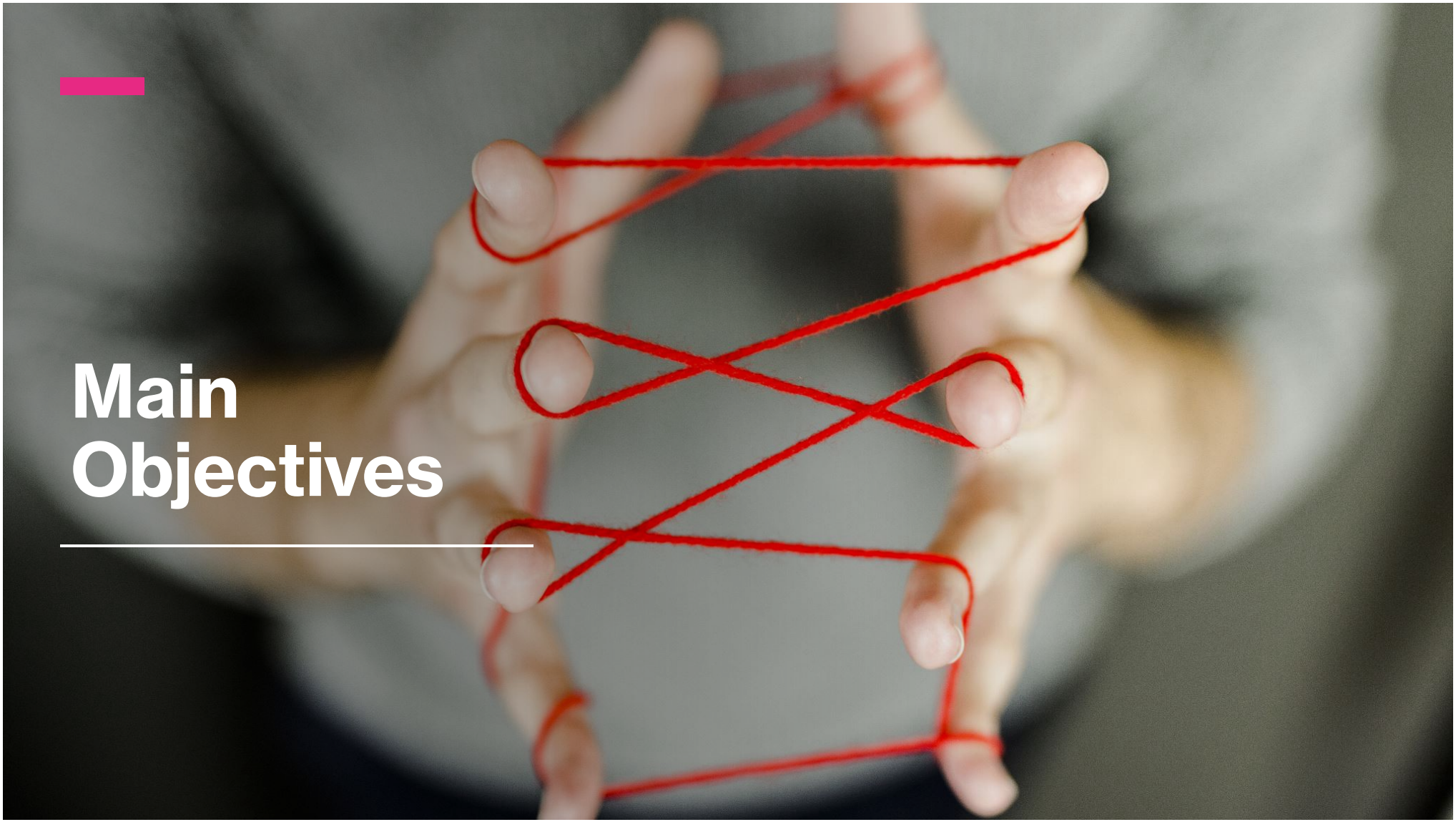
Deep Learning – Brain Tumor Detection

Ben Cottrell


Main
Objectives

A Brain tumor is considered as one of the aggressive diseases, among children and adults. Brain tumors account for 85 to 90 percent of all primary Central Nervous System(CNS) tumors. Every year, around 11,700 people are diagnosed with a brain tumor. The 5-year survival rate for people with a cancerous brain or CNS tumor is approximately 34 percent for men and36 percent for women. Brain Tumors are classified as: Benign Tumor, Malignant Tumor, Pituitary Tumor, etc. Proper treatment, planning, and accurate diagnostics should be implemented to improve the life expectancy of the patients. The best technique to detect brain tumors is Magnetic Resonance Imaging (MRI). A huge amount of image data is generated through the scans. These images are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties.

Application of automated classification techniques using Machine Learning(ML) and Artificial Intelligence(AI)has consistently shown higher accuracy than manual classification. Hence, proposing a system performing detection and classification by using Deep Learning Algorithms using Convolution-Neural Network (CNN), Artificial Neural Network (ANN), and Transfer-Learning (TL) would be helpful to doctors all around the world.

In the analysis we will explore the MRI Brain images dataset in more detail to identify a robust deep learning model which aims to help doctors with brain tumor diagnosis.

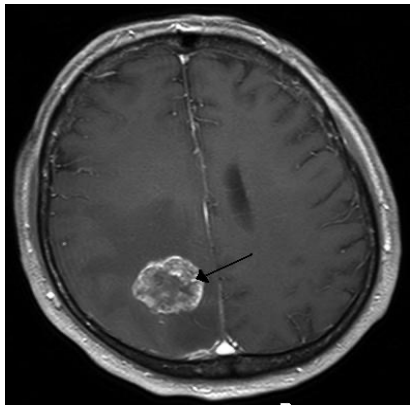
Source: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection


Dataset

What is a Brain Tumor?

A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain metastasis tumors. All types of brain tumors may produce symptoms that vary depending on the part of the brain involved.

These symptoms may include headaches, seizures, problems with vision, vomiting and mental changes. The headache is classically worse in the morning and goes away with vomiting. Other symptoms may include difficulty walking, speaking or with sensations. As the disease progresses, unconsciousness may occur.



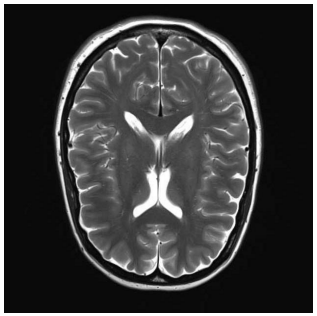
In the image above we see a brain metastasis in the right cerebral hemisphere from lung cancer, shown on an MRI scan

Dataset Description

The dataset contains 3000 images of Brain MRI images which will be used for brain tumor detection. It consists of two classes:

No – no tumor, encoded as 0

Yes – tumor – encoded as 1



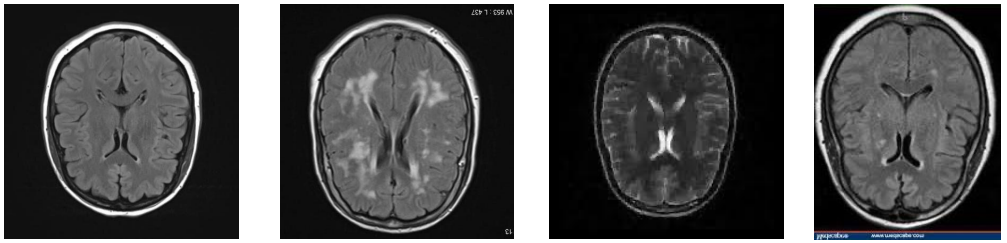
0 (no)

1 (yes)

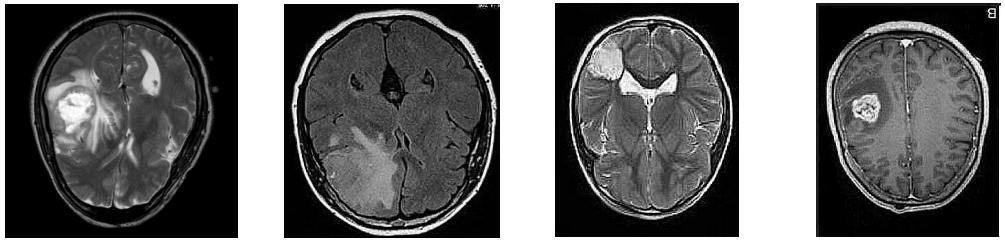
Dataset Description Cont.

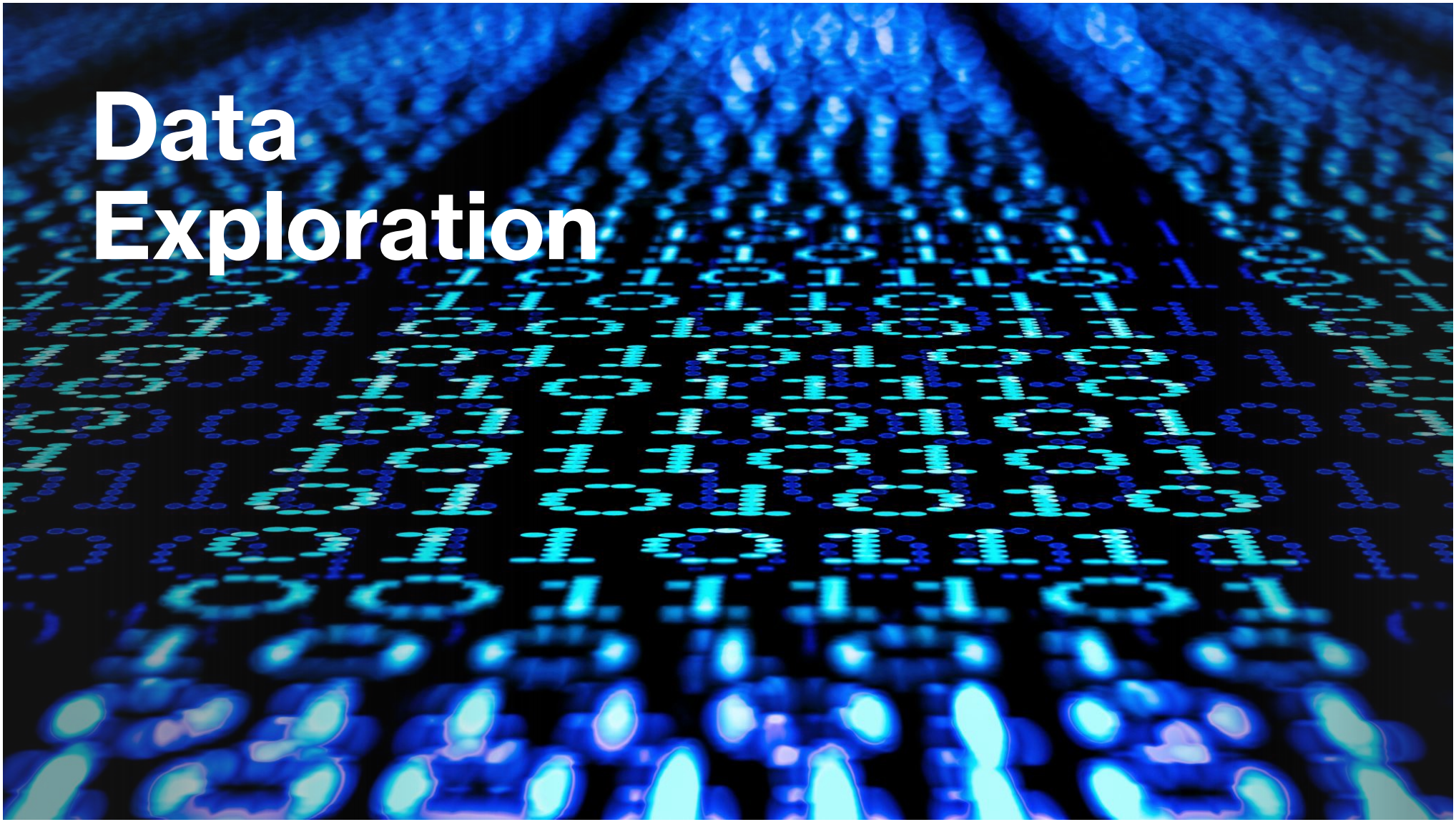
Folder	Description
No	The No folder contains 1500 brain MRI images that are non-tumorous
Yes	The Yes folder contains 1500 brain MRI images that are tumorous

Sample of Non-Tumorous MRI scans



Sample of Tumorous MRI scans



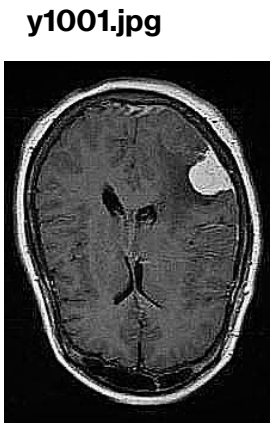

Data
Exploration

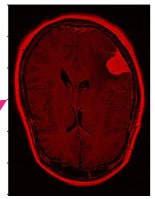
Identifying Image Types and Dimensions

Image Type: JPG

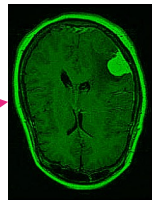
Dimensions: (Width, Height, 3)

Image Sample: “y1001.jpg”
- Dimensions: 303, 223, 3

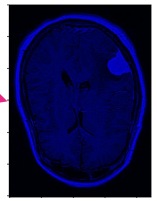




R - Channel



G - Channel



B - Channel

Image Class Comparison

Here, we are comparing the width and heights of the images for both ‘Yes’ and ‘No’ classes

	image_label	image_width	image_height
0	no26	282	230
1	no979	234	215
2	no598	213	236
3	no141	221	228
4	no715	252	200
...	...	...	...
1495	no941	243	207
1496	no853	340	339
1497	no884	225	225
1498	no1370	225	225
1499	no495	243	207

1500 rows × 3 columns

Non-Tumorous images information

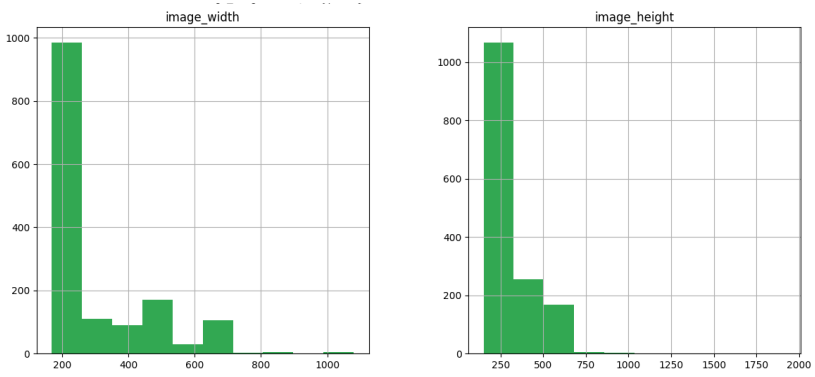
	image_label	image_width	image_height
0	y925	277	272
1	y424	336	264
2	y708	316	270
3	y115	620	620
4	y504	251	201
...	...	...	...
1495	y1379	349	425
1496	y1452	325	254
1497	y378	344	279
1498	y178	960	781
1499	y478	309	232

1500 rows × 3 columns

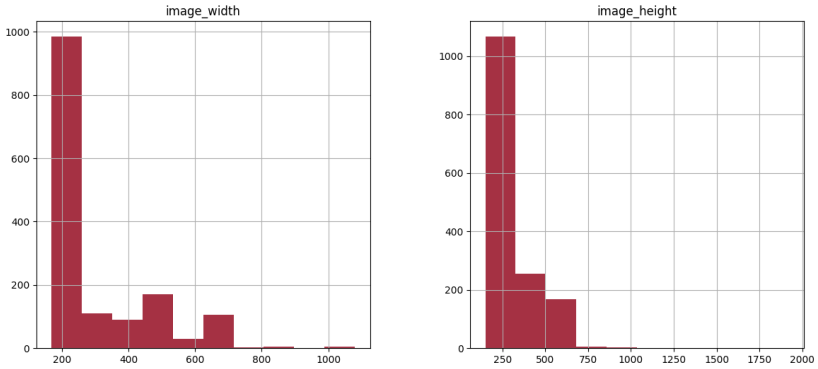
Tumorous images information

Image Width & Height for Both Classes

The graphs below visualize the differences between the height and width of both classes



Non-Tumorous images width, height distribution



Tumorous images width, height distribution

Class Comparison

Below is a summary of both Non-Tumorous and Tumorous classes

	image_width	image_height
count	1500.000000	1500.000000
mean	306.702667	299.980667
std	141.918226	148.211275
min	168.000000	150.000000
25%	225.000000	214.000000
50%	238.000000	227.000000
75%	400.000000	368.000000
max	1080.000000	1920.000000

Non-Tumorous images stats

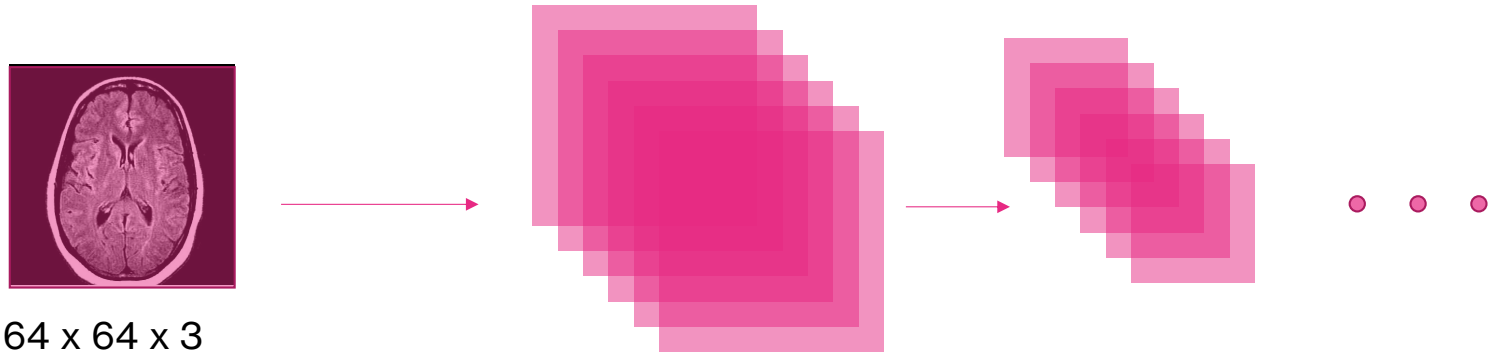
	image_width	image_height
count	1500.000000	1500.000000
mean	398.853333	350.455333
std	206.095229	193.916053
min	167.000000	175.000000
25%	294.000000	247.500000
50%	342.000000	283.000000
75%	380.000000	353.000000
max	1427.000000	1275.000000

Tumorous images stats

Feature Engineering – Image Resizing - Diagram

As shown in previous slides, the dimensions of the images in both classes vary significantly. Due to the architecture of deep learning models, we are required to resize the images to the same dimension. For our model, we will resize the images so that the shape of each image is 64 pixels.

Once this has been performed, the images are then stored in a NumPy array. The next slide will show the code used to perform the image resizing.



Feature Engineering – Image Resizing Code

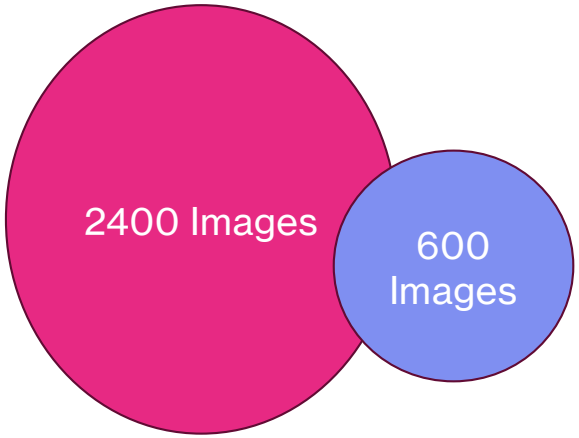
The code below is used to resize the images and store in a NumPy array

```
dataset_path = "/kaggle/input/brain-tumor-detection"
no_tumor_images = os.listdir(dataset_path + '/no/')
yes_tumor_images = os.listdir(dataset_path + '/yes/')

dataset=[]
label=[]

INPUT_SIZE=64

for i, image_name in enumerate(no_tumor_images):
    if image_name.split(".")[1] == 'jpg':
        image = cv2.imread(dataset_path+"/no/"+image_name)
        image = Image.fromarray(image, "RGB")
        image = image.resize((INPUT_SIZE, INPUT_SIZE))
        dataset.append(np.array(image))
        label.append(0)

for i, image_name in enumerate(yes_tumor_images):
    if image_name.split(".")[1] == 'jpg':
        image = cv2.imread(dataset_path+"/yes/"+image_name)
        image = Image.fromarray(image, "RGB")
        image = image.resize((INPUT_SIZE, INPUT_SIZE))
        dataset.append(np.array(image))
        label.append(1)
```

Feature Engineering – Dataset Splitting

In total we have 3,000 images, so we will split these images into two sets. One is a training set and the other is a testing set.

80% will be the training set (2400 images), and the remaining 20% the test set (600 images)




Machine
Learning
Analysis &
Findings

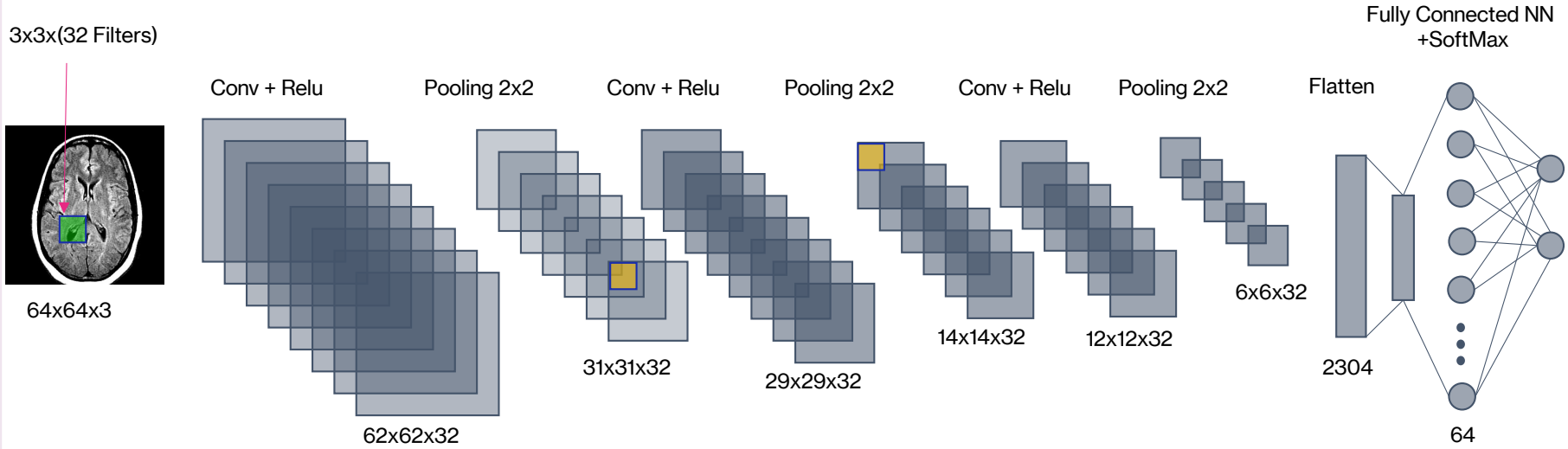
Model Approach

The following analysis will compare 2 different convolutional neural networks (CNN). These are:

- Categorical Cross Entropy with SoftMax function
- Binary Cross Entropy with Sigmoid function

The aim of these models is to classify the brain MRI scan images to distinguish between the tumorous images, and the non-tumorous images, helping to assist doctors with diagnosing brain tumors

CNN Categorical Cross Entropy & SoftMax Function - Architecture



CNN Categorical Cross Entropy & SoftMax Function – Model Code

```
# Model Building
model_1 = Sequential()

# first layer
model_1.add(Conv2D(filters= 32, kernel_size=(3,3), input_shape=(INPUT_SIZE, INPUT_SIZE, 3)))
model_1.add(Activation("relu"))
model_1.add(MaxPooling2D(pool_size=(2,2)))

# 1st hidden layer
model_1.add(Conv2D(filters= 32, kernel_size=(3,3), kernel_initializer="he_uniform"))
model_1.add(Activation("relu"))
model_1.add(MaxPooling2D(pool_size=(2,2)))

# 2nd hidden layer
model_1.add(Conv2D(filters= 64, kernel_size=(3,3), kernel_initializer="he_uniform"))
model_1.add(Activation("relu"))
model_1.add(MaxPooling2D(pool_size=(2,2)))

# Flatten Layer
model_1.add(Flatten())
model_1.add(Dense(64))
model_1.add(Activation("relu"))
model_1.add(Dropout(0.5))

#Categorical Cross Entropy = 2, softmax
model_1.add(Dense(2))
model_1.add(Activation("softmax"))
model_1.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

CNN Categorical Cross Entropy & SoftMax Function

CNN Categorical Cross Entropy architecture and total parameters

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
activation (Activation)	(None, 62, 62, 32)	0
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9,248
activation_1 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18,496
activation_2 (Activation)	(None, 12, 12, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 64)	147,520
activation_3 (Activation)	(None, 64)	0
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130
activation_4 (Activation)	(None, 2)	0

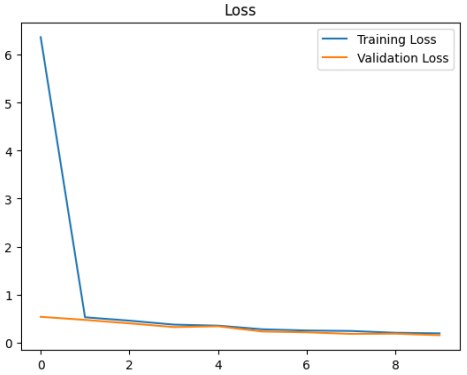
```
Total params: 176,290 (688.63 KB)
Trainable params: 176,290 (688.63 KB)
Non-trainable params: 0 (0.00 B)
```

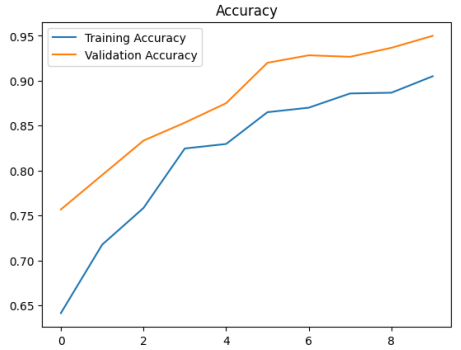
CNN Categorical Cross Entropy & SoftMax Function

```
model_1.fit(x_train, y_train, batch_size=32, verbose=True, epochs=10,
          validation_data=(x_test, y_test), shuffle=False)
```

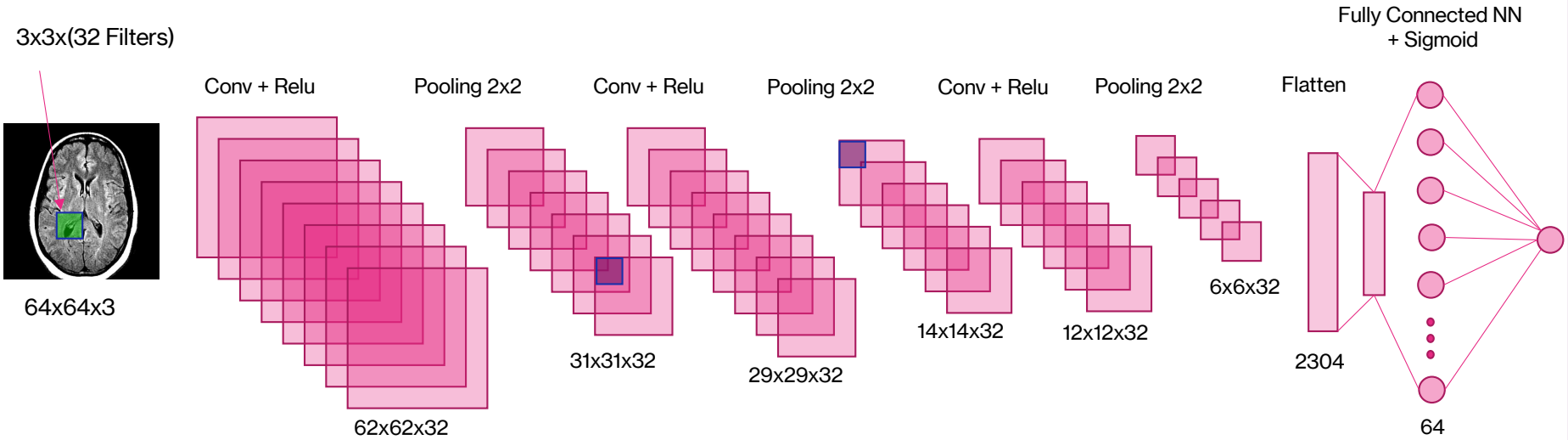
```
Epoch 1/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 9s 86ms/step - accuracy: 0.6018 - loss: 20.3337 - val_accuracy: 0.7567 - val_loss: 0.5339
Epoch 2/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 84ms/step - accuracy: 0.7043 - loss: 0.5353 - val_accuracy: 0.7950 - val_loss: 0.4696
Epoch 3/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 11s 88ms/step - accuracy: 0.7454 - loss: 0.4639 - val_accuracy: 0.8333 - val_loss: 0.3975
Epoch 4/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 10s 81ms/step - accuracy: 0.8227 - loss: 0.3848 - val_accuracy: 0.8533 - val_loss: 0.3194
Epoch 5/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - accuracy: 0.8354 - loss: 0.3433 - val_accuracy: 0.8750 - val_loss: 0.3364
Epoch 6/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - accuracy: 0.8731 - loss: 0.2794 - val_accuracy: 0.9200 - val_loss: 0.2279
Epoch 7/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - accuracy: 0.8673 - loss: 0.2515 - val_accuracy: 0.9283 - val_loss: 0.2109
Epoch 8/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 7s 90ms/step - accuracy: 0.8899 - loss: 0.2400 - val_accuracy: 0.9267 - val_loss: 0.1775
Epoch 9/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 81ms/step - accuracy: 0.8864 - loss: 0.1966 - val_accuracy: 0.9367 - val_loss: 0.1820
Epoch 10/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 11s 84ms/step - accuracy: 0.9011 - loss: 0.1958 - val_accuracy: 0.9500 - val_loss: 0.1504
```

Set	Accuracy	Loss
Training	90.11%	0.1958
Validation	95.00%	0.1504





CNN Binary Cross Entropy & Sigmoid Function - Architecture



CNN Binary Cross Entropy & Sigmoid Function – Model Code

```
# Model Building
model_2 = Sequential()

# first layer
model_2.add(Conv2D(filters= 32, kernel_size=(3,3), input_shape=(INPUT_SIZE, INPUT_SIZE, 3)))
model_2.add(Activation("relu"))
model_2.add(MaxPooling2D(pool_size=(2,2)))

# 1st hidden layer
model_2.add(Conv2D(filters= 32, kernel_size=(3,3), kernel_initializer="he_uniform"))
model_2.add(Activation("relu"))
model_2.add(MaxPooling2D(pool_size=(2,2)))

# 2nd hidden layer
model_2.add(Conv2D(filters= 64, kernel_size=(3,3), kernel_initializer="he_uniform"))
model_2.add(Activation("relu"))
model_2.add(MaxPooling2D(pool_size=(2,2)))

# Flatten Layer
model_2.add(Flatten())
model_2.add(Dense(64))
model_2.add(Activation("relu"))
model_2.add(Dropout(0.5))

# Binary Cross Entropy = 1, sigmoid
model_2.add(Dense(1))
model_2.add(Activation("sigmoid"))
model_2.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

CNN Binary Cross Entropy & Sigmoid Function

CNN Binary Cross Entropy architecture and total parameters

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 62, 62, 32)	896
activation_5 (Activation)	(None, 62, 62, 32)	0
max_pooling2d_3 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_4 (Conv2D)	(None, 29, 29, 32)	9,248
activation_6 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_5 (Conv2D)	(None, 12, 12, 64)	18,496
activation_7 (Activation)	(None, 12, 12, 64)	0
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_2 (Dense)	(None, 64)	147,520
activation_8 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65
activation_9 (Activation)	(None, 1)	0

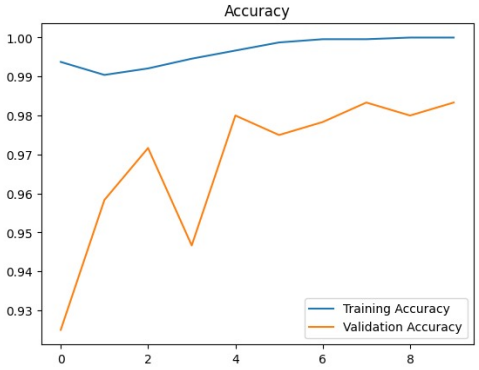
Total params: 528,677 (2.02 MB)
Trainable params: 176,225 (688.38 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 352,452 (1.34 MB)

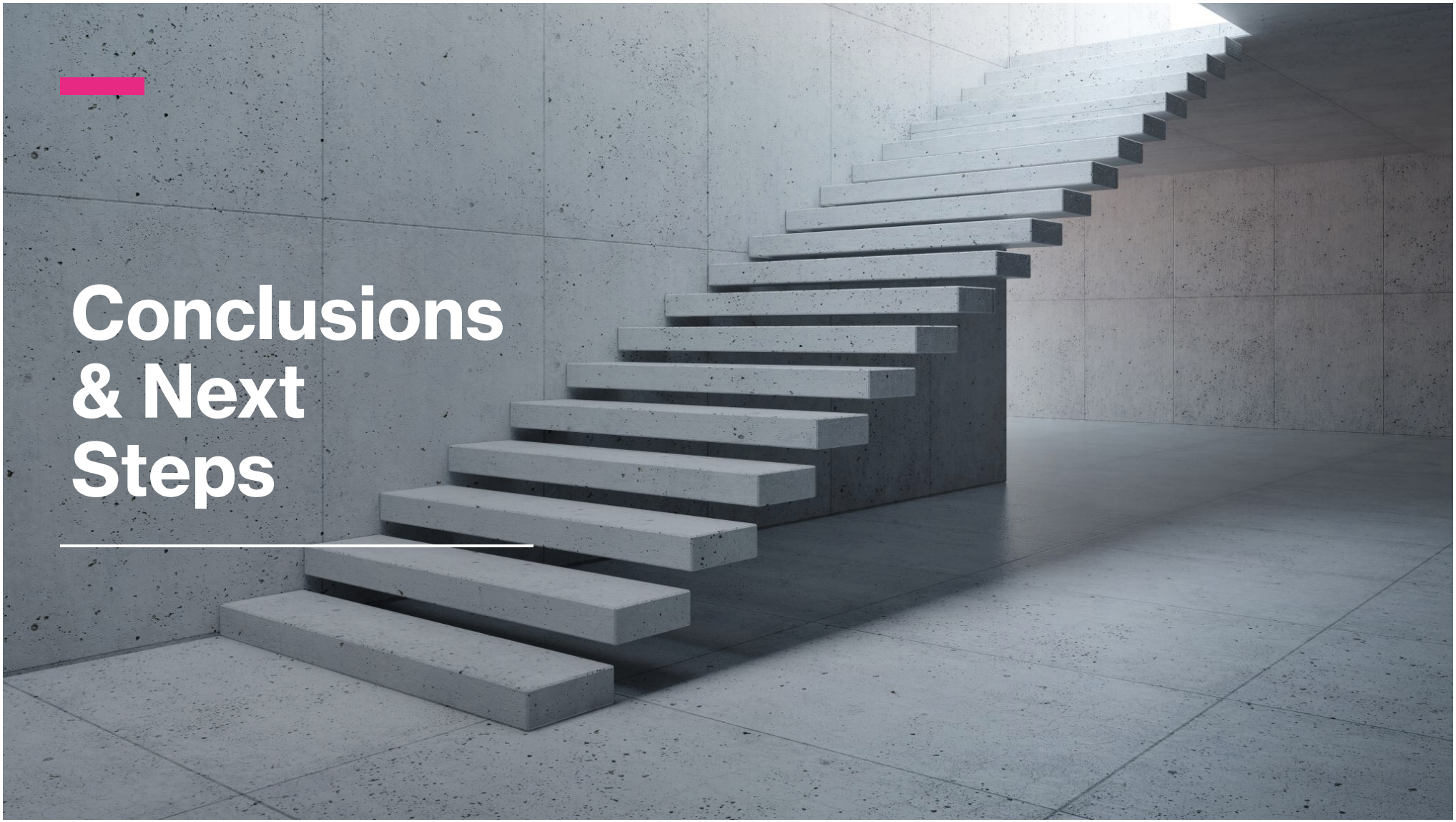
CNN Binary Cross Entropy & Sigmoid Function

```
model_2.fit(x_train, y_train, batch_size=32, verbose=True, epochs=10,
          validation_data=(x_test, y_test), shuffle=False)
```

```
Epoch 1/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 7s 91ms/step - accuracy: 0.9932 - loss: 0.0286 - val_accuracy: 0.9250 - val_loss: 0.1909
Epoch 2/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 81ms/step - accuracy: 0.9942 - loss: 0.0260 - val_accuracy: 0.9583 - val_loss: 0.0973
Epoch 3/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 10s 84ms/step - accuracy: 0.9951 - loss: 0.0203 - val_accuracy: 0.9717 - val_loss: 0.0904
Epoch 4/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 10s 84ms/step - accuracy: 0.9967 - loss: 0.0146 - val_accuracy: 0.9467 - val_loss: 0.1677
Epoch 5/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 7s 91ms/step - accuracy: 0.9935 - loss: 0.0174 - val_accuracy: 0.9800 - val_loss: 0.0689
Epoch 6/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 85ms/step - accuracy: 0.9972 - loss: 0.0122 - val_accuracy: 0.9750 - val_loss: 0.0979
Epoch 7/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - accuracy: 0.9990 - loss: 0.0025 - val_accuracy: 0.9783 - val_loss: 0.0642
Epoch 8/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 81ms/step - accuracy: 0.9988 - loss: 0.0035 - val_accuracy: 0.9833 - val_loss: 0.0762
Epoch 9/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 81ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accuracy: 0.9800 - val_loss: 0.0697
Epoch 10/10
75/75 ━━━━━━━━━━━━━━━━━━━━ 7s 89ms/step - accuracy: 1.0000 - loss: 9.1870e-04 - val_accuracy: 0.9833 - val_loss: 0.0724
```

Set	Accuracy	Loss
Training	100%	0.0011
Validation	98.00%	0.0697




Conclusions
& Next
Steps

Models Comparison

Model Strengths

Both models achieved high accuracy, with the CNN model which is based on Categorical Cross Entropy as loss function and SoftMax for final prediction layer achieving 90% on the training set and 95% on the validation set, and the Binary Cross Entropy and sigmoid function for final prediction layer achieving 100% on the training set and 98.0% on the validation set.

As we would be wanting to pick the most accurate model here, we would select the Binary Cross Entropy model as it had the higher overall accuracy.

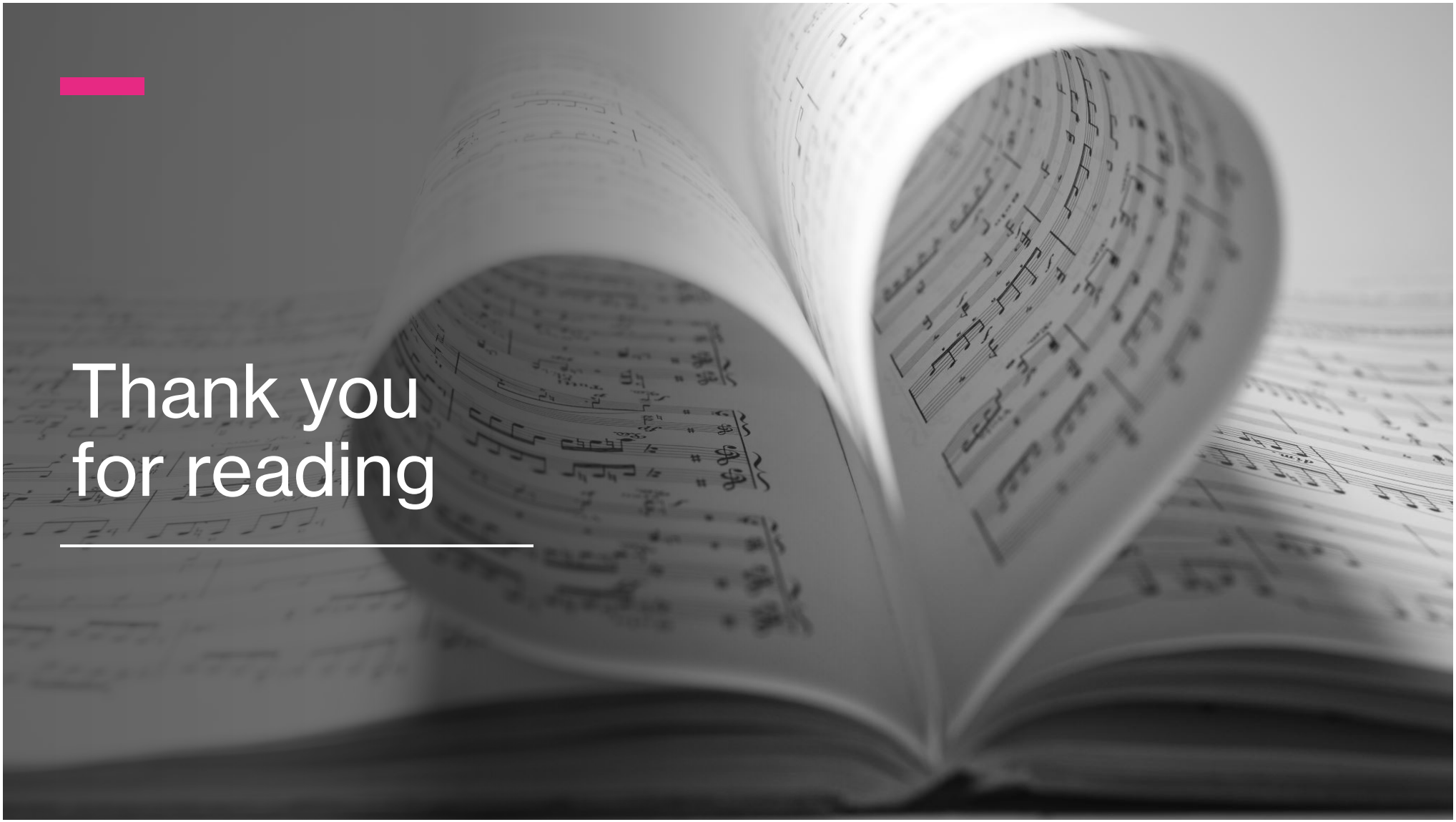
Model Flaws

One of main flaws in these models is the inability to detect the type of tumor, whether it is a benign or malignant tumor. In addition, they are unable to detect the location of the tumor, it is just a ‘Yes/No’ diagnosis.

Future Development

In terms of future development, I would consider the following:

- Adding classification of tumor type, e.g. benign, malignant
- Using object detection algorithms to detect the boundaries of the tumor, helping doctors locate the area of the brain where the tumor is.


Thank you
for reading