Multi-class Classification of Brain MRI Scans using Machine Learning

Eric Lehmphul

5/1/2023

Contents

1 Abstract

The use of medical imaging has become a normality for properly diagnosing a plethora of illnesses, including cancer. The focus of this project is on the classification of brain MRI scans with the following conditions: Normal Brain function, Glioma tumor, Meningioma tumor, and Pituitary tumor. The methodology for addressing this objective consisted of three phases: Data Exploration, Data Preprocessing, and Convolutional Neural Network (CNN) Analysis. This project analyzed 3264 image files from https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri (Bhuvaji, 2020). The dataset was examined to identify important characteristics like image shape, size, and color. The data then underwent preprocessing steps to be fed into the CNN models. As the dataset size is small, image augmentation was implemented using basic image transformations. The dataset was standardized for size, pixel intensity, and color gradient. Two different CNN architectures were built from scratch and optimized for classification accuracy. The best model predicted with an overall accuracy of 74.87% on the testing data and high F1 scores for three of the four classes.

Keywords: Convolutional Neural Network, Medical Imaging, Data Augmentation, Multiclass Classification, Brain Tumor Classification

2 Introduction

2.1 The Author

Eric Lehmphul is a Master of Data Science candidate at CUNY School of Professional Studies, who resides in Newburgh, New York.

2.2 The Problem

Brain tumors constitute a serious health concern worldwide, accounting for a substantial proportion of cancer related deaths. In 2019, brain cancer and central nervous system cancer was the tenth deadliest cancer in terms of mortality quantity (Roser & Ritchie, 2015). Timely and accurate diagnosis is pivotal in treatment and survival success rates. Traditionally, the process of tumor classification has relied heavily on the expertise of radiologists. This project looks to leverage machine learning to aid in tumor classification and answer the following question:

- Can machine learning algorithms accurately classify brain tumors based on medical imaging data?

3 Literature Review

3.0.1 Convolutional Neural Network Classification on Medical Images

CNNs have proven to be effective at achieving significant results in the ImageNet dataset, a benchmark for deep learning-based image classification (Krizhevsky et al., n.d.). The use of CNNs and their predictive power have been expanded to the medical field for a variety of image classification tasks. Kermany et al. (2018) leveraged transfer learning on Optical Coherence Tomography (OCT) images to classify 4 different types of retinal conditions. They achieved an accuracy of 96.6%, sensitivity of 97.8%, and a specificity of 97.4%. Ophthalmology experts were asked to classify 1000 images, with results then compared to the model. The study found that the best model achieved higher performance than some experts with years of experience. Khan et al. (2021) developed a CNN to segment and classify various types of breast abnormalities. They utilized the previously trained model, ResNet50, via transfer learning. The model architecture contained 13 convolutional layers, 5 pooling layers, 3 dense layers, and 1 output layer. The proposed model achieved an accuracy of 88% when analyzing 4 different types of breast abnormalities.

3.0.2 Data Augmentation

Deep Learning applications require a large amount of data to provide highly accurate results. Not all domains have access to the quantity of data essential for deep learning applications. One such domain is the medical field, when image analysis is required. Kiryati & Landau (2021) describe the challenging and costly methods for obtaining relevant medical images. Typically, regulatory approvals must be obtained before usage, they must be found in an institutional archive, and must have identifying details removed prior to use. For this reason, most medical imaging data sources are relatively small. The limited data can cause deep learning models to overfit. Pan et al. (2019) explored the effects of image augmentation on a food recognition CNN algorithm. They enacted basic image transformations like flipping and rotating images to expand the training size. The augmented dataset outperformed the non-augmented dataset in two of the three model frameworks. Yang et al. (2022) used several data augmentation methods to improve overall model accuracy. Image manipulation methods, image erasing methods, and image mix methods were all applied to the data to create the augmentation dataset. They found that the accuracy improvements ranged from 0.07% to 2.8% on the datasets used.

4 Methodology

As stated in the introduction, this research was focused on attempting to answer the following question:

- Can machine learning algorithms accurately classify brain tumors based on medical imaging data?

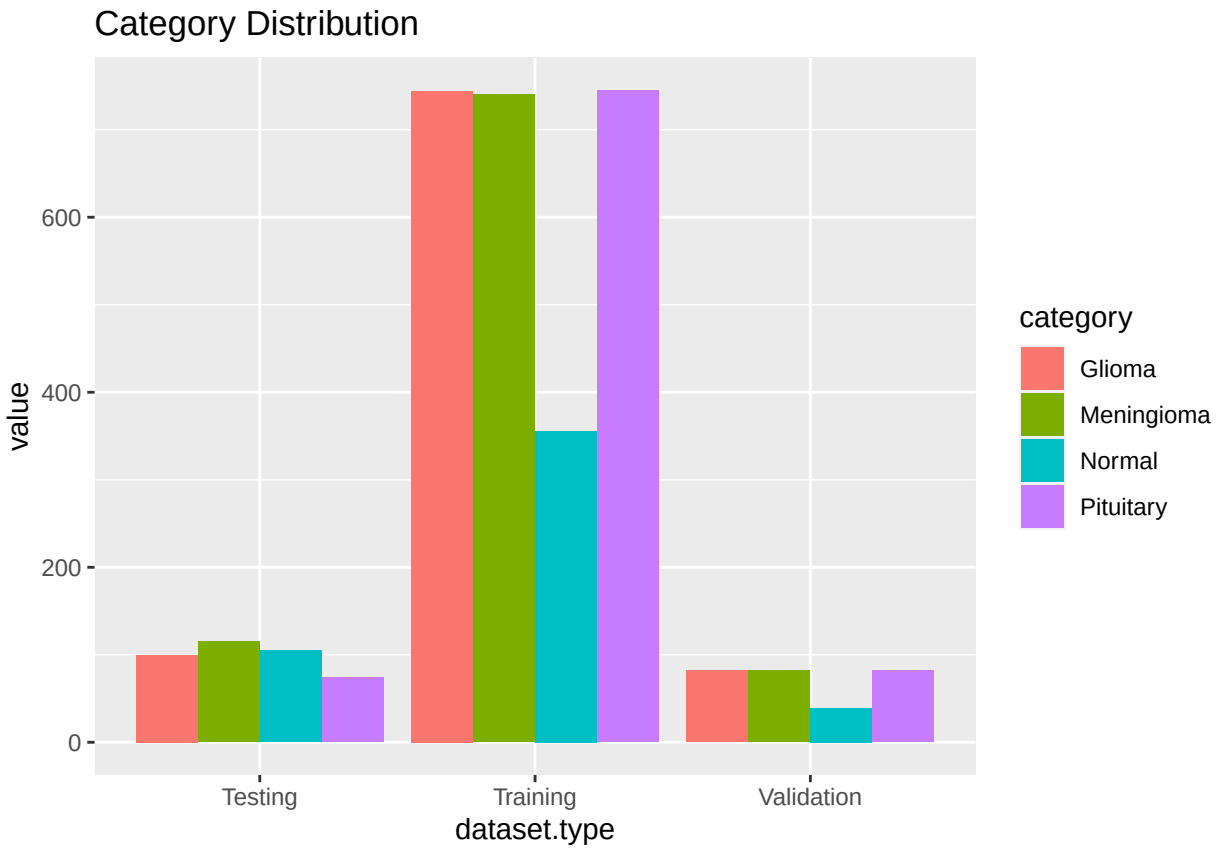
The methodology for answering this question was a three-step process: Data Exploration, Data Preparation, Convolutional Neural Network Analysis. The essence of each step is briefly described below:

- *Data Exploration:* Examined the data directories of images to understand the characteristics of each image and the total size of the dataset. Explored the image dimensions, scale, and color mode of the images.
- *Data Preparation:* Developed strategies to standardize images to be implemented into a Convolutional Neural Network architecture as well as augment images to expand training sample size. Devised methods to standardize image dimensions, normalize image pixel intensity to be between 0 and 1, and convert all images to grayscale for faster model results.
- *Convolutional Neural Network Analysis:* The development and testing of different convolutional neural network architectures. The best model was selected based on accuracy, precision, specificity, sensitivity, and F1 score.

4.1 Data Exploration

4.1.1 Data Distribution

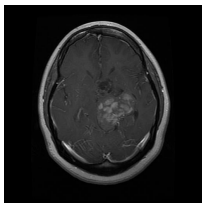
The data source for this project can be accessed with the following link: https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri?select=Training (Bhuvaji, 2020). The dataset for this project contained 3264 magnetic resonance imaging (MRI) scans of healthy brains and brains with glioma tumors, meningioma tumors, or pituitary tumors. The dataset provided training and testing directories to be used for model creation and evaluation. Ten percent of the training data was used to create a validation directory to further assess model performance. The healthy-looking brain MRI scans are the least represented category within the training directory with each tumor class having a similar number of observations. The testing set provides images for each class which should provide good representation of how the proposed models generalize.



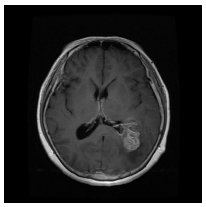
4.1.2 Images

All images within the dataset are stored in a jpeg file. Each image file is not an identical size and will need to be resized for before it is applied to the CNN architecture. Each image is stored in a 3-dimensional array meaning that it is represented by red / green / blue color intensities. The images vary in size from 198 x 254 x 3 to 1446 x 1375 x 3. The most common size of the images is 512 x 512 x 3. Below are a small sample of the training images for each category. The images contain both top-down views and side profile views of the brain.

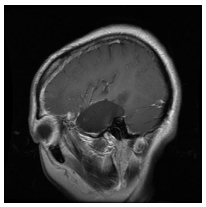
Glioma Tumor 1



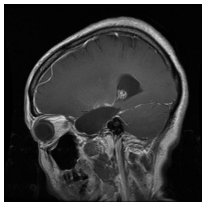
Glioma Tumor 2



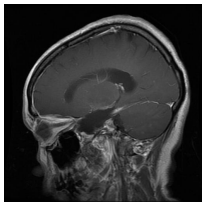
Glioma Tumor 3



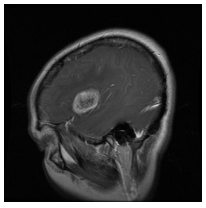
Glioma Tumor 4



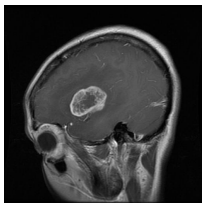
Glioma Tumor 5



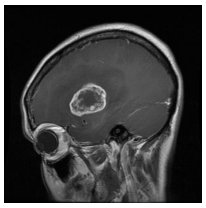
Glioma Tumor 6



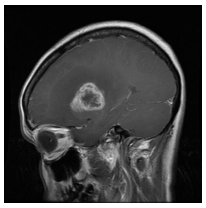
Glioma Tumor 7



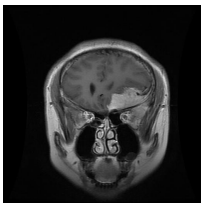
Glioma Tumor 8



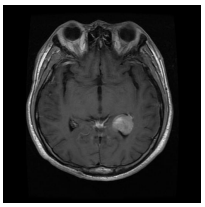
Glioma Tumor 9



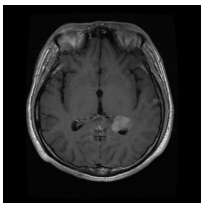
Meningioma Tumor 1



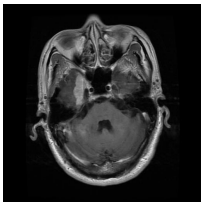
Meningioma Tumor 2



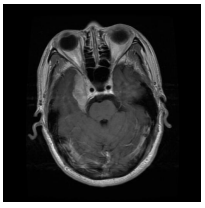
Meningioma Tumor 3



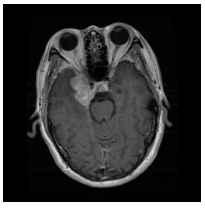
Meningioma Tumor 4



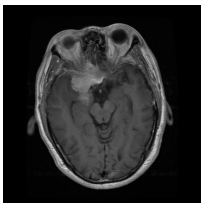
Meningioma Tumor 5



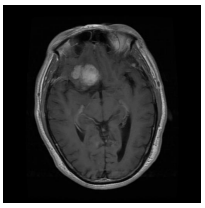
Meningioma Tumor 6



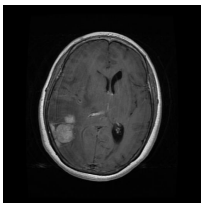
Meningioma Tumor 7



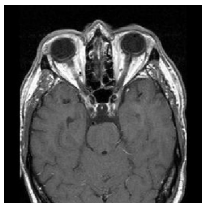
Meningioma Tumor 8



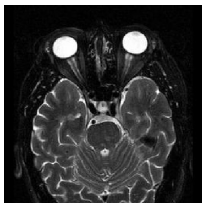
Meningioma Tumor 9



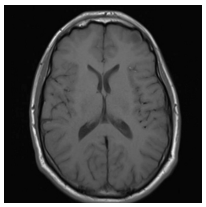
Normal Scan 1



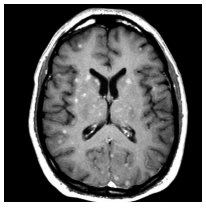
Normal Scan 2



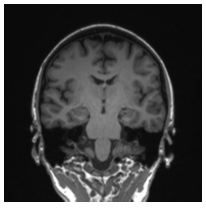
Normal Scan 3



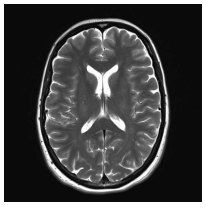
Normal Scan 4



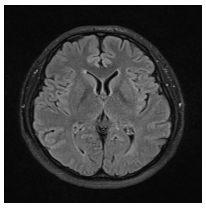
Normal Scan 5



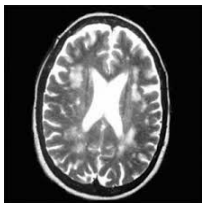
Normal Scan 6



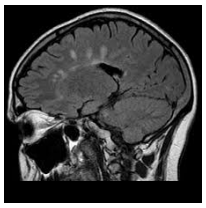
Normal Scan 7

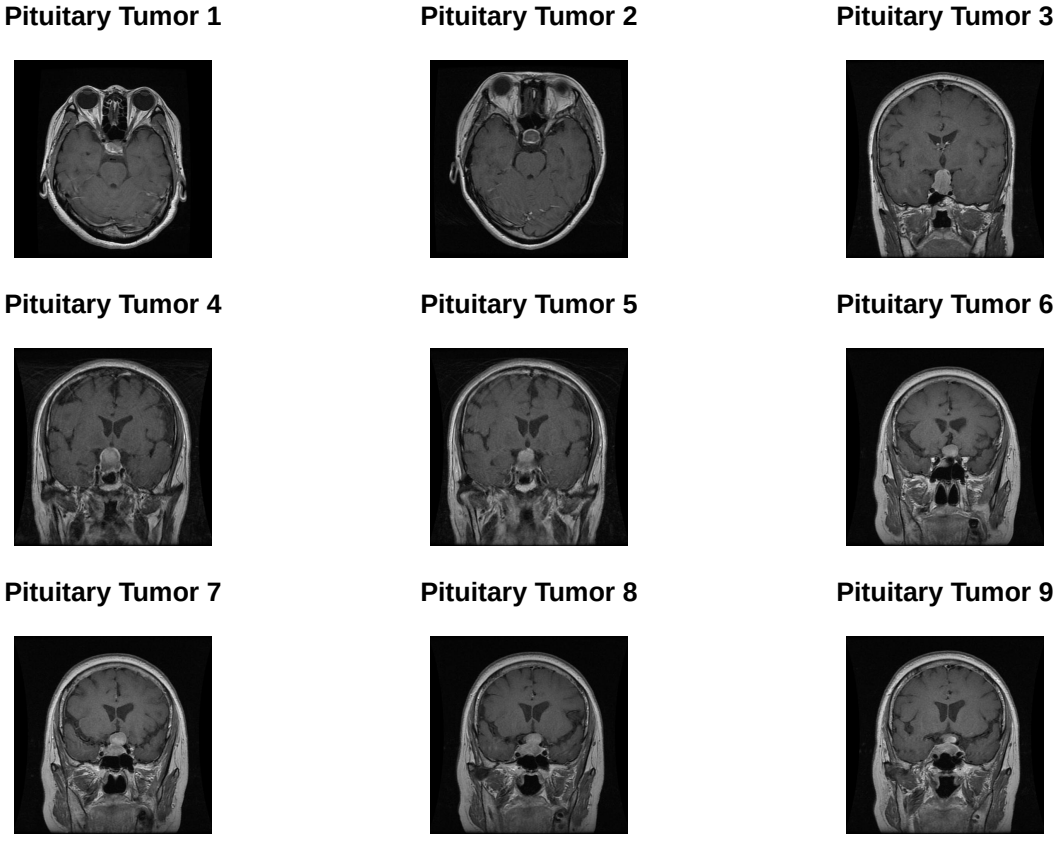


Normal Scan 8



Normal Scan 9



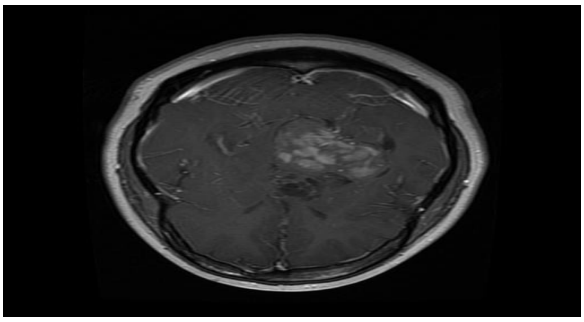
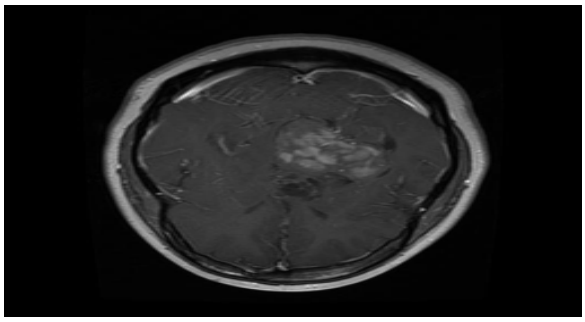


4.2 Data Prepration

The data exploration uncovered steps to be taken to prepare the data for modeling. Each image was assigned a class based on the directory the image file originated from. The data was resized to be uniform for all images. The images were resized to 256 x 256 pixels. The reason for choosing 256 x 256 pixels as the resized shape is because it was found to be the most optimal resolution in terms of f1 score in a study on Endoscopy Image Classification (Thambawita et al., 2021).

The data was transformed from a three-dimensional array to a two-dimensional array. The use of red, green, and blue color channels could achieve more accurate results compared to grayscale images; however, grayscale was chosen as it is less computationally expensive. After transformation, the grayscale images do not appear to be much different than their RGB image counterparts. Only images that required upscaling or downscaling will appear somewhat different than the original image.

RGB Colors and Native **Grayscale and Standardized**

RGB Colors and Native **Grayscale and Standardized**

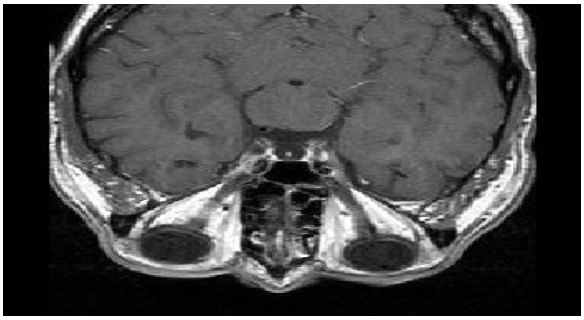
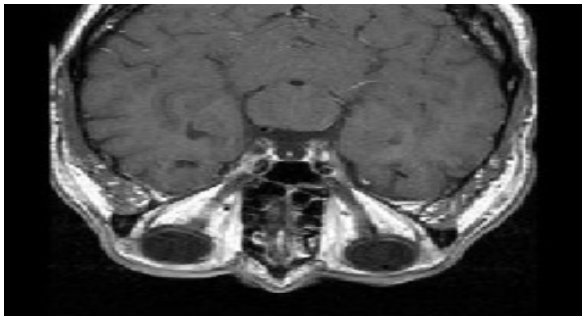
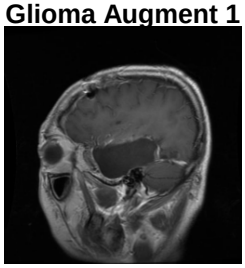
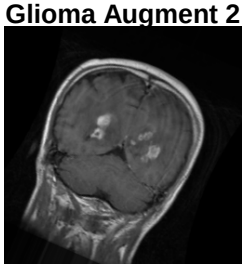



Image augmentation was used to expand the number of images the dataset, allowing for the model to generalize more effectively. The images were rotated and flipped across the y axis. The maximum range for the rotation of an image was set between 0 degree and 30 degrees. More complex data augmentation techniques were excluded from this project to keep the model computationally efficient and to maintain image integrity. To reduce computer central processing unit and memory consumption, image augmentations will occur in batches during model training. Each batch will provide the model with 20 newly augmented images to be used for weight tuning via backpropagation. Below are some samples of augmented images from the training dataset.

Glioma Augment 1

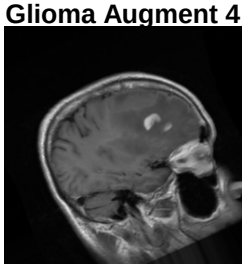


Glioma Augment 2

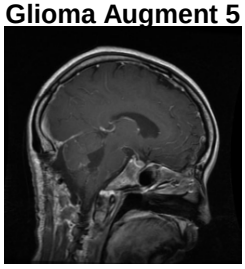


Glioma Augment 3

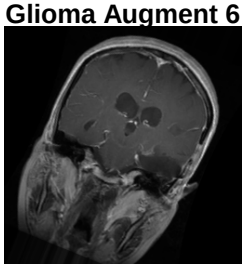
Glioma Augment 4



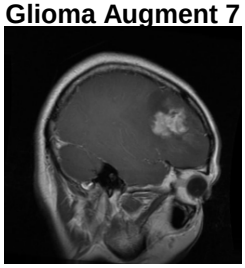
Glioma Augment 5



Glioma Augment 6

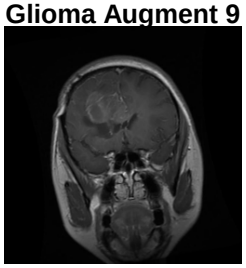


Glioma Augment 7

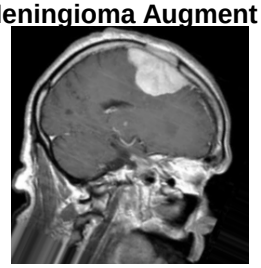


Glioma Augment 8

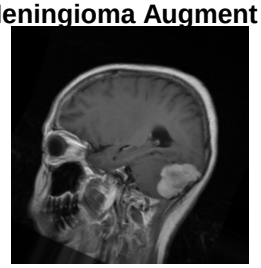
Glioma Augment 9



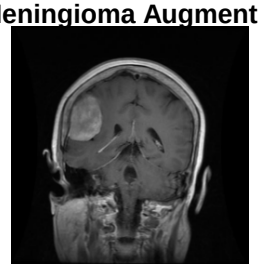
Meningioma Augment 1



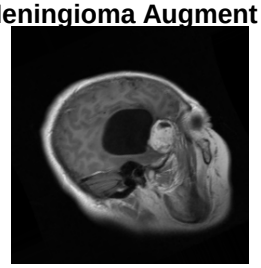
Meningioma Augment 2



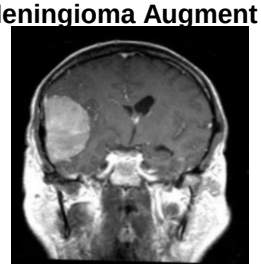
Meningioma Augment 3



Meningioma Augment 4

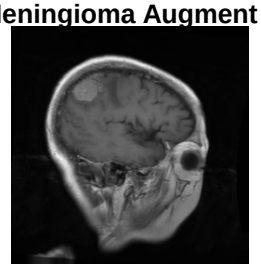


Meningioma Augment 5

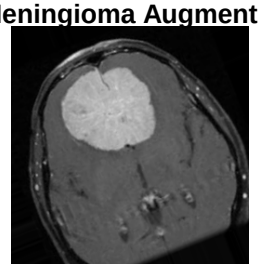


Meningioma Augment 6

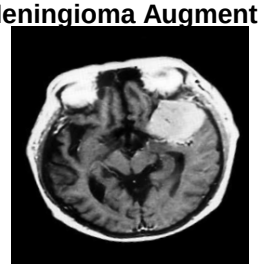
Meningioma Augment 7



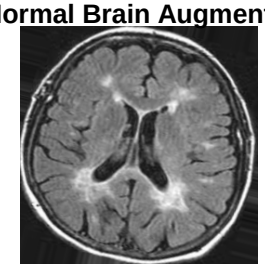
Meningioma Augment 8



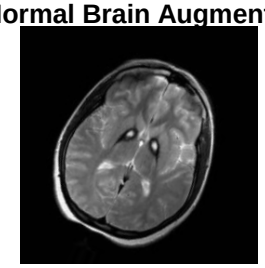
Meningioma Augment 9



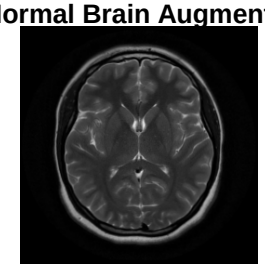
Normal Brain Augment 1



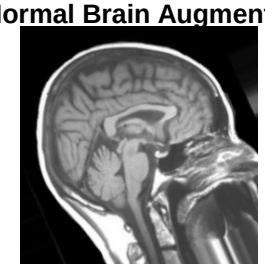
Normal Brain Augment 2



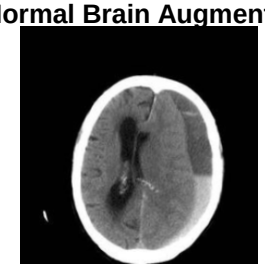
Normal Brain Augment 3



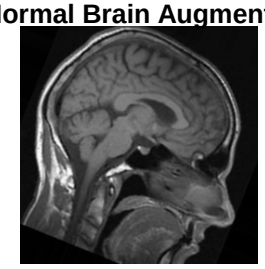
Normal Brain Augment 4



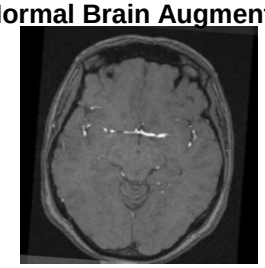
Normal Brain Augment 5



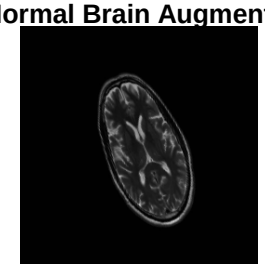
Normal Brain Augment 6



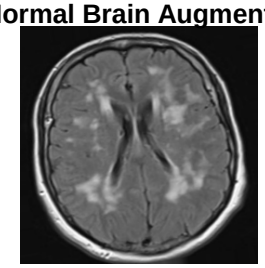
Normal Brain Augment 7

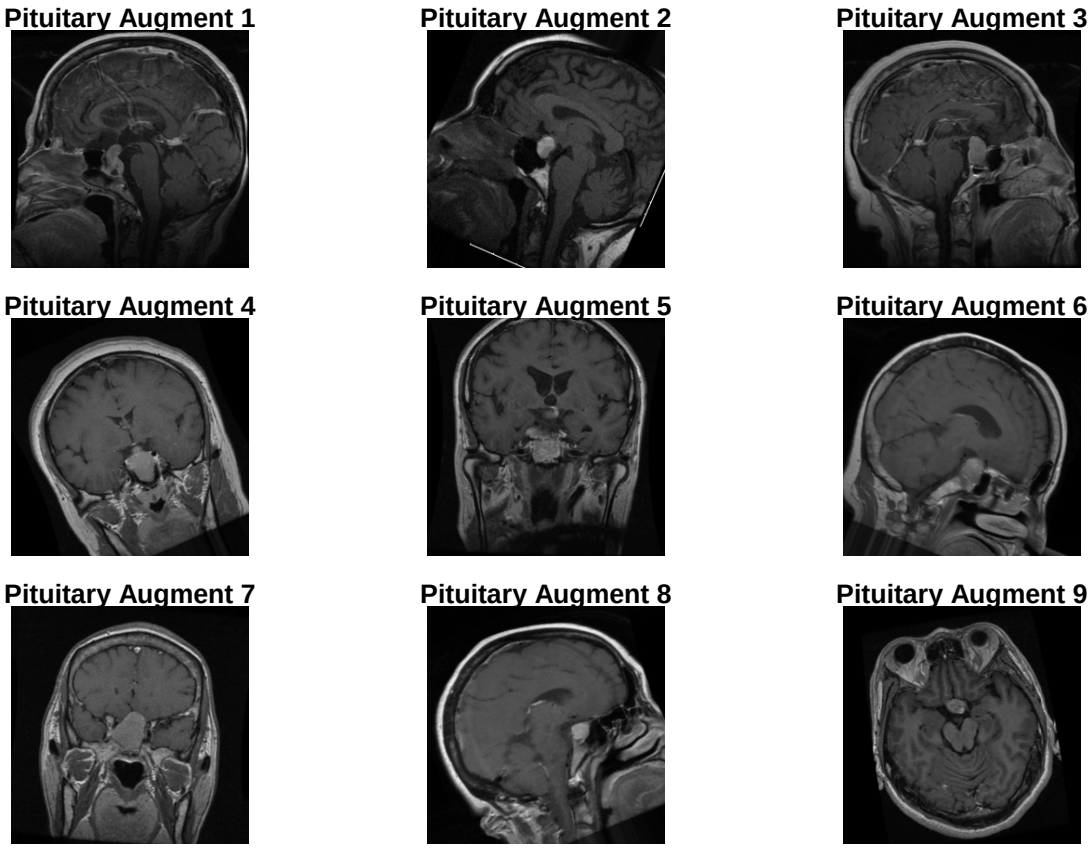


Normal Brain Augment 8



Normal Brain Augment 9





4.3 Convolutional Neural Network Analysis

Convolutional neural networks leverage many techniques to extract relevant features from a raw input file. These techniques are represented in the layers of the network's architecture. Convolutional layers apply a set of learnable filters over the input. Each filter slides over the image, performing a convolution operation to uncover local features. Pooling layers are used to reduce the spatial dimension of the image while retaining features of importance. Pooling decreases computational complexity as well as providing a control to over fitting. Multiple layers of convolution and pooling provide a deeper representation of the data, allowing for the network to capture increasingly complex and abstract features. A flattening layer is used to reshape the output of the previous layer feature maps into a format capable of being supplied to a fully connected layer. Fully connected layers represent a traditional neural network. High level features are extracted and mapped to target classes. The output layer generates the model's final prediction. To account for model over fitting, dropout layers can prove effective.

In an effort to find the best model in terms of classification accuracy, precision, specificity, and f1 score, two different convolutional neural network architectures were explored. The different model designs are listed in the table below.

ReLu was used as the activation function for all convolutional layers in both designs. Each convolutional layer was followed by a pooling layer. A dropout layer was periodically introduced to mitigate the likelihood of over fitting to the training dataset. Finally, the models were flattened and connected to a SoftMax activation function to classify the images into the four classes: Glioma, Meningioma, No tumor, and Pituitary.

Model 1	Model 2
Input Image: 256 x 256 x 1	**Input Image:** 256 x 256 x 1

Model 1	Model 2
Convolutional Layer: Filters = 32, Kernel = 4 x 4, Padding = same, Activation = ReLu	**Convolutional Layer:** Filters = 16, Kernel = 5 x 5, Padding = same, Activation = ReLu
Pooling Layer: Size = 3 x 3	**Pooling Layer:** Size = 4 x 4
Dropout Layer: Rate = 0.1	**Dropout Layer:** Rate = 0.1
Convolutional Layer: Filters = 32, Kernel = 4 x 4, Padding = same, Activation = ReLu	**Convolutional Layer:** Filters = 16, Kernel = 5 x 5, Padding = same, Activation = ReLu
Pooling Layer: Size = 3 x 3	**Pooling Layer:** Size = 4 x 4
Dropout Layer: Rate = 0.1	**Flatten Layer**
Convolutional Layer: Filters = 32, Kernel = 4 x 4, Padding = same, Activation = ReLu	**Dense Layer:** Units = 12, Activation = ReLu
Pooling Layer: Size = 3 x 3	**Dense Layer:** Units = 4, Activation = Softmax
Flatten Layer	
Dense Layer: Units = 16, Activation = ReLu	
Dense Layer: Units = 4, Activation = Softmax	

5 Experimentation and Results

5.1 Model 1

Model 1 was optimized via the Adam algorithm with a learning rate of $1e^{-3}$. The Adam algorithm aims to minimize the loss function which was chosen to be categorical crossentropy. The model was trained with the image augmentation generator and assigned to train for 50 epochs with a batch size of 20 images. 130 batches were used to complete each epoch. The model made large improvements in performance within the first 15 epochs. For the remaining 35 epochs, performance improvement was minimal. Training accuracy peaked at 97% and validation accuracy peaked at 88%. The model appeared to overfit to the training data as the loss function and accuracy metric both increased as training continued.

The model performance on the testing data that was not utilized during training is provided in the 'Model Selection' section below.

```
## Model: "CNN_Model"
## _________________________________________________________________________
##  Layer (type)                 Output Shape              Param #
## =========================================================================
##  conv2d_2 (Conv2D)            (None, 256, 256, 32)      544
##  max_pooling2d_2 (MaxPooling2D)  (None, 85, 85, 32)     0
##  dropout_1 (Dropout)          (None, 85, 85, 32)        0
##  conv2d_1 (Conv2D)            (None, 85, 85, 32)        16416
##  max_pooling2d_1 (MaxPooling2D)  (None, 28, 28, 32)     0
##  dropout (Dropout)            (None, 28, 28, 32)        0
##  conv2d (Conv2D)              (None, 28, 28, 32)        16416
##  max_pooling2d (MaxPooling2D)  (None, 9, 9, 32)         0
##  flatten (Flatten)            (None, 2592)              0
##  dense (Dense)                (None, 16)                41488
##  Output (Dense)               (None, 4)                 68
## =========================================================================
## Total params: 74,932
## Trainable params: 74,932
```

```
## Non-trainable params: 0
## _________________________________________________________________________
```

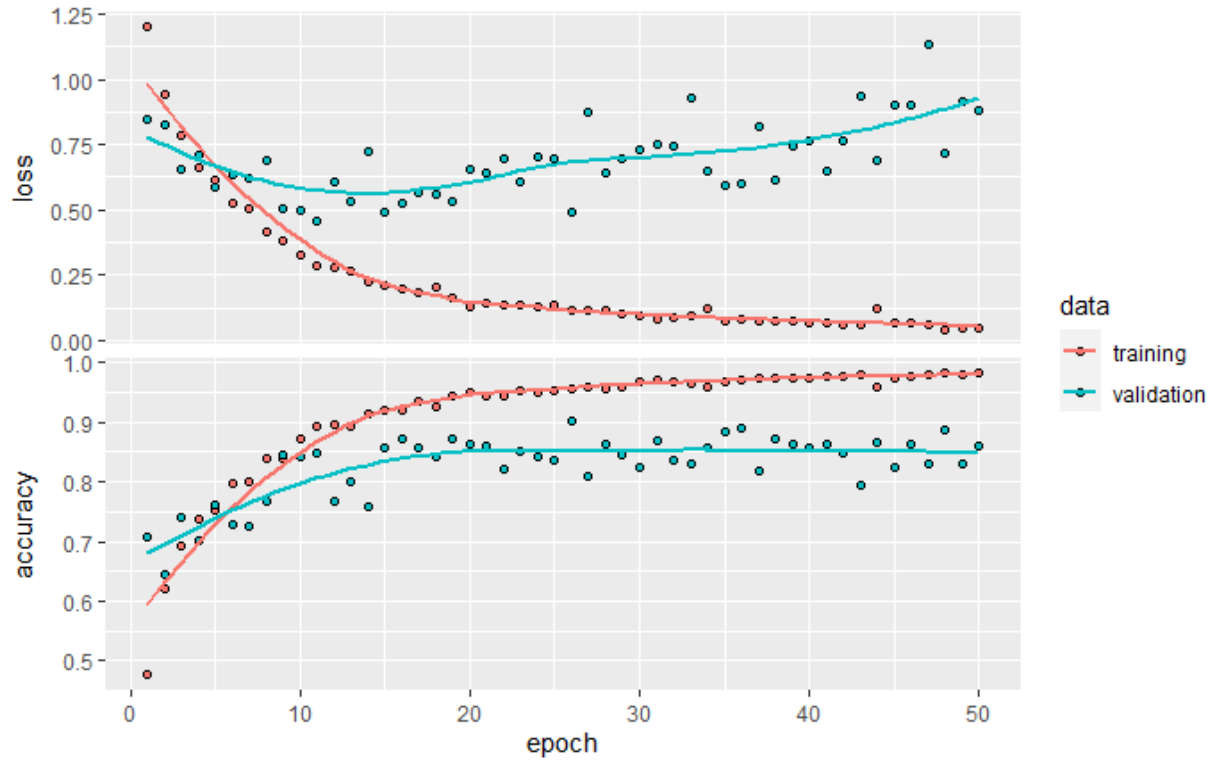


Figure 1: Model 1: Training History

5.2 Model 2

Model 2 had different optimizers to Model 1. The Adam algorithm with a learning rate of $1e^{-3}$ was used to mitigate the loss function which was chosen to be categorical crossentropy. The model was trained with the same image augmentation generator as Model 1. The model was trained using 50 epochs with a batch size of 32 images. Each epoch used 80 batches to complete 1 cycle. The model converged within the 50 epochs, generating a training accuracy of 96% and a validation accuracy of 83%.

The model performance on the testing data that was not utilized during training is provided in the 'Model Selection' section below.

```
## Model: "CNN_Model2"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  conv2d_4 (Conv2D)                  (None, 256, 256, 16)            416
##  max_pooling2d_4 (MaxPooling2D)     (None, 64, 64, 16)              0
##  dropout_2 (Dropout)                (None, 64, 64, 16)              0
##  conv2d_3 (Conv2D)                  (None, 64, 64, 16)              6416
##  max_pooling2d_3 (MaxPooling2D)     (None, 16, 16, 16)              0
##  flatten_1 (Flatten)                (None, 4096)                    0
```

```
##  dense_1 (Dense)                  (None, 12)                    49164
##  Output (Dense)                   (None, 4)                     52
## ================================================================================
## Total params: 56,048
## Trainable params: 56,048
## Non-trainable params: 0
## ________________________________________________________________________________
```

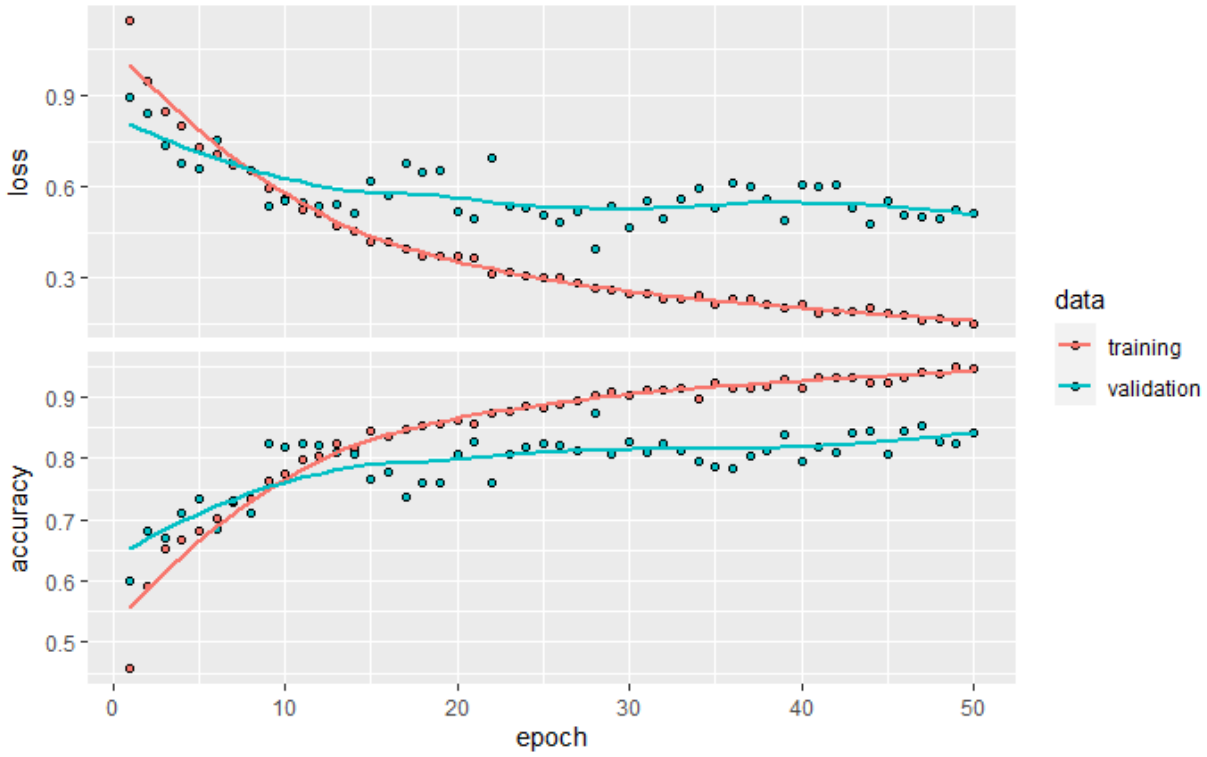


Figure 2: Model 2: Training History

5.3 Model Selection

The performance metrics from the two CNN models are in the tables below. Each model was evaluated using accuracy, sensitivity, specificity, precision, and F1 score based on the results from testing data predictions. The testing data was not used during the training phase and is explicitly used for evaluation purposes.

The overall testing data prediction accuracy was 74.87% for Model 1 and 68.27% for Model 2. Both models struggled to classify the tumor type of glioma with Model 1 performing slightly better with a F1 score of 37.4% compared to Model 2's F1 score of 27.64%. Model 1 is exceptionally good at mitigating false positives for pituitary tumors as the specificity is 98%. Model 2 slightly outperforms Model 1 at minimizing false positive for meningioma tumors by 2.9%. Model 2 also had higher precision by 1.5% when classifying for the meningioma class. Model 1 achieved a sensitivity of 96.5% for meningioma tumors, indicating that almost all cases were correctly predicted.

Out of the two proposed convolutional neural network models, Model 1 has generated superior performance for almost all metrics on unknown data. Model 1 is the most promising prospect in classifying brain MRI scans into the four categories explored in this project. Model 1 was exceptionally strong at classifying three of the four categories.

5.3.0.1 Model 1 Metrics

	x
Accuracy	0.748731

	Sensitivity	Specificity	Precision	F1
Class: glioma_tumor	0.2300000	1.0000000	1.0000000	0.3739837
Class: meningioma_tumor	0.9652174	0.8207885	0.6894410	0.8043478
Class: no_tumor	1.0000000	0.8512111	0.7094595	0.8300395
Class: pituitary_tumor	0.7567568	0.9812500	0.9032258	0.8235294

5.3.0.2 Model 2 Metrics

	x
Accuracy	0.6827411

	Sensitivity	Specificity	Precision	F1
Class: glioma_tumor	0.1700000	0.9795918	0.7391304	0.2764228
Class: meningioma_tumor	0.8695652	0.8494624	0.7042254	0.7782101
Class: no_tumor	1.0000000	0.7647059	0.6069364	0.7553957
Class: pituitary_tumor	0.6351351	0.9718750	0.8392857	0.7230769

6 Conclusions and Next Steps

The diagnosis of brain tumors in an early stage can vastly improve survival rates. Deep learning has dramatically increased the effectiveness of computer vision tasks, including those in the medical domain. There are identifiable differences between normal brain MRI scans, glioma, meningioma, and pituitary tumors MRI scans that radiologists use for diagnosis. This project developed deep learning models with the goal of aiding radiologists in brain tumor classification. The dataset was small for a deep learning model, containing only one 3264 image file. Image augmentation was used to expand the number of training images the models received. The best model achieved an overall accuracy of 74.87% on the testing data. The model shows that there was overfitting to the training data. With more model tuning the model will likely improve its ability to generalize to unknown data. Both models were unable to capture the nuances for classifying glioma tumors as Model 1 had a F1 score of 37.4% and Model 2 of 27.64%. It is likely that the glioma tumor class did not have enough diverse images to successfully generalize the classification process.

Future research for this project would include:

- Transfer Learning using prebuilt CNN architetures to further optimize training the data. Models like ResNet could be leveraged.
- Expand the number of classes to include more types of brain tumors.
- Using more diverse image augmentation techniques like changing the color space, sharpness, and contrast of the image. Even more complex augmentation techniques could be used like image mixing.
- Develop more comprehensive model architetures with more parameters to train. The models used in this paper were limited to 75,000 parameters and 56,000 parameters respectively.

It is understood that this project contained several limitations. It is important to have a large amount of data when working with deep learning algorithms to yield the most accurate results. This project was limited in terms of the amount of usable data due to the difficulty of obtaining publicly available medical data. Most medical imaging datasets are rather small in the scope of deep learning. Within this project, attempts were made to mitigate the effects of the small dataset, thereby increasing overall accuracy. However, it should be noted that a larger dataset would likely perform better than the enhanced dataset used here. Another limitation was processing resources and time. As deep learning algorithms like CNN take a large amount of computer resources and time to train, the proposed models contained less than 100,000 parameters. This guaranteed a run time that was commensurate with the time allocated for the project.

7 Data Availability

The dataset of JPEG brain MRI images are available of kaggle.com: https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri

8 Bibliography

Bhuvaji, S. (2020, May 24). Brain tumor classification (MRI). Kaggle. https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C. S., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M. K., Pei, J., Ting, M. Y. L., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., ... Zhang, K. (2018). Identifying medical diagnoses and treatable diseases by image-based Deep Learning. Cell, 172(5), 1122–1131. https://doi.org/10.1016/j.cell.2018.02.010

Khan, M. H.-M., Boodoo-Jahangeer, N., Dullull, W., Nathire, S., Gao, X., Sinha, G. R., & Nagwanshi, K. K. (2021, August 26). Multi- class classification of breast cancer abnormalities using deep convolutional neural network (CNN). PLOS ONE. https://doi.org/10.1371%2Fjournal.pone.0256500

Kiryati, N., & Landau, Y. (2021, August 20). Dataset growth in Medical Image Analysis Research. MDPI. https://www.mdpi.com/2313-433X/7/8/155

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Pan, L., Qin, J., Chen, H., Xiang, X., Li, C., & Chen, R. (2019). Image augmentation-based food recognition with Convolutional Neural Networks. Tech Science Press. https://www.techscience.com/cmc/v59n1/27933

Roser, M., & Ritchie, H. (2015, July 3). Cancer. Our World in Data. https://ourworldindata.org/cancer#:~:text=About%20ten%20million%20people%20die,health%20problems%20in%20the%20world

Thambawita, V., Strümke, I., Hicks, S. A., Halvorsen, P., Parasa, S., & Riegler, M. A. (2021, November 24). Impact of image resolution on deep learning performance in Endoscopy Image Classification: An experimental study using a large dataset of endoscopic images. Diagnostics (Basel, Switzerland). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/

Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2022, April 19). Image Data Augmentation for Deep Learning: A Survey. arXiv.org. https://arxiv.org/abs/2204.08610

9 Appendix

```
library(tidyverse)
library(jpeg)
library(rlist)
library(imager)
library(EBImage)
library(SpatialPack)
library(tensorflow)
library(reticulate)
library(keras)
library(caret)
```

```r
tensorflow::set_random_seed(123)
use_virtualenv("r-reticulate")
tensorflow::set_random_seed(123)


y.labels <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Science\\Semester

# Training

train.names.glioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data S
train.names.meningioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Da
train.names.no_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Scien
train.names.pituitary_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data


glioma_tumor <- rep("glioma_tumor", each = length(train.names.glioma_tumor))
meningioma_tumor <- rep("meningioma_tumor", each = length(train.names.meningioma_tumor))
no_tumor <- rep("no_tumor", each = length(train.names.no_tumor))
pituitary_tumor <- rep("pituitary_tumor", each = length(train.names.pituitary_tumor))

train.y.list <- c(glioma_tumor, meningioma_tumor, no_tumor, pituitary_tumor)


# Testing

test.names.glioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Sc
test.names.meningioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data
test.names.no_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Scienc
test.names.pituitary_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data


glioma_tumor2 <- rep("glioma_tumor", each = length(test.names.glioma_tumor))
meningioma_tumor2 <- rep("meningioma_tumor", each = length(test.names.meningioma_tumor))
no_tumor2 <- rep("no_tumor", each = length(test.names.no_tumor))
pituitary_tumor2 <- rep("pituitary_tumor", each = length(test.names.pituitary_tumor))

test.y.list <- c(glioma_tumor2, meningioma_tumor2, no_tumor2, pituitary_tumor2)


# Validation

val.names.glioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Sci
val.names.meningioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data
val.names.no_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Science
val.names.pituitary_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data


glioma_tumor3 <- rep("glioma_tumor", each = length(val.names.glioma_tumor))
meningioma_tumor3 <- rep("meningioma_tumor", each = length(val.names.meningioma_tumor))
no_tumor3 <- rep("no_tumor", each = length(val.names.no_tumor))
pituitary_tumor3 <- rep("pituitary_tumor", each = length(val.names.pituitary_tumor))

val.y.list <- c(glioma_tumor3, meningioma_tumor3, no_tumor3, pituitary_tumor3)
```

```
dataset.type <- c("Training", "Training", "Training", "Training",
                  "Testing", "Testing", "Testing", "Testing",
                  "Validation", "Validation", "Validation", "Validation")
category <- c("Glioma" , "Meningioma" , "Normal", "Pituitary",
              "Glioma" , "Meningioma" , "Normal", "Pituitary",
              "Glioma" , "Meningioma" , "Normal", "Pituitary")
value <- c(length(train.names.glioma_tumor), length(train.names.meningioma_tumor), length(train.names.no
           length(test.names.glioma_tumor), length(test.names.meningioma_tumor), length(test.names.no_tu
           length(val.names.glioma_tumor), length(val.names.meningioma_tumor), length(val.names.no_tumo
data <- data.frame(dataset.type, category, value)


ggplot(data, aes(fill=category, y=value, x=dataset.type)) +
    geom_bar(position="dodge", stat="identity") +
  ggtitle("Category Distribution")



# Load in Data and Convert to Grayscale


############################################################

# glioma

train.cnn.glioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(train.names.glioma_tumor)){
  image.raw <- readJPEG(train.names.glioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  train.cnn.glioma[[k]] <- im
}


############################################################

# meningioma
```

```
train.cnn.meningioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(train.names.meningioma_tumor)){
  image.raw <- readJPEG(train.names.meningioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  train.cnn.meningioma[[k]] <- im
}


############################################################

# no



train.cnn.no <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(train.names.no_tumor)){
  image.raw <- readJPEG(train.names.no_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  train.cnn.no[[k]] <- im
```

```
}



############################################################

# pituitary



train.cnn.pituitary <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(train.names.pituitary_tumor)){
  image.raw <- readJPEG(train.names.pituitary_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  train.cnn.pituitary[[k]] <- im
}









# Testing


############################################################

# glioma

test.cnn.glioma <- list()

im <- matrix(nrow = 256, ncol = 256)
```

```
for(k in 1:length(test.names.glioma_tumor)){
  image.raw <- readJPEG(test.names.glioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.glioma[[k]] <- im
}


################################################################

# meningioma



test.cnn.meningioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.meningioma_tumor)){
  image.raw <- readJPEG(test.names.meningioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.meningioma[[k]] <- im
}


################################################################

# no
```

```r
test.cnn.no <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.no_tumor)){
  image.raw <- readJPEG(test.names.no_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.no[[k]] <- im
}



############################################################

# pituitary



test.cnn.pituitary <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.pituitary_tumor)){
  image.raw <- readJPEG(test.names.pituitary_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
```

```
  test.cnn.pituitary[[k]] <- im
}




# Validation


############################################################

# glioma

val.cnn.glioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(val.names.glioma_tumor)){
  image.raw <- readJPEG(val.names.glioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  val.cnn.glioma[[k]] <- im
}


############################################################

# meningioma



val.cnn.meningioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(val.names.meningioma_tumor)){
  image.raw <- readJPEG(val.names.meningioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
```

```
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  val.cnn.meningioma[[k]] <- im
}


############################################################

# no



val.cnn.no <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(val.names.no_tumor)){
  image.raw <- readJPEG(val.names.no_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  val.cnn.no[[k]] <- im
}



############################################################

# pituitary
```

```
val.cnn.pituitary <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(val.names.pituitary_tumor)){
  image.raw <- readJPEG(val.names.pituitary_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  val.cnn.pituitary[[k]] <- im
}


train.cnn <- c(train.cnn.glioma, train.cnn.meningioma, train.cnn.no, train.cnn.pituitary)
test.cnn <- c(test.cnn.glioma, test.cnn.meningioma, test.cnn.no, test.cnn.pituitary)
val.cnn <- c(val.cnn.glioma, val.cnn.meningioma, val.cnn.no, val.cnn.pituitary)


image1 <- readJPEG(train.names.glioma_tumor[1])

#dim(image1)

image2 <- readJPEG(train.names.no_tumor[300])

#dim(image2)


train.data.glioma_tumor1 <- train.names.glioma_tumor[1:9] %>%
  map(readJPEG)

train.data.meningioma_tumor1 <- train.names.meningioma_tumor[1:9] %>%
  map(readJPEG)

train.data.no_tumor1 <- train.names.no_tumor[1:9] %>%
  map(readJPEG)

train.data.pituitary_tumor1 <- train.names.pituitary_tumor[1:9] %>%
  map(readJPEG)

par(mfrow=c(3, 3), mar=c(0, 0, 3, 0))
sapply(1:9, function(i) {
  plot.new()
```

```
  plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
  rasterImage(train.data.glioma_tumor1[[i]], 0, 0, 1, 1)
  title(paste0('Train Glioma Tumor ', i), font.main=2)
})

sapply(1:9, function(i) {
  plot.new()
  plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
  rasterImage(train.data.meningioma_tumor1[[i]], 0, 0, 1, 1)
  title(paste0('Train Meningioma Tumor ', i), font.main=2)
})


sapply(1:9, function(i) {
  plot.new()
  plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
  rasterImage(train.data.no_tumor1[[i]], 0, 0, 1, 1)
  title(paste0('Train Normal Scan ', i), font.main=2)
})


sapply(1:9, function(i) {
  plot.new()
  plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
  rasterImage(train.data.pituitary_tumor1[[i]], 0, 0, 1, 1)
  title(paste0('Train Pituitary Tumor ', i), font.main=2)
})



par(mfrow=c(2, 2), mar=c(0, 0, 3, 0))
plot.new()
rasterImage(train.data.glioma_tumor1[[1]], xleft = 0, xright = 1,
            ytop = 0, ybottom = 1, interpolate = FALSE)
title(paste0("RGB Colors and Native Scale"))

plot.new()
rasterImage(train.cnn.glioma[[1]], xleft = 0, xright = 1,
            ytop = 0, ybottom = 1, interpolate = FALSE)
title(paste0("Grayscale and Standardized Scale"))


plot.new()
rasterImage(train.data.no_tumor1[[1]], xleft = 0, xright = 1,
            ytop = 0, ybottom = 1, interpolate = FALSE)
title(paste0("RGB Colors and Native Scale"))


plot.new()
rasterImage(train.cnn.no[[1]], xleft = 0, xright = 1,
            ytop = 0, ybottom = 1, interpolate = FALSE)
title(paste0("Grayscale and Standardized Scale"))
```

```
use_virtualenv("r-reticulate")
tensorflow::set_random_seed(123)


train.data.glioma <- array_reshape(train.cnn.glioma,
                                   dim = c(length(train.cnn.glioma), 256, 256, 1)
                                   )

train.data.meningioma <- array_reshape(train.cnn.meningioma,
                                   dim = c(length(train.cnn.meningioma), 256, 256, 1)
                                   )

train.data.no <- array_reshape(train.cnn.no,
                                   dim = c(length(train.cnn.no), 256, 256, 1)
                                   )

train.data.pituitary <- array_reshape(train.cnn.pituitary,
                                   dim = c(length(train.cnn.pituitary), 256, 256, 1)
                                   )

datagen <- image_data_generator(
  rescale = 1/1,
  rotation_range = 30,
  horizontal_flip = TRUE,
  fill_mode = "nearest"
)

augmentation_generator <- flow_images_from_data(
  train.data.glioma,
  generator = datagen,
  batch_size = 1,
  seed = 123
)


par(mfrow = c(3,3), pty = 's', mar = c(1, 0, 1, 0))
for (i in 1:9) {
aug_img <- generator_next(augmentation_generator)
plot(as.raster(aug_img[1, , , ]))
title(paste0('Train Glioma Augmentation ', i), font.main=2)
}

augmentation_generator <- flow_images_from_data(
  train.data.meningioma,
  generator = datagen,
  batch_size = 1,
  seed = 123
)


par(mfrow = c(3,3), pty = 's', mar = c(1, 0, 1, 0))
for (i in 1:9) {
```

```
aug_img <- generator_next(augmentation_generator)
plot(as.raster(aug_img[1, , , ]))
title(paste0('Train Meningioma Augmentation ', i), font.main=2)
}

augmentation_generator <- flow_images_from_data(
  train.data.no,
  generator = datagen,
  batch_size = 1,
  seed = 123
)


par(mfrow = c(3,3), pty = 's', mar = c(1, 0, 1, 0))
for (i in 1:9) {
aug_img <- generator_next(augmentation_generator)
plot(as.raster(aug_img[1, , , ]))
title(paste0('Train Normal Brain Augmentation ', i), font.main=2)
}



augmentation_generator <- flow_images_from_data(
  train.data.pituitary,
  generator = datagen,
  batch_size = 1,
  seed = 123
)


par(mfrow = c(3,3), pty = 's', mar = c(1, 0, 1, 0))
for (i in 1:9) {
aug_img <- generator_next(augmentation_generator)
plot(as.raster(aug_img[1, , , ]))
title(paste0('Train Pituitary Augmentation ', i), font.main=2)
}



train_dir <-  file.path("C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Science\\Semester4\\DAT

validation_dir <-  file.path("C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Science\\Semester4


use_virtualenv("r-reticulate")
tensorflow::set_random_seed(123)

################################################
# Training Data

train_datagen <- image_data_generator(
  rescale = 1/255,
  rotation_range = 30,
```

```
  horizontal_flip = TRUE,
  fill_mode = "nearest"
)



train_generator <- flow_images_from_directory(
  directory = train_dir,
  generator = train_datagen,
  target_size = c(256, 256),
  batch_size = 20,
  class_mode = 'categorical',
  color_mode = "grayscale",
  seed = 123
)



################################################
# Validation Data

validation_datagen <- image_data_generator(rescale = 1/255)


# Flow training images in batches of 20 using test_datagen generator
validation_generator <- flow_images_from_directory(
  directory = validation_dir,
  generator = validation_datagen,
  target_size = c(256, 256),
  batch_size = 20,
  class_mode = 'categorical',
  color_mode = "grayscale",
  seed = 123
)


model <- keras_model_sequential(name = "CNN_Model") %>%

  layer_conv_2d(filters = 32,
                kernel_size = c(4,4),
                padding = "same", activation = "relu",
                input_shape = c(256, 256, 1)
                ) %>%

  layer_max_pooling_2d(pool_size = c(3,3)) %>%

  layer_dropout(rate = 0.1) %>%

  layer_conv_2d(filters = 32,
                kernel_size = c(4,4),
                padding = "same", activation = "relu",
                input_shape = c(256, 256, 1)
                ) %>%
```

```
  layer_max_pooling_2d(pool_size = c(3,3)) %>%

  layer_dropout(rate = 0.1) %>%

  layer_conv_2d(filters = 32,
                kernel_size = c(4,4),
                padding = "same", activation = "relu",
                input_shape = c(256, 256, 1)
                ) %>%

  layer_max_pooling_2d(pool_size = c(3,3)) %>%


  layer_flatten() %>%

  layer_dense(units = 16,
              activation = "relu") %>%

  layer_dense(units = 4,
              activation = "softmax",
              name = "Output"
              )

model


start_time <- Sys.time()

model %>%
  compile(loss = "categorical_crossentropy",
          optimizer = optimizer_adam(learning_rate = 0.001),
          metrics = "accuracy"
          )


train_history <- model %>%
 fit_generator(
  generator = train_generator,
  steps_per_epoch = 130,
  epochs = 50,
  validation_data = validation_generator,
  validation_steps = 15
      )


end_time <- Sys.time()
end_time - start_time

plot(train_history)

# model %>% save_model_hdf5("model1.h5")
```

```r
# Testing

test.names.glioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Sci
test.names.meningioma_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data
test.names.no_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data Scienc
test.names.pituitary_tumor <- list.files(path = "C:\\Users\\ericl\\OneDrive\\Documents\\CUNY MS in Data


glioma_tumor2 <- rep("glioma_tumor", each = length(test.names.glioma_tumor))
meningioma_tumor2 <- rep("meningioma_tumor", each = length(test.names.meningioma_tumor))
no_tumor2 <- rep("no_tumor", each = length(test.names.no_tumor))
pituitary_tumor2 <- rep("pituitary_tumor", each = length(test.names.pituitary_tumor))

test.y.list <- c(glioma_tumor2, meningioma_tumor2, no_tumor2, pituitary_tumor2)

# Testing


############################################################

# glioma

test.cnn.glioma <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.glioma_tumor)){
  image.raw <- readJPEG(test.names.glioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.glioma[[k]] <- im
}


############################################################

# meningioma



test.cnn.meningioma <- list()
```

```
im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.meningioma_tumor)){
  image.raw <- readJPEG(test.names.meningioma_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.meningioma[[k]] <- im
}


############################################################

# no



test.cnn.no <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.no_tumor)){
  image.raw <- readJPEG(test.names.no_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.no[[k]] <- im
}
```

```
#############################################################

# pituitary



test.cnn.pituitary <- list()

im <- matrix(nrow = 256, ncol = 256)

for(k in 1:length(test.names.pituitary_tumor)){
  image.raw <- readJPEG(test.names.pituitary_tumor[k])
  image.resize <- resize(image.raw, h = 256, w = 256)
  image.grayscale <- RGB2gray(image.resize)
  x <- as.numeric(c(image.grayscale))

  j <- 1
  for(i in 1:256){



  im[,i] <- x[j:(j+255)]

  j <- j+256

  }
  test.cnn.pituitary[[k]] <- im
}


test.cnn <- c(test.cnn.glioma, test.cnn.meningioma, test.cnn.no, test.cnn.pituitary)


for(i in length(test.cnn)){
  test.cnn[[i]] <- test.cnn[[i]] / 255
}


test.data <- array_reshape(test.cnn,
                           dim = c(length(test.cnn), 256, 256, 1))

pred.cnn <- model %>% predict(test.data) %>% k_argmax()

pred.factor <- as.factor(as.numeric(pred.cnn))

levels(pred.factor) <- list(glioma_tumor = "0",
                            meningioma_tumor = "1",
                            no_tumor = "2",
                            pituitary_tumor = "3")

cm <- confusionMatrix(data = pred.factor, reference = as.factor(test.y.list))

cm
```

```
model1.loaded <- load_model_hdf5('model1.h5')


test.data <- array_reshape(test.cnn,
                           dim = c(length(test.cnn), 256, 256, 1))

pred.cnn1.loaded <- model1.loaded %>% predict(test.data) %>% k_argmax()

pred.factor1.loaded <- as.factor(as.numeric(pred.cnn1.loaded))

levels(pred.factor1.loaded) <- list(glioma_tumor = "0",
                            meningioma_tumor = "1",
                            no_tumor = "2",
                            pituitary_tumor = "3")

cm1.loaded <- confusionMatrix(data = pred.factor1.loaded, reference = as.factor(test.y.list))

cm1.loaded



use_virtualenv("r-reticulate")

tensorflow::set_random_seed(123)

model2 <- keras_model_sequential(name = "CNN_Model2") %>%

  layer_conv_2d(filters = 16,
                kernel_size = c(5,5),
                padding = "same", activation = "relu",
                input_shape = c(256, 256, 1)
                ) %>%

  layer_max_pooling_2d(pool_size = c(4,4)) %>%

  layer_dropout(rate = 0.1) %>%

  layer_conv_2d(filters = 16,
                kernel_size = c(5,5),
                padding = "same", activation = "relu",
                input_shape = c(256, 256, 1)
                ) %>%

  layer_max_pooling_2d(pool_size = c(4,4)) %>%



  layer_flatten() %>%

  layer_dense(units = 12,
              activation = "relu") %>%
```

```
  layer_dense(units = 4,
              activation = "softmax",
              name = "Output"
              )

model2

use_virtualenv("r-reticulate")
tensorflow::set_random_seed(123)

################################################
# Training Data

train_datagen <- image_data_generator(
  rescale = 1/255,
  rotation_range = 30,
  horizontal_flip = TRUE,
  fill_mode = "nearest"
)



train_generator <- flow_images_from_directory(
  directory = train_dir,
  generator = train_datagen,
  target_size = c(256, 256),
  batch_size = 32,
  class_mode = 'categorical',
  color_mode = "grayscale",
  seed = 123
)



################################################
# Validation Data

validation_datagen <- image_data_generator(rescale = 1/255)


# Flow training images in batches of 20 using test_datagen generator
validation_generator <- flow_images_from_directory(
  directory = validation_dir,
  generator = validation_datagen,
  target_size = c(256, 256),
  batch_size = 32,
  class_mode = 'categorical',
  color_mode = "grayscale",
  seed = 123
)


start_time <- Sys.time()
```

```
model2 %>%
  compile(loss = "categorical_crossentropy",
          optimizer = optimizer_adam(learning_rate = 0.001),
          metrics = "accuracy"
          )


train_history2 <- model2 %>%
 fit_generator(
  generator = train_generator,
  steps_per_epoch = 80,
  epochs = 50,
  validation_data = validation_generator,
  validation_steps = 9
      )


end_time <- Sys.time()
end_time - start_time

plot(train_history2)

# model2 %>% save_model_hdf5("model2.h5")

test.data <- array_reshape(test.cnn,
                                dim = c(length(test.cnn), 256, 256, 1))

pred.cnn2 <- model2 %>% predict(test.data) %>% k_argmax()

pred.factor2 <- as.factor(as.numeric(pred.cnn2))

levels(pred.factor2) <- list(glioma_tumor = "0",
                             meningioma_tumor = "1",
                             no_tumor = "2",
                             pituitary_tumor = "3")

cm2 <- confusionMatrix(data = pred.factor2, reference = as.factor(test.y.list))

cm2

model2.loaded <- load_model_hdf5('model2.h5')


test.data <- array_reshape(test.cnn,
                                dim = c(length(test.cnn), 256, 256, 1))

pred.cnn2.loaded <- model2.loaded %>% predict(test.data) %>% k_argmax()

pred.factor2.loaded <- as.factor(as.numeric(pred.cnn2.loaded))

levels(pred.factor2.loaded) <- list(glioma_tumor = "0",
                             meningioma_tumor = "1",
                             no_tumor = "2",
```

```
                            pituitary_tumor = "3")

cm2.loaded <- confusionMatrix(data = pred.factor2.loaded, reference = as.factor(test.y.list))

cm2.loaded

kableExtra::kable(cm1.loaded$overall[1])
kableExtra::kable(cm1.loaded$byClass[,c(1,2,5,7)])


kableExtra::kable(cm2.loaded$overall[1])
kableExtra::kable(cm2.loaded$byClass[,c(1,2,5,7)])
```