Deep Learning

Brain Tumor Classification

Group 1: Medical Image Classifiers
Anulitha Vardhan
Carsten Lingemann
Sundari Kala

GitHub - anulithavardhan/AI_Deep-learning_CNN-for-medical-images

DSBA 6165: Artificial Intelligence and Deep Learning
UNC Charlotte - School of Data Science
Spring 2023

1 Introduction

In this paper, we present a novel approach to medical image classification using three popular convolutional neural network models: ResNet-50, VGG-16, and Inception-V3. Our goal is to develop a unified model that can accurately classify Brain MRI tumor classification, Leukemia classification, and Chest X-ray classification for diseases. Medical images analysis can be a very useful research idea and can essentially change the course of early-diagnosis. While AI has shown promise in aiding medical professionals in disease diagnosis, there are still challenges that need to be addressed, such as limited access to large, diverse datasets, biases in the data used to train models, and ethical considerations regarding the use of AI in healthcare. To achieve our goal, we used existing research and attempted to make the models better, made use of pre-trained models & incorporated additional techniques such as data augmentation and transfer learning.

2 Background / Related Work

We identified research papers on 3 different pre-trained models: VGG-16, ResNet-50, and Inception-V3. The sections below outline the details from each approach.

2.1 VGG-16

The existing method from Khan et al. [1] intended to build a model that will work with 2 Brain Tumor datasets: a multiclass dataset with about 3000 images and 3 classes of tumors (Meningioma, Glioma, and Pituitary), and a binary class dataset with about 150 images and 2 classes (normal and abnormal).

They started with a custom 23-layer CNN architecture depicted below in Figure 1 on the multiclass Brain Tumor dataset.

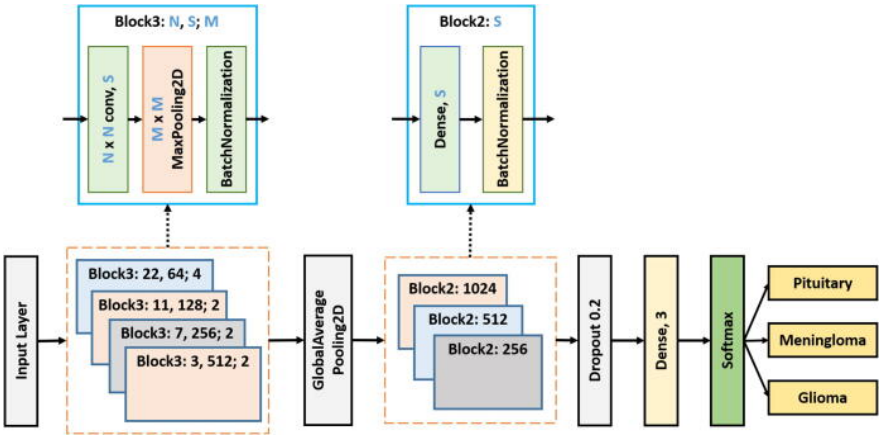

Figure 1: Proposed 23-layer CNN architecture [1]

They achieved very strong results with 97.8% accuracy on the multi-class Brain Tumor image data set. However, they did not achieve very good results on the binary class dataset due to overfitting with the limited volume of data [1]. To resolve this, they leveraged transfer learning with a VGG-16 architecture as depicted below in Figure 2.

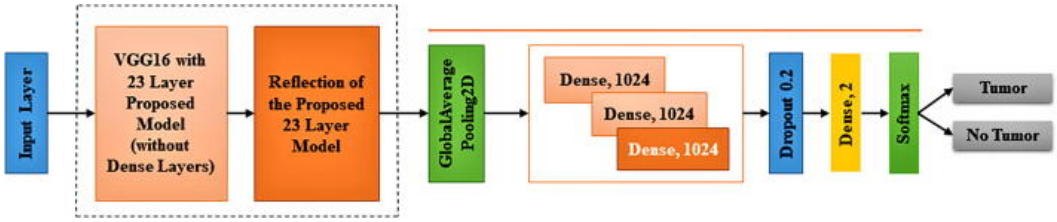

Figure 2: Proposed transfer learning based VGG-16 architecture [1]

They achieved a stellar 100% accuracy on this model with no signs of overfitting [1], however they were not able to leverage the same model for both datasets. In looking at the code, I didn't see any references to the custom 23-layer CNN architecture that was used for the multiclass dataset. I also observed that the code that was used to load and augment the data has been deprecated. Table 1 below outlines the pros and cons of the approaches that were used by these authors.

Table 1: Pros and cons for VGG-16

PROS	CONS
• Relatively simple architecture • Excellent results with the 2 different models	• Unable to use the same model on multiple datasets • Uses deprecated methods for loading data and performing data augmentation

For my VGG-16 based approach, I started with the transfer learning based VGG-16 architecture and updated it to replace any deprecated methods with the latest methods. We were also able to find a multiclass brain tumor image dataset with 7000 images, instead of just the 3000 that was used in this paper.

2.2 ResNet-50

The existing approach by Hussain [2] used ResNet-50 to extract features of pre-preprocessed (normalized and resized) images, and were fed into a feature selection algorithm. Selected features were used to train a SVM classifier. Learning rate, weight decay, and optimizers were used to improve performance. It used Grad-CAM to visualize the regions of the image that contributed the most towards the model's prediction. After pre-processing and image augmentation, ResNet-50 was trained on the dataset along with other pre-trained models like EfficientNetB1, EfficientNetB7, and EfficientNetV2B1. However, the results showed that the EfficientNet models outperformed ResNet-50 in terms of accuracy and loss. The authors noted that one possible reason for ResNet-50's lower performance could be due to the vanishing gradient problem. This occurs when gradients in the earlier layers of a deep network become very small, making it difficult for the network to update the weights and improve performance during training. ResNet-50 uses identity mapping in its residual blocks to address this problem, which allows for the input from previous layers to be used without any change. However, the authors hypothesized that this may not be sufficient for the specific task of brain tumor classification, which requires the network to learn more complex features. Overall, the paper suggests that pre-trained models like ResNet-50 can be used for medical image classification tasks, but their performance may be limited by factors such as dataset size and complexity of the task. The authors also highlight the importance of using appropriate pre-processing and augmentation techniques to improve the performance of the

models. My model also involves using image augmentation, however, the papers related to ResNet-50 have not provided too many details on the exact architecture or the source code. Hence, I worked on the deep network architecture based on my own results.

Table 2 below outlines the pros and cons of the approaches that were used in this paper.

Table 2: Pros and cons for ResNet-50

PROS	CONS
• Converges quickly • Training an entire model can produce surprising results and is able to generalize to different datasets quite well	• Cannot work well with complex architectures, which might be required for some datasets • Very few models have utilized pure ResNet-50 models, instead using it only for feature extraction or as part of an ensemble model

2.3 Inception-V3

The existing article compared the accuracy of VGG-16 with the Inception-V3 model. In order to overcome the drawbacks of overfitting and updating the gradient weights is difficult with the deeper neural neural networks as in VGG-16, this article from Tamilarassi proposed going wider rather than deeper using Inception-V3 [3]. The existing article has used Brain MRI images and got the average accuracy of 95.1% using Inception-V3.

The existing article is using binary classification images. They predict whether the person's MRI image comes under normal or abnormal class.

The VGG-16 architecture used in the paper is as follows:

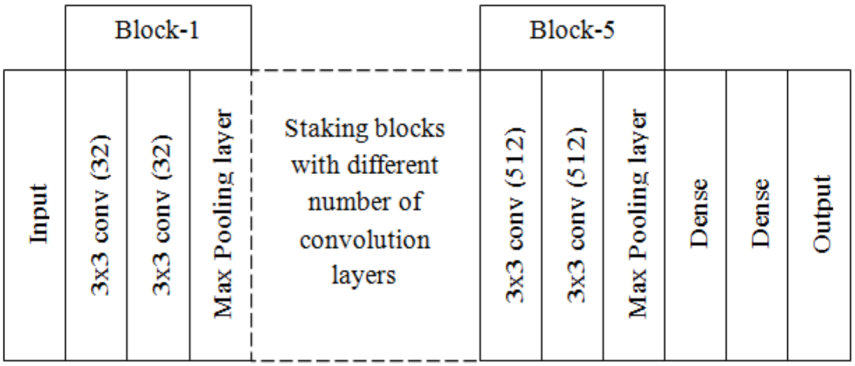

Figure 3: VGG-16 architecture

As in the Figure 3, the VGG-16 model is using smaller-sized kernels which are essential for extracting locally distributed information in the brain tumor MRI images. Using these smaller-sized kernels for the deeper and deeper layers in the VGG-16 model is computationally expensive. Thus, the paper proposed the Naive Inception Module (NIM) using the Inception-V3 architecture. The proposed NIM architecture is as follows:

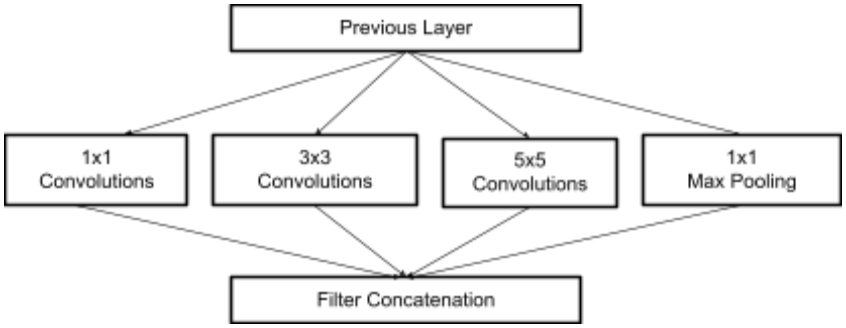

Figure 4: Naive Inception Module

Taking the above proposed NIM as the base model, the below Inception-V3 model is developed by stacking the NIM for the effective image classification of brain MRI images. Relu activation function is used for the dense layers, the sigmoid activation function is used for the output layer and the cross entropy as loss function.

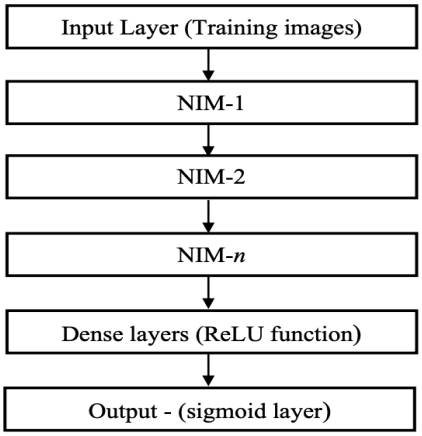

Figure 5: Stacked Inception-V3 Module

The paper has taken brain MRI images from 130 subjects with 100 images in each category (normal/abnormal). The Table 3 summarizes the training parameters for VGG-16 and Inception-V3 architures used in the paper

Table 3: Training parameters for VGG-16 and Inception-V3 architectures

Parameters	**Values**
Optimizer	Stochastic gradient descent
Loss function	Cross entropy loss
Activation function @ input layer	ReLU
Batch size	64
Learning rate	0.01
Iteration	20
Activation function @output layer	sigmoid

The average accuracy of the Inception-V3 and VGG-16 for 10 runs is compared in Table 4.

Table 4: Performances of VGG-16 and Inception-V3 on REMBRANDT MRI brain images

#Run	*CA* (%)	
	VGG-16	Inception-V3
1	91	93
2	95	93
3	92	94
4	92	95
5	94	94
6	93	96
7	88	96
8	90	**97**
9	96	97
10	**97**	96
Avg	**92.8**	**95.1**

It is inferred from Table 2 that the average Classification Accuracy (CA) of Inception-V3 architecture has a very slight improvement in the accuracy compared to VGG-16 architecture. However, the maximum CA obtained from VGG-16 on the 10th run is higher than Inception-V3.

Table 5 below outlines the pros and cons of the approaches that were used in this paper.

Table 5: Pros and cons for Inception-V3

PROS	**CONS**
● The model architectures are clearly explained. ● The parameters used are summarized in a table.	● The feature extractions used are not explained. ● The existing article has tested models on only brain MRI images. ● It is a binary classification problem. ● The predicted values of images were not given in the paper to determine how well the model predicted the images (normal/abnormal). The paper has only focused on the accuracy comparison between VGG-16 and Inception-V3.

For our model, we have used multi-class Brain Tumor MRI images. To test how well our model works for other datasets, we have used Chest X-Ray and Leukemia datasets.

3 Method

3.1 Data

In response to feedback that we received about trying to make the project more interesting, we decided that we wanted to try to build an image classification model with transfer learning that could effectively be used on multiple medical image datasets. We identified 3 datasets to use for this project, as outlined in Table 6 below.

Table 6: Datasets leveraged by this project

Dataset	Classes	# of Images
Brain Tumor [4]	• Glioma Tumor • Meningioma Tumor • Pituitary Tumor • No Tumor	7023
Leukemia [5]	• ALL - Acute Lymphoblastic Leukemia • HEM - Normal	8613
Chest X-Ray [6]	• COVID-19 • Pneumonia-Bacterial • Pneumonia-Viral • Normal	9208

The figures below display sample images from the Brain Tumor, Leukemia, and Chest X-Ray dataset.

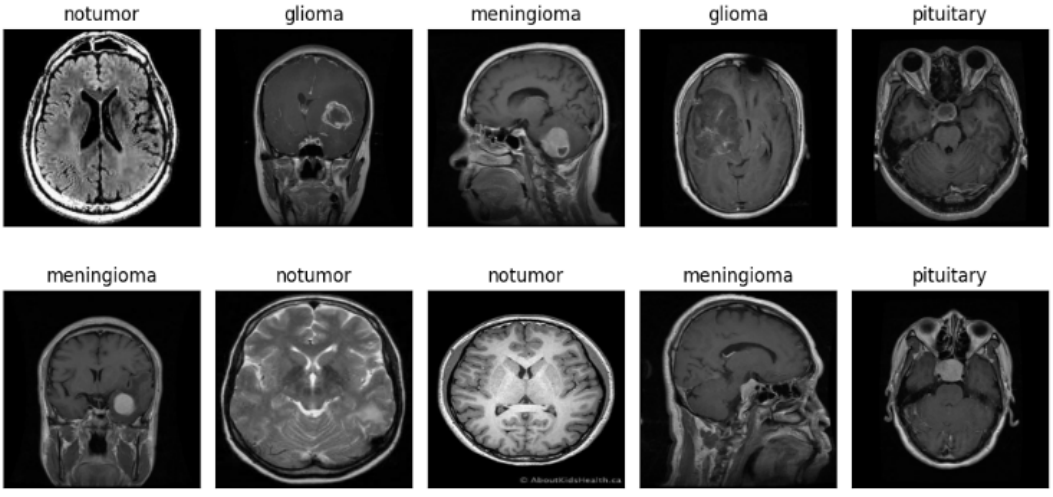

Figure 6: Sample brain tumor MRI images

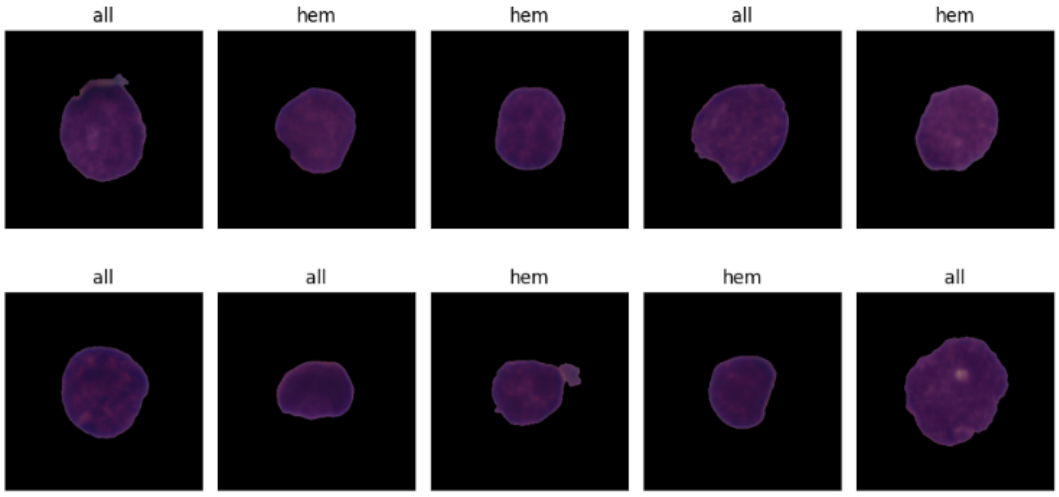

Figure 7: Sample Leukemia images

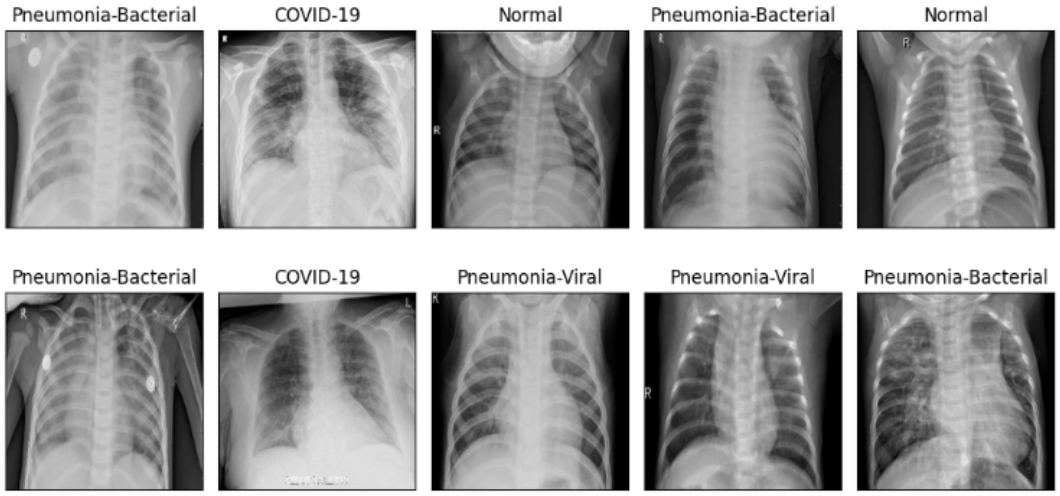

Figure 8: Sample chest X-Ray images

3.2 VGG-16

For our model, we started with the transfer learning based VGG-16 architecture that was created by Khan et al. [1] and used on the binary classification brain tumor image dataset referenced in their paper. It was implemented using deprecated methods such as ImageDataGenerator and flow_from_directory, so I first updated the process to use more current methods. The architecture that was used for our pre-trained VGG-16 model is shown below in Figure 9.

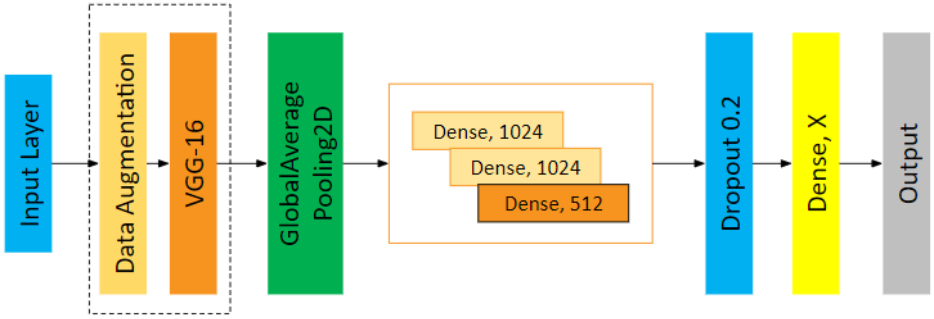

Figure 9: VGG-16 model architecture

For data augmentation, I used most of the augmentations that were used by Khan et al, as shown below in Figure 10.

```
train_datagen=ImageDataGenerator(
    rescale=1/255,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    width_shift_range=0.1,
    height_shift_range=0.1,
    vertical_flip=True,
    validation_split=0.2)
```

Figure 10: Khan et al. VGG-16 image augmentation [1]

I removed the rescale from our implementation of augmentation because I was unable to view a sample of the augmented images with this in there. I also removed the shear_range, as I was not able to find an equivalent for this in the functions that replaced ImageDataGenerator. I also bumped up the factors on the RandomZoom, RandomWidth, and RandomHeight. Figure 11 below shows our final implementation of image augmentation for the VGG-16 based model.

```
data_augmentation = keras.Sequential([
    layers.RandomZoom(0.3),
    layers.RandomFlip("horizontal"),
    layers.RandomWidth(0.2),
    layers.RandomHeight(0.2),
    layers.RandomFlip("vertical")
])
```

Figure 11: Our VGG-16 image augmentation

The figures below show examples of augmented images from each of the 3 datasets.

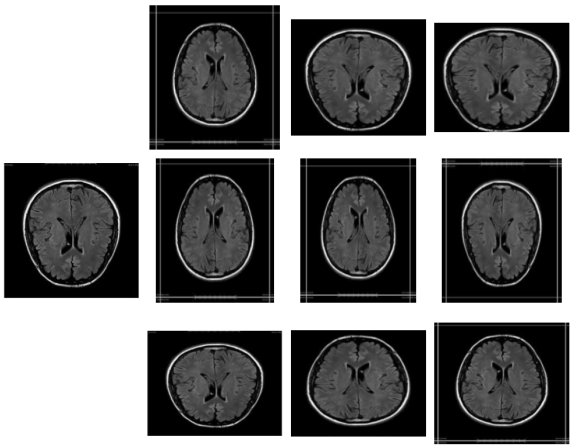

Figure 12: VGG-16 Brain Tumor augmented image example

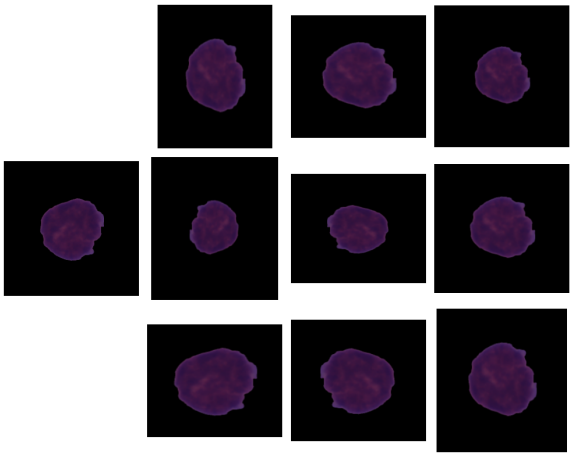

Figure 13: VGG-16 Leukemia augmented image example

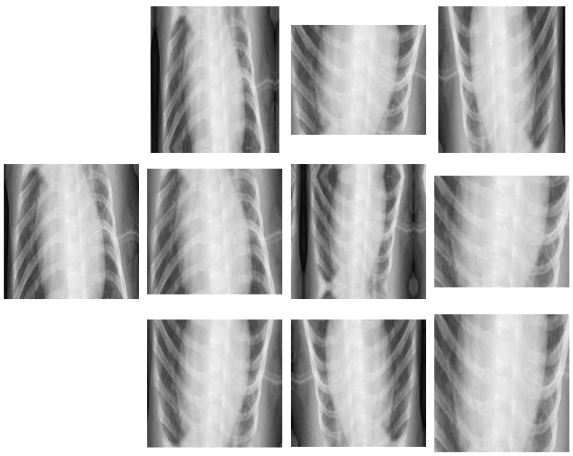

Figure 14: VGG-16 Chest X-Ray augmented image example

Since we used RandomWidth and RandomHeight in the image augmentation, we then needed to resize the images before executing the VGG-16 layers. Figure 15 below shows the model summary for the VGG-16 based transfer learning model.

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 224, 224, 3)]	0
sequential_1 (Sequential)	(None, None, None, 3)	0
resizing_1 (Resizing)	(None, 224, 224, 3)	0
tf.__operators__.getitem_1 (SlicingOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add_1 (TFOpLambda)	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 512)	0
dense_4 (Dense)	(None, 1024)	525312
dense_5 (Dense)	(None, 1024)	1049600
dense_6 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 4)	2052

Total params: 16,816,452
Trainable params: 2,101,764
Non-trainable params: 14,714,688

Figure 15: VGG-16 model summary

3.3 ResNet-50

This is a machine learning code in Python for training a model to classify images of brain tumors as either benign or malignant. The code uses the TensorFlow and Keras libraries to build a convolutional neural network (CNN) model, which is a type of deep learning algorithm that is commonly used for image classification.

The model architecture is based on the ResNet-50 model, which is a pre-trained CNN that has achieved state-of-the-art performance on various computer vision tasks. The ResNet-50 architecture has been modified and fine-tuned to suit the specific task of classifying brain tumor images. The code includes functions for displaying images and predictions, as well as for plotting the learning curves of the model during training.

The model's architecture is quite simple, however, the entire ResNet-50 model was trained from scratch instead of freezing the layers. A trial was done by freezing the layers and the lower layers of the network that extract basic features from the input image are frozen, while the higher layers that perform more complex computations and feature extraction are left trainable. However, this resulted in bad results, probably because a medical image dataset is significantly different from the pre-training dataset which includes weights from imagenet. If the pre-trained model was specifically designed for medical images, it could be a different story.

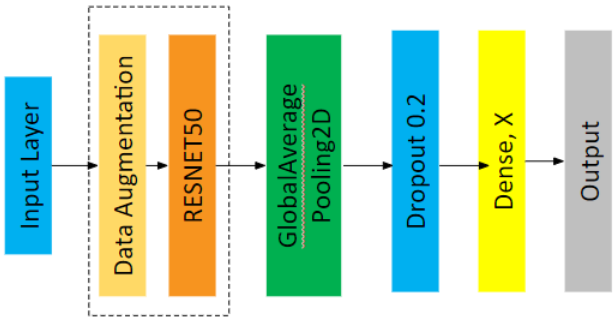

Figure 16: ResNet-50 proposed model architecture

The frozen layer model resulted in widely varying val accuracy and a very poor test accuracy. The model also showed signs of overfitting, due to the fact that the training accuracy was quite high while the val accuracy was significantly lower and had varying results in every EPOCH ranging from 40% to 80%.

The second problem observed was building a complex architecture, involving too many layers and the initial few architectures did not include batch normalization. This resulted in overfitting to the training data. Another issue that may arise with complex architectures is the computational cost and the time it takes to train the model. Complex architectures with many layers and parameters may require more computational resources, which can result in longer training times or may even be computationally infeasible on some hardware. This had a direct impact with running out of GPU quite easily due to the complexity of the architecture.

To handle the above problems, the architecture was simplified, including adding procedures that handle overfitting. Essentially the entire ResNet-50 model was trained without freezing layers, with early stopping, and LROnPlateau with only Global Average pooling layer, Dropout and Dense layers. LR on Plateau is a technique that dynamically adjusts the learning rate of the optimizer during training. It monitors the model's validation loss and reduces the learning rate if the validation loss stops improving. Early stopping involves monitoring the model's validation loss during training and stopping the training process when the validation loss stops decreasing.

Both early stopping and LR on Plateau are used to improve the performance of the model and prevent overfitting. By stopping the training process when the validation loss stops decreasing, and reducing the learning rate when the validation loss stops improving, these techniques help to prevent the model from becoming too complex and overfitting to the training data.

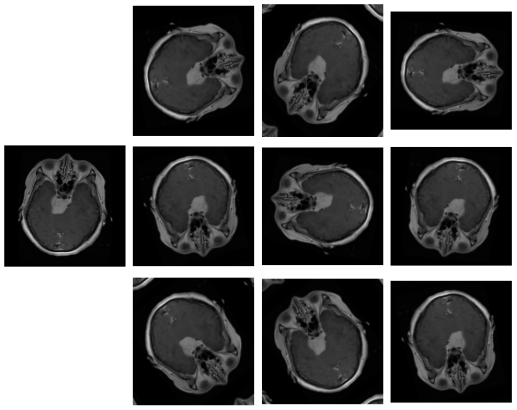

Figure 17: ResNet-50 Brain MRI Augmented Images

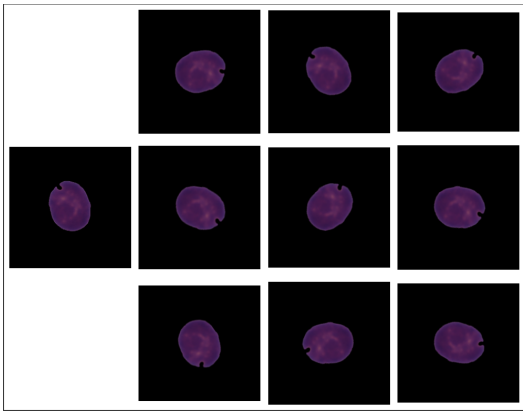

Figure 18: ResNet-50 Leukemia Augmented Images

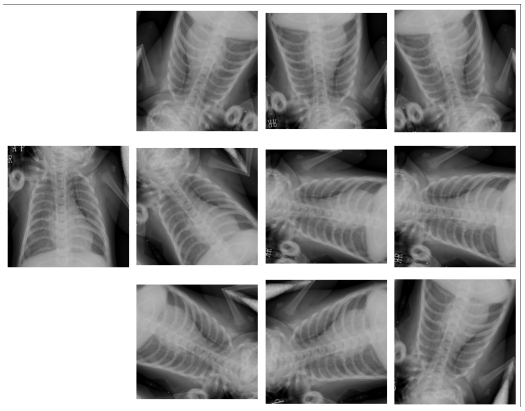

Figure 19: ResNet-50 Chest X-Ray Augmented Images

3.4 Inception-V3

As the existing paper didn't give the number of Naive Inception Modules stacked. I have used the pretrained Inception-V3 model for transfer learning. The image augmentation, the resizing of images are added as proposed in paper and Inception-V3 preprocessing layers are added between the Input layer and based pre-trained Inception-V3 model. Batch Normalization layer is added to overcome the overfitting problem. Global average pooling 2d layer is added, Flatten layer is added by taking the previous layer as input, a dense layer of 256 units with activation function as Relu is added on the previous layer. To keep the model simple, a dropout layer of 0.5 is added. Since our dataset is a multi-class(i.e., 4 class) image dataset, an output layer with 4 units is added with an activation function of softmax.

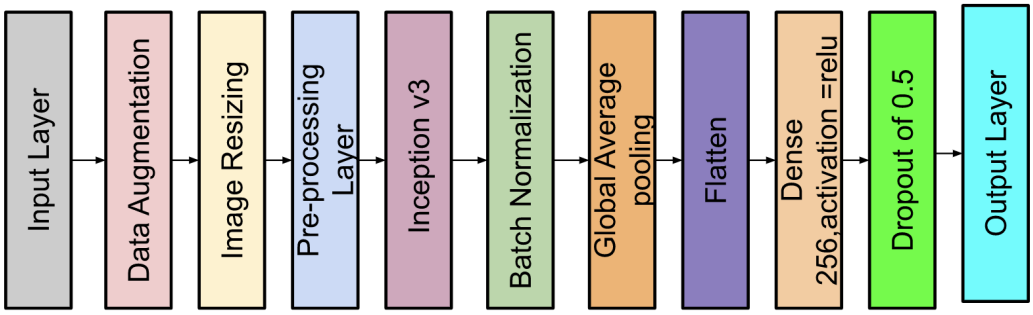

Figure 20: Inception-V3 proposed model architecture

For data augmentation as many details are not mentioned in the Tamilarassi [3] paper, I have referred to the other paper proposed by Ali Mohammad Alqudah et al. [7], in this paper he has used CNN model. Data Augmentation steps used in my model are presented in Figure 21.

```
data_augmentation = keras.Sequential([
    layers.RandomZoom(0.2),
    layers.RandomFlip("horizontal"),
    layers.RandomWidth(0.1),
    layers.RandomHeight(0.1),
    layers.RandomFlip("vertical"),
    layers.RandomCrop(180,180)
])
```

Figure 21: Data augmentation steps used in proposed model

The results obtained from the data augmentation layer for 3 different datasets are given in Figures 22, 23 and 24.

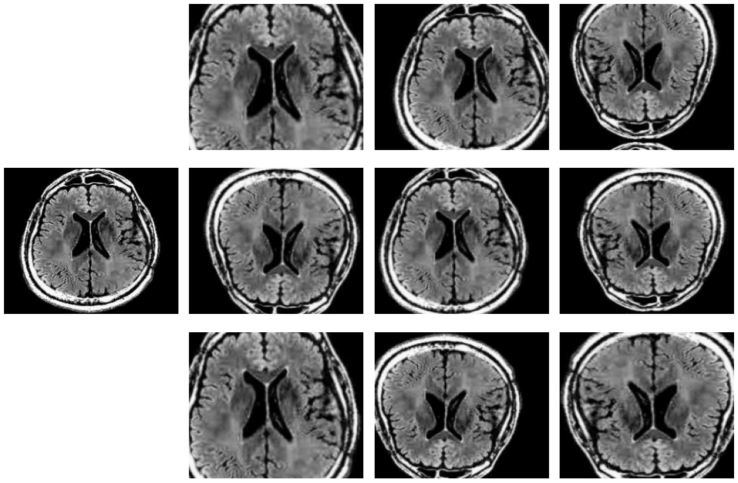

Figure 22: Brain MRI augmented images

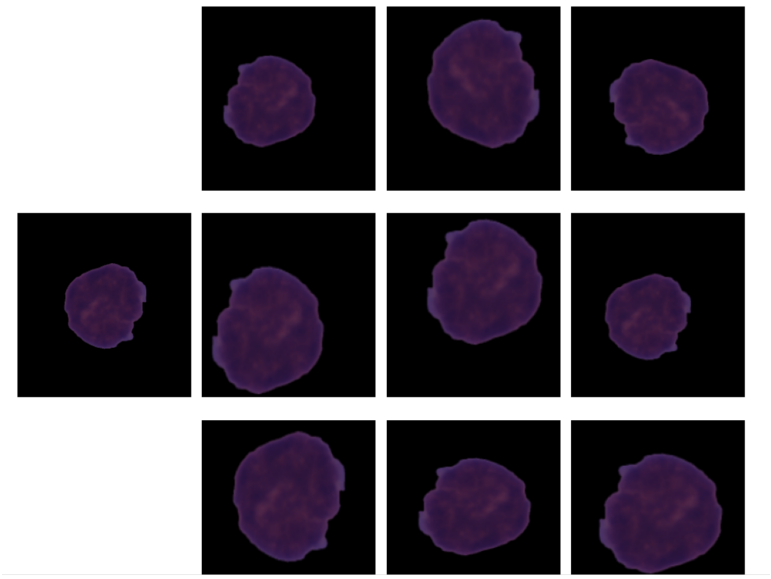

Figure 23: Leukemia augmented images

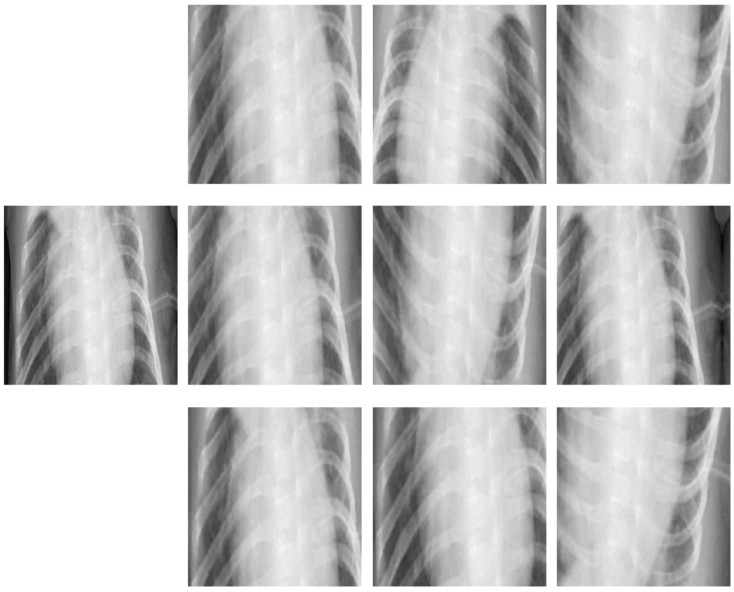

Figure 24: Chest X-ray augmented images

The proposed model summary is given in Figure 25. Adam is used as an optimizer and loss function is sparse_categorical_crossentropy.

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_13 (InputLayer)       [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 resizing_11 (Resizing)      (None, 224, 224, 3)       0

 tf.math.truediv_11 (TFOpLam  (None, 224, 224, 3)      0
 bda)

 tf.math.subtract_11 (TFOpLa  (None, 224, 224, 3)      0
 mbda)

 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 max_pooling2d_6 (MaxPooling  (None, 2, 2, 2048)       0
 2D)

 flatten_8 (Flatten)         (None, 8192)              0

 dense_14 (Dense)            (None, 512)               4194816

 dropout_10 (Dropout)        (None, 512)               0

 dense_15 (Dense)            (None, 4)                 2052

=================================================================
Total params: 25,999,652
Trainable params: 4,196,868
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 25: Inception-V3 proposed model summary

4 Experiments

Include: explanation of experimental setup, test results of proposed method, deep analysis/discussion about the results. Present all your journey! Do not skip your work to be presented because you think the results are poor.

4.1 VGG-16

4.1.1 Baseline Model

The baseline model for VGG-16 that was outlined in section 3.2 above was run on all 3 datasets with the following hyperparameters:

- Batch Size: 32
- Optimizer: Adam
- Loss: sparse_categorical_crossentropy
- Number of epochs: 20

Since the datasets have a different number of output classes, I created a function to build the model, and parameterized the number of classes to use in the final dense output layer. Table 7 below summarizes the test accuracy that we saw with this baseline model on all 3 datasets.

Table 7: VGG-16 baseline model test accuracy

Dataset	Test Accuracy
Brain Tumor	91.99%
Leukemia	69.20%
Chest X-Ray	84.19%

The figures below show graphs of loss and accuracy for each of the datasets.

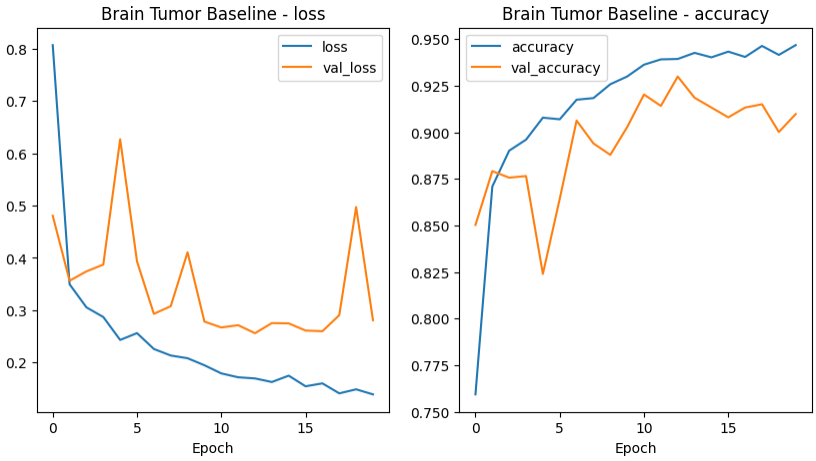

Figure 26: Brain Tumor VGG-16 baseline loss and accuracy

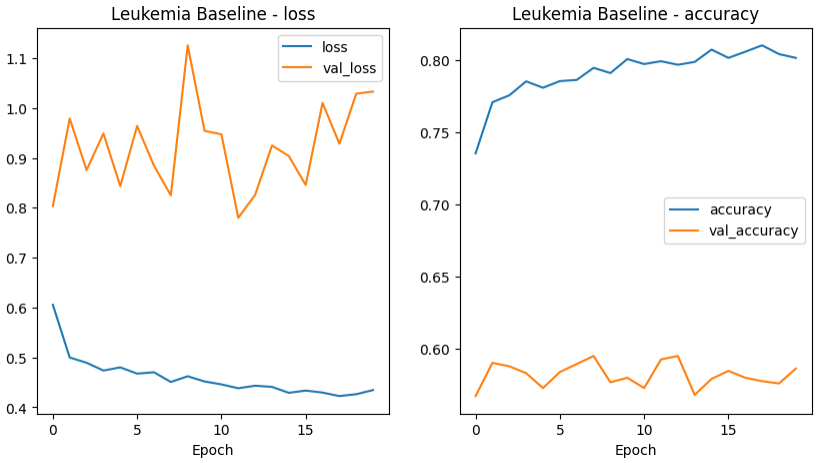

Figure 27: Leukemia VGG-16 baseline loss and accuracy

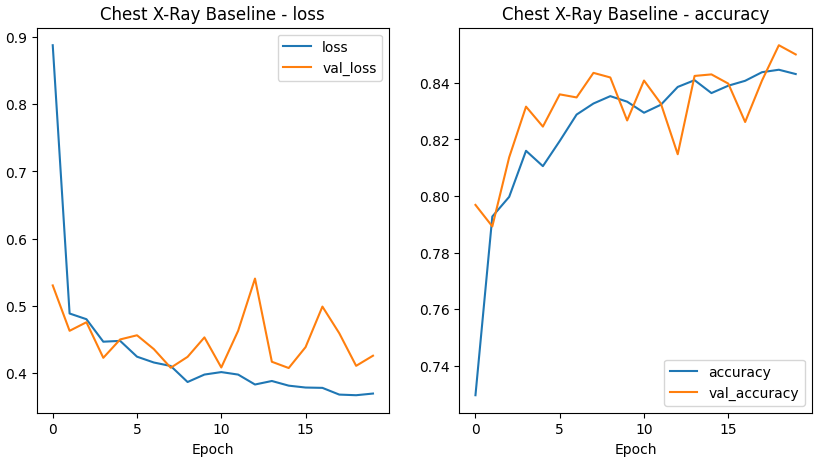

Figure 28: Chest X-Ray VGG-16 baseline loss and accuracy

4.1.2 Experiment 1 - binary classification

The first experiment was on the Leukemia binary classification model only, since the baseline results from the Brain Tumor dataset were pretty decent. I wanted to see if I could improve the binary classification by making the following changes:

- Changed the activation on the final output layer from softmax to sigmoid
- Changed the loss function from sparse_categorical_crossentropy to binary_crossentropy
- Changed the optimizer from Adam to RMSProp

This experiment resulted in a 74.27% test accuracy, which is about 5% higher than the baseline model with the Leukemia dataset. Figure 29 below shows a graph of the loss and accuracy from training the model for experiment 1.

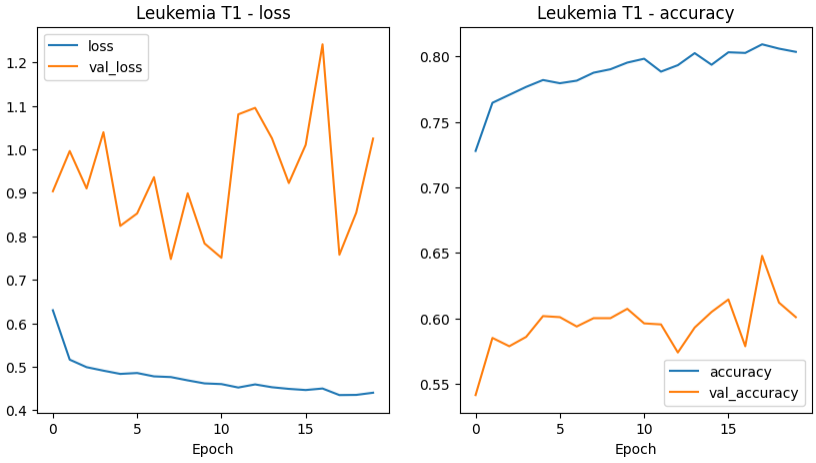

Figure 29: Leukemia VGG-16 loss and accuracy - experiment 1

4.1.3 Experiment 2 - binary classification

The second experiment was also conducted on just the Leukemia dataset as I wanted to see if I could improve the binary classification results again. I kept RMSProp as the optimizer as was done in experiment 1, but I changed the output layer activation function and loss function back to

what was used in the baseline (softmax and sparse_categorical_crossentropy). This experiment resulted in a 73.32% test accuracy, which is better than the baseline, but slightly less than experiment 1 using sigmoid activation and binary_crossentropy loss. Figure 30 below shows a graph of the loss and accuracy from training the model for experiment 1.

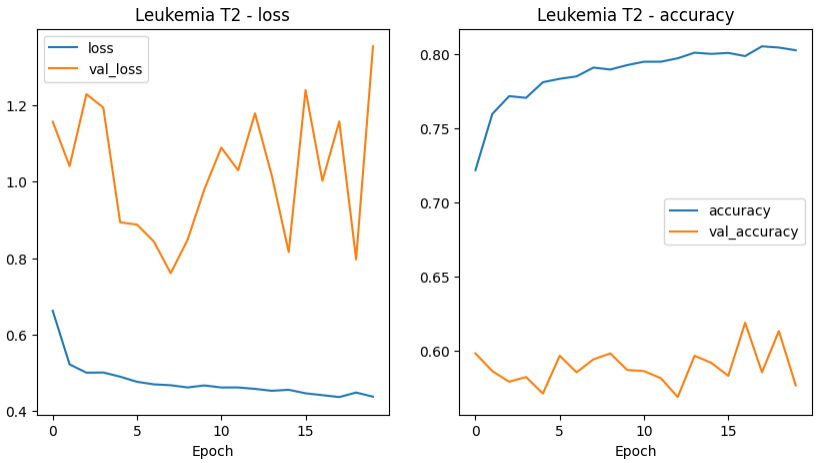

Figure 30: Leukemia VGG-16 loss and accuracy - experiment 2

4.1.4 Experiment 3 - batch normalization

For the next experiment, I enabled batch normalization in between each of the dense layers of the baseline model because my teammate said that she saw her performance improve by introducing batch normalization in the ResNet-50 based model. I ran this model on all 3 datasets. For the Leukemia dataset, I used the same activation function, loss function, and optimizer that were used in experiment 1, since that had the best results thus far (sigmoid, binary_crossentropy, RMSProp).

For the Brain Tumor dataset, this experiment resulted in a 91.53% test accuracy, which was slightly lower than the baseline of 91.99%. Figure 31 below shows a graph of the loss and accuracy from training the model for experiment 3 on the Brain Tumor dataset.

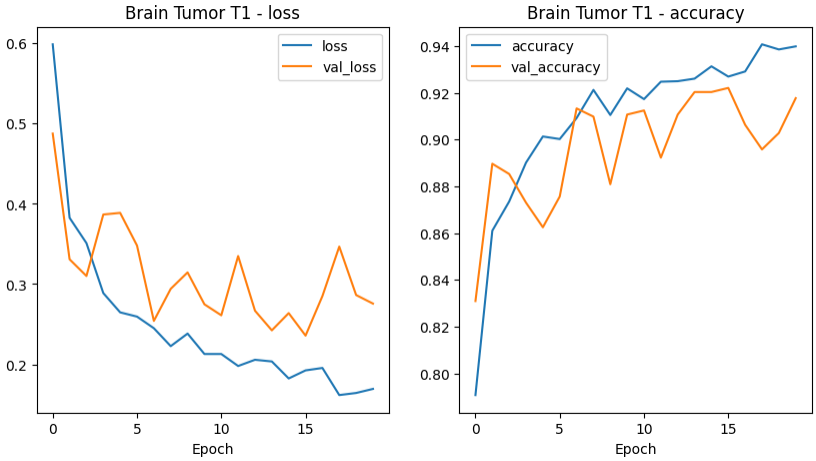

Figure 31: Brain Tumor VGG-16 loss and accuracy - experiment 3

For the Leukemia dataset, this experiment resulted in a 63.74% test accuracy, which is considerably lower than the baseline of 69.20%, and experiment 1 accuracy of 74.27%. Figure 32 below shows a graph of the loss and accuracy from training the model for experiment 3 on the Leukemia dataset.

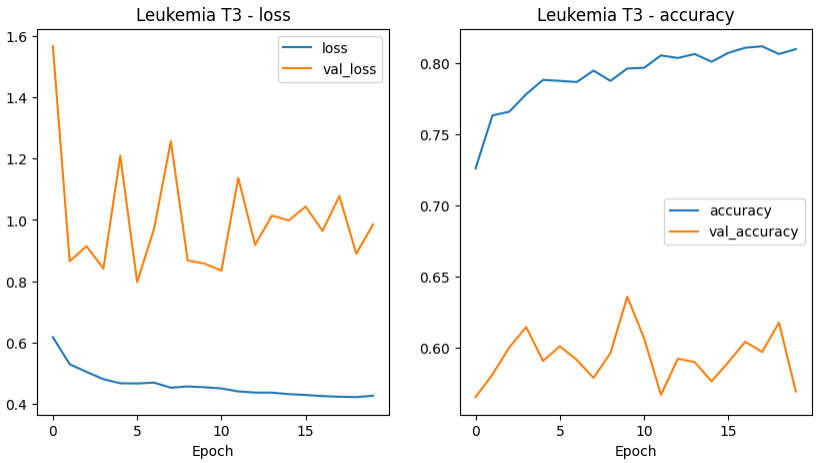

Figure 32: Leukemia VGG-16 loss and accuracy - experiment 3

For the Chest X-Ray dataset, this experiment resulted in a 83.57% test accuracy, which was slightly lower than the baseline of 84.19%. Figure 33 below shows a graph of the loss and accuracy from training the model for experiment 3 on the Chest X-Ray dataset.

Figure 33: Chest X-Ray VGG-16 loss and accuracy - experiment 3

4.1.4 Experiment 4 - batch normalization w/ 50 epochs

For the final experiment, I used the model from Experiment 3 with batch normalization, and increased the number of epochs from 20 to 50, because it looked like the general trend line on accuracy was increasing overall, and I wondered if accuracy would continue to improve if the model ran longer. I ran this on all 3 datasets. For the Leukemia dataset, I again used the same

activation function, loss function, and optimizer that were used in experiments 1 and 3 (sigmoid, binary_crossentropy, RMSProp).

For the Brain Tumor dataset, this experiment resulted in a 91.38% test accuracy, which was slightly lower than experiment 3 accuracy of 91.53%. Figure 34 below shows a graph of the loss and accuracy from training the model for experiment 4 on the Brain Tumor dataset.

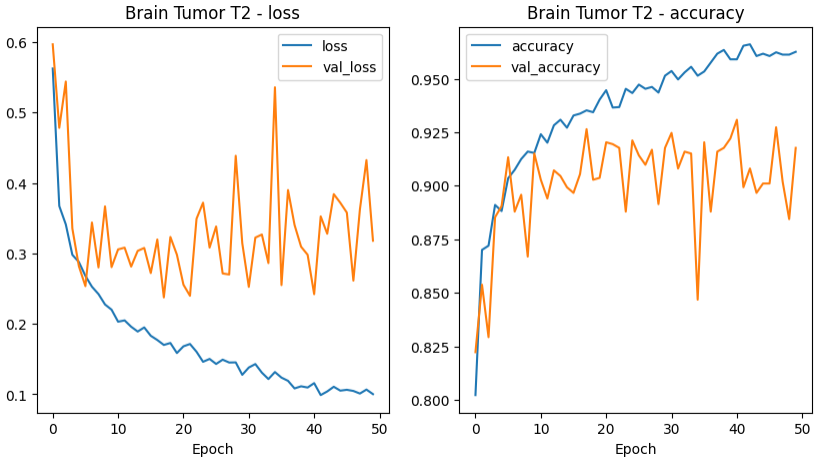

Figure 34: Brain Tumor VGG-16 loss and accuracy - experiment 4

For the Leukemia dataset, this experiment resulted in a 63.74% test accuracy, which is considerably lower than the baseline of 69.20%, and experiment 1 accuracy of 74.27%. Figure 35 below shows a graph of the loss and accuracy from training the model for experiment 4 on the Leukemia dataset.

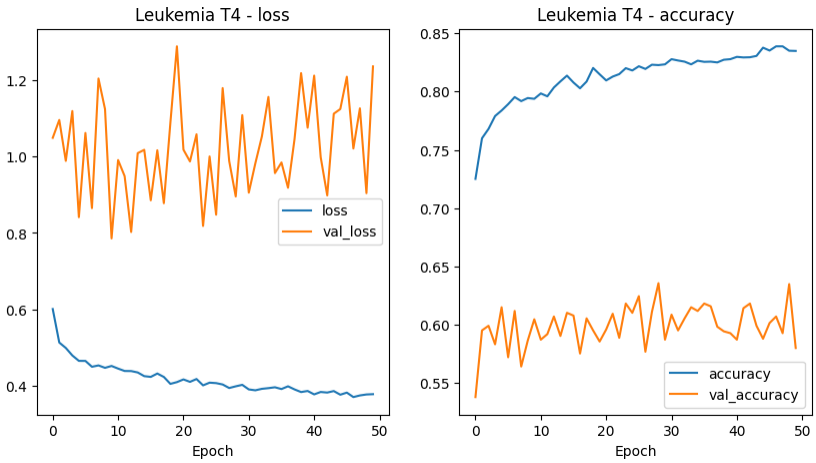

Figure 35: Leukemia VGG-16 loss and accuracy - experiment 4

For the Chest X-Ray dataset, this experiment resulted in a 84.80% test accuracy, which was slightly higher than both the baseline of 84.19% and experiment 3 accuracy of 83.57%. Figure 36 below shows a graph of the loss and accuracy from training the model for experiment 4 on the Chest X-Ray dataset.

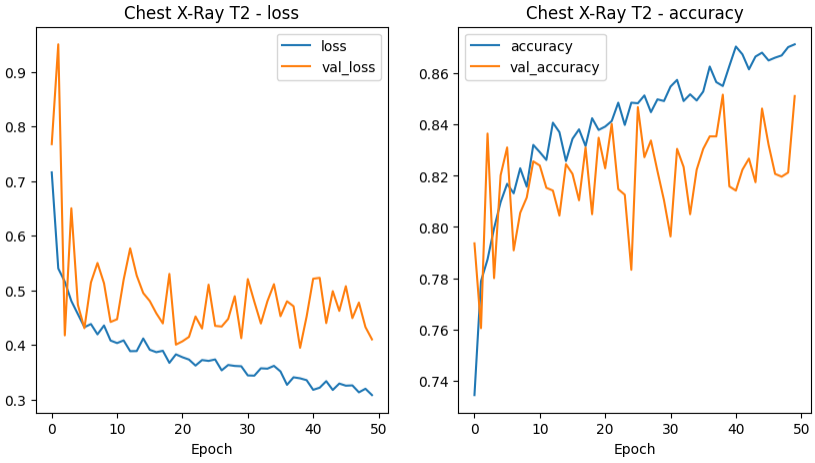

Figure 36: Chest X-Ray VGG-16 loss and accuracy - experiment 4

4.1.5 Summary of results

Table 8 below shows a comparison of the test accuracy for the baseline and all 4 experiments that were performed. The model that performed the best for each dataset is highlighted in green. As you can see, the Baseline model performed best for the Brain Tumor dataset. Experiment 1 performed the best for the Leukemia dataset. And finally, Experiment 4 performed the best for the Chest X-Ray dataset.

Table 8: Comparison of test accuracy

Dataset	Baseline	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Brain Tumor	91.99%	N/A	N/A	91.53%	91.38%
Leukemia	69.20%	74.27%	73.32%	63.74%	58.35%
Chest X-Ray	84.19%	N/A	N/A	83.57%	84.80%

The figures below show examples of the predicted classes for all 3 datasets, with the incorrectly classified images circled in red. All of these predictions are from the baseline model.

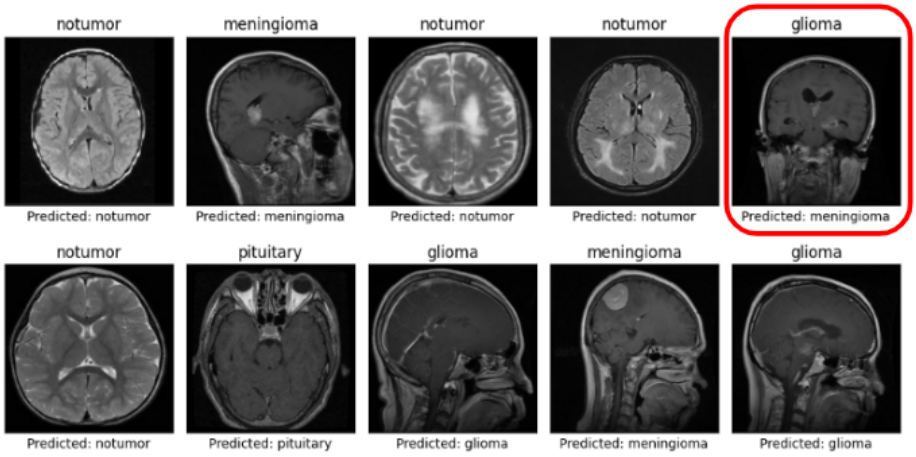

Figure 37: Brain Tumor VGG-16 baseline example predictions

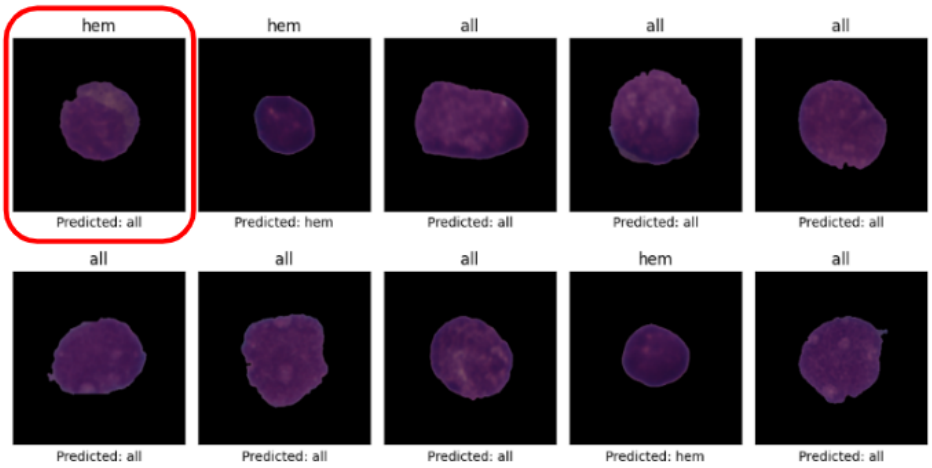

Figure 38: Leukemia VGG-16 baseline example predictions

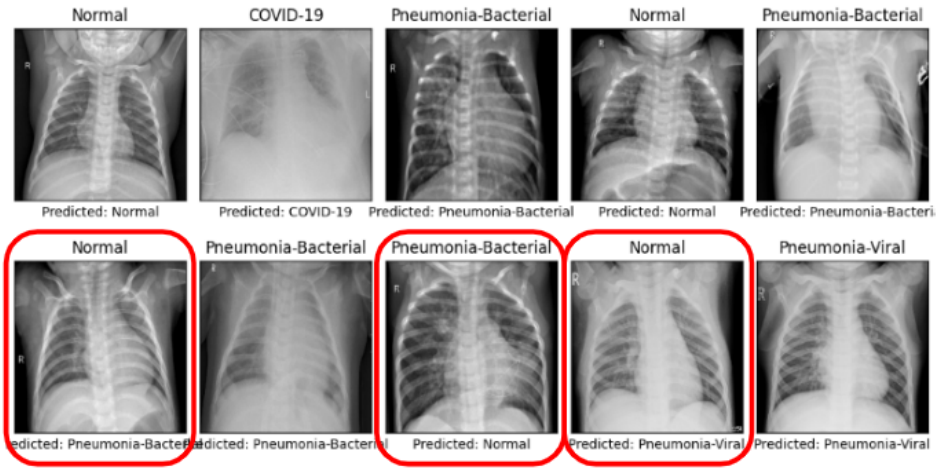

Figure 39: Chest X-Ray VGG-16 baseline example predictions

4.2 ResNet-50

4.2.1 Experiment 1 - Global Best Model

The baseline model for ResNet-50 that was outlined in section 3.2 above was run on all 3 datasets with the following hyperparameters:

- Batch Size: 64
- Optimizer: Adam
- Loss: sparse_categorical_crossentropy
- Number of epochs: 50

Table 9: ResNet-50 Model Results

Dataset	Test Accuracy
Brain Tumor	98.93%
Leukemia	86.27%
Chest X-Ray	84.55%

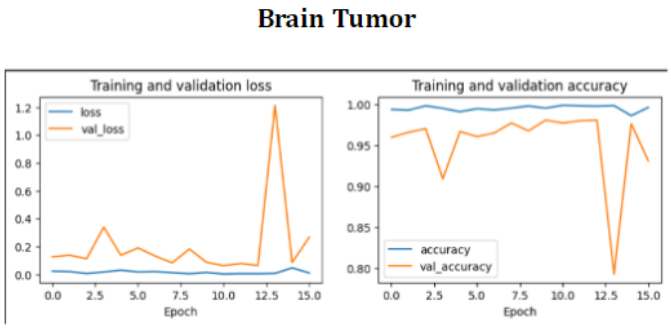

Figure 40: ResNet-50 Brain Tumor Plot

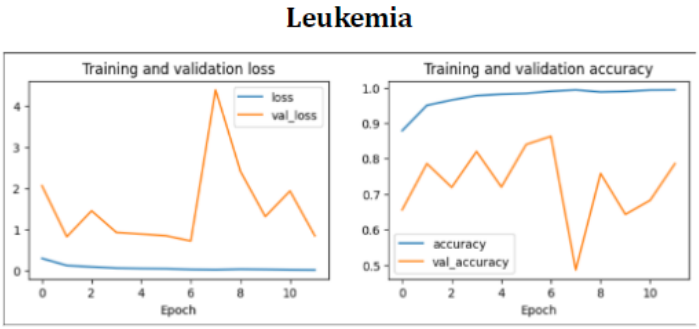

Figure 41: ResNet-50 Leukemia Plot

Chest X-ray

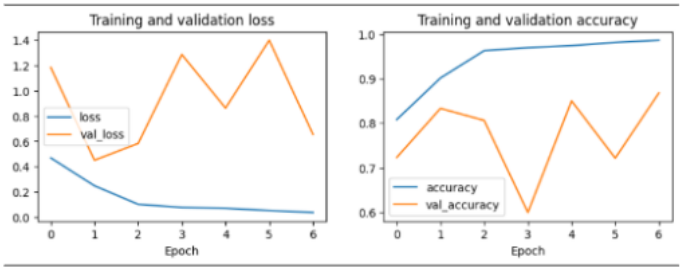

Figure 42: ResNet-50 Chest X-Ray Plot

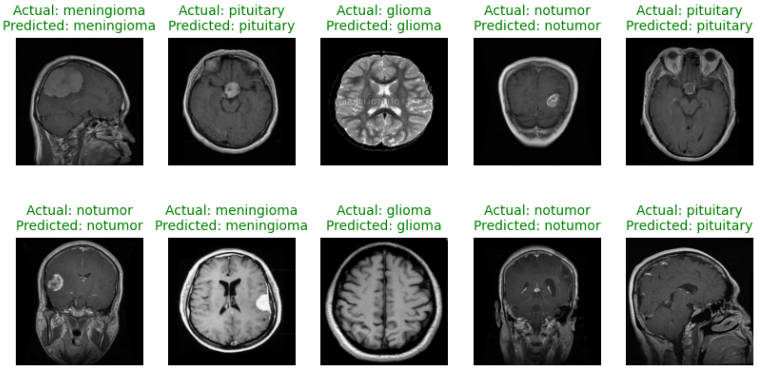

Figure 43: ResNet-50 Brain MRI Model Predictions

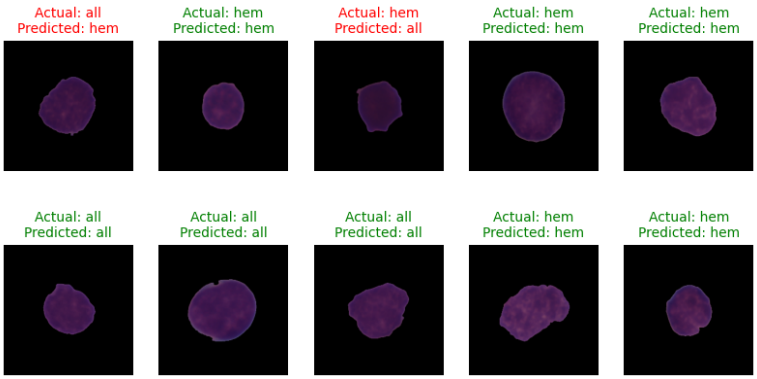

Figure 44: ResNet-50 Leukemia Model Predictions

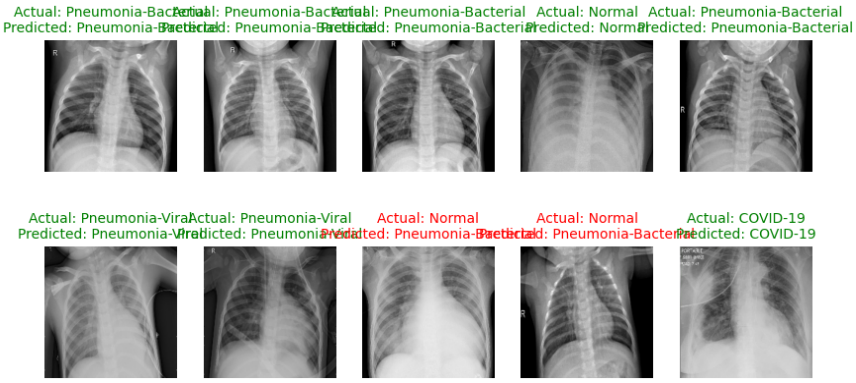

Figure 45: ResNet-50 Chest X-Ray Model Predictions

4.2.2 Experiment 2

Initially, all 3 datasets were taken to build a classification model, however, cv2 library was used to load the data and Image Data generator was used for augmentation. However, a couple 100 images in each training, validation and testing were not loaded correctly, and the improper way of loading the data maxed out GPU computation. This resulted in very poor generalization of the model on Leukemia and Chest X-Ray datasets. The augmentations and architecture remain the same as the current global architecture.

Table 10: ResNet-50 trial 1 model results

Dataset	Test Accuracy
Brain Tumor	98.93%
Leukemia	53.92%
Chest X-Ray	35.6%

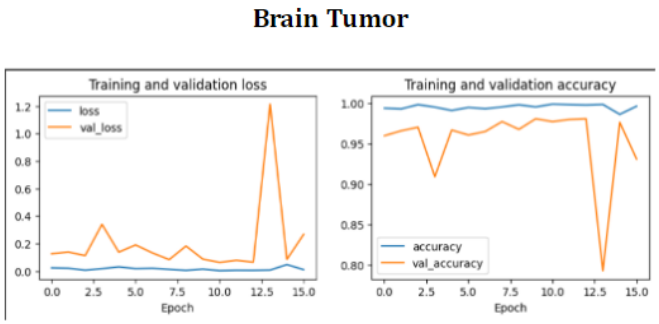

Figure 46: ResNet-50 Trial I Brain Tumor Plot

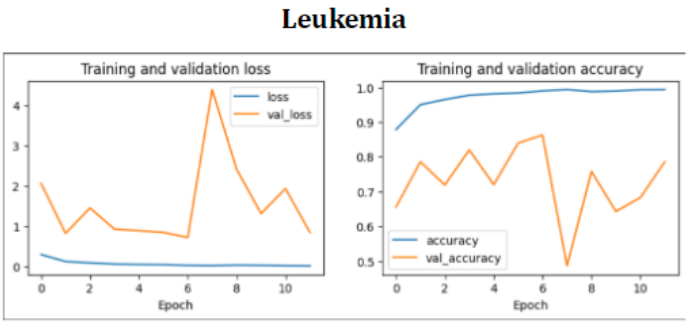

Figure 47: ResNet-50 Trial I Leukemia Plot

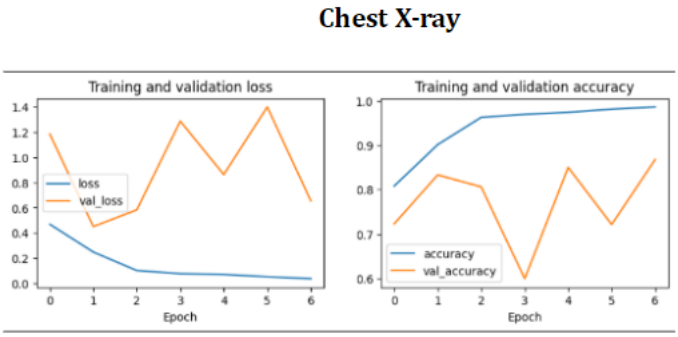

Figure 48: ResNet-50 Trial I Chest X-Ray Plot

The model overfit to the training dataset in case of Leukemia and Chest X-ray and could not generalize to the unseen data.

4.2.3 Experiment 3

A second architecture was then built using Image_Dataset_from_directory and adding augmentations as a sequential layer in the main architecture. This essentially loaded the data correctly and stopped maxing out Colab GPU. The new architecture had a complex design. I constructed the ResNet-50 model with the pre-trained weights from the ImageNet dataset. I added a few layers on top of the pre-trained model to make the classification task more suitable for the current dataset. Specifically, I added a global average pooling layer, followed by a flatten layer and three fully connected layers with ReLU activation. I also included batch normalization and dropout layers for regularization.

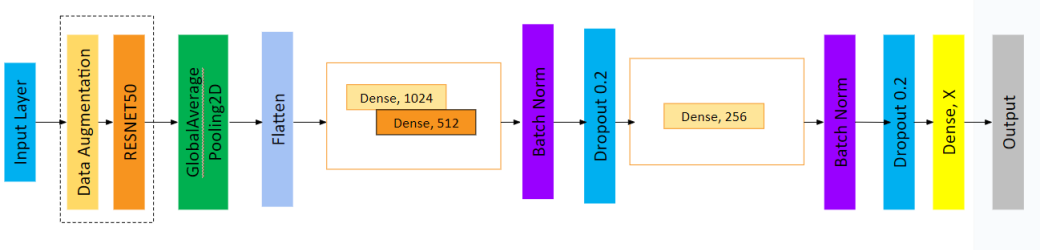

Figure 49 : ResNet-50 - Trial Proposed Architecture

The layers were frozen, and only the higher level complex computation layers were left trainable. The model was trained for 50 epochs with a batch size of 64 and validation loss was monitored. I used the ModelCheckpoint callback to save the model weights with the best validation loss. ES and LRonPlateau were not used. The final model achieved an accuracy of 90.05% on the training set and 87.95% on the validation set. The test accuracy came out at 90.77% for the Brain MRI dataset. It resulted in much poorer results for Leukemia and Chest X-Ray datasets at 55.56% and 83.36% respectively on test datasets. The bad result on Leukemia could probably be attributed to the RGB images while the others were Black and White.

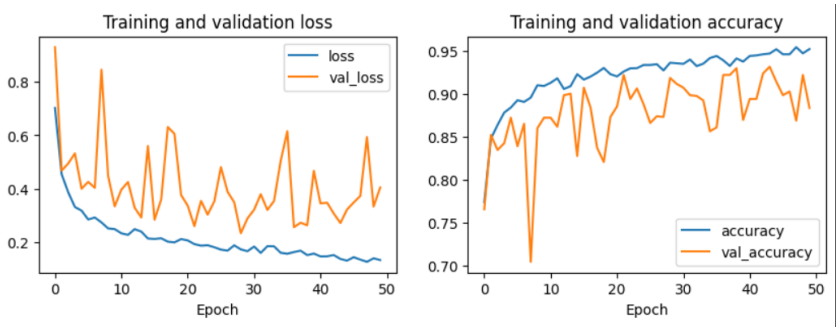

Figure 50: ResNet-50 Trial II Brain Tumor Plot

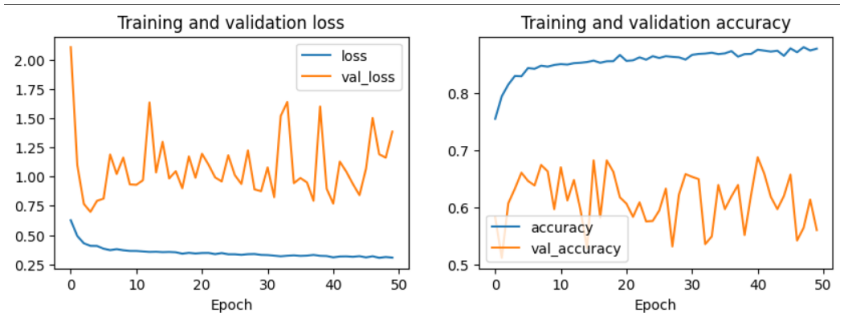

Figure 51: ResNet-50 Trial II Leukemia Plot

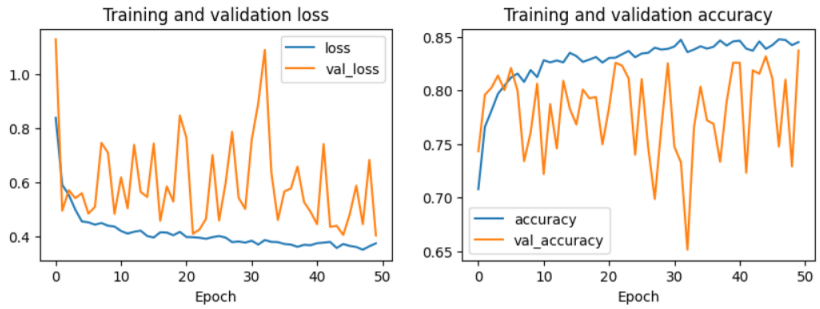

Figure 52: ResNet-50 Trial II Chest X-Ray Plot

4.3 Inception-V3

With the Inception-V3 as the base model. The weights are freezed for the pre-trained Inception-V3 model. The experiments conducted using the pre-trained Inception-V3 model are as follows.

4.3.1 Experiment 1

As part of experment1, I first added the flatten layer after the pre-trained Inception-V3, then a dropout layer of 0.2 and then added an output layer of 4 units with an activation function as softmax. The summary of the model is in Figure 53. The optimizer used is stochastic gradient descent with a learning rate of 0.01 as proposed in the article mentioned in section 2.3 and loss function used is sparse_categorical_crossentropy.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 resizing (Resizing)         (None, 224, 224, 3)       0

 tf.math.truediv (TFOpLambda  (None, 224, 224, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 224, 224, 3)      0
 a)

 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 flatten (Flatten)           (None, 51200)             0

 dropout (Dropout)           (None, 51200)             0

 dense (Dense)               (None, 4)                 204804

=================================================================
Total params: 22,007,588
Trainable params: 204,804
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 53: Summary of experiment 1 model

I have trained the model on the Brain tumor dataset, for 10 epochs. Because of the limited GPU in google colab, I wanted to run the best model on the other leukemia and chest x-ray datasets. The test accuracy for this model is 83.14%. However, there are so many ups and downs in the loss and accuracy graphs for validation data because of overfitting. The Figures for the same are given below.

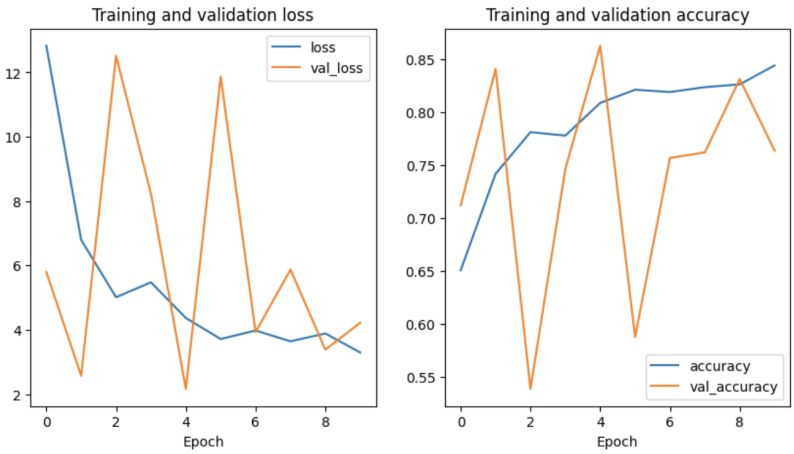

Figure 54: Training and Validation graph for experiment 1

The predicted values of images from experiment 1 is given in Figure 55.

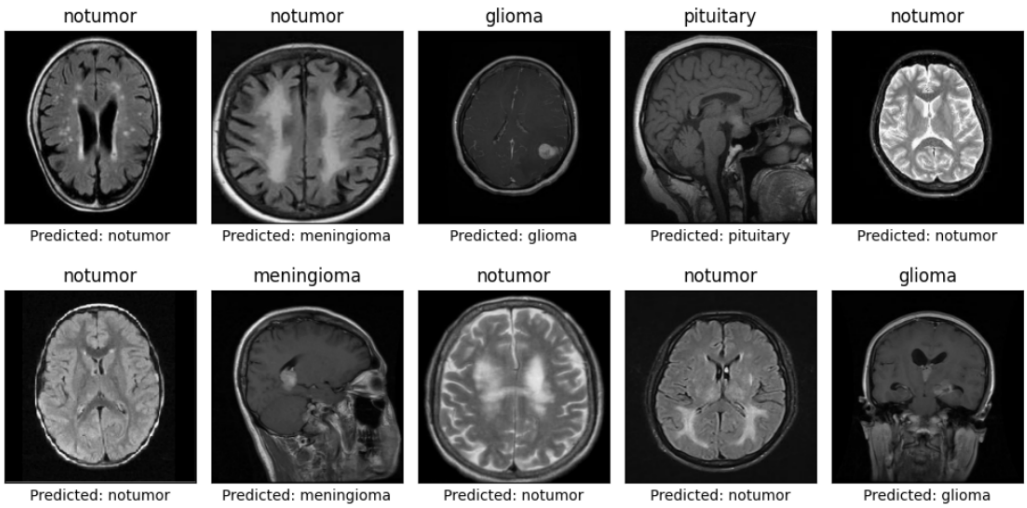

Figure 55: Predicted values of experiment 1

Because of the overfitting in validation data, I tried mse loss function in experiment2 for the same model as in experiment 1.

4.3.2 Experiment 2

For the same model as in the previous experiment1, used mse as loss function. The test accuracy is down to 65.22% and it didn't predict values as expected. The accuracy and loss graphs for 10 epochs is given in Figure 56.

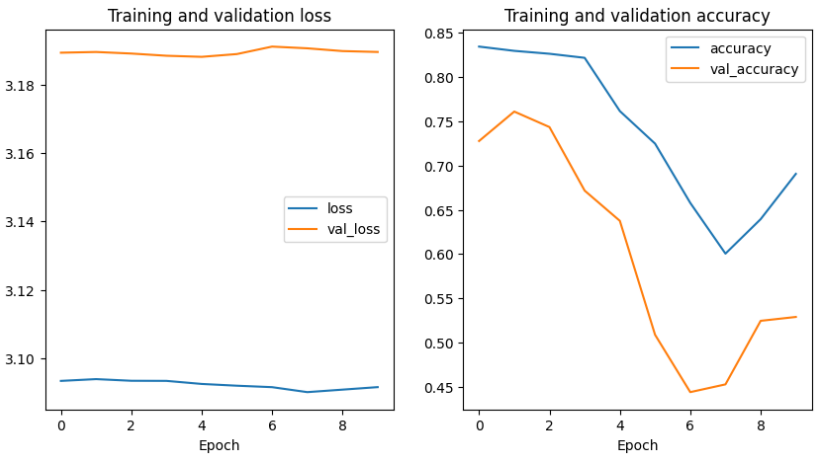

Figure 56: Loss and accuracy for experiment 2

The predicted images from experiment 2 are given in Figure 57.

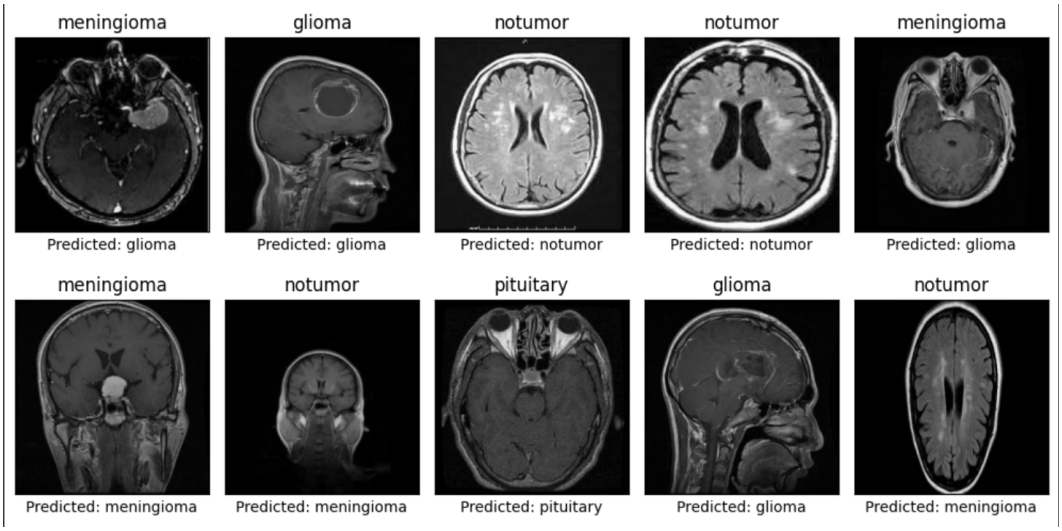

Figure 57: Predicted values of experiment 2

The loss from experiment 2 is very high and stable for all the 10 epochs and even though the accuracy is high for the first 3 epochs it gradually started to decrease till epoch 8 and started to increase. Maybe with the higher epoch count the accuracy may increase. I even tried to check with a learning rate of 0.001, with the same loss function. The test accuracy is 52.71% and the model did not predict images as expected. The loss and accuracy are given in Figure 58 and the predictions are given in Figure 59.

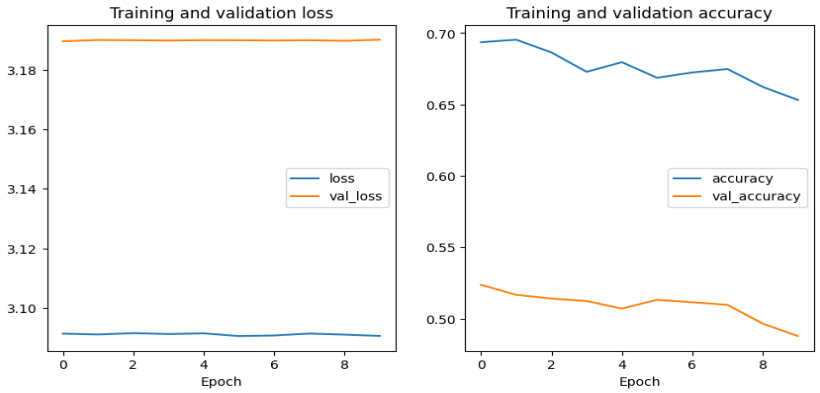

Figure 58: Loss and accuracy with learning rate of 0.001

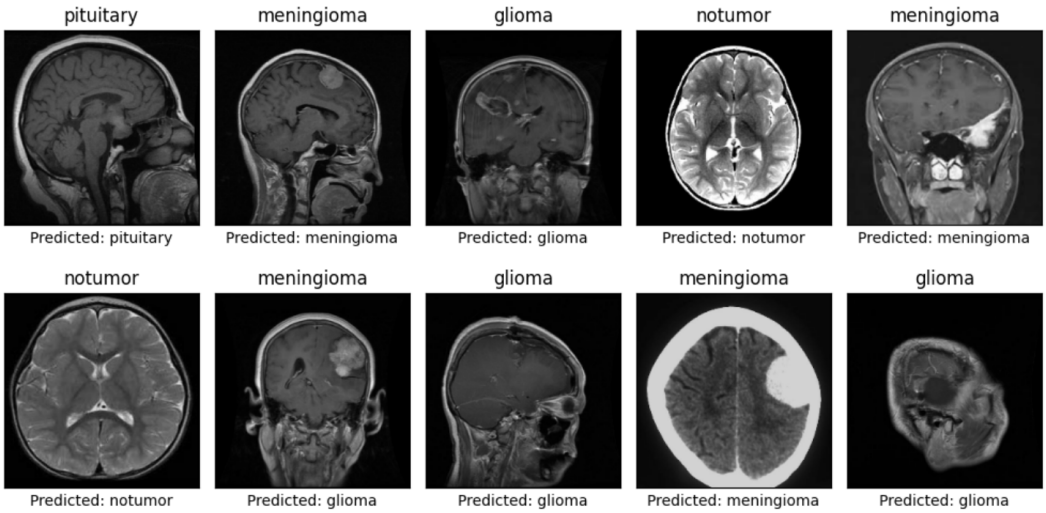

Figure 59: Images predicted with learning rate of 0.001

The model with learning rate of 0.01 or 0.001, and mse as loss function did not predict value and the higher stable loss in the data. I don't think even with the higher epoch count mse loss function will perform well. So I moved on to experiment 3.

4.3.3 Experiment 3

For the model as described in experiment 1, changed the dropout from 0.2 to 0.5 and added a dense layer of 256 units with an activation function as Relu after the flatten layer. The summary of the model is given in Figure 60.

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 resizing_7 (Resizing)       (None, 224, 224, 3)       0

 tf.math.truediv_7 (TFOpLamb  (None, 224, 224, 3)      0
 da)

 tf.math.subtract_7 (TFOpLam  (None, 224, 224, 3)      0
 bda)

 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 flatten_4 (Flatten)         (None, 51200)             0

 dense_3 (Dense)             (None, 256)               13107456

 dropout_3 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 4)                 1028

=================================================================
Total params: 34,911,268
Trainable params: 13,108,484
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 60: Model summary for experiment 3

As there is no much loss in experiment 1 while using sparse_categorical_crossentropy decided to use the same in experiment 3. The optimizer chosen is RMSprop as I'm trying not to overfit the data which was happening in the experiment 1. The loss and accuracy graph for experiment 3 is given in Figure 61.

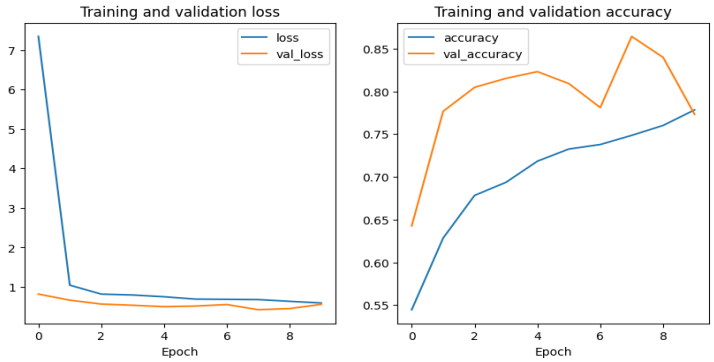

Figure 61: Loss and accuracy graphs for experiment 3

The test accuracy is 84.67% and even the image prediction is also well. In the early epoch the loss is high but gradually the loss is reduced and the model accuracy is also increasing as the epochs are increasing. The images predict values are given in Figure 62.

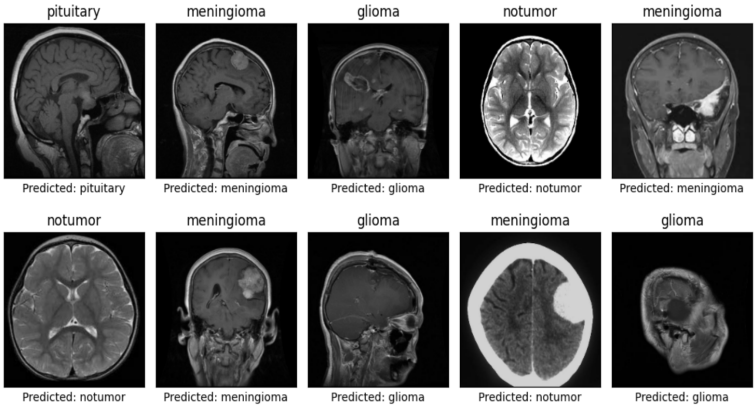

Figure 62: Images predicted in experiment 3

Then tried with 20 epochs, but there is no increase in the model accuracy and there is a chance of overfitting the data as the validation accuracy is greater than train data accuracy. The same is presented in Figure 63.

Figure 63: Loss and accuracy graph with 20 epochs

4.3.4 Experiment 4

In experiment 4, which I thought was the best model, it did not predict well when I ran the model on the next day. In this model Maxpool 2d layer with a pool size of 2 as proposed in the paper represented in Section 2.3. This layer is added before the Flatten layer. The units in the Dense layer are changed to 512 and dropout is changed to 0.2. The summary of the model is given in Figure 64.

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_13 (InputLayer)       [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 resizing_11 (Resizing)      (None, 224, 224, 3)       0

 tf.math.truediv_11 (TFOpLam  (None, 224, 224, 3)      0
 bda)

 tf.math.subtract_11 (TFOpLa  (None, 224, 224, 3)      0
 mbda)

 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 max_pooling2d_6 (MaxPooling  (None, 2, 2, 2048)       0
 2D)

 flatten_8 (Flatten)         (None, 8192)              0

 dense_14 (Dense)            (None, 512)               4194816

 dropout_10 (Dropout)        (None, 512)               0

 dense_15 (Dense)            (None, 4)                 2052

=================================================================
Total params: 25,999,652
Trainable params: 4,196,868
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 64: Model summary

I have used 10 epochs with Adam as an optimizer and sparse categorical cross entropy as loss function were used. The results given in Figures were taken from the first day run of experiment 4.

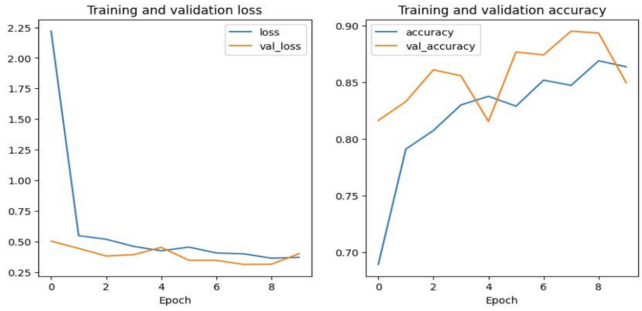

Figure 65: Loss and accuracy graph for Brain tumor

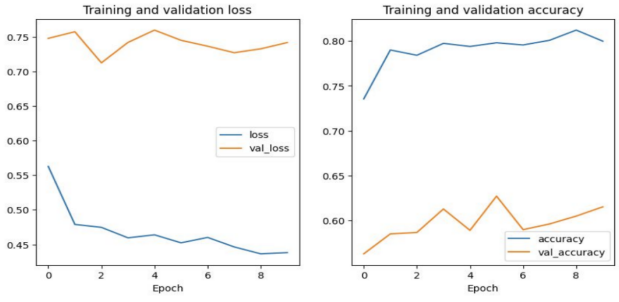

Figure 66: Loss and accuracy graph for Leukemia

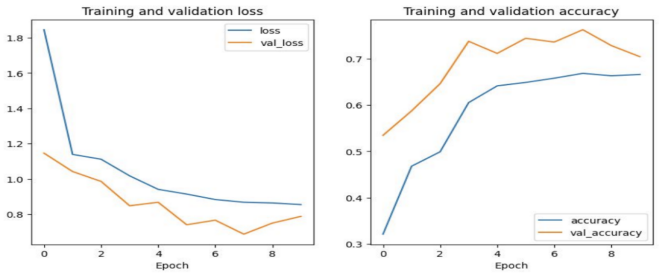

Figure 67: Loss and accuracy graph for Chest x-ray

The test accuracy summary for 3 datasets is given in Table 11.

Table 11: Test accuracy summary table - experiment 4

Data set	Test accuracy
Brain Tumor	86.12%
Leukemia	75.14%
Chest X-ray	75.58%

Even though the accuracies are consistent for all the 3 datasets, the model did not perform well for the next day. That may be because of the overfitting in the dataset. Thus in order to overcome the problem added Batch Normalization in the experiment 5.

4.3.5 Experiment 5

After realizing the overfitting problem from the model mentioned in section 4.3.4, added a batch normalization layer after the transfer learning model. Instead of a Max-pooling layer added Global average pooling 2D layer. To reduce the complexity of the model, the units in the model from experiment 3 were reduced from 512 to 256. The dropout has been increased to 0.5. The model summary is mentioned in Figure 68.

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 resizing_2 (Resizing)       (None, 224, 224, 3)       0

 tf.math.truediv_2 (TFOpLamb  (None, 224, 224, 3)      0
 da)

 tf.math.subtract_2 (TFOpLam  (None, 224, 224, 3)      0
 bda)

 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 batch_normalization_95 (Bat  (None, 5, 5, 2048)       8192
 chNormalization)

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dense_3 (Dense)             (None, 256)               524544

 dropout_1 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 4)                 1028

=================================================================
Total params: 22,336,548
Trainable params: 529,668
Non-trainable params: 21,806,880
_________________________________________________________________
```

Figure 68: Model summary

The model is trained and tested on all 3 datasets for 15 iterations. The loss and accuracy graphs are given in Figure 69,70,71.

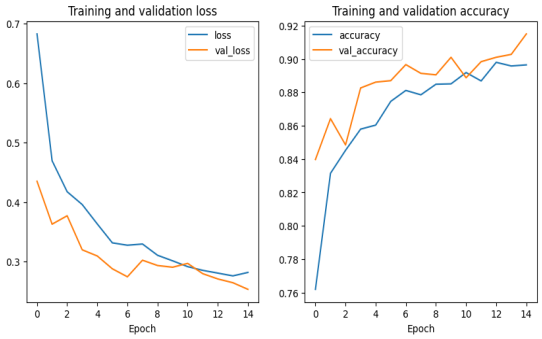

Figure 69: Loss and accuracy graph for Brain tumor

Figure 70: Loss and accuracy graph for Leukemia

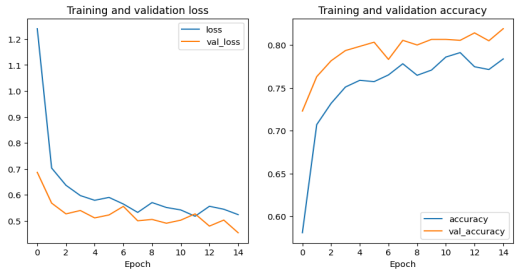

Figure 71: Loss and accuracy graph for Chest x-ray

The test accuracy summary for 3 datasets is given in Table 12.

Table 12: Test accuracy summary table - experiment 4

Data set	**Test accuracy**
Brain Tumor	88.86%
Leukemia	76.8%
Chest X-ray	80.75%

Based on the plots and test accuracy results, it is observed that there is no huge loss in the data and the Chest x-ray and brain Mri images data sets val accuracy is greater than model train data accuracy. Maybe because of the smaller data it is overfitting, I need to tune the hyper parameters as well. Overall this model performed well for all the 3 datasets and predictions are also good compared to the other models proposed in the experiments from Section 4.3.1,4.3.2, 4.3.3 and 4.3.4.

4.4 Summary of Results

Table 13 below shows a comparison of the best results from each of the 3 pre-trained models.

Table 13: Comparison of results from different pre-trained models

Dataset	**VGG-16 Test Accuracy**	**ResNet-50 Test Accuracy**	**Inception-V3 Test Accuracy**
Brain Tumor	91.99%	98.93%	88.86%
Leukemia	74.27%	86.27%	76.8%
Chest X-Ray	84.80%	84.55%	80.75%

Some of the reasons why the images got incorrectly classified could be attributed to the variability in the image appearance. Tumors vary in sizes, and so do leukemia cells, and a model has to be

sensitive enough to identify them correctly. This is also directly related to the problem of limited dataset size, where in the model overfitted to training data and could not generalize to the testing data. Another reason is lack of clinical context, meaning that medical images need to be interpreted in the context of the patient's medical history, symptoms and other diagnostic tests. One more plausible reason we identified is that we are unaware of how diverse the patient images were due to privacy issues. A disease could show up very differently in patients of different ages, sex, ethnicity, and other factors.

5 Conclusion

The pre-trained models named VGG-16, ResNet-50 and Inception-V3 all performed well on the 3 different datasets. Even though the Brain MRI and Chest X-ray images are black and white, the proposed models performed well for the Leukemia dataset too, which is an RGB dataset and binary classification instead of multiclass. However, we couldn't generalize which is the best model among the 3 pre-trained models. ResNet-50 performed well for Brain MRI images, VGG-16 performed well for Chest x-ray and Inception-V3 performed well for Leukemia. As for the different dataset types, different activation and loss functions should be considered to predict the images correctly.

Initially, the plan of the project is to propose a model which identifies the abnormality in the brain MRI and Leukemia datasets. While creating and testing the models with different hyper parameters. We came up with an idea to create an unified model which identifies the abnormality in all the different datasets.

With limited GPU, we were not able to tune hyperparameters and run epochs more than 20. The type of activation function, loss function and different types of optimizers to be used depending on the data was identified. The complexity of the model was also taken into picture while creating the models.

As part of the future plan, we would like to merge the 3 different datasets as one big dataset of 10 classes and predict the value as expected.

6 Contributions

As a group, we came up with the idea of trying to use the same model on different medical image datasets.

6.1 VGG-16

For the VGG-16 based transfer learning model, I used the VGG-16 transfer learning model from Khan et al. [1] to determine which layers to include in the model. I updated the code to replace the deprecated methods with the more current methods for loading image data and performing data image augmentation. I defined reusable functions with parameterized hyperparameters to easily train and evaluate models for the different datasets and experiments. I customized the image augmentation, and came up with the ideas for the different VGG-16 experiments to run.

6.2 ResNet-50

For the ResNet-50 based transfer learning model, I applied my own architectures after multiple experiments because I could not find any source codes or detailed architectures from the papers, mostly due to the fact that ResNet was used as part of an ensemble model and not individual. For the main model, I used overfitting solutions Brownlee [8], and Khanna [9]. I defined reusable functions for displaying augmented images, predictions and plot learning.

6.3 Inception-V3

The concept behind Inception-V3 was understood from the paper by Szegedy [10]. For the proposed Inception-V3 model, I have considered hyperparameters used in the Tamilarassi paper [3] and for the data augmentation steps I have considered the paper by Alqudah et al. [7]. I have created the proposed model as mentioned in Section 4.3.5 , after many experiments. The global average pooling idea is taken from Kim [11]. The batch normalization layer is added as part of overcoming the overfitting problem.

References

[1] Khan, Md Saikat Islam et al. “Accurate brain tumor detection using deep convolutional neural network.” Computational and structural biotechnology journal vol. 20 4733-4745. 27 Aug. 2022, doi:10.1016/j.csbj.2022.08.039

[2] Hussain, M., Raza, K., Hafeez, A., Rehman, A., Nisar, A., & Bhatti, A. (2022). Performance analysis of deep learning models for brain tumor segmentation and classification. medRxiv. doi: 10.1101/2022.07.18.22277779.

[3] Tamilarasi, R., and S. Gopinathan. "Inception architecture for brain image classification." Journal of Physics: Conference Series. Vol. 1964. No. 7. IOP Publishing, 2021.

[4] Nickparvar, Msoud. (2021, September 24). Brain Tumor MRI Dataset [Data set]. Kaggle. https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

[5] Elmorshedy, Abeer. (2022, December 10). leukemia [Data set]. Kaggle. https://www.kaggle.com/datasets/abeerelmorshedy/leukemia

[6] Kolas, Artsiom. (2022, May 30). 3 kinds of Pneumonia [Data set]. (2022, May 30). Kaggle. https://www.kaggle.com/datasets/artyomkolas/3-kinds-of-pneumonia

[7] Alqudah, Ali Mohammad, et al. "Brain tumor classification using deep learning technique--a comparison between cropped, uncropped, and segmented lesion images with different sizes." arXiv preprint arXiv:2001.08844 (2020).

[8] Brownlee, J. (2020). Introduction to regularization to reduce overfitting and improve generalization error. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/

[9] Khanna, A., & Jalal, A. S. (2021). An Overview of Artificial Intelligence in the Age of Deep Learning. International Journal of Information and System Assurance, 15(2), 20-30. Retrieved from https://www.mecs-press.org/ijisa/ijisa-v15-n2/IJISA-V15-N2-3.pdf

[10] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[11] Kim, Jung-Hun. "Improvement of inceptionv3 model classification performance using chest X-ray images." Journal of Mechanics in Medicine and Biology 22.08 (2022): 2240032.