Alzheimer's Detection Using Convolutional Neural Networks

Milind Sharma & Tanishq Tyagi

November 4, 2024

Introduction

Alzheimer's disease (AD) is a neurodegenerative condition characterized by memory loss, cognitive impairment, and behavioral changes. It is the most prevalent form of dementia, affecting millions of people globally, with increasing incidence rates as populations age. The progressive nature of AD means that early detection is vital for implementing timely interventions, enhancing the quality of life for patients and their families. This project explores a machine learning-based approach that utilizes Convolutional Neural Networks (CNNs) to classify brain images and predict the stages of Alzheimer's disease. By leveraging the capabilities of CNNs, we aim to create a model that can effectively analyze medical imaging data, facilitating quicker and more accurate diagnoses.

Literature Review

Previous studies have shown that CNNs can effectively classify medical images, including those of the brain, for early detection of neurological diseases. CNNs utilize multiple layers of convolutional filters to automatically extract and learn features from image data, making them particularly adept at identifying complex patterns. Research has demonstrated that CNNs could achieve high accuracy in classifying Alzheimer's MRI scans, surpassing traditional methods. Furthermore, studies have explored the benefits of transfer learning, where models pre-trained on large datasets are fine-tuned for specific tasks, resulting in improved performance even with smaller datasets. The literature emphasizes the growing importance of machine learning in enhancing diagnostic processes in healthcare, particularly for neurodegenerative diseases like Alzheimer's.

Problem Statement

The goal of this project is to design a CNN model capable of classifying brain MRI images into different stages of Alzheimer's disease: Non-Demented (ND),

Very Mild Dementia (VMD), Mild Dementia (MD), and Moderate Dementia (MOD). Current diagnostic methods, often reliant on subjective interpretation by clinicians, can lead to delays and inaccuracies. By automating the classification process, we aim to develop a reliable tool that assists healthcare professionals in making informed decisions and provides patients with timely access to treatment. This model will not only aim to increase the accuracy of Alzheimer's detection but also strive for a user-friendly interface that can be utilized in clinical settings.

Objectives

The primary objectives of this project are:

- Develop a CNN model tailored specifically for the classification of Alzheimer's disease stages based on brain MRI images.
- Create a user-friendly interface for loading images and displaying predictions, making the tool accessible to healthcare professionals with varying levels of technical expertise.
- Analyze model performance and optimize for accuracy and speed, ensuring the model operates efficiently in real-world clinical environments.
- Explore the integration of additional imaging modalities and patient data to enhance the model's robustness and applicability.
- Disseminate findings through documentation, providing detailed insights into the methodology, results, and practical applications of the developed model.

Methodology

Requirement Analysis

The success of this project hinges on the availability of a high-quality, labeled dataset of brain MRI images. We have sourced a dataset from the Open Access Series of Imaging Studies (OASIS), which provides a comprehensive collection of MRI scans with associated clinical information. In addition to the dataset, we will utilize a deep learning framework such as TensorFlow, along with Keras for building and training the CNN model. Tools for data preprocessing and augmentation are also essential to enhance the training process and mitigate issues related to overfitting.

Algorithm Implementation

To ensure a balanced dataset, we will create a smaller, curated dataset by selecting a limited number of images from each category. This approach helps in maintaining class balance and allows for effective training of the model.

Listing 1: Creating a Smaller Dataset

```
import os
import shutil
import random

def create_smaller_dataset(source_dir, dest_dir,
    ↪ num_images_per_class=250):
    if not os.path.exists(dest_dir):
        os.makedirs(dest_dir)
    for class_dir in os.listdir(source_dir):
        class_path = os.path.join(source_dir, class_dir)
        dest_class_path = os.path.join(dest_dir, class_dir)
        if not os.path.exists(dest_class_path):
            os.makedirs(dest_class_path)
        images = os.listdir(class_path)
        random.shuffle(images)
        selected_images = images[:num_images_per_class]
        for img in selected_images:
            shutil.copy(os.path.join(class_path, img), os.
                ↪ path.join(dest_class_path, img))
```

This code snippet demonstrates how we will create a smaller dataset by randomly selecting a specified number of images from each class. This process not only ensures a balanced representation of classes but also aids in preventing model bias during training.

System Development

The CNN model was implemented using TensorFlow's Keras API. Our architecture employs several convolutional layers designed to extract hierarchical features from the input images, followed by dense layers that perform classification based on these features.

Listing 2: CNN Model Definition

```
from tensorflow.keras import layers, models

def create_cnn_model(input_shape):
    model = models.Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation='relu',
        ↪ input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(4, activation='softmax'))  # Four
        ↪  classes
    return model
```

This code outlines the creation of a CNN model, where the architecture consists of multiple convolutional layers to capture different levels of image features, and a final output layer that uses the softmax activation function to classify the input images into one of the four specified categories.

Testing and Validation

The model was tested with a subset of the dataset to validate its accuracy across different stages of Alzheimer's disease. We utilized an 'ImageDataGenerator' to facilitate efficient loading and augmentation of the images during training.

Listing 3: Preparing Data

```
from tensorflow.keras.preprocessing.image import
    ↪ ImageDataGenerator

def prepare_data(image_directory):
    datagen = ImageDataGenerator(rescale=1./255,
        ↪ validation_split=0.2)
    train_generator = datagen.flow_from_directory(
        image_directory,
        target_size=(64, 64),
        batch_size=64,
        class_mode='categorical',
        subset='training'
    )
    validation_generator = datagen.flow_from_directory(
        image_directory,
        target_size=(64, 64),
        batch_size=64,
        class_mode='categorical',
        subset='validation'
    )
    return train_generator, validation_generator
```

This implementation effectively splits the dataset into training and validation sets, applying real-time data augmentation to improve the model's generalization capabilities. The training generator feeds images into the model while maintaining the aspect ratio, which is crucial for ensuring that important features are preserved.

Performance Evaluation

We evaluate the model's performance based on various metrics, including accuracy, loss, precision, and recall. This multifaceted evaluation provides insights into the model's ability to correctly classify each stage of Alzheimer's disease. The use of a confusion matrix will further help visualize performance across different classes.

Optimization

To improve performance, several model parameters were optimized, including filter sizes and learning rates. Hyperparameter tuning was conducted using techniques such as grid search and random search to identify the optimal settings for our CNN model.

Documentation and Reporting

The project documentation will encompass a detailed report that outlines the CNN architecture, dataset preprocessing steps, training procedures, and performance metrics. A comprehensive user manual will also be developed, providing instructions for deploying the model in clinical settings.

Schedule

- **Week 1-2:** Requirement analysis and dataset preparation.
- **Week 3-4:** Development of the CNN model and initial training.
- **Week 5-6:** Performance evaluation and optimization of the model.
- **Week 7-8:** Development of the user interface for model interaction.
- **Week 9-10:** Testing and validation of the model and interface.
- **Week 11-12:** Preparation of the final report and presentation of findings.

System Requirements

- **Software:** Python 3.x, TensorFlow, Keras, Tkinter for GUI, and additional libraries such as NumPy and Matplotlib for data manipulation and visualization.
- **Hardware:** A high-performance GPU is recommended for training the model effectively, with a minimum of 16 GB of RAM to handle data processing tasks efficiently.
- **Dataset:** Brain MRI dataset sourced from Kaggle: ImagesOASIS, ensuring that the dataset is representative and comprehensive for accurate classification.