FPGA-based Implementation of a Resource-Efficient UNET Model for Brain Tumour Segmentation

Modise Kagiso Neiso[1], Dr. Nicasio Maguu Muchuka[2], Dr. Shadrack Maina Mambo[3]
Department of Electrical & Electronics Engineering, PAUSTI, Juja, Kenya[1, 2, 3]
Department of Electrical & Control Engineering, Egerton University, Nakuru, Kenya[2]
Electrical Engineering Department, Walter Sisulu University, Ibika, South Africa[3]

***Abstract*—In this study an optimized UNET model is used for FPGA-based inference in the context of brain tumour segmentation using the BraTS dataset. The presented model features reduced depth and fewer filters, tailored to enhance efficiency on FPGA hardware. The implementation leverages High-Level Synthesis for Machine Learning (HLS4ML) to optimize and convert a Keras-based UNET model to Hardware Description Language (HDL) in the Kintex Ultrascale (xcku085-flva1517-3-e) FPGA. Resource strategy, First in First out (FIFO) depth optimization, and precision adjustment were employed to optimize FPGA resource utilization. Resource strategy is demonstrated to be effective, with resource utilization reaching a saturation point at a 1000-reuse factor. Following FIFO optimization, significant reductions are observed, including a 55 percent decrease in Block RAM (BRAM) usage, a 43 percent reduction in Flip-Flops (FF), and a 49 percent reduction in Look-Up Tables (LUT). In C/RTL co-simulation, the proposed FPGA-based UNET model achieves an Intersection over Union (IoU) score of 74 percent, demonstrating comparable segmentation accuracy to the original Keras model. These findings underscore the viability of the optimized UNET model for efficient brain tumour segmentation on FPGA platforms.**

Keywords*—UNET; field programmable gate array; high-level synthesis for machine learning; brain tumour segmentation

I. Introduction

Brain tumours are abnormal growths of cells in or around the brain and can be cancerous or non-cancerous [1]. Mutations in the DNA, radiation exposure and immune system problems are the causes of brain tumours which cause a noticeable mortality and low recovery rates. Early detection of brain tumours is crucial as it increases the possibility of successful treatment [2]. Magnetic resonance imaging (MRI) and computed tomography (CT) are imaging modalities used to diagnose brain tumours, with MRI producing more detailed brain scans [3]. Diagnosis of brain tumours from MRI requires skilled manpower in the medical field [4]. The expertise required for brain tumour diagnosis is insufficient and is susceptible to the human error factor, which has resulted in the implementation of deep learning (DL) to predict tumours to assist doctors [5].

Convolutional Neural Networks (CNN) are the deep DL model that perform better for feature extraction, but they require large datasets for efficient training which is hindrance for applications in medical imaging as large dataset are not easily accessible. Ronnerberger et al. proposes a UNET architecture that requires less image samples for successful model learning [6]. The UNET architecture has a drawback of consuming a lot of resources and computing inefficiency when applied in CPU and GPU [7]. The computational inefficiency of existing tumour prediction methods has become a critical concern in the medical imaging field. In response to this challenge, recent research has focused on areas to apply the UNET model for brain tumour detection in field programmable gate arrays (FPGA's), due to their inherent speed advantage over traditional processors. In a related study [8], the FPGA implementation of CNN was discussed, emphasizing on the necessity of a balanced design that considers resource utilization against performance.

In this work the challenge of balancing computational resource against performance in UNET for brain tumour prediction is addressed by proposing a modified UNET model and a comprehensive optimization of hardware resource utilization during FPGA inference. The modifications methods used in this work are tailored to enhance efficiency and efficacy of the UNET model for brain tumour prediction. The UNET model was optimized by reducing the size of original UNET model and FPGA hardware optimization strategies used entailed resource strategy, FIFO buffer depth optimization and precision.

The rest of this paper is organized as follows: Section II is literature review of relevant theory and related work; Section III describes materials and methods to carry out the work; Section IV is analysis and interpretation of results and discussion and conclusion is given in Section V and Section VI respectively.

II. Literature Review

Brain tumour segmentation is the technique of automatically detecting and labelling malignant brain tissues depending on tumour type [9]. Convolutional Neural Network has been successful in image segmentation applications in deep learning with applications in the medical field. The use of CNN in image segmentation entails deciding on the dataset to train the model; CNN architecture to use; loss function and the back-propagation weight adjustment algorithm.

A. Brain Tumour Dataset

Brain MRI scans is private patient's data which require confidential handling by health practitioners. However with the rise in the use of technology in the medical field, there is a need for the data to be made public and anonymous of patient's identity [10]. International research institutes have

made this data available for public use in developing medical imaging solutions such as The Cancer Genome Atlas (TCGA) dataset, BrainWeb dataset and MICCAI Brain Tumour Segmentation (BraTS 2020) Challenge dataset.

1) The Cancer Genome Atlas (TCGA): TCGA has different data such as TCGA Lower Grade Gliomas (LGG) and Glioblastoma Multiforme (GBM) which are basically grades of gliomas standardized by the World Health Organization. LGG are less aggressive while GBM is an exceptionally aggressive kind of brain glioma that develops from astrocytes or their progenitors [11]. In reserach [12], automatic and manual identification of GBM sub-compartments segmentation was performed and results showed that automated segmentation gave the highest area under the curve (AUC) as compared to manual segmentation. The TCGA dataset was created to identify a causal relation between genomic alterations and cancers [13]. TCGA data does not have specific pixel segmentation of tumour regions which will require additional radiologist expert knowledge to label the scans for training a machine learning model [14].

2) BrainWeb: Simulated Brain Database (SBD): The SBD currently has brain MRI data that has been simulated using two anatomical models: normal and multiple sclerosis (MS). T1-weighted (spin-lattice relaxation), T2-weighted (spin-spin relaxation), and PD-weighted (proton-density) sequences were employed to simulate entire three-dimensional data volumes for these models [15]. In study [16], the SBD database was utilized for improving the magnetic resonance imaging (MRI) segmentation using fuzzy C-means method and obtained experimental results that were more stable and accurate when compared to existing methods. SBD was created for computer aided image analysis by providing samples with ground truth. The SBD dataset is simulated for general use in computer-based analysis algorithms which diminishes its appeal when compared to other datasets sourced from actual patients.

3) MICCAI Brain Tumour Segmentation (BraTS 2020) challenge dataset: The BraTS 2020 dataset is made up of clinically obtained pre-operative multimodal MRI images of Glioblastoma (GBM/HGG) and lower grade glioma (LGG) that were collected from multiple institutions [17,18]. It contains a diagnosis that has been verified pathologically and has been divided into training and validation data. The dataset has expert manual segmentations that define the boundaries of the tumour regions, which include enhancing tumour, the peritumoral edema, and the necrotic and non-enhancing tumour core. All the scans of BraTS are in Neuroimaging Informatics Technology Initiative (NIfTIS) file format and they describe native (T1) and post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2-Fluid Attenuated Inversion Recovery (FLAIR) volumes, and were acquired using various clinical protocols and scanners from numerous institutions. The BraTS dataset has been used in several research with [19] introducing a residual mobile U-Net (RMU-Net) for MRI brain tumour segmentation. The RMU-Net archived dice coefficient scores for WT, TC, and ET on the BraTS 2020 dataset of 91.35%, 88.13%, and 83.26%, respectively, and 91.76%, 91.23%, and 83.19% on the BraTS 2019 dataset, and 90.80%, 86.75%, and 79.36% on the BraTS 2018 dataset.

The BraTS dataset is more versatile than the TCGA as it has manual segmentation that has been done by experts, which is essential for the training of a prediction model. In comparison to the SBD, BraTS dataset is more competitive as it has been sourced from real MRI scans that represents the real life cases as compared to simulated SBD. BraTS dataset is also specialized for segmentation with UNET as it has ground truth that has multiclass labels [20].

B. U-NET

U-NET is a convolutional neural network that was proposed by [6] for biomedical image segmentation applications. It was optimized to archive accurate prediction with few training images, as they are normally few sample images in the medical field. The original model constitutes of a contraction path which records context and expansion path which enables accurate localization. The UNET model is computationally demanding which translates into high resource utilization on hardware. There have been proposed methods to reduce the high resource demand by the UNET.

In study [21] a reduced UNET model architecture for classification of weeds and crops using segmentation was proposed. Reduction of the model was archived by reducing the number of filters per convolution layer. The proposed model in [21] has parameters that are 27% smaller while maintaining accuracy of 95% and an error rate that is 7% lower than the original UNET model. The reduction is however done on the number of filter per layer, and maintains the UNET architecture in terms of layers. In [22], the reduced depth UNET architecture with three down-sampling and two up-sampling sections is proposed, replacing the five down-sampling and four up-sampling sections of the original UNET architecture. Results obtained showed that the approach produced more accurate results [22]. There is gap in the existing work to combine reduction in number of filters and model depth as proposed by [21, 22].

1) Evaluation criteria: In order to evaluate segmentation there are metrics in image processing that can be used for quality assurance. The output image pixels are compared against those of the ground truth to establish the extent of accuracy. Intersection over union (IoU) and dice similarity coefficient (DSC) are the most common used metrics in medical imaging.

a) Intersection over union (Iou): Intersection over union is an evaluation metric that measures the intersection between the predicted mask and the actual mask, also known as the Jaccard index [23]. IoU calculation incorporates two indicators of false positive (FP) and true positive (TP) results. A true positive occurs when the model accurately predicts that a pixel is a component of an object when in fact it is. If the model forecasts a pixel as belonging to an item when in fact it belongs to the background, this is known as a false positive. The intersection of the anticipated segmentation mask and the ground truth mask, when compared to the union of the two

masks, is known as the IoU. IoU is particularly useful in multiclass segmentation where there is an imbalance in classes.

$$IoU = TP / (TP + FP + FN) \quad (1)$$

where, TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

b) Dice similarity coefficient: The Dice similarity coefficient also termed as Srensen-Dice index or simply the Dice coefficient is a statistical instrument used to determine the similarity of two sets of data [24, 25]. The dice similarity coefficient is both a spatial overlap indicator and a tool for validating reproducibility. The proportion of specific agreement was another name for it. A DSC value ranges from 0 to 1, with 0 indicating no spatial overlap and 1 representing total overlap. DSC examines the agreement between a predicted segmentation and its ground truth at a pixel level.

The equation for the DSC metric is:

$$DSC=2 * |X \cap Y| / (|X| + |Y|) \quad (2)$$

- Where X and Y are two sets.
- A set with vertical bars on either side denotes the set's cardinality, or the number of elements in that set, e.g. |X| denotes the number of elements in set X.
- $\cap$ is used to express the intersection of two sets and refers to the items that are shared by both.

2) Model Training Optimization Algorithm

Cost Function

The error between real y and predicted y at its present position is measured by the cost (or loss) function [26]. A loss function can be used to increase the effectiveness of the machine learning model by giving it feedback in order to change the parameters reducing the error, and ultimately locating the local or global minimum. Until the cost function is near to or equal to zero, it iterates repeatedly, moving in the direction of steepest descent. Learning in the model stops at the steepest descent. A cost function determines the average error across the whole training set, whereas a loss function just considers the error of one training sample [27].

Gradient Descent (Gd)

Optimizers update the model in response to the output of the loss function, consequently reducing the loss function. To locate a local minimum or maximum of a given function, gradient descent (GD), an iterative first-order optimization technique, is utilized. This method reduces the cost function and is mostly used as an entry level optimization in machine learning application [28]. Gradient descent however uses a constant learning rate which may need to be tuned manually to reach optimal performance. A higher learning results in faster training which require fewer epochs at the cost of overshooting the minimum. On the contrary a smaller learning rate will result in slower learning which requires more epochs for convergence [29]. The most common optimizers derived from gradient descent are Adaptive Gradient (Adagrad), Adaptive Delta (AdaDelta), Stochastic Gradient Descent (SGD), Adaptive Momentum (Adam), Cyclic Learning Rate (CLR), Adaptive Max Pooling (Adamax), Root Mean Square Propagation (RMS Prop), Nesterov Adaptive Momentum (Nadam), and Nesterov accelerated gradient (NAG) for CNN [30].

Adaptive Moment Estimation Optimizer (Adam Optimizer)

When training neural networks and machine learning models, the gradient descent approach is typically employed for optimization [27].

The Adam optimizer was made for deep neural network training optimization. It can be described as a combination of momentum-based stochastic gradient descent and RMSprop. Adam optimizer delivers computational performance, lower memory usage, and invariant to diagonal rescaling of gradients for applications with huge amounts of data or parameters [31]. Adam optimizer also computes adaptive learning rate, which entails tuning the learning rate during back-propagation, a property which gives it a competitive edge over other gradient descent optimizers. In study [30], ten common GD based optimizer algorithms were compared and analysed, and results obtained showed that Adam optimizer was more competitive with an accuracy of 99.2%.

C. Model Conversion to Hardware Description Language (HDL)

Workflow of implementing neural network in FPGA entails building of model architecture by deciding on the relevant layers and the training of the model using frameworks such as Tensorflow, Keras or Pytorch in high level language such as python and then converts the trained model to hardware description language (HDL). FPGA vendors have high level synthesis tools which synthesize from C++ to HDL. Converting a model in python trained to C++ can be a daunting task. However there are automation tools that bridge between trained models and C++ representations, such as LegUp, DNN Weaver, FINN and HLS4ML among others.

A. Conversion Tools

1) LegUp: LegUp is a high-level synthesis tool that uses C programming to get the system's behavioural description and creates an RTL netlist in Verilog HDL [32]. LegUp is compatible C/C++ language program and gives an output of Verilog HDL. The four major steps in LegUp HLS process are allocation, scheduling, binding, and RTL generation, which run consecutively [33]. LegUp supports Microchip PolarFire FPGA's.

2) FINN: FINN is an experimental framework developed by Xilinix Research Labs that focuses on the use of deep neural networks on FPGAs. It is designed for Xilinx FPGAS for converting quantized neural networks (QNNs) to HDL. FINN supports Brevitas which is a Pytorch package for neural network quantization that supports post training quantization (PTQ) and quantization aware training (QAT) [34].

3) DNNWeaver: DNNWeaver is an alternate converter for trained model to HDL implementation in FPGA, made at the

Alternative Computing Technologies (ACT) Laboratory, University of California [35]. A synthesizable FPGA accelerator with high efficiency and performance was achieved in [35]. The tool however only supports Caffe model which is a limitation for Tensorflow and Pytorch frameworks applications.

4) HLS4ML: HLS4ML is a tool for machine learning model implementation in FPGA covering both Vivado, Vitis and Quartus HLS backend and C++ inference templates [36]. The tool was developed for the CERN Hadron Collider (LHC), for fast capturing of results from detectors in the LHC. HLS4ML features Keras, Pytorch and ONNX frameworks, models C++ equivalent representations conversion, which can then be transformed to HDL by the backend HLS. Fully connected NN (multilayer perceptron, MLP), Convolutional NN, Recurrent NN (LSTM) and Graph NN (GarNet) are neural network architectures supported by HLS4ML. The support for different backend, frameworks and neural network architectures makes HLS4ML a more suitable tool for FPGA neural network inference.

Neural network implementation consumes a lot of FPGA resources that would be impossible to implement without optimization, which has led research work that incorporates FPGA hardware optimization when using HLS4ML. In study [37], inference of jet substructure model for particle physics in FPGA, archived Flip-flop and BRAM utilization below 4% of budget by employing bit width adjustment. FIFO buffer depth optimization was applied in [38] by recording buffer occupancy during the RTL simulation and then re-running the synthesis with updated FIFO buffer depth reduced resource utilization BRAM by 81%; LUT by 35% and FF by 37%.

III. Method and Materials

A. 2D UNET Architecture

The original UNET has five blocks in the contraction path that entail convolution, ReLU and max-pooling operations. Fig. 1 shows four blocks that perform up-sampling, convolution and ReLU in the expansion path, that are connected to their corresponding layers in contracting path. In this work the UNET architecture was reduced to reduce the resource requirement of the model [39].

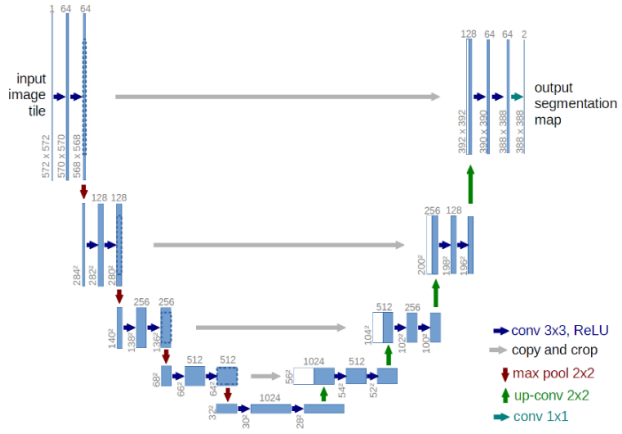


Fig. 1. UNET architecture [39].

B. Reduced UNET Model

The depth of the UNET architecture was reduced and the numbers of filters per layer were reduced. Filter numbers were reduced by 93.8% per layer and only one block of filter was used instead of two per layer. In the contraction path the feature extraction blocks were reduced from five to two consisting of 3*3 convolution kernels, 2*2 max-pooling. For the expansion path feature extraction blocks were reduced to two from four consisting of 2*2 up-sampling and concatenate path from the contracting layer as depicted in Fig. 2. The resulting model reduced model total parameters by 99%. Decreasing the model however comes with a cost of accuracy as the features that the model can capture are reduced. Tensorflow Keras framework was used to train the model.

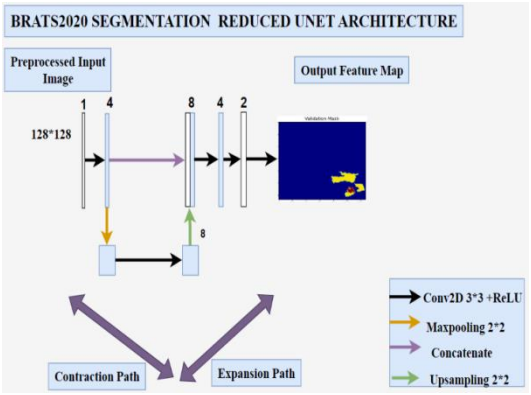


Fig. 2. Reduced UNET architecture.

C. Data Pre-Processing

Data pre-processing entails formatting dataset images to match the input size and features of the UNET model architecture. The BraTS 2020 dataset was used in this work and was pre-processed by resizing, min-max scaling and slicing before being used for training and testing of the model. In order to capture more features T1CE, T2 and FLAIR modalities were used for training.

Resizing

BraTS 2020 dataset consists of 3D MRI images with modalities and segmentation masks marked by experts. The 3D-MRI scans were resized from 240*240*155 to 128*128*128 dimension to reduce the unnecessary background data and for uniformity in the data. Fig 3 shows the resized image with dimensions 128*128 and background cropped out.

1) Min-max scaler: The images were scaled using the min-max scaler for a mean of 0 and max value of 1. Scaling helps to standardize the data for efficient training of the model.

2) Slicing: Since the model was defined for 2D convolution kernels each image was sliced to from 3D to 2D slices for all the modalities and masks. The modalities for each sample were converted to numpy array and combined or stacked into one image which translate to input channels in the model architecture. Fig. 3 is a plot of a 2D slice from the 127 slices obtained per 3D image slicing.

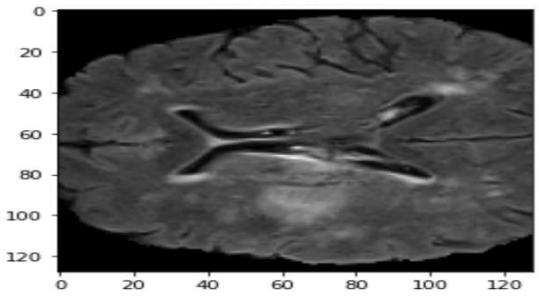


Fig. 3. Pre-processed BraTS 2020 2D slice.

3) Data Generator: A data generator was defined for loading the images and masks during training. The images and masks were loaded as an (X,Y) tuple. In the tuple method the X represents the data which was passed through the model for training and Y is the expected output which was used for calculating the loss against the model output at each iteration.

D. Adam Optimization

Adam optimizer was used during the training to update the weights and learning rate to reach lowest possible loss, through gradient descent method. In order to calculate the loss, dice score coefficient (DSC) was used to compute the total loss which was then back-propagated for updating weights and the learning rate. The optimizer weights were removed from the model after the training since they are not required for inference.

E. Tensorflow Keras Model Conversion to C++ and RTL

HLS4ML was used for converting the trained model to HDL. Adam optimizer Weights added to the model during training was removed before conversion. This reduced the memory footprint of the model as optimizer weights are not required for inference. A script for generating the C++ equivalence of the model was developed. The script describes the optimization category in terms of resource or latency, clock, input/output type (io_type) and backend HLS tool that was used. Fig. 4 illustrates the process flow from model training to FPGA implementation using HLS4ML.

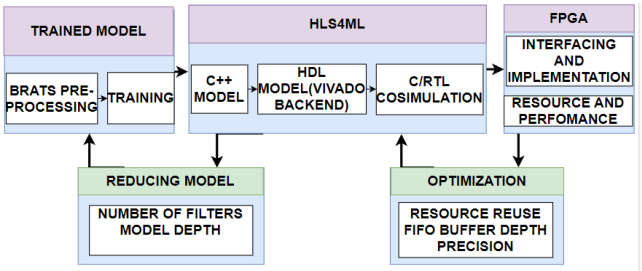


Fig. 4. HLS4ML conversion process.

Conversion to RTL

Table I illustrates the configuration used to convert the reduced model to HDL. The C++ generated representation of the model was obtained from the templates in hls4ml that define CONV2D, MAXPOOL and other layers defined in the model. Vivado HLS supports converting C++ model representation to HDL. In this work the C++ model was converted to Verilog, System-C and VHDL. The main files generated from the conversion are the HDL, data and constraints. HDL files describe various layers of the model; data files contain the weights and biases and the constraint files define the timing and layer interconnections.

TABLE I. REDUCED UNET CONVERSION TO HDL CONFIGURATION

Configuration	Parameter
Backend	Vivado HLS
FPGA	Kintex Ultrascale Part- xcku085-flva1517-3-e
Strategy	Resource (Reuse Factor)
FIFO Depth (Initial)	100_000
Precision	Fixed point arithmetic(ap<>)

F. HDL Optimization

In order to optimize the FPGA resource utilization the precision, resource strategy and FIFO buffer depth optimization techniques were used.

1) Precision: Floating point numbers due to their limitless precision in computation leads to an increased utilization of resources. Arbitrary fixed point type was used in this work which is defined by 'ap_fixed<a,b>', where 'a' is the total bit width and 'b' is the fractional part from the total size of 'b'. Fixed-point arithmetic is more efficient in reducing resource utilization when compared to floating-point arithmetic.

2) Resource strategy: In multilayer neural networks, each neuron in a layer (consisting of n neurons) produces an output computed by the weighted sum of the output from the previous layer. The weights associated with these connections are represented as a matrix W of size n*m, where m is the number of neurons in the previous layer. Each neuron has a bias that is independent and represented by vector b. The weighted sums and the biases are summed and added non linearity by activation function denoted by f, resulting in the final output of the neuron. This process of weights, biases and activation function is expressed as:

$$y=f(W*\text{previous layer output} + b) \quad (3)$$

The multiplications in neural networks are computed by multipliers in FPGA, which result in high resource utilization if each multiplication was to represent a physical multiplier. Resource strategy was used to optimize the design by reusing of multipliers as demonstrated in Fig. 5.

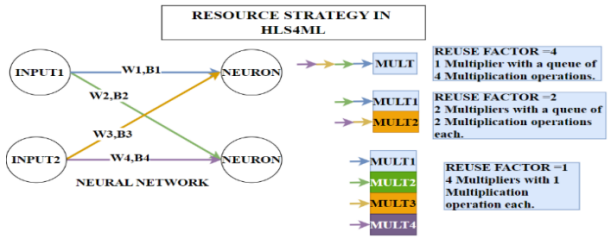


Fig. 5. Multiplier reuse reduces number of multipliers at the expense of parallel processing.

Multiplier Limit

Processing elements are a functional block that preforms specific operations such as multiplication and addition. In the FPGA, operations are efficiently conducted in parallel across multiple PE's, thereby improving computational efficiency. For resource strategy multiplier_limit variable was defined to represent the maximum number of multiplications that can be done in parallel for the available resources. The calculation is based on the number of input values (mult_n_in), the number of output values (mult_n_out) and the reuse factor. A higher reuse factor will mean a lower value of multiplier_limit which translates to actual multipliers used in the FPGA implementation, however the universal time of operation will increase as the multipliers will be shared by a number of operations.

$$\text{Mult\_n\_in} = \text{filt\_height} * \text{filt\_width} * \text{n\_chan} \quad (4)$$

$$\text{Mult\_n\_out} = \text{n\_filt} \quad (5)$$

$$\text{Multiplier\_limit} = \text{DIV\_ROUNDUP}(\text{mult\_n\_in} * \text{mult\_n\_out}, \text{reuse\_factor}) \quad (6)$$

3) First In First Out (FIFO) Buffer Depth Optimization: FIFO buffers store data in between layers. In the initial C++ conversion the size of the buffers is estimated, however the estimated size is above the utilization during simulation. This results in a BRAM and LUT usage that is higher than what the design requires. In FIFO depth optimization method the buffer size for each layer was set to 100_000, which is a value above what is required by the design. The design was then synthesized and during RTL co-simulation the buffer occupation was recorded and used to update the FIFO buffer sizes which translated to a 71% reduction in buffer size. Fig. 6 is an overview of the buffer method capturing of buffer utilization during simulation to the updating of the new buffer size.

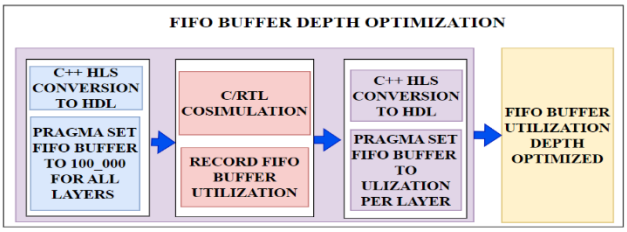


Fig. 6. FIFO buffer depth optimization overview.

G. C++ /Register Transfer Level (C/RTL) CO-SIMULATION

C/RTL co-simulation was used to verify the functional preservation of the HDL converted model. The test images were converted to a 1D array and saved as data files with pixel values saved as strings for compatibility with simulator. C/RTL co-simulation was done using test-bench and test data. The comparison of C-Simulation and RTL simulation passed validation and output for inference was a 1D array that was converted to 2D numpy array.

IV. RESULTS

A. Precision

The increase in configured precision for the reduced UNET model during conversion to HDL was directly proportional to the increase in resource utilization as shown in Table II.

TABLE II. INCREASING AP PRECISION INCREASES RESOURCE UTILIZATION FOR THE REDUCED UNET MODEL CONVERSION IN HLS4ML

PRECISION	BRAM_18K	DSP48E	FF	LUT
<16,6>	2118	5	67033	149764
<32,6>	3978	13	97408	188001
<64,6>	7845	65	155388	219201

B. Resource Strategy

Multiplier reuse reduced the resources used as multipliers were shared among operations at the expense of performance. Maximum possible reuse factor is 1152 hence beyond 1500 reuse factor the performance and resource utilization is not affected, as shown by Fig. 7 graph of resource utilization. This is because there is an upper-limit to optimization of resource utilization against performance.

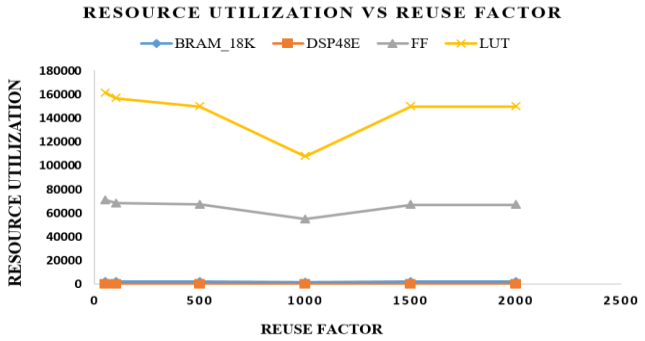


Fig. 7. Increase in reuse factor is directly proportional to resource utilization and reach roofline at 1000 reuse factor.

C. First In First Out (FIFO) Buffer Depth Optimization

New FIFO buffer depth values were inserted in the C++ firmware as pragma directives to guide synthesis. Resource utilization after FIFO optimization showed 54.8% reduction in BRAM's, 29% reduction in FF's and 44% reduction in LUT's. The new FIFO buffer depth was the occupancy increased by value of 1. Table III shows the reduction in resource utilization pre and post buffer size reduction. Table IV illustrates how the buffer size was reduced using pragma directives during synthesis.

TABLE III. FIFO DEPTH OPTIMIZATION REDUCTION IN RESOURCES

Optimization	BRAM_18K	DSP48E	FF	LUT	LATENCY(ms)
No	1634	8	52679	120583	9.8
Yes	739	5	29811	62059	9.8
% Change	-54.8	-37.5	-43.4	-48.5	0

TABLE IV. FIFO BUFFER DEPTH SIMULATION OCCUPANCY RESULTS AND OPTIMIZATION PRAGMAS TO REDUCE BUFFER DEPTH

Layer	Occupancy	New FIFO BUFFER DEPTH	PRAGMA DIRECTIVE
layer20_out_V_data_0_V_U	33348	33349	#pragma HLS STREAM variable=layer20_out depth=33349
layer21_out_V_data_0_V_U	16608	16609	#pragma HLS STREAM variable=layer21_out depth=16609
layer19_cpy2_V_data_0_V_U	819	820	#pragma HLS STREAM variable=layer19_cpy2 depth=820
layer22_out_V_data_0_V_U	134	135	#pragma HLS STREAM variable=layer22_out depth=135
layer23_out_V_data_0_V_U	199	200	#pragma HLS STREAM variable=layer23_out depth=200
layer12_out_V_data_0_V_U	3	4	#pragma HLS STREAM variable=layer12_out depth=4
layer24_out_V_data_0_V_U	37372	37373	#pragma HLS STREAM variable=layer24_out depth=37373
layer25_out_V_data_0_V_U	261	262	#pragma HLS STREAM variable=layer25_out depth=262
layer26_out_V_data_0_V_U	1	2	#pragma HLS STREAM variable=layer26_out depth=2

TABLE V. SHOWING THE REDUCED MODEL COMPARISON WITH EXISTING WORK

Ref	Model	Dataset	Parameters	IoU Score	Accuracy	Execution Time(ms)
UNET MODEL	UNET	BraTS 2020	1.9 M	0.60	0.97	132
[40]Ercüment GÜVENÇ (2023)	FLAIR MR IMAGES WITH U-NET	BraTS 2018	-	0.59	0.99	-
[41]Jwaid (2021)	3D U-Net CNN	BraTS 2017	-	0.69	0.99	-
Proposed	Reduced UNET	BraTS 2020	864	0.74	0.95	38

D. Reduced Model Result Comparison with Existing Work

A full UNET model with original architecture was developed to compare with the reduced UNET proposed in this work. Model parameters were reduced from 1.9 million in original 2D-UNET model to 864 in proposed reduced UNET model.

E. Intellectual Property (IP) Core

Fig. 8 is the graphical representation of the intellectual property core that was generated from the RTL design. The generate IP core has 3-inputs which correspond to the three input channels in the model architecture and there are four output channels which are aligned to the number of classes for type of tumour and background. IP cores introduce modular design which can then be interfaced with input and output IP cores such as Ethernet.

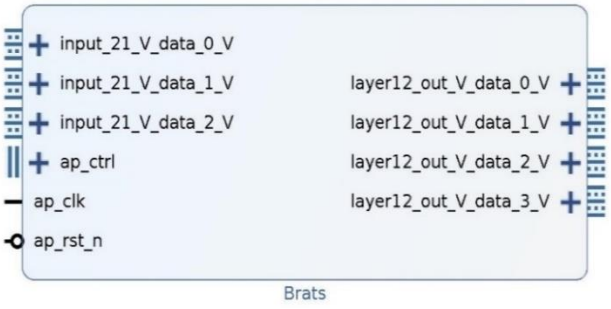
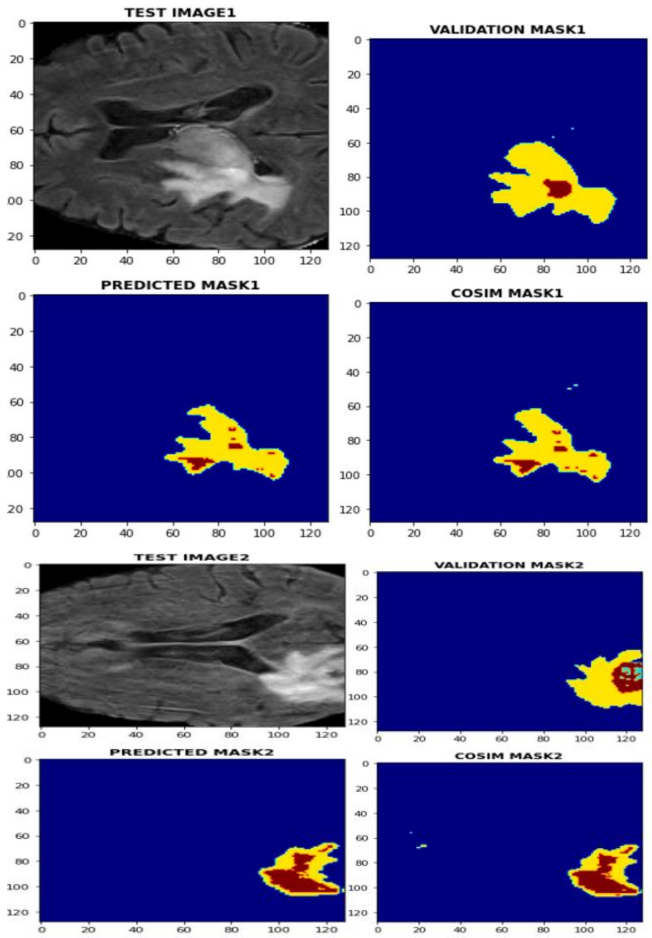


Fig. 8. BraTS 2D UNET tumour segmentation IP core.

F. C/RTL Co-simulation Results

The co-simulation results presented in Fig. 9, visually demonstrate the similarity between prediction output masks of the python model and RTL simulation. Quantitative analysis, computing the IoU score output a comparable value of 74% between the original python model and the RTL simulation, validating the fidelity of the FPGA-based implementation.



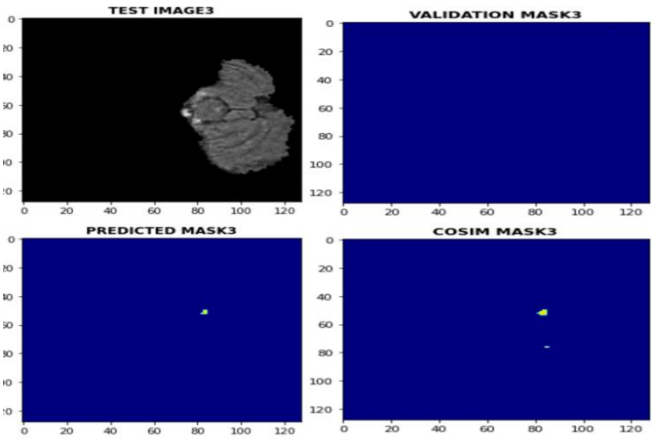


Fig. 9. Reduced UNET python model and C/RTL co-simulation brain tumour prediction masks.

V. Discussion

Table V shows that the proposed model has a higher IoU score over the existing work in [40, 41]. In study [40], only FLAIR images modality which limited the amount of features that can be learned by the model as compared to T1CE, T2 and FLAIR modalities which were used in the proposed model to learn more features from the dataset. The proposed model however has limitations in terms of accurately predicting tumour class. Future work can be focused on improving multiclass prediction of the proposed model.

VI. Conclusion

In this work a reduced UNET model was built and trained in Tensorflow Keras for brain tumour segmentation applications and archived an IoU score of 74%. The reduced model significantly reduced the model parameters by 99% which translates into reduced computational requirements. Converting the reduced model into C++ and HDL equivalent representations using HLS4ML, FPGA resources were used economically while preserving the original model segmentation output mask accuracy. Resource strategy, FIFO buffer depth and precision methods significantly reduced FPGA resource usage. The reduced model segmentation performs well in predicting tumour region, however the tumour classes are still poorly predicted. Further work on choosing the right loss function suitable for unbalanced multi-class segmentation for the reduced model can be done to improve inference accuracy while reducing FPGA resource utilization

Acknowledgment

Authors are thankful to PAUSTI University for the funding of this research work and the department of Electrical and Electronic Engineering for the unwavering support in completion of this work.

References

[1] Clinic, Mayo, "Brain tumor," Mayo Clinic, 2023. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084. [Accessed 20 November 2023].

[2] S. R. H. K. S. R. N. K. Soheila Saeedi, "MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques," BMC Medical Informatics and Decision Making, vol. 23, no. 16, 2023.

[3] Cancer.Net, "Brain Tumor: Diagnosis," Cancer.Net, March 2023. [Online]. Available: https://www.cancer.net/cancer-types/brain-tumor/diagnosis. [Accessed 21 November 2023].

[4] A. Abdullah Asiri et al, "Brain Tumor Detection and Classification Using Fine-Tuned CNN with ResNet50 and U-Net Model: A Study on TCGA-LGG and TCIA Dataset for MRI Applications," Artificial Intelligence Applications in Medical Imaging, vol. 13, no. 7, 2023.

[5] A. K. T. M. P. S. A. B. Mukul Aggarwal, "An early detection and segmentation of Brain Tumor using Deep Neural Network," BMC Medical Informatics and Decision Making, vol. 23, no. 78, 2023.

[6] P. F. T. B. Olaf Ronneberger, "U-Net: Convolutional Networks for Biomedical," in International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015.

[7] Xuan Cheng et al, "Efficient hardware design of a deep U-net model for pixel-level ECG classification in healthcare device," Microelectronics Journal, vol. 126, 2022.

[8] Z. L. Chenghao Wang, "A Review of the Optimal Design of Neural Networks Based on FPGA," MDPI, p. 44, 2022.

[9] M. B. B. K. A. K. Dinthisrang Daimary, "Brain Tumor Segmentation from MRI Images using Hybrid Convolutional Neural Networks," in International Conference on Computational Intelligence and Data Science (ICCIDS 2019), 2020.

[10] E. B. V. D. C. Tonya White, "Data sharing and privacy issues in neuroimaging research: Opportunities, obstacles, challenges, and monsters under the bed," Hum Brain Mapping, vol. 43, no. 1, pp. 278-291, 2020.

[11] K. T. K. A. Keiko Sato, "Five Genes Associated With Survival in Patients With Lower-grade Gliomas Were Identified by Information-theoretical Analysis," Anticancer, pp. 2777-2785, 2020.

[12] Emmanuel Rios Velazquez et al, "Fully automatic GBM segmentation in the TCGA-GBM dataset: Prognosis and correlation with VASARI features," Scientific Reports, vol. 5, no. 16822, 2015.

[13] P. C. W. Katarzyna Tomczak, "The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge," Contemporary Oncology, vol. 19, no. 1A/2015, pp. A68-A77, 2015.

[14] Spyridon Bakas et al, "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features," Scientific Data, vol. 4, no. 1, 2017.

[15] BrainWeb, "BrainWeb: Simulated Brain Database," McGil University, [Online]. Available: https://brainweb.bic.mni.mcgill.ca/. [Accessed 22 May 2023].

[16] A. Z. Elnomery, "An Adaptive Fuzzy C-Means Algorithm for Improving MRI Segmentation," Open Journal of Medical Imaging, vol. 3, pp. 125-135, 2013.

[17] S. Bakas, "Multimodal Brain Tumor Segmentation Challenge 2020: Data," Perelman School of Medicine, University of Pennsylvania, 2020. [Online]. Available: https://www.med.upenn.edu/cbica/brats2020/data.html. [Accessed 22 May 2023].

[18] Bjoern H. Menze et al, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," IEEE Transactions on Medical Imaging, vol. 34, no. 10, pp. 1993-2024, 2015.

[19] Muhammad Usman Saeed et al, "RMU-Net: A Novel Residual Mobile U-Net Model for Brain Tumor Segmentation from MR Images," Electronics, vol. 10, no. 16, 2021.

[20] Spyridon Bakas et al, "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge," arXiv preprint arXiv:1811.02629, 2019.

[21] S. U. A. V. J. R. Arumuga Arun, "Reduced U-Net Architecture for Classifying Crop and Weed using Pixel-wise Segemntation," in 2020 IEEE International Conference for Innovation in Technology (INOCON) , Bengaluru, India, 2020.

[22] M. S. R. Rupal R. Agravat, "Prediction of Overall Survival of Brain Tumor Patients," in TENCON , Kochi India, 2019.

[23] O. Sheremet, "Intersection over union (IoU) calculation for evaluating an image segmentation model," Towards Data Science, 25 July 2020. [Online]. Available: https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation-model-8b22e2e84686. [Accessed 02 July 2023].

[24] B. D. J, "Dice similarity coefficient," Radiopaedia, 02 08 2021. [Online]. Available: https://radiopaedia.org/articles/dice-similarity-coefficient. [Accessed 10 02 2023].

[25] D. L. C. I. P. F. K. Anthony D. Yao, "Deep Learning in Neuroradiology: A Systematic Review of Current Algorithms and Approaches for the New Wave of Imaging Technology," Radiology: Artificial Intelligence, vol. 2, no. 2, pp. 1-6, 2020.

[26] D. S. S. L. X. L. Yingjie Tian, "Recent advances on loss functions in deep learning for computer vision," Neurocomputing, vol. 497, pp. 129-158, 2022.

[27] IBM, "What is gradient descent?," IBM, 2023. [Online]. Available: https://www.ibm.com/topics/gradient-descent. [Accessed 01 July 2023].

[28] R. Kwiatkowski, "Gradient Descent Algorithm — a deep dive," Towards Data Science, 22 May 2021. [Online]. Available: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21. [Accessed 01 July 2023].

[29] J. Brownlee, "Understand the Impact of Learning Rate on Neural Network Performance," MACHINE LEARNING MASTERY, 25 January 2019. [Online]. Available: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/. [Accessed 01 July 2023].

[30] Muhammad Yaqub et al, "State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images," Brain Sciences, vol. 10, no. 7, p. 427, 2020.

[31] J. L. B. Dlederlk P.Kingma, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," in 3rd International Conference for Learning Representations, San Diego, 2015.

[32] S. R. a. M. Joseph, "Open source HLS tools: A stepping stone for modern electronic CAD," in 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, India, 2016.

[33] Inc, Microchip Technology, "2User Guide: 2.1 Introduction to High-Level Synthesis," Microchip Technology Inc, 2021. [Online]. Available: https://download-soc.microsemi.com/FPGA/HLS-EAP/docs/legup-2021.1-docs/userguide.html. [Accessed 30 May 2023].

[34] pypi, "Quantization-aware training in PyTorch," pypi, 28 April 2023. [Online]. Available: https://pypi.org/project/brevitas/#:~:text=Brevitas%20is%20a%20PyTorch%20library,not%20an%20official%20Xilinx%20product.. [Accessed 27 May 2023].

[35] Hardick Sharma et al, "From High-Level Deep Neural Models to FPGAs," in The 49th Annual IEEE/ACM International Symposium on Microarchitecture, Taipei, Taiwan, 2016.

[36] Team, FastML, "fastmachinelearning/hls4ml," FastML Team, 2023. [Online]. Available: https://github.com/fastmachinelearning/hls4ml. [Accessed 27 October 2023].

[37] Javier Duartea et al, "Fast inference of deep neural networks in FPGAs for particle physics," Journal of Instrumentation, vol. 13, no. 07, p. P07027, 2018.

[38] Nicolò Ghielmetti et al, "REAL-TIME SEMANTIC SEGMENTATION ON FPGAS FOR AUTONOMOUS VEHICLES WITH HLS4ML," Machine Learning: Science and Technology, vol. 3, no. 4, 2022.

[39] J. Zhang, "UNET- Line by Line Explanation," Towards Data Science, 4 October 2019. [Online]. Available: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5. [Accessed 05 November 2023].

[40] M. E. G. Ç. Ercüment GÜVENÇ, "BRAIN TUMOR SEGMENTATION ON FLAIR MR IMAGES WITH U-NET," Mugla Journal of Science and Technology, vol. 9, no. 1, pp. 34-41, 2023.

[41] W. M. Jwaid et al, "Development of Brain Tumor Segmentation of," Eastern-European Journal of, vol. 4, no. 9, pp. 23-31, 2021.