**Generalizable Machine Learning Model For Early Detection Of Alzheimer’s From Structural MRIs**

Alzheimer’s Disease (AD) is a neurodegenerative disorder affecting the aging population worldwide. AD worsens, individuals struggle with daily activities, creating significant challenges for both patients and caregivers. Early detection is crucial, as timely medical intervention can slow its progression.

Machine learning (ML) techniques, particularly deep learning models, are highly effective in analyzing MRI scans. We have implemented CNN and their architectures-ResNet101 and EffecientNetB0re

**Dataset Collection and splitting:**

The dataset used for Alzheimer’s disease detection is sourced from kaggle and comprises 6400 MRI scan images categorized into four classes:

Non- Demented - (3200)

Very Mild Demented - (2240)

Mild Demented - (896)

Moderate Demented - (64)

Out of 6400 images we have classified 80% - train data, 10% - test and 10% - validation.  
 Training set (5119), Test set (642), Validation set (639)

**Data Preprocessing:**

* Preprocessing is a crucial step in medical image analysis, especially for deep learning models like CNNs. It improves image quality, enhances feature extraction, and ensures consistent input for neural networks.
* To standardize input dimensions for deep learning models, all MRI images are resized to 128×128 pixels. Rescaling the height and width ensures uniformity across samples.
* Pixel intensity values are normalized to fall within the [0,1] range, improving computational efficiency and convergence during training.
* Data augmentation is an essential preprocessing technique helps improving classification efficiency, model generalization and minimizes overfitting.
* A variety of augmentation techniques include rotation, scaling, are applied to improve efficiency.
* These transformations enable CNN, ResNet101, and EfficientNetB0 to learn robust and invariant features, improving their ability to classify different AD stages with high accuracy.

**Model Selection and training:**

We have selected three models CNN, ResNet101 and EffecientNetB0.

* CNN is a deep learning model specifically designed for image classification. It consists of convolutional layers that extract important features, followed by pooling layers to reduce dimensionality. Dropout (20%) of connections to prevent overfitting.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| CNN | 98.36% | 0.984 | 0.983 | 0.983 |
| ResNet101 | 85.9% | 0.865 | 0.855 | 0.975 |
| EfficientNetB0 | 71.09% | 0.719 | 0.699 | 0.917 |

* ResNet101 is a 101-layer deep residual network known for its skip connections (residual connections). These connections prevent the vanishing gradient problem, allowing very deep networks to learn efficiently and can learn complex patterns effectively.
* EfficientNetB0 is a highly optimized CNN architecture that balances model depth, width, and resolution. EfficientNetB0 is the smallest version but offers excellent accuracy with fewer parameters.
* Transfer Learning is used where pretrained ImageNet is fine tuned and trained on our dataset.

**Model Evaluation:**

To measure model performance the following evaluation metrics are used to evaluate models performance accuracy, precision, recall, AUC.

**Technologies Used:**

* TensorFlow: An open-source machine learning framework used for building and training deep learning models.
* Keras: A high-level neural networks API that runs on top of TensorFlow. It provides an intuitive interface for designing and training models.
* Pandas: A powerful data manipulation library used for data preprocessing and analysis.
* Matplotlib: A popular plotting library used for data visualization, including the visualization of MRI images and performance metrics.
* NumPy: A fundamental library for scientific computing in Python, used for numerical operations and array manipulation.
* Scikit-learn: A machine learning library that provides tools for data preprocessing, model evaluation, and performance metrics.

**Implementation:**

* To implement this project we have install all the packages tensorflow, matplotlib, numpy, pandas and streamlit framework for frontend.
* Run the models first and the run frontend i.e streamlit by using following command streamlit run app1.py
* We have created a UI where user can upload MRI scan and the model classifies stage of Alzheimer’s.

**Results:**

A group of different colored bars

AI-generated content may be incorrect. **A screenshot of a medical device

AI-generated content may be incorrect.**