DenseNet was able to achieve high accuracy compared with ResNet using fewer parameters. In this architecture, each layer of the network obtains additional input from all its previous layers and passes its own feature map to next subsequent layers of the network. Thus, each layer receives collective knowledge from its all preceding layers, which helps in improving the overall accuracy of model. DenseNet network become compact and thinner as each layer receives feature map from its all preceding layers. DenseNet121 with 121 layers used in this research to solve the problem of brain tumor classification.

3.5. A proposed CNN Model for brain abnormality classification

Brain tumor classification performs using CNN that is consists with several layers as displayed in the following figure 2. The convolutional layer is followed by an activation layer. We have used Rectified Linear Unit (ReLU) as an activation function. Moreover, batch normalization is used to make the network faster and stable. The addition of pooling layer after convolutional layer is a common pattern in CNN network. Here, max pooling layers are applied after some of the convolutional layers that calculate maximum value for each patch of a feature map. At last, flatten layer is applied to convert the data into one dimensional array to pass it as an input to the next layer. The fully connected layers are used for classification and the last fully connected layer outputs in three class. It gives "1" as an output for meningioma, "2" for glioma and "3" for pituitary tumor.

Layer	**Properties**
Image Input	128 X 128 X 3
Convolutional + ReLU	64 Filters of size (3 x 3 x 3) with Padding = same, Rectified Linear Unit
Convolutional + ReLU + MaxPooling	64 Filters of size (3 x 3 x 3) with Padding = same, Rectified Linear Unit, 2×2 max pooling with stride [2 2]
Convolutional + Batch Normalization + ReLU	64 Filters of size (3 x 3 x 3) with Padding = same, Batch Normalization, Rectified Linear Unit
Convolutional + Batch Normalization + ReLU + MaxPooling	64 Filters of size (3 x 3 x 3) with Padding = same, Batch Normalization, Rectified Linear Unit, 2×2 max pooling with stride [2 2]
Convolutional + Batch Normalization + ReLU	64 Filters of size (3 x 3 x 3) with Padding = same, Batch Normalization, Rectified Linear Unit
Convolutional + Batch Normalization +ReLU	64 Filters of size (3 x 3 x 3) with Padding = same, Batch Normalization, Rectified Linear Unit
Convolutional + Batch Normalization +ReLU + MaxPooling	64 Filters of size (3 x 3 x 3) with Padding = same, Batch Normalization, Rectified Linear Unit, 2×2 max pooling with stride [2 2]
Convolutional + Batch Normalization +ReLU	64 Filters of size (3 x 3 x 3) with Padding = same Batch Normalization, Rectified Linear Unit
Convolutional + Batch Normalization +ReLU	128 Filters of size (3 x 3 x 3) with Padding = same Batch Normalization, Rectified Linear Unit
Convolutional + Batch Normalization +ReLU	128 Filters of size (3 x 3 x 3) with Padding = same Batch Normalization, Rectified Linear Unit
Flatten	Flatten into MLP
Fully Connected layer + Batch Normalization + ReLU	256 Neurons with Batch Normalization and Rectified Linear Unit
Fully Connected layer + Batch Normalization + ReLU	256 Neurons with Batch Normalization and Rectified Linear Unit
Fully Connected + Softmax	3 Neurons in Last Layer for final output

Fig. 2. A Network Architecture of a proposed CNN Model for brain abnormality classification

4. Experimental Methodology

The following figure 3 describes the experimental methods used in this research. Each deep learning model requires data for training. In this research, the dataset is acquired from Figshare data repository. The data is then converted from .mat file to .jpg file. There are three classes are available in dataset. Therefore, the images are organised into appropriate folders. Data augmentation is applied to the dataset to artificially enhance the size and diversity in the data. Then benchmark CNN models are identified that includes VGG16, VGG19, ResNET101, MobileNetV2 etc. For conducting experiment, we have used Keras and TensorFlow libraries for CNN model implementation [23]. The dataset is then after split into training, validation and testing. The ratio preserves is 80%, 10% and 10% respectively. To load the image, Keras provided ImageDataGenerator class that is used in the experiment. The models are built and compiled to start the training. The model is evaluated to obtain the results. Standard metrics like accuracy and loss are used for performance evaluation.

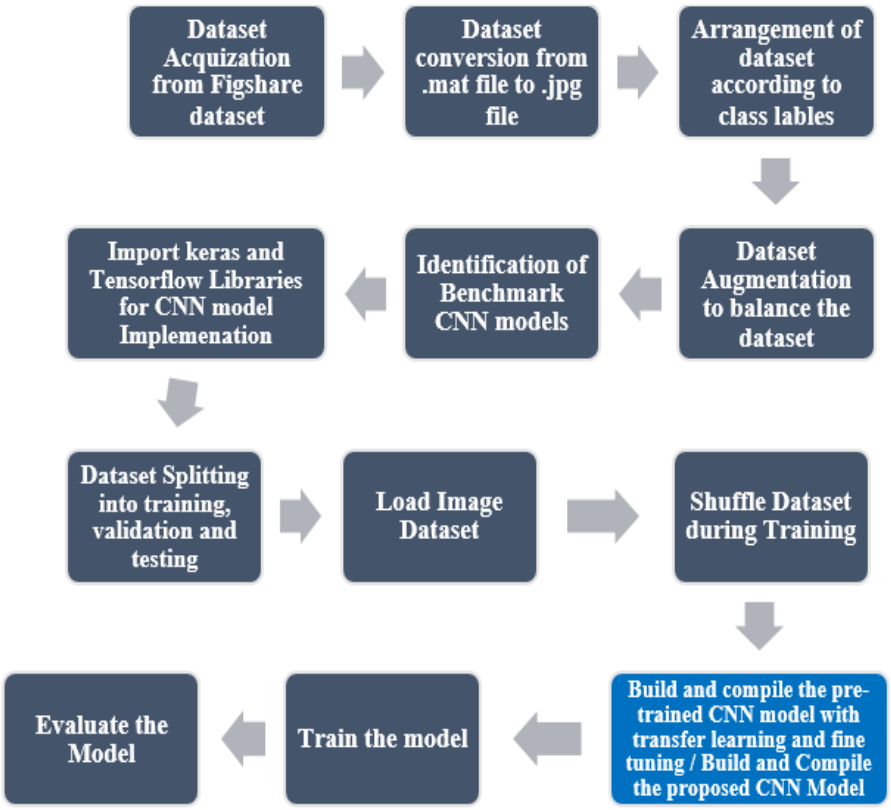

Fig. 3. An Experimental methodology for Training and Evaluation of CNN Models

4.1. Hardware and Software

For implementation, we have used Keras and TensorFlow deep learning libraries. Keras is a high level API that builds on the top of TensorFlow library. Moreover, Keras is specially designed for performing deep learning based applications and provides a convenient way to build CNN models. Keras is a high level API that uses Tensorflow or Theano as a backend. Keras is very simple and available with very good documentation and learning resources. TensorFlow is an open-source library that offers both high-level and low-level APIs.

For hardware and software support, a cloud platform namely Google Colab is used. Colab is a free Jupyter notebook environment that runs completely in the cloud. We have used Google Colab notebook and executed code on Google's GPU using Google's cloud servers. Colab supports a development environment in Python that makes easy to use other packages like Scikit, NumPy, Matplotlib etc.

4.2. Hyperparameters and Network Training

Hyperparameters are used to define the CNN structure and determines how the network is trained. The entire training process is governed by set of hyperparameters. It is required to set hyperparameters before training. There are various hyperparameters available including learning rate, optimizer, loss function, batch size, epoch etc [24]. The following are the details regarding the hyperparameters used in this experiment.

- **Learning rate:** Learning rate regulates the speed for learning weights by a model. A larger learning rate speeds up the learning but may the model not converged. A lower learning rate converges smoothly compared to a larger learning rate. A decaying rate of learning is preferred

- **Batch size:** Batch size decides the number of samples passed in a model for a particular iteration. The typical sizes chosen are 32, 64, 128, etc.
- **Number of Steps:** the number of steps defines the total number of training iterations required for the proposed object detection model. This number depends on the dataset's size and several other factors (including for how long the model does train). A training step is referred to as an update in one gradient. During one step, the amount of training samples assigns to batch size is processed.
- **Activation function:** Activation functions introduce non-linearity in the network and help to learn nonlinear prediction boundaries. ReLU, Sigmoid, Tanh, etc. are some of the most popular activation functions used in object detection models.
- **Dropout rate:** The dropout rate is used to solve the overfitting problem of the network. Dropout means dropping neurons of the network. They are chosen at random during the training process.
- **Optimizer:** Optimizer algorithms are used to calculate the optimal values for internal parameters (i.e., weights, bias, etc.). It helps to minimize the error function.

The following table 1 describes the values set for hyperparameters before training. The following values are obtained after tuning the hyperparameters to get optimum level of accuracy.

Table 1. Values sets for Hyperparameters

Hyperparameter Name	**Value**
Learning Rate	0.001
Batch Size	32
Training step or Iterations	9652/32 (Total training samples/batch size)
Epoch	20
Optimizer	SGD
Loss Function	Categorical Cross-entropy
Dropout rate	0.2
Epoch	20

4.3. Loss Functions

There are two performance parameters most often used to measure the performance of CNN model i.e. accuracy and loss. The aim of a good training is not only about increase the accuracy but also to decrease the loss. This can be achieved by optimizing the weights of CNN model. Loss is a function that measures an error occurred in prediction by a model [25]. There are two functions generally used to measure loss i.e. mean square error and categorical cross entropy. To determine the performance of a CNN model, accuracy and loss can be graphically described as shown in figure 4. The loss and accuracy are measured at two stages; training and validation. In following figure 4, training and validation accuracy and losses are going parallel and closely to each other. This shows that model is trained adequately without underfitting and overfitting.

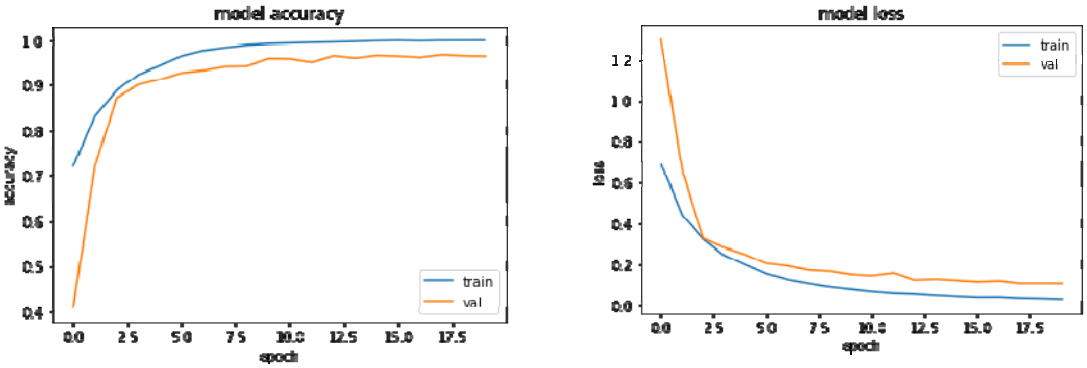

Fig. 4. Accuracy and Loss for the Proposed Model

5. Results and Discussion

The following table 2 shows the total number of parameters, trainable parameters and non-trainable parameters contain by a CNN model. Most of the benchmark models are having large number of parameters to learn as they having very deep network architecture. Whereas, from table 2, it has found that, the proposed CNN model is having lowest number of parameters without compromising accuracy.

Table 2. Parameters Learned by CNN Models

	VGG16	**VGG19**	**MobileNet V2**	**ResNet50**	**ResNet101**	**Densenet 121**	**Proposed CNN**
Total params	21,204,035	26,513,731	18,381,123	49,321,731	68,383,491	19,949,379	2,683,075
Trainable params	6,489,347	6,489,347	16,157,251	25,802,371	25,854,595	12,995,523	2,664,387
Non-trainable params	14,714,688	20,024,384	2,223,872	23,519,360	42,528,896	6,953,856	18,688

The following table 3 shows the accuracy and loss values for different CNN models. The values for accuracy and loss are provided for training, validation and testing. From the results, it can be found that, VGG16 and VGG19 obtained the lowest accuracy and highest loss. MobileNetV2 performs better than VGG16 and VGG19. Both of the variants for ResNet i.e. ResNet50 and ResNet101 work well including Densenet121. Among all, the proposed CNN model outperforms with highest training, validation and testing accuracy i.e. 0.9985, 0.9637 and 0.9670. Moreover, the values for losses are also lowest i.e. 0.0246, 0.1018 and 0.1038 respectively. Along with the optimum values of accuracy and loss, the time taken to train the entire model is significantly less for the proposed CNN model.

Table 3. Accuracy and Loss for CNN Models

Parameters	**VGG16**	**VGG19**	**MobileNet V2**	**ResNet50**	**ResNet101**	**Densenet 121**	**Proposed CNN**
Training accuracy	0.8877	0.8625	0.9762	0.9925	0.9902	0.9910	0.9985
Training Loss	0.2838	0.3426	0.0647	0.0254	0.0319	0.0283	0.0246
Validation Accuracy	0.8730	0.8606	0.8994	0.9417	0.9506	0.9633	0.9637
Validation Loss	0.2159	0.3207	0.2477	0.1687	0.3082	0.1418	0.1018
Testing Accuracy	0.8657	0.8690	0.9062	0.9594	0.9552	0.9670	0.9670
Testing Loss	0.4759	0.3215	0.1839	0.1166	0.1248	0.1620	0.1038
Time taken to Train the Model	0:33:53.256950	0:32:54.186228	0:31:25.399036	0:47:29.717618	1:14:41.849550	1:15:42.386142	0:14:18.366423

6. Conclusion

In this paper, we have developed and proposed a convolutional neural network (CNN) model for multi-class classification of brain abnormality. Brain tumor is considered one of the major brain abnormalities and therefore, the model is applied and experimented on brain tumor dataset. Several benchmark CNN models that include VGG16, VGG19, MobileNetV2, ResNet50, ResNet101 and DenseNet121 are applied and evaluated for performance comparison. By seeing the results, it can be found that, the proposed CNN model outperforms among other CNN models. It consumes less parameters and less time for training. From the graphs of accuracy and loss, it is observed that the proposed model converge optimally as compared to other CNN models. Moreover, the highest accuracy and lowest loss are also obtained from the proposed model.

References

[1] Villanueva-Meyer, JE.; Mabray, MC.; Cha S. (2017): Current Clinical Brain Tumor Imaging. Neurosurgery, 81(3), pp. 397-415.
[2] Zacharaki, EI.; Wang, S.; Chawla, S; et al. (2009): Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme. Magn Reson Med., 62(6), pp.1609-1618.
[3] Zhao, J.; Meng, Z.; Wei, L.; Sun, C.; Zou, Q.; & Su, R. (2019): Supervised Brain Tumor Segmentation Based on Gradient and Context-Sensitive Features. Frontiers in neuroscience, 13(144).
[4] Brain Tumor: Statistics, Cancer.Net Editorial Board, 11/2017 (Accessed on 17th January 2019)
[5] Kavitha, A.R. & Chellamuthu, Chinna. (2018): Advanced Brain Tumour Segmentation from MRI Images. 10.5772/intechopen.71416.
[6] Patel, S. (2020): A Comprehensive Analysis of Convolutional Neural Network Models. International Journal of Advanced Science and Technology, 29(04), pp. 771 - 777.
[7] Patel, S.; Patel, A. (2018): Deep Leaning Architectures and its Applications: A Survey. International Journal of Computer Sciences and Engineering, 6(6), pp. 1177-1183.
[8] Li, Ming & Kuang; Lishan & Xu; Shuhua & Sha; Zhanguo. (2019). Brain Tumor Detection Based on Multimodal Information Fusion and Convolutional Neural Network. IEEE Access, 7, pp. 180134-180146.
[9] T., Hossain; F., S., Shishir; M., Ashraf; M., A., Al Nasim; F. Muhammad Shah: "Brain Tumor Detection Using Convolutional Neural Network," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 2019, pp. 1-6.
[10] H. UCUZAL,; A. G. İ. BALIKÇI ÇİÇEK,; A. G. A. K. ARSLAN; and C. ÇOLAK: "A Web-Based Application for Identifying Objects In Images: Object Recognition Software," 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2019, pp. 1-5, doi: 10.1109/ISMSIT.2019.8932735.
[11] Zahra, Sobhaninia; Safiyeh, Rezaei; Alireza, Noroozi; Mehdi, Ahmadi; Hamidreza, Zarrabi; Nader, Karimi; Ali, Emami; Shadrokh, Samavi.(2018): Brain Tumor Segmentation Using Deep Learning by Type Specific Sorting of Images, arXiv:1809.07786.
[12] S., Zhou; D., Nie; E., Adeli; J., Yin; J., Lian; D., Shen. (2020): High-Resolution Encoder–Decoder Networks for Low-Contrast Medical Image Segmentation, in IEEE Transactions on Image Processing, vol. 29, pp. 461-475
[13] Pereira S,; Pinto A,; Alves V,; Silva CA.(2016): Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images. IEEE Trans Med Imaging. 35(5), pp. 1240-1251.
[14] Alqazzaz, S.; Sun, X.; Yang, X. et al. (2019): Automated brain tumor segmentation on multi-modal MR image using SegNet. Comp. Visual Media 5, 209–219
[15] S. Hussain,; S. M. Anwar; and M. Majid, "Brain tumor segmentation using cascaded deep convolutional neural network," 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, 2017, pp. 1998-2001.
[16] fighshare data repository, https://figshare.com/articles/brain_tumor_dataset/1512427#:~:text=This%20brain%20tumor%20dataset%20contains,be%20found%20in%20readme%20file.
[17] A. Mikołajczyk; M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Swinoujście, 2018, pp. 117-122.
[18] Ganatra, N.; & Patel, A. (2018). A Comprehensive Study of Deep Learning Architectures, Applications and Tools. International Journal of Computer Sciences and Engineering. vol. 6. pp. 701-705.
[19] Ganatra, N.; and Patel, A. (2020). A Multiclass Plant Leaf Disease Detection using Image Processing and Machine Learning Techniques. International Journal on Emerging Technologies, 11(2), pp. 1082–1086.
[20] Simonyan, K.; Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.
[21] Andrew G. Howard; Menglong Zhu; Bo Chen, Dmitry Kalenichenko; Weijun Wang; Tobias Weyand; Marco Andreetto; Hartwig Adam; MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv:1704.04861.
[22] Liu, Bing; Liu, Qiao; Zhu, Zhengyu; Zhang, Taiping; Yang, Yong. (2019). MSST-ResNet: Deep multi-scale spatiotemporal features for robust visual object tracking. pp.235-252.
[23] TensorFlow, https://www.tensorflow.org/
[24] Mboga, Nicholus; Persello, Claudio; Bergado, John Ray; Stein, Alfred. (2017). Detection of Informal Settlements from VHR Images Using Convolutional Neural Networks. Remote Sensing. 9(1106), 10.3390/rs9111106.
[25] Galván Tejada, Carlos & Zanella Calzada, Laura & García-Domínguez, Antonio & Magallanes-Quintanar, Rafael & Luna-Garcia, Huizilopoztli & Celaya Padilla, Jose & Galván Tejada, Jorge & Vélez-Rodríguez, Alberto & Gamboa-Rosales, Hamurabi.: (2020). Estimation of Indoor Location Through Magnetic Field Data: An Approach Based On Convolutional Neural Networks. ISPRS International Journal of Geo-Information, 9(226) 10.3390/ijgi9040226.