

Implementation and Analysis of Predictive Models for Alzheimer's and Amyotrophic Lateral Sclerosis Diseases Stage Detection using SMOTE

ILLURI USHASRI¹, Prof. K. BABULU²

¹Department of Electronics & Communication Engineering & JNTU-GV College Of Engineering, Vizianagaram

²Department of Electronics & Communication Engineering & JNTU-GV College of Engineering, Vizianagaram

Abstract - Early diagnosis of neurodegenerative diseases is important for effective disease management and treatment planning. This paper presents a software-hardware co-design framework for six-class MRI-based classification of ALS, three Alzheimer's Disease stages, Normal, and Others. During software training, pre-processed MRI images are used with pre-trained EfficientNetB0 for feature extraction, while SMOTE is applied only to the extracted training features to address class imbalance. A lightweight Custom CNN classifier is then trained using the balanced feature vectors. After training, the CNN weights and biases are converted to fixed-point representation and stored in hardware-compatible memory files. The trained classifier is implemented using Verilog HDL, synthesized in AMD Vivado, and integrated as a custom IP core on the PYNQ-Z2 FPGA. AXI-Stream and AXI DMA provide communication between the Processing System and Programmable Logic for hardware inference. The proposed framework demonstrates the feasibility of combining EfficientNetB0 feature extraction, SMOTE-based training, lightweight CNN classification, and FPGA acceleration to provide a resource-efficient and scalable solution for embedded MRI-based neurodegenerative disease classification.

Key Words: Alzheimer's, ALS, SMOTE, Lightweight CNN, FPGA, Software-Hardware Co-Design.

I. INTRODUCTION

Neurodegenerative diseases such as Alzheimer's Disease (AD) and Amyotrophic Lateral Sclerosis (ALS) represent major healthcare challenges due to their progressive effects on cognitive and motor functions. Early and reliable disease-stage identification is important for treatment planning and patient management. Magnetic Resonance Imaging (MRI) provides detailed structural information about neurological tissues without ionizing radiation and is therefore an important imaging source for automated disease analysis. However, manual MRI interpretation can be time-consuming and may be affected by variations in image quality and clinical interpretation. Recent advances in artificial intelligence, particularly Convolutional Neural Networks (CNNs) and transfer learning, have enabled automated medical image

classification. In the proposed work, pre-trained EfficientNetB0 is used for high-level feature extraction from pre-processed MRI images. Since medical datasets commonly suffer from class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is applied exclusively to the extracted training features to generate synthetic minority-class samples. The resulting balanced feature vectors are used to train a lightweight custom CNN classifier. EfficientNetB0 and SMOTE are restricted to the software training stage and are not implemented in FPGA inference.

To support hardware-oriented deployment, the trained lightweight classifier parameters are extracted, converted into fixed-point representation, and organized into memory initialization files. The classifier is implemented using Verilog HDL and integrated as a custom hardware accelerator on the PYNQ-Z2 platform using AMD Vivado Design Suite. The proposed hardware-software co-design architecture uses the Zynq Processing System for preprocessing, feature extraction, data preparation, and control, while the Programmable Logic performs classifier inference. AXI DMA and AXI-Stream interfaces provide data communication between the processing system and FPGA accelerator.

The proposed framework performs six-class MRI classification comprising Alzheimer's Disease Stage 1, Alzheimer's Disease Stage 2, Alzheimer's Disease Stage 3, ALS, Normal, and Others. By combining EfficientNetB0-based feature extraction, SMOTE-based class balancing, lightweight CNN classification, fixed-point parameter preparation, and FPGA acceleration, the proposed work provides a compact and scalable VLSI-oriented framework for embedded MRI-based neurodegenerative disease classification and future edge-AI healthcare applications.

II. RELATED WORK

Deep learning has become one of the most promising technologies for automatic medical image analysis. Numerous researchers have proposed CNN-based approaches for detecting neurological disorders using MRI images. This section reviews previous work related to Alzheimer's Disease detection, ALS classification, MRI-based deep learning, transfer learning, FPGA acceleration, and identifies the research gap addressed in this work.

A. Alzheimer's Disease (AD) Detection

Alzheimer's Disease (AD) has been extensively investigated using machine learning and deep learning techniques because early detection can support clinical decision-making and patient management. Magnetic Resonance Imaging (MRI) is one of the widely used imaging modalities for identifying structural changes associated with Alzheimer's Disease. Deep learning-based approaches, particularly CNNs, have demonstrated the ability to automatically extract discriminative spatial features from MRI images without requiring extensive handcrafted feature engineering.

Several studies have investigated transfer learning architectures such as VGG16, ResNet50, DenseNet, MobileNet, and EfficientNet for Alzheimer's Disease detection and stage classification. These approaches can provide strong classification performance by utilizing feature representations learned from large-scale datasets. However, many existing studies focus primarily on binary disease detection or a limited number of Alzheimer's stages. In addition, computationally intensive deep learning architectures may be difficult to directly deploy on resource-constrained FPGA platforms.

B. Amyotrophic Lateral Sclerosis (ALS) Detection

Compared with Alzheimer's Disease, research on MRI-based deep learning for Amyotrophic Lateral Sclerosis (ALS) remains relatively limited, partly because of the limited availability of suitable publicly accessible MRI datasets. Existing studies have predominantly investigated the discrimination of ALS patients from healthy subjects rather than comprehensive multi-class classification involving multiple neurological conditions.

Furthermore, comparatively fewer studies investigate the deployment of ALS-related MRI classification models on embedded hardware platforms. This creates an opportunity for developing computationally efficient AI-based approaches capable of supporting ALS identification while also considering hardware acceleration and embedded deployment requirements.

C. MRI-Based Deep Learning and Transfer Learning

CNN-based deep learning models have become widely adopted for medical image classification because they can automatically learn hierarchical representations from image data. Transfer learning further improves the practicality of deep learning by utilizing feature representations learned from large-scale datasets and adapting them to medical image classification tasks. EfficientNetB0 provides an effective balance between feature extraction capability and computational complexity compared with larger deep learning architectures. In the proposed work, EfficientNetB0 is used as a software-based feature extractor to obtain high-level feature representations from pre-processed MRI images. These extracted feature vectors are subsequently used for training the lightweight classifier. Although

EfficientNetB0 provides effective feature representations, implementing the complete network directly in FPGA hardware would introduce considerable computational and memory requirements. Therefore, the proposed framework separates software-based feature extraction from the lightweight classification stage targeted for hardware implementation.

D. Class Imbalance Handling

Class imbalance is a significant challenge in medical image classification because some disease categories may contain substantially fewer samples than majority classes. Training a classifier directly on such imbalanced data can result in biased predictions toward majority classes and reduced recognition of minority disease categories. The Synthetic Minority Oversampling Technique (SMOTE) is a widely used approach for addressing class imbalance. Instead of simply duplicating minority samples, SMOTE generates synthetic samples based on neighbouring minority-class feature vectors. This can provide a more balanced training distribution and improve the classifier's ability to learn minority classes.

In the proposed framework, SMOTE is applied only to the training feature vectors obtained after EfficientNetB0 feature extraction. The validation and test datasets are not synthetically oversampled. SMOTE is therefore part of the software training stage only and is not implemented in the FPGA hardware or executed during inference. The FPGA uses the trained classifier parameters obtained after completion of the software training process.

E. FPGA-Based Medical Image Classification

Field Programmable Gate Arrays (FPGAs) have attracted considerable interest for accelerating deep learning inference because they provide configurable hardware parallelism, deterministic execution, and the potential for improved energy efficiency. Unlike general-purpose processors, FPGA architectures can be customized for the computational requirements of a specific application. Zynq-based FPGA platforms are particularly suitable for hardware-software co-design because they combine a Processing System (PS) with Programmable Logic (PL). The PYNQ-Z2 platform provides an accessible environment for integrating Python-based software applications with custom FPGA hardware. Communication between the PS and PL can be established using Advanced eXtensible Interface (AXI) protocols, while AXI Direct Memory Access (DMA) can be used to transfer data efficiently between processor memory and hardware accelerators.

Previously, FPGA-based CNN implementations have demonstrated the potential for reducing inference latency and improving computational efficiency through hardware parallelism and dedicated arithmetic resources. However, deploying large transfer-learning networks directly on resource-constrained FPGA platforms can

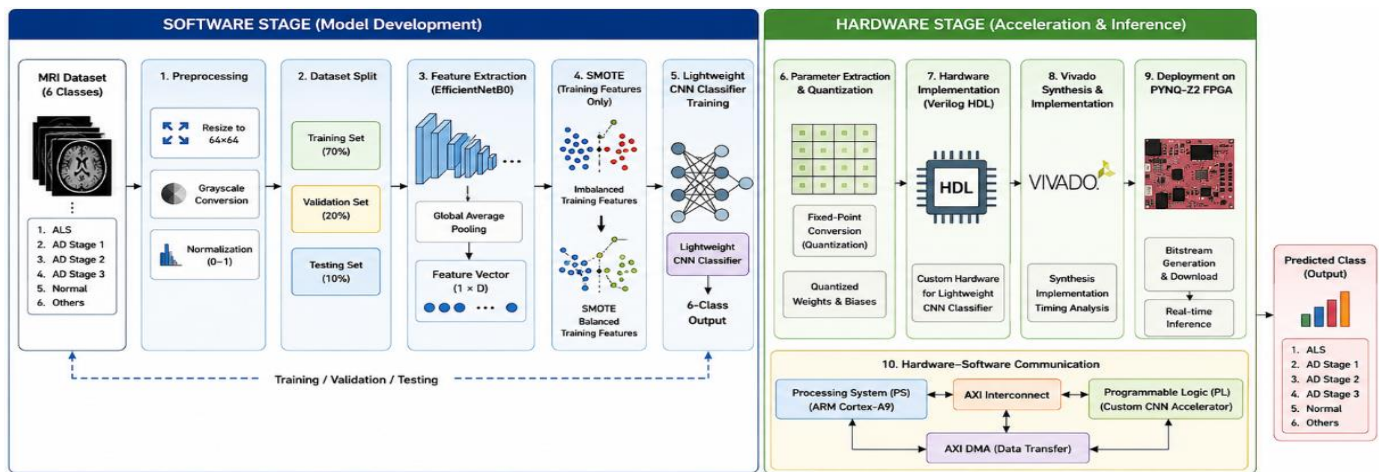


Fig.1. Overall Proposed Software-Hardware Co-Design Framework

require substantial hardware resources. Consequently, lightweight Custom CNN architectures and optimized fixed-point implementations are attractive alternatives for FPGA-based medical image classification.

F. Research Gaps

Although substantial research has been conducted independently on Alzheimer's Disease detection, ALS classification, transfer learning, class imbalance handling, and FPGA-based CNN acceleration, several limitations remain. Many existing studies concentrate on a single neurological disorder or binary classification, while fewer studies investigate unified multi-class classification involving Alzheimer's Disease stages, ALS, normal MRI scans, and other neurological conditions. In addition, computationally intensive transfer learning architectures are not always suitable for direct implementation on resource-constrained FPGA platforms.

Another important limitation is the presence of class imbalance in medical imaging datasets. Although SMOTE has been widely used to improve minority-class representation, its integration with a hardware-oriented classification framework is less commonly considered. Furthermore, the separation of computationally intensive feature extraction from a lightweight hardware-compatible classifier provides an opportunity to reduce FPGA implementation complexity.

To address these limitations, the proposed work develops a unified six-class MRI classification framework comprising ALS, Alzheimer's Disease Stage 1, Alzheimer's Disease Stage 2, Alzheimer's Disease Stage 3, Normal, and Others. EfficientNetB0 is employed for software-based feature extraction, while SMOTE is applied exclusively to the training feature vectors to address class imbalance. A lightweight classifier is subsequently trained using the balanced feature representation, and its parameters are prepared for FPGA implementation using fixed-point representation and memory initialization files. The hardware classification stage is implemented using Verilog HDL and integrated with the PYNQ-Z2 platform through AXI-based communication and DMA. This combination establishes a practical VLSI-embedded

hardware-software co-design framework for efficient MRI-based neurological disease classification.

III. PROPOSED METHODOLOGY

The proposed methodology presents a software-hardware co-design framework for six-class classification of brain Magnetic Resonance Imaging (MRI) images corresponding to Amyotrophic Lateral Sclerosis (ALS), Alzheimer's Disease Stage 1, Alzheimer's Disease Stage 2, Alzheimer's Disease Stage 3, Normal, and Others. The framework separates the software-based model development stage from the hardware-oriented classification stage. MRI images are first organized and preprocessed in the software environment. EfficientNetB0 is employed as a transfer-learning-based feature extractor to obtain high-level feature representations from the preprocessed MRI images. SMOTE is subsequently applied exclusively to the training feature vectors to address class imbalance. A lightweight classifier is then trained using the balanced training features. After training, the classifier parameters are extracted, quantized into a hardware-compatible representation, and prepared for implementation using Verilog HDL. The resulting hardware architecture is synthesized and implemented using AMD Vivado Design Suite and prepared for deployment on the PYNQ-Z2 platform. The overall methodology is illustrated in Fig. 1.

Fig.1 illustrates the complete workflow of the proposed system. The software stage begins with MRI dataset organization and image preprocessing, followed by EfficientNetB0-based feature extraction. The extracted feature vectors are divided into training, validation, and testing data, with SMOTE applied only to the training features to address class imbalance. A lightweight classifier is trained using the balanced training features and evaluated using standard classification metrics. Following successful software training, the classifier parameters are extracted, converted into a suitable fixed-point representation, and stored as memory initialization files for hardware implementation. The hardware stage

consists of Verilog HDL implementation, Vivado synthesis and implementation, bitstream generation, and PYNQ-Z2 deployment. The software and hardware components communicate through the Zynq Processing System (PS), AXI infrastructure, and DMA-based data transfer where applicable.

A. MRI Dataset

The proposed framework utilizes brain MRI datasets representing Alzheimer's Disease, Amyotrophic Lateral Sclerosis (ALS), Normal brain conditions, and Other neurological conditions. The Alzheimer's MRI dataset is organized into three disease stages corresponding to Very Mild Dementia, Mild Dementia, and Moderate Dementia. These categories are represented in the proposed framework as Alzheimer's Disease Stage 1, Stage 2, and Stage 3, respectively. Together with ALS, Normal, and Others, the proposed system performs six-class classification. The MRI images are collected from publicly available datasets and organized according to their corresponding class labels. Dataset organization is performed before preprocessing so that each image can be associated with its corresponding disease category.

Table 1: MRI Dataset Description

Class	Description
ALS	Amyotrophic Lateral Sclerosis
Alzheimer's Stage1	Very Mild Dementia
Alzheimer's Stage 2	Mild Dementia
Alzheimer's Stage3	Moderate Dementia
Normal	Normal / Healthy brain MRI
Others	Other Neurological or Non-target brain conditions

Table 1 summarizes the six diagnostic categories considered in the proposed classification framework. The three Alzheimer's categories represent different stages of disease progression, while ALS, Normal, and Others provide additional classes for multi-class neurological classification.

B. Image Preprocessing

Image preprocessing is performed to standardize the MRI images and prepare them for feature extraction and subsequent classification. Each MRI image is resized to 64×64 pixels to obtain a uniform input dimension and reduce computational and memory requirements. The images are converted into grayscale because the classification task relies primarily on structural intensity information rather than colour information. Following grayscale conversion, the pixel intensity values are normalized to the range 0–1. The normalized images are then supplied to the software-based feature extraction stage. The same basic preprocessing procedure must be maintained during inference so that the input representation remains consistent with the representation used during model development.

C. Data Splitting and Feature Extraction

After dataset organization and preprocessing, the dataset is divided into training, validation, and testing subsets.

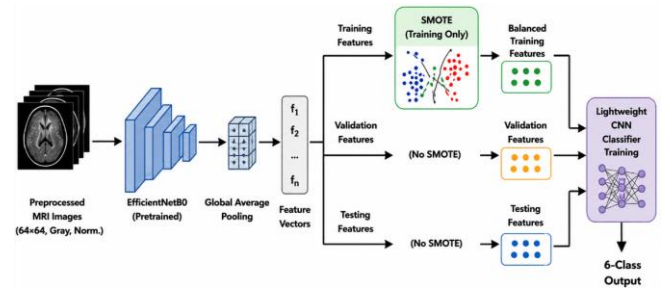


Fig. 2. EfficientNetB0-based Feature Extraction and Lightweight Custom CNN training workflow.

The splitting process is performed before SMOTE so that information from the validation and test sets does not enter the synthetic oversampling process. EfficientNetB0 with pretrained ImageNet weights is used as a transfer-learning-based feature extractor. The pretrained classification head is excluded, and the network is used to obtain high-level feature representations from the preprocessed MRI images. Global Average Pooling is subsequently applied to convert the extracted feature maps into compact feature vectors. The extracted training, validation, and testing feature vectors are maintained separately for subsequent classifier development and evaluation.

Figure 2 shows the software-side training workflow. Preprocessed MRI images are passed through EfficientNetB0 to obtain high-level feature representations. The extracted training features are subsequently balanced using SMOTE, while validation and testing features remain unchanged. The balanced training features are used to train the lightweight classifier, and the validation and testing features are used for model evaluation.

D. SMOTE-Based Class Balancing

Medical MRI datasets commonly exhibit class imbalance because some disease categories contain substantially fewer samples than others. Such imbalance may cause a classifier to favor majority classes and consequently reduce its ability to recognize minority disease categories. To address this problem, the proposed framework employs the Synthetic Minority Over-sampling Technique (SMOTE) during the software training stage. SMOTE is applied after EfficientNetB0 feature extraction to the training feature vectors. Synthetic feature vectors are generated for minority classes based on neighbouring minority samples, thereby improving the balance of the training data. And most importantly, SMOTE is not applied to the validation or testing data, and it is not implemented in the FPGA hardware. During inference, the hardware uses only the trained classifier parameters obtained after completion of the software training process. This separation prevents synthetic test information from influencing model evaluation and ensures that the FPGA

implementation does not require an additional oversampling mechanism.

E. Lightweight Custom CNN Classifier Architecture

The proposed Lightweight Custom CNN classifier is designed to provide a suitable balance between classification performance and hardware implementation complexity. Unlike a complete large-scale transfer learning network, the lightweight classifier contains a reduced number of trainable parameters and computational operations, making it more suitable for FPGA-based implementation. During the software development stage, EfficientNetB0 is used as a feature extraction network to obtain high-level representations from the preprocessed MRI images. The extracted feature vectors are then used for classifier development. To address the class imbalance present in the training data, SMOTE is applied only to the training feature vectors. The resulting balanced training features are used to train the Lightweight Custom CNN classifier.

The classifier consists of fully connected layers that learn the relationship between the extracted feature representations and the six target classes: ALS, Alzheimer's Disease Stage 1, Alzheimer's Disease Stage 2, Alzheimer's Disease Stage 3, Normal, and Others. Nonlinear activation, normalization, and dropout operations are employed during software training to improve learning stability and reduce overfitting. The final classification layer produces six output scores corresponding to the six disease categories. After completion of software training, the learned classifier parameters are extracted for hardware implementation. The trained parameters are subsequently converted into a hardware-compatible fixed-point representation and used by the FPGA accelerator during inference. Thus, EfficientNetB0 and SMOTE are part of the software model-development/training pipeline, whereas the trained lightweight classifier parameters form the basis of the hardware inference implementation.

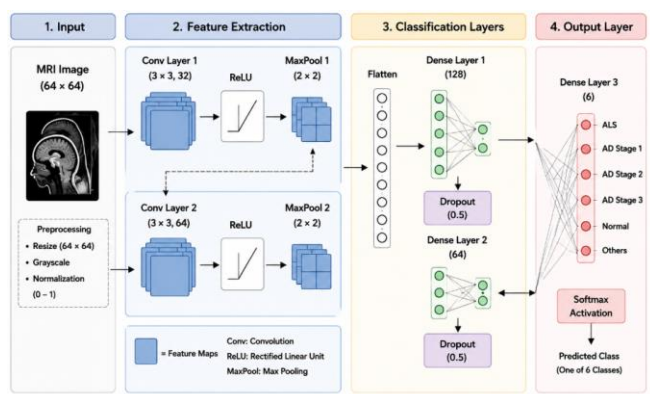


Fig. 3. Lightweight Custom CNN classifier architecture for six-class MRI classification.

Figure 3 illustrates the architecture of the proposed Lightweight Custom CNN classifier. The input to the classifier is the feature representation obtained from EfficientNetB0. The feature vector is processed through fully connected layers with nonlinear activation and regularization operations. The final output layer contains six class outputs corresponding to ALS, Alzheimer's Disease Stage 1, Alzheimer's Disease Stage 2, Alzheimer's Disease Stage 3, Normal, and Others. During software training, dropout is used to reduce overfitting, while the trained weights and biases are subsequently extracted and converted into a hardware-compatible format for FPGA inference.

F. FPGA Deployment Workflow

Following software-based model development and evaluation, the trained Lightweight Custom CNN classifier is prepared for FPGA implementation. The deployment workflow converts the trained neural-network parameters into a representation that can be processed efficiently by the programmable logic of the PYNQ-Z2 platform. Initially, the trained classifier parameters are extracted from the software model. The floating-point weights and biases are then converted into an appropriate fixed-point representation to reduce memory requirements and hardware computational complexity. The quantized parameters are organized into memory initialization files, such as .mem files, which can be loaded into the corresponding Verilog HDL memory structures.

The CNN computation modules are subsequently developed using Verilog HDL. The hardware modules implement the required classifier operations and are integrated into a custom CNN accelerator IP. The IP is then incorporated into the AMD Vivado hardware design together with the Zynq Processing System, AXI infrastructure, and AXI DMA interface. After integration, the complete hardware design undergoes synthesis and implementation to verify functional correctness, resource utilization, and timing performance. Once the design satisfies the required implementation constraints, the FPGA bitstream and corresponding hardware description file are generated. These files form the hardware overlay required for software-controlled FPGA operation through the PYNQ environment. During deployment, the Python/Jupyter Notebook application loads the generated overlay, prepares the input data, allocates DMA buffers, and transfers the data to the FPGA accelerator. The Lightweight Custom CNN accelerator performs inference in the programmable logic and returns the classification result to the Processing System for display.

Figure 4 presents the complete FPGA deployment workflow of the proposed system. The trained Lightweight Custom CNN classifier is first processed to



Fig. 4. FPGA deployment workflow for the proposed Lightweight Custom CNN classifier.

extract its learned weights and biases. These parameters are quantized from floating-point to fixed-point representation and converted into memory initialization files for hardware use. The corresponding CNN computational modules are implemented using Verilog HDL and packaged as a custom accelerator IP.

The custom IP is integrated with the Zynq Processing System and AXI DMA in the Vivado Block Design environment. The hardware design is then synthesized and implemented, followed by bitstream generation. The generated bitstream and hardware description file are used to prepare the PYNQ overlay. During operation, Python software running through the PYNQ environment loads the overlay, preprocesses the MRI input, transfers the required data through DMA, and initiates FPGA inference. The resulting disease-class prediction is returned to the Processing System and displayed to the user.

IV. FPGA IMPLEMENTATION

The trained Lightweight Custom CNN classifier is implemented as a custom hardware accelerator on the PYNQ-Z2 platform using a hardware–software co-design approach. The implementation divides the overall application between the Processing System (PS) and Programmable Logic (PL). Software operations such as image preprocessing, FPGA overlay management, DMA configuration, and result visualization are performed on the PS using Python and the Jupyter Notebook environment, while the computationally intensive CNN inference operations are implemented in the PL using Verilog HDL. The hardware design is developed, synthesized, and implemented using the AMD Vivado Design Suite.

A. Hardware Design

The Lightweight Custom CNN architecture is translated into a custom hardware accelerator using Verilog HDL. The accelerator consists of hardware modules corresponding to the principal CNN operations, including convolution, ReLU activation, max-pooling, fully connected computation, and class prediction. These

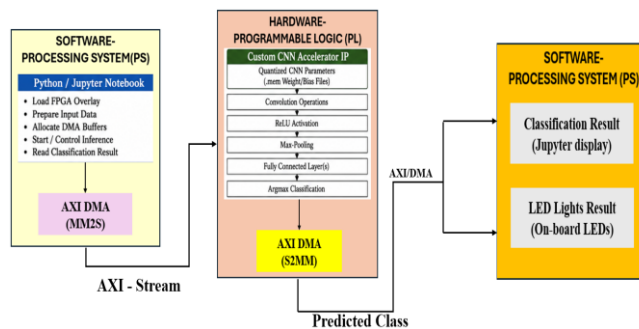


Fig. 5. Hardware–software co-design architecture of the proposed FPGA implementation.

modules are integrated to form the complete CNN inference pipeline and are packaged as a reusable custom IP core.

The custom CNN IP is integrated into the Vivado Block Design environment and interfaced with the Zynq Processing System. AXI-Stream communication is used for streaming input and output data, while the AXI Direct Memory Access (DMA) controller provides efficient data transfer between the Processing System and Programmable Logic with reduced processor intervention.

After integration, the complete design is subjected to synthesis and implementation using the AMD Vivado Design Suite. The resulting hardware design is used to generate the FPGA bitstream and corresponding hardware description required for deployment on the PYNQ-Z2 platform.

Figure.5 illustrates the hardware–software co-design architecture of the proposed FPGA implementation on the PYNQ-Z2 platform. The system is divided into a software domain executed by the Processing System (PS) and a hardware domain implemented in the Programmable Logic (PL). The software domain uses Python/Jupyter Notebook for overlay management, input-data preparation, DMA buffer allocation, inference control, and classification-result visualization. The trained CNN parameters obtained during the software development stage are converted into fixed-point representation and stored as memory initialization files (.mem). These quantized weights and biases are incorporated into the custom CNN accelerator implemented in the programmable logic.

During inference, the input data are transferred from the Processing System to the custom CNN accelerator through the AXI Direct Memory Access (DMA) controller using the AXI-Stream interface. The hardware accelerator performs the CNN computations using the preloaded quantized parameters, including convolution, activation, pooling, and fully connected operations, followed by Argmax-based classification. The resulting class label is transferred back to the Processing System through the DMA interface and displayed through the Python/Jupyter Notebook environment. Thus, the proposed architecture separates software-level control and data management from hardware-based CNN computation, enabling efficient FPGA-accelerated

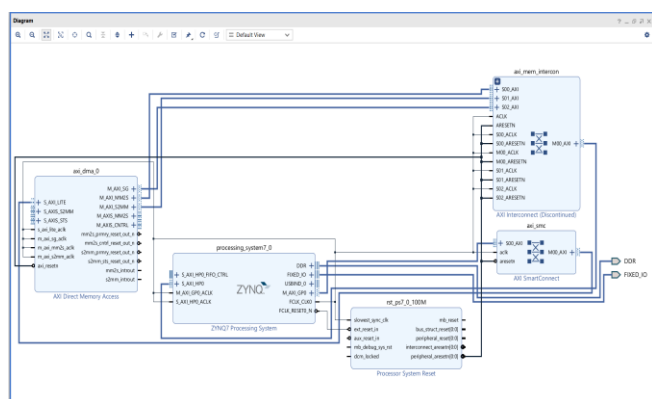


Fig. 6. Vivado Block Design of the proposed FPGA implementation.

inference using the previously trained and quantized model.

Figure 6 presents the Vivado Block Design used for FPGA implementation. The design integrates the Zynq Processing System, AXI DMA, custom CNN IP, AXI interfaces, clocking components, and associated interconnection logic required for communication between the Processing System and Programmable Logic.

B. Software Implementation

The software component of the proposed system is developed in Python using the Jupyter Notebook environment associated with the PYNQ framework. Following generation of the FPGA bitstream and hardware description, the corresponding overlay is loaded into the PYNQ environment for hardware interaction. For inference, the input MRI image is read and subjected to the required preprocessing operations, including resizing, grayscale conversion, and pixel normalization. The processed image data are arranged into a suitable one-dimensional buffer for transfer to the FPGA accelerator. Input and output buffers are allocated using the PYNQ Python interface, enabling communication between the Processing System and Programmable Logic.

The preprocessed image data are transferred to the custom CNN accelerator through the AXI DMA engine. The DMA controller streams the input data through the AXI-Stream interface to the Programmable Logic, where the Lightweight Custom CNN performs the required inference operations. After completion of inference, the predicted class information is returned to the Processing System and can be accessed and displayed through the Jupyter Notebook environment.

C. Hardware-Software integration

The proposed implementation follows a hardware–software co-design methodology in which software and hardware operations are partitioned according to their computational requirements. The Processing System executes Python-based image preprocessing, FPGA overlay management, DMA configuration, memory-buffer management, and classification-result visualization. The Programmable Logic performs the

computationally intensive operations of the Lightweight Custom CNN accelerator.

Communication between the Processing System and Programmable Logic is established using the AXI DMA controller and AXI-Stream interfaces. The Processing System transfers the preprocessed MRI data to the accelerator through the DMA engine, while the Programmable Logic performs CNN inference and returns the classification result through the corresponding communication path.

This partitioning reduces the computational burden on the processor and enables the CNN inference operations to be executed using dedicated programmable-logic resources. The architecture therefore provides a practical foundation for FPGA-based acceleration of lightweight neural-network inference in VLSI-embedded medical image classification applications.

The actual hardware performance, including inference latency, throughput, resource utilization, and power consumption, is evaluated using the synthesis, implementation, timing, and power reports generated by the AMD Vivado Design Suite and, where applicable, through PYNQ-Z2 board-level experiments.

V. RESULTS

The proposed software–hardware framework was evaluated in two stages: software-based model evaluation and FPGA implementation analysis. The software experiments were performed using the six MRI classes, namely ALS, Alzheimer’s Stage 1, Alzheimer’s Stage 2, Alzheimer’s Stage 3, Normal, and Others. The dataset was divided into training, validation, and testing subsets using a stratified 70:20:10 split. The software evaluation considered the EfficientNetB0-based feature extraction and SMOTE-assisted classifier as well as the Lightweight Custom CNN classifier developed for FPGA deployment. The FPGA implementation was subsequently evaluated using the synthesized hardware design and its resource utilization.

A. Experimental Configuration

The experimental configuration used for software model development and FPGA-oriented CNN training is summarized in Table II. The EfficientNetB0 feature-extraction path uses 112×112 input images, whereas the Lightweight Custom CNN classifier uses 64×64 grayscale images to reduce computational and memory requirements for FPGA implementation.

Table 2 summarizes the principal software and hardware configuration used in the proposed framework. The EfficientNetB0 path operates at 112×112 resolution and generates deep feature representations using Global Average Pooling. SMOTE is applied to the training feature vectors to address class imbalance. The training configuration in the supplied implementation uses the

Table 2: Experimental Configuration

Parameter	Configuration
Programming Language	Python
Deep Learning Framework	TensorFlow / Keras
Optimizer	Adam
Learning Rate	0.0001
Batch Size	32
Epochs	25
Dataset Split	70 / 20 / 10
Number of Classes	6
EfficientNetB0 Input Size	112 × 112
Lightweight Custom CNN Input Size	64 × 64
Lightweight Custom CNN Input Type	Grayscale
CNN Conv1	16 Filters, 3 × 3
CNN Conv2	32 Filters, 3 × 3
Dense Layer	64 neurons
Feature Extraction Model	EfficientNetB0
Feature Pooling	Global Average Pooling
Class Balancing	SMOTE
FPGA Platform	PYNQ-Z2
HDL	Verilog HDL
FPGA Design Tool	AMD Vivado Design Suite

Adam optimizer with a learning rate of 0.0001, a batch size of 32, and a maximum of 25 epochs. For FPGA-oriented CNN processing, the supplied implementation uses a 64 × 64 grayscale input and a compact CNN consisting of two convolution layers, max-pooling operations, a fully connected layer, and a six-class output layer.

B. Classification Results

The classification performance was evaluated using precision, recall, F1-score, and overall accuracy. The supplied implementation generates these metrics from the independent test set using the predicted and true class labels.

The six disease categories are explicitly evaluated as ALS, Alzheimer Stage 1, Alzheimer Stage 2, Alzheimer Stage 3, Normal, and Others. The classification analysis therefore provides class-wise information rather than relying only on overall accuracy.

Table 3: Classification Performance of the Proposed Lightweight Custom CNN Classifier

Class	Precision (%)	Recall (%)	F1-Score (%)
ALS	96.30	96.30	96.30
Alzheimer's Stage 1	95.53	95.14	95.33
Alzheimer's Stage 2	95.41	95.41	95.41
Alzheimer's Stage 3	93.88	93.88	93.88
Normal	97.62	98.40	98.01
Others	99.28	98.81	99.05
Macro Average	96.34	96.32	96.33
Weighted Average	96.79	96.79	96.78
Overall Accuracy	-----	-----	96.79

Table 3 presents the class-wise performance of the proposed Lightweight Custom CNN classifier on the test dataset. The classifier achieved an overall accuracy of 96.79%, with macro-average precision, recall, and F1-score of 96.34%, 96.32%, and 96.33%, respectively. The highest F1-score was obtained for the Others class at 99.05%, followed by the Normal class at 98.01%. Alzheimer's Stage 3 obtained the lowest F1-score of

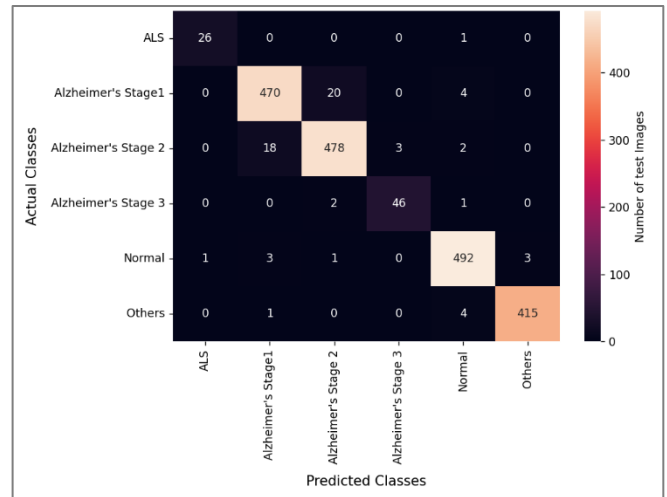


Fig. 7. Confusion matrix of the proposed six-class MRI Lightweight Custom CNN Classifier on the independent test set.

93.88%, which may be attributed to the relatively smaller number of test samples in this class. The results demonstrate that the proposed classifier provides consistent classification performance across all six MRI categories.

C. Confusion Matrix Analysis

The confusion matrix provides a class-wise representation of the prediction behavior of the proposed classifier. The rows correspond to the true disease classes, while the columns represent the predicted classes. The diagonal elements indicate correctly classified samples, whereas the off-diagonal elements indicate misclassifications.

The supplied code generates the test confusion matrix directly from the independent test labels and predictions and uses the six disease categories as both row and column labels.

Figure 7 illustrates the confusion matrix obtained from the test dataset containing 1,991 MRI images. The diagonal elements represent correctly classified samples, while the off-diagonal elements indicate misclassifications. A total of 1,927 samples were correctly classified, resulting in an overall accuracy of 96.79%. The largest misclassification occurs between Alzheimer's Stage 1 and Alzheimer's Stage 2, indicating that these two disease stages exhibit comparatively similar classification characteristics. The strong diagonal dominance across all six classes demonstrates the effectiveness of the proposed Lightweight Custom CNN classifier.

D. FPGA Resource Utilization

Following software model development, the CNN hardware design was implemented using Verilog HDL and synthesized using AMD Vivado Design Suite. The FPGA implementation uses convolutional and fully connected operations that are mapped to FPGA computational resources. The supplied Vivado synthesis log confirms the use of DSP resources for operations

associated with conv1, conv2, and the fully connected layer.

Table 4 FPGA Resource Utilization

FPGA Resource	Available	Used	Utilization (%)
LUTs	53,200	310	0.58
Flip-Flops (Registers)	106,400	37	0.03
BRAM Tiles	140	1	0.71
DSP Slices	220	67	30.45

Table 4 presents the post-synthesis resource utilization of the proposed CNN accelerator implemented for the PYNQ-Z2 platform. The synthesized design utilizes 310 LUTs, corresponding to 0.58% of the available LUT resources, while 37 flip-flops are utilized, representing only 0.03% of the available registers. The design uses one Block RAM tile, corresponding to 0.71% utilization. A total of 67 DSP slices are utilized, resulting in 30.45% DSP utilization. The low LUT, register, and BRAM utilization indicates that the proposed hardware architecture occupies a relatively small portion of the available programmable-logic resources, while the higher DSP utilization reflects the arithmetic computation associated with the CNN accelerator. These results demonstrate that the proposed lightweight Custom CNN architecture is suitable for implementation on the resource-constrained PYNQ-Z2 FPGA platform.

CONCLUSION

This work presented a hardware–software co-design framework for six-class MRI classification of ALS, Alzheimer’s Stage 1, Alzheimer’s Stage 2, Alzheimer’s Stage 3, Normal, and Others. EfficientNetB0 was used during the software stage for feature extraction, while SMOTE was applied only to the training data to address class imbalance. A lightweight Custom CNN classifier was subsequently trained and its learned parameters were prepared for fixed-point hardware implementation. The trained classifier achieved an overall test accuracy of 96.79%, with a macro-average F1-score of 96.33%. The CNN accelerator was implemented using Verilog HDL and synthesized for the PYNQ-Z2-compatible Zynq-7020 device. The post-synthesis results showed 0.58% LUT, 0.03% flip-flop, 0.71% BRAM, and 30.45% DSP utilization, demonstrating the low hardware resource requirement of the proposed lightweight architecture.

The proposed framework therefore demonstrates the feasibility of combining deep-learning-based MRI classification with FPGA acceleration. By separating software model development from hardware inference, the architecture provides a practical foundation for resource-efficient embedded AI applications.

Future work will focus on improving the proposed system through larger and more diverse MRI datasets, additional independent validation, and further

optimization of the lightweight Custom CNN architecture. The FPGA implementation can also be extended through post-implementation timing analysis, measured inference latency, power and energy evaluation, and further fixed-point optimization. Additional hardware parallelism and optimized memory/data-transfer architectures can be investigated to improve throughput. Finally, integration with more comprehensive clinical datasets and explainable AI techniques can be explored to improve the reliability and practical applicability of the proposed FPGA-based medical classification system.

REFERENCES

- [1] E. Kina, "TLEABLCNN: Brain and Alzheimer's Disease Detection Using Attention-Based Explainable Deep Learning and SMOTE Using Imbalanced Brain MRI," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3539550.
- [2] S. M. Mahim, M. S. Ali, M. O. Hasan, A. A. N. Nafi, A. Sadat, S. A. Hasan, B. Shareef, M. M. Ahsan, M. K. Islam, M. S. Miah, and M. B. Niu, "Unlocking the Potential of XAI for Improved Alzheimer's Disease Detection and Classification Using a ViT-GRU Model," IEEE Access, vol. 12, pp. 8390–8412, 2024, doi: 10.1109/ACCESS.2024.3351809.
- [3] S. E. Sorour, A. A. A. El-Mageed, K. M. Albarrak, A. K. Alnaim, A. A. Wafa, and E. El-Shafeiy, "Classification of Alzheimer's Disease Using MRI Data Based on Deep Learning Techniques," Journal of King Saud University – Computer and Information Sciences, vol. 36, no. 2, Art. no. 101940, Feb. 2024, doi: 10.1016/j.jksuci.2024.101940.
- [4] P. Goyal, R. Rani, and K. Singh, "A Multilayered Framework for Diagnosis and Classification of Alzheimer's Disease Using Transfer Learned AlexNet and LSTM," Neural Computing and Applications, vol. 36, no. 7, pp. 3777–3801, Mar. 2024, doi: 10.1007/s00521-023-09301-6.
- [5] M. Likhita, K. M. Kumar, N. S. Sasank, and M. Abhinaya, "AD-ResNet50: An Ensemble Deep Transfer Learning and SMOTE Model for Classification of Alzheimer's Disease," in Proc. International Conference on Innovative Computing and Communications, Singapore: Springer, 2023, pp. 699–713.
- [6] Alzheimer's Association, "2024 Alzheimer's Disease Facts and Figures," Alzheimer's & Dementia, vol. 20, no. 5, pp. 3708–3821, 2024.
- [7] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.
- [8] M. Basaia et al., "Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks," NeuroImage: Clinical, vol. 21, pp. 101645, 2019.
- [9] S. Payan and G. Montana, "Predicting Alzheimer's disease: A neuroimaging study with 3D convolutional neural networks," Pattern Recognition, vol. 63, pp. 355–365, 2017.
- [10] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems (NeurIPS), 2012.
- [11] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," Proceedings of IEEE CVPR, pp. 770–778, 2016.
- [12] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Proceedings of ICML, pp. 6105–6114, 2019.
- [13] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
- [14] N. Japkowicz and S. Stephen, "The Class Imbalance Problem: A Systematic Study," Intelligent Data Analysis, vol. 6, no. 5, pp. 429–449, 2002.
- [15] H. He and E. A. Garcia, "Learning from Imbalanced Data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.
- [16] Y. Ma, Y. Cao, S. Vrudhula and J. Seo, "Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks," Proceedings of ACM FPGA, pp. 45–54, 2017.

- [17] C. Zhang et al., "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," *Proceedings of ACM FPGA*, pp. 161–170, 2015.
- [18] S. Han, H. Mao and W. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *International Conference on Learning Representations (ICLR)*, 2016.
- [19] Y. Chen et al., "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
- [20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.
- [22] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. ICML*, 2019.
- [23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
- [24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [25] G. Douzas and F. Bacao, "Effective Data Generation for Imbalanced Learning Using Conditional Generative Adversarial Networks," *Expert Systems with Applications*, vol. 91, pp. 464–471, 2018.
- [26] O. Sarraf and G. Tofighi, "DeepAD: Alzheimer's disease classification via deep convolutional neural networks using MRI and fMRI," *bioRxiv*, 2016.