Introduction

Over the past decades, significant development has been made using Neural networks on the classification of medical images. Deep Learning (DL) is one of the branches of machine learning dependent on studying various layers of perception, through the development of a hierarchy of characteristics that distinguish lower levels from higher levels and that can lead to identifying several upper-level characteristics. A significant issue in computer vision is the classification of pictures, which can be represented as the categorization of pictures within one of the multiple predetermined groups.

Convolutional neural networks (CNNs) [1] are a type of neural network architecture that has shown remarkable success in image classification tasks. Convolutional neural networks provide a more scalable approach to image classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image. That said, they can be computationally demanding, requiring graphical processing units (GPUs) to train models.

The advent of deep learning models and their application towards medical image classification has garnered massive attention, recently. However, there exist a plethora of challenges; one such example is the collection of medical images i.e., data collection, since the gathering and classification of health information involve an intensive process which is both time consuming and has the potential for infringement of patient privacy in terms of their records. Furthermore, data cleaning and accuracy is of paramount importance which requires the expertise of several experts. Another important aspect is the fragile nature of the practice, thus ensuring high accuracy and precision is required for medical image classification – this is where CNNs have been particularly successful due to their ability to generalize well with the dataset and subsequently produce the desired results. However, it has been observed that the computational (processing) power required when training deep convolutional neural networks is too high for standard machines and can be seen as an obstacle in scalability of models.

Our goal is to devise an end-to-end framework to classify medical imagery with CNNs. The data set we have utilized consists of high-resolution MRI scans of the brain to effectively distinguish between patients that have a brain tumor form those who do not. The context of medical image classification introduces certain challenges that we must address to turn CNNs into a relevant classification tool. Notably, we must: 1) apply specific neural network architecture(s) for our problem; 2) acquire an appropriate training data and handle its eventual inaccuracies; and 3) generate highly accurate output classification scores.

The report has been structured as follows: Section 2 includes a literature review of existing work and highlights the architecture of the selected paper. Section 3 is a in depth overview of the deep learning architecture implemented, the associated dataset, and the methodology implemented. Section 4 discusses the results obtained and presents a comparative analysis of the models used, and highlights the limitations and problems faced. Finally, Section 5 identifies the key findings and discusses the potential for future work.

Literature Review

Maggiori, Tarabalka, Charpiat, Alliez [2] have devised an end-to framework to classify satellite imagery with CNNs. They propose a dense pixel-wise classification architecture which is based on the patch-based model, where the problem is to assign a category to an entire image patch, such as “residential” or “agricultural” area. However, they have extended this facility to develop a CNN that outputs a per pixel classification that constitutes the entire input based on a series of convolutions and deconvolutions only. The authors have relied on CNN architecture proposed by Mnih [3], which is characterized as having a fully connected layer i.e., connected to all the outputs of the previous layer, to produce the output classification patches.

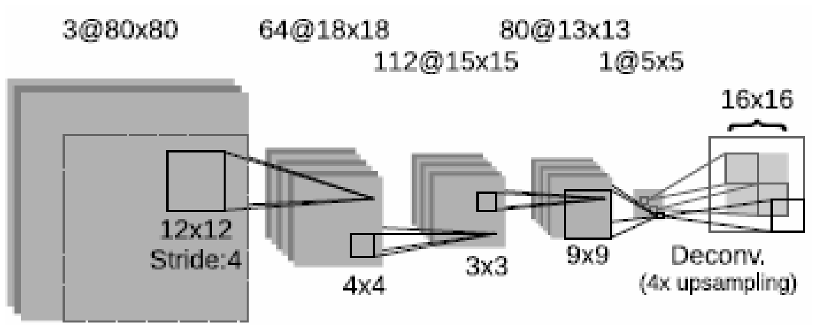


Figure 1: Fully Convolutional Architecture

The area of application is remote-sensing images and their subsequent analysis – a significant area of research in many practical applications e.g., precision agriculture and urban planning. In addition, they have employed a large-scale data set of satellite imagery that has its own set of challenges. The authors have then identified a two-step approach to deal with the eventual inaccuracies of the dataset. The methodology proposed leverages the use of transfer learning, a popular technique whereby existing pre-trained models are fine-tuned on unique problems by training the later layers and using pre-trained weights to optimize the current problem.

Finally, they have conducted a thorough analysis of their proposed architecture with those in literature to review model performance and accuracy. In this regard, they evaluated their model on several benchmark datasets to output relevant metrics.

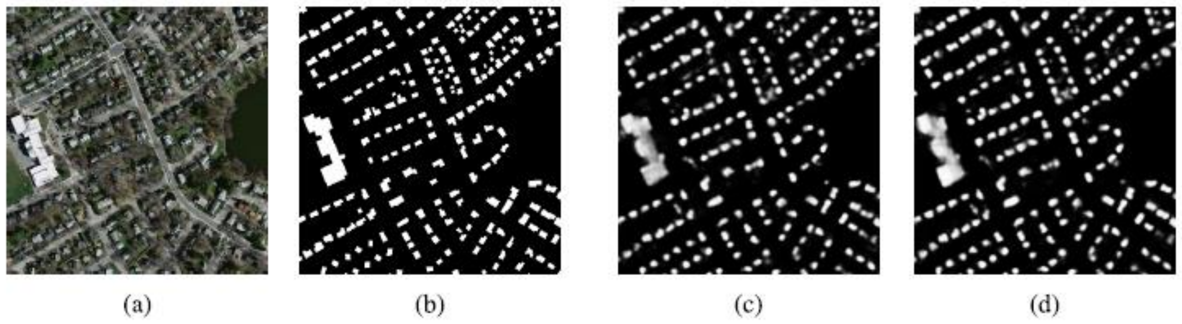


Figure 2: Experimental results on a fragment of the Boston data set. (a) Color image. (b) Reference data. (c) Patch-based fuzzy map. (d) FCN fuzzy map.

Afshar. P, et al. 2018 [4] proposed a new Capsule network model which mainly works as neurons that are implemented on biomedical images. In the architecture, along with the CapsNet model, the Authors combined Convolutional Neural Networks to segment the images based on parameters: segmented tumors and brain images. Dataset deals with 233 subjects and their MRI images of 3,064 that contain three types of brain tumours. After a comparison of different architectures, the highest accuracy of 86.56% has been predicted on the 1-Convolutional layer that has 64 feature maps.

Kermany. DS, et al. 2018 [5] gave an introduction about Transfer learning and implemented on the datasets of pediatric radiograph chest images. But the primary goal is to build the architecture to predict the retinal diagnosis based on the Optical Coherence Topography images using Transfer learning. By consulting six random experts, they examined the 1000 OCT images and the diagnosis for vision loss. Those 1000 images of 633 patients were given as an input to the developed model and got a sensitivity of 96.6% and weight error has 12.7%. The authors compared the four different stages of retinal infections using the confusion matrix and got an approximation of 99%.

S. Hoo-Chang, et al. 2016 [6] implemented 5 different CNN architecture based neural networks methods to identify the interstitial lung disease using the dataset of 2D images of CT scan slices. The dataset contains 905 images of 120 patients with 6 types of lung tissue conditions. The authors trained the model using GoogleNet, CifarNet, AlexNet, and ImageNet architectures. The authors also implemented different architectures to detect the Thoracoabdominal Lymph node and made empirical evaluation.

Mohsen. Heba, et al. 2018 [7] discussed brain tumors and their importance to classify the tumor in MRI images. Dataset consists of human brain MRI images of 66 people of which 44 were abnormal and 22 were normal conditions. Fuzzy C-means are used for image segmentation and feature extraction, authors used discrete wavelet transformation. Image segmentation is done by using DNN and got a classification rate of 96.97%. The project was built on the MATLAB R2015a.

Cui, Z., Yang, J., & Qiao, Y. 2016 [8] proposed a patch-based CNN made up of 7 layers. The proposed CNN was implemented on the MRI images of the human brain. Dataset considered is available publicly that contains 103 MRI images contains 4 categories of brain diseases. For training, 100000 patches were considered from the MRI images. A dice ratio of 95.19% is achieved which is used to measure the accuracy of segmentation whereas CNN architecture accuracy was 90.83%.

M. Russo, M. Stella et al. 2019 [9] proposed a CNN method on which they compared VGG16 and ResNet50 architectures were compared on the histopathology images of lungs to detect lung cancer. The dataset considered was ACDC@LUNGHP (Automatic Cancer Detection and classification in Whole slide Lung histopathology). From the dataset, the Authors considered only an ROI part of images. VGG16 gave accuracy for patch classification is 97.9%.

Methodology

I. Convolutional Neural Networks

CNNs are a type of neural network architecture that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and pattern recognition. To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data. A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and with speech or audio signal inputs. A CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer.

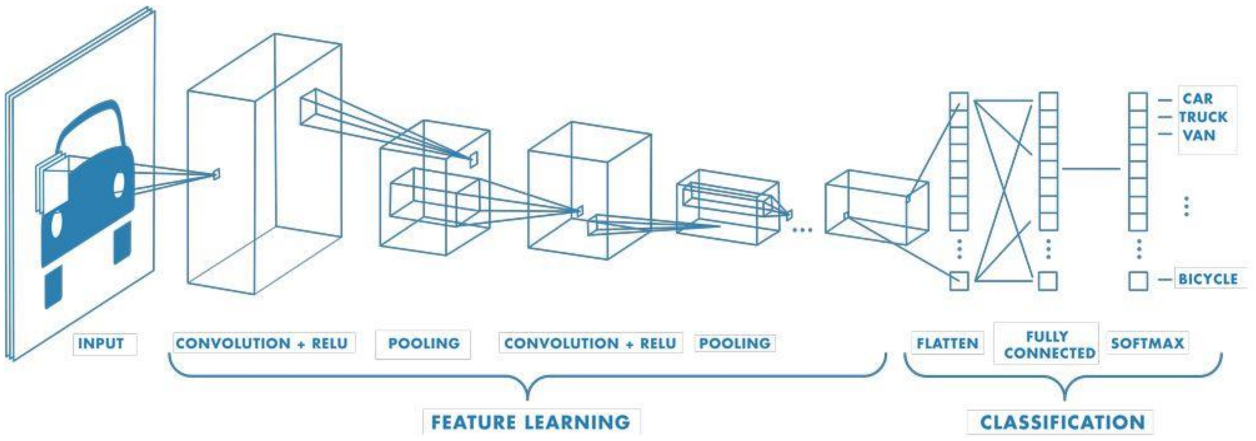


Figure 3: Schematic representation of a convolutional neural network

Convolutional Layer

The convolutional layer [10] is the core building block of a CNN, responsible for most computations. In this layer, a kernel or filter is convolved with the input image, sliding over the receptive fields of the image to detect features. This process creates a feature map or convolved feature, which is ultimately used to interpret the image and extract relevant patterns from it. Multiple convolutional layers can be stacked to create a deep network, allowing for more complex features to be learned.

Spatial Arrangement

Three hyperparameters control the size of the output volume: the depth, stride, and zero-padding. First, the depth of the output volume is a hyperparameter: it corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in the presence of variously oriented edges, or blobs of colour. We will refer to a set of neurons that are all looking at the same region of the input as a depth column (some people also prefer the term fiber).

Second, we must specify the stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

It may be practical to pad the input volume with zeros at the border on occasion. This zero-padding's size is a hyperparameter. Zero padding has the advantage of giving us control over the spatial size of the output volumes; most frequently, as we'll see in a moment, we'll use it to precisely maintain the spatial size of the input volume so that the input and output width and height are identical.

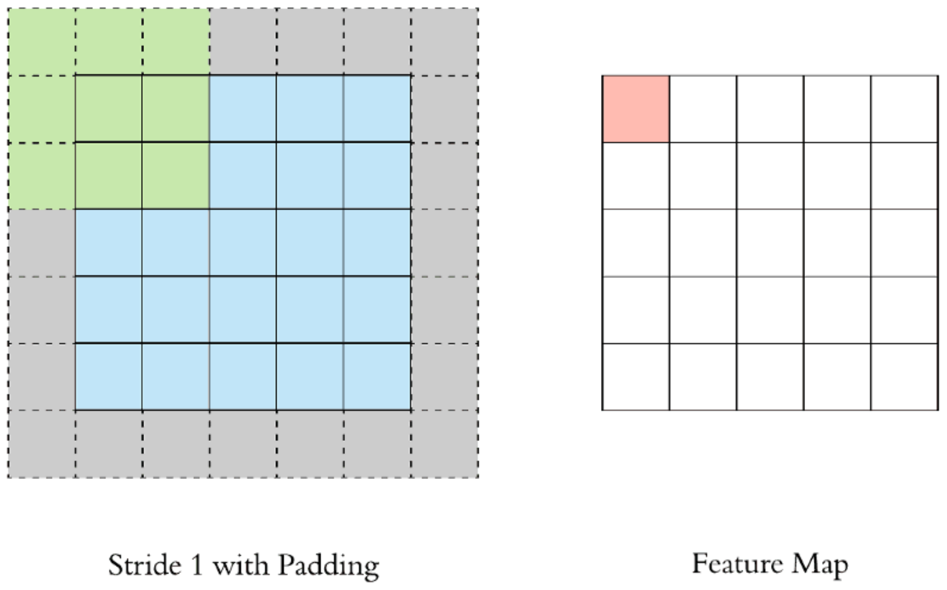


Figure 4: Stride with Padding [source]

To summarize, the convolutional layer can be defined by the following:

Accepts a volume of size **W1×H1×D1** (input)
Requires four hyperparameters:

- Number of filters **K**,
- their spatial extent **F**,
- the stride **S**,
- the amount of zero padding **P**

Produces a volume of size **W2×H2×D2** (output) where:

- **W2=(W1−F+2P)/S+1**
- **H2=(H1−F+2P)/S+1**
- **D2=K**

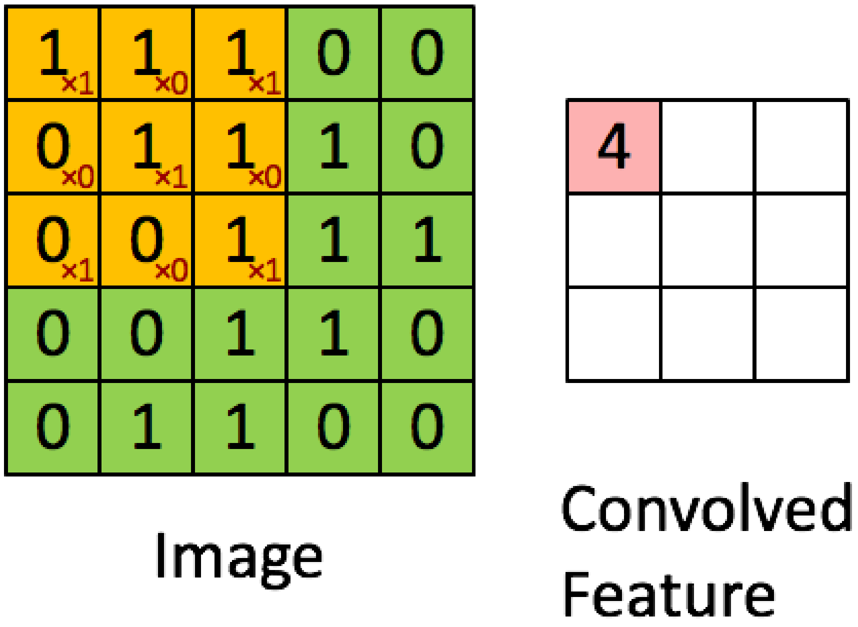


Figure 5: Convolution Demonstrated [source]

Pooling Layer

The pooling layer is responsible for reducing the number of parameters in the input and improving the efficiency of the CNN. Like the convolutional layer, a kernel or filter is applied to the input image. However, instead of detecting features, the pooling layer aggregates information and down samples the feature maps. This reduces the dimensionality of the data, resulting in computational savings and a more efficient network. The pooling layer also introduces some information loss, but this is offset by the improved computational efficiency. There are several different pooling layer types i.e., max, average etc., with each having their own characteristics. However, max pooling is the most widely adopted layer type with a proven record in literature. It is shown below in Figure 6.

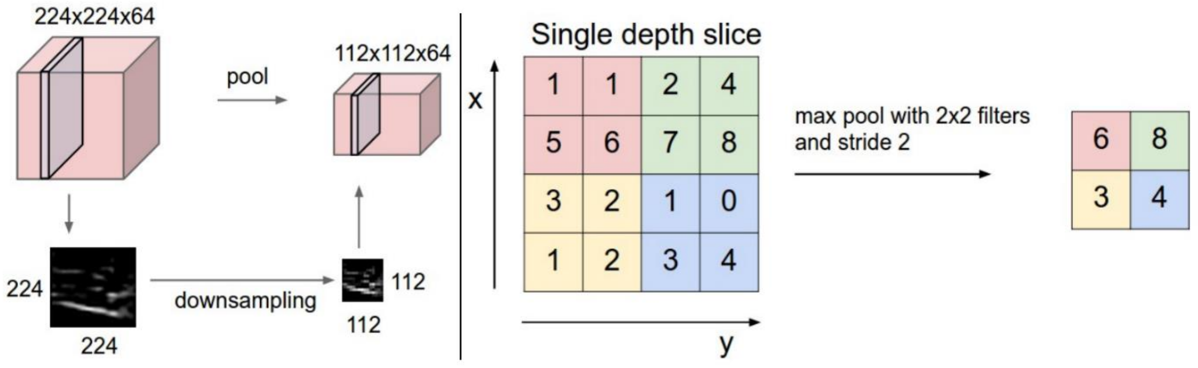


Figure 6: Max Pooling Layer

Fully Connected Layer

The fully connected (FC) layer is where image classification happens in the CNN, based on the features extracted in the previous layers. In this layer, all the inputs from the previous layer are connected to every activation unit or node of the next layer. However, not all layers in the CNN are fully connected, as this would result in a dense network that is computationally expensive and would increase losses, affecting the output quality. The FC layer is the final layer of the CNN, responsible for classifying the input image into a specific category.

II. Transfer Learning/ Fine Tuning

Fine-tuning is a very common procedure in neural network literature. The idea is to adapt an existing pre-trained model to a different domain by executing a few training iterations on a new data set. The notion of fine-tuning is based on the intuition that low-level information/features can be reused in different applications, without training from scratch. Even when the final classification objective is different, it is also a relevant approach for initializing the learnable parameters close to good local minima, instead of initializing with random weights. After proper fine-tuning, low-level features tend to be quite preserved from one data set to another, while the higher layers' parameters are updated to adapt the network to the new problem. When fine-tuning, the training set for the new domain is usually substantially smaller than the one used to train the original network. This is because one assumes that some generalities of both domains are well conveyed in the pre-trained network (e.g., edge detectors in different directions) and the fine-tuning phase is just needed to conduct the domain adaptation. When the training set used for fine-tuning is very small, additional considerations to avoid overfitting are commonly taken, such as early stopping (executing just a few iterations on the new training data set), fixing the weights at the lower layers, or reducing the learning rate.

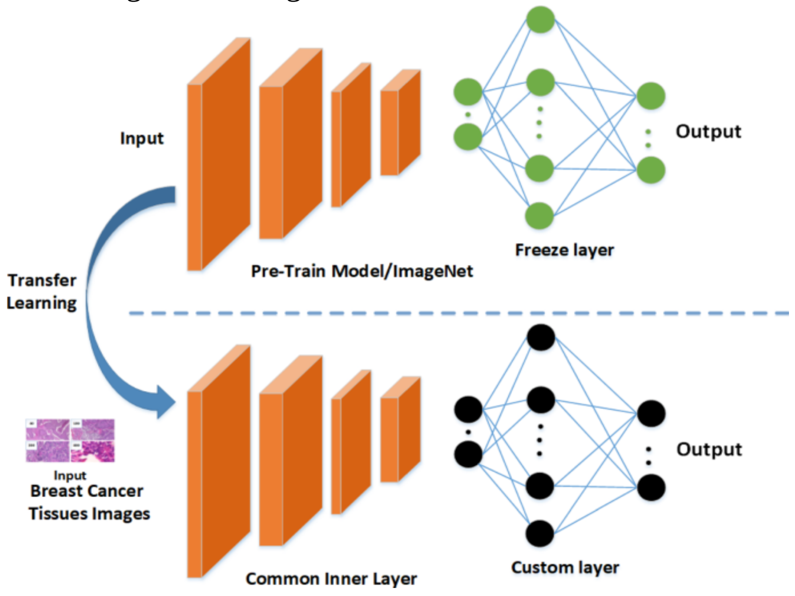


Figure 7: Transfer Learning [source]

We now incorporate the idea of neural network fine-tuning, to perform training on imperfect data. Our approach proceeds in two steps. In step 1, we employ existing benchmark deep neural networks such as VGG19, VGG16 etc., and add additional layers to the model. The original model is not trained, and we utilize the existing trained weights to train the latter layers on our dataset. In this way, the models generalize well on the dataset whilst ensuring the training process is efficient and effective.

III. Medical Image Dataset

For this project, we have selected the following dataset: ‘Brain MRI Images for Brain Tumor Detection’. The dataset consists of 253 image samples of high-resolution brain MRI scans. The images are grayscale in nature and vary in size. The aim of the dataset is to provide evidence for conducting image analysis to predict whether each image belongs to the Tumor or Non-tumor category. Table 1 describes some characteristics of the dataset.

Table 1: Dataset Characteristics

Characteristics	
Nature	Grayscale
Total	253
Categories	2 – (Tumor/No Tumor)
Tumor – No Tumor	155 – 98 Images

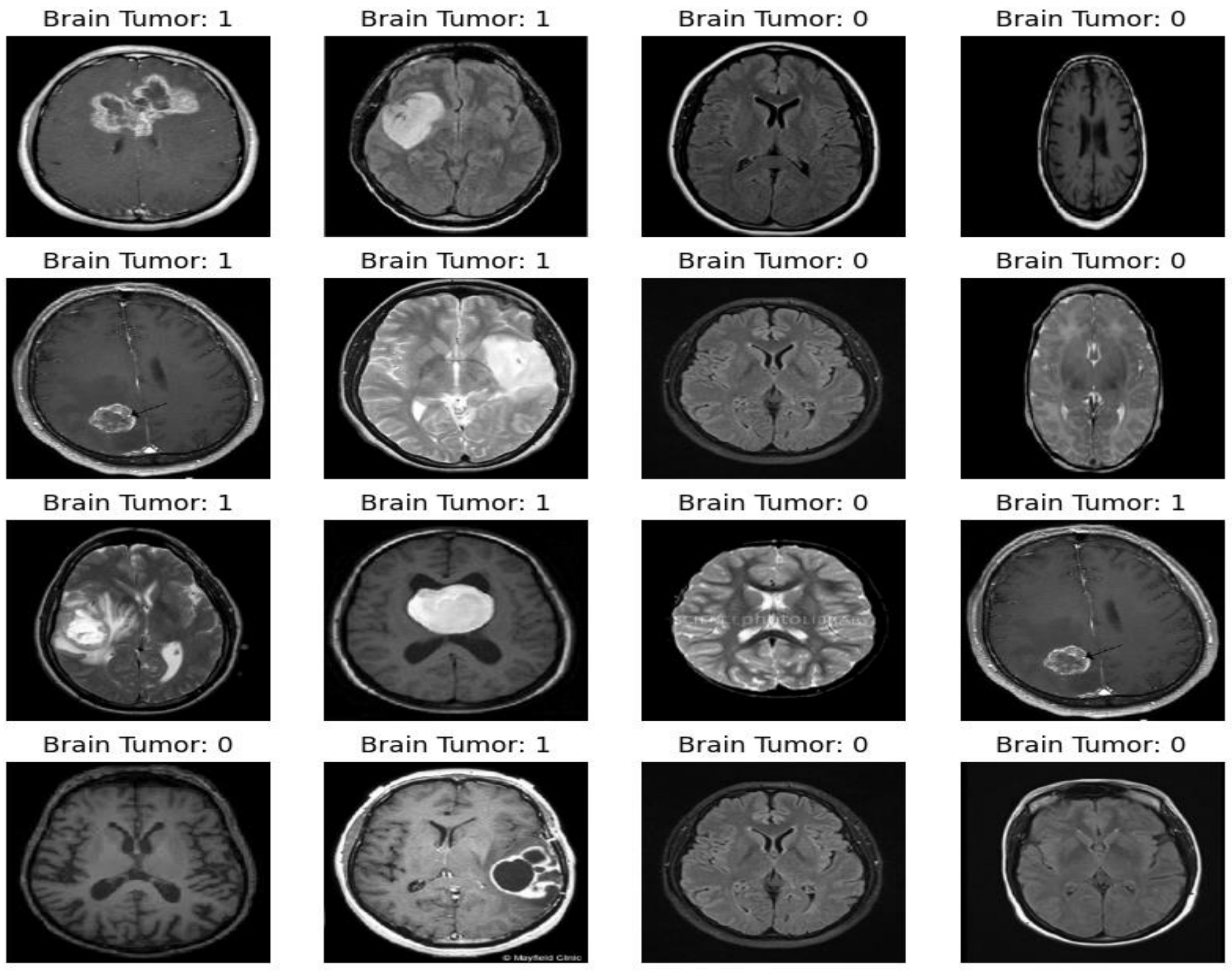


Figure 8: Dataset Instances

IV. Architecture

For this project, we have implemented the entire machine learning workflow that comprises of dataset procurement and preprocessing to model building and training. This is followed by the evaluation phase, where multiple metrics are output, and the model is subsequently optimized based on the functional objectives. Finally, the model is tested on unseen data to gauge its overall performance.

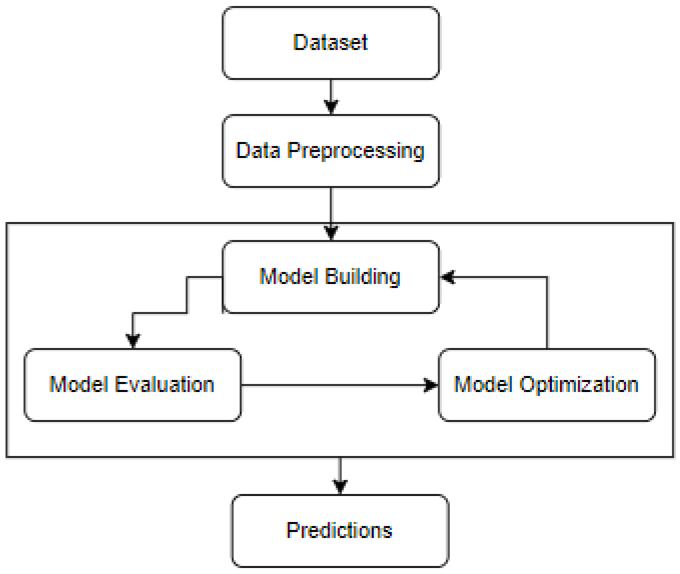


Figure 9: Model Architecture

V. Data Preprocessing

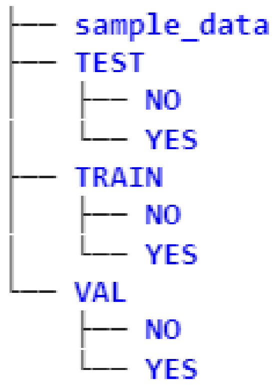
Under this phase there are a few steps that are performed to transform the dataset into the right format and shape to feed the model in the latter phases.

- *Train, Validation and Testing Sets*

  We have created three separate directories/folders for the dataset. These are as follows:

  - Training Directory: This directory contains images from both classes and is used to train the model.
  - Validation Directory: This directory contains a smaller subset of data instances that are used to evaluate model performance during the training phase, however the model does not use them to train. It is after validation results are obtained; can one begin the process of optimization.

- Testing Directory: This contains another small subset of data that is hidden from the model and is only employed to evaluate its performance once trained.



```
├── sample_data
├── TEST
│   ├── NO
│   └── YES
├── TRAIN
│   ├── NO
│   └── YES
└── VAL
    ├── NO
    └── YES
```

Figure 10: File Structure

- *Data Augmentation*

Data augmentation is a set of techniques that enhance the size and quality of machine learning training datasets so that better deep learning models can be trained with them. Data Augmentation artificially inflates datasets using label-preserving data transformations. [11]

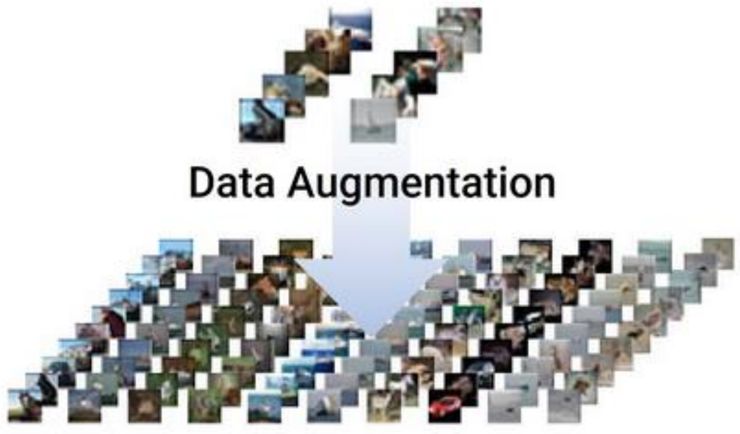


Figure 11: Data Augmentation

To build useful deep learning models, Data Augmentation is a very powerful method to reduce overfitting by providing a more comprehensive set of possible data points to minimize the distance between the training and testing sets.

The underlying idea is that more information can be gained from the original image dataset through the creation of augmentations. These augmentations artificially inflate the training dataset size by data warping or oversampling. We have employed data warping techniques which involve transforming existing images while preserving their label (annotated information). This includes augmentations such as geometric and color transformations, rotations, rescaling, enlarging etc.

In this regard, we have employed the use of *ImageDataGenerator* class from Keras, which allows the users to perform image augmentation on the fly in a very easy way. The file structure for our binary classification problem is as follows:

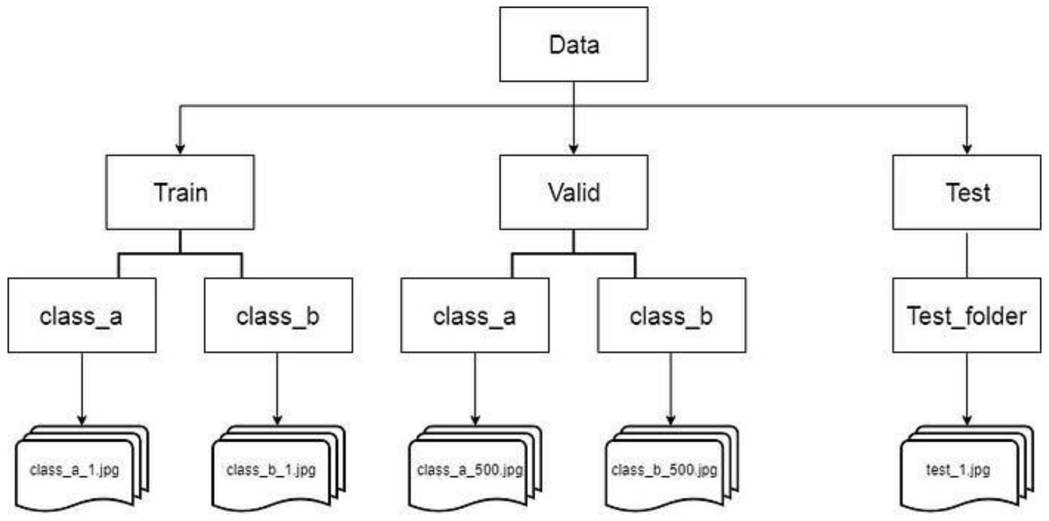


Figure 12: File structure for binary classification

- *Feature Scaling*

Normalization is applied when you have features with different units and scales. It curtails the range of values to be between 0-1. This is done to effectively remove bias which can arise due to features having different scales and thus impacting the overall prediction. Normalization applies the following formula to each numerical feature for scaling:

$$X_{new} = \frac{(x_i - x_{min})}{x_{max} - x_{min}}$$

Where:

• x_i: The i_{th} value of the dataset.

• x_{min}: The minimum value in the dataset.

• x_{max}: The maximum value in the dataset.

In this case, normalization enables efficient calculation and processing of the image arrays when fed to the model for learning weights and extracting features.

VI. Model Building and Evaluation

Under this section, once the dataset has been refined and preprocessed into its final form, we feed the data to the model for training and subsequent evaluation. We have employed a host of frameworks and libraries in this regard.

- Language – Python
- Integrated Development Environment (IDE) – Google Colab. Allows the writing and execution of arbitrary python code on the browser for machine learning and data science applications.
- Numpy & Pandas – Libraries used for data manipulation and analysis.
- OpenCV – Computer vision and image processing library.
- Matplotlib & Seaborn – Plotting and data visualization library.
- Sklearn – Machine learning library
- Tensorflow & Keras – Machine learning and in-particular deep learning library.

The device employed for the purpose of executing and rendering code is an Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz, 12 GB installed RAM, 64-bit operating system, and Windows 11 Education operating system.

Evaluation metrics are used to measure the quality of the statistical or machine learning model. In this regard, for our binary classification problem, we have attempted to use a few metrics to gauge the performance of the various models.

- *Accuracy*

  Accuracy is one of the metrics employed for evaluating classification models. It is represented as:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

  In addition, binary classification accuracy can also be expressed in terms positives and negatives:

$$Accuracy = \frac{TP + TN}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

- *Confusion Matrix*

It is a table that visualizes the model's performance. It shows all the possible labels and how the model can correctly or incorrectly predict them. There are four quadrants in the matrix and our explained as follows:

- True Positive (TP): Positively labeled instance that the model correctly predicts as positive.
- False Positive (FP): Negatively labeled instance that the model incorrectly predicts as positive.
- False Negative (FN): Positively labeled instance that the model incorrectly predicts as positive.
- True Negative (TN): Negatively labeled instance that the model correctly predicts as negative.

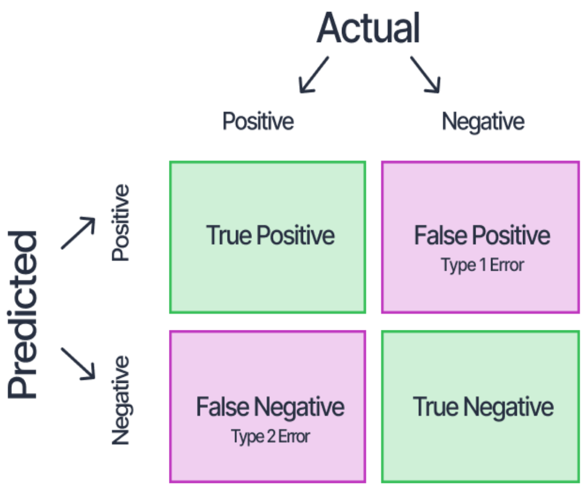


Figure 13: Confusion Matrix

- *BinaryCrossEntropy*

A loss function is a measurement between the algorithm's current output and expected output. It enables us to evaluate its performance.

Binary cross entropy is a loss function that is employed for binary classification problems. It compares the predicted probabilities with the actual class output to compute the penalty score that penalizes probabilities based on their distance from the actual values.

Results & Analysis

In this section, we discuss the results that have been achieved following the model building and evaluation phase. We have employed three deep convolutional neural networks that are considered benchmark models in the field of computer vision and in particular image classification. We will address each of these and give an overview of their characteristics.

- **ResNet50V2**

  The ResNet50V2 is a deep learning architecture that has been trained on acclaimed '*Imagenet'* dataset, containing more than a million images belonging to 1000 different classes. The architecture is 50 layers deep, comprising of several convolutional, pooling, and fully connected layers. We import the pre-trained version of the model and append layers to it for our project.

- **VGG19**

  VGG19 is another deep learning convolutional neural network that consists of 19 layers. It has a proven track record in the domain of image classification assignments and is considered a benchmark model. The weights for this architecture are also trained using the '*Imagenet'* dataset.

- **VGG16**

  A variation of the VGG19, comprising of 16 layers. It's unique in that it has only 16 layers that have weights, as opposed to relying on many hyper-parameters. It's considered one of the best computer vision model architectures.

Now we look at how each of these models have performed under different scenarios using the evaluation metrics stated earlier and discuss their comparative performance. The results that have been obtained are used to develop an objective view of the overall efficacy of the architecture employed in this project.

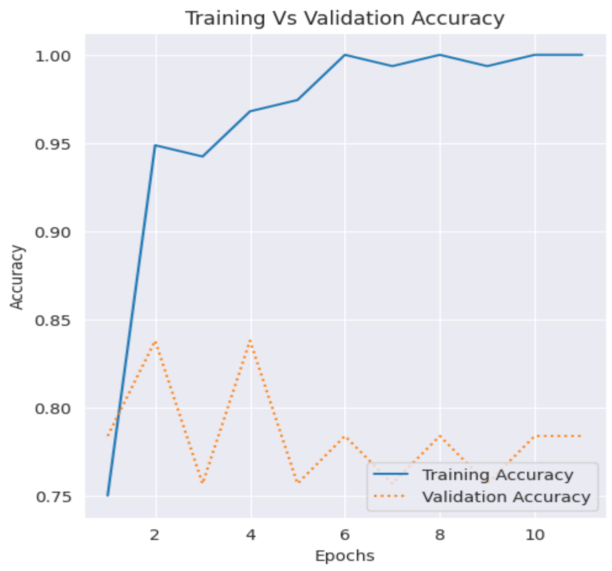
Our dataset has been split in the following manner:

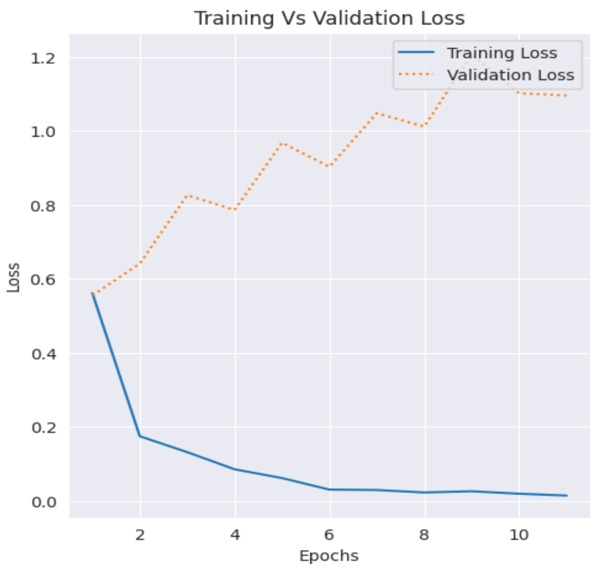
Table 2: Dataset Split

Training Set	156 images
Validation Set	37 images
Testing Set	10 images

ResNet50V2

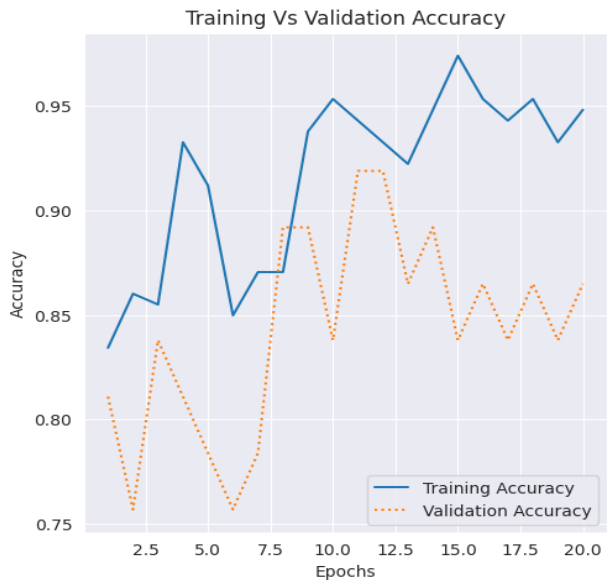
Set	**Accuracy**	**Loss**
Training Set	1.000	0.0244
Validation Set	0.7838	0.9029
Testing Set	0.9000	0.2301

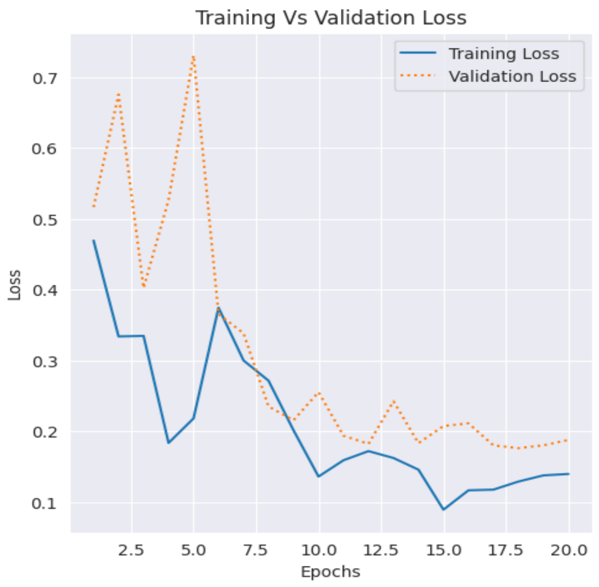




ResNet50V2 – With Augmented Data

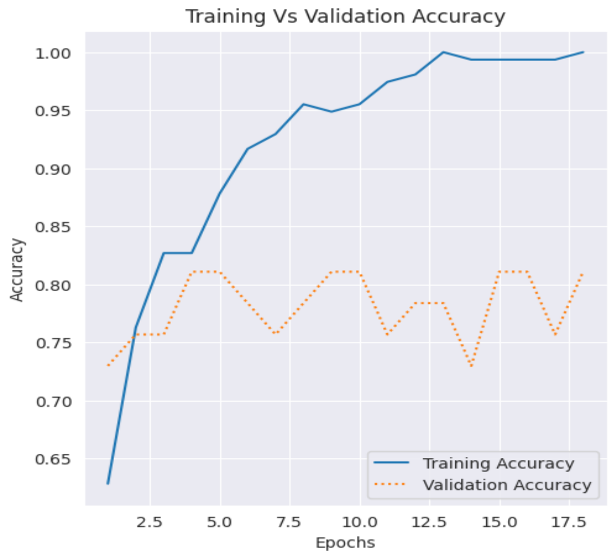
Set	**Accuracy**	**Loss**
Training Set	0.9808	0.0605
Validation Set	0.9189	0.1939
Testing Set	0.9000	0.1222

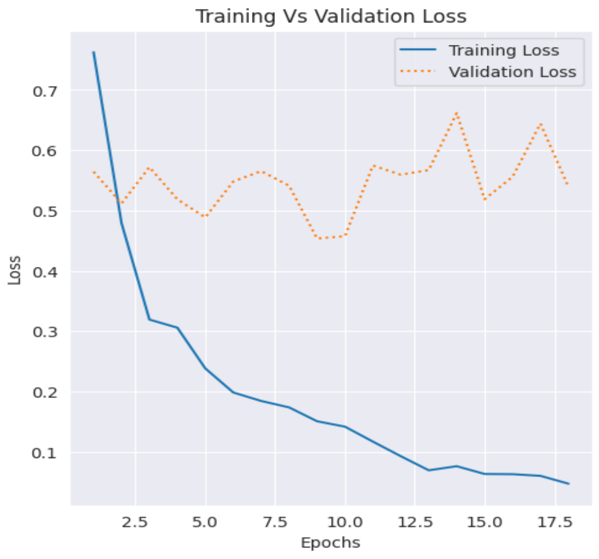




VGG19

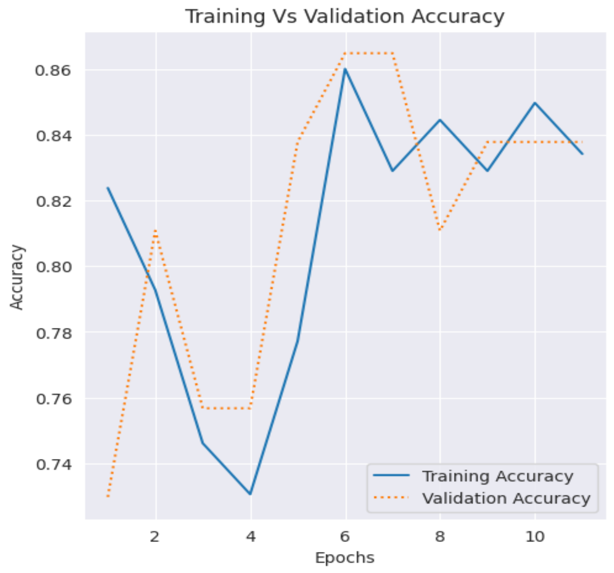
Set	Accuracy	Loss
Training Set	0.9808	0.1305
Validation Set	0.7838	0.5260
Testing Set	0.8000	0.5391

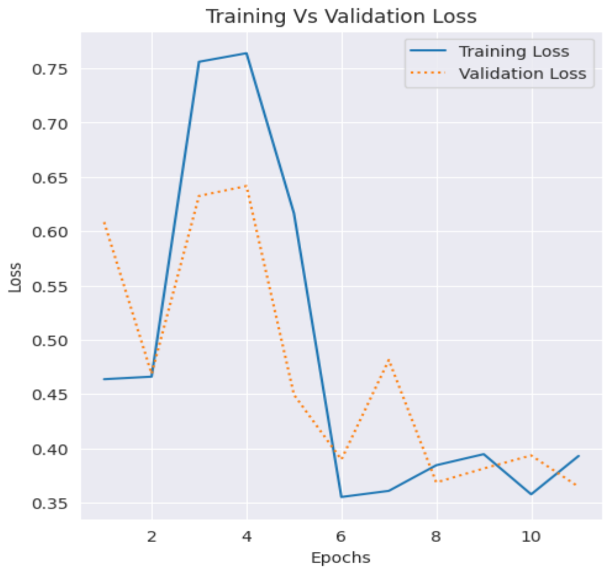




VGG19 – With Augmented Data

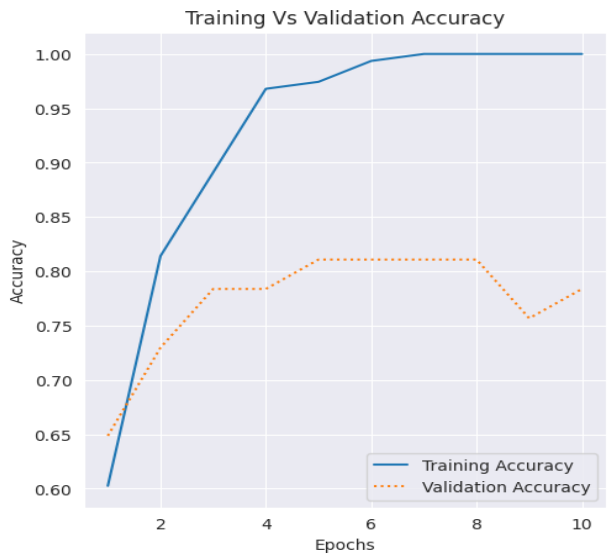
Set	Accuracy	Loss
Training Set	0.8290	0.4438
Validation Set	0.8649	0.3897
Testing Set	0.8000	0.7549

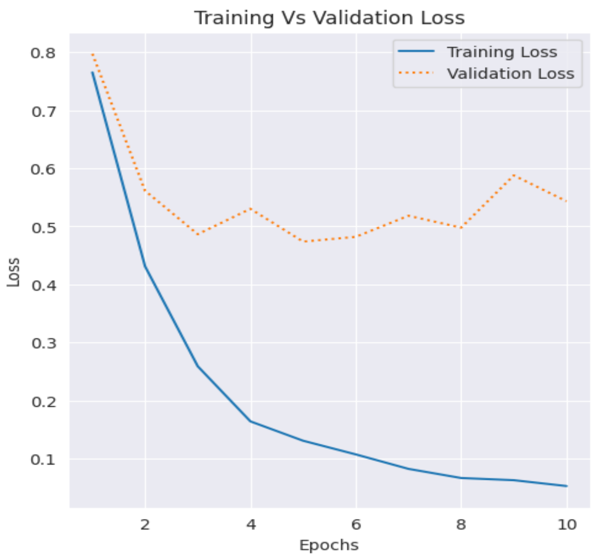




VGG16

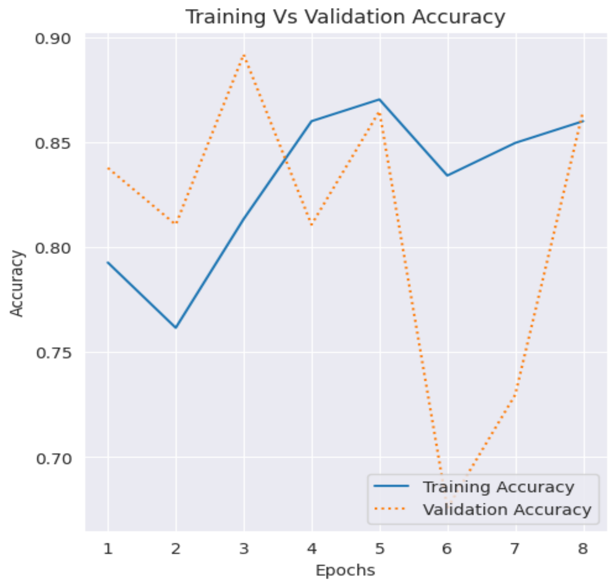
Set	**Accuracy**	**Loss**
Training Set	1.0000	0.0701
Validation Set	0.8108	0.5183
Testing Set	0.8000	0.4695





VGG16 – With Augmented Data

Set	**Accuracy**	**Loss**
Training Set	0.9936	0.0697
Validation Set	0.8649	0.4349
Testing Set	0.8000	0.5182



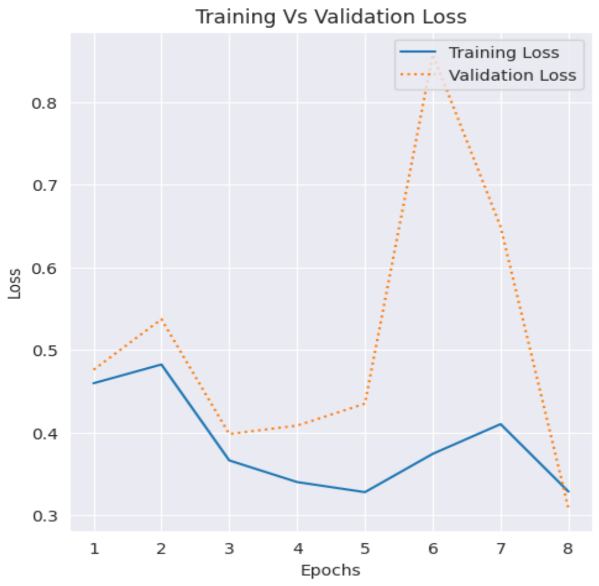


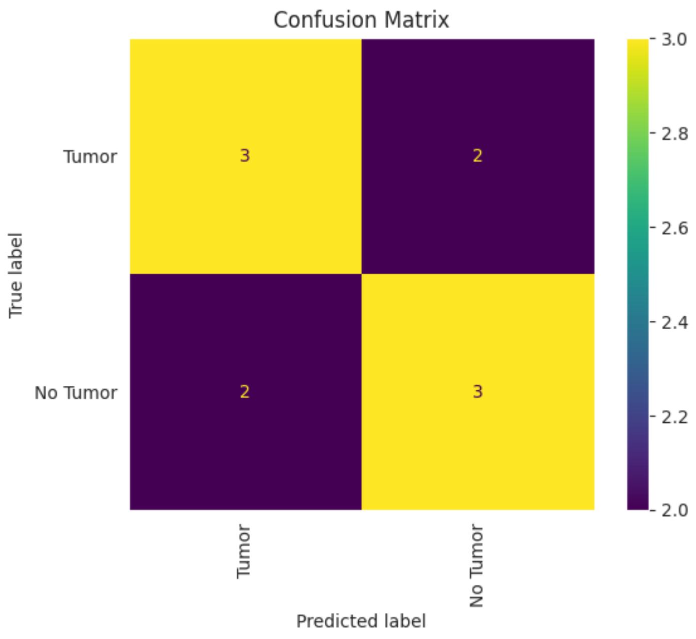
Table 3: Model Hyperparameters

Optimizer	**Loss**	**Epochs**	**Activations**	**Learning Rate**	**Early Stopping**
Adam	BinaryCrossEntropy	30	ReLu + Sigmoid	0.001	Yes

All models and their variations have a few constant hyperparameter values. This is to ensure equitable comparison between models and to facilitate deductions.

There are several observations that have been deduced from the results achieved. We will list them below:

- In terms of loss, we observe that for each model, the training set loss is the least, followed by the validation set and then the testing set. In addition, we also observe that the loss for augmented dataset as opposed to the original dataset is less. This is observed for all models. Finally, the optimum loss reported is by the Fully Connected VGG16 model, which has a loss of just 0.0035 for the testing set.
- In terms of accuracy, once again a similar trend is observed when viewing the results for three data subsets, with the highest accuracy reported for the training set followed by the validation and testing set. In addition, accuracy improves when we employ the augmented data set for model training, however the fully connected model versions have accuracy at best on par or less than the standard deep learning architecture. Finally, the optimum accuracy recorded is 90% for the ResNet50V2 standard and augmented version of the model.
- The confusion matrix has similar results for all three models. We have equal values for TP and TN, which is a promising sign. However, all three models report some inaccuracy and incorrectly classify both non-tumor and tumor cases. The adjacent figure depicts the confusion matrix obtained for all models. We observe that since the testing set is rather small i.e., only 10 images, the confusion matrix is not well reflective of the overall learnability of the models.



- The accuracy and loss curves depict that all models when working with the unadulterated dataset have some level of overfitting, where the models perform exceedingly well with the training set whilst underfit with the validation sets. This is drastically reduced when working with the augmented dataset. This shows that the impact of data preprocessing on the model performance is significant and helps achieve generalizability.
- Although, we employ the concept of transfer learning which translates into using pre-trained weights for our model and only training weights for the new layers appended; however, due to the large number of learnable parameters involved, the training time is very large i.e., 25-30 mins per model. This can be a challenge if we want to scale our model to large datasets or train for more epochs.
- Similarly, model complexity is another impending issue as we have thousands of neuron connections and associated weights for each layer. We can observe that the number of learnable parameters for the fully convolutional models is less than those of the standard convolutional model. This is because fully connected layers are now replaced with convolutional layers, which only have regional connectivity per neuron. Of course, this means a decrease in trainable parameters, but research shows that this can be beneficial, especially since we have a simpler error surface, leading to higher accuracy.
- This brings us to a comparative analysis between the proposed FCN architecture versus the existing CNN architecture. Although, (in theory) the FCN model can retain greater information and should expectedly perform better in terms of classification accuracy and reported loss. However, based on the results achieved, it is observed that this is not entirely true; the evaluation metrics when compared are at best on par with those of the CNN architecture. One reason for this can be the nature of the dataset employed. Whilst, in the original paper, the authors worked with a large dataset of high resolution multi-spectral images, we only have a small dataset with slight inaccuracies.
- Finally, we monitor the overall generalizability of the models. Based on the results achieved for training and validation sets, we can deduce that the model has performed considerably well.
- Given the obvious advantages of this methodology, we highlight some shortcomings of this approach:
- Memory limitations: CNNs can have millions of parameters and training them on large datasets can require a significant amount of memory.
- Slow training times: Training a deep CNN on a large dataset can take a long time, especially if the model is complex and the dataset is diverse. This can be an issue if the training process needs to be repeated multiple times to optimize the model.
- Model complexity: Deep learning models leverage a large number of trainable parameters i.e., weights and bias, to produce highly accurate results. However, this is at the cost of high model complexity.

- Overfitting: Overfitting occurs when the model is too complex and memorizes the training data instead of learning the underlying patterns. Since we face the issue of limited dataset, this aggravates the issue of overfitting.
- Hyperparameter tuning: CNNs have several hyperparameters, such as learning rate, batch size, and the number of layers, that need to be tuned to achieve optimal performance. Tuning these hyperparameters can be time-consuming and computationally expensive.

Table 4: Comparative Performances

Model	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**	**Loss (BinaryCrossEntropy)**
ResNet50V2	**100%**	78.38%	**90%**	0.2301
ResNet50V2 (Augmented Data)	98.08%	**91.89%**	**90%**	0.1222
VGG19	98.08%	78.38%	80%	0.5391
VGG19 (Augmented Data)	82.90%	86.49%	80%	0.7549
VGG16	**100%**	81.08%	80%	0.4695
VGG16 (Augmented Data)	99.36%	86.49%	80%	0.5182

- ResNet50V2 with standard and augmented training data has the highest test set classification score of 90%.
- ResNet50V2 with augmented data has the highest validation classification accuracy.
- VGG16 – Fully convolutional has the least BinaryCrossEntropy loss of 0.0035.

Conclusion

To conclude, we started by selecting a novel dataset for the chosen methodology. The brain MRI scan images serve as a great platform to test the proposed architecture. Our dataset is small and has certain inconsistencies. We follow this up performing data preprocessing to transform our dataset into the right shape and structure.

This is followed by the model building phase, wherein we define the architecture of our model by incrementing existing pre-trained models with custom layers under the premise of transfer learning/fine tuning. We leverage from the intuition that low level features are common to various datasets and that high-level features can be learnt by the added layers, allowing for greater efficiency and less computational complexity. In addition, we also convert the standard deep convolutional neural networks to fully convolutional neural networks, enabling the application of the proposed architecture in the original paper to our problem.

We then apply three different benchmark architectures to our dataset and evaluate them based on several metrics. The results that have been obtained reveal that the initial hypothesis stands true. A comparative analysis between these models reveals that better results can be achieved relying on the findings of [2]. However, this is not to say that there do not exist issues and problems. We have identified numerous such issues with this approach: Memory limitations, Slow training times, Overfitting, Model Complexity, Hyperparameter tuning etc.

Finally, we conclude by suggesting ground for future work. Our suggestions are that there is potential for fully convolutional neural networks, especially in relation to the domain of image segmentation and classification. This architecture enables dense pixel-wise classification enabling greater scrutiny of images irrespective of their location. Our project helps clear the space for other datasets for a variety of domains. However, the limitations identified pose challenges for scalability and efficiency, which requires attention.

References

[1] M Bharath Simha Reddy and Pooja Rana 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012020

[2] Maggiori, E. *et al.* (2017) "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), pp. 645–657. Available at: https://doi.org/10.1109/tgrs.2016.2612821.

[3] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2013.

[4] Afshar P, Mohammadi A, Plataniotis KN. Brain tumor type classification via capsule networks. In: 2018 25th IEEE international conference on image processing (ICIP). IEEE. 2018. p. 3129–33.

[5] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C. S., Liang, H., Baxter, S. L., Zhang, K. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. Cell, 172(5), 1122- 1131.e9. https://doi.org/10.1016/j.cell.2018.02.010

[6] Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., … Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics, and Transfer Learning. IEEE Transactions on Medical Imaging, 35(5), 1285–1298. https://doi.org/10.1109/TMI.2016.2528162

[7] Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., & Salem, A.-B. M. (2018). Classification using deep learning neural networks for brain tumors. Future Computing and Informatics Journal, 3(1), 68–71. https://doi.org/10.1016/j.fcij.2017.12.001

[8] Cui, Z., Yang, J., & Qiao, Y. (2016). Brain MRI segmentation with a patch-based CNN approach. Chinese Control Conference, CCC, 2016-August, 7026–7031. https://doi.org/10.1109/ChiCC.2016.7554465

[9] Russo, M., Stella, M., Sikora, M., & Sari, M. (n.d.). CNN-based Method for Lung Cancer Detection in Whole Slide Histopathology Images. 14–17.

[10] CS231N convolutional neural networks for visual recognition. Available at: https://cs231n.github.io/convolutional-networks/ (Accessed: April 29, 2023).

[11] https://viso.ai/computer-vision/image-data-augmentation-for-computer-vision/

[12] https://www.kaggle.com/code/mdmosarrofhossen/brain-tumor-detection-with-resnet50-vgg-inception/notebook