brain sciences

MDPI

Article

State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images

Muhammad Yaqub [1], Jinchao Feng [1,*], M. Sultan Zia [2], Kaleem Arshid [1], Kebin Jia [1,3], Zaka Ur Rehman [2] and Atif Mehmood [4]

1 Faculty of Information Technology, Beijing University of Technology, Beijing 100000, China; myaqubciitswl@gmail.com (M.Y.); a_kaleem@outlook.com (K.A.); kebinj@bjut.edu.cn (K.J.)
2 Department of Computer Science and IT, The University of Lahore, Gujrat Campus, Main GT Road, Adjacent Chenab Bridge Gujrat, Gujranwala, Punjab 52250, Pakistan; sultan.zia@cs.uol.edu.pk (M.S.Z.); rao.zaka@yahoo.com (Z.U.R.)
3 Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing 100000, China
4 School of Artificial Intelligence, Xidian University, No. 2 South Taibai Road, Xi'an 710071, China; atifedu151@yahoo.com
* Correspondence: fengjc@bjut.edu.cn; Tel.: +86-17810604461

Received: 7 June 2020; Accepted: 1 July 2020; Published: 3 July 2020

check for updates

Abstract: Brain tumors have become a leading cause of death around the globe. The main reason for this epidemic is the difficulty conducting a timely diagnosis of the tumor. Fortunately, magnetic resonance images (MRI) are utilized to diagnose tumors in most cases. The performance of a Convolutional Neural Network (CNN) depends on many factors (i.e., weight initialization, optimization, batches and epochs, learning rate, activation function, loss function, and network topology), data quality, and specific combinations of these model attributes. When we deal with a segmentation or classification problem, utilizing a single optimizer is considered weak testing or validity unless the decision of the selection of an optimizer is backed up by a strong argument. Therefore, optimizer selection processes are considered important to validate the usage of a single optimizer in order to attain these decision problems. In this paper, we provides a comprehensive comparative analysis of popular optimizers of CNN to benchmark the segmentation for improvement. In detail, we perform a comparative analysis of 10 different state-of-the-art gradient descent-based optimizers, namely 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. The experiments were performed on the BraTS2015 data set. The Adam optimizer had the best accuracy of 99.2% in enhancing the CNN ability in classification and segmentation.

Keywords: brain tumor; optimizer; deep learning; convolutional neural network; gradient descent; segmentation; Adam

1. Introduction

A disease is defined as a disorder of function in a living being. If we drill down the definition, it can be defined as a disorder of structure or function in the division of cells in a living organism. If a disorder of unnatural mass is developed in the cerebrum of a brain, we call it a brain tumor. Brain tumors are of different types and can be dangerous at times, and glioma is the most common type of nonpermanent or treatable tumor. Glioma can be classified into two types, namely High-Grade

Gliomas (HGG) and Low-Grade Gliomas (LGG). LGG is a slow-spreading tumor, while HGG is a rapidly growing tumor, which explains why HGG is a fatal disease. People who are diagnosed with HGG and who are aged between 20–44 years have a survival rate of 19% with treatment after 14 months of diagnosis, based on a recent survey of the central nervous system (CNS) [1] on a Canadian population from 2009–2013. Figure 1 shows the distribution of survival rates between different types of brain tumors.

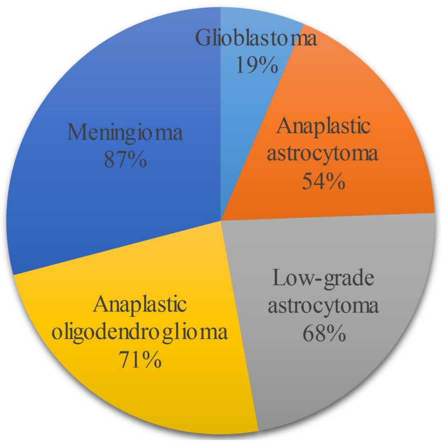


Figure 1. The 5 tumor types with their survival rates for patients aged between 20–44.

Though there are many medical imaging modalities available to differentiate the characteristics of brain tumors, magnetic resonance images (MRIs) are the most commonly used medical imaging modalities due to its advantage of visual analysis and its flexibility in the domain of computer-aided analysis of medical images. It plays a vital role at many stages of the clinical work flow for population screening; the role of MRI modalities will ramp up in the coming future due to developments in the domain of analysis methods along the lines of cost effectiveness and accuracy. With the help of MRIs, tumors can be differentiated into different grades of gliomas. Among the latest high-tech technologies, MRIs can be considered one of the most advanced techniques used to characterize brain tumors for diagnosis and evaluation. Accurate identification of tumor distance can be considered a critical phase of various neuroimaging studies [2]. The types of MRI modalities are clearly outlined in Figure 2. They [3] focused their experimental analysis on the fully annotated brain tumor segmentation (BraTS) challenge 2013 data set using the well-defined training and testing splits, thereby allowing us to compare directly and quantitatively a wide variety of other methods. Deep learning (DL) and Convolutional Neural Networks (CNN) stood at the center of all these developments in brain MRI image analysis and computer interventions and proved their adoption to be a successful execution to drive for continuous improvements.

Convolutional networks were inspired by biological processes [4,5] in that the connectivity pattern between neurons resembled the organization of the animal visual cortex. Initially, Artifical Neural Network (ANN) was used to study the data from digital images, but in order to do so, the domain experts or the researches have to manually decide and extract features from the digital images and to feed it to the ANN. CNN came to the rescue in eliminating the cumbersome manual work of deciding the features. CNN is one of the most remarkable forms of ANN that is inspired by natural visual recognition phenomenon [6]. There are innumerable applications of CNN in the field of image classification and pattern recognition [7]. The architecture of CNN was introduced in the late 80 s [8]. After the introduction of CNN, it was improved by LeCun in the late 90s [9], but the introduction of the ConvNet architecture [10] in the 21st century has taken CNNs to a different level, with an error rate of 15.3% as compared to conventional computer vision (CV) techniques [11].

CNN has made huge impacts in the medical imaging domain [12] and many other fields such as computer vision, digital image processing, and artificial intelligence. Due to its multilayered architecture, CNN is the most popular technique employed for image analysis although there are many deep learning algorithms introduced over the past decades [13–15]. Similar to ANN, CNN also uses an adaptive approach to learn spatial hierarchies of features through back propagation, but unlike ANN, CNN does not have fully connected neurons for all the layers and it has only the last layer as fully connected layer. CNN consists of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers [16]. The convolution layer is responsible for feature extraction, which makes it special compared to ANN; this layer is typically responsible for convolution operation and activation function.

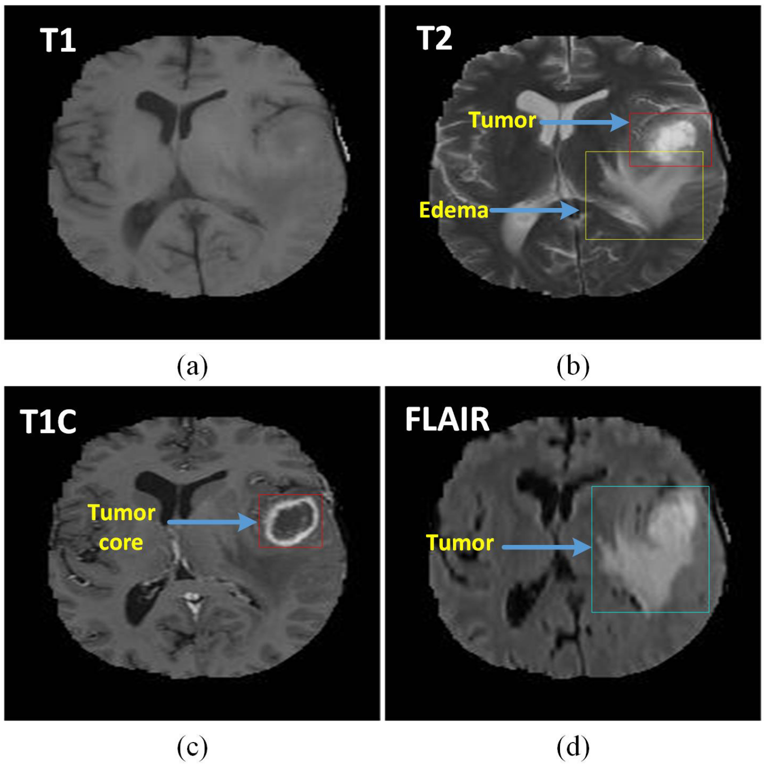


Figure 2. The different tumor types with different shapes in four magnetic resonance images (MRI) sequences: (**a**) T1 MRI sequence, (**b**) T2 MRI sequence with tumor type edema, (**c**) T1C MRI sequence with core tumor, and (**d**) Search Results Web results Fluid attenuation inversion recovery (FLAIR) sequence showing the ground truth of a tumor.

2. Literature Review

Over the last few years, there has been significant effort from fellow researchers in the development of CNN; some articles focused on studying the characteristics of machine learning algorithms [17], which tried to explain the multilayered network methodology along the lines of back propagation and updating weights. In Table 1 detailed literature review presented with methodology, result and future directions. The working mechanism of CNN and other neural networks along with their usage in the machine learning and deep learning algorithms is also discussed briefly. Their study opened the door towards future research directives leading to image segmentation. It has made it clear that the acquisition of an edge is a necessary part of a good segmentation [18]. Segmentation usually means segmenting the image into a required partition; edge detection is employed by many researchers to find the best regions in the image. Canny and his coworkers were introduced to edge-based segmentation, which utilized the optimal smoothing filter to maintain the edges while performing image segmentation. In a pilot study conducted by [19], in an effort to monitor the performance of a handwriting recognition algorithm, segmentation was employed to handle the spatially sparse components and these segmented parts were utilized as an input image in the input layer of the CNN algorithm. Reference [20] proposed a deep siamese convolution neural network-based approach for classification of Alzheimer's disease stages and produced promising results in term of classification on brain images, which is able to identify normal control and disease patients. Reference [15] introduced a thresholding technique, which became a popular technique used for segmentation over the years. They suggested the aforementioned limitation by designing a framework for optimizing Bayesian design and the stability of integrated multidisciplinary systems. Their proposed framework was built on the Gibbs estimation process and applies the gradient information policy (KG) for sequential ordering to achieve the largest one-time increase expected in the design process [21]. In this technique, the gray scale and white scale values are filtered or manipulated by selecting a threshold value. Otsus method, k-means clustering, and maximum entropy method are among the most famous threshold techniques. The threshold value acts as a mid-value, and input values above or below the nominated threshold value are finally displayed. Thresholding methods are nowadays used in segmenting computed tomography (CT), Magnetic Resonance (MR), and Ultrasonic and Positron Emission Tomography (PET) images. In a recent study, a blend of RNN and CNN was introduced with the name Convolutional Recurrent Neural Network (CRNN) [14]. The specialty of CRNN is the recurrent layer, which adds the extracted features, and the feed forward layer, which provides the output. An improved output is obtained due to the improved back propagation. In [22], a 3 CNN layer-based deep learning method was proposed to classify different brain tumour types and grades. Deep-CNN-based transfer learning and fine-tuning was used to segment brain tumors [23]. A recent study clearly outlined the components of CNN (layers, ReLU, dropout, response, and pooling) and its working mechanism [24]; this study is a comparison of scale-invariant feature transform (SIFT) and sparse coding for ImageNet LSVRC (Large Scale Visual Recognition Challenge) held in 2010. A data set of over one million images was utilized with over 1000 categories for training and 50,000 images for testing their system, and they adopted a methodology which enhanced the error rate approximately by 2%. Furthermore, when researchers were struggling with discriminative classifiers by separating the hyper planes, supervised learning-based Support Vector Machine (SVM) came to the rescue. A recent study applied Least Square Support Vector Machine (LS-SVM) to separate the White Matter (WM) and Grey Matter (GM) regions [25] and used a brain atlas for their analysis by performing manual intervention. Later on, scientists started to introduce the multilayered combined multidimensional methodology [26]. The winner neuron was taken as an input after maxpooling the function and took part in further training and segmentation for deep learning. This 2D methodology somehow improved the algorithm performance.

Table 1. Literature review.

Sr. No.	Methodology	Results	Future Directions
1	Three layered feed forward ANNs and two real world problems are set as a benchmark to access the performance of Group Search Optimizer (GSO) [27].	GSOANN has a far better performance as compared to regular ANN.	—
2	A hybrid model of DSA and DL to help improve the relationship of computer science and bioinformatics [28].	Differential Search Algorithm (DSA) and DL can help produce more xylitol for sugar free gums.	Computational biologists and computer scientist can together produce a hybrid model using deep learning OA.
3	In auto-encoders like VVG-9 and CIFAR-10, they design some experiments to study the properties of RMSProp and Adam against Nesterov's Accelerated Gradient method [29].	On very high values of $\beta 1 = 0.99$ Adam outperforms lower training and test losses, whereas with $\beta 1 = 0.9$, NAG performs better.	Advance theory in getting more better results by getting $\beta 1$ close to 1.

Table 1. *Cont.*

Sr. No.	Methodology	Results	Future Directions
4	Different optimization algorithms are studied by side CNN architecture [30].	Among 7 optimizers, on the LeNet architecture, Adam provides the smallest MSE whereas SGD and Adagrad failed.	Can build analytical protable image devices
5	Constructed a few illustrative binary classification problems and examined empirical generalization capability of adptive methods agaisnt GD.	Solutions found by adaptive methods generalize worse than GSD.	Adaptive methods should be reconsidered.
6	Energy Index based Optimization Method (EIOM) that automatically adjusts the learning rate in backpropagation [31].	EIOM proves to be the best when compared with state-of-the-art optimzation methods.	—–
7	A non-asymptotic analysis of the convergence of two algorithms: SGD and simple averaging [32].	The analysis suggests that the learning rate is proportional to the inverse of the number of iterations.	Differential and non-differential stochastic
8	Adaptive learning rate and laplacian approach have been proposed for Deep Learning in MLP [33].	Improved classification accuracy	—–
9	Proposed a fundamental approach for anatomical, celluler stuctures, and tissue segmentation using CNN through image patches measuring 13 × 13 voxels [34].	On different data sets, comparing the six commonly used tools (i.e., ROBEX, HWA, BET, BEaST, BSE, and 3dSkullStrip), they achived the highest average specifity.	Can be performed on most advanced tools and used a real time data set to get better result.
10	Used a pretrained CNN model on augmented and orginal data for brain tumor classification [35]	They achieved 90.67 accuracy before and after data augmentation on the proposed methed and compared with most advanced methods	Used light weight CNN to entend their work for fine-grained classification differential stochastic.
11	A CapsNet for brain tumor classification and investigation of the overfitting problem based on CapNet [36].	On 10 epochs, they achieved 86.56% accuracy, with the comparative analysis with CNN learning rate proportional to the inverse of the number of iterations.	In the future, investigations on the effects of more layers on the classification accuracy will be performed.
12	A review on deep learning techniques in the field of medical images classification [37]	They discussed in detail the deep learning approaches and their suitability for medical images. The learning rate is proportional to the inverse of the number of iterations.	Further research is required to apply the techniques to the modalities, where these are not applied.
13	GA-SVM and PSO-SVM method used to classify heart disease [38].	GA and particle swarm optimization (PSO) algorithms combined with SVM achieved a high accuracy.	—–
14	Applied U-NET approach using BraTS2017 data set and prediction of patient survival [39]	89.6% Accuracy achieved with less computational time	—–
15	Two-way path architecture based on CNN for brain tumor segmentation on the BraTS 2013 and 2015 data sets [3]	Input cascaded CNN got a high accuracy with 88.2% on the comparitive analysis with other architechtures.	Further improved the results with increasing architechture layers and data set.

3. Optimization Algorithms

Most of the neural network-based techniques including CNN utilize gradient descent to lower the error rate for the training process and for reforming the internal parameters. Gradient descent is a first-order optimization algorithm, and its derivatives provides direction and increasing or decreasing error function. Information guides the error function, altering it downward to the local minimum [29]. The orthodox batch gradient descent technique computes a gradient of the whole training data, which makes its process computationally slow. To overcome this problem, some algorithms were developed as follows.

3.1. Adaptive Momentum (Adam)

The Adaptive Momentum (Adam) technique estimates the adaptive learning rate for all parameters involved in the training of gradients. It is a computationally efficient and very simple technique that includes first-order gradients with a small memory requirement for stochastic optimization. The proposed technique is utilized in the case of machine learning issues with high-dimensional parameter spaces and huge data sets that calculate learning rates individually for various parameters from approximations that include first- and 2nd-order moments [14]. The mathematically notation for Adam are as follows:

$$x_t = \delta_1 * x_{t-1} - (1 - \delta_1) * g_t \tag{1}$$

$$y_t = \delta_2 * y_{t-1} - (1 - \delta_2) * g_t^2 \tag{2}$$

$$\triangle\omega_t = -\eta \frac{x_t}{\sqrt{y_t + \epsilon}} * g_t \tag{3}$$

$$\omega_{t+1} = \omega_t + \triangle\omega_t \tag{4}$$

- η: Initial learning rate
- g_t: Gradient at time t along ω^j
- x_t: Exponential average of gradient along ω_j
- y_t: Exponential average of squares of gradient along ω_j
- δ_1, δ_2: Hyperparameters

Adam reduces the computational cost, requires less memory for implementation, and is invariant to diagonal rescaling of the gradients. This takes care of the issues such as but not limited to huge data sets, hyperparameters, noisy data, inadequate gradients, and nonstationary problems that required small tuning. Adam configuration parameters are alpha α: this is a learning rate or step size, most presumably picking large esteem (e.g., 0.3) in light of the actuality that it achieves quick learning instead of a smaller esteem and that the outcomes back adapting are perfect during training.

3.2. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) and variants of SGD are commonly used for deep learning. These algorithms have well-defined steps that produce outputs by taking inputs to produce exact results [40,41]. According to cost work and target work, the best algorithms are determined by strategic relapse, linear regression, and neural networks. The main purpose of SGD is to limit the cost work. If there is a huge preparing set, gradient descent is a computationally exceptionally sumptuous system itself. SGD-based algorithms allow users to customize the measures of the algorithms for huge data sets. We performed linear regression utilizing gradient descent, and the equation was as follows:

$$W = \omega - \eta \nabla Q_i(\omega) \tag{5}$$

where $Q_i(\omega)$ is the estimated data, $-Q_i$ being the current data under observation. Generally, Q is an error function; subsequently, by tracking the gradient direction in the space of values of (ω), we move in the direction of (ω) that reduces the error. SGD computes the best (ω) by minimizing Q simultaneously. More importantly, with either perception segmentation or linear regression, (ω) requires the weight parameters of the model and $Q(\omega)$ is a the error for the model. Regular gradient descent is composed as follows:

$$W \leftarrow \eta\nabla Q(\omega) \tag{6}$$

where the error objective is (with its gradient)

$$Q(\omega) = ln\sum iQi(\omega) \;\Rightarrow\; \nabla Q(\omega) = ln\sum i\nabla Qi(\omega) \tag{7}$$

3.3. Momentum

Stochastic gradient descent is a popular optimization technique, but the run time is comparatively high when training the model. Momentum is intended for quick learning, particularly in the face of high curvatures, small but noisy gradients, or steady gradients. The neighborhood minima can be obtained by the utilization of momentum by the quantity of motion of a moving body with respect to its mass and velocity [42]. This is the add-on of the backpropagation technique that updates the weights by decreasing the error rate from the backward direction. A change of direction in the gradient will change the momentum accordingly. Momentum comes in handy, especially when the network is not well defined. Various directions will cause the development of long tight valleys. In these conditions, the inaccurate surface has a comprehensively unique ebb and flow along a gradient descent that does not point towards the base as most point to the surface and the continuous step size of the GD can vacillate from one side to the next. It progresses very slowly to the minima. The expansion of momentum accelerates the intermingling at least by damping these motions. The weight w, momentum m, and given time (t) become:

$$\triangle\omega_{i,j} = \mu\delta_i y_i + m\triangle\omega_{i,j}(t-1) \tag{8}$$

This equation shows that the overall parameters are essentially dictated by experimentations where $0 < m < 1$. Weight refresh presents as one, and momentum adds the fraction m that increases step size towards the minimum when the gradient technique suggests a similar solution. The overall learning rate can be reduced when utilizing a great deal of momentum (m near 1) by surging past the base with excellent stages, and it is joined by an excessive learning rate with momentum. Momentum provides an updated rule which is inspired by the physical perspective of optimization. Imagine a ball in the mountainous area trying to reach the deepest valley, it passes through slight hills when the slope is very high and the ball gains a lot of momentum. The speed of the ball depends on the momentum of the ball, and momentum provides a boost to speed up learning that changes very little to SGD and velocity to make the updates that store velocity for the parameters. The adapted function for SGD uses the momentum updated rule. However, while momentum is very high, the goal is very close which does not know how to slow down the speed. At the beginning, the oscillate minima do not reach the goal. GD has extra cure surfaces in one direction but not in the other direction. It also reduces the oscillation. For updating the weights, it takes the gradient of the current and previous time steps which move faster towards convergence. Convergence is faster when we apply the momentum optimizer to surfaces with curves.

$$v_t = \gamma v_{t-1} + \eta\nabla J(\theta; x, y) \tag{9}$$

$$\theta = \theta - v_t \tag{10}$$

3.4. Adaptive Gradient (Adagrad)

Adagrad adjusts the learning rate according to the parameters, performing bigger updates for inconsistent parameters and smaller updates for successive parameters [43]. The update for each parameter θ_i in each iteration t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\eta}{\sqrt{G_{t-1,ii} + \epsilon}} \cdot g_{t-1,i} \tag{11}$$

where $g_{t-1,i}$ is the gradient to the parameter of the target function θ_i at iteration $t-1$ and ϵ is a smoothing term which dodges division by zero:

$$g_{t-1,i} = \nabla_\theta J(\theta_i) \tag{12}$$

where $g_{t-1,ii} \epsilon R_{d\times}$ is a diagonal matrix and element i is the sum of the squares of the gradients to θ_i to iteration $t-1$. By means of an element-wise matrix– vector multiplication $\odot$between G_{t-1} and g_{t-1}, the vectorization of

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_{t-1,ii} + \epsilon}} \cdot g_{t-1,i} \tag{13}$$

Adagrad takes out the requirement to manually tune the learning rate; however, its gathering of the squared gradients in the denominator causes the learning rate to shrivel and to moderate intermingling speed.

3.5. Adaptive Delta (AdaDelta)

AdaDelta is an extension of Adagrad which clears the rotting learning rate issue but, unlike Ada Grad, does not collect past squared gradient. It restricts the window of gathered previous gradient [44]. The AdaDelta technique strongly adjusts weights after using the first-order time only and takes minimum computational costs compared to previous techniques. In this technique, there is no manual tuning or learning. Moreover, it is robust to raucous gradient information, data modalities, hyperparameters, and model design decisions. It improves the steepest descent direction stated by a negative gradient.

$$\nabla x_t = -\eta g_t \tag{14}$$

where g_t is the gradient at the ith iteration $\frac{\delta f(x_t}{\delta f(x_t})$ and η is a learning rate which controls how vast the stage is toward the negative gradient. Picking a learning rate and exhibiting another unique learning rate evaluated per measurement by utilizing the first order, it utilizes a modest quantity of calculation per iteration in gradient descent, which is a downside to AdaDelta. Some of their discovered hyper parameters are not up to the best degree to adjust outcomes. Inspired from Adagrad, the two primary disadvantages are

- the incessant rot of learning rates for the training time and
- the requirement for automatically chosen comprehensive learning rates.

Notwithstanding across the board assortment of input data types, nonlinearity, total hidden units, number of distributed imitations, and the hyperparameters that do not need to be balanced are some concrete reasons exhibiting that AdaDelta has strong learning that can be useful in grouped assortment of conditions and that there is no requirements for the physical setting of a learning rate.

3.6. Adaptive Max Pooling (Adamax)

Adamax is inspired from Adam; the changes are made on how the infinity norm (ut) is used. It was demonstrated that the v_t value in Adam with 1 will merge to a progressively stable value [45].

$$u_t = \beta_2^\infty \cdot v_{t-1} + (\beta_2^\infty) \cdot |g_t|^\infty = max(\beta_2 \cdot v_{t-1}, |g_t|) \tag{15}$$

3.7. Nesterov Adaptive Momentum (Nadam)

Reference [5] presented a variant of the momentum algorithm inspired by Nesterovs accelerated gradient method [46].

$$v \leftarrow \alpha v - \epsilon \nabla_\theta [\frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta + \alpha v), y^(i))] \tag{16}$$

$$\theta \leftarrow \theta + v \tag{17}$$

where the parameters α and ϵ play similar roles as in the standard momentum method. The difference between Nesterov momentum and standard momentum is evaluation of the gradient. With Nesterov momentum, the gradient is evaluated after the existing velocity is applied. Thus, one can interpret Nesterov momentum as an attempt to add a correction factor to the standard method of momentum.

3.8. Root Mean Square Propagation (RMSProp)

Root Mean Square Propagation (RMSProp) was invented by Geoffrey Hinton. It is similar to the gradient descent algorithm with momentum. RMSProp tries to resolve Adagrad's radically diminishing learning rates by using a moving average of the squared gradient, which utilizes the magnitude of recent gradient descents for normalization of the gradient. Therefore, with the increase of the learning rate, the algorithm used would move in a horizontal direction with larger steps converging faster.

$$E[g^2]_t = 0.9E[g^2]_{t+1} + 0.1g_t^2 \tag{18}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1-\gamma)g_{t-1}^2} + \gamma g_t + \epsilon} \cdot g_t \tag{19}$$

γ is the decay term that takes a value from 0 to 1. g_t is the moving average of squared gradients.

3.9. Cyclic Learning Rate (CLR)

Learning rate is a hyperparameter that controls how much you are adjusting the weights of our network with respect to the loss gradient. It is because you are on your way to optimizing a neural network that you have just created with gradient descent. Now, essentially the goal of gradient descent is to find the minima of the loss function that your neural network is trying to optimize.

- CLR provides a technique for setting the global learning rates for training neural systems that take out the the need to perform tons of investigations to locate the best values with no extra computations.
- CLR provides an excellent learning rate range (LR range) for an experiment by introducing the concept of LR range test.

3.10. Nesterov Accelerated Gradient (NAG)

Nesterov acceleration optimization is similar to a ball rolling down the peak but knowing exactly when to slow down before the gradient of the hill increases again. We can calculate the gradient not with respect to the present step but with respect to the future step. We estimate the gradient of the gain, and based on the importance, we will update the weights accordingly. When going down the peak where we can look ahead in the future, we can optimize the descent faster, which is the reason it works slightly better than standard momentum.

$$\theta = \theta - v_t \tag{20}$$

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta - \gamma v_{t-1}) \tag{21}$$

Selection of a good starting learning rate is just the first step. Further, we need to gradually decrease the learning rate while training for a robust model. The learning rate becomes constant during the course of training, which might become too large to converge and causes the loss function to change around the local minimum. This approach is used for a higher learning rate to rapidly reach the regions of (local) minima, while in the initial training stage, a smaller learning rate is set, as training progresses, exploring deeper and more thoroughly in the region to evaluate the minimum.

4. Data Set and Methodology

The proposed technique has been trained and validated on the BraTS2015 databases [47]. In BraTS2015, there are four MRI sequences available for every patient: FLAIR, T1-weighted (T1), T2-weighted (T2), and T1-weighted (T1c). The training set involves 220 High-Grade Gliomas (HGG) and 54 Low-Grade Gliomas (LGG) in the BraTS2015 challenge data set. We extracted around 268,000 and 360,000 patches to train our proposed CNNs for LGG and HGG, respectively. The proposed CNNs were developed using Tensorflow backend.

The proposed ConvNet CNN architecture was utilized in the investigation. In order to get enough information on the optimizer's performance and accuracy, the proposed convNet architecture was trained and validated with different optimizers explained in the above section. We divided the data set in Table 2 for training 80% and validation 20%. The ConvNet architecture was validated using 10 optimizers based on gradient descent (as shown in the above section), 2 options of the data batch size (128), 3 options of the epoch $(0, 50, \ldots 250, 300)$, and 4 options of the learning rate (1e^{-1}, 1e^{-2} …1e^{-10}). We used the patches of the different patients in the training and validation processes. Details of the architecture are given in Table 3. The proposed ConvNet architecture consists of 8 convolution layers, 3 max-pooling layers, and 3 fully connected layers. Some activation functions used in the ConvNet architecture include Rectified Linear Unit (ReLU) and Soft Max. The pooling layer is responsible for dimensionality reduction over time; it reduces the spatial size in every step to decrease the number of selected parameters for the subsequent step by operating on each feature map independently. The output from the final pooling or convolution layer follows a one-dimensional (1D) array; from here, the architecture is completely the same as ANN, in which, all the values in the 1D array are fully connected to every output by a temporary weight given to each of them. Each fully connected layer is followed by a nonlinear function; the fully connected layers typically have the same number of output nodes as the number of classes. Design details can be found in Table 3, and the ConvNet architectural is shown in Figure 3.

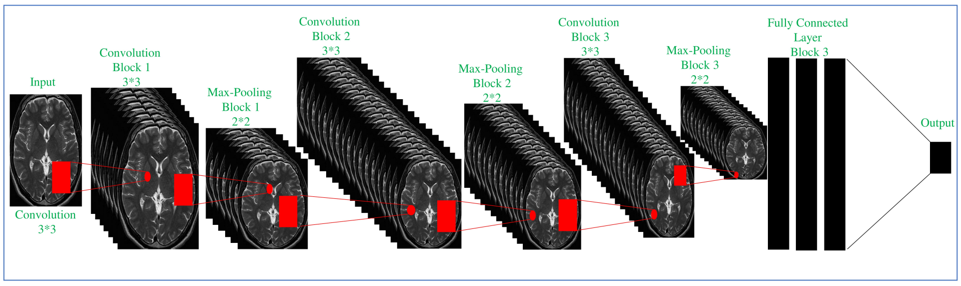


Figure 3. Proposed architecture.

Table 2. Data set used for proposed technique: Four MRI modalities are used for both tumor types High-Grade Glioma (HGG) and Low-Grade Glioma (LGG). The modalities are T1-weighted (T1), 2-weighted (T2), T1-weighted (T1c), and FLAIR. The initial two parameters represent the tumor type and number of patients, and the next parameter represents the number of patches extracted for training and validation.

Tumor Type	No. Patients	No. Patches Extracted	
		Training	**Testing**
HGG	220	360,000	90,000
LGG	54	268,000	67,000

Table 3. Detail of the proposed patch-wise model architecture of Convolutional Neural Network (CNN): In inputs, the first dimension refers to the number of channels and the next two are the size of patch and feature maps, respectively.

Block	No.of Filter	Name (Size)	Stride	Kernel Size
Input		Input Image		-
Convolution block 1	64	Con-1ayer 1 (4 × 33 × 33)	1 × 1	3 × 3
	-	Relu-1ayer		-
	64	Con-1ayer 2 (33 × 33 × 64)		3 × 3
	-	Relu-1ayer		-
	64	Con-1ayer 3 (33 × 33 × 64)		3 × 3
	-	Relu-1ayer		-
Pooling block 1	-	Max-Pooling layer 4 (33 × 33 × 64)	2 × 2	3 × 3
Convolution block 2	128	Con-1ayer 5 (4 × 33 × 33)	1 × 1	3 × 3
	-	Relu-1ayer		-
	128	Con-1ayer 6 (33 × 33 × 64)		3 × 3
	-	Relu-1ayer		-
	128	Con-1ayer 7 (33 × 33 × 128)		3 × 3
		Relu-1ayer		-
Pooling block 2	-	MAX-Pooling layer8 (33 × 33 × 128)	2 × 2	3 × 3
Convolution block 3	128	Con-1ayer 9 (33 × 33 × 128)	1 × 1	3 × 3
	-	Relu-1ayer		-
	128	Con-1ayer 10 (33 × 33 × 128)		3 × 3
	-	Relu-1ayer		-
Pooling block 3	-	MAX-Pooling layer 11 (33 × 33 × 128)	2 × 2	3 × 3
Fully Connected block	-	FC-1ayer 12 32768	-	-
	-	FC-1ayer 13 256		-
	-	FC-1ayer 14 256		-
	-	Softmax-1ayer		-

5. Experimental Results and Discussion

We used Tensorflow library for the implementation of our model on Z840 workstation Intel Xeon (R) CPU E5-2630v3 @2.40GHz*32 with 64 GB memory. To validate the effectiveness, the proposed CNN-based approach with extra convolutional layers was used to classify brain tumor disease. We used the Monte Carlo method to check the significance of the classification and segmentation results under optimal parameters. We performed the analysis on the different number of epochs, and we noticed that the average performance results were achieved on 300 epochs. Table 4 shows the hyperparameters of our proposed technique.

Table 4. Hyperparameters for our proposed technique.

Stage	Hyperparameter	Value
	Bias	0.1
	Weights	Xavier
ReLU	α	0.333
Dropout	HGG	0.1
	LGG	0.5
Training	Epochs-HGG	50–300
	Epochs-LGG	50–300
	Intial ϵ	0.03
	Final ϵ	0.0003
	Batch Size	128
Post processing	Tvol-HGG	10,000
	Tvol-HGG	3000

Tables 5–10 show a summary of the experiment results. The experimental results uses ten gradient descent optimizers with different number of epochs and various learning rates. Tables 5–8 give the results at epochs 50, 100, 200, and 300. Tables 9 and 10 provide the results with learning rate 1e^{-1}, 1e^{-2}, . . . 1e^{-10}.

Table 5. The training accuracy of ten optimizers on a proposed patch-wise model architecture of CNN.

Epoch $\rightarrow$ **Optimizers** $\downarrow$	**50**	**100**	**150**	**200**	**250**	**300**
Adam	0.97	0.98	0.98	0.98	0.99	0.99
Adagrad	0.95	0.96	0.96	0.96	0.96	0.96
AdaDelta	0.95	0.96	0.96	0.96	0.96	0.96
SGD	0.95	0.967	0.968	0.97	0.97	0.97
NAG	0.94	0.94	0.94	0.94	0.95	0.95
Rmsprop	0.95	0.95	0.95	0.95	0.95	0.95
Momentum	0.96	0.96	0.97	0.97	0.974	0.97
Adamax	0.95	0.95	0.95	0.96	0.96	0.96
CLR	0.96	0.96	0.96	0.96	0.96	0.96
Nadam	0.96	0.96	0.96	0.96	0.97	0.97

Table 6. Validation accuracy of ten optimizers on a proposed patch-wise model architecture of CNN.

Epoch $\rightarrow$ **Optimizers** $\downarrow$	**50**	**100**	**150**	**200**	**250**	**300**
Adam	0.97	0.97	0.97	0.98	0.98	0.99
Adagrad	0.96	0.96	0.96	0.96	0.96	0.96
AdaDelta	0.95	0.96	0.96	0.96	0.96	0.96
SGD	0.96	0.96	0.96	0.97	0.97	0.97
NAG	0.94	0.94	0.94	0.94	0.95	0.95
Rmsprop	0.95	0.95	0.95	0.95	0.95	0.95
Momentum	0.96	0.96	0.97	0.97	0.97	0.97
Adamax	0.95	0.95	0.95	0.96	0.96	0.96
CLR	0.96	0.96	0.96	0.96	0.96	0.96
Nadam	0.96	0.96	0.96	0.96	0.97	0.97

Table 7. Validation loss of ten optimizers on a proposed patch-wise model architecture of CNN.

Epoch $\rightarrow$ **Optimizers** $\downarrow$	**50**	**100**	**150**	**300**	**250**	**300**
Adam	0.05	0.05	0.04	0.04	0.04	0.04
Adagrad	0.07	0.06	0.06	0.06	0.06	0.06
AdaDelta	0.07	0.06	0.06	0.06	0.06	0.05
SGD	0.06	0.06	0.06	0.06	0.06	0.06
NAG	0.09	0.08	0.08	0.08	0.08	0.07
Rmsprop	0.07	0.07	0.07	0.07	0.07	0.07
Momentum	0.05	0.05	0.05	0.05	0.05	0.04
Adamax	0.07	0.07	0.07	0.07	0.07	0.07
CLR	0.06	0.05	0.04	0.04	0.04	0.04
Nadam	0.05	0.05	0.05	0.05	0.05	0.05

Table 8. Training loss of ten optimizers on a proposed patch-wise model architecture of CNN.

Epoch $\rightarrow$ **Optimizers** $\downarrow$	**50**	**100**	**150**	**300**	**250**	**300**
Adam	0.08	0.06	0.06	0.05	0.05	0.04
Adagrad	0.12	0.11	0.11	0.09	0.09	0.08
AdaDelta	0.13	0.09	0.09	0.09	0.09	0.09
SGD	0.15	0.13	0.13	0.12	0.12	0.12
NAG	0.16	0.16	0.15	0.15	0.14	0.14
Rmsprop	0.17	0.16	0.16	0.15	0.15	0.15
Momentum	0.12	0.11	0.10	0.09	0.09	0.09
Adamax	0.11	0.11	0.09	0.09	0.09	0.09
CLR	0.15	0.13	0.12	0.12	0.11	0.11
Nadam	0.10	0.09	0.08	0.08	0.08	0.08

Table 9. Accuracy rate of ten optimizers with various learning rates on a proposed patch-wise model architecture of CNN.

Learning Rate $\rightarrow$ **Optimizers** $\downarrow$	$\mathbf{1e^{-1}}$	$\mathbf{1e^{-2}}$	$\mathbf{1e^{-3}}$	$\mathbf{1e^{-4}}$	$\mathbf{1e^{-5}}$	$\mathbf{1e^{-6}}$	$\mathbf{1e^{-7}}$	$\mathbf{1e^{-8}}$	$\mathbf{1e^{-9}}$	$\mathbf{1e^{-10}}$
Adam	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.98	0.98
Adagrad	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.96
AdaDelta	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
SGD	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96
NAG	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Rmsprop	0.96	0.96	0.96	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Momentum	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.98	0.97	0.97
Adamax	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
CLR	0.97	0.97	0.97	0.97	0.97	0.96	0.96	0.97	0.96	0.96
Nadam	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.969	0.97	0.96

Table 10. Error rate of ten optimizers with various learning rates on a proposed patch-wise model architecture of CNN.

Learning Rate → **Optimizers** ↓	**1e^{-1}**	**1e^{-2}**	**1e^{-3}**	**1e^{-4}**	**1e^{-5}**	**1e^{-6}**	**1e^{-7}**	**1e^{-8}**	**1e^{-9}**	**1e^{-10}**
Adam	0.05	0.03	0.04	0.06	0.07	0.1	0.1	0.07	0.1	0.1
Adagrad	1.04	0.86	0.56	0.37	0.54	0.77	0.81	0.54	0.77	0.81
AdaDelta	1.86	1.81	1.83	2.01	2.08	2.17	2.2	2.08	2.17	2.2
SGD	0.53	0.49	0.47	0.63	0.67	0.81	0.97	0.67	0.81	0.97
NAG	2.3	2.25	2.19	2.39	2.43	2.67	2.79	2.43	2.67	2.79
Rmsprop	2.12	2.13	2.1	2.17	2.29	2.15	2.49	2.29	2.15	2.49
Momentum	0.25	0.26	0.28	0.43	0.49	0.51	0.57	0.49	0.51	0.57
Adamax	1.69	1.52	1.26	1.49	1.6	1.92	2.09	1.6	1.92	2.09
CLR	1.88	1.88	1.79	1.97	2.05	2.15	2.45	2.05	2.15	2.45
Nadam	1.79	1.76	1.65	1.69	1.81	2.07	2.31	1.81	2.07	2.31

The flowchart of the proposed method is defined in Figure 4. There is likewise no sign of overfitting, implying that the architecture performs well. The smallest error rate was obtained by the Adam optimizer using our proposed CNN architecture. Adam is the most successful optimizer in our all experiments. Other optimizers also performed well, and the performances of SGD and momentum are close to that of Adam. Figure 5 shows the performances of ten optimizers using CNN architecture with various epochs and learning rates. From Figure 5a, we can observe that Adam, AdaDelta, and SGD provide the highest validation rates at 300 epochs. Figure 5b shows that Adam has the highest training accuracy whereas NAG has the lowest training accuracy. Other than Adam, momentum and SGD likewise have the potential to accomplish high training accuracy. Figure 5c shows that Adam provides the smallest validation loss whereas RMSProp, NAG, and Adamax have the biggest validation losses. From Figure 5d, we can see that RMSProp, NAG, and CLR provide the highest trainng losses whereas Adam has the smallest training loss. Henceforth, it could be construed that each optimizer shows diverse execution crosswise over various epochs using CNN architecture. A comparison of our proposed method has been presented in Table 11. Reference [29] experimented on different auto-encoders to demonstrate that NAG has better abilities in terms of reducing the gradient norms, and it also produces iterates which exhibit an increasing trend for the minimum eigenvalue of the Hessian of the loss function at the iterates. Four classes of the CIFAR-10 data set are chosen for the experiments.The proposed EIOM is compared with other optimizers, namely AdaGrad, AdaDelta, RMSProp, Adam, and CLR that produced 97% accuracy [31]. Extensive experiments on four widely used benchmark databases were conducted to verify the effectiveness of the proposed deep convolutional neural network (DCNN) and obtained 97.9% accuracy.

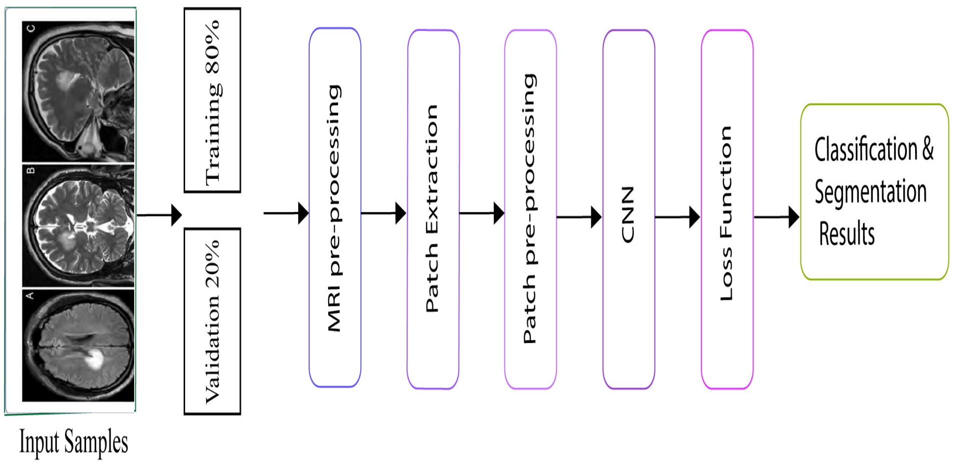
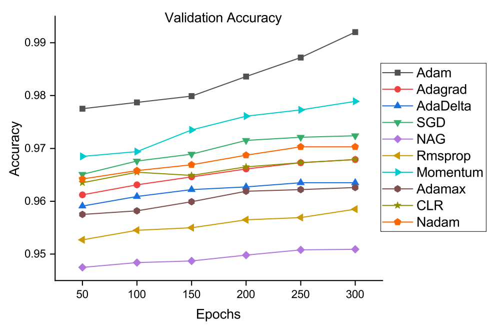
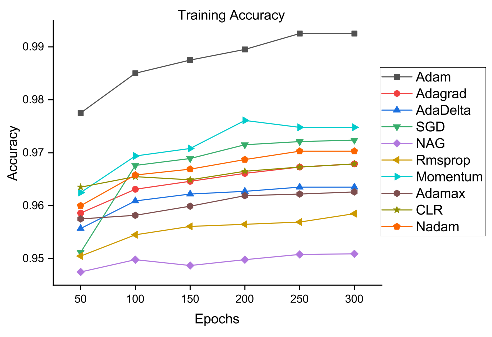


Figure 4. Proposed model flow chart.

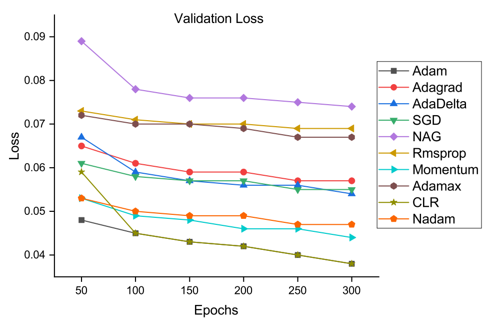
From Figure 6a,b, the segmentations results for the HGG and LGG cases are compared with the delineated ground truth: (a) segmented tumor of a HGG case overlaid on the FLAIR image; (b) segmented tumor of a LGG case overlaid on the Flair image. From Figure 7a,b, further examination of the potential optimizers for CNN architecture is delineated. Some of learning rates were added to portray its conduct against the likelihood of overtraining. As much as $1e^{-1}$, $1e^{-2} \ldots 1e^{-10}$ learning rate was experimented to train our proposed CNN with all optimizers. It was demonstrated that all optimizers did not require much learning rate to arrive at the minimum error rate and that all optimizers could converge well overall. In any case, Adam is the most steady one among the ten optimizers. Then again, NAG, RMSProp, and CLR were not successful to do as such. The error rate of SGD and momentum tended to diminish when the learning rate was smaller than $1e^{-5}$. Compared to Nadam, Adagrade and AdaDelta have worse performance. Therefore, Adam is the best choice for brain tumor segmentation using our proposed CNN architecture. From the above discussion, it is uncovered that our proposed CNN architecture with various optimization algorithms gives impressive results for brain tumour segmentation using MRIs. From the pack of our experiment data, it is conceivable to explain the behaviour of each optimizer against CNN architecture. Finally, it can be seen that our proposed model achieves state-of-the-art results when comparing it with existing models. We also provide an in-depth analysis of our proposed method and produce 99.20% accuracy.



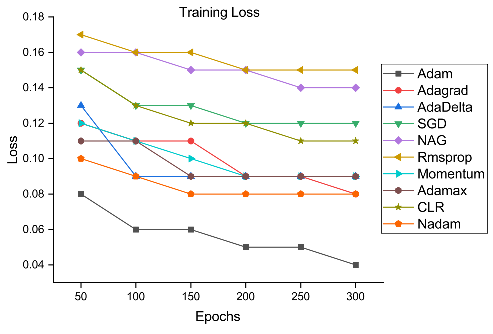
(**a**)



(**b**)

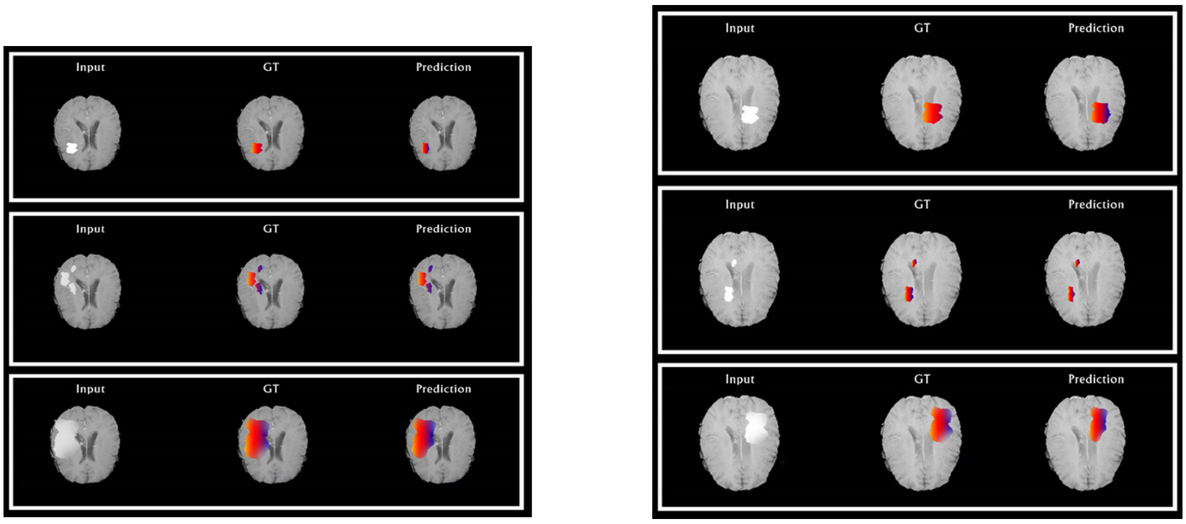


(**c**)



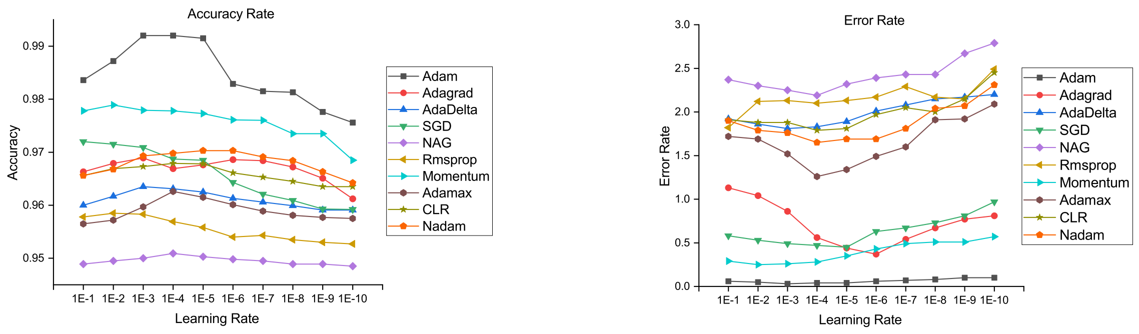
(**d**)

Figure 5. Validation accuracy and loss comparison of all optimizers using our proposed architecture: (**a**) validation accuracy, (**b**) training accuracy, (**c**) validation loss, and (**d**) training loss.



(**a**) (**b**)

Figure 6. Automatic segmentation results of HGG (**a**) and LGG (**b**) cases. Red: edema, blue: non-enhancing tumor. From left to right: (i) original image, (ii) ground truth, and (iii) demonstration of automatic segmentation results from the proposed method. (**a**) The segmentation results of HGG cases compared to their ground truth and (**b**) the segmentation results of LGG cases compared to their ground truth.



(**a**) (**b**)

Figure 7. Accuracy rate and error rate comparison of all optimizers using our proposed architecture: (**a**) accuracy rate and (**b**) error rate.

Table 11. Comparison of accuracy values with state-of-the-art techniques.

Paper	Method	Data Set	Accuracy
[43]	CIFAR ConvNet	Mnist Dataset	90.00%
[48]	DCNN(AlexNet)	ILSVRC2012	97.90%
[29]	VGG-9	CIFAR-10	99.00%
[31]	2D-CNN	CIFAR-10	97.00%
Proposed Method	ConvNet based	BraTS2015	99.20%

6. Conclusions

This paper is a comparative analysis of different optimization algorithms used in our proposed CNN architecture to measure the performance for brain tumor segmentation. The comparison is made on publicly available an MRI brain image data set, i.e., BraTS2015. Both quantitative and graphical results show that all optimizers perform consistently but that Adam performs much better. Among the 10 optimizers for our architecture, Adam has the smallest error rate and the highest accuracy rate when it reaches the minimum on a particular epoch. The NAG and RMSProp optimizers failed badly. Due to limited resources to run several architectures, AdaDelta and Adamax should be used to provide minimal risk. The performances of the momentum and SGD optimizers were inferior to that of Adam. The adapted pipeline of the CNN optimizer comparison concludes that the performance of Adam is comparable with the latest research. Future work will compare this state-of-the-art optimizer with multiple CNN architectures used for brain tumor segmentation.

Author Contributions: M.Y.: conceptualization, methodology, writing—orignal draft. J.F.: writing—review and editing, writing—original draft preparation, and supervision. K.A.: visualization, project administration, and formal analysis. M.S.Z.: software and validation. Z.U.R.: conceptualization and investigation. K.J.: formal analysis. A.M.: resources and data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been sponsored by the National Science Foundation of China under grant No. 81871394, by the Beijing Municipal Education Committee Science Foundation under grant No. KM201810005030, and by Beijing Laboratory of Advanced Information Networks under grant No. PXM2019_014204_500029.

Acknowledgments: Bundle of thanks for the database organizers of BraTS2015 for the evaluation of automated methods of brain tumour on MRI scans.

Conflicts of Interest: We (the authors) certify that we have no affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interests; and expert testimony or patent-licensing arrangements) or nonfinancial interest (such as personal or professional relationships, affiliations, knowledge, or beliefs) in the subject matter or materials discussed in this manuscript.

Abbreviations

The following abbreviations are used in this manuscript:

DLA	Deep learning algorithms
CNN	Convolutional Neural Network
MRI	Magnetic Resonance Images
Adagrad	Adaptive Gradient
AdaDelta	Adaptive Delta
SGD	Stochastic Gradient14Descent
Adam	Adaptive Momentum
CLR	Cyclic Learning Rate
Adamax	Adaptive Max Pooling15
RMS Prop	Root Mean Square Propagation
NADAM	Nesterov Adaptive Momentum
NAG	Nesterov accelerated gradient
HGG	High-Grade Gliomas
LGG	Low-Grade Gliomas
CNS	Central nervous system
ANN	Artificial Neural Network
GT	Ground Truth
CV	Computer Vision
PSO	Particle Swarm Optimization
FLAIR	Fluid Attenuation Inversion Recovery
SIFT	Scale-Invariant Feature Transform
DCNN	Deep Convolutional Neural Network

References

1. Walker, E.V.; Davis, F.G. Malignant primary brain and other central nervous system tumors diagnosed in Canada from 2009 to 2013. *Neuro Oncol.* **2019**, *21*, 360–369. [CrossRef] [PubMed]
2. Mzoughi, H.; Njeh, I.; Wali, A.; Slima, M.B.; BenHamida, A.; Mhiri, C.; Mahfoudhe, K.B. Deep Multi-Scale 3D Convolutional Neural Network (CNN) for MRI Gliomas Brain Tumor Classification. *J. Digit. Imaging* **2020**. [CrossRef] [PubMed]

3. Havaei, M.; Davy, A.; Warde-Farley, D.; Biard, A.; Courville, A.; Bengio, Y.; Pal, C.; Jodoin, P.M.; Larochelle, H. Brain tumor segmentation with deep neural networks. *Med Image Anal.* **2017**, *35*, 18–31. [CrossRef] [PubMed]
4. Fukushima, K. Neocognitron. *Scholarpedia* **2007**, *2*, 1717. [CrossRef]
5. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. *Int. Conf. Mach. Learn.* **2013**, 1139–1147.
6. Fletcher, E.; Knaack, A. Applications of deep learning to brain segmentation and labeling of mri brain structures. *Handb. Pattern Recognit. Comput. Vis.* **2020**, 251.
7. Ahmad, A.; Hassan, M.; Abdullah, M.; Rahman, H.; Hussin, F.; Abdullah, H.; Saidur, R. A review on applications of ANN and SVM for building electrical energy consumption forecasting. *Renew. Sustain. Energy Rev.* **2014**, *33*, 102–109. [CrossRef]
8. Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Netw.* **1988**, *1*, 119–130. [CrossRef]
9. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
10. Lee, S.G.; Sung, Y.; Kim, Y.G.; Cha, E.Y. Variations of AlexNet and GoogLeNet to Improve Korean Character Recognition Performance. *J. Inf. Process. Syst.* **2018**, *14*.
11. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
12. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef] [PubMed]
13. Yang, T.; Wu, Y.; Zhao, J.; Guan, L. Semantic segmentation via highly fused convolutional network with multiple soft cost functions. *Cogn. Syst. Res.* **2019**, *53*, 20–30. [CrossRef]
14. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
15. Graham, B. Spatially-sparse convolutional neural networks. *arXiv* **2014**, arXiv:1409.6070.
16. Kayalibay, B.; Jensen, G.; van der Smagt, P. CNN-based segmentation of medical imaging data. *arXiv* **2017**, arXiv:1701.03056.
17. Hosseini, H.; Xiao, B.; Jaiswal, M.; Poovendran, R. On the limitation of convolutional neural networks in recognizing negative images. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 352–358.
18. Bao, P.; Zhang, L.; Wu, X. Canny edge detection enhancement by scale multiplication. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1485–1490. [CrossRef] [PubMed]
19. Bouvrie, J. Notes on Convolutional Neural Networks. 2006. Unpublished.
20. Mehmood, A.; Maqsood, M.; Bashir, M.; Shuyuan, Y. A Deep Siamese Convolution Neural Network for Multi-Class Classification of Alzheimer Disease. *Brain Sci.* **2020**, *10*, 84. [CrossRef]
21. Ghoreishi, S.F.; Imani, M. Bayesian optimization for efficient design of uncertain coupled multidisciplinary systems. In Proceedings of the 2020 American Control Conference (ACC 2020), Denver, CO, USA, 1–3 July 2020; IEEE: Piscataway, NJ, USA, 2020.
22. Sultan, H.H.; Salem, N.M.; Al-Atabany, W. Multi-classification of Brain Tumor Images using Deep Neural Network. *IEEE Access* **2019**, *7*, 69215–69225. [CrossRef]
23. Swati, Z.N.K.; Zhao, Q.; Kabir, M.; Ali, F.; Ali, Z.; Ahmed, S.; Lu, J. Content-Based Brain Tumor Retrieval for MR Images Using Transfer Learning. *IEEE Access* **2019**, *7*, 17809–17822. [CrossRef]
24. Becherer, N.; Pecarina, J.; Nykl, S.; Hopkinson, K. Improving optimization of convolutional neural networks through parameter fine-tuning. *Neural Comput. Appl.* **2019**, *31*, 3469–3479. [CrossRef]
25. Barkana, B.D.; Saricicek, I.; Yildirim, B. Performance analysis of descriptive statistical features in retinal vessel segmentation via fuzzy logic, ANN, SVM, and classifier fusion. *Knowl. Based Syst.* **2017**, *118*, 165–176. [CrossRef]
26. Emery, N.J.; Seed, A.M.; Von Bayern, A.M.; Clayton, N.S. Cognitive adaptations of social bonding in birds. *Philos. Trans. R. Soc. Biol. Sci.* **2007**, *362*, 489–505. [CrossRef] [PubMed]

27. He, S.; Wu, Q.; Saunders, J. A group search optimizer for neural network training. In *International Conference on Computational Science and Its Applications*; Springer: Berlin, Germany, 2006; pp. 934–943.
28. Yousoff, S.N.M.; Baharin, A.; Abdullah, A. A review on optimization algorithm for deep learning method in bioinformatics field. In Proceedings of the 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), Kuala Lumpur, Malaysia, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 707–711.
29. De, S.; Mukherjee, A.; Ullah, E. Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration. *arXiv* **2018**, arXiv:1807.06766.
30. Prilianti, K.; Brotosudarmo, T.; Anam, S.; Suryanto, A. Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image. *AIP Conf. Proc.* **2019**, *2084*, 020020.
31. Zhao, H.; Liu, F.; Zhang, H.; Liang, Z. Research on a learning rate with energy index in deep learning. *Neural Netw.* **2019**, *110*, 225–231. [CrossRef]
32. Moulines, E.; Bach, F.R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In Proceedings of the Advances in Neural Information Processing Systems, Granada, Spain, 12–14 December 2011; pp. 451–459.
33. Chandra, B.; Sharma, R.K. Deep learning with adaptive learning rate using laplacian score. *Expert Syst. Appl.* **2016**, *63*, 1–7. [CrossRef]
34. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef]
35. Sajjad, M.; Khan, S.; Muhammad, K.; Wu, W.; Ullah, A.; Baik, S.W. Multi-grade brain tumor classification using deep CNN with extensive data augmentation. *J. Comput. Sci.* **2019**, *30*, 174–182. [CrossRef]
36. Afshar, P.; Mohammadi, A.; Plataniotis, K.N. Brain tumor type classification via capsule networks. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 3129–3133.
37. Qayyum, A.; Anwar, S.M.; Majid, M.; Awais, M.; Alnowami, M. Medical image analysis using convolutional neural networks: A review. *arXiv* **2017**, arXiv:1709.02250.
38. Iftikhar, S.; Fatima, K.; Rehman, A.; Almazyad, A.S.; Saba, T. An evolution based hybrid approach for heart diseases classification and associated risk factors identification. *Biomed. Res.* **2017**, *28*, 3451–3455.
39. Isensee, F.; Kickingereder, P.; Wick, W.; Bendszus, M.; Maier-Hein, K.H. Brain tumor segmentation and radiomics survival prediction: contribution to the BRATS 2017 challenge. In *International MICCAI Brainlesion Workshop*; Springer: Berlin, Germany, 2017; pp. 287–297.
40. Wu, H.; Yang, S.; Huang, Z.; He, J.; Wang, X. Type 2 diabetes mellitus prediction model based on data mining. *Inform. Med. Unlocked* **2018**, *10*, 100–107. [CrossRef]
41. Doike, T.; Hayashi, K.; Arata, S.; Mohammad, K.N.; Kobayashi, A.; Niitsu, K. A Blood Glucose Level Prediction System Using Machine Learning Based on Recurrent Neural Network for Hypoglycemia Prevention. In Proceedings of the 2018 16th IEEE International New Circuits and Systems Conference (NEWCAS), Montreal, QC, Canada, 24–27 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 291–295.
42. Pereira, D.A.; Ourique de Morais, W.; Pignaton de Freitas, E. NoSQL real-time database performance comparison. *Int. J. Parallel Emergent Distrib. Syst.* **2018**, *33*, 144–156. [CrossRef]
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Alfian, G.; Syafrudin, M.; Ijaz, M.; Syaekhoni, M.; Fitriyani, N.; Rhee, J. A Personalized Healthcare Monitoring System for Diabetic Patients by Utilizing BLE-Based Sensors and Real-Time Data Processing. *Sensors* **2018**, *18*, 2183. [CrossRef]
45. Huh, J.H. Big Data Analysis for Personalized Health Activities: Machine Learning Processing for Automatic Keyword Extraction Approach. *Symmetry* **2018**, *10*, 93. [CrossRef]
46. Kögel, M.; Findeisen, R. A fast gradient method for embedded linear predictive control. *IFAC Proc. Vol.* **2011**, *44*, 1362–1367. [CrossRef]

47. Menze, B.H.; Jakab, A.; Bauer, S.; Kalpathy-Cramer, J.; Farahani, K.; Kirby, J.; Burren, Y.; Porz, N.; Slotboom, J.; Wiest, R.; et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Trans. Med. Imaging* **2014**, 34, 1993–2024. [CrossRef]
48. Bai, C.; Huang, L.; Pan, X.; Zheng, J.; Chen, S. Optimization of deep convolutional neural network for large scale image retrieval. *Neurocomputing* **2018**, *303*, 60–67. [CrossRef]