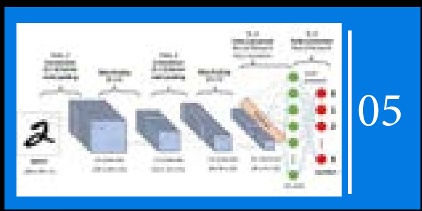
Pepgra

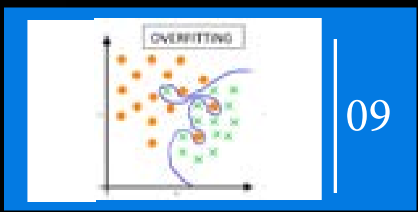
OVERCOMING 'OVERFITTING' CHALLENGES IN BRAIN TUMOUR DETECTION AND ENHANCING ACCURACY WITH DATA AUGMENTATION

HIGHLIGHTS



Convolutional Neural Network (CNN)

Model for Data Augmentation



Methods to Reduce Overfitting

Pepgra
Drives Innovation

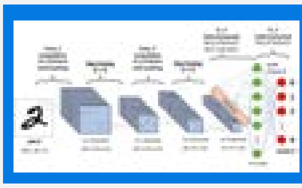
PREFACE

A brain tumor is an abnormal growth of cells inside the brain or skull; some are benign, others malignant. Tumors can grow from the brain tissue itself (primary), or cancer from elsewhere in the body can spread to the brain (metastasis). Treatment options vary depending on the tumour type, size and location. Treatment goals may be curative or focus on relieving symptoms. Many of the 120 types of brain tumors can be successfully treated. New therapies are improving the life span and quality of life for many people.

Dataset

Our dataset which we have collected contains about 253 MRI scans. 155 images of those are tumorous and 98 are non-tumorous.

HIGHLIGHTS

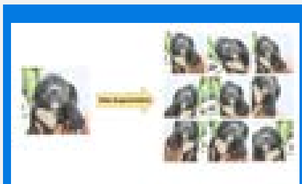


05

Convolutional Neural Network (CNN)

09

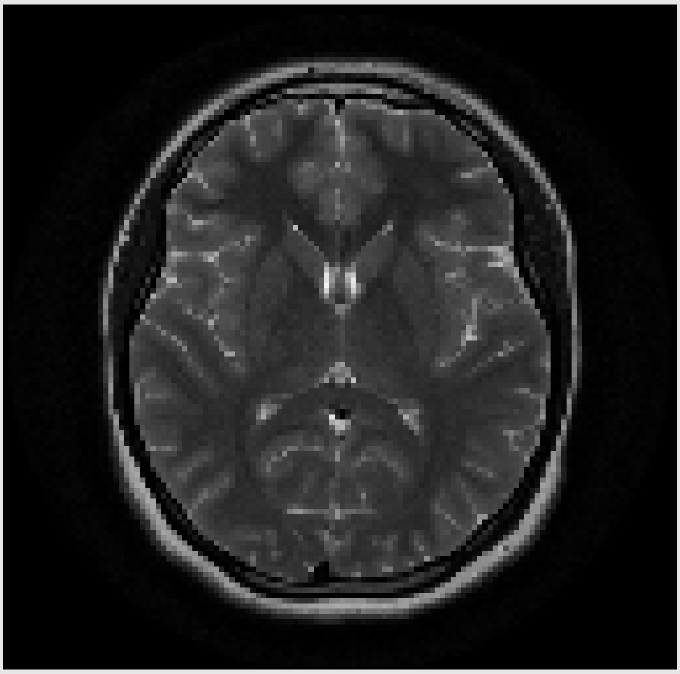
Methods to Reduce Overfitting



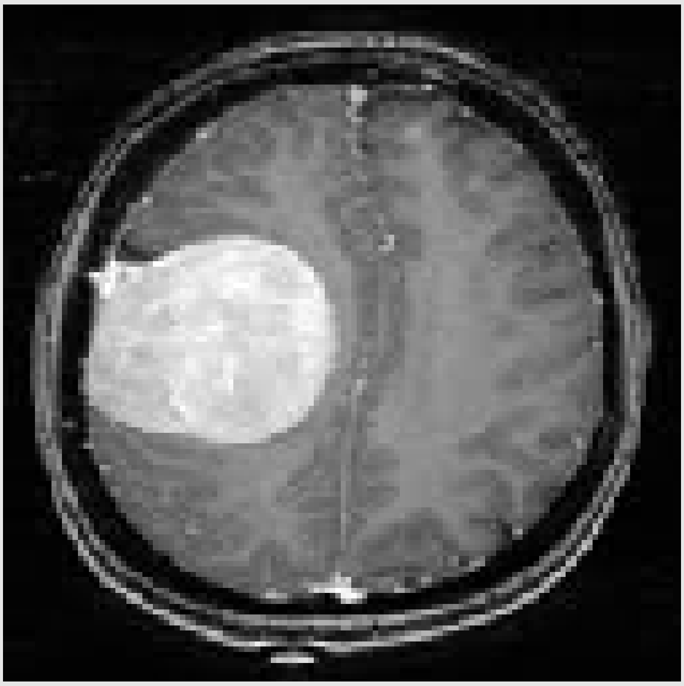
09

Model for Data Augmentation

Normal MRI scan of a Brain -



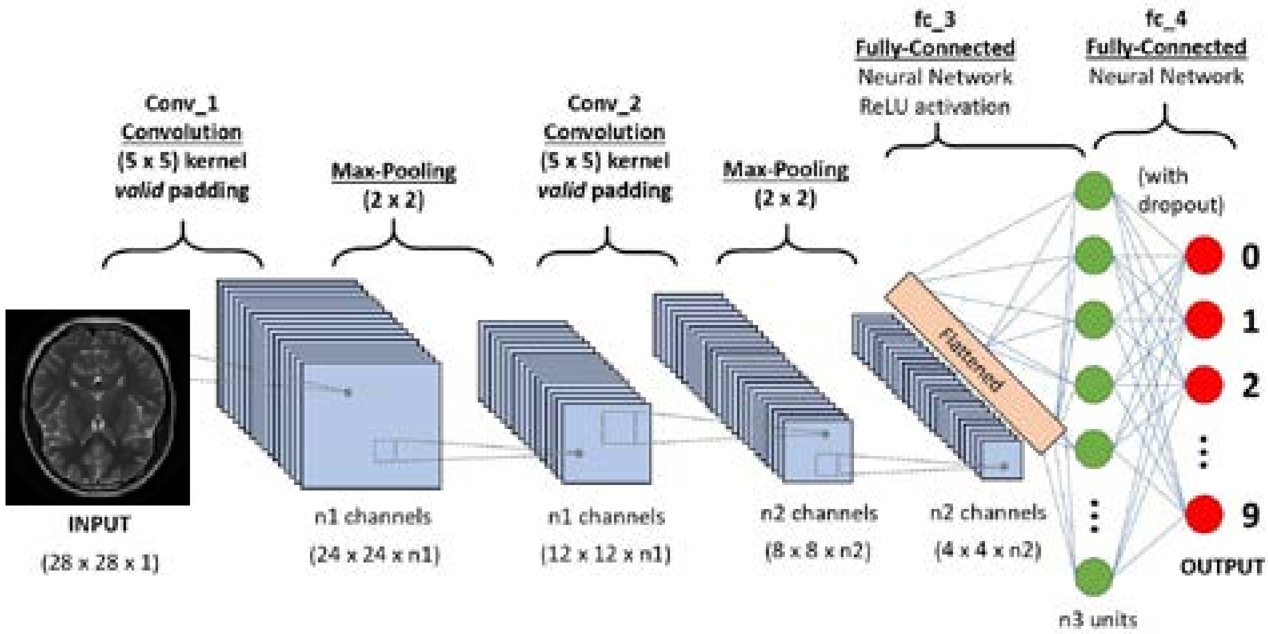
Tumorous brain –



We will be using CNN (Convolutional neural network) with different layers combination, with and without drop-outs, different optimizers and their learning rates.

Convolutional Neural Network (CNN):

The enormity of Computer vision's usage is high due to its applications in various industries solving complex problems, reducing cost of resources etc. Convolution Neural Network (CNN) is a widely used architecture in image classification. Some of the applications are Face recognition, Documents analysis, recommender systems, Prediction modelling, Image classification, & fault diagnosis. CNN being a neural network model has parameters like Epoch, Batch_size, weights, learning rate etc.



WHY CNN?

CNN can learn useful features from the image so there is no need for performing feature extraction. The number of neurons required to train the model is less since the dimensionality of the images is reduced after convolution.

Our target variable is being coded as 0's and 1's –

```
data_target
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
X_new.shape
```

```
(253, 32, 32, 3)
```

For modeling we will be using Keras Package with TensoFlow as backend. Our Data will be split into Training, validation and testing data. The data split code is given below.

```
(X_train, y_train), (X_test, y_test) = (X_new[:190],data_target[:190]) , (X_new[190:] , data_target[190:])
(X_valid , y_valid) = (X_test[:63], y_test[:63])
```

With the help of keras module we will use convolutional neural network as Conv2D. This model will be sequential, there will be 2 layer of Conv2D with MaxPooling2D and Drop-out.

Main Parameters:

Kernel size – (3,3)
Activation –

- Relu for CNN layers
- Sigmoid for final layer (As our model is binary class output)

Dropout - different dropouts at each layer
Optimizer – **ADAM**
Pooling size – Max pooling size is 2
Metrics – Accuracy
Epochs – 200
Batch size – 64
Validation data – YES
Learning Rate – different learning rate tested (lr- 0.1, 0.01, 0.01)

“ **LOSS** –
Binary cross entropy

Model Summary:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        3904
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 32)        41504
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1049088
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,095,009
Trainable params: 1,095,009
Non-trainable params: 0
```

“ **ADAM** - An algorithm for first-order gradient-based optimization of stochastic objective functions.

Model Compilation:

```
model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam(), metrics=['acc'])
```

Model Fit:

```
history = model.fit(X_train, y_train,batch_size=64,epochs=200,validation_data=(X_valid, y_valid),)
```

```
Train on 190 samples, validate on 63 samples
Epoch 1/200
190/190 [==============================] - 3s 16ms/sample - loss: 0.7816 - acc: 0.8158 - val_loss: 0.8300 - val_acc: 0.0000e+00
Epoch 2/200
190/190 [==============================] - 0s 248us/sample - loss: 0.5649 - acc: 0.8158 - val_loss: 0.7980 - val_acc: 0.0000e+00
```

Training:

```
190/190 [==============================] - 0s 222us/sample - loss: 0.0034 - acc: 1.0000 - val_loss: 3.1624 - val_acc: 0.6508
Epoch 194/200
190/190 [==============================] - 0s 223us/sample - loss: 0.0025 - acc: 1.0000 - val_loss: 3.3836 - val_acc: 0.6349
Epoch 195/200
190/190 [==============================] - 0s 251us/sample - loss: 0.0018 - acc: 1.0000 - val_loss: 3.5568 - val_acc: 0.6190
Epoch 196/200
190/190 [==============================] - 0s 210us/sample - loss: 0.0031 - acc: 1.0000 - val_loss: 4.2131 - val_acc: 0.6032
Epoch 197/200
190/190 [==============================] - 0s 217us/sample - loss: 0.0012 - acc: 1.0000 - val_loss: 5.0656 - val_acc: 0.5079
Epoch 198/200
190/190 [==============================] - 0s 202us/sample - loss: 0.0015 - acc: 1.0000 - val_loss: 5.4731 - val_acc: 0.4921
Epoch 199/200
190/190 [==============================] - 0s 216us/sample - loss: 0.0034 - acc: 1.0000 - val_loss: 4.8045 - val_acc: 0.5238
Epoch 200/200
190/190 [==============================] - 0s 216us/sample - loss: 0.0020 - acc: 1.0000 - val_loss: 4.0138 - val_acc: 0.6190
```

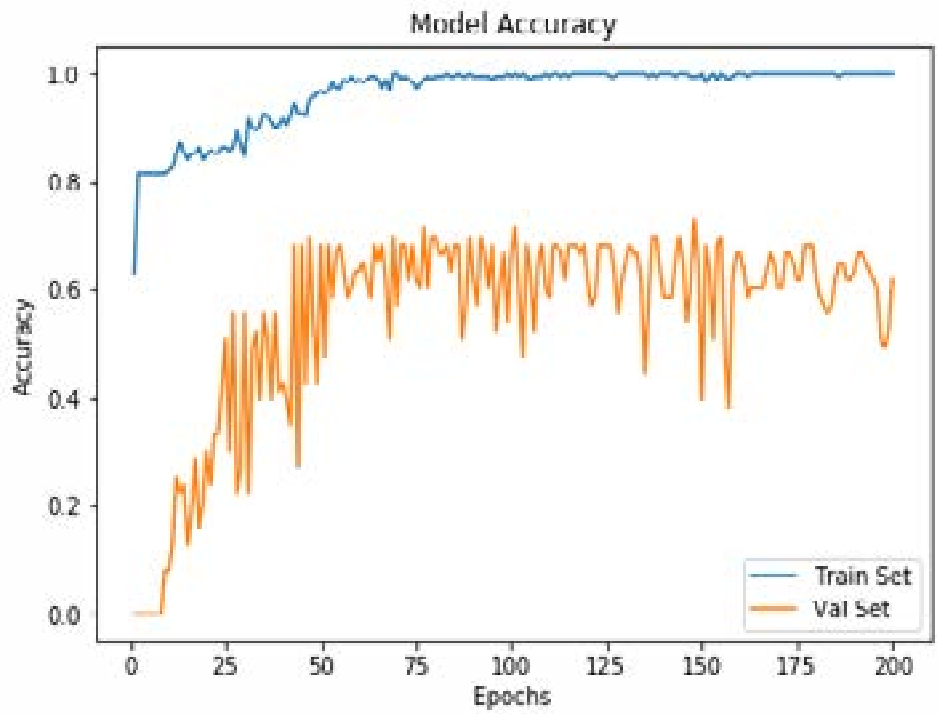
Test Accuracy:

```
Test accuracy: 0.61904764
```

> “This is the case of
> - **Overfitting**.

This model has not performed well, as we can see that training accuracy went to 100%, and our validation stays at 60%.

Model Accuracy:



After 50 epochs, Validation set accuracy remains the same and training accuracy keeps on overfitting.

Methods to Reduce Overfitting:

- Use **Dropout** increase its value and increase the number of training epochs.
- Reduce Fully Connected Layers and try to tweak your CNN model by adding more training pa rameters.
- Increase Dataset by using Data augmen tation

We have tried dropout at all the layers with different ranges and it does not make the model any better.We tried different epoch rates for training and accuracywas going lower than the better one.

By tweaking the hyperparameters we can get a little better accuracy but over fitting problem was not fixed, training model was still over fitting.

Lastly, we should either try to add more MRI
scan data or to do Data augmentation. As we don't have more MRI scan data, we will start doing Data Augmentation.

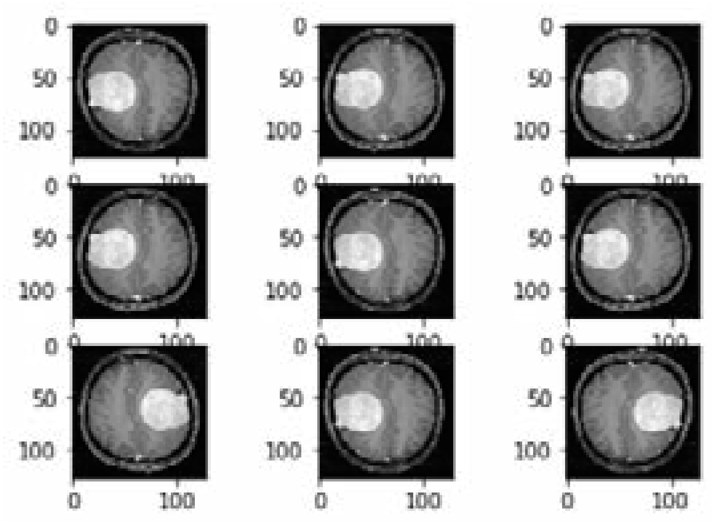
Data Augmentation:

"The **Image Data Generator** accepts the original data, randomly transforms it, and returns only the new, transformed data

Data augmentation is a strategy that enables to significantly increase the diversity of data available for training models, without collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. What we are going to do is rotate our Data and do flips on it.
Data augmentation has been widely used deep learning experts; Popular deep learn-ing package is KERAS. We will be using data augmentation from keras package with TensorFlow background.

Data augmentation on our dataset with tumorous image.

Result of Data Augmentation:



We can see the horizontal and vertical flip made on this image.
Using keras Image data Generator we are rotating and flipping images of our dataset.
When fitting our model, we should call Model.fit_Generator function to get image data generator into consideration.

Model Summary for Data Augmentation:

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 32, 32, 16)        3904
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 16, 16, 16)        0
_________________________________________________________________
dropout_32 (Dropout)         (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 16, 16, 32)        41504
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 8, 8, 32)          0
_________________________________________________________________
dropout_33 (Dropout)         (None, 8, 8, 32)          0
_________________________________________________________________
```

```
conv2d_28 (Conv2D)           (None, 8, 8, 64)          165952
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_34 (Dropout)         (None, 4, 4, 64)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_16 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_35 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 513
=================================================================
Total params: 736,673
Trainable params: 736,673
```

Model Fit Generator :

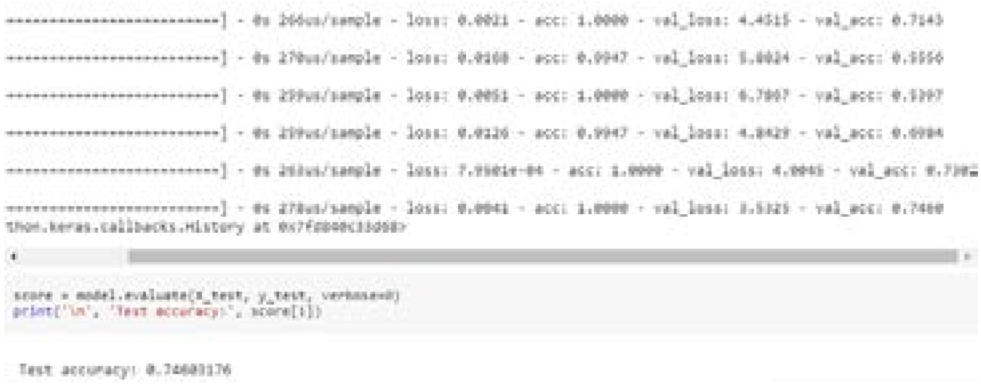
```
H = model1.fit_generator(aug.flow(X_train1, y_train1, batch_size=32), steps_per_epoch=50,validation_data=(X_valid1, y_valid1),epochs=80)

Epoch 1/80
50/50 [==============================] - 2s 34ms/step - loss: 0.0502 - acc: 0.9791 - val_loss: 3.5057 - val_acc: 0.5714
Epoch 2/80
50/50 [==============================] - 2s 33ms/step - loss: 0.0351 - acc: 0.9874 - val_loss: 3.2828 - val_acc: 0.6349
Epoch 3/80
50/50 [==============================] - 2s 33ms/step - loss: 0.0661 - acc: 0.9754 - val_loss: 3.3333 - val_acc: 0.6032
Epoch 4/80
50/50 [==============================] - 2s 34ms/step - loss: 0.0432 - acc: 0.9836 - val_loss: 3.9503 - val_acc: 0.6190
Epoch 5/80
50/50 [==============================] - 2s 33ms/step - loss: 0.0578 - acc: 0.9829 - val_loss: 2.7684 - val_acc: 0.5714
Epoch 6/80
50/50 [==============================] - 2s 33ms/step - loss: 0.0603 - acc: 0.9811 - val_loss: 3.1431 - val_acc: 0.6032
```

“ EPOCHS-80

Data augmentation happens while training and it uses the augmented data and not the main data.

Our main Image dataset is not used for training, instead our Image augmented data from training set is used for our model.

Training and Testing Accuracy



```
========================] - 0s 266us/sample - loss: 0.0021 - acc: 1.0000 - val_loss: 4.4515 - val_acc: 0.7143

========================] - 0s 270us/sample - loss: 0.0188 - acc: 0.9947 - val_loss: 5.0024 - val_acc: 0.5556

========================] - 0s 259us/sample - loss: 0.0051 - acc: 1.0000 - val_loss: 6.7867 - val_acc: 0.5397

========================] - 0s 259us/sample - loss: 0.0126 - acc: 0.9947 - val_loss: 4.8429 - val_acc: 0.6984

========================] - 0s 263us/sample - loss: 7.9501e-04 - acc: 1.0000 - val_loss: 4.0045 - val_acc: 0.7302

========================] - 0s 278us/sample - loss: 0.0041 - acc: 1.0000 - val_loss: 3.5325 - val_acc: 0.7460
thon.keras.callbacks.History at 0x7fd0040c33d68>

score = model.evaluate(x_test, y_test, verbose=0)
print('\n', 'Test accuracy:', score[1])

 Test accuracy: 0.74603176
```

This is the accuracy as better it gets – 75%

We have tried tweaking each parameters of different layers, tried dropout methods again, used different Epochs and batch normalization. We have fixed Overfitting problem and got a better accuracy.

Further Proceedings

As Overfitting has been the major problem, trying to get more data would be preferable. With more data we can start using heavy algorithms with more layers and algorithms like VGG16, RESNET50, INCEPTION V3 can be used.Cropping out the brain image out of MRI scan using min-max contours can also be a way to delete unwanted image data and focus on feature extraction on only brain part of the image.

ABOUT US

Pepgra is a leading global contract research outsourcing organization provider of scientific, knowledge-based services to bio-pharmaceutical, generic pharmaceutical, biotech, medical device companies and healthcare companies in the areas of clinical trial monitoring, regulatory writing, post-market surveillance, biostatistics and statistical programming services. Our mission is to become a strategic partner to global life science companies providing high quality knowledge-based expertise across the product lifecycle with the ultimate objective of improving quality of healthcare for patients worldwide. Our corporate headquarter is located in India with operations in USA, and the Philippines

Format type: E-Book

UK: 10 Park Place,
Manchester M4 4EY.
UK: +44-1143520021
Email:info@pepgra.com
Web:www.pepgra.com