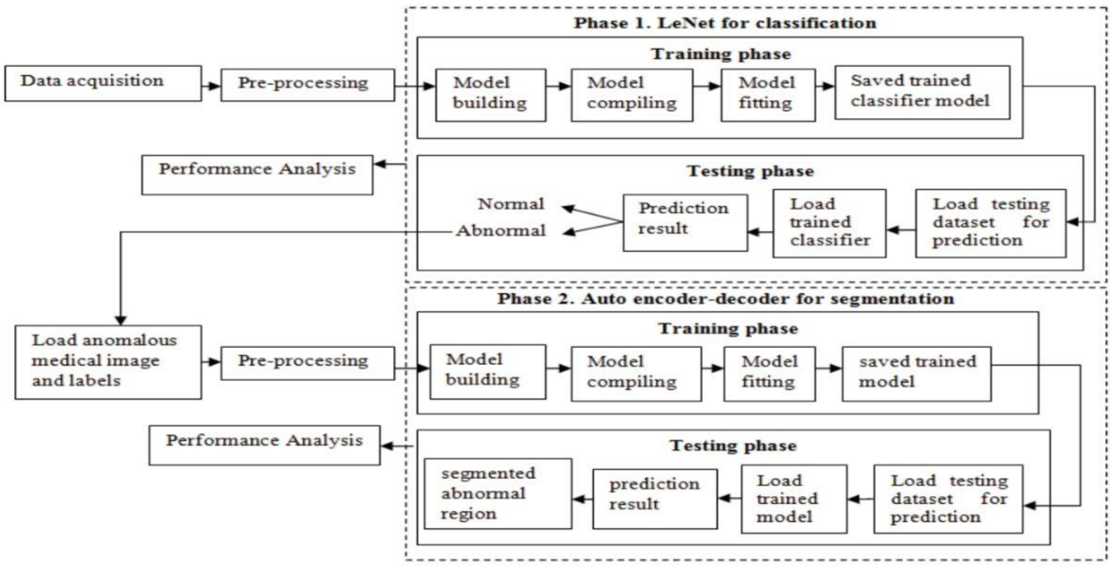
```
              ↓
+------------------------------+
|  Machine Learning  |
|  Model Training     |
+------------------------------+
              ↓
  +--------------------------+
  | Model Integration  |
  | with OpenCV         |
  +--------------------------+
              ↓
+------------------------------+
|  Real-time Image     |
|  Recognition          |
+------------------------------+
              ↓
+------------------------------+
|  Output/Diagnosis  |
+------------------------------+
```

Flowchart:

Start

|--> Input Image --> Preprocessing --> Feature Extraction --> Model Training --> Model Integration --> Real-time Image Recognition --> Output/Diagnosis

End



Algorithm:

{Sample Algorithm (Pseudocode)}:

```
# Import necessary libraries
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Step 1: Data Preprocessing
# ...

# Step 2: Feature Extraction
# ...

# Step 3: Model Training
def train_model(features, labels):
    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2,
random_state=42)

    # Choose a machine learning algorithm (SVM is used as an example)
    model = SVC(kernel='linear', C=1)

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the test set
    predictions = model.predict(X_test)

    # Evaluate model performance
    accuracy = accuracy_score(y_test, predictions)
    print(f"Model Accuracy: {accuracy}")

    return model

# Step 4: Model Integration with OpenCV
# ...

# Step 5: Real-time Image Recognition
# ...

# Step 6: Output/Diagnosis
# ...

# Main program
if __name__ == "__main__":
    # Execute the steps sequentially
```

```
    features, labels = preprocess_data()  # Assuming we have a function for data preprocessing
    trained_model = train_model(features, labels)
    integrate_with_opencv(trained_model)  # Assuming we have a function for OpenCV integration
    perform_real_time_recognition()  # Assuming we have a function for real-time image recognition
    generate_output()  # Assuming we have a function for generating the final diagnosis
```

This algorithm provides a high-level overview of the steps involved in brain stroke image recognition and processing. The actual implementation will require detailed coding for each step, and the parameters of the machine learning model, as well as the preprocessing and feature extraction techniques, should be fine-tuned based on the characteristics of the dataset and the specific requirements of the project.

Conclusion:

In conclusion, the development of a brain stroke image recognition and processing system using OpenCV holds great promise for enhancing the efficiency and accuracy of stroke diagnosis in the medical field. Through the integration of computer vision techniques, this project aims to contribute to timely and effective medical interventions, ultimately improving patient outcomes.

Acknowledgement:

We would like to express our sincere gratitude and appreciation to all those who have contributed to the successful completion of the project on "Brain Stroke Detection using OpenCV."

We are profoundly grateful to Prof. Pratik Bhattacharya for his invaluable guidance, expertise, and unwavering support throughout the entire duration of the project. Their insights and encouragement have been instrumental in shaping the direction and success of this research.

We extend our thanks to the faculty members of Computer Science Department SNU for their academic support, constructive feedback, and encouragement. Their dedication to fostering a conducive learning environment has been a cornerstone in the development of this project.

A heartfelt thank you to our fellow research team members for their collaborative spirit, shared enthusiasm, and collective efforts in tackling the challenges presented by this project. Their commitment and teamwork have been integral to the project's accomplishments.

We extend our gratitude to the peer reviewers who provided valuable feedback during the project's development. Their constructive comments and suggestions have enhanced the quality and rigor of the research.

To our friends and families, thank you for your unwavering encouragement, understanding, and patience throughout this journey. Your support has been a constant source of motivation.

We acknowledge the contributions of the open-source community, particularly the developers and contributors to the OpenCV library. Their collaborative efforts have provided a robust foundation for implementing computer vision solutions, including the focus of this project.

We are grateful for the support received from Sister Nivedita University. The institutional resources and conducive research environment have played a pivotal role in the successful execution of this project.

This project would not have been possible without the collective effort, encouragement, and expertise of each individual and organization mentioned above. Your contributions have been indispensable, and we are sincerely thankful for the positive impact you have had on the project's outcomes.

Anupam Kumar Ram

Riddhith Banerjee

Shivam Basak

Kuntal Mondal

Students of CSE Dept, SNU

29.11.2023