Brain Tumor Detection Using a Convolutional Neural Network (CNN)

By

Veer Pal Singh

Introduction:

Magnetic resonance imaging (MRI) is the imaging technique used to diagnosing brain tumor disease. Early diagnosis of brain tumors is an essential task in medical work to find out whether the tumor can potentially become cancerous.

Deep learning is a handy and efficient method for image classification. Deep learning has been widely applied in various fields including medical imaging, because its application does not require the reliability of an expert in the related field, but requires the amount of data and diverse data to produce good classification results.

Convolutional Neural Network (CNN) is the deep learning technique to perform image classification. In this paper, we compared two model CNN find the best model CNN to classify tumors in Brain MRI Image and at the end, we have trained CNN and obtained a prediction accuracy of up to 91%.

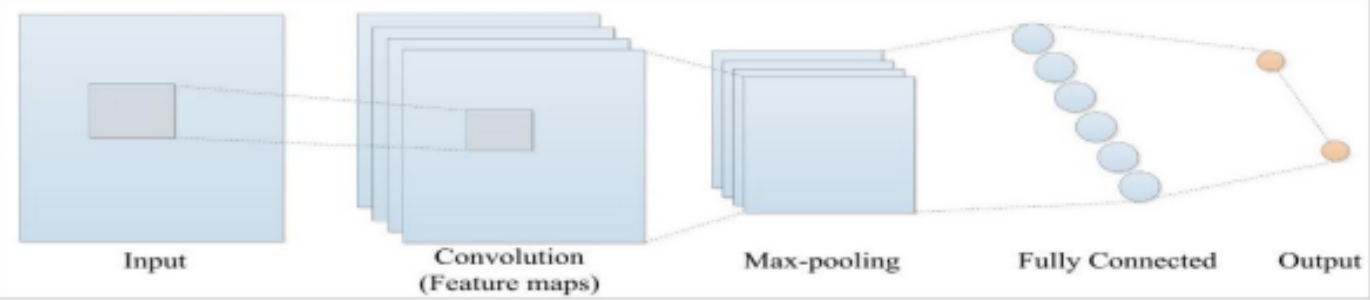
Dataset:

The dataset contains 2 folders: yes and no which contains 253 Brain MRI Images. The folder yes contains 155 Brain MRI Images that are tumorous and the folder no contains 98 Brain MRI Images that are non-tumorous. You can find it here.

Convolutional Neural Network

CNN is a neural network that aims to process data that has a grid structure. Convolution is an operation in convolution layer that is based on a linear algebra operation that multiplies matrix of the filter in the image to be processed [9]. The convolution layer is the primary layer that is most important to use. Another type of layer that is commonly used is the pooling layer, which is a layer that is used to take the maximum value or the average value of the pixel portions of the image.

CNN has the ability to learn complicated features by forming a feature map. The convolution layer kernel is wrapped around the input sample to calculate several feature maps. Features are detected from input samples than represented by small boxes on the feature map. These maps are forwarded to the maximum collection layer, which preserves relevant features and discards the rest. The features of the max-pooling layer are converted to a one-dimensional feature vector in the fully connected layer, which is then used to calculate the output probability. The configuration of CNN is shown in Figure



CNN architecture

Convolution Layer

Convolution Layer is the core layer in the CNN method which aims to extract features from the input. Convolution performs linear transformations of input data without changing spatial information in the data. Convolution kernels are determined from the weight of the layer so that the convolution kernels can process the input data training on CNN.

Subsampling Layer

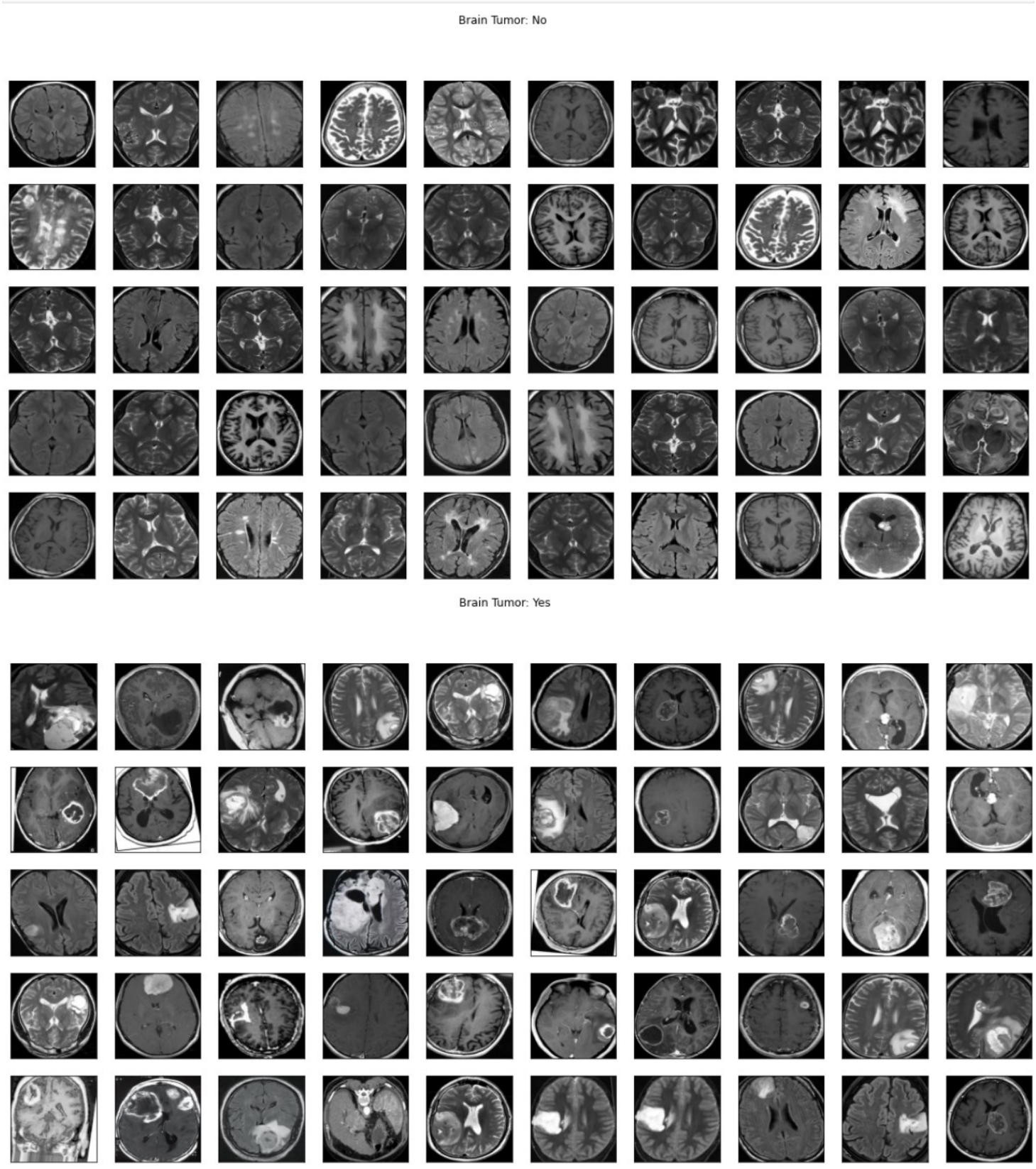
Subsampling aims to reduce the size of image data and to increase the invariance of feature positions. CNN uses Max Pooling as a subsampling method. The way Max Pooling works is to divide the output of the convolution layer into several smaller grids and then take the maximum value from each grid to produce a smaller image matrix. With a small image size will make it easier to process the next convolution layer.

Fully Connected Layer

The Fully Connected Layer changes the dimensions of the data so that it can be classified linearly. In the convolution layer, each neuron must be transformed into one-dimensional data before being inserted into another layer that is connected as a whole[11]. This process is caused by data losing its spatial information and at the end of the Fully Connected Layer network is applied.

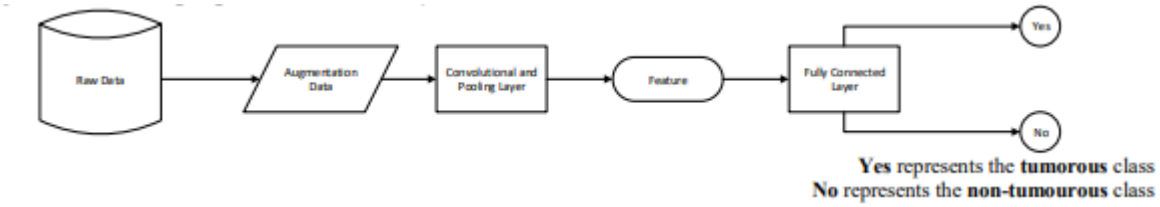
Method

The dataset used in this study is Brain MRI Images for Brain Tumor Detection obtained from kaggle.com. The dataset consists of 253 images grouped into 2 groups, 155 brain images that have tumors, and 98 brain images that do not have tumors. Figure shows images in the dataset.



Proposed Method

This paper uses CNN for the automatic detection of brain tumors. This study uses input images labeled (yes/no) from the raw data and then uses these patterns to distinguish between tissues that do not contain tumors and those that contain tumors. CNN was trained to use 2065 sample images consisting of 1085 images containing tumors and 980 not containing tumors. Therefore, the proposed system is illustrated in Figure 3 method proposed in this study.



Method proposed

Data Augmentation

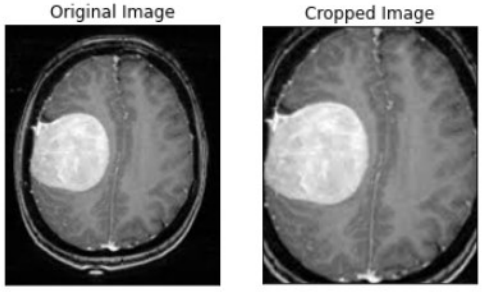
The amount of data in the dataset is not enough to be used as training data for CNN. Therefore the augmentation method is used to overcome the imbalance of issues. Augmentation is an algorithm that can utilize statistical data information and form an integrated model. This algorithm can produce a number of two-dimensional images of various poses and sizes. The application of augmentation to obtain image variants can improve the accuracy of CNN segmentation[12]. In this paper, each image with a tumor is segmented into 6 images, and an image with no tumor is segmented into 9 images. After data augmentation, the dataset consists of 1085 samples containing tumors (53%) and 980 samples not containing tumors (47%), bringing a total of 2065 images.

Image Pre-processing

Pre-processing performs to create smooth training because there are different variants of intensity, contrast, and size in images [13]. Input image will be processed into the first pre-process that is the process of wrapping and cropping. In wrapping, the input image is checked against the edge of the main object in the image. From the edge of the image, the maximum edge is determined so that when the results of cropping, the object in the image remains intact. After cropping, resize the image to shape (240, 240, 3) = (image width, image height, number of channels) because the images in the dataset have different sizes. Apply normalization: to scale pixel values to the range 0-1 to facilitate the learning process

In order to crop the part that contains only the brain of the image, I used a cropping technique to find the extreme top, bottom, left and right points of the brain. You can read more about it here Finding extreme points in contours with OpenCV.

```
ex_img = cv2.imread('yes/Y1.jpg')
ex_new_img = crop_brain_contour(ex_img, True)
```



Split the data:

Split X and y into training, validation (development) and validation sets.

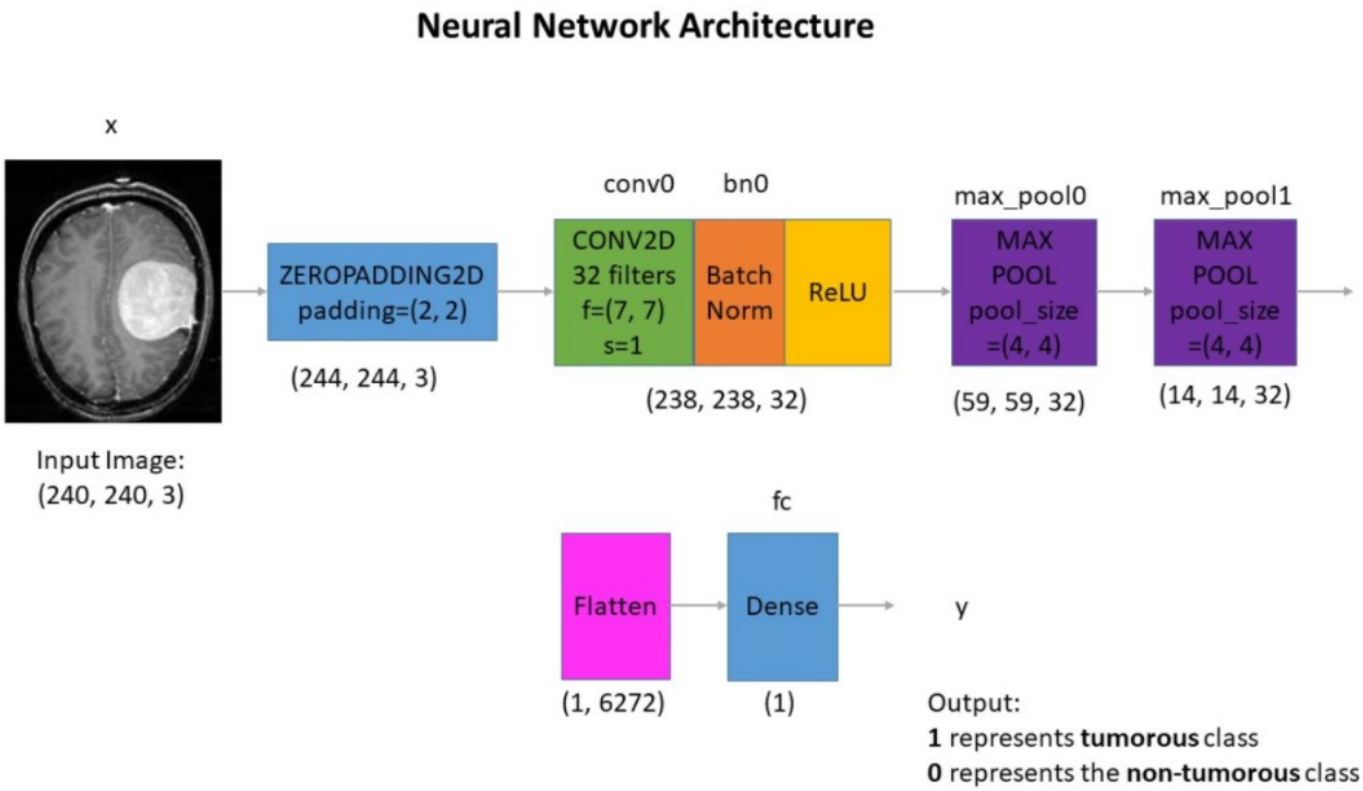
Let's use the following way to split:

1. 70% of the data for training.
2. 15% of the data for validation.
3. 15% of the data for testing.

```
number of training examples = 1444
number of development examples = 310
number of test examples = 310
X_train shape: (1444, 240, 240, 3)
Y_train shape: (1444, 1)
X_val (dev) shape: (310, 240, 240, 3)
Y_val (dev) shape: (310, 1)
X_test shape: (310, 240, 240, 3)
Y_test shape: (310, 1)
```

Build the model

Let's build a convolutional neural network model:



Define the image shapes:

```
Model: "BrainDetectionModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 zero_padding2d (ZeroPadding  (None, 244, 244, 3)      0
 2D)

 conv0 (Conv2D)              (None, 238, 238, 32)      4736

 bn0 (BatchNormalization)    (None, 238, 238, 32)      128

 activation (Activation)     (None, 238, 238, 32)      0

 max_pool0 (MaxPooling2D)    (None, 59, 59, 32)        0

 max_pool1 (MaxPooling2D)    (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 fc (Dense)                  (None, 1)                 6273

=================================================================
Total params: 11,137
Trainable params: 11,073
Non-trainable params: 64
_________________________________________________________________
```

Compile the Model:

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
# tensorboard
log_file_name = f'brain_tumor_detection_cnn_{int(time.time())}'
tensorboard = TensorBoard(log_dir=f'logs/{log_file_name}')
```

```
# checkpoint
# unique file name that will include the epoch and the validation (development) accuracy
filepath="cnn-parameters-improvement-{epoch:02d}-{val_accuracy:.2f}"
# save the model with the best validation (development) accuracy till now
checkpoint = ModelCheckpoint("models/{}.model".format(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max'))
```

Train the model

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/10
46/46 [==============================] - ETA: 0s - loss: 0.8443 - accuracy: 0.6060INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
1-0.59.model\assets
46/46 [==============================] - 98s 2s/step - loss: 0.8443 - accuracy: 0.6060 - val_loss: 0.6733 - val_accuracy: 0.5903
Epoch 2/10
46/46 [==============================] - ETA: 0s - loss: 0.4819 - accuracy: 0.7722INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
2-0.71.model\assets
46/46 [==============================] - 84s 2s/step - loss: 0.4819 - accuracy: 0.7722 - val_loss: 0.6200 - val_accuracy: 0.7065
Epoch 3/10
46/46 [==============================] - ETA: 0s - loss: 0.4182 - accuracy: 0.8193INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
3-0.68.model\assets
46/46 [==============================] - 78s 2s/step - loss: 0.4182 - accuracy: 0.8193 - val_loss: 0.5786 - val_accuracy: 0.6774
Epoch 4/10
46/46 [==============================] - ETA: 0s - loss: 0.3575 - accuracy: 0.8511INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
4-0.80.model\assets
46/46 [==============================] - 78s 2s/step - loss: 0.3575 - accuracy: 0.8511 - val_loss: 0.5183 - val_accuracy: 0.8032
Epoch 5/10
46/46 [==============================] - ETA: 0s - loss: 0.3441 - accuracy: 0.8532INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
5-0.63.model\assets
46/46 [==============================] - 79s 2s/step - loss: 0.3441 - accuracy: 0.8532 - val_loss: 0.5925 - val_accuracy: 0.6290
Epoch 6/10
46/46 [==============================] - ETA: 0s - loss: 0.3498 - accuracy: 0.8442INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
6-0.82.model\assets
46/46 [==============================] - 76s 2s/step - loss: 0.3498 - accuracy: 0.8442 - val_loss: 0.4162 - val_accuracy: 0.8226
Epoch 7/10
46/46 [==============================] - ETA: 0s - loss: 0.3102 - accuracy: 0.8760INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
7-0.85.model\assets
46/46 [==============================] - 76s 2s/step - loss: 0.3102 - accuracy: 0.8760 - val_loss: 0.3781 - val_accuracy: 0.8516
Epoch 8/10
46/46 [==============================] - ETA: 0s - loss: 0.2657 - accuracy: 0.8975INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
8-0.83.model\assets
46/46 [==============================] - 75s 2s/step - loss: 0.2657 - accuracy: 0.8975 - val_loss: 0.3990 - val_accuracy: 0.8323
Epoch 9/10
46/46 [==============================] - ETA: 0s - loss: 0.3541 - accuracy: 0.8518INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
9-0.86.model\assets
46/46 [==============================] - 77s 2s/step - loss: 0.3541 - accuracy: 0.8518 - val_loss: 0.3501 - val_accuracy: 0.8645
Epoch 10/10
46/46 [==============================] - ETA: 0s - loss: 0.2526 - accuracy: 0.9065INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-1
0-0.88.model\assets
```

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=5, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/5
46/46 [==============================] - ETA: 0s - loss: 0.1361 - accuracy: 0.9515INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
1-0.90.model\assets
46/46 [==============================] - 85s 2s/step - loss: 0.1361 - accuracy: 0.9515 - val_loss: 0.2678 - val_accuracy: 0.8968
Epoch 2/5
46/46 [==============================] - ETA: 0s - loss: 0.1065 - accuracy: 0.9695INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
2-0.91.model\assets
46/46 [==============================] - 83s 2s/step - loss: 0.1065 - accuracy: 0.9695 - val_loss: 0.2631 - val_accuracy: 0.9097
Epoch 3/5
46/46 [==============================] - ETA: 0s - loss: 0.1194 - accuracy: 0.9654INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
3-0.80.model\assets
46/46 [==============================] - 78s 2s/step - loss: 0.1194 - accuracy: 0.9654 - val_loss: 0.4825 - val_accuracy: 0.8000
Epoch 4/5
46/46 [==============================] - ETA: 0s - loss: 0.0892 - accuracy: 0.9785INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
4-0.90.model\assets
46/46 [==============================] - 82s 2s/step - loss: 0.0892 - accuracy: 0.9785 - val_loss: 0.2851 - val_accuracy: 0.9000
Epoch 5/5
46/46 [==============================] - ETA: 0s - loss: 0.0902 - accuracy: 0.9765INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-0
5-0.75.model\assets
46/46 [==============================] - 79s 2s/step - loss: 0.0902 - accuracy: 0.9765 - val_loss: 0.5961 - val_accuracy: 0.7548
Elapsed time: 0:6:52.1
```

Plot Loss & Accuracy

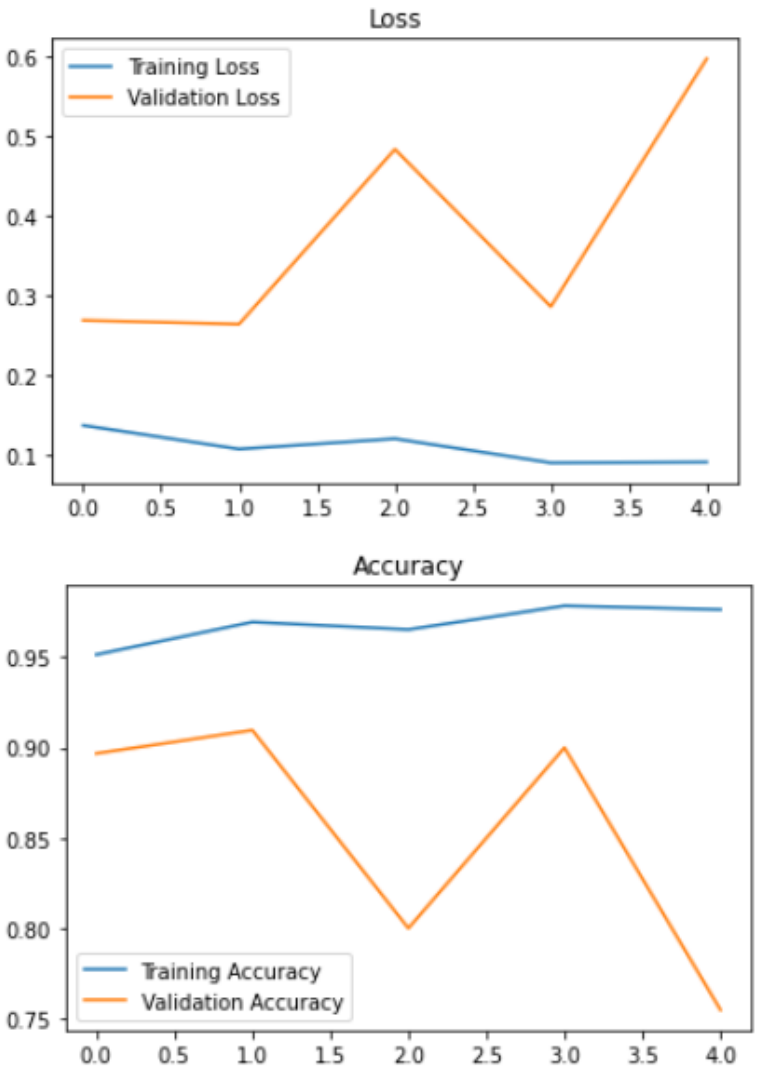
```
def plot_metrics(history):

    train_loss = history['loss']
    val_loss = history['val_loss']
    train_acc = history['accuracy']
    val_acc = history['val_accuracy']

    # Loss
    plt.figure()
    plt.plot(train_loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.title('Loss')
    plt.legend()
    plt.show()

    # Accuracy
    plt.figure()
    plt.plot(train_acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.title('Accuracy')
    plt.legend()
    plt.show()
```

```
plot_metrics(history)
```



Results

Let's experiment with the best model (the one with the best validation accuracy):

Concretely, the model at the 23rd iteration with validation accuracy of 91%

Accuracy of the best model on the testing data:

```
Test Loss = 0.2682258188724518
Test Accuracy = 0.8935483694076538
```

F1 score for the best model on the testing data:

```
F1 score: 0.878892733564014
```

Results Interpretation

Let's remember the percentage of positive and negative examples:

```
Number of examples: 2064
Percentage of positive examples: 52.51937984496124%, number of pos examples: 1084
Percentage of negative examples: 47.48062015503876%, number of neg examples: 980

Training Data:
Number of examples: 1444
Percentage of positive examples: 52.908587257617725%, number of pos examples: 764
Percentage of negative examples: 47.091412742382275%, number of neg examples: 680
Validation Data:
Number of examples: 310
Percentage of positive examples: 47.096774193548384%, number of pos examples: 146
Percentage of negative examples: 52.903225806451616%, number of neg examples: 164
Testing Data:
Number of examples: 310
Percentage of positive examples: 56.12903225806452%, number of pos examples: 174
Percentage of negative examples: 43.87096774193548%, number of neg examples: 136
```

Conclusion:

Now, the model detects brain tumor with:

Convolutional Neural Networks are good enough to diagnose brain tumors on MRI images. This study resulted in 88.7% accuracy on the test set. 0.88 f1 score on the test set and a loss value of 0.23264. The number of convolution layers affects the quality of classification, more convolution layers increase the accuracy results, but more number of convolution layers will require more time for training. The process of image augmentation can improve the variants of existing datasets, thereby increasing the classification results. Finally, for future suggestions, more images can be used to improve classification results. Future studies can also classify certain types of tumors.

Performance Table:

	Validation set	Test set
Accuracy	91%	89%
F1 score	0.91	0.88

References:

1. "Penyakit Kanker di Indonesia Berada Pada Urutan 8 di Asia Tenggara dan Urutan 23 di Asia." [Online]. Available: http://p2p.kemkes.go.id/penyakit-kanker-di-indonesia-berada-padaurutan-8-di-asia-tenggara-dan-urutan-23-di-asia/.2.

2. S. EDY, W. I, and W. A, "Clinical Characteristics and Histopathology of Brain Tumor at Two Hospitals in Bandar Lampung," Fac. Med. Lampung Univ., vol. 69, pp. 48–56, 2014. 3.

3. H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty, and A.-B. M. Salem, "Classification using deep learning neural networks for brain tumors," Futur. Comput. Informatics J., vol. 3, no. 1, 2nd International Conference on Engineering and Applied Sciences (2nd InCEAS) IOP Conf. Series: Materials Science and Engineering 771 (2020) 012031 IOP Publishing doi:10.1088/1757-899X/771/1/012031 6 pp. 68–71, 2018. 4.

4. A. Wadhwa, A. Bhardwaj, and V. Singh Verma, "A review on brain tumor segmentation of MRI images," Magn. Reson. Imaging, vol. 61, no. January, pp. 247–259, 2019. [5.

5. Parveen and A. Singh, "Detection of brain tumor in MRI images, using combination of fuzzy cmeans and SVM," 2nd Int. Conf. Signal Process. Integr. Networks, SPIN 2015, pp. 98–102, 2015. 6.

6. M. H. Avizenna, I. Soesanti, and I. Ardiyanto, "Classification of Brain Magnetic Resonance Images Based on Statistical Texture," Proc. - 2018 1st Int. Conf. Bioinformatics, Biotechnol. Biomed. Eng. BioMIC 2018, vol. 1, pp. 1–5, 2019. 7.

7. S. Dabeer, M. M. Khan, and S. Islam, "Cancer diagnosis in histopathological image: CNN based approach," Informatics Med. Unlocked, vol. 16, no. May, p. 100231, 2019. 8.

8. A. Işin, C. Direkoğlu, and M. Şah, "Review of MRI-based Brain Tumor Image Segmentation Using Deep Learning Methods," Procedia Comput. Sci., vol. 102, no. August, pp. 317–324, 2016. 9.

9. K. P. Danukusumo, Pranowo, and M. Maslim, "Indonesia ancient temple classification using convolutional neural network," ICCREC 2017 - 2017 Int. Conf. Control. Electron. Renew. Energy, Commun. Proc., vol. 2017 -January, pp. 50–54, 2017. 10.

10. S. Hussain, S. M. Anwar, and M. Majid, "Segmentation of glioma tumors in brain using deep convolutional neural network," Neurocomputing, vol. 282, pp. 248–261, 2018. [11.

11. R. Refianti, A. B. Mutiara, and R. P. Priyandini, "Classification of melanoma skin cancer using convolutional neural network," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 3, pp. 409–417, 2019. 12.

12. Z. Tang, K. Chen, M. Pan, M. Wang, and Z. Song, "An augmentation strategy for medical image processing based on Statistical Shape Model and 3D Thin Plate Spline for deep learning," IEEE Access, vol. 7, pp. 1–1, 2019. 13.

13. T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," Array, vol. 4, no. July, p. 100004, 2019.