Brain Tumor Classification Using Convolutional Neural Network

Nyoman Abiwinanda, Muhammad Hanif, S. Tafwida Hesaputra, Astri Handayani, and Tati Rajab Mengko

Abstract

Misdiagnosis of brain tumor types will prevent effective response to medical intervention and decrease the chance of survival among patients. One conventional method to differentiate brain tumors is by inspecting the MRI images of the patient's brain. For large amount of data and different specific types of brain tumors, this method is time consuming and prone to human errors. In this study, we attempted to train a Convolutional Neural Network (CNN) to recognize the three most common types of brain tumors, i.e. the Glioma, Meningioma, and Pituitary. We implemented the simplest possible architecture of CNN; i.e. one each of convolution, max-pooling, and flattening layers, followed by a full connection from one hidden layer. The CNN was trained on a brain tumor dataset consisting of 3064 T-1 weighted CE-MRI images publicly available via figshare Cheng (Brain Tumor Dataset, 2017 [1]). Using our simple architecture and without any prior region-based segmentation, we could achieve a training accuracy of 98.51% and validation accuracy of 84.19% at best. These figures are comparable to the performance of more complicated region-based segmentation algorithms, which accuracies ranged between 71.39 and 94.68% on identical dataset Cheng (Brain Tumor Dataset, 2017 [1], Cheng et al. (PLoS One 11, 2017 [2]).

Keywords

Training loss • Training accuracy • Validation loss
Validation accuracy • Overfitting

1 Introduction

On 2016, brain tumor was the leading cause of cancer-related death in children (ages 0–14) in the United States and ranked above Leukemia [3]. Brain and CNS tumors are also the third most common cancer occurring among teenager and adolescents (ages 15–39) [4]. Different types of brain tumors require different medical interventions. In conventional computer-aided diagnosis systems, the tumor mass itself has to be identified and segmented before it can be classified into different types. Upon tumor mass segmentation, the segmented region is then subjected to feature extraction and classification.

Recent studies of identification and segmentation of brain tumor [5, 6] found no universal system for accurate tumor detection system regardless of its location, shape, and intensity [6]. There are numerous proposed algorithms in recent studies for feature extraction and classification of brain tumors. Grey-level co-occurrence matrix (GLCM) [7–9] is commonly used for extraction of low-level features. Several other feature extraction algorithms which attempt to handle the complex texture of brain tumor are Neural Network [9, 10], Bag-of-Words (BoW) [2, 8], and Fisher Vector [2]. One recent study showed that by using a combination of adaptive spatial pooling and fisher vector algorithm, brain tumor classification into Glioma, Meningioma, and Pituitary can be achieved with 71.39–94.68% accuracy [2].

Conventional brain tumor classification methods commonly involve region-based tumor segmentation prior to feature extraction and classification. In this paper, we propose an automatic brain tumor segmentation/classification method based on Convolutional Neural Networks. CNN consists of a convolutional network to perform automatic segmentation

N. Abiwinanda (✉) · M. Hanif · S. T. Hesaputra · A. Handayani
T. R. Mengko
Bandung Institute of Technology, Bandung, West Java 40134, Indonesia
e-mail: abiwinanda@outlook.com

M. Hanif
e-mail: mhaniffarhat@gmail.com

S. T. Hesaputra
e-mail: tafwidahesaputra@gmail.com

A. Handayani
e-mail: a.handayani@stei.itb.ac.id

T. R. Mengko
e-mail: tmengko@stei.itb.ac.id

L. Lhotska et al. (eds.), *World Congress on Medical Physics and Biomedical Engineering 2018*,
IFMBE Proceedings 68/1, https://doi.org/10.1007/978-981-10-9035-6_33

and feature extraction, followed by a conventional neural network to perform classification task. The well-known basic architecture of CNN involves a Rectified Linear Unit (ReLu), a convolution, and a pooling layer [11]. In contrast to conventional methods which require prior segmentation of tumor mass, our CNN approach does not involve region-based pre-processing step. We validate our algorithm on the same dataset which was used in previous publications [1, 2].

2 Previous Work

Various methodologies have been developed in the past years to recognize brain tumor in MRI images. These methods are ranging from classical image processing to neural network based machine learning approach. Jun Cheng et al. [2] developed a tumor classification method that consists of two phases: offline database building and online retrieval. In the offline database phase, the brain tumor images are processed in sequential steps. The steps are consisted of tumor segmentation, feature extraction, and distance metric learning. In the online learning, the input brain image will be processed similarly and compare the extracted feature with the learned distanced metrics which are stored in the online database. This method does not use neural network approach but could achieve a classification accuracy of 94.68%. On the other hand, Gawande and Mendre [12] used Deep Neural Network using autoencoders in order to classify the brain tumor. Image segmentation and feature extraction had been implemented on the image before it was processed by DNN layers. The texture and intensity based features of the image were extracted with help of Gray Level Co-occurence Matrix (GLCM) and Discrete Wavelet Transform (DWT). In the final step, DNN layers which consist of two autoencoders and one softmax layer were performed for classification. Furthermore, Pereira et al. [13] also exploring the used of Convolutional Neural Networks (CNN) with small 3×3 kernels in order to get to the deeper architecture and avoid the overfitting. They also investigated the use of intensity normalization as the pre-processing step before getting into the CNN layers. In this study, inspired by those works, we investigate and explore the implementation of deep CNN on classifying several brain tumor type diagnosis problems in order to get better accuracy result.

3 Method

3.1 Convolutional Neural Network

CNN convolution layer is a network where an image will be convolved with filters to produce feature maps. This feature maps will be forwarded to the next convolution layer to receive or extract another higher level features from the input image. Between convolution layers, non-linearity functions and down sampling operation is used to add non-linearity and reduced the dimensionality of the image respectively. Max-pooling usually used as the down sampling operation as it reduced the dimension while preserving the dominant feature in the feature maps. Just after the last convolution layer or before the first layer of the neural network, flattening layer exist to vectorize the feature maps. In the neural network or classification phase, the flatten input vector will be forwarded into the network to produce a number at each output neurons. This number tells how much an input vector is classified as a certain class. Usually a softmax activation function is used at the output layer to normalize the output sum such that all numbers at the output neuron will add up to one.

In the training phase, CNN use a learning or optimizer algorithm to update the filters at the convolution layers and weights at the neural network or fully connected layer. The learning algorithm takes a classification error or loss as an input and back propagates the error into the network to update the filters and the weights.

3.2 CNN Architecture

In this paper, we use five different architectures to test the accuracy of brain tumor classification. There are already well defined CNN architecture such as AlexNet [14], VGG16 [15], and ResNet [16] but the architecture that are implemented in this paper are much simpler that the one mentioned. The CNN architecture that are implemented in this paper are summarize in the Fig. 1a–e.

3.3 Hyper-parameters Optimization

Hyper-parameter is a parameter in deep learning process whose value can be set and tuned before the learning process. This parameter will determine the algorithm to be used in the learning process. Different model training algorithms need different hyper-parameters that also affecting the result of the learning process too. Hyper-parameter optimizer has to be chosen and tuned so that the classifier will have the most optimal way to solve the problem.

In this study, we will use ‘adam’ optimizer in the learning process which is a method for stochastic optimization by utilizing the stochastic gradient descent principle. ‘Adam’ optimizer which stands for adaptive moment estimation is chosen because of it’s advantage that can handle sparse gradients on noisy problems.

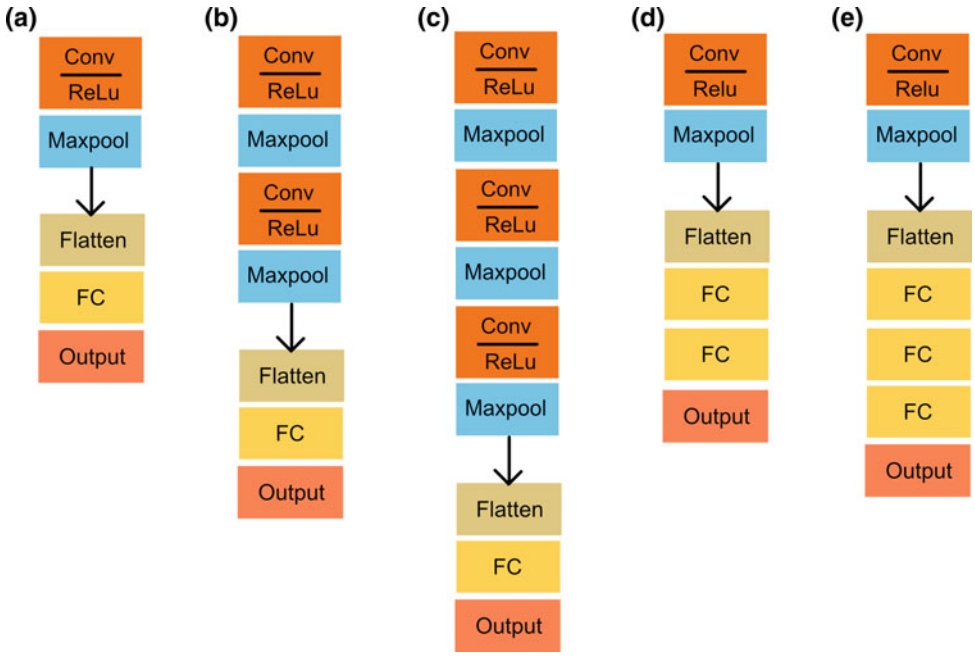

Fig. 1 **a**–**e** The proposed Convolutional Neural Network Architecture 1–5

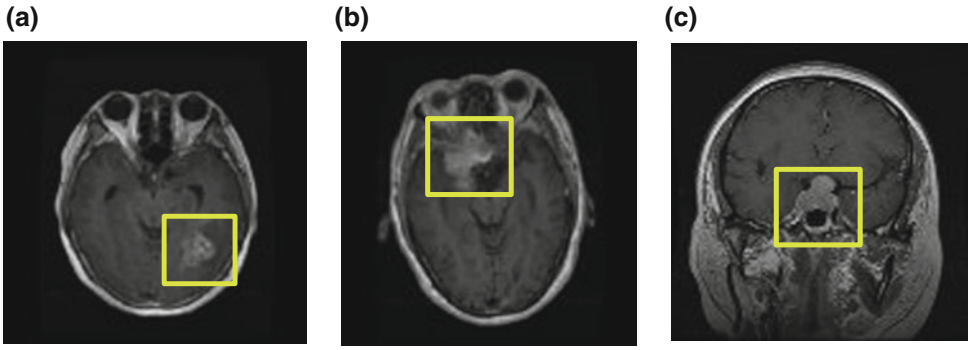

Fig. 2 **a** Glioma, **b** Meningioma, **c** Pituitary (each labeled in green)

4 Data

In this paper, our CNN is trained using 3064 T-1 weighted CE-MRI of brain tumor images. The dataset is provided by Jun Cheng and was previously used in his paper [1, 2]. This dataset consists of 708 images with glioma, 1426 images with meningioma, and 930 images with pituitary tumors. In our training phase, we equalize the amount of images that are used to train the CNN for each class or type of tumors. Out of all available images, we only used 700 images from each class where 500 of those images were used for training phase and the other 200 images were used for validation phase. The dataset was originally provided in matlab.mat format where each file stores a struct containing a label which specify the type of tumor for a particular brain image, patient ID, image data in 512×512 uint16 format, vector storing the coordinates of discrete points on tumor border, and a binary mask image with 1 s indicating tumor region. In our paper we only make use the label and image data in the.mat files therefore our brain tumor classifier is a simple CNN network which only takes image as an input. Figure 2a–c represent example of the dataset from each of the classes.

5 Result

In our experiment, the hyperparameter at each layer such as number and size of filters in the convolution layers, size of maxpooling kernel, number of neurons in the fully connected layers are held fixed. Only the depth of the architecture are varied between different architectures. The architecture and hyperparameter that are used are served in Table 1.

Table 1 Architecture 1–5

Architecture 1

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_6 (MaxPooling2	(None, 31, 31, 32)	0
flatten_6 (Flatten)	(None, 30752)	0
dense_11 (Dense)	(None, 64)	1968192
dense_12 (Dense)	(None, 3)	195

Total params: 1,969,283
Trainable params: 1,969,283
Non-trainable params: 0

Architecture 2

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_7 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_8 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_8 (MaxPooling2	(None, 14, 14, 32)	0
flatten_4 (Flatten)	(None, 6272)	0
dense_10 (Dense)	(None, 64)	401472
dense_11 (Dense)	(None, 3)	195

Total params: 411,811
Trainable params: 411,811
Non-trainable params: 0

Architecture 3

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 32)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 64)	73792
dense_2 (Dense)	(None, 3)	195

Total params: 93,379
Trainable params: 93,379
Non-trainable params: 0

Architecture 4

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_3 (MaxPooling2	(None, 31, 31, 32)	0
flatten_3 (Flatten)	(None, 30752)	0
dense_7 (Dense)	(None, 64)	1968192
dense_8 (Dense)	(None, 64)	4160
dense_9 (Dense)	(None, 3)	195

Total params: 1,973,443
Trainable params: 1,973,443
Non-trainable params: 0

Architecture 5

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
flatten_1 (Flatten)	(None, 30752)	0
dense_1 (Dense)	(None, 64)	1968192
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 3)	195

Total params: 1,977,603
Trainable params: 1,977,603
Non-trainable params: 0

Sizes of the input images that are forwarded into the network are 64 × 64. The original images are in the size of 512 × 512. This reduction is performed because of computational cost reason. All architectures are compiled without a GPU therefore to speed up the training phase smaller image size is used.

All convolution layers in the architectures use 32 filters of size 3 × 3. We use ReLu as our activation function as it already the standard activation function used in image classification task. The size of the maxpool kernel is 2 × 2 and all the fully connected layer (called 'dense' in keras) use 64 neurons.

Finally, there are 3 neurons in the output layer since we try to classify an image with three types of brain tumors (glioma, meningioma, and pituitary). The activation functions that are used at the output layer are softmax so that all three output neurons are summed up to one.

Based on Table 1, each of architecture produces different numbers of params and features depending on the depth of the convolution layer and the fully connected network. The

architecture with deeper layers of convolution will have fewer numbers of the trainable params.

Implementation of the above architectures will produce four parameter values that will describe the success of the classifier model in classifying the input image. The four parameter values are loss and accuracy from the training set and validation set. Accuracy is defined as the percentage of the correct guesses by the classifier either for the training set input or the validation set input. Loss is defined as feasible error that represents the price paid for inaccuracy of predictions in classification problem. We use cross-entropy method for loss calculation. The cross-entropy loss calculation can be represented in the mathematical form [17] below:

$$H(y,\hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i \qquad (1)$$

y_i represents the result of the classifier output from class i. While $\hat{y}_i$ represents the expected output from class i. The classification accuracy of each architecture are presented in Figs. 3, 4, 5, 6 and 7.

Based on Figs. 3, 4, 5, 6 and 7, the values of loss and accuracy vary according to the implementation of the architecture. The classifier model is said to be 'good fit' if the accuracy of training set and validation set tend to increase for every epoch of training. However, if the accuracy of validation set tends to decrease while the accuracy of training set increases, then the classifier model is estimated to have overfitting. Overfitting happens when the model learns the detail and noise in the training data thus reducing its ability to generalize other datasets well [18].

By looking at the result of each architectures, we could see that all architecture's validation loss shows an increasing trend with respect to the number of epoch except for architecture 2. This indicates that architecture 2 is the best architecture out of the five architectures at generalizing the brain tumor images. The decreasing pattern in the validation loss indicates that using the available training images, architecture 2 could classify the unknown images in the

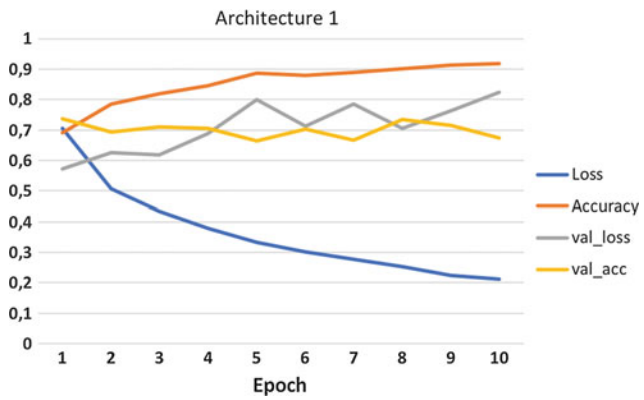

Fig. 3 Accuracy and loss of architecture 1

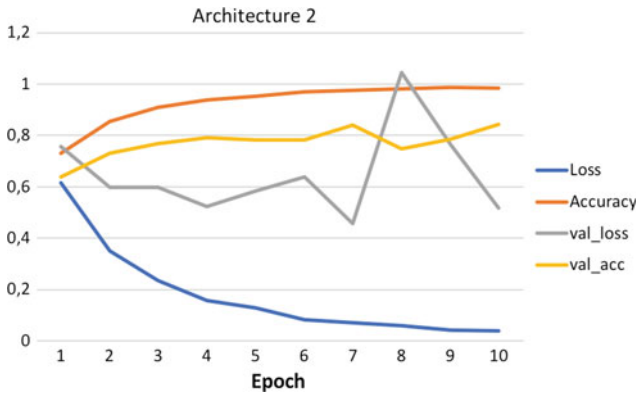

Fig. 4 Accuracy and loss of architecture 2

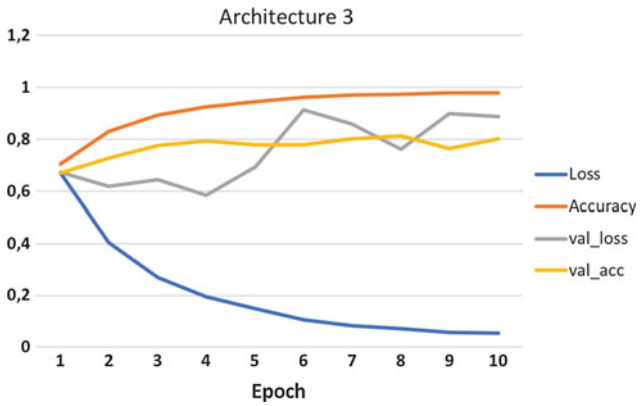

Fig. 5 Accuracy and loss of architecture 3

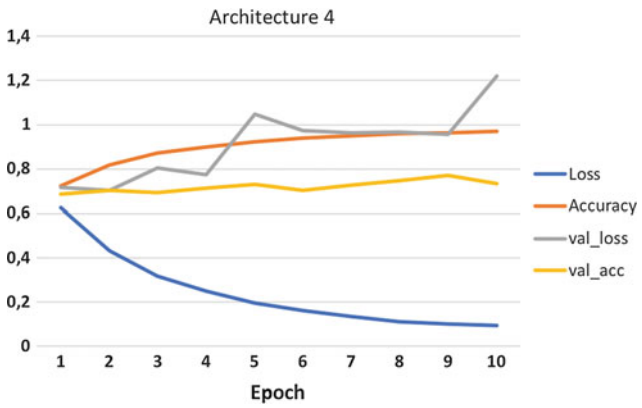

Fig. 6 Accuracy and loss of architecture 4

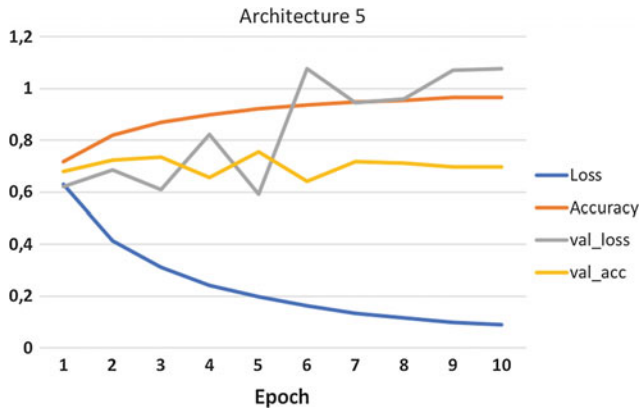

Fig. 7 Accuracy and loss of architecture 5

validation set with medium performance. We conclude a medium performance since the validation loss does not show a perfect decreasing pattern as the number of epoch increase. In the last epoch, architecture 2 could achieve a validation accuracy of 84.19%.

Based on the best architectural choice on the previous part, we try to vary the number of the filter in the convolution layer into 64 filters and 128 filters we called it as architecture 6 and architecture 7 respectively. In this experiment, we will identify the effect of the filter numbers in the convolution layer on the accuracy of the validation set. The architecture and hyper parameters that are used are served in the Table 2.

After implementation of those architectures, the classification accuracy of those architectures are presented in Figs. 8 and 9.

From the result, the architecture that has the highest validation accuracy is still architecture 2 with 32 filters in the convolution layers. Although an increase in the number of filters in the convolution layer does not necessarily contribute to better CNN performance, in our case the number of filter in a convolution layer does influence the accuracy of the classifier. We stick or recommend the use of architecture 2 to classify a brain tumor since it has the highest validation accuracy.

Table 2 Architecture 6–7

Architecture 6

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 64)	1792
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 64)	0
conv2d_2 (Conv2D)	(None, 29, 29, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_1 (Dense)	(None, 64)	802880
dense_2 (Dense)	(None, 3)	195

Total params: 841,795
Trainable params: 841,795
Non-trainable params: 0

Architecture 7

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 128)	3584
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 128)	0
conv2d_2 (Conv2D)	(None, 29, 29, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 128)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 64)	1605696
dense_2 (Dense)	(None, 3)	195

Total params: 1,757,059
Trainable params: 1,757,059
Non-trainable params: 0

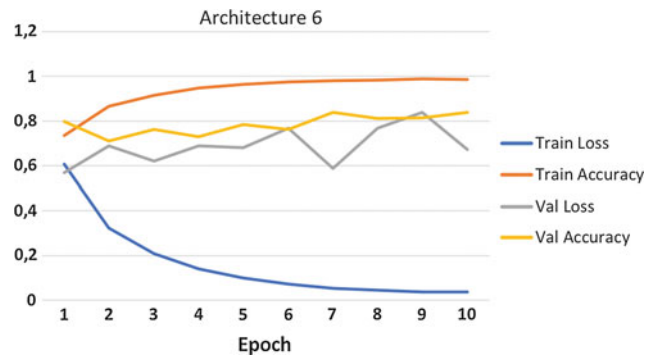

Fig. 8 Accuracy and loss of architecture 6

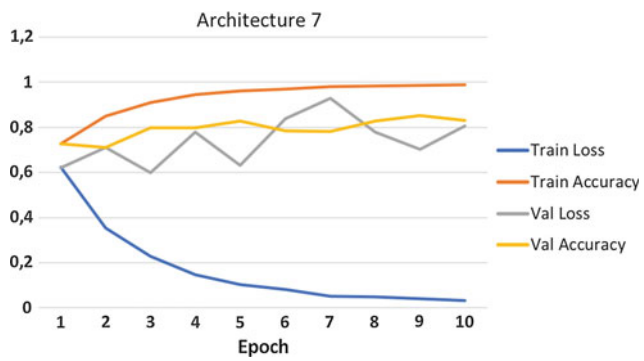

Fig. 9 Accuracy and loss of architecture 7

6 Conclusion

In this paper, we introduced CNN to automatically classify the three most common types of brain tumor, i.e. the Glioma, Meningioma, and Pituitary; without requiring region-based pre-processing steps. We identified an optimal CNN architecture (architecture 2) consisting of 2 layers of convolution, activation (ReLu), and maxpool, followed by one hidden layer of 64 neurons. Architecture 2 is the only architecture that show a consistently decreasing pattern in the validation loss as the number of epoch increases, leading to the highest validation accuracy out of all five architectures. The training and validation accuracies of architecture 2 at best is 98.51% and 84.19%, respectively. These figures, although somewhat lower, are still comparable to the accuracies of conventional algorithms with region-based pre-processing, which performed at 71.39–94.68% [1, 2]. For future work, we consider to include color balancing step into our CNN, to improve classification accuracy of textured brain MRI pixels [19]. Our algorithm may be implemented in as a simple supportive tool for medical doctor in classifying brain tumor.

Disclosure The authors declare that they have no conflict of interest.

References

1. Jun Cheng. (2017). *Brain Tumor Dataset* (Version 5). Retrieved from https://doi.org/10.6084/m9.figshare.1512427.v5
2. Cheng, J., Huang, W., Cao, S., et al. (2015). Classification via Tumor Region Augmentation and Partition. PL*oS One*, *10*(10).
3. Varade, A. A., Ingle, K. S. (2017). Brain MRI Classification Using PNN and Segmentation Using K Means Clustering, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, *6*(5), 6181–6188.
4. Cheng, J., Yang, W., Huang, M., et al. (2016). Retrieval of Brain Tumors by Adaptive Spatial Pooling and Fisher Vector Representation. *PLoS One*, *11*(6).
5. Lavanyadevi, R., Machakowsalya, M., Nivehitha, J., et al. (2017), Brain tumor classification and segmentation in MRI images using PNN, *IEEE Xplore*.
6. Siegel, R. L., Miller, K. D., Jemal, A. (2016). Cancer statistics, 2016. *CA: A Cancer Journal for Clinicans*, *66*(1), 07–30.
7. Quinn, T., Ostrom, M.A., Gittleman, H., et al. (2016). American Brain Tumor Association Adolescent and Young Adult Primary Brain and Central Nervous System Tumors Diagnosed in the United States in 2008–2012. *Neuro-Oncology*, *18*(1), i1–i50.
8. Gordillo, N., Montseney, E., Sobrevilla, P. (2013). State of the art survey on MRI brain tumor segmentation. *Magnetic Resonance Imaging*, *31*(8), 1426–1438.
9. Kaur, H., Sharma, R. (2016). A Survey on Techniques for Brain Tumor Segmentation from MRI. *IOSR Journal of Electronics and Communication Engineering*, *11*(5), 01–05.
10. Hasan, M. H., Meziane, F., et al. (2016). Segmentation of Brain Tumors in MRI Images Using Three-Dimensional Active Contour without Edge. *Symmetry*, *8*(11), 132.
11. Wu, J. (2017). Introduction to Convolutional Neural Networks. National Key Lab for Novel Software Technology Nanjing University, China.
12. He, K., Zhang, X., Ren, S., et al. (2015). Deep residual learning for Image Recognition. *CVPR*, 770–778.
13. Browniee, J. (2016, March 21). Overfitting and Underfitting With Machine Learning Algorithms. Retrieved from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.
14. Zacharaki, E. I., Wang, Sumei., Chawla, S., et al. (2009). Classification of Brain Tumor Type and Grade Using MRI Texture and Shape in a Machine Learning Scheme.*Magnetic Resonance in Medicine, 62*(6), 1609–1618.
15. Kingma, D. P., Ba, J. (2014). Adam: A Method for Stochastic Optimization, *arXiv preprint* arXiv:1412.6980.
16. Gawande, S. S., Mendre, V. (2017). Brain Tumor Diagnosis Using Deep Neural Network (DNN). International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 5(5).
17. Simonyan, K., Zisserman, A. (2015). Very Deep Convolution Networks for Large-Scale Image Recognition, *ICLR 2015*.
18. Pereira, S., Pinto, A., Alves, V., et al. (2016). Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images. *IEEE Transactions on Medical* Imaging, *35*(5), 1240–1251.
19. DiPietro, Rob. (2016, May 2). A Friendly Introduction to Cross-Entropy Loss. Retrieved from https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/.