
BRAIN TUMOR DETECTION

Capstone Phase 5 Project

Name: Ansel Vallejo

Source:

> *Dataset: Brain Tumor Detection*
> *Author: Ahmed Hamada*
> *Website: Kaggle*
> *URL: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection*

Overview

In the medical field, Brain tumor is seen as a serious, abnormal growth of cells in or near the brain that can be either benign, which is non-cancerous or malignant, which is cancerous. There are many treatments an individual with such tumor can receive, such as radiation therapy, chemotherapy, therapeutic drug, etc. Brain tumor can be seen via Magnetic Resonance Imaging (MRI) scanned images. Human error is prone to occur in any industry, and because of such error in the medical field can cost a life. To detect and identify whether there is an absence of abnormal cell growth without human intervention would be to build a robust deep learning model to help the medical practitioner properly classify a tumor from no tumor using over 3000 MRI scan trained images and testing images to help better serve the model.

Algorithm Implemented: *Convolutional Neural Network* (Deep Learning)
Data type: Unstructured

Data Classification:

- **NO** (no tumor) - classified as 0
- **YES** (yes tumor) - classified as 1

Model Used: *brain_tumor_base_100_epochs_64_basics.h5*
Model Accuracy: approx. **99%**

Brain Tumor Detection v.1.0.0 *(Beta)*: *Work in Progress*

Import Libraries

In [54]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
%matplotlib inline
import tensorflow as tf
import seaborn as sns

from PIL import Image
import random
import pickle
import cv2
import os

# TENSORFLOW
from tensorflow.keras.utils import array_to_img, img_to_array, load_img
from tensorflow.keras.optimizers.schedules import ExponentialDecay
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.models import load_model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam

#SKLEARN
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
from sklearn.dummy import DummyClassifier

#KERAS
from keras.layers import Activation, Dropout, Flatten, Dense
```

```python
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from keras.utils import normalize
```

Number of GPUs Available

In [2]:
```python
#print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

In [25]:
```python
os.environ["CUDA_VISIBLE_DEVICES"]="-1"
```

Functions

(Tools)

In [26]:
```python
def model_acc_loss(test, loss):
    print('Model Accuracy (Test data)')
    print('________________________________________')
    print('')
    print('Model Accuracy:      ', test)
    print('Test Loss:           ', loss)
    print('________________________________________')
    print('')

    return
```

In [27]:
```python
def plot_training_results(results, model):
    # Extract loss and accuracy values from the training results
    train_loss = results.history['loss']
    train_acc = results.history['accuracy']
    val_loss = results.history['val_loss']
    val_acc = results.history['val_accuracy']

    # Create subplots for loss and accuracy plots
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

    # Plot loss values
    sns.lineplot(x=results.epoch, y=train_loss, ax=ax1, label='train_loss')
    sns.lineplot(x=results.epoch, y=val_loss, ax=ax1, label='val_loss')
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Loss')

    # Plot accuracy values
    sns.lineplot(x=results.epoch, y=train_acc, ax=ax2, label='train_accuracy')
    sns.lineplot(x=results.epoch, y=val_acc, ax=ax2, label='val_accuracy')
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Accuracy')

    # Display the plots
    plt.tight_layout()
    plt.show()

    # Updated function in 01_brain_tumor_classification_hyperparameter_tuning
    ###

    # Use the trained model to predict probabilities for the test data
    y_pred_prob = model.predict(X_val)

    # Convert probabilities to class labels based on a threshold
    threshold = 0.5
    y_pred = (y_pred_prob > threshold).astype(int)
    # print(y_pred)

    # Print the classification report
    print(classification_report(y_val, y_pred, zero_division=1))


    model.summary()
```

In [28]:
```python
def split_train_val_data(image, label, test_size, random_state):

    """
    Split the data into training and validation sets.

    Parameters:
    - image: Input data (images)
```

```
    - label: Target labels
    - test_size: Percentage of data to allocate for validation
    - random_state: Random seed for reproducibility

    Returns:
    - X_train: Training data (images)
    - X_val: Validation data (images)
    - y_train: Training labels
    - y_val: Validation labels

    """

    X_train, X_val, y_train, y_val = train_test_split(image, label, test_size=test_size, random_state=random_state)

    return X_train, X_val, y_train, y_val
```

Import Data

load_data *(Function)*

The function is iterating through the *directory*, and *categories* defined, then classifying the data by 'yes' **{1}** or 'no' **{0}**, depending on the data folder the images are extracted from. After categorizing the data, we proceed in converting the images into grayscale, and resizing according to spec. we then append the *data* list to combine both the resized array with its respective categorical number. After the data is prepared, the data is shuffled then separated into the *image* and *label* list, followed up by reshaping the *image* variable list. Then, the data is returned and included into the *image* and *label* variables outside of the function to then be referenced and follow up with saving into a pickle file format to then be referenced when needed.

In [29]:

```
data = []                    # Empty list to store images and labels

directory = "data/"          # Folder path
categories = ["no", "yes"]   # Folder
IMG_SIZE = 128               # Image size

# Function that loads the image data, categorizes images, resize, shuffles, and creates image and label lists.
def load_data():

    for classification in categories:

        # Defining file path and category numnber.
        path = os.path.join(directory, classification)
        cat_num = categories.index(classification)

        for img in os.listdir(path):
            try:

        # Iterating and changing the size, color, and adding both categories and images into the data list.
                img_array = cv2.imread(os.path.join(path, img))
                img_array = cv2.cvtColor(img_array, cv2.COLOR_BGR2RGB)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                data.append([new_array, cat_num])
            except Exception as e:
                pass

    random.shuffle(data)              # Shuffles data so that its not concatenated and sorted by category.


    image = []                        # List for image arrays.
    label = []                        # List for label arrays.


    # For loop that iterates over the data list, and separates the image from the labels after the above.
    for images, labels in data:
        image.append(images)
        label.append(labels)

    image = np.array(image)           # Convert the image list into n numpy array.
    label = np.array(label)           # Convert the label list into a numpy array.


    return image, label, img_array


image, label, img_array= load_data()  # Assign value to the image, label, and img_array variables from load_data().
```

Saving Data

Save Data *(.pkl file extension)*

This code saves two objects, image and label, into separate pickle files named "image.pickle" and "label.pickle", respectively. The pickle module is used to serialize the objects, converting them into a byte stream that can be stored in a file. Later, these objects can be loaded and deserialized using the pickle module to retrieve their original state.

In [5]:
```
# Creates and writes the '.pickle' file in write format, saves then closes file
pickle_out = open('image.pickle', 'wb')
pickle.dump(image, pickle_out)
pickle_out.close()

pickle_out = open('label.pickle', 'wb')
pickle.dump(label, pickle_out)
pickle_out.close()
```

In [6]:
```
# Opens and loads the '.pickle' file in readable format
pickle_in = open('image.pickle', 'rb')
image = pickle.load(pickle_in)

pickle_in = open('label.pickle', 'rb')
label = pickle.load(pickle_in)
```

TESTING pickle_in

This is a test code to see if the image variable arrays were properly saved in pickle file format.

And as you can see, the array is 3x3, meaning that the color channel is RGB.

In [8]:
```
image[1]
```

Out[8]:
```
array([[[38, 37, 33],
        [38, 37, 33],
        [38, 37, 33],
        ...,
        [43, 42, 38],
        [41, 40, 36],
        [ 9,  8,  4]],

       [[38, 37, 33],
        [38, 37, 33],
        [38, 37, 33],
        ...,
        [43, 42, 38],
        [40, 38, 35],
        [ 7,  6,  3]],

       [[36, 35, 31],
        [36, 35, 31],
        [36, 35, 31],
        ...,
        [41, 40, 36],
        [39, 39, 35],
        [ 8,  7,  3]],

       ...,

       [[40, 39, 35],
        [40, 39, 35],
        [40, 39, 35],
        ...,
        [42, 41, 37],
        [39, 39, 35],
        [ 8,  7,  3]],

       [[40, 39, 35],
        [40, 39, 35],
        [40, 39, 35],
        ...,
        [42, 41, 37],
        [40, 39, 35],
        [ 8,  7,  3]],

       [[40, 39, 35],
        [40, 39, 35],
        [40, 39, 35],
        ...,
        [42, 41, 37],
        [40, 38, 35],
        [ 8,  7,  3]]], dtype=uint8)
```

Plot Multiple Images by Category

plot_images_5_5 *(Function)*

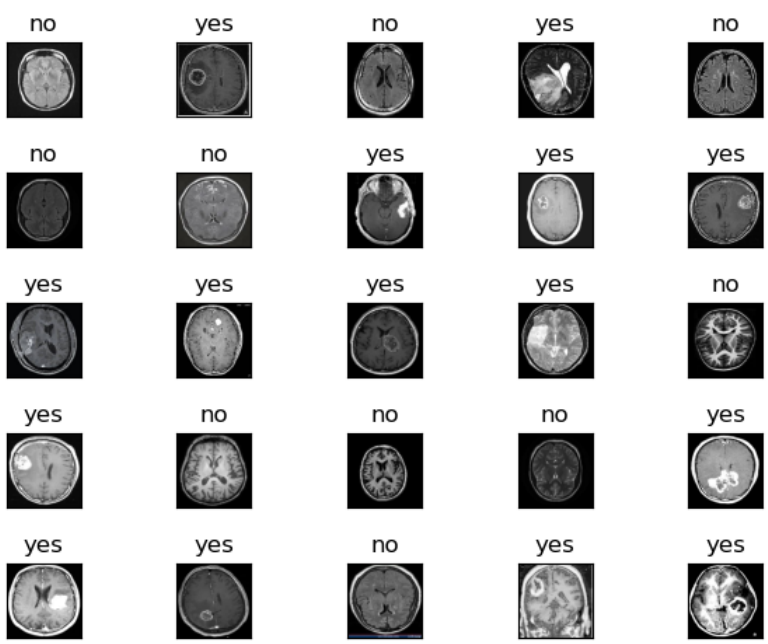
The function creates a 5x5 grid of subplots using the subplots() function. It then loops through each subplot and resizes the corresponding image using the resize() function from OpenCV. It then displays the image in the subplot using the imshow() function from matplotlib. The title of each subplot is set to the corresponding category label using the set_title() function from matplotlib. The x and y ticks are removed from each subplot using the set_xticks() and set_yticks() functions from matplotlib.

In [7]:

```
def plot_images_5_5():
    # Create a figure with 5 rows and 5 columns of subplots
    fig, axs = plt.subplots(5, 5)

    # Iterate over the rows
    for i in range(len(axs)):
        # Iterate over the columns
        for j in range(len(axs[i])):
            # Resize the image to a specified size
            new_array = cv2.resize(image[i*len(axs[i])+j], (IMG_SIZE, IMG_SIZE))

            # Display the image in the current subplot
            axs[i][j].imshow(new_array, cmap='gray')

            # Set the title of the subplot to the corresponding label/category
            axs[i][j].set_title(categories[label[i*len(axs[i])+j]])

            # Remove the x-axis ticks
            axs[i][j].set_xticks([])

            # Remove the y-axis ticks
            axs[i][j].set_yticks([])

    # Adjust the spacing between subplots to prevent overlapping
    plt.tight_layout()

    # Display the plot
    plt.show()
```

In [8]:

```
plot_images_5_5()
```



---

plot_images_2_3 *(Function)*

The function loops through the first 6 images in the image list and creates a 2x3 grid of subplots using the subplot() function. It then displays the corresponding image in each subplot using the imshow() function from matplotlib. The title of each subplot is set to the corresponding category label using the title() function from matplotlib. The x and y ticks are removed from each subplot using the xticks() and yticks() functions from matplotlib.

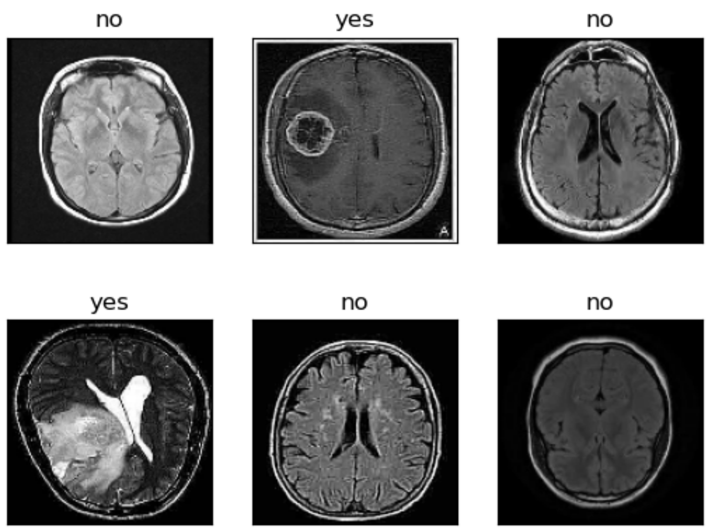
In [9]:

```
def plot_images_2_3():
    # Iterate over the range 0-5 (6 iterations)
    for i in range(6):
        # Create a subplot grid of 2 rows and 3 columns and select the i+1-th subplot
        plt.subplot(2, 3, i+1)

        # Display the image in the current subplot
        plt.imshow(image[i])

        # Set the title of the subplot to the corresponding label/category
```

```
        plt.title(categories[label[i]])

        # Remove the x-axis ticks
        plt.xticks([])

        # Remove the y-axis ticks
        plt.yticks([])

    # Display the plot
    plt.show()
```

In [10]:

```
plot_images_2_3()
```



DISPLAY IMAGE BY IMAGE SIZE

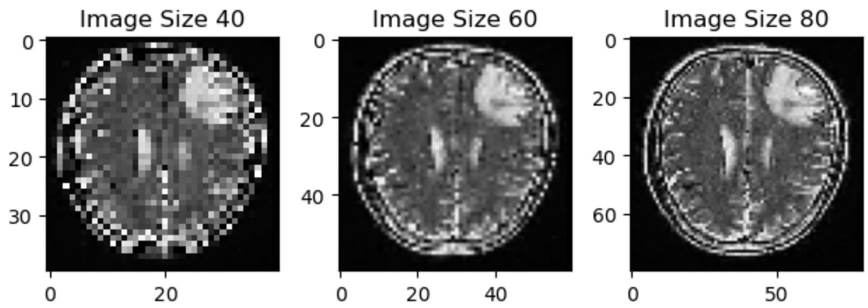
The code plots runs a for-loop displaying 3 images side by side comparing the different image size. As it iterates through the loop, the IMG_SIZE will increase by 20, thus plotting images in sizes of 40, 60, 80.

A 3x3 version variant is made for visualizing better image classification. The code follows similar concepts but its tailored for its size, making it slighly unique.

(1 x 3)

In [11]:

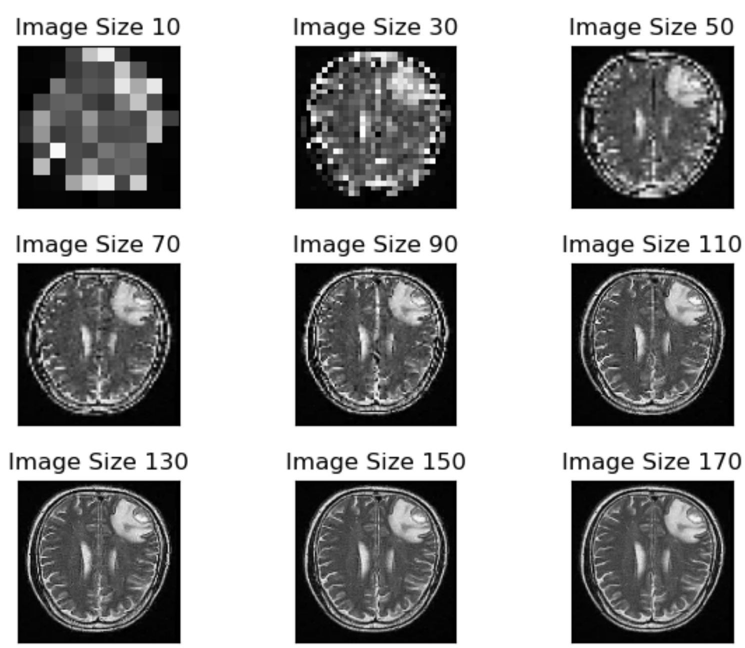
```
# Figure with 1 row and 3 columns for subplots
fig, axs = plt.subplots(1, 3)

# Directory path where the images are stored
directory = "data/"

# Categories or folders containing the images
categories = ["no", "yes"]

# Initial image size
IMG_SIZE = 40

# Loop through the subplots
for i in range(len(axs)):
    # Resize the image to the specified size
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))

    # Display the resized image in the current subplot as grayscale
    axs[i].imshow(new_array, cmap='gray')

    # Set the title for the current subplot
    axs[i].set_title(f'Image Size {IMG_SIZE}')

    # Increase the image size by 20 for the next iteration
    IMG_SIZE += 20

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```



(3 x 3)

In [12]:

```python
# Create a 3x3 grid of subplots
fig, axs = plt.subplots(3, 3)

# Initial image size
IMG_SIZE = 10

# Loop through the rows of subplots
for i in range(len(axs)):
    # Loop through the columns of subplots
    for j in range(len(axs)):
        # Resize the image to the specified size
        new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))

        # Display the resized image in the current subplot as grayscale
        axs[i, j].imshow(new_array, cmap='gray')

        # Set the title for the current subplot
        axs[i, j].set_title(f'Image Size {IMG_SIZE}')

        # Increase the image size by 20 for the next iteration
        IMG_SIZE += 20

# Remove x-axis and y-axis ticks for all subplots
for ax in axs.flat:
    ax.set(xticks=[], yticks=[])

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```



In [ ]:

Train Test Split

A Train Test Split is conducted by splitting the image array (containing the image) and the label array (containing the respective labels for the images) into training and validation data, with a 20% split, and random state of 42 to ensure the code will be ran the same way for consistency.

In [30]:
```
#  Train Test Split function
#  (can be found above in the "Functions (Tools) dropdown")

X_train, X_val, y_train, y_val = split_train_val_data(image, # Input
                                                       label, # Target
                                                       0.20,  # Percentage data allocation to validation
                                                       42)    # fixed random seed for reuseability
```

Train / Validation Size Check

To make sure the model will run properly, a sanity check is necessary to ensure the train and validation sizes are the same for the model to run. If there is a imbalance of image size, then the model wont run, resulting in an error message.

In [31]:
```
print(X_train.shape)
print(y_train.shape)
print(X_val.shape)
print(y_val.shape)
```

```
(2400, 128, 128, 3)
(2400,)
(600, 128, 128, 3)
(600,)
```

In [32]:
```
# Redifining Image size due to changing variable in the plot example
IMG_SIZE = 128
```

Normalization

As a common form of preprocessing images, we need to standardize the image to prevent any feature from receiving the most attention during the learning process. It also helps with model performance and stability.

In [33]:
```
X_train = normalize(X_train, axis =1)
X_val = normalize(X_val, axis =1)
```

Convolutionary Neural Network *(CNN)* Model

00 DUMMY MODEL

For the dummy model, we use the Brain Tumor images that is categorized as YES and NO, split it into training and validation sets, then normalizing the data. We then create a DummyClassifier model using the most_frequent strategy, which predicts the most frequent class in the training data. After training the model, we make predictions on the test set and calculate the accuracy using the accuracy_score metric.

In [19]:
```
# Create a dummy classifier model
dummy = DummyClassifier(strategy='most_frequent')

# Train the model
dummy.fit(X_train, y_train)

# Make predictions on the test set
y_pred = dummy.predict(X_val)

# Calculate accuracy
accuracy = accuracy_score(y_val, predictions)

# Print the accuracy
print("Accuracy:", accuracy)
```

```
Accuracy: 0.49833333333333335
```

In [20]:
```
print (classification_report(y_val, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       299
           1       0.00      0.00      0.00       301

    accuracy                           0.50       600
   macro avg       0.25      0.50      0.33       600
weighted avg       0.25      0.50      0.33       600
```

```
C:\Users\msavg\anaconda3\envs\learn-env\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
```

```
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\msavg\anaconda3\envs\learn-env\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\msavg\anaconda3\envs\learn-env\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
```

The DummyClassifier resulted in a 0.49833333333333335 on predicting whether or not the MRI image contains a tumor. It is evident the dummy model did not perform well, but can be used as baseline for model comparison.

00 BASE MODEL

In [30]:

```
# Instantiate
# base_model = Sequential()
```

In [36]:

```
def create_base_model():
    """
    Create the base model.

    Returns:
    - model: The base model with the defined architecture
    """

    # Create a Sequential model
    base_model = Sequential()

    # Add a 2D convolutional layer with 32 filters, each of size 3x3,
    # and input shape of (128, 128, 3) representing the image dimensions and color channels
    base_model.add(Conv2D(32, (3,3), input_shape=(128, 128, 3)))

    # Apply the ReLU activation function to introduce non-linearity
    base_model.add(Activation('relu'))

    # Add a max pooling layer with pool size of 2x2
    base_model.add(MaxPooling2D(pool_size=(2,2)))

    # Add another 2D convolutional layer with 32 filters, each of size 3x3,
    # and use the 'he_uniform' kernel initializer
    base_model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))

    # Apply the ReLU activation function
    base_model.add(Activation('relu'))

    # Add another max pooling layer with pool size of 2x2
    base_model.add(MaxPooling2D(pool_size=(2,2)))

    # Add another 2D convolutional layer with 64 filters, each of size 3x3,
    # and use the 'he_uniform' kernel initializer
    base_model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))

    # Apply the ReLU activation function
    base_model.add(Activation('relu'))

    # Add another max pooling layer with pool size of 2x2
    base_model.add(MaxPooling2D(pool_size=(2,2)))

    # Flatten the output of the previous layer to a 1D array
    base_model.add(Flatten())

    # Add a fully connected (dense) layer with 64 neurons
    base_model.add(Dense(64))

    # Apply the ReLU activation function
    base_model.add(Activation('relu'))

    # Apply dropout with a rate of 0.5 to prevent overfitting
    base_model.add(Dropout(0.5))

    # Add the output layer with a single neuron, using the sigmoid activation function
    base_model.add(Dense(1))
    base_model.add(Activation('sigmoid'))

    # Compile the model with binary cross-entropy loss function,
    # Adam optimizer, and accuracy as the metric to monitor
    base_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return base_model
```

EXECUTE

100 Epoch

In [34]:
```
# Exexcute the model
base_results_100 = base_model.fit(X_train, y_train, batch_size = 16,
          verbose=1,
          epochs=100,
          validation_data= (X_val, y_val),
          shuffle = False)

# Save model
base_model.save('brain_tumor_base_100_epochs_16.h5')
```

---

50 Epoch

In [40]:
```
# Exexcute the model
base_model = create_base_model()
base_results_50 = base_model.fit(X_train, y_train, batch_size = 16,
                                       verbose=1,
                                       epochs=50,
                                       validation_data=(X_val,y_val),
                                       shuffle=False)

# Save model
base_model.save('brain_tumor_base_50_epochs_16_batch.h5')
```

```
Epoch 1/50
150/150 [==============================] - 28s 181ms/step - loss: 0.4937 - accuracy: 0.7671 - val_loss: 0.3530 - val_accuracy: 0.8567
Epoch 2/50
150/150 [==============================] - 27s 182ms/step - loss: 0.3100 - accuracy: 0.8717 - val_loss: 0.2298 - val_accuracy: 0.9083
Epoch 3/50
150/150 [==============================] - 27s 183ms/step - loss: 0.2110 - accuracy: 0.9158 - val_loss: 0.1798 - val_accuracy: 0.9333
Epoch 4/50
150/150 [==============================] - 28s 185ms/step - loss: 0.1321 - accuracy: 0.9488 - val_loss: 0.1329 - val_accuracy: 0.9533
Epoch 5/50
150/150 [==============================] - 28s 185ms/step - loss: 0.0841 - accuracy: 0.9729 - val_loss: 0.0937 - val_accuracy: 0.9650
Epoch 6/50
150/150 [==============================] - 27s 180ms/step - loss: 0.0579 - accuracy: 0.9800 - val_loss: 0.1058 - val_accuracy: 0.9733
Epoch 7/50
150/150 [==============================] - 28s 185ms/step - loss: 0.0427 - accuracy: 0.9858 - val_loss: 0.1100 - val_accuracy: 0.9633
Epoch 8/50
150/150 [==============================] - 28s 188ms/step - loss: 0.0411 - accuracy: 0.9837 - val_loss: 0.1198 - val_accuracy: 0.9733
Epoch 9/50
150/150 [==============================] - 28s 184ms/step - loss: 0.0259 - accuracy: 0.9925 - val_loss: 0.1090 - val_accuracy: 0.9717
Epoch 10/50
150/150 [==============================] - 28s 186ms/step - loss: 0.0188 - accuracy: 0.9933 - val_loss: 0.0964 - val_accuracy: 0.9817
Epoch 11/50
150/150 [==============================] - 28s 187ms/step - loss: 0.0208 - accuracy: 0.9921 - val_loss: 0.1261 - val_accuracy: 0.9650
Epoch 12/50
150/150 [==============================] - 28s 186ms/step - loss: 0.0276 - accuracy: 0.9896 - val_loss: 0.1520 - val_accuracy: 0.9650
Epoch 13/50
150/150 [==============================] - 29s 191ms/step - loss: 0.0218 - accuracy: 0.9925 - val_loss: 0.1618 - val_accuracy: 0.9717
Epoch 14/50
150/150 [==============================] - 28s 190ms/step - loss: 0.0116 - accuracy: 0.9967 - val_loss: 0.1229 - val_accuracy: 0.9767
Epoch 15/50
150/150 [==============================] - 29s 194ms/step - loss: 0.0255 - accuracy: 0.9912 - val_loss: 0.1071 - val_accuracy: 0.9783
Epoch 16/50
150/150 [==============================] - 27s 180ms/step - loss: 0.0135 - accuracy: 0.9958 - val_loss: 0.1207 - val_accuracy: 0.9767
Epoch 17/50
150/150 [==============================] - 26s 174ms/step - loss: 0.0075 - accuracy: 0.9971 - val_loss: 0.1405 - val_accuracy: 0.9750
Epoch 18/50
150/150 [==============================] - 26s 175ms/step - loss: 0.0049 - accuracy: 0.9971 - val_loss: 0.1791 - val_accuracy: 0.9733
Epoch 19/50
150/150 [==============================] - 26s 176ms/step - loss: 0.0067 - accuracy: 0.9979 - val_loss: 0.1627 - val_accuracy: 0.9733
Epoch 20/50
150/150 [==============================] - 26s 177ms/step - loss: 0.0114 - accuracy: 0.9946 - val_loss: 0.1041 - val_accuracy: 0.9750
Epoch 21/50
150/150 [==============================] - 26s 175ms/step - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.1287 - val_accuracy: 0.9733
Epoch 22/50
150/150 [==============================] - 26s 175ms/step - loss: 0.0018 - accuracy: 0.9996 - val_loss: 0.1555 - val_accuracy: 0.9767
Epoch 23/50
150/150 [==============================] - 31s 206ms/step - loss: 0.0129 - accuracy: 0.9954 - val_loss: 0.1251 - val_accuracy: 0.9800
Epoch 24/50
150/150 [==============================] - 29s 196ms/step - loss: 0.0277 - accuracy: 0.9912 - val_loss: 0.1873 - val_accuracy: 0.9733
Epoch 25/50
150/150 [==============================] - 26s 173ms/step - loss: 0.0124 - accuracy: 0.9946 - val_loss: 0.1442 - val_accuracy: 0.9733
Epoch 26/50
150/150 [==============================] - 27s 179ms/step - loss: 0.0111 - accuracy: 0.9950 - val_loss: 0.1814 - val_accuracy: 0.9700
Epoch 27/50
150/150 [==============================] - 28s 184ms/step - loss: 0.0046 - accuracy: 0.9983 - val_loss: 0.1641 - val_accuracy: 0.9800
Epoch 28/50
150/150 [==============================] - 26s 175ms/step - loss: 0.0031 - accuracy: 0.9987 - val_loss: 0.1531 - val_accuracy: 0.9717
Epoch 29/50
150/150 [==============================] - 27s 177ms/step - loss: 0.0024 - accuracy: 0.9992 - val_loss: 0.1901 - val_accuracy: 0.9767
Epoch 30/50
150/150 [==============================] - 27s 179ms/step - loss: 0.0042 - accuracy: 0.9983 - val_loss: 0.1781 - val_accuracy: 0.9700
```

```
Epoch 31/50
150/150 [==============================] - 27s 178ms/step - loss: 0.0074 - accuracy: 0.9971 - val_loss: 0.1735 - val_accuracy: 0.9717
Epoch 32/50
150/150 [==============================] - 28s 184ms/step - loss: 0.0110 - accuracy: 0.9962 - val_loss: 0.1857 - val_accuracy: 0.9733
Epoch 33/50
150/150 [==============================] - 27s 181ms/step - loss: 0.0106 - accuracy: 0.9954 - val_loss: 0.1751 - val_accuracy: 0.9733
Epoch 34/50
150/150 [==============================] - 25s 170ms/step - loss: 0.0206 - accuracy: 0.9925 - val_loss: 0.1148 - val_accuracy: 0.9750
Epoch 35/50
150/150 [==============================] - 28s 187ms/step - loss: 0.0152 - accuracy: 0.9958 - val_loss: 0.1551 - val_accuracy: 0.9650
Epoch 36/50
150/150 [==============================] - 26s 173ms/step - loss: 0.0189 - accuracy: 0.9929 - val_loss: 0.1947 - val_accuracy: 0.9717
Epoch 37/50
150/150 [==============================] - 29s 194ms/step - loss: 0.0101 - accuracy: 0.9962 - val_loss: 0.2085 - val_accuracy: 0.9750
Epoch 38/50
150/150 [==============================] - 29s 194ms/step - loss: 0.0136 - accuracy: 0.9950 - val_loss: 0.2180 - val_accuracy: 0.9767
Epoch 39/50
150/150 [==============================] - 29s 192ms/step - loss: 0.0115 - accuracy: 0.9962 - val_loss: 0.3023 - val_accuracy: 0.9733
Epoch 40/50
150/150 [==============================] - 26s 177ms/step - loss: 0.0094 - accuracy: 0.9971 - val_loss: 0.1759 - val_accuracy: 0.9733
Epoch 41/50
150/150 [==============================] - 26s 170ms/step - loss: 0.0115 - accuracy: 0.9958 - val_loss: 0.2395 - val_accuracy: 0.9700
Epoch 42/50
150/150 [==============================] - 25s 170ms/step - loss: 0.0045 - accuracy: 0.9996 - val_loss: 0.2156 - val_accuracy: 0.9750
Epoch 43/50
150/150 [==============================] - 26s 171ms/step - loss: 0.0022 - accuracy: 0.9992 - val_loss: 0.2253 - val_accuracy: 0.9750
Epoch 44/50
150/150 [==============================] - 26s 170ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 0.2549 - val_accuracy: 0.9767
Epoch 45/50
150/150 [==============================] - 26s 171ms/step - loss: 5.9367e-04 - accuracy: 1.0000 - val_loss: 0.2690 - val_accuracy: 0.97
33
Epoch 46/50
150/150 [==============================] - 25s 170ms/step - loss: 0.0024 - accuracy: 0.9992 - val_loss: 0.2653 - val_accuracy: 0.9733
Epoch 47/50
150/150 [==============================] - 25s 169ms/step - loss: 0.0037 - accuracy: 0.9983 - val_loss: 0.2086 - val_accuracy: 0.9750
Epoch 48/50
150/150 [==============================] - 25s 169ms/step - loss: 0.0021 - accuracy: 0.9996 - val_loss: 0.1953 - val_accuracy: 0.9750
Epoch 49/50
150/150 [==============================] - 26s 171ms/step - loss: 0.0074 - accuracy: 0.9971 - val_loss: 0.1158 - val_accuracy: 0.9817
Epoch 50/50
150/150 [==============================] - 26s 173ms/step - loss: 0.0050 - accuracy: 0.9987 - val_loss: 0.1530 - val_accuracy: 0.9833
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [40], in <cell line: 10>()
      3 base_results_50 = base_model.fit(X_train, y_train, batch_size = 16,
      4                                  verbose=1,
      5                                  epochs=50,
      6                                  validation_data=(X_val,y_val),
      7                                  shuffle=False)
      9 # Save model
---> 10 model.save('brain_tumor_base_50_epochs_16_batch.h5')

NameError: name 'model' is not defined
```

20 Epoch

In [47]:

```python
# Exexcute the model
base_model = create_base_model()
base_results_20 = base_model.fit(X_train, y_train, batch_size = 16,
          verbose=1,
          epochs=20,
          validation_data= (X_val, y_val),
          shuffle = False)

# Save model
base_model.save('brain_tumor_base_20_epochs_16_batch.h5')
```

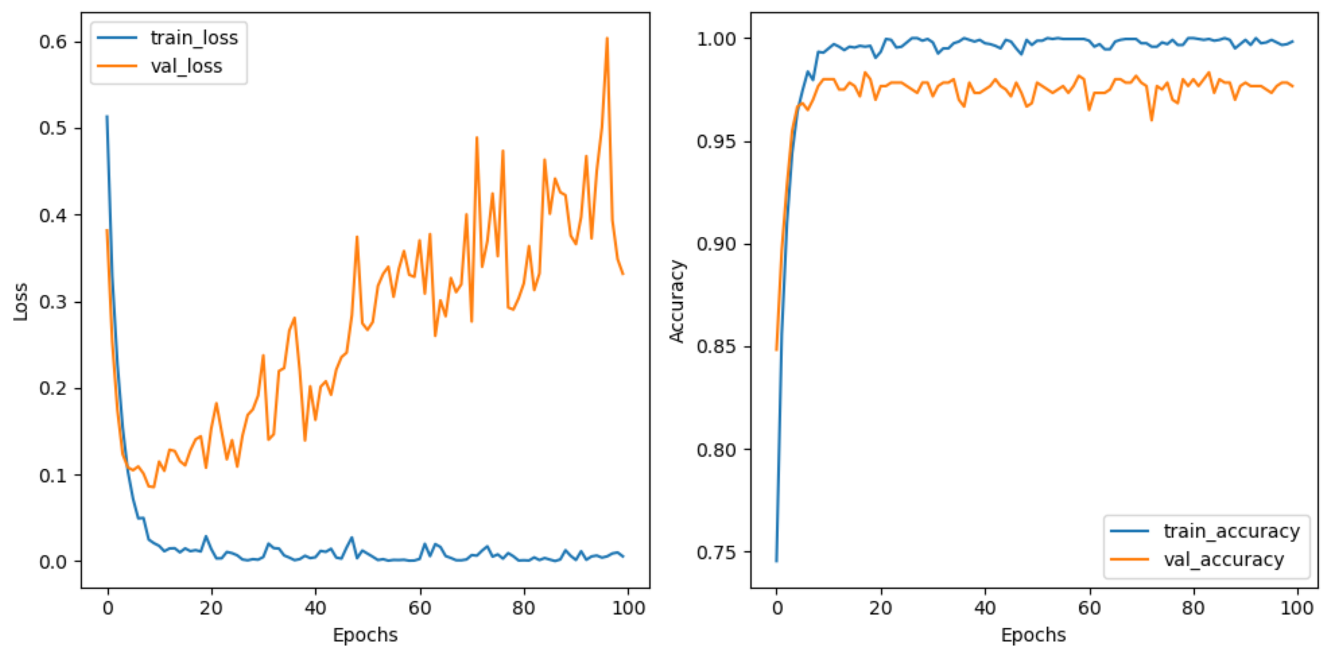
```
Epoch 1/20
150/150 [==============================] - 25s 166ms/step - loss: 0.4847 - accuracy: 0.7679 - val_loss: 0.3741 - val_accuracy: 0.8533
Epoch 2/20
150/150 [==============================] - 25s 169ms/step - loss: 0.3150 - accuracy: 0.8704 - val_loss: 0.2372 - val_accuracy: 0.8967
Epoch 3/20
150/150 [==============================] - 25s 169ms/step - loss: 0.2142 - accuracy: 0.9133 - val_loss: 0.1506 - val_accuracy: 0.9367
Epoch 4/20
150/150 [==============================] - 25s 170ms/step - loss: 0.1351 - accuracy: 0.9504 - val_loss: 0.1149 - val_accuracy: 0.9583
Epoch 5/20
150/150 [==============================] - 26s 171ms/step - loss: 0.0799 - accuracy: 0.9758 - val_loss: 0.1274 - val_accuracy: 0.9467
Epoch 6/20
150/150 [==============================] - 26s 171ms/step - loss: 0.0609 - accuracy: 0.9808 - val_loss: 0.0927 - val_accuracy: 0.9700
Epoch 7/20
150/150 [==============================] - 26s 170ms/step - loss: 0.0331 - accuracy: 0.9917 - val_loss: 0.0911 - val_accuracy: 0.9733
Epoch 8/20
150/150 [==============================] - 26s 171ms/step - loss: 0.0335 - accuracy: 0.9887 - val_loss: 0.0872 - val_accuracy: 0.9783
Epoch 9/20
150/150 [==============================] - 26s 173ms/step - loss: 0.0282 - accuracy: 0.9912 - val_loss: 0.0646 - val_accuracy: 0.9833
Epoch 10/20
150/150 [==============================] - 26s 172ms/step - loss: 0.0189 - accuracy: 0.9958 - val_loss: 0.0810 - val_accuracy: 0.9783
Epoch 11/20
150/150 [==============================] - 26s 172ms/step - loss: 0.0152 - accuracy: 0.9950 - val_loss: 0.0946 - val_accuracy: 0.9800
Epoch 12/20
150/150 [==============================] - 26s 172ms/step - loss: 0.0094 - accuracy: 0.9971 - val_loss: 0.0782 - val_accuracy: 0.9833
```

```
Epoch 13/20
150/150 [==============================] - 26s 175ms/step - loss: 0.0185 - accuracy: 0.9937 - val_loss: 0.1071 - val_accuracy: 0.9783
Epoch 14/20
150/150 [==============================] - 26s 172ms/step - loss: 0.0029 - accuracy: 0.9996 - val_loss: 0.1133 - val_accuracy: 0.9783
Epoch 15/20
150/150 [==============================] - 27s 177ms/step - loss: 0.0042 - accuracy: 0.9987 - val_loss: 0.1322 - val_accuracy: 0.9733
Epoch 16/20
150/150 [==============================] - 26s 172ms/step - loss: 0.0211 - accuracy: 0.9912 - val_loss: 0.0974 - val_accuracy: 0.9817
Epoch 17/20
150/150 [==============================] - 26s 173ms/step - loss: 0.0199 - accuracy: 0.9925 - val_loss: 0.1192 - val_accuracy: 0.9750
Epoch 18/20
150/150 [==============================] - 26s 174ms/step - loss: 0.0162 - accuracy: 0.9950 - val_loss: 0.1401 - val_accuracy: 0.9750
Epoch 19/20
150/150 [==============================] - 26s 173ms/step - loss: 0.0050 - accuracy: 0.9979 - val_loss: 0.1202 - val_accuracy: 0.9817
Epoch 20/20
150/150 [==============================] - 26s 174ms/step - loss: 0.0047 - accuracy: 0.9983 - val_loss: 0.1291 - val_accuracy: 0.9800
```

PLOT RESULTS

In [43]:

```
#Plot Results - 100 Epoch | 16 batch
plot_training_results(base_results_100, base_model)
```

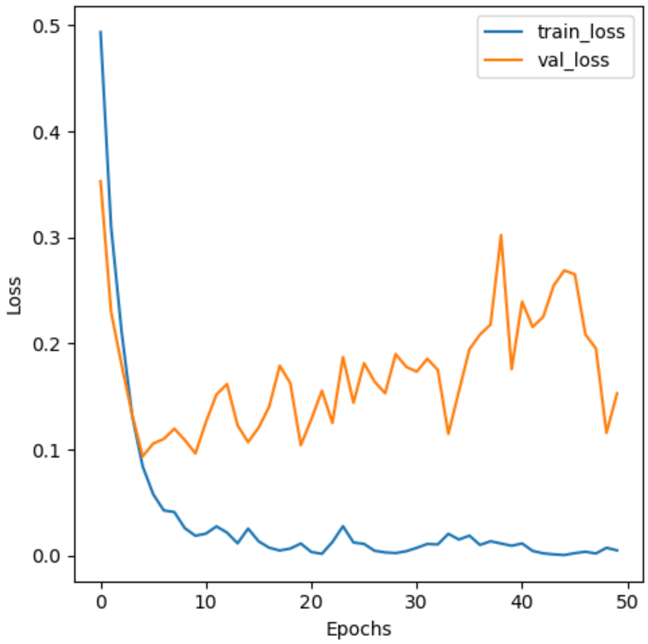


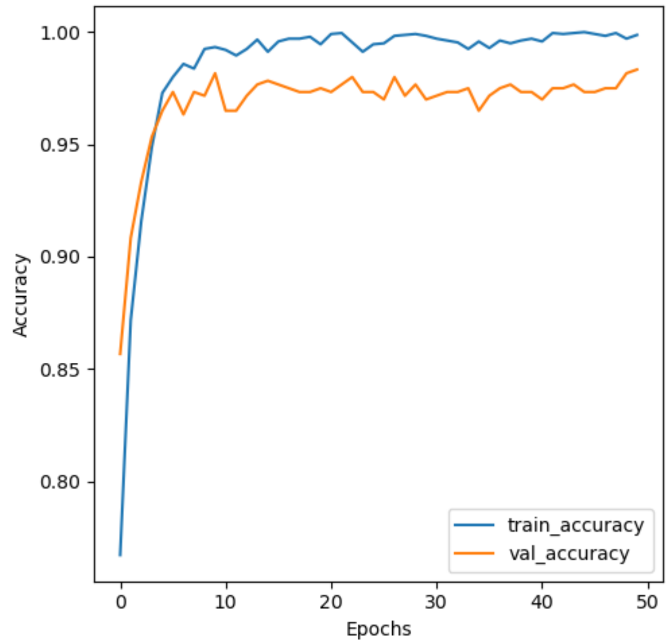

```
19/19 [==============================] - 2s 75ms/step
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       310
           1       0.99      0.97      0.98       290

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_21 (Conv2D)          (None, 126, 126, 32)      896

 activation_35 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_21 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 61, 61, 32)        9248

 activation_36 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_22 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 28, 28, 64)        18496

 activation_37 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_23 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_7 (Flatten)         (None, 12544)             0
```

```
 dense_14 (Dense)            (None, 64)                802880

 activation_38 (Activation)  (None, 64)                0

 dropout_7 (Dropout)         (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

 activation_39 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

In [44]:

```
#Plot Results - 50 Epoch | 16 batch
plot_training_results(base_results_50, base_model)
```



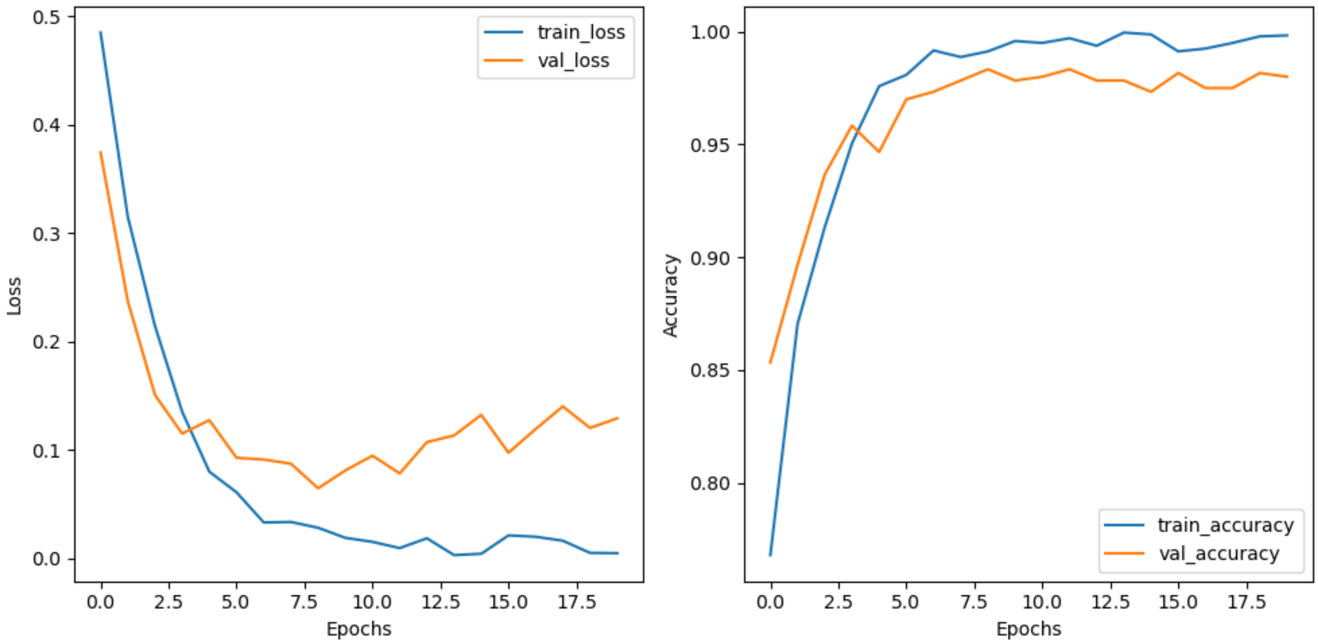



```
19/19 [==============================] - 1s 76ms/step
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       310
           1       0.99      0.97      0.98       290

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_21 (Conv2D)          (None, 126, 126, 32)      896

 activation_35 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_21 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 61, 61, 32)        9248

 activation_36 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_22 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 28, 28, 64)        18496

 activation_37 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_23 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_7 (Flatten)         (None, 12544)             0

 dense_14 (Dense)            (None, 64)                802880

 activation_38 (Activation)  (None, 64)                0
```

```
 dropout_7 (Dropout)         (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

 activation_39 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

In [48]:

```python
#Plot Results - 20 Epoch | 16 batch
plot_training_results(base_results_20, base_model)
```




```
19/19 [==============================] - 1s 72ms/step
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       310
           1       0.98      0.98      0.98       290

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_24 (Conv2D)          (None, 126, 126, 32)      896

 activation_40 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_24 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_25 (Conv2D)          (None, 61, 61, 32)        9248

 activation_41 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_25 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_26 (Conv2D)          (None, 28, 28, 64)        18496

 activation_42 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_26 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_8 (Flatten)         (None, 12544)             0

 dense_16 (Dense)            (None, 64)                802880

 activation_43 (Activation)  (None, 64)                0

 dropout_8 (Dropout)         (None, 64)                0

 dense_17 (Dense)            (None, 1)                 65
```

```
 activation_44 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

Classification Report

In [49]:

```
# Use the trained model to predict probabilities for the test data
y_pred_prob = base_model.predict(X_val)

# Convert probabilities to class labels based on a threshold
threshold = 0.5
y_pred = (y_pred_prob > threshold).astype(int)
# print(y_pred)

# Print the classification report
print(classification_report(y_val, y_pred, zero_division=1))
```

```
19/19 [==============================] - 1s 72ms/step
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       310
           1       0.98      0.98      0.98       290

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600
```

Model Summary

In [50]:

```
base_model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_24 (Conv2D)          (None, 126, 126, 32)      896       
                                                                 
 activation_40 (Activation)  (None, 126, 126, 32)      0         
                                                                 
 max_pooling2d_24 (MaxPoolin  (None, 63, 63, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_25 (Conv2D)          (None, 61, 61, 32)        9248      
                                                                 
 activation_41 (Activation)  (None, 61, 61, 32)        0         
                                                                 
 max_pooling2d_25 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_26 (Conv2D)          (None, 28, 28, 64)        18496     
                                                                 
 activation_42 (Activation)  (None, 28, 28, 64)        0         
                                                                 
 max_pooling2d_26 (MaxPoolin  (None, 14, 14, 64)       0         
 g2D)                                                            
                                                                 
 flatten_8 (Flatten)         (None, 12544)             0         
                                                                 
 dense_16 (Dense)            (None, 64)                802880    
                                                                 
 activation_43 (Activation)  (None, 64)                0         
                                                                 
 dropout_8 (Dropout)         (None, 64)                0         
                                                                 
 dense_17 (Dense)            (None, 1)                 65        
                                                                 
 activation_44 (Activation)  (None, 1)                 0         
                                                                 
=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

EVALUATION

100 epochs

According to the train loss vs the val loss, the data seems ideally going well at around 7 epochs, then converges. But before I consider the possibility of overfitting, batch size will be the first parameter to tweak as a low batch size renders less memory but takes more time, leading to better generalization.

When considering the Classification Report, it seems like the model might be running perfect at 99% across the board. To account for the loss variance and volatility, regularization is an option to consider, augmenting the data, or hyperparameter tuning.
50 epochs
Running the model at 50 epochs is an amplified version of 100 epochs. Its clearly evident the model is converging as the val losses and val accuracy going into unfavorable direction at a given point. (val loss > train loss) (val accuracy < train accuracy) 20 epochs
At 20 epochs, I can see the model is doing fairly well as the lines are smoothened out. It seems as if the validation data is not performing as well, which will require some adjustments to the model.

Improvements ***(To be made)***

Code

- **When writing the code for the plot, classification report, and model summary, its best to combine all codes into one function so that after the model is done running, the information is properly displayed for that point in time the model ran.**
- **Making a function for the model will be considered to avoid redundency.**

Model

- **Improve model by changing the parameters**
- **Consider the possibility of overfitting**
- ***TBD after the above was considered***

01 HYPERPARAMETER TUNING

As for improvements to the model and functionality, I decided to create a cell for tuning parameters. Its made easy to adjust batch size, epoch, learning rate, and augmentation in one simple cell. This helps with consistency and easy-to-use tuning.

For this portion of tuning, different epoch levels will be evaluated to determine what epoch should be considered moving forward so the tuning parameters wont be much of use, but will be referenced upon determination.

Tuning Paramenters

In [18]:

```
##############(Image Size)###############
IMG_SIZE = 128

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


##############(Batch Size)###############
batch_size = 32

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


################(Epoch)##################
epoch = 30

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


###########(Exponential Decay)###########
# Learning rate schedule parameters
initial_lr = 0.001
decay_steps = 1000
decay_rate = 0.96
#########################################


############(ImageGenerator)############
# Parameters used in ImageGenerator
rotation_range=20
width_shift_range=0.2
height_shift_range=0.2
shear_range=0.2
zoom_range=0.2
horizontal_flip=True
fill_mode='nearest'
#########################################
```

In [ ]:

```

```

In [ ]:

Ver 1 - ADJUST EPOCH

Instantiate

In [60]:
```
#Instantiate model
model1 = Sequential()
```

Compile

In [65]:
```
def create_model1():
    """
    Create Model 1

    Returns:
    - model: Model 1 with the defined architecture
    """

    # Create a Sequential model
    model1 = Sequential()

    # Add a 2D convolutional layer with 32 filters, each of size 3x3,
    # and input shape of (IMG_SIZE, IMG_SIZE, 3) representing the image dimensions and color channels
    model1.add(Conv2D(32, (3,3), input_shape=(IMG_SIZE, IMG_SIZE, 3)))

    # Apply the ReLU activation function
    model1.add(Activation('relu'))

    # Add a max pooling layer with pool size of 2x2
    model1.add(MaxPooling2D(pool_size=(2,2)))

    # Add another 2D convolutional layer with 32 filters, each of size 3x3,
    # and use the 'he_uniform' kernel initializer
    model1.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))

    # Apply the ReLU activation function
    model1.add(Activation('relu'))

    # Add another max pooling layer with pool size of 2x2
    model1.add(MaxPooling2D(pool_size=(2,2)))

    # Add another 2D convolutional layer with 64 filters, each of size 3x3,
    # and use the 'he_uniform' kernel initializer
    model1.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))

    # Apply the ReLU activation function
    model1.add(Activation('relu'))

    # Add another max pooling layer with pool size of 2x2
    model1.add(MaxPooling2D(pool_size=(2,2)))

    # Flatten the output of the previous layer to a 1D array
    model1.add(Flatten())

    # Add a fully connected (dense) layer with 64 neurons
    model1.add(Dense(64))

    # Apply the ReLU activation function
    model1.add(Activation('relu'))

    # Apply dropout with a rate of 0.5 to prevent overfitting
    model1.add(Dropout(0.5))

    # Add the output layer with a single neuron, using the sigmoid activation function
    model1.add(Dense(1))
    model1.add(Activation('sigmoid'))

    # Compile the model with binary cross-entropy loss function,
    # Adam optimizer, and accuracy as the metric to monitor
    model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model1


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

EXECUTE

100 Epoch *at 32 batch*

In [62]:

```python
# Exexcute the model
results100 = model1.fit(X_train, y_train, batch_size = 32,
          verbose=1,
          epochs=100,
          validation_data= (X_val, y_val),
          shuffle = False)

# Save model
model1.save('model/brain_tumor_ver1_100_epochs_32.h5')
```

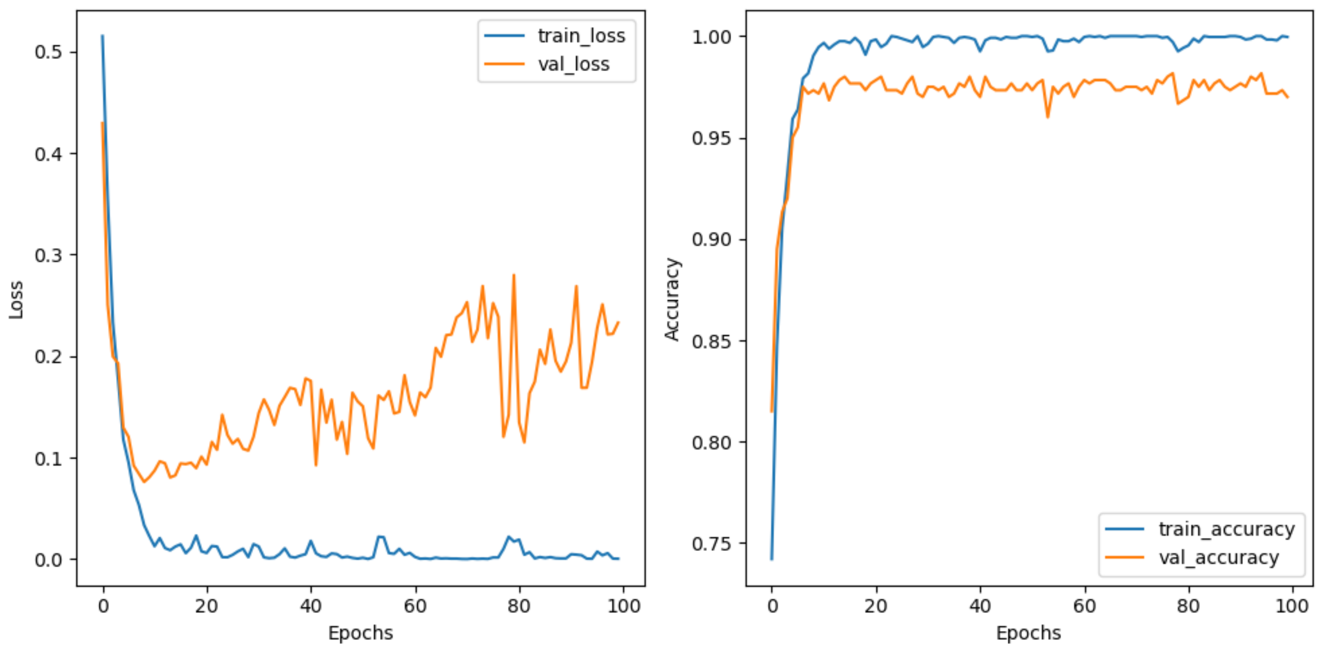
```
Epoch 1/100
75/75 [==============================] - 24s 318ms/step - loss: 0.5150 - accuracy: 0.7421 - val_loss: 0.4294 - val_accuracy: 0.8150
Epoch 2/100
75/75 [==============================] - 24s 321ms/step - loss: 0.3557 - accuracy: 0.8471 - val_loss: 0.2506 - val_accuracy: 0.8950
Epoch 3/100
75/75 [==============================] - 25s 327ms/step - loss: 0.2333 - accuracy: 0.9054 - val_loss: 0.1996 - val_accuracy: 0.9133
Epoch 4/100
75/75 [==============================] - 24s 318ms/step - loss: 0.1769 - accuracy: 0.9317 - val_loss: 0.1928 - val_accuracy: 0.9200
Epoch 5/100
75/75 [==============================] - 24s 321ms/step - loss: 0.1178 - accuracy: 0.9592 - val_loss: 0.1296 - val_accuracy: 0.9500
Epoch 6/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0955 - accuracy: 0.9638 - val_loss: 0.1205 - val_accuracy: 0.9550
Epoch 7/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0678 - accuracy: 0.9792 - val_loss: 0.0921 - val_accuracy: 0.9750
Epoch 8/100
75/75 [==============================] - 25s 328ms/step - loss: 0.0534 - accuracy: 0.9817 - val_loss: 0.0840 - val_accuracy: 0.9717
Epoch 9/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0335 - accuracy: 0.9904 - val_loss: 0.0760 - val_accuracy: 0.9733
Epoch 10/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0227 - accuracy: 0.9946 - val_loss: 0.0808 - val_accuracy: 0.9717
Epoch 11/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0127 - accuracy: 0.9967 - val_loss: 0.0871 - val_accuracy: 0.9767
Epoch 12/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0208 - accuracy: 0.9937 - val_loss: 0.0963 - val_accuracy: 0.9683
Epoch 13/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0110 - accuracy: 0.9958 - val_loss: 0.0944 - val_accuracy: 0.9750
Epoch 14/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0088 - accuracy: 0.9975 - val_loss: 0.0804 - val_accuracy: 0.9783
Epoch 15/100
75/75 [==============================] - 25s 332ms/step - loss: 0.0124 - accuracy: 0.9975 - val_loss: 0.0826 - val_accuracy: 0.9800
Epoch 16/100
75/75 [==============================] - 25s 328ms/step - loss: 0.0146 - accuracy: 0.9967 - val_loss: 0.0942 - val_accuracy: 0.9767
Epoch 17/100
75/75 [==============================] - 25s 336ms/step - loss: 0.0059 - accuracy: 0.9992 - val_loss: 0.0936 - val_accuracy: 0.9767
Epoch 18/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0113 - accuracy: 0.9967 - val_loss: 0.0950 - val_accuracy: 0.9767
Epoch 19/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0232 - accuracy: 0.9908 - val_loss: 0.0896 - val_accuracy: 0.9733
Epoch 20/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0078 - accuracy: 0.9975 - val_loss: 0.1009 - val_accuracy: 0.9767
Epoch 21/100
75/75 [==============================] - 25s 330ms/step - loss: 0.0061 - accuracy: 0.9983 - val_loss: 0.0931 - val_accuracy: 0.9783
Epoch 22/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0130 - accuracy: 0.9946 - val_loss: 0.1154 - val_accuracy: 0.9800
Epoch 23/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0124 - accuracy: 0.9962 - val_loss: 0.1077 - val_accuracy: 0.9733
Epoch 24/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.1422 - val_accuracy: 0.9733
Epoch 25/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0019 - accuracy: 0.9996 - val_loss: 0.1223 - val_accuracy: 0.9733
Epoch 26/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.1137 - val_accuracy: 0.9717
Epoch 27/100
75/75 [==============================] - 24s 325ms/step - loss: 0.0077 - accuracy: 0.9979 - val_loss: 0.1184 - val_accuracy: 0.9767
Epoch 28/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0102 - accuracy: 0.9971 - val_loss: 0.1086 - val_accuracy: 0.9800
Epoch 29/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.1069 - val_accuracy: 0.9717
Epoch 30/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0148 - accuracy: 0.9946 - val_loss: 0.1204 - val_accuracy: 0.9700
Epoch 31/100
75/75 [==============================] - 25s 328ms/step - loss: 0.0125 - accuracy: 0.9962 - val_loss: 0.1438 - val_accuracy: 0.9750
Epoch 32/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0018 - accuracy: 0.9996 - val_loss: 0.1573 - val_accuracy: 0.9750
Epoch 33/100
75/75 [==============================] - 24s 323ms/step - loss: 9.4407e-04 - accuracy: 1.0000 - val_loss: 0.1470 - val_accuracy: 0.9733
Epoch 34/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0014 - accuracy: 0.9996 - val_loss: 0.1321 - val_accuracy: 0.9750
Epoch 35/100
75/75 [==============================] - 24s 326ms/step - loss: 0.0049 - accuracy: 0.9992 - val_loss: 0.1511 - val_accuracy: 0.9700
Epoch 36/100
75/75 [==============================] - 24s 320ms/step - loss: 0.0107 - accuracy: 0.9967 - val_loss: 0.1597 - val_accuracy: 0.9717
Epoch 37/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0025 - accuracy: 0.9992 - val_loss: 0.1687 - val_accuracy: 0.9767
Epoch 38/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.1674 - val_accuracy: 0.9750
Epoch 39/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.1518 - val_accuracy: 0.9800
Epoch 40/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0049 - accuracy: 0.9983 - val_loss: 0.1780 - val_accuracy: 0.9733
```

```
Epoch 41/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0180 - accuracy: 0.9925 - val_loss: 0.1755 - val_accuracy: 0.9700
Epoch 42/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0058 - accuracy: 0.9979 - val_loss: 0.0925 - val_accuracy: 0.9800
Epoch 43/100
75/75 [==============================] - 24s 323ms/step - loss: 0.0029 - accuracy: 0.9992 - val_loss: 0.1669 - val_accuracy: 0.9750
Epoch 44/100
75/75 [==============================] - 24s 326ms/step - loss: 0.0022 - accuracy: 0.9992 - val_loss: 0.1344 - val_accuracy: 0.9733
Epoch 45/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0057 - accuracy: 0.9983 - val_loss: 0.1570 - val_accuracy: 0.9733
Epoch 46/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0050 - accuracy: 0.9996 - val_loss: 0.1177 - val_accuracy: 0.9733
Epoch 47/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0016 - accuracy: 0.9992 - val_loss: 0.1352 - val_accuracy: 0.9767
Epoch 48/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0026 - accuracy: 0.9992 - val_loss: 0.1037 - val_accuracy: 0.9733
Epoch 49/100
75/75 [==============================] - 24s 319ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.1639 - val_accuracy: 0.9733
Epoch 50/100
75/75 [==============================] - 24s 322ms/step - loss: 5.2397e-04 - accuracy: 1.0000 - val_loss: 0.1553 - val_accuracy: 0.9767
Epoch 51/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0014 - accuracy: 0.9996 - val_loss: 0.1505 - val_accuracy: 0.9733
Epoch 52/100
75/75 [==============================] - 24s 323ms/step - loss: 2.1042e-04 - accuracy: 1.0000 - val_loss: 0.1191 - val_accuracy: 0.9767
Epoch 53/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0019 - accuracy: 0.9987 - val_loss: 0.1090 - val_accuracy: 0.9783
Epoch 54/100
75/75 [==============================] - 25s 331ms/step - loss: 0.0221 - accuracy: 0.9925 - val_loss: 0.1611 - val_accuracy: 0.9600
Epoch 55/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0216 - accuracy: 0.9929 - val_loss: 0.1567 - val_accuracy: 0.9750
Epoch 56/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0060 - accuracy: 0.9983 - val_loss: 0.1653 - val_accuracy: 0.9717
Epoch 57/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0053 - accuracy: 0.9975 - val_loss: 0.1434 - val_accuracy: 0.9750
Epoch 58/100
75/75 [==============================] - 24s 319ms/step - loss: 0.0102 - accuracy: 0.9975 - val_loss: 0.1451 - val_accuracy: 0.9767
Epoch 59/100
75/75 [==============================] - 24s 318ms/step - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.1811 - val_accuracy: 0.9700
Epoch 60/100
75/75 [==============================] - 24s 318ms/step - loss: 0.0062 - accuracy: 0.9971 - val_loss: 0.1548 - val_accuracy: 0.9750
Epoch 61/100
75/75 [==============================] - 24s 319ms/step - loss: 0.0023 - accuracy: 0.9996 - val_loss: 0.1416 - val_accuracy: 0.9783
Epoch 62/100
75/75 [==============================] - 24s 322ms/step - loss: 4.0992e-04 - accuracy: 1.0000 - val_loss: 0.1641 - val_accuracy: 0.9767
Epoch 63/100
75/75 [==============================] - 24s 321ms/step - loss: 6.7663e-04 - accuracy: 0.9996 - val_loss: 0.1594 - val_accuracy: 0.9783
Epoch 64/100
75/75 [==============================] - 24s 321ms/step - loss: 2.0698e-04 - accuracy: 1.0000 - val_loss: 0.1689 - val_accuracy: 0.9783
Epoch 65/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0015 - accuracy: 0.9992 - val_loss: 0.2080 - val_accuracy: 0.9783
Epoch 66/100
75/75 [==============================] - 24s 317ms/step - loss: 6.4265e-04 - accuracy: 1.0000 - val_loss: 0.1993 - val_accuracy: 0.9767
Epoch 67/100
75/75 [==============================] - 24s 322ms/step - loss: 8.2165e-04 - accuracy: 1.0000 - val_loss: 0.2206 - val_accuracy: 0.9733
Epoch 68/100
75/75 [==============================] - 24s 323ms/step - loss: 5.3763e-04 - accuracy: 1.0000 - val_loss: 0.2213 - val_accuracy: 0.9733
Epoch 69/100
75/75 [==============================] - 24s 317ms/step - loss: 5.3355e-04 - accuracy: 1.0000 - val_loss: 0.2381 - val_accuracy: 0.9750
Epoch 70/100
75/75 [==============================] - 24s 319ms/step - loss: 2.1817e-04 - accuracy: 1.0000 - val_loss: 0.2423 - val_accuracy: 0.9750
Epoch 71/100
75/75 [==============================] - 24s 321ms/step - loss: 1.2639e-04 - accuracy: 1.0000 - val_loss: 0.2531 - val_accuracy: 0.9750
Epoch 72/100
75/75 [==============================] - 24s 325ms/step - loss: 6.4561e-04 - accuracy: 0.9996 - val_loss: 0.2139 - val_accuracy: 0.9733
Epoch 73/100
75/75 [==============================] - 24s 321ms/step - loss: 2.7721e-04 - accuracy: 1.0000 - val_loss: 0.2261 - val_accuracy: 0.9750
Epoch 74/100
75/75 [==============================] - 24s 323ms/step - loss: 5.2768e-04 - accuracy: 1.0000 - val_loss: 0.2689 - val_accuracy: 0.9717
Epoch 75/100
75/75 [==============================] - 25s 331ms/step - loss: 2.7979e-04 - accuracy: 1.0000 - val_loss: 0.2176 - val_accuracy: 0.9783
Epoch 76/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0017 - accuracy: 0.9992 - val_loss: 0.2521 - val_accuracy: 0.9767
Epoch 77/100
75/75 [==============================] - 24s 322ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.2387 - val_accuracy: 0.9800
Epoch 78/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0102 - accuracy: 0.9971 - val_loss: 0.1205 - val_accuracy: 0.9817
Epoch 79/100
75/75 [==============================] - 24s 320ms/step - loss: 0.0221 - accuracy: 0.9925 - val_loss: 0.1422 - val_accuracy: 0.9667
Epoch 80/100
75/75 [==============================] - 24s 320ms/step - loss: 0.0173 - accuracy: 0.9942 - val_loss: 0.2798 - val_accuracy: 0.9683
Epoch 81/100
75/75 [==============================] - 24s 317ms/step - loss: 0.0193 - accuracy: 0.9954 - val_loss: 0.1347 - val_accuracy: 0.9700
Epoch 82/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0044 - accuracy: 0.9987 - val_loss: 0.1150 - val_accuracy: 0.9783
Epoch 83/100
75/75 [==============================] - 24s 321ms/step - loss: 0.0069 - accuracy: 0.9971 - val_loss: 0.1635 - val_accuracy: 0.9750
Epoch 84/100
75/75 [==============================] - 24s 322ms/step - loss: 7.1606e-04 - accuracy: 1.0000 - val_loss: 0.1748 - val_accuracy: 0.9783
Epoch 85/100
75/75 [==============================] - 25s 328ms/step - loss: 0.0021 - accuracy: 0.9996 - val_loss: 0.2063 - val_accuracy: 0.9733
Epoch 86/100
75/75 [==============================] - 24s 320ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 0.1923 - val_accuracy: 0.9767
Epoch 87/100
```

```
75/75 [==============================] - 24s 325ms/step - loss: 0.0020 - accuracy: 0.9996 - val_loss: 0.2261 - val_accuracy: 0.9783
Epoch 88/100
75/75 [==============================] - 24s 321ms/step - loss: 9.7520e-04 - accuracy: 0.9996 - val_loss: 0.1954 - val_accuracy: 0.9750
Epoch 89/100
75/75 [==============================] - 24s 322ms/step - loss: 7.4380e-04 - accuracy: 1.0000 - val_loss: 0.1846 - val_accuracy: 0.9733
Epoch 90/100
75/75 [==============================] - 24s 325ms/step - loss: 7.4980e-04 - accuracy: 1.0000 - val_loss: 0.1947 - val_accuracy: 0.9750
Epoch 91/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0049 - accuracy: 0.9996 - val_loss: 0.2131 - val_accuracy: 0.9767
Epoch 92/100
75/75 [==============================] - 24s 326ms/step - loss: 0.0045 - accuracy: 0.9983 - val_loss: 0.2688 - val_accuracy: 0.9750
Epoch 93/100
75/75 [==============================] - 25s 338ms/step - loss: 0.0038 - accuracy: 0.9987 - val_loss: 0.1687 - val_accuracy: 0.9800
Epoch 94/100
75/75 [==============================] - 24s 324ms/step - loss: 5.1901e-04 - accuracy: 1.0000 - val_loss: 0.1688 - val_accuracy: 0.9783
Epoch 95/100
75/75 [==============================] - 24s 321ms/step - loss: 4.0852e-04 - accuracy: 1.0000 - val_loss: 0.1942 - val_accuracy: 0.9817
Epoch 96/100
75/75 [==============================] - 25s 328ms/step - loss: 0.0075 - accuracy: 0.9983 - val_loss: 0.2278 - val_accuracy: 0.9717
Epoch 97/100
75/75 [==============================] - 24s 324ms/step - loss: 0.0038 - accuracy: 0.9983 - val_loss: 0.2509 - val_accuracy: 0.9717
Epoch 98/100
75/75 [==============================] - 24s 320ms/step - loss: 0.0059 - accuracy: 0.9979 - val_loss: 0.2213 - val_accuracy: 0.9717
Epoch 99/100
75/75 [==============================] - 24s 321ms/step - loss: 5.5744e-04 - accuracy: 1.0000 - val_loss: 0.2219 - val_accuracy: 0.9733
Epoch 100/100
75/75 [==============================] - 24s 318ms/step - loss: 4.3564e-04 - accuracy: 0.9996 - val_loss: 0.2329 - val_accuracy: 0.9700
```

In [64]:

```
plot_training_results(results100, model1)
```




```
19/19 [==============================] - 1s 73ms/step
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       310
           1       0.97      0.97      0.97       290

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600

Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_33 (Conv2D)          (None, 126, 126, 32)      896

 activation_55 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_33 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_34 (Conv2D)          (None, 61, 61, 32)        9248

 activation_56 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_34 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_35 (Conv2D)          (None, 28, 28, 64)        18496
```

```
 activation_57 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_35 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_11 (Flatten)        (None, 12544)             0

 dense_22 (Dense)            (None, 64)                802880

 activation_58 (Activation)  (None, 64)                0

 dropout_11 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 1)                 65

 activation_59 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

As you can see, when adjusting the batch size, we were able to reduce the convergence gap between loss and accuracy charts.

Lets try to run the model at 64 batch, and see if it will help improve closing the convergence gap.

100 Epoch *at 64 batch*

In [66]:

```python
# Exexcute the model
model1 = create_model1()
results100_64 = model1.fit(X_train, y_train, batch_size = 64,
          verbose=1,
          epochs=100,
          validation_data= (X_val, y_val),
          shuffle = False)

# Save model
model1.save('model/brain_tumor_ver1_100_epochs_64.h5')
```

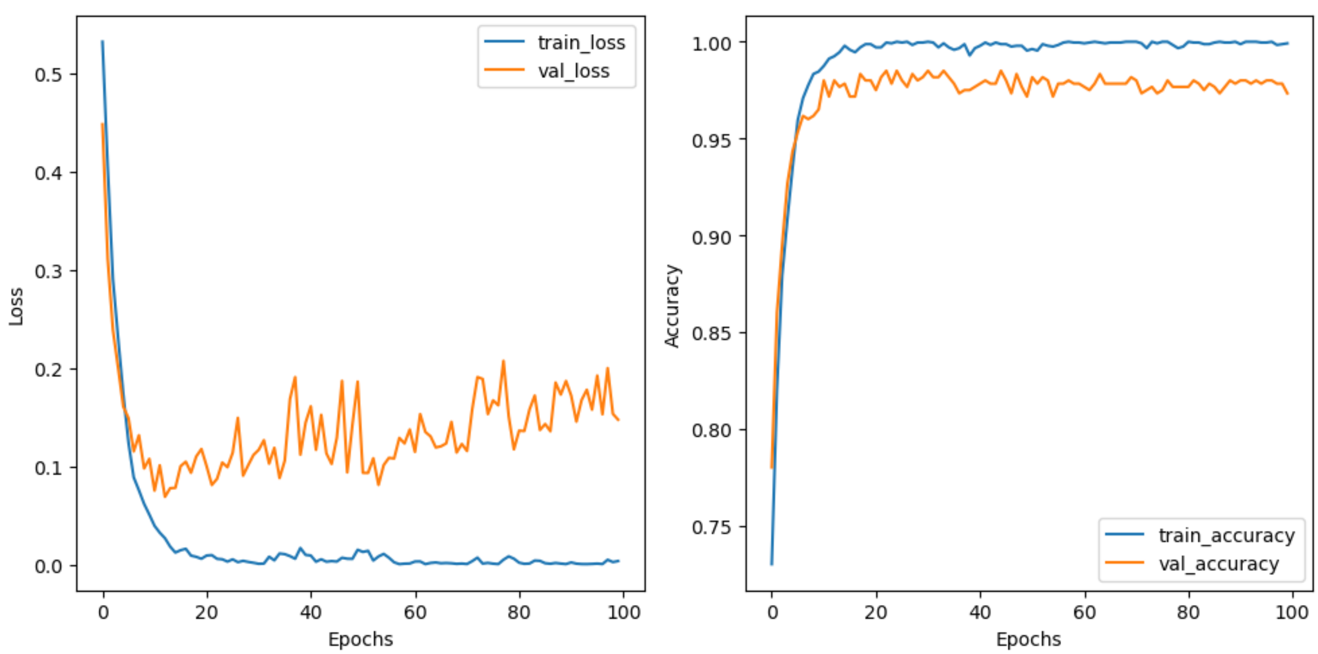
```
Epoch 1/100
38/38 [==============================] - 24s 615ms/step - loss: 0.5318 - accuracy: 0.7300 - val_loss: 0.4478 - val_accuracy: 0.7800
Epoch 2/100
38/38 [==============================] - 24s 624ms/step - loss: 0.4053 - accuracy: 0.8200 - val_loss: 0.3110 - val_accuracy: 0.8600
Epoch 3/100
38/38 [==============================] - 23s 617ms/step - loss: 0.2908 - accuracy: 0.8792 - val_loss: 0.2389 - val_accuracy: 0.8950
Epoch 4/100
38/38 [==============================] - 24s 621ms/step - loss: 0.2306 - accuracy: 0.9083 - val_loss: 0.2002 - val_accuracy: 0.9267
Epoch 5/100
38/38 [==============================] - 24s 624ms/step - loss: 0.1732 - accuracy: 0.9350 - val_loss: 0.1605 - val_accuracy: 0.9433
Epoch 6/100
38/38 [==============================] - 24s 621ms/step - loss: 0.1238 - accuracy: 0.9596 - val_loss: 0.1488 - val_accuracy: 0.9533
Epoch 7/100
38/38 [==============================] - 24s 621ms/step - loss: 0.0886 - accuracy: 0.9708 - val_loss: 0.1150 - val_accuracy: 0.9617
Epoch 8/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0754 - accuracy: 0.9775 - val_loss: 0.1315 - val_accuracy: 0.9600
Epoch 9/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0616 - accuracy: 0.9833 - val_loss: 0.0977 - val_accuracy: 0.9617
Epoch 10/100
38/38 [==============================] - 24s 619ms/step - loss: 0.0507 - accuracy: 0.9846 - val_loss: 0.1075 - val_accuracy: 0.9650
Epoch 11/100
38/38 [==============================] - 25s 651ms/step - loss: 0.0393 - accuracy: 0.9875 - val_loss: 0.0751 - val_accuracy: 0.9800
Epoch 12/100
38/38 [==============================] - 24s 643ms/step - loss: 0.0324 - accuracy: 0.9912 - val_loss: 0.1009 - val_accuracy: 0.9717
Epoch 13/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0267 - accuracy: 0.9925 - val_loss: 0.0688 - val_accuracy: 0.9800
Epoch 14/100
38/38 [==============================] - 24s 637ms/step - loss: 0.0182 - accuracy: 0.9946 - val_loss: 0.0776 - val_accuracy: 0.9767
Epoch 15/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0122 - accuracy: 0.9979 - val_loss: 0.0779 - val_accuracy: 0.9783
Epoch 16/100
38/38 [==============================] - 24s 639ms/step - loss: 0.0145 - accuracy: 0.9958 - val_loss: 0.1000 - val_accuracy: 0.9717
Epoch 17/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0160 - accuracy: 0.9946 - val_loss: 0.1046 - val_accuracy: 0.9717
Epoch 18/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0090 - accuracy: 0.9971 - val_loss: 0.0933 - val_accuracy: 0.9833
Epoch 19/100
38/38 [==============================] - 24s 637ms/step - loss: 0.0078 - accuracy: 0.9987 - val_loss: 0.1099 - val_accuracy: 0.9800
Epoch 20/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0058 - accuracy: 0.9987 - val_loss: 0.1175 - val_accuracy: 0.9800
Epoch 21/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0091 - accuracy: 0.9971 - val_loss: 0.0999 - val_accuracy: 0.9750
Epoch 22/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0095 - accuracy: 0.9971 - val_loss: 0.0809 - val_accuracy: 0.9817
Epoch 23/100
38/38 [==============================] - 25s 646ms/step - loss: 0.0058 - accuracy: 0.9996 - val_loss: 0.0872 - val_accuracy: 0.9850
Epoch 24/100
38/38 [==============================] - 24s 634ms/step - loss: 0.0053 - accuracy: 0.9992 - val_loss: 0.1037 - val_accuracy: 0.9783
Epoch 25/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.0990 - val_accuracy: 0.9850
Epoch 26/100
```

```
38/38 [==============================] - 24s 628ms/step - loss: 0.0052 - accuracy: 0.9996 - val_loss: 0.1136 - val_accuracy: 0.9800
Epoch 27/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.1492 - val_accuracy: 0.9767
Epoch 28/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0037 - accuracy: 0.9983 - val_loss: 0.0903 - val_accuracy: 0.9833
Epoch 29/100
38/38 [==============================] - 25s 662ms/step - loss: 0.0026 - accuracy: 0.9996 - val_loss: 0.1012 - val_accuracy: 0.9800
Epoch 30/100
38/38 [==============================] - 25s 654ms/step - loss: 0.0017 - accuracy: 0.9996 - val_loss: 0.1114 - val_accuracy: 0.9817
Epoch 31/100
38/38 [==============================] - 26s 679ms/step - loss: 7.0733e-04 - accuracy: 1.0000 - val_loss: 0.1166 - val_accuracy: 0.9850
Epoch 32/100
38/38 [==============================] - 25s 658ms/step - loss: 9.1828e-04 - accuracy: 0.9996 - val_loss: 0.1264 - val_accuracy: 0.9817
Epoch 33/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0079 - accuracy: 0.9971 - val_loss: 0.1026 - val_accuracy: 0.9817
Epoch 34/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0042 - accuracy: 0.9992 - val_loss: 0.1187 - val_accuracy: 0.9850
Epoch 35/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0112 - accuracy: 0.9971 - val_loss: 0.0879 - val_accuracy: 0.9817
Epoch 36/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0105 - accuracy: 0.9958 - val_loss: 0.1056 - val_accuracy: 0.9783
Epoch 37/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0085 - accuracy: 0.9967 - val_loss: 0.1683 - val_accuracy: 0.9733
Epoch 38/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0058 - accuracy: 0.9987 - val_loss: 0.1906 - val_accuracy: 0.9750
Epoch 39/100
38/38 [==============================] - 24s 639ms/step - loss: 0.0168 - accuracy: 0.9929 - val_loss: 0.1117 - val_accuracy: 0.9750
Epoch 40/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0097 - accuracy: 0.9967 - val_loss: 0.1443 - val_accuracy: 0.9767
Epoch 41/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0091 - accuracy: 0.9979 - val_loss: 0.1609 - val_accuracy: 0.9783
Epoch 42/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0028 - accuracy: 0.9996 - val_loss: 0.1167 - val_accuracy: 0.9800
Epoch 43/100
38/38 [==============================] - 24s 621ms/step - loss: 0.0053 - accuracy: 0.9983 - val_loss: 0.1522 - val_accuracy: 0.9783
Epoch 44/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0028 - accuracy: 0.9996 - val_loss: 0.1130 - val_accuracy: 0.9783
Epoch 45/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0035 - accuracy: 0.9987 - val_loss: 0.1023 - val_accuracy: 0.9850
Epoch 46/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0030 - accuracy: 0.9987 - val_loss: 0.1290 - val_accuracy: 0.9800
Epoch 47/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0069 - accuracy: 0.9975 - val_loss: 0.1868 - val_accuracy: 0.9733
Epoch 48/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0059 - accuracy: 0.9979 - val_loss: 0.0937 - val_accuracy: 0.9833
Epoch 49/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0058 - accuracy: 0.9979 - val_loss: 0.1424 - val_accuracy: 0.9767
Epoch 50/100
38/38 [==============================] - 24s 636ms/step - loss: 0.0150 - accuracy: 0.9954 - val_loss: 0.1860 - val_accuracy: 0.9717
Epoch 51/100
38/38 [==============================] - 24s 645ms/step - loss: 0.0128 - accuracy: 0.9962 - val_loss: 0.0933 - val_accuracy: 0.9817
Epoch 52/100
38/38 [==============================] - 25s 657ms/step - loss: 0.0139 - accuracy: 0.9954 - val_loss: 0.0932 - val_accuracy: 0.9783
Epoch 53/100
38/38 [==============================] - 25s 672ms/step - loss: 0.0040 - accuracy: 0.9987 - val_loss: 0.1080 - val_accuracy: 0.9817
Epoch 54/100
38/38 [==============================] - 25s 668ms/step - loss: 0.0082 - accuracy: 0.9979 - val_loss: 0.0810 - val_accuracy: 0.9800
Epoch 55/100
38/38 [==============================] - 25s 669ms/step - loss: 0.0108 - accuracy: 0.9975 - val_loss: 0.1012 - val_accuracy: 0.9717
Epoch 56/100
38/38 [==============================] - 26s 681ms/step - loss: 0.0069 - accuracy: 0.9983 - val_loss: 0.1085 - val_accuracy: 0.9783
Epoch 57/100
38/38 [==============================] - 26s 687ms/step - loss: 0.0022 - accuracy: 0.9996 - val_loss: 0.1078 - val_accuracy: 0.9783
Epoch 58/100
38/38 [==============================] - 26s 684ms/step - loss: 3.7600e-04 - accuracy: 1.0000 - val_loss: 0.1288 - val_accuracy: 0.9800
Epoch 59/100
38/38 [==============================] - 26s 680ms/step - loss: 8.2921e-04 - accuracy: 0.9996 - val_loss: 0.1232 - val_accuracy: 0.9783
Epoch 60/100
38/38 [==============================] - 26s 687ms/step - loss: 9.9065e-04 - accuracy: 0.9996 - val_loss: 0.1373 - val_accuracy: 0.9783
Epoch 61/100
38/38 [==============================] - 26s 677ms/step - loss: 0.0030 - accuracy: 0.9992 - val_loss: 0.1145 - val_accuracy: 0.9767
Epoch 62/100
38/38 [==============================] - 25s 656ms/step - loss: 0.0031 - accuracy: 0.9996 - val_loss: 0.1530 - val_accuracy: 0.9750
Epoch 63/100
38/38 [==============================] - 25s 660ms/step - loss: 3.6971e-04 - accuracy: 1.0000 - val_loss: 0.1348 - val_accuracy: 0.9783
Epoch 64/100
38/38 [==============================] - 24s 643ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.1303 - val_accuracy: 0.9833
Epoch 65/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0021 - accuracy: 0.9992 - val_loss: 0.1190 - val_accuracy: 0.9783
Epoch 66/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0013 - accuracy: 0.9996 - val_loss: 0.1202 - val_accuracy: 0.9783
Epoch 67/100
38/38 [==============================] - 24s 623ms/step - loss: 0.0015 - accuracy: 0.9996 - val_loss: 0.1231 - val_accuracy: 0.9783
Epoch 68/100
38/38 [==============================] - 24s 645ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 0.1451 - val_accuracy: 0.9783
Epoch 69/100
38/38 [==============================] - 24s 627ms/step - loss: 6.6168e-04 - accuracy: 1.0000 - val_loss: 0.1139 - val_accuracy: 0.9783
Epoch 70/100
38/38 [==============================] - 24s 632ms/step - loss: 9.3560e-04 - accuracy: 1.0000 - val_loss: 0.1226 - val_accuracy: 0.9817
Epoch 71/100
38/38 [==============================] - 24s 633ms/step - loss: 4.8742e-04 - accuracy: 1.0000 - val_loss: 0.1154 - val_accuracy: 0.9800
Epoch 72/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0034 - accuracy: 0.9992 - val_loss: 0.1581 - val_accuracy: 0.9733
```

```
Epoch 73/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0070 - accuracy: 0.9967 - val_loss: 0.1907 - val_accuracy: 0.9750
Epoch 74/100
38/38 [==============================] - 24s 622ms/step - loss: 9.3820e-04 - accuracy: 1.0000 - val_loss: 0.1887 - val_accuracy: 0.9767
Epoch 75/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0016 - accuracy: 0.9992 - val_loss: 0.1530 - val_accuracy: 0.9733
Epoch 76/100
38/38 [==============================] - 24s 628ms/step - loss: 8.4184e-04 - accuracy: 1.0000 - val_loss: 0.1669 - val_accuracy: 0.9750
Epoch 77/100
38/38 [==============================] - 24s 622ms/step - loss: 3.9923e-04 - accuracy: 1.0000 - val_loss: 0.1619 - val_accuracy: 0.9800
Epoch 78/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0049 - accuracy: 0.9983 - val_loss: 0.2071 - val_accuracy: 0.9767
Epoch 79/100
38/38 [==============================] - 24s 622ms/step - loss: 0.0083 - accuracy: 0.9967 - val_loss: 0.1507 - val_accuracy: 0.9767
Epoch 80/100
38/38 [==============================] - 24s 622ms/step - loss: 0.0059 - accuracy: 0.9975 - val_loss: 0.1171 - val_accuracy: 0.9767
Epoch 81/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.1362 - val_accuracy: 0.9767
Epoch 82/100
38/38 [==============================] - 24s 624ms/step - loss: 6.8026e-04 - accuracy: 0.9996 - val_loss: 0.1358 - val_accuracy: 0.9800
Epoch 83/100
38/38 [==============================] - 24s 636ms/step - loss: 9.7356e-04 - accuracy: 0.9996 - val_loss: 0.1577 - val_accuracy: 0.9783
Epoch 84/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0040 - accuracy: 0.9987 - val_loss: 0.1719 - val_accuracy: 0.9750
Epoch 85/100
38/38 [==============================] - 24s 622ms/step - loss: 0.0037 - accuracy: 0.9987 - val_loss: 0.1368 - val_accuracy: 0.9783
Epoch 86/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0013 - accuracy: 0.9996 - val_loss: 0.1429 - val_accuracy: 0.9767
Epoch 87/100
38/38 [==============================] - 24s 629ms/step - loss: 7.5165e-04 - accuracy: 1.0000 - val_loss: 0.1355 - val_accuracy: 0.9733
Epoch 88/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.1851 - val_accuracy: 0.9767
Epoch 89/100
38/38 [==============================] - 24s 631ms/step - loss: 9.2269e-04 - accuracy: 0.9996 - val_loss: 0.1728 - val_accuracy: 0.9800
Epoch 90/100
38/38 [==============================] - 24s 631ms/step - loss: 4.2589e-04 - accuracy: 1.0000 - val_loss: 0.1867 - val_accuracy: 0.9783
Epoch 91/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0021 - accuracy: 0.9987 - val_loss: 0.1717 - val_accuracy: 0.9800
Epoch 92/100
38/38 [==============================] - 24s 629ms/step - loss: 8.4094e-04 - accuracy: 1.0000 - val_loss: 0.1453 - val_accuracy: 0.9800
Epoch 93/100
38/38 [==============================] - 24s 634ms/step - loss: 4.5506e-04 - accuracy: 1.0000 - val_loss: 0.1673 - val_accuracy: 0.9783
Epoch 94/100
38/38 [==============================] - 24s 633ms/step - loss: 4.0977e-04 - accuracy: 1.0000 - val_loss: 0.1778 - val_accuracy: 0.9800
Epoch 95/100
38/38 [==============================] - 24s 627ms/step - loss: 6.1051e-04 - accuracy: 0.9996 - val_loss: 0.1575 - val_accuracy: 0.9783
Epoch 96/100
38/38 [==============================] - 24s 627ms/step - loss: 8.9954e-04 - accuracy: 0.9996 - val_loss: 0.1921 - val_accuracy: 0.9800
Epoch 97/100
38/38 [==============================] - 24s 636ms/step - loss: 4.4921e-04 - accuracy: 1.0000 - val_loss: 0.1528 - val_accuracy: 0.9800
Epoch 98/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0048 - accuracy: 0.9983 - val_loss: 0.1998 - val_accuracy: 0.9783
Epoch 99/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0025 - accuracy: 0.9987 - val_loss: 0.1532 - val_accuracy: 0.9783
Epoch 100/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.1473 - val_accuracy: 0.9733
```

In [67]:

```
plot_training_results(results100_64, model1)
```



```
19/19 [==============================] - 1s 75ms/step
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       310
           1       0.98      0.97      0.97       290

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600

Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_36 (Conv2D)          (None, 126, 126, 32)      896

 activation_60 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_36 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_37 (Conv2D)          (None, 61, 61, 32)        9248

 activation_61 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_37 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_38 (Conv2D)          (None, 28, 28, 64)        18496

 activation_62 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_38 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_12 (Flatten)        (None, 12544)             0

 dense_24 (Dense)            (None, 64)                802880

 activation_63 (Activation)  (None, 64)                0

 dropout_12 (Dropout)        (None, 64)                0

 dense_25 (Dense)            (None, 1)                 65

 activation_64 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

Making the batch 64 did help smoothen out the accuracy and loss, but made the validation accuracy and loss a bit more volatile. Next step is to regularize the model to reduce sudden jumps in validation loss and accuracy.

Ver 2 - ADD REGULARIZATION

Regularization techniques, such as L2 regularization, play a crucial role in managing the complexity of a model and enhancing its generalization capabilities by mitigating overfitting. L2 regularization achieves this by imposing a penalty on large weight values within the model. By penalizing these weights, L2 regularization encourages the model to prioritize robust and significant patterns in the data. Consequently, the model becomes less likely to rely on noise or irrelevant features, resulting in improved performance when presented with new, unseen data.

Instantiate

In [21]:

```
# INSIDE THE FUNCTION BELOW
```

Compile

In [69]:

```
def create_model2():

    model = Sequential()

    model.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3), kernel_regularizer=l2(0.001)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, (3, 3), kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(64, (3, 3), kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

```python
    # Flatten the output of the previous layer to a 1D array
    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))


    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
################################################################################
################################################################################
```

Execute

100 Epoch *at 64 batch* with *Regularizer*

In [72]:

```python
# Exexcute the model
model2 = create_model2()
results100_64_ver2 = model2.fit(X_train, y_train, batch_size = 64,
          verbose=1,
          epochs=100,
          validation_data= (X_val, y_val),
          shuffle = False)

# Save model
model2.save('model/brain_tumor_ver2_100_epochs_64_reg.h5')
```

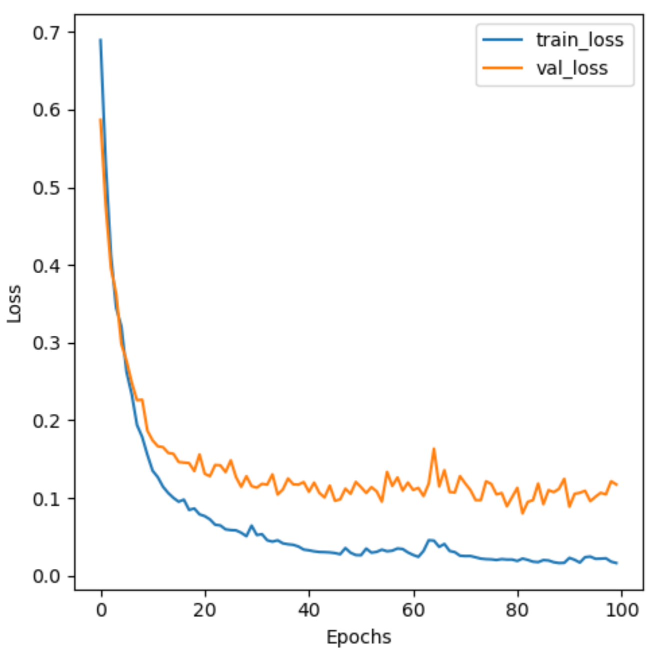
```
Epoch 1/100
38/38 [==============================] - 24s 605ms/step - loss: 0.6894 - accuracy: 0.7312 - val_loss: 0.5867 - val_accuracy: 0.8100
Epoch 2/100
38/38 [==============================] - 23s 616ms/step - loss: 0.5313 - accuracy: 0.8183 - val_loss: 0.4759 - val_accuracy: 0.8500
Epoch 3/100
38/38 [==============================] - 24s 620ms/step - loss: 0.4134 - accuracy: 0.8788 - val_loss: 0.3962 - val_accuracy: 0.8883
Epoch 4/100
38/38 [==============================] - 24s 622ms/step - loss: 0.3448 - accuracy: 0.9071 - val_loss: 0.3628 - val_accuracy: 0.8933
Epoch 5/100
38/38 [==============================] - 23s 616ms/step - loss: 0.3210 - accuracy: 0.9146 - val_loss: 0.2983 - val_accuracy: 0.9350
Epoch 6/100
38/38 [==============================] - 24s 623ms/step - loss: 0.2621 - accuracy: 0.9400 - val_loss: 0.2764 - val_accuracy: 0.9433
Epoch 7/100
38/38 [==============================] - 24s 624ms/step - loss: 0.2333 - accuracy: 0.9538 - val_loss: 0.2485 - val_accuracy: 0.9500
Epoch 8/100
38/38 [==============================] - 24s 622ms/step - loss: 0.1941 - accuracy: 0.9642 - val_loss: 0.2258 - val_accuracy: 0.9583
Epoch 9/100
38/38 [==============================] - 24s 636ms/step - loss: 0.1785 - accuracy: 0.9717 - val_loss: 0.2265 - val_accuracy: 0.9517
Epoch 10/100
38/38 [==============================] - 24s 624ms/step - loss: 0.1555 - accuracy: 0.9812 - val_loss: 0.1866 - val_accuracy: 0.9683
Epoch 11/100
38/38 [==============================] - 24s 643ms/step - loss: 0.1352 - accuracy: 0.9867 - val_loss: 0.1742 - val_accuracy: 0.9717
Epoch 12/100
38/38 [==============================] - 24s 637ms/step - loss: 0.1267 - accuracy: 0.9892 - val_loss: 0.1666 - val_accuracy: 0.9783
Epoch 13/100
38/38 [==============================] - 24s 632ms/step - loss: 0.1145 - accuracy: 0.9887 - val_loss: 0.1654 - val_accuracy: 0.9733
Epoch 14/100
38/38 [==============================] - 24s 626ms/step - loss: 0.1064 - accuracy: 0.9917 - val_loss: 0.1580 - val_accuracy: 0.9783
Epoch 15/100
38/38 [==============================] - 24s 624ms/step - loss: 0.1001 - accuracy: 0.9917 - val_loss: 0.1568 - val_accuracy: 0.9767
Epoch 16/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0952 - accuracy: 0.9925 - val_loss: 0.1463 - val_accuracy: 0.9767
Epoch 17/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0979 - accuracy: 0.9921 - val_loss: 0.1456 - val_accuracy: 0.9767
Epoch 18/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0846 - accuracy: 0.9958 - val_loss: 0.1450 - val_accuracy: 0.9800
Epoch 19/100
38/38 [==============================] - 24s 621ms/step - loss: 0.0866 - accuracy: 0.9917 - val_loss: 0.1347 - val_accuracy: 0.9800
Epoch 20/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0789 - accuracy: 0.9962 - val_loss: 0.1560 - val_accuracy: 0.9767
Epoch 21/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0767 - accuracy: 0.9954 - val_loss: 0.1313 - val_accuracy: 0.9800
Epoch 22/100
38/38 [==============================] - 24s 621ms/step - loss: 0.0725 - accuracy: 0.9971 - val_loss: 0.1280 - val_accuracy: 0.9850
Epoch 23/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0656 - accuracy: 0.9983 - val_loss: 0.1425 - val_accuracy: 0.9783
Epoch 24/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0647 - accuracy: 0.9971 - val_loss: 0.1421 - val_accuracy: 0.9750
Epoch 25/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0596 - accuracy: 0.9987 - val_loss: 0.1333 - val_accuracy: 0.9783
Epoch 26/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0587 - accuracy: 0.9983 - val_loss: 0.1484 - val_accuracy: 0.9750
Epoch 27/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0584 - accuracy: 0.9979 - val_loss: 0.1270 - val_accuracy: 0.9750
Epoch 28/100
38/38 [==============================] - 24s 622ms/step - loss: 0.0553 - accuracy: 0.9987 - val_loss: 0.1143 - val_accuracy: 0.9833
Epoch 29/100
```

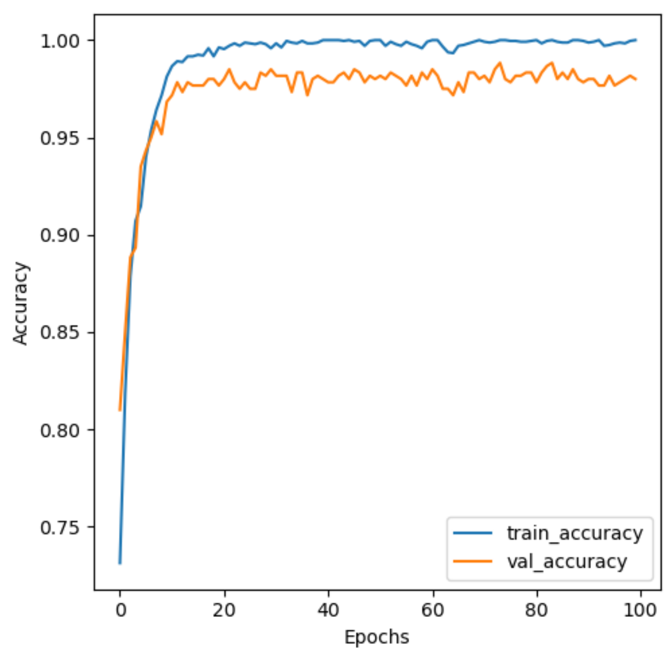
```
38/38 [==============================] - 24s 630ms/step - loss: 0.0508 - accuracy: 0.9979 - val_loss: 0.1280 - val_accuracy: 0.9817
Epoch 30/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0644 - accuracy: 0.9958 - val_loss: 0.1153 - val_accuracy: 0.9850
Epoch 31/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0523 - accuracy: 0.9983 - val_loss: 0.1133 - val_accuracy: 0.9817
Epoch 32/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0534 - accuracy: 0.9962 - val_loss: 0.1182 - val_accuracy: 0.9817
Epoch 33/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0455 - accuracy: 0.9996 - val_loss: 0.1172 - val_accuracy: 0.9817
Epoch 34/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0439 - accuracy: 0.9987 - val_loss: 0.1304 - val_accuracy: 0.9733
Epoch 35/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0455 - accuracy: 0.9983 - val_loss: 0.1044 - val_accuracy: 0.9833
Epoch 36/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0416 - accuracy: 0.9996 - val_loss: 0.1110 - val_accuracy: 0.9833
Epoch 37/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0404 - accuracy: 0.9983 - val_loss: 0.1251 - val_accuracy: 0.9717
Epoch 38/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0397 - accuracy: 0.9983 - val_loss: 0.1176 - val_accuracy: 0.9800
Epoch 39/100
38/38 [==============================] - 24s 623ms/step - loss: 0.0375 - accuracy: 0.9987 - val_loss: 0.1172 - val_accuracy: 0.9817
Epoch 40/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0336 - accuracy: 1.0000 - val_loss: 0.1205 - val_accuracy: 0.9800
Epoch 41/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0326 - accuracy: 1.0000 - val_loss: 0.1080 - val_accuracy: 0.9783
Epoch 42/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0313 - accuracy: 1.0000 - val_loss: 0.1198 - val_accuracy: 0.9783
Epoch 43/100
38/38 [==============================] - 25s 652ms/step - loss: 0.0305 - accuracy: 1.0000 - val_loss: 0.1066 - val_accuracy: 0.9817
Epoch 44/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0303 - accuracy: 0.9996 - val_loss: 0.1007 - val_accuracy: 0.9833
Epoch 45/100
38/38 [==============================] - 24s 634ms/step - loss: 0.0299 - accuracy: 1.0000 - val_loss: 0.1160 - val_accuracy: 0.9800
Epoch 46/100
38/38 [==============================] - 24s 641ms/step - loss: 0.0291 - accuracy: 0.9992 - val_loss: 0.0964 - val_accuracy: 0.9850
Epoch 47/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0276 - accuracy: 0.9996 - val_loss: 0.0983 - val_accuracy: 0.9833
Epoch 48/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0356 - accuracy: 0.9971 - val_loss: 0.1120 - val_accuracy: 0.9783
Epoch 49/100
38/38 [==============================] - 24s 634ms/step - loss: 0.0295 - accuracy: 0.9996 - val_loss: 0.1054 - val_accuracy: 0.9817
Epoch 50/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0265 - accuracy: 1.0000 - val_loss: 0.1207 - val_accuracy: 0.9800
Epoch 51/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0263 - accuracy: 1.0000 - val_loss: 0.1139 - val_accuracy: 0.9817
Epoch 52/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0349 - accuracy: 0.9971 - val_loss: 0.1065 - val_accuracy: 0.9800
Epoch 53/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0296 - accuracy: 0.9992 - val_loss: 0.1140 - val_accuracy: 0.9833
Epoch 54/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0306 - accuracy: 0.9979 - val_loss: 0.1086 - val_accuracy: 0.9817
Epoch 55/100
38/38 [==============================] - 24s 641ms/step - loss: 0.0335 - accuracy: 0.9971 - val_loss: 0.0953 - val_accuracy: 0.9800
Epoch 56/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0314 - accuracy: 0.9992 - val_loss: 0.1336 - val_accuracy: 0.9767
Epoch 57/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0322 - accuracy: 0.9979 - val_loss: 0.1155 - val_accuracy: 0.9817
Epoch 58/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0350 - accuracy: 0.9971 - val_loss: 0.1265 - val_accuracy: 0.9767
Epoch 59/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0343 - accuracy: 0.9958 - val_loss: 0.1095 - val_accuracy: 0.9833
Epoch 60/100
38/38 [==============================] - 24s 623ms/step - loss: 0.0299 - accuracy: 0.9992 - val_loss: 0.1198 - val_accuracy: 0.9800
Epoch 61/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0267 - accuracy: 1.0000 - val_loss: 0.1103 - val_accuracy: 0.9850
Epoch 62/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0242 - accuracy: 1.0000 - val_loss: 0.1130 - val_accuracy: 0.9817
Epoch 63/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0319 - accuracy: 0.9967 - val_loss: 0.1025 - val_accuracy: 0.9750
Epoch 64/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0457 - accuracy: 0.9937 - val_loss: 0.1186 - val_accuracy: 0.9750
Epoch 65/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0451 - accuracy: 0.9933 - val_loss: 0.1635 - val_accuracy: 0.9717
Epoch 66/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0370 - accuracy: 0.9971 - val_loss: 0.1148 - val_accuracy: 0.9783
Epoch 67/100
38/38 [==============================] - 23s 618ms/step - loss: 0.0409 - accuracy: 0.9975 - val_loss: 0.1359 - val_accuracy: 0.9733
Epoch 68/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0318 - accuracy: 0.9983 - val_loss: 0.1077 - val_accuracy: 0.9833
Epoch 69/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0303 - accuracy: 0.9992 - val_loss: 0.1071 - val_accuracy: 0.9833
Epoch 70/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0257 - accuracy: 1.0000 - val_loss: 0.1281 - val_accuracy: 0.9800
Epoch 71/100
38/38 [==============================] - 25s 655ms/step - loss: 0.0252 - accuracy: 0.9992 - val_loss: 0.1187 - val_accuracy: 0.9817
Epoch 72/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0253 - accuracy: 0.9987 - val_loss: 0.1102 - val_accuracy: 0.9783
Epoch 73/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0237 - accuracy: 0.9992 - val_loss: 0.0972 - val_accuracy: 0.9850
Epoch 74/100
38/38 [==============================] - 24s 637ms/step - loss: 0.0220 - accuracy: 1.0000 - val_loss: 0.0973 - val_accuracy: 0.9883
Epoch 75/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0213 - accuracy: 1.0000 - val_loss: 0.1215 - val_accuracy: 0.9800
```

```
Epoch 76/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0210 - accuracy: 0.9996 - val_loss: 0.1180 - val_accuracy: 0.9783
Epoch 77/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0202 - accuracy: 0.9996 - val_loss: 0.1047 - val_accuracy: 0.9817
Epoch 78/100
38/38 [==============================] - 24s 623ms/step - loss: 0.0213 - accuracy: 0.9992 - val_loss: 0.1065 - val_accuracy: 0.9817
Epoch 79/100
38/38 [==============================] - 24s 632ms/step - loss: 0.0207 - accuracy: 0.9992 - val_loss: 0.0894 - val_accuracy: 0.9833
Epoch 80/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0208 - accuracy: 0.9996 - val_loss: 0.1011 - val_accuracy: 0.9833
Epoch 81/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0189 - accuracy: 1.0000 - val_loss: 0.1131 - val_accuracy: 0.9783
Epoch 82/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0220 - accuracy: 0.9983 - val_loss: 0.0801 - val_accuracy: 0.9833
Epoch 83/100
38/38 [==============================] - 24s 631ms/step - loss: 0.0203 - accuracy: 0.9996 - val_loss: 0.0950 - val_accuracy: 0.9867
Epoch 84/100
38/38 [==============================] - 24s 626ms/step - loss: 0.0180 - accuracy: 1.0000 - val_loss: 0.0971 - val_accuracy: 0.9883
Epoch 85/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0174 - accuracy: 0.9992 - val_loss: 0.1186 - val_accuracy: 0.9800
Epoch 86/100
38/38 [==============================] - 24s 633ms/step - loss: 0.0201 - accuracy: 0.9987 - val_loss: 0.0920 - val_accuracy: 0.9833
Epoch 87/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0196 - accuracy: 0.9987 - val_loss: 0.1101 - val_accuracy: 0.9800
Epoch 88/100
38/38 [==============================] - 24s 629ms/step - loss: 0.0173 - accuracy: 1.0000 - val_loss: 0.1075 - val_accuracy: 0.9850
Epoch 89/100
38/38 [==============================] - 24s 635ms/step - loss: 0.0163 - accuracy: 1.0000 - val_loss: 0.1118 - val_accuracy: 0.9800
Epoch 90/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0165 - accuracy: 0.9996 - val_loss: 0.1247 - val_accuracy: 0.9783
Epoch 91/100
38/38 [==============================] - 24s 628ms/step - loss: 0.0229 - accuracy: 0.9987 - val_loss: 0.0888 - val_accuracy: 0.9800
Epoch 92/100
38/38 [==============================] - 24s 623ms/step - loss: 0.0204 - accuracy: 0.9992 - val_loss: 0.1054 - val_accuracy: 0.9800
Epoch 93/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0168 - accuracy: 1.0000 - val_loss: 0.1067 - val_accuracy: 0.9767
Epoch 94/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0238 - accuracy: 0.9971 - val_loss: 0.1093 - val_accuracy: 0.9767
Epoch 95/100
38/38 [==============================] - 24s 624ms/step - loss: 0.0246 - accuracy: 0.9975 - val_loss: 0.0959 - val_accuracy: 0.9817
Epoch 96/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0217 - accuracy: 0.9983 - val_loss: 0.1016 - val_accuracy: 0.9767
Epoch 97/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0219 - accuracy: 0.9987 - val_loss: 0.1067 - val_accuracy: 0.9783
Epoch 98/100
38/38 [==============================] - 24s 627ms/step - loss: 0.0223 - accuracy: 0.9983 - val_loss: 0.1046 - val_accuracy: 0.9800
Epoch 99/100
38/38 [==============================] - 24s 630ms/step - loss: 0.0180 - accuracy: 0.9996 - val_loss: 0.1215 - val_accuracy: 0.9817
Epoch 100/100
38/38 [==============================] - 24s 625ms/step - loss: 0.0163 - accuracy: 1.0000 - val_loss: 0.1174 - val_accuracy: 0.9800
```

In [74]:

```
plot_training_results(results100_64_ver2, model2)
```






```
19/19 [==============================] - 1s 72ms/step
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       310
           1       0.98      0.98      0.98       290

    accuracy                           0.98       600
```

```
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_42 (Conv2D)          (None, 126, 126, 32)      896       
                                                                 
 activation_70 (Activation)  (None, 126, 126, 32)      0         
                                                                 
 max_pooling2d_42 (MaxPoolin  (None, 63, 63, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_43 (Conv2D)          (None, 61, 61, 32)        9248      
                                                                 
 activation_71 (Activation)  (None, 61, 61, 32)        0         
                                                                 
 max_pooling2d_43 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_44 (Conv2D)          (None, 28, 28, 64)        18496     
                                                                 
 activation_72 (Activation)  (None, 28, 28, 64)        0         
                                                                 
 max_pooling2d_44 (MaxPoolin  (None, 14, 14, 64)       0         
 g2D)                                                            
                                                                 
 flatten_14 (Flatten)        (None, 12544)             0         
                                                                 
 dense_28 (Dense)            (None, 64)                802880    
                                                                 
 activation_73 (Activation)  (None, 64)                0         
                                                                 
 dropout_14 (Dropout)        (None, 64)                0         
                                                                 
 dense_29 (Dense)            (None, 1)                 65        
                                                                 
 activation_74 (Activation)  (None, 1)                 0         
                                                                 
=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

Although the gap has closed up a bit, it made the training loss and accuracy more volatile, resulting to be ineffective. To test the waters, data augmentation will be added into the model. This will artificially increase the data size and add more diversity to the dataset with hopes to reduce overfitting and add more generalization to close the gap and reduce volatility.

Ver 3 - ADD DATA AUGMENTATION

Instantiate

In [32]:

```
# INSIDE THE FUNCTION BELOW
```

Compile

In [80]:

```
def create_model3():

    #Instantiate model
    model = Sequential()

    # Create an instance of the ImageDataGenerator with desired augmentation parameters
    datagen = ImageDataGenerator(
        rotation_range=rotation_range,  # Randomly rotate images by 10 degrees
        width_shift_range=width_shift_range,  # Randomly shift images horizontally by 10% of the total width
        height_shift_range=height_shift_range,  # Randomly shift images vertically by 10% of the total height
        zoom_range=zoom_range,  # Randomly zoom images by 10%
        horizontal_flip=horizontal_flip  # Randomly flip images horizontally
    )

    # Apply data augmentation to the training data generator
    train_generator = datagen.flow(X_train, y_train, batch_size=batch_size)

    # Define and compile your model
    model = Sequential()
    model.add(Conv2D(32, (3,3), input_shape=(IMG_SIZE, IMG_SIZE, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
```

```
    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model, train_generator
```

100 Epoch *at* *64* *batch* with *Augmentation*

In [81]:

```
# Exexcute the model
# Train the model using the augmented data generator
model3, train_generator = create_model3()
results100_augmented = model3.fit(train_generator, epochs=100, validation_data=(X_val, y_val))

# Save model
model.save('model/brain_tumor_ver3_100_epochs_64_aug.h5')
```

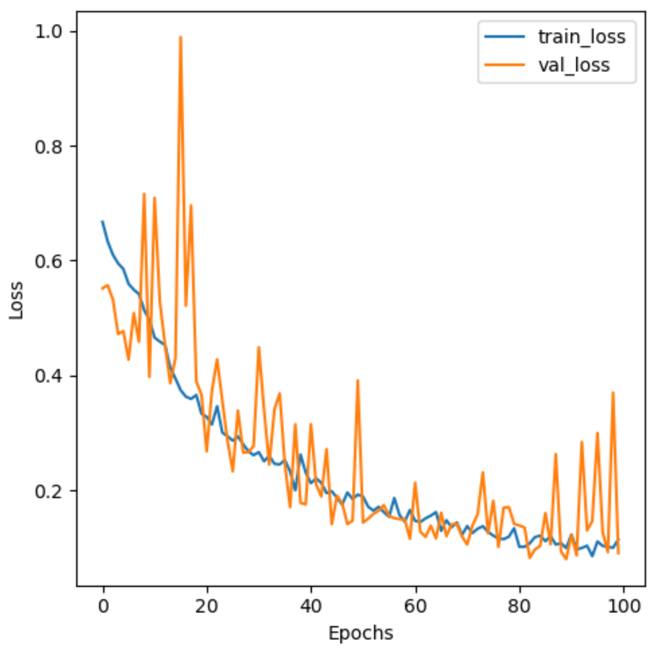
```
Epoch 1/100
75/75 [==============================] - 26s 343ms/step - loss: 0.6673 - accuracy: 0.5933 - val_loss: 0.5517 - val_accuracy: 0.7100
Epoch 2/100
75/75 [==============================] - 27s 362ms/step - loss: 0.6331 - accuracy: 0.6417 - val_loss: 0.5571 - val_accuracy: 0.7467
Epoch 3/100
75/75 [==============================] - 27s 363ms/step - loss: 0.6099 - accuracy: 0.6733 - val_loss: 0.5320 - val_accuracy: 0.7617
Epoch 4/100
75/75 [==============================] - 27s 355ms/step - loss: 0.5951 - accuracy: 0.6862 - val_loss: 0.4722 - val_accuracy: 0.7917
Epoch 5/100
75/75 [==============================] - 27s 359ms/step - loss: 0.5853 - accuracy: 0.7017 - val_loss: 0.4772 - val_accuracy: 0.7983
Epoch 6/100
75/75 [==============================] - 27s 359ms/step - loss: 0.5594 - accuracy: 0.7267 - val_loss: 0.4276 - val_accuracy: 0.8033
Epoch 7/100
75/75 [==============================] - 27s 362ms/step - loss: 0.5492 - accuracy: 0.7262 - val_loss: 0.5086 - val_accuracy: 0.7433
Epoch 8/100
75/75 [==============================] - 27s 360ms/step - loss: 0.5418 - accuracy: 0.7379 - val_loss: 0.4588 - val_accuracy: 0.7833
Epoch 9/100
75/75 [==============================] - 27s 363ms/step - loss: 0.5139 - accuracy: 0.7542 - val_loss: 0.7160 - val_accuracy: 0.6433
Epoch 10/100
75/75 [==============================] - 27s 363ms/step - loss: 0.4965 - accuracy: 0.7567 - val_loss: 0.3975 - val_accuracy: 0.8250
Epoch 11/100
75/75 [==============================] - 28s 367ms/step - loss: 0.4660 - accuracy: 0.7808 - val_loss: 0.7090 - val_accuracy: 0.6917
Epoch 12/100
75/75 [==============================] - 28s 371ms/step - loss: 0.4589 - accuracy: 0.7867 - val_loss: 0.5288 - val_accuracy: 0.7550
Epoch 13/100
75/75 [==============================] - 27s 363ms/step - loss: 0.4529 - accuracy: 0.8058 - val_loss: 0.4533 - val_accuracy: 0.7950
Epoch 14/100
75/75 [==============================] - 27s 360ms/step - loss: 0.4136 - accuracy: 0.8138 - val_loss: 0.3865 - val_accuracy: 0.8283
Epoch 15/100
75/75 [==============================] - 27s 364ms/step - loss: 0.3948 - accuracy: 0.8271 - val_loss: 0.4302 - val_accuracy: 0.8167
Epoch 16/100
75/75 [==============================] - 27s 365ms/step - loss: 0.3744 - accuracy: 0.8425 - val_loss: 0.9889 - val_accuracy: 0.6567
Epoch 17/100
75/75 [==============================] - 27s 362ms/step - loss: 0.3632 - accuracy: 0.8483 - val_loss: 0.5214 - val_accuracy: 0.7550
Epoch 18/100
75/75 [==============================] - 27s 365ms/step - loss: 0.3593 - accuracy: 0.8492 - val_loss: 0.6961 - val_accuracy: 0.7350
Epoch 19/100
75/75 [==============================] - 27s 362ms/step - loss: 0.3661 - accuracy: 0.8500 - val_loss: 0.3900 - val_accuracy: 0.8167
Epoch 20/100
75/75 [==============================] - 28s 366ms/step - loss: 0.3335 - accuracy: 0.8708 - val_loss: 0.3666 - val_accuracy: 0.8450
Epoch 21/100
75/75 [==============================] - 27s 362ms/step - loss: 0.3279 - accuracy: 0.8679 - val_loss: 0.2680 - val_accuracy: 0.8933
Epoch 22/100
75/75 [==============================] - 28s 367ms/step - loss: 0.3148 - accuracy: 0.8763 - val_loss: 0.3733 - val_accuracy: 0.8267
Epoch 23/100
75/75 [==============================] - 27s 362ms/step - loss: 0.3466 - accuracy: 0.8546 - val_loss: 0.4280 - val_accuracy: 0.8067
Epoch 24/100
75/75 [==============================] - 28s 367ms/step - loss: 0.3005 - accuracy: 0.8804 - val_loss: 0.3556 - val_accuracy: 0.8433
Epoch 25/100
75/75 [==============================] - 27s 363ms/step - loss: 0.2943 - accuracy: 0.8854 - val_loss: 0.2861 - val_accuracy: 0.8800
Epoch 26/100
75/75 [==============================] - 27s 364ms/step - loss: 0.2866 - accuracy: 0.8975 - val_loss: 0.2333 - val_accuracy: 0.9017
Epoch 27/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2940 - accuracy: 0.8771 - val_loss: 0.3388 - val_accuracy: 0.8500
Epoch 28/100
75/75 [==============================] - 27s 356ms/step - loss: 0.2803 - accuracy: 0.8854 - val_loss: 0.2655 - val_accuracy: 0.8767
Epoch 29/100
75/75 [==============================] - 27s 363ms/step - loss: 0.2681 - accuracy: 0.8929 - val_loss: 0.2668 - val_accuracy: 0.8833
Epoch 30/100
75/75 [==============================] - 27s 359ms/step - loss: 0.2613 - accuracy: 0.8954 - val_loss: 0.2771 - val_accuracy: 0.8767
Epoch 31/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2667 - accuracy: 0.8971 - val_loss: 0.4490 - val_accuracy: 0.8067
Epoch 32/100
75/75 [==============================] - 27s 358ms/step - loss: 0.2507 - accuracy: 0.9033 - val_loss: 0.3411 - val_accuracy: 0.8533
Epoch 33/100
```

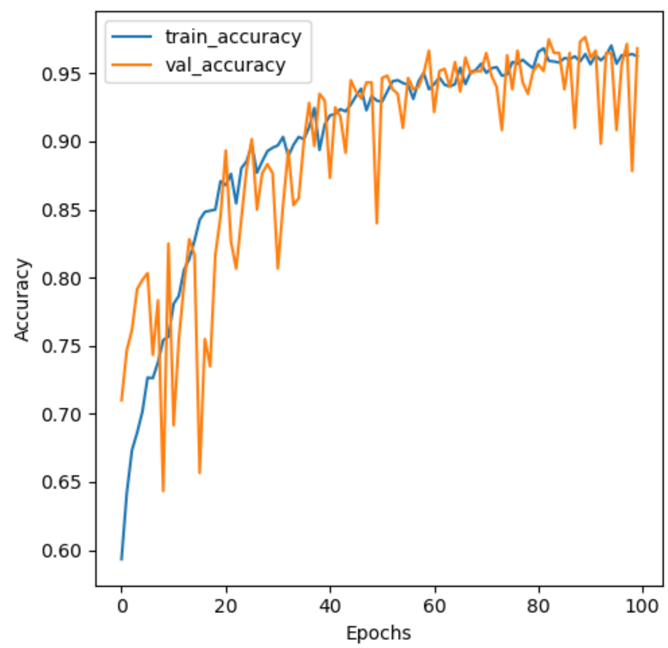
```
75/75 [==============================] - 28s 374ms/step - loss: 0.2601 - accuracy: 0.8892 - val_loss: 0.2449 - val_accuracy: 0.8933
Epoch 34/100
75/75 [==============================] - 27s 356ms/step - loss: 0.2464 - accuracy: 0.8975 - val_loss: 0.3412 - val_accuracy: 0.8533
Epoch 35/100
75/75 [==============================] - 27s 357ms/step - loss: 0.2452 - accuracy: 0.9033 - val_loss: 0.3687 - val_accuracy: 0.8583
Epoch 36/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2526 - accuracy: 0.9017 - val_loss: 0.2460 - val_accuracy: 0.9000
Epoch 37/100
75/75 [==============================] - 27s 357ms/step - loss: 0.2332 - accuracy: 0.9104 - val_loss: 0.1709 - val_accuracy: 0.9283
Epoch 38/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2002 - accuracy: 0.9246 - val_loss: 0.3148 - val_accuracy: 0.8967
Epoch 39/100
75/75 [==============================] - 27s 358ms/step - loss: 0.2626 - accuracy: 0.8938 - val_loss: 0.1783 - val_accuracy: 0.9350
Epoch 40/100
75/75 [==============================] - 27s 360ms/step - loss: 0.2310 - accuracy: 0.9125 - val_loss: 0.1757 - val_accuracy: 0.9300
Epoch 41/100
75/75 [==============================] - 27s 354ms/step - loss: 0.2127 - accuracy: 0.9192 - val_loss: 0.3152 - val_accuracy: 0.8733
Epoch 42/100
75/75 [==============================] - 27s 361ms/step - loss: 0.2209 - accuracy: 0.9200 - val_loss: 0.2103 - val_accuracy: 0.9250
Epoch 43/100
75/75 [==============================] - 27s 357ms/step - loss: 0.2132 - accuracy: 0.9237 - val_loss: 0.1894 - val_accuracy: 0.9183
Epoch 44/100
75/75 [==============================] - 27s 359ms/step - loss: 0.1953 - accuracy: 0.9221 - val_loss: 0.2719 - val_accuracy: 0.8917
Epoch 45/100
75/75 [==============================] - 27s 357ms/step - loss: 0.1987 - accuracy: 0.9271 - val_loss: 0.1413 - val_accuracy: 0.9450
Epoch 46/100
75/75 [==============================] - 27s 358ms/step - loss: 0.1873 - accuracy: 0.9333 - val_loss: 0.1907 - val_accuracy: 0.9367
Epoch 47/100
75/75 [==============================] - 27s 360ms/step - loss: 0.1755 - accuracy: 0.9388 - val_loss: 0.1778 - val_accuracy: 0.9317
Epoch 48/100
75/75 [==============================] - 27s 358ms/step - loss: 0.1964 - accuracy: 0.9229 - val_loss: 0.1415 - val_accuracy: 0.9433
Epoch 49/100
75/75 [==============================] - 27s 362ms/step - loss: 0.1850 - accuracy: 0.9333 - val_loss: 0.1473 - val_accuracy: 0.9433
Epoch 50/100
75/75 [==============================] - 27s 359ms/step - loss: 0.1927 - accuracy: 0.9300 - val_loss: 0.3914 - val_accuracy: 0.8400
Epoch 51/100
75/75 [==============================] - 29s 387ms/step - loss: 0.1893 - accuracy: 0.9292 - val_loss: 0.1441 - val_accuracy: 0.9467
Epoch 52/100
75/75 [==============================] - 27s 359ms/step - loss: 0.1716 - accuracy: 0.9367 - val_loss: 0.1510 - val_accuracy: 0.9483
Epoch 53/100
75/75 [==============================] - 28s 369ms/step - loss: 0.1648 - accuracy: 0.9442 - val_loss: 0.1593 - val_accuracy: 0.9383
Epoch 54/100
75/75 [==============================] - 27s 365ms/step - loss: 0.1715 - accuracy: 0.9450 - val_loss: 0.1646 - val_accuracy: 0.9350
Epoch 55/100
75/75 [==============================] - 28s 375ms/step - loss: 0.1622 - accuracy: 0.9429 - val_loss: 0.1744 - val_accuracy: 0.9100
Epoch 56/100
75/75 [==============================] - 27s 362ms/step - loss: 0.1543 - accuracy: 0.9417 - val_loss: 0.1555 - val_accuracy: 0.9467
Epoch 57/100
75/75 [==============================] - 28s 368ms/step - loss: 0.1868 - accuracy: 0.9312 - val_loss: 0.1517 - val_accuracy: 0.9383
Epoch 58/100
75/75 [==============================] - 27s 360ms/step - loss: 0.1572 - accuracy: 0.9442 - val_loss: 0.1500 - val_accuracy: 0.9400
Epoch 59/100
75/75 [==============================] - 27s 361ms/step - loss: 0.1457 - accuracy: 0.9504 - val_loss: 0.1499 - val_accuracy: 0.9500
Epoch 60/100
75/75 [==============================] - 27s 366ms/step - loss: 0.1658 - accuracy: 0.9383 - val_loss: 0.1157 - val_accuracy: 0.9667
Epoch 61/100
75/75 [==============================] - 27s 360ms/step - loss: 0.1470 - accuracy: 0.9421 - val_loss: 0.2134 - val_accuracy: 0.9217
Epoch 62/100
75/75 [==============================] - 28s 370ms/step - loss: 0.1450 - accuracy: 0.9471 - val_loss: 0.1281 - val_accuracy: 0.9517
Epoch 63/100
75/75 [==============================] - 27s 364ms/step - loss: 0.1514 - accuracy: 0.9417 - val_loss: 0.1192 - val_accuracy: 0.9533
Epoch 64/100
75/75 [==============================] - 29s 382ms/step - loss: 0.1560 - accuracy: 0.9404 - val_loss: 0.1387 - val_accuracy: 0.9400
Epoch 65/100
75/75 [==============================] - 27s 359ms/step - loss: 0.1623 - accuracy: 0.9421 - val_loss: 0.1165 - val_accuracy: 0.9583
Epoch 66/100
75/75 [==============================] - 33s 437ms/step - loss: 0.1297 - accuracy: 0.9542 - val_loss: 0.1611 - val_accuracy: 0.9367
Epoch 67/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1481 - accuracy: 0.9421 - val_loss: 0.1204 - val_accuracy: 0.9617
Epoch 68/100
75/75 [==============================] - 29s 380ms/step - loss: 0.1359 - accuracy: 0.9508 - val_loss: 0.1406 - val_accuracy: 0.9500
Epoch 69/100
75/75 [==============================] - 28s 367ms/step - loss: 0.1442 - accuracy: 0.9525 - val_loss: 0.1419 - val_accuracy: 0.9517
Epoch 70/100
75/75 [==============================] - 27s 365ms/step - loss: 0.1227 - accuracy: 0.9571 - val_loss: 0.1198 - val_accuracy: 0.9517
Epoch 71/100
75/75 [==============================] - 27s 362ms/step - loss: 0.1381 - accuracy: 0.9504 - val_loss: 0.1059 - val_accuracy: 0.9650
Epoch 72/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1258 - accuracy: 0.9538 - val_loss: 0.1387 - val_accuracy: 0.9483
Epoch 73/100
75/75 [==============================] - 27s 364ms/step - loss: 0.1333 - accuracy: 0.9546 - val_loss: 0.1588 - val_accuracy: 0.9400
Epoch 74/100
75/75 [==============================] - 27s 364ms/step - loss: 0.1377 - accuracy: 0.9483 - val_loss: 0.2313 - val_accuracy: 0.9083
Epoch 75/100
75/75 [==============================] - 27s 361ms/step - loss: 0.1279 - accuracy: 0.9492 - val_loss: 0.1255 - val_accuracy: 0.9633
Epoch 76/100
75/75 [==============================] - 27s 360ms/step - loss: 0.1213 - accuracy: 0.9583 - val_loss: 0.1819 - val_accuracy: 0.9383
Epoch 77/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1163 - accuracy: 0.9571 - val_loss: 0.1017 - val_accuracy: 0.9667
Epoch 78/100
75/75 [==============================] - 27s 364ms/step - loss: 0.1151 - accuracy: 0.9600 - val_loss: 0.1700 - val_accuracy: 0.9433
Epoch 79/100
75/75 [==============================] - 28s 368ms/step - loss: 0.1196 - accuracy: 0.9567 - val_loss: 0.1710 - val_accuracy: 0.9350
```

```
Epoch 80/100
75/75 [==============================] - 27s 358ms/step - loss: 0.1342 - accuracy: 0.9538 - val_loss: 0.1417 - val_accuracy: 0.9517
Epoch 81/100
75/75 [==============================] - 28s 370ms/step - loss: 0.1019 - accuracy: 0.9658 - val_loss: 0.1390 - val_accuracy: 0.9567
Epoch 82/100
75/75 [==============================] - 27s 361ms/step - loss: 0.1019 - accuracy: 0.9683 - val_loss: 0.1354 - val_accuracy: 0.9517
Epoch 83/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1079 - accuracy: 0.9592 - val_loss: 0.0826 - val_accuracy: 0.9750
Epoch 84/100
75/75 [==============================] - 28s 365ms/step - loss: 0.1188 - accuracy: 0.9588 - val_loss: 0.0973 - val_accuracy: 0.9650
Epoch 85/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1212 - accuracy: 0.9579 - val_loss: 0.1034 - val_accuracy: 0.9650
Epoch 86/100
75/75 [==============================] - 27s 365ms/step - loss: 0.1117 - accuracy: 0.9613 - val_loss: 0.1604 - val_accuracy: 0.9383
Epoch 87/100
75/75 [==============================] - 27s 359ms/step - loss: 0.1203 - accuracy: 0.9604 - val_loss: 0.1063 - val_accuracy: 0.9650
Epoch 88/100
75/75 [==============================] - 27s 365ms/step - loss: 0.1058 - accuracy: 0.9625 - val_loss: 0.2632 - val_accuracy: 0.9100
Epoch 89/100
75/75 [==============================] - 28s 367ms/step - loss: 0.1083 - accuracy: 0.9588 - val_loss: 0.0940 - val_accuracy: 0.9733
Epoch 90/100
75/75 [==============================] - 27s 365ms/step - loss: 0.0997 - accuracy: 0.9642 - val_loss: 0.0803 - val_accuracy: 0.9767
Epoch 91/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1230 - accuracy: 0.9567 - val_loss: 0.1227 - val_accuracy: 0.9617
Epoch 92/100
75/75 [==============================] - 27s 363ms/step - loss: 0.0974 - accuracy: 0.9633 - val_loss: 0.0870 - val_accuracy: 0.9667
Epoch 93/100
75/75 [==============================] - 27s 363ms/step - loss: 0.0998 - accuracy: 0.9596 - val_loss: 0.2843 - val_accuracy: 0.8983
Epoch 94/100
75/75 [==============================] - 27s 365ms/step - loss: 0.1038 - accuracy: 0.9633 - val_loss: 0.1304 - val_accuracy: 0.9650
Epoch 95/100
75/75 [==============================] - 27s 361ms/step - loss: 0.0857 - accuracy: 0.9704 - val_loss: 0.1467 - val_accuracy: 0.9650
Epoch 96/100
75/75 [==============================] - 28s 368ms/step - loss: 0.1112 - accuracy: 0.9571 - val_loss: 0.2999 - val_accuracy: 0.9083
Epoch 97/100
75/75 [==============================] - 28s 367ms/step - loss: 0.1034 - accuracy: 0.9633 - val_loss: 0.1271 - val_accuracy: 0.9567
Epoch 98/100
75/75 [==============================] - 28s 366ms/step - loss: 0.1019 - accuracy: 0.9633 - val_loss: 0.0926 - val_accuracy: 0.9717
Epoch 99/100
75/75 [==============================] - 28s 369ms/step - loss: 0.1002 - accuracy: 0.9642 - val_loss: 0.3702 - val_accuracy: 0.8783
Epoch 100/100
75/75 [==============================] - 27s 363ms/step - loss: 0.1146 - accuracy: 0.9625 - val_loss: 0.0909 - val_accuracy: 0.9683
```

In [83]:

```
plot_training_results(results100_augmented, model3)
```






```
19/19 [==============================] - 2s 76ms/step
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       310
           1       0.97      0.97      0.97       290

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600

Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_51 (Conv2D)          (None, 126, 126, 32)      896
```

```
 activation_85 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_51 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_52 (Conv2D)          (None, 61, 61, 32)        9248

 activation_86 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_52 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_53 (Conv2D)          (None, 28, 28, 64)        18496

 activation_87 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_53 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_17 (Flatten)        (None, 12544)             0

 dense_34 (Dense)            (None, 64)                802880

 activation_88 (Activation)  (None, 64)                0

 dropout_17 (Dropout)        (None, 64)                0

 dense_35 (Dense)            (None, 1)                 65

 activation_89 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

Adding the augmentation and removing the regularizer made matters worse. The model will be ran with both augmentation and regularizer to see if they tend to balance out.

Ver 4 - DATA AUGMENTATION with *Regularization*

Instantiate

In [36]:

```python
# INSIDE THE FUNCTION BELOW
```

Compile

In [87]:

```python
def create_model4():

    model = Sequential()

    # Create an instance of the ImageDataGenerator with desired augmentation parameters
    datagen = ImageDataGenerator(
        rotation_range=rotation_range,  # Randomly rotate images by 10 degrees
        width_shift_range=width_shift_range,  # Randomly shift images horizontally by 10% of the total width
        height_shift_range=height_shift_range,  # Randomly shift images vertically by 10% of the total height
        zoom_range=zoom_range,  # Randomly zoom images by 10%
        horizontal_flip=horizontal_flip  # Randomly flip images horizontally
    )

    # Apply data augmentation to the training data generator
    train_generator = datagen.flow(X_train, y_train, batch_size=batch_size)

    # Define and compile your model
    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3), kernel_regularizer=l2(0.001)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))
```

```
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
```

100 Epoch ***at 64 batch*** **with** ***Augmentation*** **and** ***Regularization***

In [88]:
```
# Exexcute the model
# Train the model using the augmented data generator
model4 = create_model4()
results100_augmented = model4.fit(train_generator, epochs=100, validation_data=(X_val, y_val))

# Save model
model4.save('model/brain_tumor_ver4_100_epochs_64_aug_reg.h5')
```

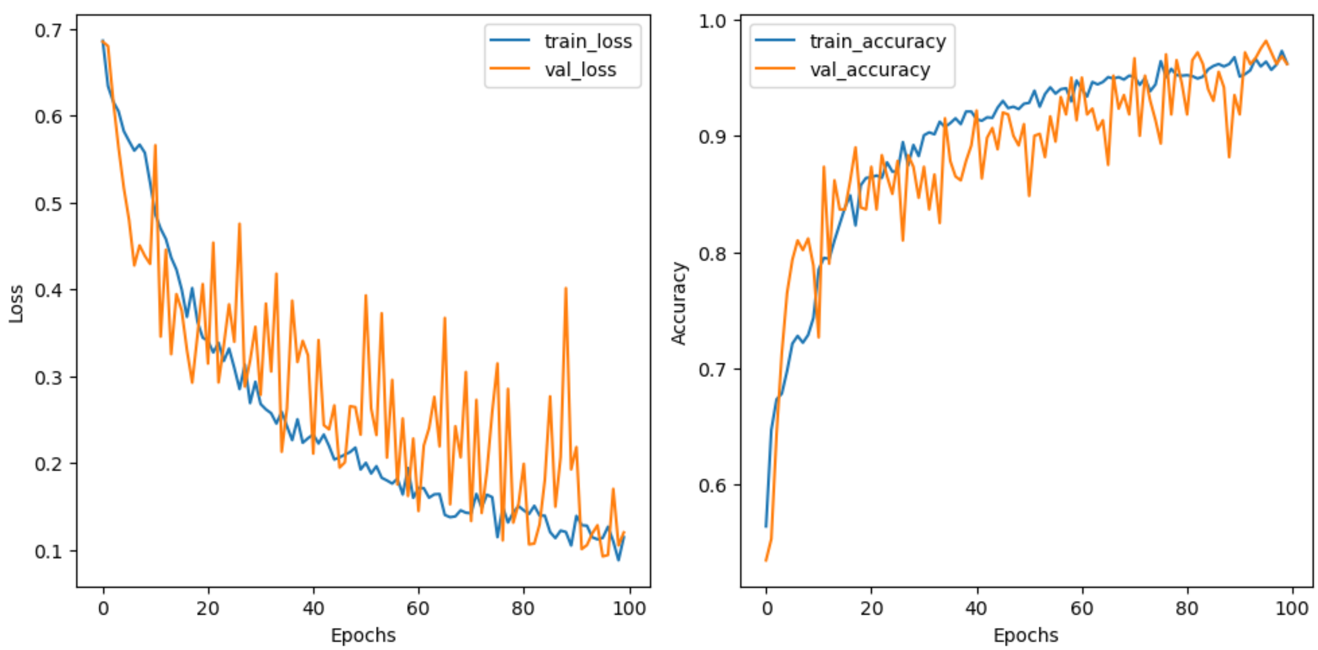
```
Epoch 1/100
75/75 [==============================] - 52s 688ms/step - loss: 0.6863 - accuracy: 0.5642 - val_loss: 0.6852 - val_accuracy: 0.5350
Epoch 2/100
75/75 [==============================] - 49s 648ms/step - loss: 0.6341 - accuracy: 0.6471 - val_loss: 0.6802 - val_accuracy: 0.5533
Epoch 3/100
75/75 [==============================] - 52s 694ms/step - loss: 0.6162 - accuracy: 0.6733 - val_loss: 0.6165 - val_accuracy: 0.6433
Epoch 4/100
75/75 [==============================] - 54s 723ms/step - loss: 0.6051 - accuracy: 0.6783 - val_loss: 0.5635 - val_accuracy: 0.7133
Epoch 5/100
75/75 [==============================] - 50s 665ms/step - loss: 0.5823 - accuracy: 0.6979 - val_loss: 0.5170 - val_accuracy: 0.7650
Epoch 6/100
75/75 [==============================] - 50s 665ms/step - loss: 0.5708 - accuracy: 0.7212 - val_loss: 0.4800 - val_accuracy: 0.7933
Epoch 7/100
75/75 [==============================] - 55s 737ms/step - loss: 0.5598 - accuracy: 0.7279 - val_loss: 0.4275 - val_accuracy: 0.8100
Epoch 8/100
75/75 [==============================] - 57s 756ms/step - loss: 0.5667 - accuracy: 0.7221 - val_loss: 0.4507 - val_accuracy: 0.8017
Epoch 9/100
75/75 [==============================] - 50s 670ms/step - loss: 0.5574 - accuracy: 0.7287 - val_loss: 0.4386 - val_accuracy: 0.8117
Epoch 10/100
75/75 [==============================] - 50s 665ms/step - loss: 0.5228 - accuracy: 0.7433 - val_loss: 0.4294 - val_accuracy: 0.7883
Epoch 11/100
75/75 [==============================] - 29s 382ms/step - loss: 0.4864 - accuracy: 0.7850 - val_loss: 0.5660 - val_accuracy: 0.7267
Epoch 12/100
75/75 [==============================] - 28s 367ms/step - loss: 0.4700 - accuracy: 0.7950 - val_loss: 0.3456 - val_accuracy: 0.8733
Epoch 13/100
75/75 [==============================] - 28s 378ms/step - loss: 0.4580 - accuracy: 0.7942 - val_loss: 0.4457 - val_accuracy: 0.7900
Epoch 14/100
75/75 [==============================] - 29s 382ms/step - loss: 0.4368 - accuracy: 0.8100 - val_loss: 0.3254 - val_accuracy: 0.8617
Epoch 15/100
75/75 [==============================] - 28s 377ms/step - loss: 0.4229 - accuracy: 0.8242 - val_loss: 0.3946 - val_accuracy: 0.8367
Epoch 16/100
75/75 [==============================] - 28s 368ms/step - loss: 0.3999 - accuracy: 0.8383 - val_loss: 0.3759 - val_accuracy: 0.8367
Epoch 17/100
75/75 [==============================] - 29s 384ms/step - loss: 0.3683 - accuracy: 0.8487 - val_loss: 0.3302 - val_accuracy: 0.8617
Epoch 18/100
75/75 [==============================] - 28s 378ms/step - loss: 0.4015 - accuracy: 0.8229 - val_loss: 0.2926 - val_accuracy: 0.8900
Epoch 19/100
75/75 [==============================] - 28s 372ms/step - loss: 0.3629 - accuracy: 0.8575 - val_loss: 0.3472 - val_accuracy: 0.8383
Epoch 20/100
75/75 [==============================] - 28s 370ms/step - loss: 0.3444 - accuracy: 0.8637 - val_loss: 0.4061 - val_accuracy: 0.8367
Epoch 21/100
75/75 [==============================] - 29s 379ms/step - loss: 0.3408 - accuracy: 0.8642 - val_loss: 0.3146 - val_accuracy: 0.8733
Epoch 22/100
75/75 [==============================] - 29s 381ms/step - loss: 0.3274 - accuracy: 0.8658 - val_loss: 0.4539 - val_accuracy: 0.8367
Epoch 23/100
75/75 [==============================] - 29s 381ms/step - loss: 0.3385 - accuracy: 0.8637 - val_loss: 0.2929 - val_accuracy: 0.8833
Epoch 24/100
75/75 [==============================] - 28s 377ms/step - loss: 0.3176 - accuracy: 0.8771 - val_loss: 0.3357 - val_accuracy: 0.8650
Epoch 25/100
75/75 [==============================] - 28s 378ms/step - loss: 0.3319 - accuracy: 0.8692 - val_loss: 0.3828 - val_accuracy: 0.8500
Epoch 26/100
75/75 [==============================] - 29s 379ms/step - loss: 0.3096 - accuracy: 0.8696 - val_loss: 0.3397 - val_accuracy: 0.8783
Epoch 27/100
75/75 [==============================] - 28s 378ms/step - loss: 0.2851 - accuracy: 0.8946 - val_loss: 0.4756 - val_accuracy: 0.8100
Epoch 28/100
75/75 [==============================] - 29s 379ms/step - loss: 0.3133 - accuracy: 0.8746 - val_loss: 0.2881 - val_accuracy: 0.8833
Epoch 29/100
75/75 [==============================] - 28s 379ms/step - loss: 0.2690 - accuracy: 0.8921 - val_loss: 0.3180 - val_accuracy: 0.8733
Epoch 30/100
75/75 [==============================] - 31s 406ms/step - loss: 0.2936 - accuracy: 0.8825 - val_loss: 0.3569 - val_accuracy: 0.8467
Epoch 31/100
75/75 [==============================] - 33s 443ms/step - loss: 0.2680 - accuracy: 0.9004 - val_loss: 0.2785 - val_accuracy: 0.8733
Epoch 32/100
75/75 [==============================] - 29s 388ms/step - loss: 0.2618 - accuracy: 0.9029 - val_loss: 0.3837 - val_accuracy: 0.8367
Epoch 33/100
75/75 [==============================] - 29s 385ms/step - loss: 0.2574 - accuracy: 0.9013 - val_loss: 0.3054 - val_accuracy: 0.8667
Epoch 34/100
75/75 [==============================] - 29s 390ms/step - loss: 0.2455 - accuracy: 0.9121 - val_loss: 0.4181 - val_accuracy: 0.8250
Epoch 35/100
75/75 [==============================] - 29s 385ms/step - loss: 0.2590 - accuracy: 0.9075 - val_loss: 0.2130 - val_accuracy: 0.9150
Epoch 36/100
75/75 [==============================] - 29s 384ms/step - loss: 0.2425 - accuracy: 0.9108 - val_loss: 0.2631 - val_accuracy: 0.8783
Epoch 37/100
75/75 [==============================] - 29s 381ms/step - loss: 0.2264 - accuracy: 0.9150 - val_loss: 0.3871 - val_accuracy: 0.8650
Epoch 38/100
75/75 [==============================] - 29s 384ms/step - loss: 0.2505 - accuracy: 0.9100 - val_loss: 0.3163 - val_accuracy: 0.8617
Epoch 39/100
```

```
75/75 [==============================] - 29s 383ms/step - loss: 0.2234 - accuracy: 0.9208 - val_loss: 0.3408 - val_accuracy: 0.8783
Epoch 40/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2286 - accuracy: 0.9208 - val_loss: 0.3242 - val_accuracy: 0.8917
Epoch 41/100
75/75 [==============================] - 29s 382ms/step - loss: 0.2331 - accuracy: 0.9142 - val_loss: 0.2109 - val_accuracy: 0.9217
Epoch 42/100
75/75 [==============================] - 29s 389ms/step - loss: 0.2226 - accuracy: 0.9129 - val_loss: 0.3417 - val_accuracy: 0.8633
Epoch 43/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2328 - accuracy: 0.9158 - val_loss: 0.2437 - val_accuracy: 0.8983
Epoch 44/100
75/75 [==============================] - 29s 384ms/step - loss: 0.2196 - accuracy: 0.9154 - val_loss: 0.2388 - val_accuracy: 0.9067
Epoch 45/100
75/75 [==============================] - 29s 384ms/step - loss: 0.2041 - accuracy: 0.9242 - val_loss: 0.2664 - val_accuracy: 0.8883
Epoch 46/100
75/75 [==============================] - 29s 383ms/step - loss: 0.2067 - accuracy: 0.9300 - val_loss: 0.1948 - val_accuracy: 0.9200
Epoch 47/100
75/75 [==============================] - 29s 381ms/step - loss: 0.2097 - accuracy: 0.9237 - val_loss: 0.2008 - val_accuracy: 0.9183
Epoch 48/100
75/75 [==============================] - 29s 384ms/step - loss: 0.2127 - accuracy: 0.9250 - val_loss: 0.2655 - val_accuracy: 0.9000
Epoch 49/100
75/75 [==============================] - 29s 383ms/step - loss: 0.2178 - accuracy: 0.9229 - val_loss: 0.2644 - val_accuracy: 0.8917
Epoch 50/100
75/75 [==============================] - 29s 385ms/step - loss: 0.1926 - accuracy: 0.9275 - val_loss: 0.2327 - val_accuracy: 0.9100
Epoch 51/100
75/75 [==============================] - 28s 378ms/step - loss: 0.2003 - accuracy: 0.9283 - val_loss: 0.3931 - val_accuracy: 0.8483
Epoch 52/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1880 - accuracy: 0.9388 - val_loss: 0.2629 - val_accuracy: 0.9000
Epoch 53/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1964 - accuracy: 0.9250 - val_loss: 0.2323 - val_accuracy: 0.9017
Epoch 54/100
75/75 [==============================] - 29s 383ms/step - loss: 0.1831 - accuracy: 0.9358 - val_loss: 0.3725 - val_accuracy: 0.8817
Epoch 55/100
75/75 [==============================] - 29s 382ms/step - loss: 0.1800 - accuracy: 0.9417 - val_loss: 0.2063 - val_accuracy: 0.9167
Epoch 56/100
75/75 [==============================] - 30s 398ms/step - loss: 0.1764 - accuracy: 0.9362 - val_loss: 0.2958 - val_accuracy: 0.8950
Epoch 57/100
75/75 [==============================] - 28s 378ms/step - loss: 0.1821 - accuracy: 0.9400 - val_loss: 0.1753 - val_accuracy: 0.9333
Epoch 58/100
75/75 [==============================] - 29s 383ms/step - loss: 0.1638 - accuracy: 0.9408 - val_loss: 0.2514 - val_accuracy: 0.9183
Epoch 59/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1943 - accuracy: 0.9296 - val_loss: 0.1621 - val_accuracy: 0.9500
Epoch 60/100
75/75 [==============================] - 29s 383ms/step - loss: 0.1600 - accuracy: 0.9475 - val_loss: 0.2282 - val_accuracy: 0.9133
Epoch 61/100
75/75 [==============================] - 29s 382ms/step - loss: 0.1715 - accuracy: 0.9400 - val_loss: 0.1448 - val_accuracy: 0.9500
Epoch 62/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1711 - accuracy: 0.9337 - val_loss: 0.2201 - val_accuracy: 0.9183
Epoch 63/100
75/75 [==============================] - 29s 390ms/step - loss: 0.1601 - accuracy: 0.9463 - val_loss: 0.2401 - val_accuracy: 0.9233
Epoch 64/100
75/75 [==============================] - 29s 382ms/step - loss: 0.1640 - accuracy: 0.9442 - val_loss: 0.2762 - val_accuracy: 0.9050
Epoch 65/100
75/75 [==============================] - 29s 382ms/step - loss: 0.1644 - accuracy: 0.9463 - val_loss: 0.2193 - val_accuracy: 0.9133
Epoch 66/100
75/75 [==============================] - 29s 390ms/step - loss: 0.1403 - accuracy: 0.9504 - val_loss: 0.3672 - val_accuracy: 0.8750
Epoch 67/100
75/75 [==============================] - 29s 391ms/step - loss: 0.1377 - accuracy: 0.9492 - val_loss: 0.1525 - val_accuracy: 0.9517
Epoch 68/100
75/75 [==============================] - 29s 384ms/step - loss: 0.1386 - accuracy: 0.9504 - val_loss: 0.2424 - val_accuracy: 0.9233
Epoch 69/100
75/75 [==============================] - 29s 388ms/step - loss: 0.1456 - accuracy: 0.9483 - val_loss: 0.2067 - val_accuracy: 0.9350
Epoch 70/100
75/75 [==============================] - 29s 386ms/step - loss: 0.1428 - accuracy: 0.9517 - val_loss: 0.3049 - val_accuracy: 0.9183
Epoch 71/100
75/75 [==============================] - 30s 401ms/step - loss: 0.1422 - accuracy: 0.9508 - val_loss: 0.1333 - val_accuracy: 0.9667
Epoch 72/100
75/75 [==============================] - 29s 389ms/step - loss: 0.1645 - accuracy: 0.9438 - val_loss: 0.2727 - val_accuracy: 0.9000
Epoch 73/100
75/75 [==============================] - 31s 409ms/step - loss: 0.1486 - accuracy: 0.9496 - val_loss: 0.1425 - val_accuracy: 0.9517
Epoch 74/100
75/75 [==============================] - 30s 393ms/step - loss: 0.1636 - accuracy: 0.9383 - val_loss: 0.1909 - val_accuracy: 0.9300
Epoch 75/100
75/75 [==============================] - 29s 391ms/step - loss: 0.1607 - accuracy: 0.9442 - val_loss: 0.2589 - val_accuracy: 0.9133
Epoch 76/100
75/75 [==============================] - 30s 401ms/step - loss: 0.1147 - accuracy: 0.9642 - val_loss: 0.3147 - val_accuracy: 0.8933
Epoch 77/100
75/75 [==============================] - 31s 419ms/step - loss: 0.1495 - accuracy: 0.9504 - val_loss: 0.1110 - val_accuracy: 0.9700
Epoch 78/100
75/75 [==============================] - 31s 418ms/step - loss: 0.1316 - accuracy: 0.9575 - val_loss: 0.2855 - val_accuracy: 0.9183
Epoch 79/100
75/75 [==============================] - 34s 457ms/step - loss: 0.1430 - accuracy: 0.9521 - val_loss: 0.1314 - val_accuracy: 0.9650
Epoch 80/100
75/75 [==============================] - 35s 456ms/step - loss: 0.1504 - accuracy: 0.9517 - val_loss: 0.1530 - val_accuracy: 0.9450
Epoch 81/100
75/75 [==============================] - 30s 405ms/step - loss: 0.1455 - accuracy: 0.9521 - val_loss: 0.1991 - val_accuracy: 0.9183
Epoch 82/100
75/75 [==============================] - 30s 398ms/step - loss: 0.1414 - accuracy: 0.9513 - val_loss: 0.1063 - val_accuracy: 0.9650
Epoch 83/100
75/75 [==============================] - 30s 404ms/step - loss: 0.1508 - accuracy: 0.9492 - val_loss: 0.1073 - val_accuracy: 0.9717
Epoch 84/100
75/75 [==============================] - 32s 423ms/step - loss: 0.1396 - accuracy: 0.9508 - val_loss: 0.1293 - val_accuracy: 0.9617
Epoch 85/100
75/75 [==============================] - 33s 438ms/step - loss: 0.1394 - accuracy: 0.9571 - val_loss: 0.1806 - val_accuracy: 0.9400
```

```
Epoch 86/100
75/75 [==============================] - 35s 471ms/step - loss: 0.1206 - accuracy: 0.9600 - val_loss: 0.2768 - val_accuracy: 0.9300
Epoch 87/100
75/75 [==============================] - 34s 457ms/step - loss: 0.1136 - accuracy: 0.9617 - val_loss: 0.1497 - val_accuracy: 0.9550
Epoch 88/100
75/75 [==============================] - 35s 462ms/step - loss: 0.1222 - accuracy: 0.9596 - val_loss: 0.2072 - val_accuracy: 0.9417
Epoch 89/100
75/75 [==============================] - 39s 513ms/step - loss: 0.1206 - accuracy: 0.9617 - val_loss: 0.4015 - val_accuracy: 0.8817
Epoch 90/100
75/75 [==============================] - 37s 493ms/step - loss: 0.1051 - accuracy: 0.9675 - val_loss: 0.1927 - val_accuracy: 0.9350
Epoch 91/100
75/75 [==============================] - 35s 467ms/step - loss: 0.1392 - accuracy: 0.9508 - val_loss: 0.2184 - val_accuracy: 0.9183
Epoch 92/100
75/75 [==============================] - 36s 474ms/step - loss: 0.1285 - accuracy: 0.9529 - val_loss: 0.1009 - val_accuracy: 0.9717
Epoch 93/100
75/75 [==============================] - 36s 485ms/step - loss: 0.1279 - accuracy: 0.9563 - val_loss: 0.1055 - val_accuracy: 0.9617
Epoch 94/100
75/75 [==============================] - 35s 464ms/step - loss: 0.1146 - accuracy: 0.9663 - val_loss: 0.1188 - val_accuracy: 0.9667
Epoch 95/100
75/75 [==============================] - 67s 901ms/step - loss: 0.1122 - accuracy: 0.9596 - val_loss: 0.1284 - val_accuracy: 0.9750
Epoch 96/100
75/75 [==============================] - 71s 952ms/step - loss: 0.1138 - accuracy: 0.9638 - val_loss: 0.0925 - val_accuracy: 0.9817
Epoch 97/100
75/75 [==============================] - 73s 970ms/step - loss: 0.1267 - accuracy: 0.9567 - val_loss: 0.0941 - val_accuracy: 0.9717
Epoch 98/100
75/75 [==============================] - 72s 956ms/step - loss: 0.1099 - accuracy: 0.9613 - val_loss: 0.1703 - val_accuracy: 0.9617
Epoch 99/100
75/75 [==============================] - 53s 712ms/step - loss: 0.0881 - accuracy: 0.9729 - val_loss: 0.1051 - val_accuracy: 0.9683
Epoch 100/100
75/75 [==============================] - 57s 763ms/step - loss: 0.1147 - accuracy: 0.9621 - val_loss: 0.1201 - val_accuracy: 0.9617
```

In [90]:

```
plot_training_results(results100_augmented, model4)
```




```
19/19 [==============================] - 4s 208ms/step
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       310
           1       0.94      0.98      0.96       290

    accuracy                           0.96       600
   macro avg       0.96      0.96      0.96       600
weighted avg       0.96      0.96      0.96       600

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_54 (Conv2D)          (None, 126, 126, 32)      896

 activation_90 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_54 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 conv2d_55 (Conv2D)          (None, 61, 61, 32)        9248

 activation_91 (Activation)  (None, 61, 61, 32)        0

 max_pooling2d_55 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)
```

```
 conv2d_56 (Conv2D)          (None, 28, 28, 64)        18496

 activation_92 (Activation)  (None, 28, 28, 64)        0

 max_pooling2d_56 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_18 (Flatten)        (None, 12544)             0

 dense_36 (Dense)            (None, 64)                802880

 activation_93 (Activation)  (None, 64)                0

 dropout_18 (Dropout)        (None, 64)                0

 dense_37 (Dense)            (None, 1)                 65

 activation_94 (Activation)  (None, 1)                 0

=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

In [106]:

```
test_loss, test_acc = model4.evaluate(X_val, y_val)
model_acc_loss(test_acc, test_loss)
```

```
19/19 [==============================] - 2s 78ms/step - loss: 0.1201 - accuracy: 0.9617
Model Accuracy (Test data)

________________________________________

Model Accuracy:       0.9616666436195374
Test Loss:           0.12013471126556396
________________________________________
```

Adding the Regularizer and the data augmentation parameters didnt really help much but it tended to create more spikes. To combat this, a learning scheduler will be applied to see if it can reduce the learning rate, and help reduce volatility.

Ver 5 - ADD LEARNING RATE SCHEDULER with *Regularization & Augmentation*

Instantiate

In [50]:

```
# INSIDE THE FUNCTION BELOW
```

Compile

In [102]:

```
def create_model5():

    #Instantiate model
    model = Sequential()

    # Create an instance of the ImageDataGenerator with desired augmentation parameters
    datagen = ImageDataGenerator(
        rotation_range=rotation_range,  # Randomly rotate images by 10 degrees
        width_shift_range=width_shift_range,  # Randomly shift images horizontally by 10% of the total width
        height_shift_range=height_shift_range,  # Randomly shift images vertically by 10% of the total height
        zoom_range=zoom_range,  # Randomly zoom images by 10%
        horizontal_flip=horizontal_flip  # Randomly flip images horizontally
    )

    # Apply data augmentation to the training data generator
    train_generator = datagen.flow(X_train, y_train, batch_size=batch_size)

    # Define and compile your model
    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3), kernel_regularizer=l2(0.001)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
```

```
    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    # Define the learning rate scheduler callback
    lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, verbose=1)


    return model, lr_scheduler, train_generator
```

100 Epoch (LR_Scheduler) ***at 64 batch*** **with** ***Augmentation*** **and** ***Regularization***

In [104]:

```
# Exexcute the model
# Train the model with the learning rate scheduler
model5, lr_scheduler, train_generator = create_model5()
results100_ver5 = model5.fit(train_generator, epochs=100, validation_data=(X_val, y_val), callbacks=[lr_scheduler])

# Save model
model5.save('model/brain_tumor_ver5_100_epochs_64_aug_reg_LR_sched.h5')
```

```
Epoch 1/100
75/75 [==============================] - 52s 686ms/step - loss: 0.6751 - accuracy: 0.5875 - val_loss: 0.6493 - val_accuracy: 0.6033 - l
r: 0.0010
Epoch 2/100
75/75 [==============================] - 51s 676ms/step - loss: 0.6377 - accuracy: 0.6400 - val_loss: 0.6188 - val_accuracy: 0.6383 - l
r: 0.0010
Epoch 3/100
75/75 [==============================] - 51s 674ms/step - loss: 0.6171 - accuracy: 0.6654 - val_loss: 0.5981 - val_accuracy: 0.6783 - l
r: 0.0010
Epoch 4/100
75/75 [==============================] - 57s 754ms/step - loss: 0.5950 - accuracy: 0.6933 - val_loss: 0.5465 - val_accuracy: 0.7317 - l
r: 0.0010
Epoch 5/100
75/75 [==============================] - 54s 724ms/step - loss: 0.5823 - accuracy: 0.6988 - val_loss: 0.5214 - val_accuracy: 0.7700 - l
r: 0.0010
Epoch 6/100
75/75 [==============================] - 53s 701ms/step - loss: 0.5623 - accuracy: 0.7217 - val_loss: 0.5789 - val_accuracy: 0.7167 - l
r: 0.0010
Epoch 7/100
75/75 [==============================] - 58s 764ms/step - loss: 0.5423 - accuracy: 0.7392 - val_loss: 0.4405 - val_accuracy: 0.7917 - l
r: 0.0010
Epoch 8/100
75/75 [==============================] - 49s 654ms/step - loss: 0.5184 - accuracy: 0.7513 - val_loss: 0.3967 - val_accuracy: 0.8167 - l
r: 0.0010
Epoch 9/100
75/75 [==============================] - 37s 496ms/step - loss: 0.4955 - accuracy: 0.7754 - val_loss: 0.3782 - val_accuracy: 0.8333 - l
r: 0.0010
Epoch 10/100
75/75 [==============================] - 28s 362ms/step - loss: 0.4866 - accuracy: 0.7763 - val_loss: 0.3400 - val_accuracy: 0.8500 - l
r: 0.0010
Epoch 11/100
75/75 [==============================] - 27s 355ms/step - loss: 0.4447 - accuracy: 0.8033 - val_loss: 0.4806 - val_accuracy: 0.7700 - l
r: 0.0010
Epoch 12/100
75/75 [==============================] - 26s 352ms/step - loss: 0.4229 - accuracy: 0.8029 - val_loss: 0.3510 - val_accuracy: 0.8467 - l
r: 0.0010
Epoch 13/100
75/75 [==============================] - 27s 354ms/step - loss: 0.3941 - accuracy: 0.8300 - val_loss: 0.3456 - val_accuracy: 0.8467 - l
r: 0.0010
Epoch 14/100
75/75 [==============================] - 27s 353ms/step - loss: 0.3847 - accuracy: 0.8338 - val_loss: 0.4958 - val_accuracy: 0.7817 - l
r: 0.0010
Epoch 15/100
75/75 [==============================] - 26s 351ms/step - loss: 0.3781 - accuracy: 0.8358 - val_loss: 0.3309 - val_accuracy: 0.8567 - l
r: 0.0010
Epoch 16/100
75/75 [==============================] - 27s 353ms/step - loss: 0.3310 - accuracy: 0.8679 - val_loss: 0.2989 - val_accuracy: 0.8750 - l
r: 0.0010
Epoch 17/100
75/75 [==============================] - 26s 350ms/step - loss: 0.3300 - accuracy: 0.8675 - val_loss: 0.3612 - val_accuracy: 0.8483 - l
r: 0.0010
Epoch 18/100
75/75 [==============================] - 27s 355ms/step - loss: 0.3322 - accuracy: 0.8662 - val_loss: 0.3073 - val_accuracy: 0.8817 - l
r: 0.0010
Epoch 19/100
75/75 [==============================] - 26s 350ms/step - loss: 0.3408 - accuracy: 0.8542 - val_loss: 0.2662 - val_accuracy: 0.8933 - l
r: 0.0010
Epoch 20/100
75/75 [==============================] - 27s 353ms/step - loss: 0.3032 - accuracy: 0.8725 - val_loss: 0.3670 - val_accuracy: 0.8500 - l
r: 0.0010
Epoch 21/100
75/75 [==============================] - 27s 356ms/step - loss: 0.3007 - accuracy: 0.8804 - val_loss: 0.3450 - val_accuracy: 0.8617 - l
r: 0.0010
Epoch 22/100
75/75 [==============================] - 27s 353ms/step - loss: 0.2902 - accuracy: 0.8896 - val_loss: 0.3464 - val_accuracy: 0.8633 - l
r: 0.0010
Epoch 23/100
75/75 [==============================] - 27s 353ms/step - loss: 0.2751 - accuracy: 0.8967 - val_loss: 0.3931 - val_accuracy: 0.8300 - l
r: 0.0010
```

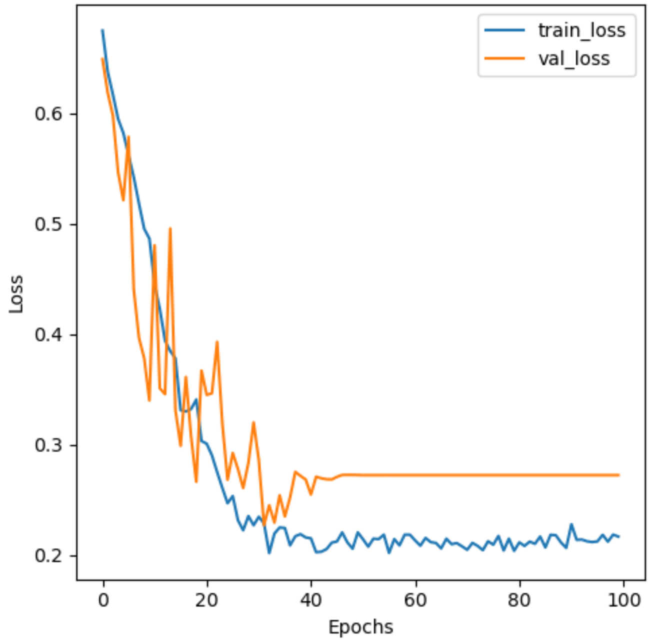
```
Epoch 24/100
75/75 [==============================] - ETA: 0s - loss: 0.2607 - accuracy: 0.8942
Epoch 24: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
75/75 [==============================] - 27s 353ms/step - loss: 0.2607 - accuracy: 0.8942 - val_loss: 0.3189 - val_accuracy: 0.8767 - l
r: 0.0010
Epoch 25/100
75/75 [==============================] - 27s 357ms/step - loss: 0.2468 - accuracy: 0.9067 - val_loss: 0.2681 - val_accuracy: 0.8917 - l
r: 1.0000e-04
Epoch 26/100
75/75 [==============================] - 27s 353ms/step - loss: 0.2533 - accuracy: 0.9108 - val_loss: 0.2923 - val_accuracy: 0.8717 - l
r: 1.0000e-04
Epoch 27/100
75/75 [==============================] - 27s 355ms/step - loss: 0.2314 - accuracy: 0.9125 - val_loss: 0.2777 - val_accuracy: 0.8767 - l
r: 1.0000e-04
Epoch 28/100
75/75 [==============================] - 26s 350ms/step - loss: 0.2223 - accuracy: 0.9150 - val_loss: 0.2606 - val_accuracy: 0.8867 - l
r: 1.0000e-04
Epoch 29/100
75/75 [==============================] - 27s 364ms/step - loss: 0.2352 - accuracy: 0.9121 - val_loss: 0.2834 - val_accuracy: 0.8750 - l
r: 1.0000e-04
Epoch 30/100
75/75 [==============================] - 27s 359ms/step - loss: 0.2268 - accuracy: 0.9150 - val_loss: 0.3200 - val_accuracy: 0.8600 - l
r: 1.0000e-04
Epoch 31/100
75/75 [==============================] - 27s 355ms/step - loss: 0.2345 - accuracy: 0.9117 - val_loss: 0.2859 - val_accuracy: 0.8650 - l
r: 1.0000e-04
Epoch 32/100
75/75 [==============================] - 27s 357ms/step - loss: 0.2285 - accuracy: 0.9154 - val_loss: 0.2268 - val_accuracy: 0.9217 - l
r: 1.0000e-04
Epoch 33/100
75/75 [==============================] - 27s 355ms/step - loss: 0.2016 - accuracy: 0.9221 - val_loss: 0.2450 - val_accuracy: 0.8933 - l
r: 1.0000e-04
Epoch 34/100
75/75 [==============================] - 1566s 21s/step - loss: 0.2193 - accuracy: 0.9212 - val_loss: 0.2292 - val_accuracy: 0.9217 - l
r: 1.0000e-04
Epoch 35/100
75/75 [==============================] - 25s 332ms/step - loss: 0.2248 - accuracy: 0.9162 - val_loss: 0.2540 - val_accuracy: 0.8850 - l
r: 1.0000e-04
Epoch 36/100
75/75 [==============================] - 25s 337ms/step - loss: 0.2243 - accuracy: 0.9258 - val_loss: 0.2348 - val_accuracy: 0.9050 - l
r: 1.0000e-04
Epoch 37/100
75/75 [==============================] - ETA: 0s - loss: 0.2086 - accuracy: 0.9250
Epoch 37: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
75/75 [==============================] - 25s 337ms/step - loss: 0.2086 - accuracy: 0.9250 - val_loss: 0.2520 - val_accuracy: 0.8917 - l
r: 1.0000e-04
Epoch 38/100
75/75 [==============================] - 26s 344ms/step - loss: 0.2169 - accuracy: 0.9200 - val_loss: 0.2752 - val_accuracy: 0.8683 - l
r: 1.0000e-05
Epoch 39/100
75/75 [==============================] - 26s 350ms/step - loss: 0.2188 - accuracy: 0.9229 - val_loss: 0.2716 - val_accuracy: 0.8733 - l
r: 1.0000e-05
Epoch 40/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2158 - accuracy: 0.9217 - val_loss: 0.2683 - val_accuracy: 0.8750 - l
r: 1.0000e-05
Epoch 41/100
75/75 [==============================] - 28s 378ms/step - loss: 0.2150 - accuracy: 0.9204 - val_loss: 0.2547 - val_accuracy: 0.8883 - l
r: 1.0000e-05
Epoch 42/100
75/75 [==============================] - ETA: 0s - loss: 0.2024 - accuracy: 0.9262
Epoch 42: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
75/75 [==============================] - 28s 366ms/step - loss: 0.2024 - accuracy: 0.9262 - val_loss: 0.2708 - val_accuracy: 0.8733 - l
r: 1.0000e-05
Epoch 43/100
75/75 [==============================] - 28s 368ms/step - loss: 0.2029 - accuracy: 0.9237 - val_loss: 0.2694 - val_accuracy: 0.8750 - l
r: 1.0000e-06
Epoch 44/100
75/75 [==============================] - 28s 366ms/step - loss: 0.2053 - accuracy: 0.9300 - val_loss: 0.2685 - val_accuracy: 0.8750 - l
r: 1.0000e-06
Epoch 45/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2111 - accuracy: 0.9208 - val_loss: 0.2684 - val_accuracy: 0.8750 - l
r: 1.0000e-06
Epoch 46/100
75/75 [==============================] - 27s 364ms/step - loss: 0.2122 - accuracy: 0.9196 - val_loss: 0.2707 - val_accuracy: 0.8733 - l
r: 1.0000e-06
Epoch 47/100
75/75 [==============================] - ETA: 0s - loss: 0.2203 - accuracy: 0.9187
Epoch 47: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
75/75 [==============================] - 27s 364ms/step - loss: 0.2203 - accuracy: 0.9187 - val_loss: 0.2724 - val_accuracy: 0.8750 - l
r: 1.0000e-06
Epoch 48/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2113 - accuracy: 0.9208 - val_loss: 0.2725 - val_accuracy: 0.8750 - l
r: 1.0000e-07
Epoch 49/100
75/75 [==============================] - 27s 362ms/step - loss: 0.2056 - accuracy: 0.9287 - val_loss: 0.2725 - val_accuracy: 0.8750 - l
r: 1.0000e-07
Epoch 50/100
75/75 [==============================] - 28s 366ms/step - loss: 0.2204 - accuracy: 0.9183 - val_loss: 0.2724 - val_accuracy: 0.8750 - l
r: 1.0000e-07
Epoch 51/100
75/75 [==============================] - 27s 363ms/step - loss: 0.2142 - accuracy: 0.9204 - val_loss: 0.2722 - val_accuracy: 0.8733 - l
r: 1.0000e-07
Epoch 52/100
```

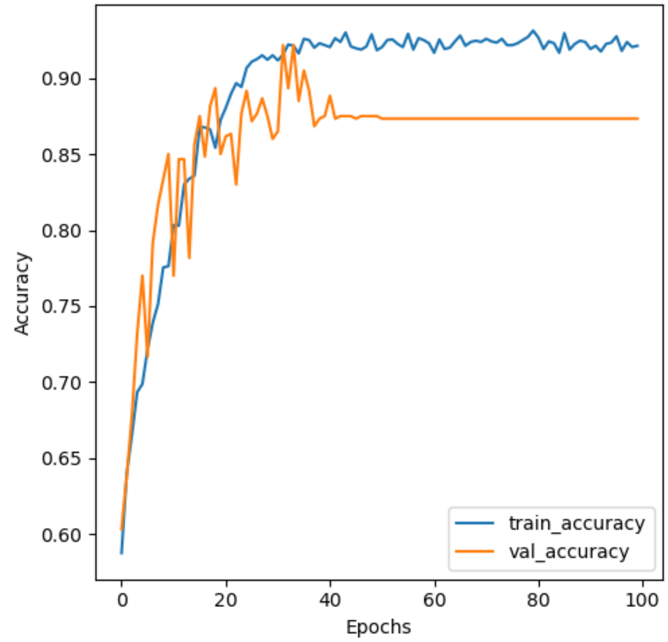
```
75/75 [==============================] - ETA: 0s - loss: 0.2075 - accuracy: 0.9250
Epoch 52: ReduceLROnPlateau reducing learning rate to 1.0000000827403710e-08.
75/75 [==============================] - 28s 370ms/step - loss: 0.2075 - accuracy: 0.9250 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-07
Epoch 53/100
75/75 [==============================] - 28s 373ms/step - loss: 0.2147 - accuracy: 0.9254 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-08
Epoch 54/100
75/75 [==============================] - 31s 414ms/step - loss: 0.2143 - accuracy: 0.9225 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-08
Epoch 55/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2183 - accuracy: 0.9204 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-08
Epoch 56/100
75/75 [==============================] - 28s 376ms/step - loss: 0.2018 - accuracy: 0.9292 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-08
Epoch 57/100
75/75 [==============================] - ETA: 0s - loss: 0.2143 - accuracy: 0.9187
Epoch 57: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
75/75 [==============================] - 28s 368ms/step - loss: 0.2143 - accuracy: 0.9187 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-08
Epoch 58/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2086 - accuracy: 0.9262 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-09
Epoch 59/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2183 - accuracy: 0.9250 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-09
Epoch 60/100
75/75 [==============================] - 28s 377ms/step - loss: 0.2183 - accuracy: 0.9229 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-09
Epoch 61/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2131 - accuracy: 0.9167 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-09
Epoch 62/100
75/75 [==============================] - ETA: 0s - loss: 0.2083 - accuracy: 0.9254
Epoch 62: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
75/75 [==============================] - 28s 372ms/step - loss: 0.2083 - accuracy: 0.9254 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-09
Epoch 63/100
75/75 [==============================] - 28s 374ms/step - loss: 0.2153 - accuracy: 0.9192 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-10
Epoch 64/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2116 - accuracy: 0.9200 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-10
Epoch 65/100
75/75 [==============================] - 28s 371ms/step - loss: 0.2108 - accuracy: 0.9242 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-10
Epoch 66/100
75/75 [==============================] - 28s 375ms/step - loss: 0.2059 - accuracy: 0.9279 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-10
Epoch 67/100
75/75 [==============================] - ETA: 0s - loss: 0.2147 - accuracy: 0.9212
Epoch 67: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-11.
75/75 [==============================] - 30s 398ms/step - loss: 0.2147 - accuracy: 0.9212 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-10
Epoch 68/100
75/75 [==============================] - 28s 374ms/step - loss: 0.2097 - accuracy: 0.9237 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-11
Epoch 69/100
75/75 [==============================] - 29s 383ms/step - loss: 0.2107 - accuracy: 0.9246 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-11
Epoch 70/100
75/75 [==============================] - 28s 375ms/step - loss: 0.2077 - accuracy: 0.9237 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-11
Epoch 71/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2046 - accuracy: 0.9258 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-11
Epoch 72/100
75/75 [==============================] - ETA: 0s - loss: 0.2109 - accuracy: 0.9242
Epoch 72: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-12.
75/75 [==============================] - 28s 374ms/step - loss: 0.2109 - accuracy: 0.9242 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-11
Epoch 73/100
75/75 [==============================] - 29s 381ms/step - loss: 0.2081 - accuracy: 0.9233 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-12
Epoch 74/100
75/75 [==============================] - 28s 376ms/step - loss: 0.2043 - accuracy: 0.9258 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-12
Epoch 75/100
75/75 [==============================] - 28s 379ms/step - loss: 0.2123 - accuracy: 0.9217 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-12
Epoch 76/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2092 - accuracy: 0.9217 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-12
Epoch 77/100
75/75 [==============================] - ETA: 0s - loss: 0.2171 - accuracy: 0.9229
Epoch 77: ReduceLROnPlateau reducing learning rate to 1.0000001044244145e-13.
75/75 [==============================] - 29s 383ms/step - loss: 0.2171 - accuracy: 0.9229 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-12
Epoch 78/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2040 - accuracy: 0.9250 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-13
Epoch 79/100
```

```
75/75 [==============================] - 29s 380ms/step - loss: 0.2147 - accuracy: 0.9271 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-13
Epoch 80/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2037 - accuracy: 0.9312 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-13
Epoch 81/100
75/75 [==============================] - 28s 375ms/step - loss: 0.2113 - accuracy: 0.9267 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-13
Epoch 82/100
75/75 [==============================] - ETA: 0s - loss: 0.2081 - accuracy: 0.9192
Epoch 82: ReduceLROnPlateau reducing learning rate to 1.0000001179769417e-14.
75/75 [==============================] - 28s 376ms/step - loss: 0.2081 - accuracy: 0.9192 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-13
Epoch 83/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2120 - accuracy: 0.9242 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-14
Epoch 84/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2100 - accuracy: 0.9229 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-14
Epoch 85/100
75/75 [==============================] - 28s 378ms/step - loss: 0.2166 - accuracy: 0.9167 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-14
Epoch 86/100
75/75 [==============================] - 29s 389ms/step - loss: 0.2067 - accuracy: 0.9296 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-14
Epoch 87/100
75/75 [==============================] - ETA: 0s - loss: 0.2182 - accuracy: 0.9187
Epoch 87: ReduceLROnPlateau reducing learning rate to 1.0000001518582595e-15.
75/75 [==============================] - 28s 378ms/step - loss: 0.2182 - accuracy: 0.9187 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-14
Epoch 88/100
75/75 [==============================] - 30s 393ms/step - loss: 0.2178 - accuracy: 0.9225 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-15
Epoch 89/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2113 - accuracy: 0.9246 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-15
Epoch 90/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2063 - accuracy: 0.9237 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-15
Epoch 91/100
75/75 [==============================] - 28s 374ms/step - loss: 0.2278 - accuracy: 0.9192 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-15
Epoch 92/100
75/75 [==============================] - ETA: 0s - loss: 0.2137 - accuracy: 0.9212
Epoch 92: ReduceLROnPlateau reducing learning rate to 1.0000001095066122e-16.
75/75 [==============================] - 28s 376ms/step - loss: 0.2137 - accuracy: 0.9212 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-15
Epoch 93/100
75/75 [==============================] - 28s 374ms/step - loss: 0.2139 - accuracy: 0.9175 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-16
Epoch 94/100
75/75 [==============================] - 28s 375ms/step - loss: 0.2123 - accuracy: 0.9225 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-16
Epoch 95/100
75/75 [==============================] - 28s 378ms/step - loss: 0.2116 - accuracy: 0.9233 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-16
Epoch 96/100
75/75 [==============================] - 28s 377ms/step - loss: 0.2122 - accuracy: 0.9275 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-16
Epoch 97/100
75/75 [==============================] - ETA: 0s - loss: 0.2182 - accuracy: 0.9179
Epoch 97: ReduceLROnPlateau reducing learning rate to 1.0000000830368326e-17.
75/75 [==============================] - 28s 377ms/step - loss: 0.2182 - accuracy: 0.9179 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-16
Epoch 98/100
75/75 [==============================] - 29s 382ms/step - loss: 0.2120 - accuracy: 0.9237 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-17
Epoch 99/100
75/75 [==============================] - 29s 380ms/step - loss: 0.2183 - accuracy: 0.9204 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-17
Epoch 100/100
75/75 [==============================] - 29s 379ms/step - loss: 0.2166 - accuracy: 0.9212 - val_loss: 0.2723 - val_accuracy: 0.8733 - l
r: 1.0000e-17
```

In [108]:

```
plot_training_results(results100_ver5, model5)
```






```
19/19 [==============================] - 2s 83ms/step
              precision    recall  f1-score   support

           0       0.94      0.80      0.87       310
           1       0.82      0.95      0.88       290

    accuracy                           0.87       600
   macro avg       0.88      0.88      0.87       600
weighted avg       0.88      0.87      0.87       600

Model: "sequential_42"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_84 (Conv2D)          (None, 126, 126, 32)      896       
                                                                 
 activation_140 (Activation)  (None, 126, 126, 32)     0         
                                                                 
 max_pooling2d_84 (MaxPoolin  (None, 63, 63, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_85 (Conv2D)          (None, 61, 61, 32)        9248      
                                                                 
 activation_141 (Activation)  (None, 61, 61, 32)       0         
                                                                 
 max_pooling2d_85 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_86 (Conv2D)          (None, 28, 28, 64)        18496     
                                                                 
 activation_142 (Activation)  (None, 28, 28, 64)       0         
                                                                 
 max_pooling2d_86 (MaxPoolin  (None, 14, 14, 64)       0         
 g2D)                                                            
                                                                 
 flatten_28 (Flatten)        (None, 12544)             0         
                                                                 
 dense_56 (Dense)            (None, 64)                802880    
                                                                 
 activation_143 (Activation)  (None, 64)               0         
                                                                 
 dropout_28 (Dropout)        (None, 64)                0         
                                                                 
 dense_57 (Dense)            (None, 1)                 65        
                                                                 
 activation_144 (Activation)  (None, 1)                0         
                                                                 
=================================================================
Total params: 831,585
Trainable params: 831,585
Non-trainable params: 0
_________________________________________________________________
```

In [105]:

```
test_loss, test_acc = model5.evaluate(X_val, y_val)
model_acc_loss(test_acc, test_loss)
```

```
19/19 [==============================] - 1s 77ms/step - loss: 0.2723 - accuracy: 0.8733
Model Accuracy (Test data)
________________________________________
```

```
Model Accuracy:       0.8733333349227905
Test Loss:           0.2722611427307129
________________________________________
```

[[ANALYSIS]]

Ver 6 - LAYER REDUCTION back to the basics

Instantiate

In [72]:
```
# INSIDE THE FUNCTION BELOW
```

Compile

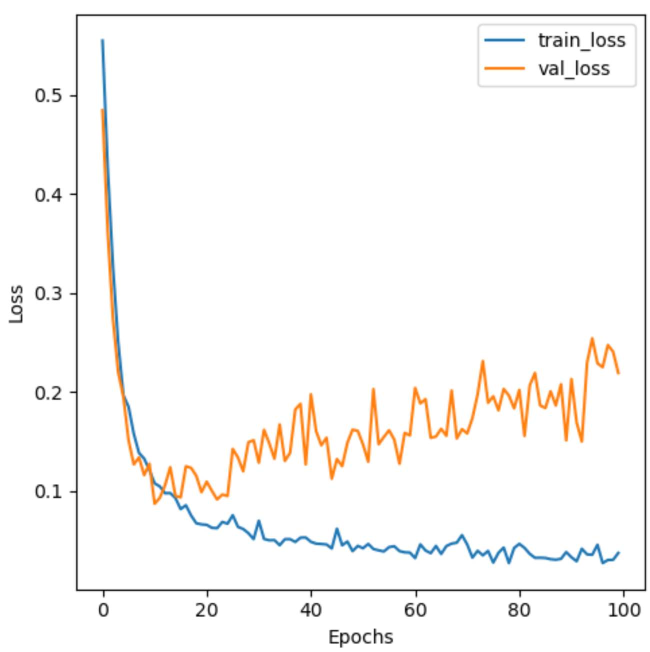
In [69]:
```
def create_model6():

    #Instantiate model
    model = Sequential()

    model.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(16, activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


    return model
```

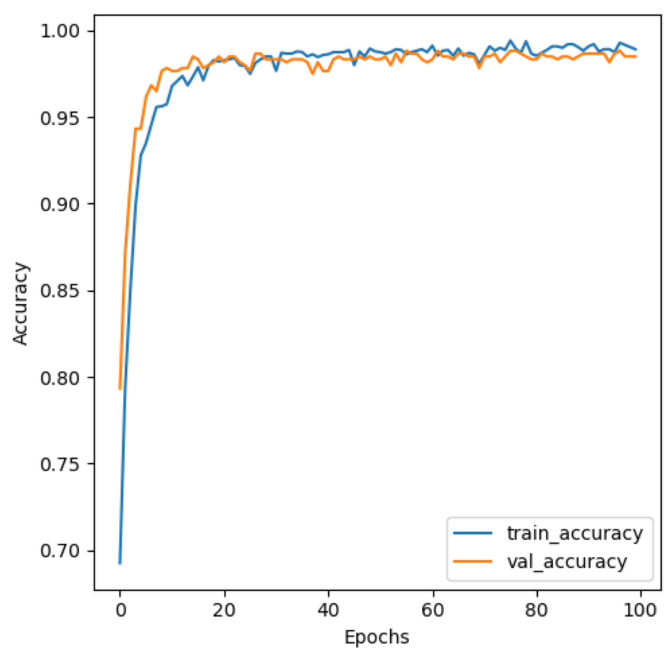
100 Epoch *at 64 batch* with minimal layers

Below, many different models were ran undergoing trial and error to find the most optimal model, and less complex, considering the avoidance of overfit models. Undergoing several hours testing, it was decided the model that seemed most appropriate visually when compared to the rest, resulting in 98% accuracy with no overfitting. The model does suffer from complexity, and that would be something to smoothen out moving forward.

***Trial and Error* (adjusting parameters to best fit the model)**

In [86]:
```
plot_training_results(results100_ver6_3, model7)
```



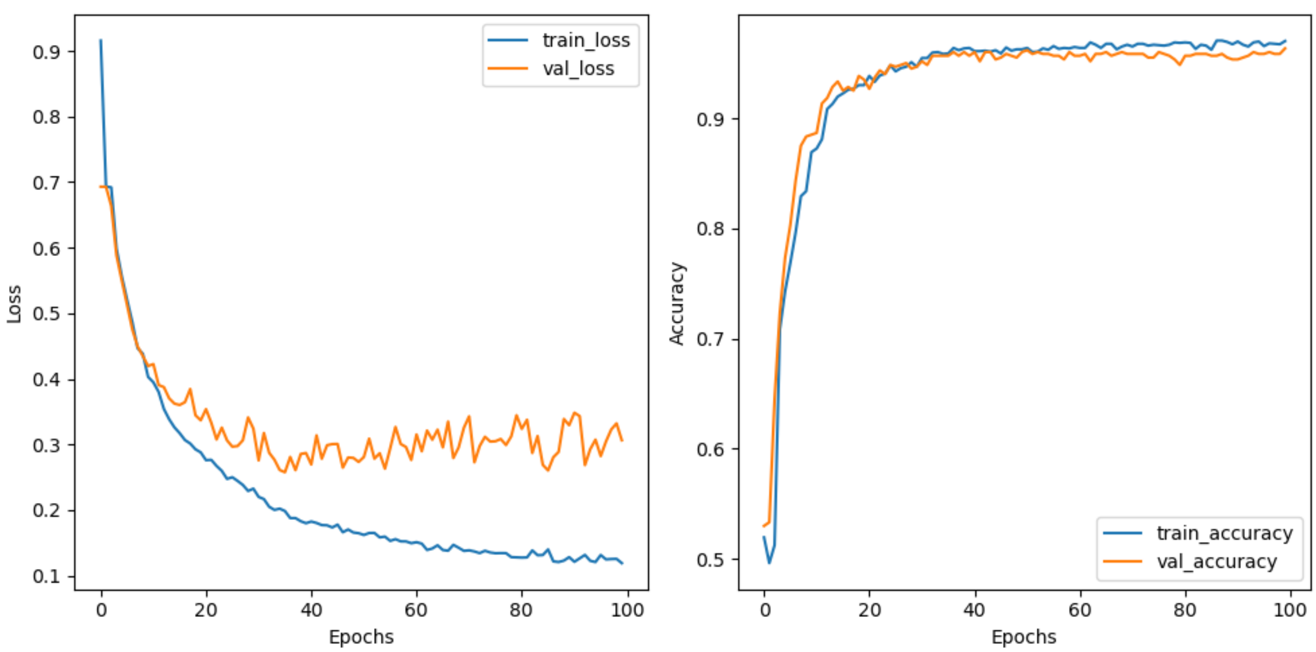



```
19/19 [==============================] - 1s 44ms/step
              precision    recall  f1-score   support
```

```
           0       0.98      0.99      0.98       285
           1       0.99      0.98      0.99       315

    accuracy                           0.98       600
   macro avg       0.98      0.99      0.98       600
weighted avg       0.99      0.98      0.99       600

Model: "sequential_36"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_68 (Conv2D)          (None, 126, 126, 16)      448       
                                                                 
 activation_104 (Activation)  (None, 126, 126, 16)     0         
                                                                 
 max_pooling2d_68 (MaxPoolin  (None, 63, 63, 16)       0         
 g2D)                                                            
                                                                 
 dropout_94 (Dropout)        (None, 63, 63, 16)        0         
                                                                 
 conv2d_69 (Conv2D)          (None, 61, 61, 32)        4640      
                                                                 
 activation_105 (Activation)  (None, 61, 61, 32)       0         
                                                                 
 max_pooling2d_69 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 dropout_95 (Dropout)        (None, 30, 30, 32)        0         
                                                                 
 flatten_36 (Flatten)        (None, 28800)             0         
                                                                 
 dense_72 (Dense)            (None, 16)                460816    
                                                                 
 dropout_96 (Dropout)        (None, 16)                0         
                                                                 
 dense_73 (Dense)            (None, 1)                 17        
                                                                 
 activation_106 (Activation)  (None, 1)                0         
                                                                 
=================================================================
Total params: 465,921
Trainable params: 465,921
Non-trainable params: 0
_________________________________________________________________
```

In [75]:

```
plot_training_results(results100_ver6_3, model6)
```

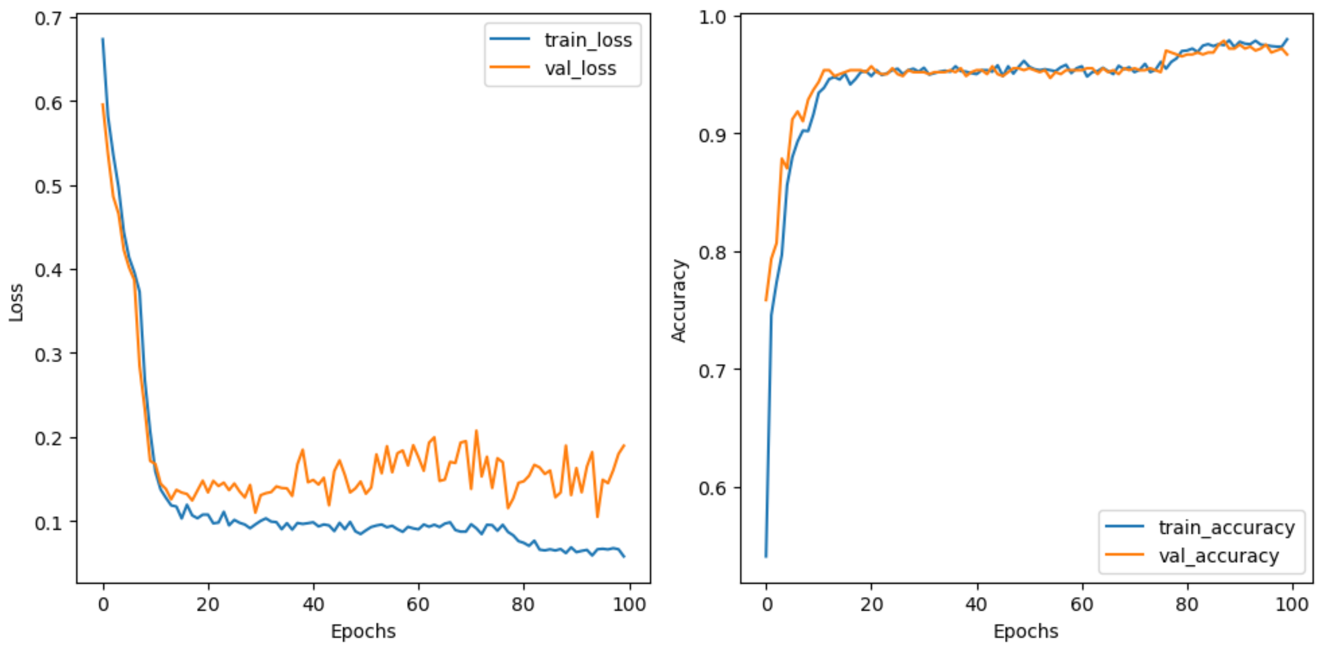



```
19/19 [==============================] - 2s 88ms/step
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       285
           1       0.95      0.98      0.97       315

    accuracy                           0.96       600
   macro avg       0.96      0.96      0.96       600
weighted avg       0.96      0.96      0.96       600

Model: "sequential_31"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
```

```
=================================================================
 conv2d_58 (Conv2D)          (None, 126, 126, 32)      896

 activation_89 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_58 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 dropout_79 (Dropout)        (None, 63, 63, 32)        0

 conv2d_59 (Conv2D)          (None, 61, 61, 64)        18496

 activation_90 (Activation)  (None, 61, 61, 64)        0

 max_pooling2d_59 (MaxPoolin  (None, 30, 30, 64)       0
 g2D)

 dropout_80 (Dropout)        (None, 30, 30, 64)        0

 flatten_31 (Flatten)        (None, 57600)             0

 dense_62 (Dense)            (None, 32)                1843232

 dropout_81 (Dropout)        (None, 32)                0

 dense_63 (Dense)            (None, 1)                 33

 activation_91 (Activation)  (None, 1)                 0

=================================================================
Total params: 1,862,657
Trainable params: 1,862,657
Non-trainable params: 0
_________________________________________________________________
```

In [118]:

```
plot_training_results(results100_ver6, model6)
```




```
19/19 [==============================] - 2s 83ms/step
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       310
           1       0.97      0.96      0.97       290

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600

Model: "sequential_45"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_89 (Conv2D)          (None, 126, 126, 64)      1792

 activation_149 (Activation)  (None, 126, 126, 64)     0

 max_pooling2d_89 (MaxPoolin  (None, 63, 63, 64)       0
 g2D)

 flatten_31 (Flatten)        (None, 254016)            0
```

```
 dense_62 (Dense)            (None, 16)                4064272

 dropout_31 (Dropout)        (None, 16)                0

 dense_63 (Dense)            (None, 1)                 17

 activation_150 (Activation)  (None, 1)                0

=================================================================
Total params: 4,066,081
Trainable params: 4,066,081
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

From the previous attempts to tune the model (adding more layers, augmentation, regularization) it all makes sense to do with a complex model, but since this model is simple, it wouldnt make sense to start at a high point, but start with less features to have a better understanding of why the loss is occuring. Having too much hyperparameters can make a simple model look complex. As we can see, with the reduction of layers and parameters, the model able to get closer to a more uniformed model that can properly train the model. Although its still not perfect, adding the necessary parameters to the model can fix the divergence happening in this version.

In [ ]:

Ver 7 - SIMPLE MODEL

Instantiate

In [167]:

```
#Instantiate model
model = Sequential()
```

Compile

In [168]:

```
# # Define and compile your model
# model.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3)))
# #model.add(Activation('relu'))
# model.add(MaxPooling2D(pool_size=(2,2)))

# model.add(Flatten())
# model.add(Dense(16, activation = 'relu'))
# model.add(Dropout(0.2))
# model.add(Dense(1))
# model.add(Activation('sigmoid'))

# model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


model = Sequential()
model.add(Conv2D(16, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3), activation='relu'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dropout(0.1))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
```

```
               metrics=['accuracy'])

# Define the learning rate scheduler callback
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, verbose=1)
```

100 Epoch

In [169]:

```
# Exexcute the model
results100 = model.fit(X_train, y_train, batch_size = 64, verbose=1, epochs=100,
                       validation_data= (X_val, y_val), callbacks=[lr_scheduler])

# Save model
model.save('model/brain_tumor_base_100_epochs_64_basics_v2.h5')
```

```
Epoch 1/100
38/38 [==============================] - 20s 515ms/step - loss: 0.5194 - accuracy: 0.7292 - val_loss: 0.3660 - val_accuracy: 0.8267 - l
r: 0.0010
Epoch 2/100
38/38 [==============================] - 20s 525ms/step - loss: 0.2372 - accuracy: 0.9179 - val_loss: 0.2043 - val_accuracy: 0.9417 - l
r: 0.0010
Epoch 3/100
38/38 [==============================] - 20s 526ms/step - loss: 0.1045 - accuracy: 0.9775 - val_loss: 0.1472 - val_accuracy: 0.9633 - l
r: 0.0010
Epoch 4/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0551 - accuracy: 0.9892 - val_loss: 0.1065 - val_accuracy: 0.9717 - l
r: 0.0010
Epoch 5/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0302 - accuracy: 0.9946 - val_loss: 0.0830 - val_accuracy: 0.9767 - l
r: 0.0010
Epoch 6/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0218 - accuracy: 0.9992 - val_loss: 0.1012 - val_accuracy: 0.9733 - l
r: 0.0010
Epoch 7/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0106 - accuracy: 1.0000 - val_loss: 0.0926 - val_accuracy: 0.9750 - l
r: 0.0010
Epoch 8/100
38/38 [==============================] - 20s 531ms/step - loss: 0.0066 - accuracy: 1.0000 - val_loss: 0.0830 - val_accuracy: 0.9750 - l
r: 0.0010
Epoch 9/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0047 - accuracy: 1.0000 - val_loss: 0.0802 - val_accuracy: 0.9800 - l
r: 0.0010
Epoch 10/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.0763 - val_accuracy: 0.9800 - l
r: 0.0010
Epoch 11/100
38/38 [==============================] - 20s 527ms/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.0879 - val_accuracy: 0.9783 - l
r: 0.0010
Epoch 12/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.0903 - val_accuracy: 0.9767 - l
r: 0.0010
Epoch 13/100
38/38 [==============================] - 20s 533ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0818 - val_accuracy: 0.9800 - l
r: 0.0010
Epoch 14/100
38/38 [==============================] - 20s 538ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0871 - val_accuracy: 0.9783 - l
r: 0.0010
Epoch 15/100
38/38 [==============================] - ETA: 0s - loss: 0.0013 - accuracy: 1.0000
Epoch 15: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
38/38 [==============================] - 20s 533ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0915 - val_accuracy: 0.9783 - l
r: 0.0010
Epoch 16/100
38/38 [==============================] - 20s 531ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0886 - val_accuracy: 0.9783 - l
r: 1.0000e-04
Epoch 17/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0880 - val_accuracy: 0.9783 - l
r: 1.0000e-04
Epoch 18/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0882 - val_accuracy: 0.9783 - l
r: 1.0000e-04
Epoch 19/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0890 - val_accuracy: 0.9783 - l
r: 1.0000e-04
Epoch 20/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 20: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0893 - val_accuracy: 0.9783 - l
r: 1.0000e-04
Epoch 21/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0891 - val_accuracy: 0.9783 - l
r: 1.0000e-05
Epoch 22/100
38/38 [==============================] - 20s 539ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0888 - val_accuracy: 0.9783 - l
r: 1.0000e-05
Epoch 23/100
38/38 [==============================] - 20s 531ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0888 - val_accuracy: 0.9783 - l
r: 1.0000e-05
Epoch 24/100
38/38 [==============================] - 20s 535ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0888 - val_accuracy: 0.9783 - l
r: 1.0000e-05
Epoch 25/100
```

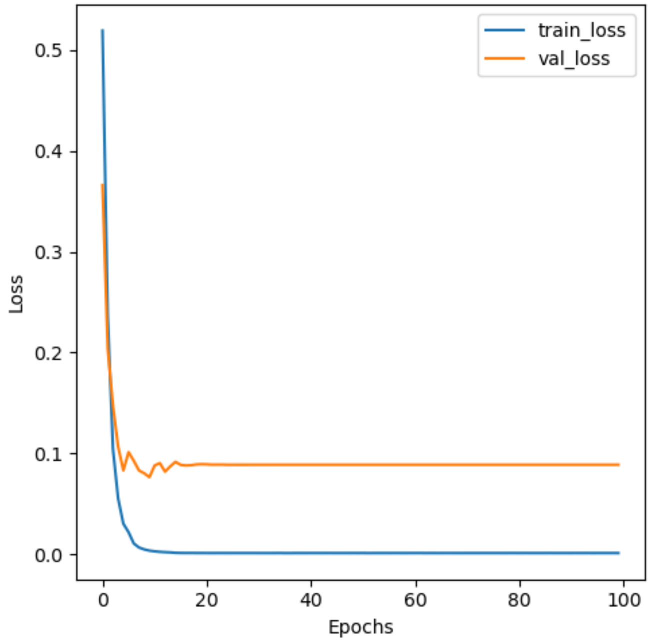
```
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 25: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
38/38 [==============================] - 20s 524ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-05
Epoch 26/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-06
Epoch 27/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-06
Epoch 28/100
38/38 [==============================] - 20s 525ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-06
Epoch 29/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-06
Epoch 30/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 30: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
38/38 [==============================] - 20s 531ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-06
Epoch 31/100
38/38 [==============================] - 20s 535ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-07
Epoch 32/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-07
Epoch 33/100
38/38 [==============================] - 20s 538ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-07
Epoch 34/100
38/38 [==============================] - 20s 536ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-07
Epoch 35/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 35: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
38/38 [==============================] - 20s 532ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-07
Epoch 36/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-08
Epoch 37/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-08
Epoch 38/100
38/38 [==============================] - 20s 533ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-08
Epoch 39/100
38/38 [==============================] - 21s 541ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-08
Epoch 40/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 40: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
38/38 [==============================] - 20s 533ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-08
Epoch 41/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-09
Epoch 42/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-09
Epoch 43/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-09
Epoch 44/100
38/38 [==============================] - 20s 534ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-09
Epoch 45/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 45: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
38/38 [==============================] - 20s 538ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-09
Epoch 46/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-10
Epoch 47/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-10
Epoch 48/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-10
Epoch 49/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-10
Epoch 50/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 50: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-11.
38/38 [==============================] - 20s 528ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-10
Epoch 51/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-11
Epoch 52/100
```

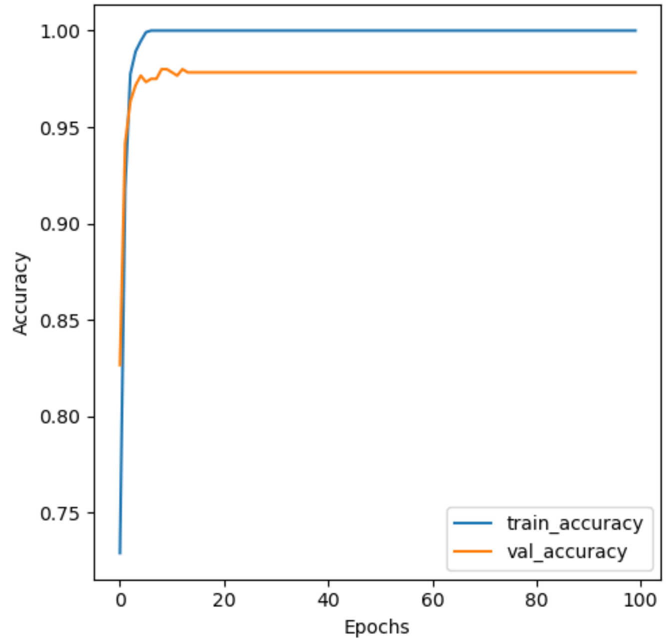
```
38/38 [==============================] - 20s 523ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-11
Epoch 53/100
38/38 [==============================] - 20s 522ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-11
Epoch 54/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-11
Epoch 55/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 55: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-12.
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-11
Epoch 56/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-12
Epoch 57/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-12
Epoch 58/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-12
Epoch 59/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-12
Epoch 60/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 60: ReduceLROnPlateau reducing learning rate to 1.0000001044244145e-13.
38/38 [==============================] - 21s 540ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-12
Epoch 61/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-13
Epoch 62/100
38/38 [==============================] - 20s 525ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-13
Epoch 63/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-13
Epoch 64/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-13
Epoch 65/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 65: ReduceLROnPlateau reducing learning rate to 1.0000001179769417e-14.
38/38 [==============================] - 20s 528ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-13
Epoch 66/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-14
Epoch 67/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-14
Epoch 68/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-14
Epoch 69/100
38/38 [==============================] - 20s 532ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-14
Epoch 70/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 70: ReduceLROnPlateau reducing learning rate to 1.0000001518582595e-15.
38/38 [==============================] - 20s 525ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-14
Epoch 71/100
38/38 [==============================] - 20s 527ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-15
Epoch 72/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-15
Epoch 73/100
38/38 [==============================] - 20s 525ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-15
Epoch 74/100
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-15
Epoch 75/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 75: ReduceLROnPlateau reducing learning rate to 1.0000001095066122e-16.
38/38 [==============================] - 20s 540ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-15
Epoch 76/100
38/38 [==============================] - 21s 548ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-16
Epoch 77/100
38/38 [==============================] - 20s 530ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-16
Epoch 78/100
38/38 [==============================] - 20s 527ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-16
Epoch 79/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-16
```

```
Epoch 80/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 80: ReduceLROnPlateau reducing learning rate to 1.0000000830368326e-17.
38/38 [==============================] - 20s 523ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-16
Epoch 81/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-17
Epoch 82/100
38/38 [==============================] - 20s 525ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-17
Epoch 83/100
38/38 [==============================] - 20s 523ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-17
Epoch 84/100
38/38 [==============================] - 20s 524ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-17
Epoch 85/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 85: ReduceLROnPlateau reducing learning rate to 1.0000000664932204e-18.
38/38 [==============================] - 20s 526ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-17
Epoch 86/100
38/38 [==============================] - 20s 535ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-18
Epoch 87/100
38/38 [==============================] - 20s 529ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-18
Epoch 88/100
38/38 [==============================] - 20s 531ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-18
Epoch 89/100
38/38 [==============================] - 20s 523ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-18
Epoch 90/100
38/38 [==============================] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 90: ReduceLROnPlateau reducing learning rate to 1.000000045813705e-19.
38/38 [==============================] - 20s 526ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-18
Epoch 91/100
38/38 [==============================] - 20s 521ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-19
Epoch 92/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-19
Epoch 93/100
38/38 [==============================] - 20s 528ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-19
Epoch 94/100
38/38 [==============================] - 20s 525ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-19
Epoch 95/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 95: ReduceLROnPlateau reducing learning rate to 1.000000032889008e-20.
38/38 [==============================] - 21s 544ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-19
Epoch 96/100
38/38 [==============================] - 22s 572ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-20
Epoch 97/100
38/38 [==============================] - 23s 611ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-20
Epoch 98/100
38/38 [==============================] - 20s 534ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-20
Epoch 99/100
38/38 [==============================] - 20s 534ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-20
Epoch 100/100
38/38 [==============================] - ETA: 0s - loss: 0.0010 - accuracy: 1.0000
Epoch 100: ReduceLROnPlateau reducing learning rate to 1.0000000490448793e-21.
38/38 [==============================] - 20s 526ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0887 - val_accuracy: 0.9783 - l
r: 1.0000e-20
```

In [170]:

```
plot_training_results(results100)
```






```
19/19 [==============================] - 1s 30ms/step
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       286
           1       0.99      0.97      0.98       314

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_63"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_59 (Conv2D)          (None, 126, 126, 16)      448

 dropout_45 (Dropout)        (None, 126, 126, 16)      0

 flatten_32 (Flatten)        (None, 254016)            0

 dropout_46 (Dropout)        (None, 254016)            0

 dense_82 (Dense)            (None, 16)                4064272

 dense_83 (Dense)            (None, 1)                 17

 activation_75 (Activation)  (None, 1)                 0

=================================================================
Total params: 4,064,737
Trainable params: 4,064,737
Non-trainable params: 0
_________________________________________________________________
```

In [162]:

```
test_loss, test_acc = model.evaluate(X_val, y_val)
model_acc_loss(test_acc, test_loss)
```

```
19/19 [==============================] - 1s 30ms/step - loss: 0.0899 - accuracy: 0.9767
Model Accuracy (Test data)
________________________________________

Model Accuracy:       0.9766666889190674
Test Loss:           0.0898614227771759
________________________________________
```

As clear as it seems, its overfitting. This can be due to several possibilities, but one area need to explore is the image size. Due to the fact that the dataset is fairly small, no need to capture so much details about the image, unlike a fairly large dataset where having more details and patterns is essential for better performance.

In [ ]:

In [ ]:

In [ ]:

In [65]:

```
output = model(train_img_final, train_labels_final, test_img_final, test_labels_final,
               num_iterations=2000, learning_rate=0.005, print_cost=True)
```

```
Cost after iteration 0: 0.693147
C:\Users\msavg\AppData\Local\Temp\ipykernel_1844\3547166593.py:4: RuntimeWarning: divide by zero encountered in log
  cost = -(1/l) * np.sum(y * np.log(y_hat) + (1-y)* np.log(1 - y_hat))
C:\Users\msavg\AppData\Local\Temp\ipykernel_1844\3547166593.py:4: RuntimeWarning: invalid value encountered in multiply
  cost = -(1/l) * np.sum(y * np.log(y_hat) + (1-y)* np.log(1 - y_hat))
Cost after iteration 50: 1.428492
Cost after iteration 100: 0.232415
Cost after iteration 150: 0.082094
Cost after iteration 200: 0.037336
Cost after iteration 250: 0.024759
Cost after iteration 300: 0.017805
Cost after iteration 350: 0.013923
Cost after iteration 400: 0.011646
Cost after iteration 450: 0.010154
Cost after iteration 500: 0.009082
Cost after iteration 550: 0.008264
Cost after iteration 600: 0.007614
Cost after iteration 650: 0.007081
Cost after iteration 700: 0.006633
Cost after iteration 750: 0.006250
Cost after iteration 800: 0.005917
Cost after iteration 850: 0.005625
Cost after iteration 900: 0.005366
Cost after iteration 950: 0.005134
Cost after iteration 1000: 0.004925
Cost after iteration 1050: 0.004735
Cost after iteration 1100: 0.004562
Cost after iteration 1150: 0.004403
Cost after iteration 1200: 0.004256
Cost after iteration 1250: 0.004121
Cost after iteration 1300: 0.003995
Cost after iteration 1350: 0.003877
Cost after iteration 1400: 0.003768
Cost after iteration 1450: 0.003665
Cost after iteration 1500: 0.003568
Cost after iteration 1550: 0.003477
Cost after iteration 1600: 0.003391
Cost after iteration 1650: 0.003310
Cost after iteration 1700: 0.003233
Cost after iteration 1750: 0.003161
Cost after iteration 1800: 0.003091
Cost after iteration 1850: 0.003025
Cost after iteration 1900: 0.002962
Cost after iteration 1950: 0.002902
train accuracy: 100.0 %
test accuracy: 71.0 %
```

FINAL

BEST MODEL *(simple model)*

model/brain_tumor_base_100_epochs_64_basics.h5

Its concluded that this is the final model that will be implemented and showcased to the shareholders at the presentation. Although the model seems a bit complexed, given the rigged lines, it seems the model is generalizing and learning. Its come a long way from the previous iterations of models conducted above. The main concern was the overfitting the models were suffering from. The complex nature of having multiple layers didnt help the fact that the model was volatile. Going simplistic, its evident the model can learn better, and close the overfit gap between train and validation data. With a 98% accuracy on the training data and little to no overfitting, but just jagged lines, we can be sure this model can serve a great purpose to predict brain tumor.

Load the model

In [34]:

```
# Load the best model
model_path = 'model/brain_tumor_base_100_epochs_64_basics.h5'

# Load the model
model = load_model(model_path)

# Folder path containing the images
image_path = 'data/pred/pred0.jpg'
new_image = cv2.imread(image_path)
```

In [91]:

```
# # Exexcute the model
# results100 = model.fit(X_train, y_train, batch_size = 64, verbose=1, epochs=100,
#                        validation_data= (X_val, y_val))
# # Save model
# model.save('model/brain_tumor_base_100_epochs_64_basics.h5')
```

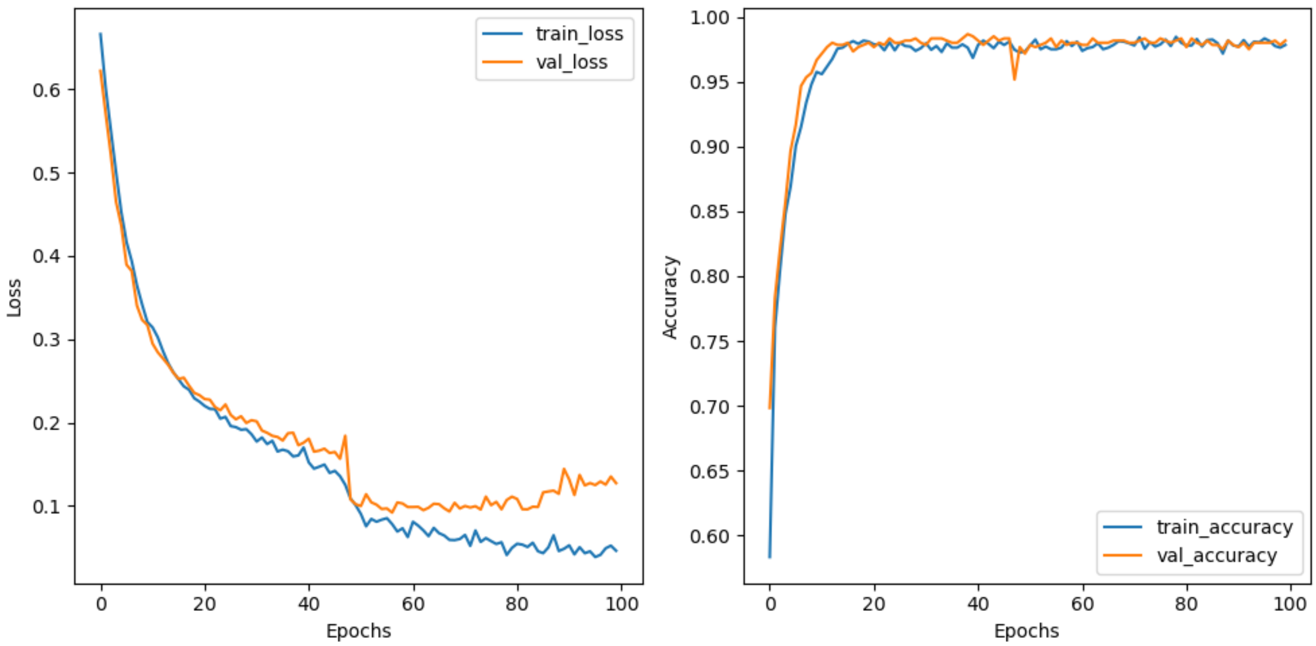
```
Epoch 1/100
38/38 [==============================] - 12s 315ms/step - loss: 0.6659 - accuracy: 0.5833 - val_loss: 0.6219 - val_accuracy: 0.6983
Epoch 2/100
38/38 [==============================] - 12s 324ms/step - loss: 0.6012 - accuracy: 0.7604 - val_loss: 0.5705 - val_accuracy: 0.7833
Epoch 3/100
38/38 [==============================] - 13s 333ms/step - loss: 0.5503 - accuracy: 0.8058 - val_loss: 0.5209 - val_accuracy: 0.8217
Epoch 4/100
38/38 [==============================] - 13s 335ms/step - loss: 0.4998 - accuracy: 0.8479 - val_loss: 0.4646 - val_accuracy: 0.8567
Epoch 5/100
38/38 [==============================] - 13s 334ms/step - loss: 0.4524 - accuracy: 0.8692 - val_loss: 0.4363 - val_accuracy: 0.8967
Epoch 6/100
38/38 [==============================] - 13s 340ms/step - loss: 0.4166 - accuracy: 0.9004 - val_loss: 0.3894 - val_accuracy: 0.9167
Epoch 7/100
38/38 [==============================] - 13s 333ms/step - loss: 0.3937 - accuracy: 0.9150 - val_loss: 0.3817 - val_accuracy: 0.9467
Epoch 8/100
38/38 [==============================] - 13s 334ms/step - loss: 0.3646 - accuracy: 0.9337 - val_loss: 0.3403 - val_accuracy: 0.9533
Epoch 9/100
38/38 [==============================] - 13s 331ms/step - loss: 0.3416 - accuracy: 0.9479 - val_loss: 0.3232 - val_accuracy: 0.9567
Epoch 10/100
38/38 [==============================] - 13s 330ms/step - loss: 0.3206 - accuracy: 0.9575 - val_loss: 0.3170 - val_accuracy: 0.9667
Epoch 11/100
38/38 [==============================] - 13s 331ms/step - loss: 0.3142 - accuracy: 0.9558 - val_loss: 0.2947 - val_accuracy: 0.9717
Epoch 12/100
38/38 [==============================] - 13s 330ms/step - loss: 0.3023 - accuracy: 0.9617 - val_loss: 0.2843 - val_accuracy: 0.9767
Epoch 13/100
38/38 [==============================] - 13s 333ms/step - loss: 0.2857 - accuracy: 0.9675 - val_loss: 0.2768 - val_accuracy: 0.9800
Epoch 14/100
38/38 [==============================] - 13s 332ms/step - loss: 0.2711 - accuracy: 0.9754 - val_loss: 0.2692 - val_accuracy: 0.9783
Epoch 15/100
38/38 [==============================] - 13s 333ms/step - loss: 0.2602 - accuracy: 0.9762 - val_loss: 0.2597 - val_accuracy: 0.9783
Epoch 16/100
38/38 [==============================] - 13s 334ms/step - loss: 0.2517 - accuracy: 0.9792 - val_loss: 0.2523 - val_accuracy: 0.9800
Epoch 17/100
38/38 [==============================] - 13s 333ms/step - loss: 0.2434 - accuracy: 0.9812 - val_loss: 0.2537 - val_accuracy: 0.9733
Epoch 18/100
38/38 [==============================] - 13s 337ms/step - loss: 0.2393 - accuracy: 0.9792 - val_loss: 0.2443 - val_accuracy: 0.9767
Epoch 19/100
38/38 [==============================] - 13s 341ms/step - loss: 0.2291 - accuracy: 0.9817 - val_loss: 0.2356 - val_accuracy: 0.9783
Epoch 20/100
38/38 [==============================] - 13s 334ms/step - loss: 0.2251 - accuracy: 0.9808 - val_loss: 0.2328 - val_accuracy: 0.9800
Epoch 21/100
38/38 [==============================] - 13s 342ms/step - loss: 0.2199 - accuracy: 0.9787 - val_loss: 0.2282 - val_accuracy: 0.9767
Epoch 22/100
38/38 [==============================] - 13s 339ms/step - loss: 0.2164 - accuracy: 0.9787 - val_loss: 0.2274 - val_accuracy: 0.9800
Epoch 23/100
38/38 [==============================] - 13s 339ms/step - loss: 0.2160 - accuracy: 0.9742 - val_loss: 0.2186 - val_accuracy: 0.9783
Epoch 24/100
38/38 [==============================] - 13s 341ms/step - loss: 0.2047 - accuracy: 0.9804 - val_loss: 0.2147 - val_accuracy: 0.9833
Epoch 25/100
38/38 [==============================] - 13s 335ms/step - loss: 0.2067 - accuracy: 0.9742 - val_loss: 0.2216 - val_accuracy: 0.9800
Epoch 26/100
38/38 [==============================] - 13s 334ms/step - loss: 0.1957 - accuracy: 0.9800 - val_loss: 0.2092 - val_accuracy: 0.9800
Epoch 27/100
38/38 [==============================] - 13s 341ms/step - loss: 0.1945 - accuracy: 0.9775 - val_loss: 0.2038 - val_accuracy: 0.9817
Epoch 28/100
38/38 [==============================] - 13s 335ms/step - loss: 0.1913 - accuracy: 0.9771 - val_loss: 0.2075 - val_accuracy: 0.9817
Epoch 29/100
38/38 [==============================] - 13s 335ms/step - loss: 0.1920 - accuracy: 0.9737 - val_loss: 0.1995 - val_accuracy: 0.9833
Epoch 30/100
38/38 [==============================] - 13s 340ms/step - loss: 0.1859 - accuracy: 0.9758 - val_loss: 0.2028 - val_accuracy: 0.9800
Epoch 31/100
38/38 [==============================] - 13s 346ms/step - loss: 0.1772 - accuracy: 0.9796 - val_loss: 0.2014 - val_accuracy: 0.9783
Epoch 32/100
38/38 [==============================] - 13s 333ms/step - loss: 0.1820 - accuracy: 0.9746 - val_loss: 0.1904 - val_accuracy: 0.9833
Epoch 33/100
38/38 [==============================] - 13s 337ms/step - loss: 0.1742 - accuracy: 0.9775 - val_loss: 0.1877 - val_accuracy: 0.9833
Epoch 34/100
38/38 [==============================] - 13s 338ms/step - loss: 0.1781 - accuracy: 0.9729 - val_loss: 0.1840 - val_accuracy: 0.9833
Epoch 35/100
38/38 [==============================] - 13s 344ms/step - loss: 0.1652 - accuracy: 0.9796 - val_loss: 0.1827 - val_accuracy: 0.9817
Epoch 36/100
38/38 [==============================] - 13s 339ms/step - loss: 0.1674 - accuracy: 0.9762 - val_loss: 0.1784 - val_accuracy: 0.9800
Epoch 37/100
38/38 [==============================] - 13s 335ms/step - loss: 0.1655 - accuracy: 0.9762 - val_loss: 0.1871 - val_accuracy: 0.9800
Epoch 38/100
38/38 [==============================] - 13s 340ms/step - loss: 0.1593 - accuracy: 0.9787 - val_loss: 0.1877 - val_accuracy: 0.9833
Epoch 39/100
38/38 [==============================] - 13s 337ms/step - loss: 0.1605 - accuracy: 0.9762 - val_loss: 0.1728 - val_accuracy: 0.9867
Epoch 40/100
38/38 [==============================] - 13s 341ms/step - loss: 0.1700 - accuracy: 0.9683 - val_loss: 0.1758 - val_accuracy: 0.9850
Epoch 41/100
38/38 [==============================] - 13s 337ms/step - loss: 0.1521 - accuracy: 0.9783 - val_loss: 0.1806 - val_accuracy: 0.9817
Epoch 42/100
38/38 [==============================] - 13s 336ms/step - loss: 0.1446 - accuracy: 0.9817 - val_loss: 0.1651 - val_accuracy: 0.9783
Epoch 43/100
```

```
38/38 [==============================] - 13s 337ms/step - loss: 0.1470 - accuracy: 0.9792 - val_loss: 0.1663 - val_accuracy: 0.9817
Epoch 44/100
38/38 [==============================] - 13s 338ms/step - loss: 0.1496 - accuracy: 0.9758 - val_loss: 0.1686 - val_accuracy: 0.9850
Epoch 45/100
38/38 [==============================] - 13s 341ms/step - loss: 0.1395 - accuracy: 0.9808 - val_loss: 0.1634 - val_accuracy: 0.9817
Epoch 46/100
38/38 [==============================] - 13s 337ms/step - loss: 0.1420 - accuracy: 0.9783 - val_loss: 0.1646 - val_accuracy: 0.9833
Epoch 47/100
38/38 [==============================] - 13s 342ms/step - loss: 0.1356 - accuracy: 0.9808 - val_loss: 0.1565 - val_accuracy: 0.9833
Epoch 48/100
38/38 [==============================] - 13s 335ms/step - loss: 0.1253 - accuracy: 0.9746 - val_loss: 0.1841 - val_accuracy: 0.9517
Epoch 49/100
38/38 [==============================] - 13s 330ms/step - loss: 0.1090 - accuracy: 0.9725 - val_loss: 0.1078 - val_accuracy: 0.9767
Epoch 50/100
38/38 [==============================] - 13s 333ms/step - loss: 0.1008 - accuracy: 0.9733 - val_loss: 0.1019 - val_accuracy: 0.9717
Epoch 51/100
38/38 [==============================] - 13s 333ms/step - loss: 0.0902 - accuracy: 0.9779 - val_loss: 0.0998 - val_accuracy: 0.9783
Epoch 52/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0755 - accuracy: 0.9825 - val_loss: 0.1140 - val_accuracy: 0.9767
Epoch 53/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0843 - accuracy: 0.9750 - val_loss: 0.1042 - val_accuracy: 0.9783
Epoch 54/100
38/38 [==============================] - 13s 335ms/step - loss: 0.0810 - accuracy: 0.9771 - val_loss: 0.1014 - val_accuracy: 0.9800
Epoch 55/100
38/38 [==============================] - 13s 336ms/step - loss: 0.0834 - accuracy: 0.9750 - val_loss: 0.0963 - val_accuracy: 0.9833
Epoch 56/100
38/38 [==============================] - 13s 337ms/step - loss: 0.0852 - accuracy: 0.9750 - val_loss: 0.0969 - val_accuracy: 0.9767
Epoch 57/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0781 - accuracy: 0.9762 - val_loss: 0.0919 - val_accuracy: 0.9817
Epoch 58/100
38/38 [==============================] - 13s 329ms/step - loss: 0.0692 - accuracy: 0.9812 - val_loss: 0.1041 - val_accuracy: 0.9783
Epoch 59/100
38/38 [==============================] - 12s 329ms/step - loss: 0.0731 - accuracy: 0.9775 - val_loss: 0.1028 - val_accuracy: 0.9800
Epoch 60/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0626 - accuracy: 0.9808 - val_loss: 0.0986 - val_accuracy: 0.9800
Epoch 61/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0809 - accuracy: 0.9737 - val_loss: 0.0986 - val_accuracy: 0.9783
Epoch 62/100
38/38 [==============================] - 13s 336ms/step - loss: 0.0763 - accuracy: 0.9758 - val_loss: 0.0988 - val_accuracy: 0.9783
Epoch 63/100
38/38 [==============================] - 13s 339ms/step - loss: 0.0706 - accuracy: 0.9767 - val_loss: 0.0949 - val_accuracy: 0.9833
Epoch 64/100
38/38 [==============================] - 13s 346ms/step - loss: 0.0637 - accuracy: 0.9796 - val_loss: 0.0977 - val_accuracy: 0.9800
Epoch 65/100
38/38 [==============================] - 13s 339ms/step - loss: 0.0736 - accuracy: 0.9750 - val_loss: 0.1025 - val_accuracy: 0.9800
Epoch 66/100
38/38 [==============================] - 13s 347ms/step - loss: 0.0673 - accuracy: 0.9762 - val_loss: 0.1020 - val_accuracy: 0.9800
Epoch 67/100
38/38 [==============================] - 13s 343ms/step - loss: 0.0645 - accuracy: 0.9783 - val_loss: 0.0971 - val_accuracy: 0.9817
Epoch 68/100
38/38 [==============================] - 13s 346ms/step - loss: 0.0592 - accuracy: 0.9812 - val_loss: 0.0933 - val_accuracy: 0.9817
Epoch 69/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0589 - accuracy: 0.9808 - val_loss: 0.1037 - val_accuracy: 0.9817
Epoch 70/100
38/38 [==============================] - 13s 336ms/step - loss: 0.0603 - accuracy: 0.9796 - val_loss: 0.0969 - val_accuracy: 0.9800
Epoch 71/100
38/38 [==============================] - 13s 335ms/step - loss: 0.0652 - accuracy: 0.9779 - val_loss: 0.0999 - val_accuracy: 0.9800
Epoch 72/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0521 - accuracy: 0.9842 - val_loss: 0.0980 - val_accuracy: 0.9817
Epoch 73/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0705 - accuracy: 0.9754 - val_loss: 0.0996 - val_accuracy: 0.9833
Epoch 74/100
38/38 [==============================] - 13s 333ms/step - loss: 0.0568 - accuracy: 0.9804 - val_loss: 0.0956 - val_accuracy: 0.9800
Epoch 75/100
38/38 [==============================] - 13s 333ms/step - loss: 0.0614 - accuracy: 0.9771 - val_loss: 0.1110 - val_accuracy: 0.9800
Epoch 76/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0577 - accuracy: 0.9787 - val_loss: 0.1009 - val_accuracy: 0.9833
Epoch 77/100
38/38 [==============================] - 13s 330ms/step - loss: 0.0543 - accuracy: 0.9825 - val_loss: 0.1048 - val_accuracy: 0.9817
Epoch 78/100
38/38 [==============================] - 13s 330ms/step - loss: 0.0564 - accuracy: 0.9775 - val_loss: 0.0959 - val_accuracy: 0.9800
Epoch 79/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0411 - accuracy: 0.9846 - val_loss: 0.1070 - val_accuracy: 0.9817
Epoch 80/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0496 - accuracy: 0.9800 - val_loss: 0.1110 - val_accuracy: 0.9833
Epoch 81/100
38/38 [==============================] - 13s 330ms/step - loss: 0.0545 - accuracy: 0.9779 - val_loss: 0.1083 - val_accuracy: 0.9767
Epoch 82/100
38/38 [==============================] - 13s 335ms/step - loss: 0.0535 - accuracy: 0.9779 - val_loss: 0.0960 - val_accuracy: 0.9833
Epoch 83/100
38/38 [==============================] - 13s 338ms/step - loss: 0.0507 - accuracy: 0.9829 - val_loss: 0.0959 - val_accuracy: 0.9800
Epoch 84/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0558 - accuracy: 0.9771 - val_loss: 0.0990 - val_accuracy: 0.9783
Epoch 85/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0456 - accuracy: 0.9821 - val_loss: 0.0986 - val_accuracy: 0.9817
Epoch 86/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0433 - accuracy: 0.9825 - val_loss: 0.1162 - val_accuracy: 0.9783
Epoch 87/100
38/38 [==============================] - 13s 331ms/step - loss: 0.0501 - accuracy: 0.9796 - val_loss: 0.1173 - val_accuracy: 0.9783
Epoch 88/100
38/38 [==============================] - 12s 329ms/step - loss: 0.0648 - accuracy: 0.9717 - val_loss: 0.1183 - val_accuracy: 0.9750
Epoch 89/100
38/38 [==============================] - 13s 343ms/step - loss: 0.0458 - accuracy: 0.9817 - val_loss: 0.1145 - val_accuracy: 0.9817
```

```
Epoch 90/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0485 - accuracy: 0.9779 - val_loss: 0.1444 - val_accuracy: 0.9783
Epoch 91/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0527 - accuracy: 0.9771 - val_loss: 0.1316 - val_accuracy: 0.9767
Epoch 92/100
38/38 [==============================] - 13s 333ms/step - loss: 0.0419 - accuracy: 0.9821 - val_loss: 0.1131 - val_accuracy: 0.9800
Epoch 93/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0504 - accuracy: 0.9771 - val_loss: 0.1371 - val_accuracy: 0.9750
Epoch 94/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0432 - accuracy: 0.9808 - val_loss: 0.1246 - val_accuracy: 0.9800
Epoch 95/100
38/38 [==============================] - 13s 333ms/step - loss: 0.0455 - accuracy: 0.9804 - val_loss: 0.1274 - val_accuracy: 0.9800
Epoch 96/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0386 - accuracy: 0.9833 - val_loss: 0.1250 - val_accuracy: 0.9800
Epoch 97/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0413 - accuracy: 0.9812 - val_loss: 0.1293 - val_accuracy: 0.9800
Epoch 98/100
38/38 [==============================] - 13s 334ms/step - loss: 0.0491 - accuracy: 0.9775 - val_loss: 0.1255 - val_accuracy: 0.9817
Epoch 99/100
38/38 [==============================] - 13s 332ms/step - loss: 0.0523 - accuracy: 0.9762 - val_loss: 0.1352 - val_accuracy: 0.9783
Epoch 100/100
38/38 [==============================] - 13s 335ms/step - loss: 0.0462 - accuracy: 0.9783 - val_loss: 0.1273 - val_accuracy: 0.9817
```

In [92]:

```
# plot_training_results(results100)
```




```
19/19 [==============================] - 1s 53ms/step
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       286
           1       0.98      0.98      0.98       314

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600

Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_30 (Conv2D)          (None, 126, 126, 32)      896

 activation_55 (Activation)  (None, 126, 126, 32)      0

 max_pooling2d_28 (MaxPoolin  (None, 63, 63, 32)       0
 g2D)

 flatten_7 (Flatten)         (None, 127008)            0

 dense_41 (Dense)            (None, 16)                2032144

 dropout_20 (Dropout)        (None, 16)                0

 dense_42 (Dense)            (None, 1)                 17

 activation_56 (Activation)  (None, 1)                 0

=================================================================
Total params: 2,033,057
Trainable params: 2,033,057
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [46]:

```
test_loss, test_acc = model.evaluate(X_val, y_val)
model_acc_loss(test_acc, test_loss)
```

```
19/19 [==============================] - 1s 45ms/step - loss: 0.0195 - accuracy: 0.9967
Model Accuracy (Test data)
________________________________________

Model Accuracy:       0.996666669845581
Test Loss:           0.019506091251969337
________________________________________
```

Test the model with unforseen images

Load the model

In [47]:

```
# Load the best model
model_path = 'model/brain_tumor_base_100_epochs_64_basics.h5'

# Load the model
model = load_model(model_path)

# Folder path containing the images
image_path = 'data/pred'
new_image = cv2.imread(image_path)
```

Load the images

In [40]:

```
def load_images(folder_path):
    images = []
    labels = []

    # Iterate over each image file in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            # Load and resize the image
            img = cv2.imread(os.path.join(folder_path, filename))
            img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Append the preprocessed image and label
            images.append(img)
            labels.append(folder_path.split('/')[-1])  # Assuming folder name is the label



    return images, labels
```

Predict image classification from the 'pred' folder (unforeseen)

In [48]:

```
# Load unseen images from 'pred' folder for prediction
pred_images, pred_labels = load_images(image_path)

# Convert the images to NumPy arrays and normalize pixel values
pred_images = np.array(pred_images) / 255.0

# Make predictions on unseen images
predictions = model.predict(pred_images)

# Print the predicted labels
for i, pred in enumerate(predictions):
    label = 'yes' if pred >= 0.5 else 'no'
    print(f"Image {i+1}: Predicted Label - {label}")
```

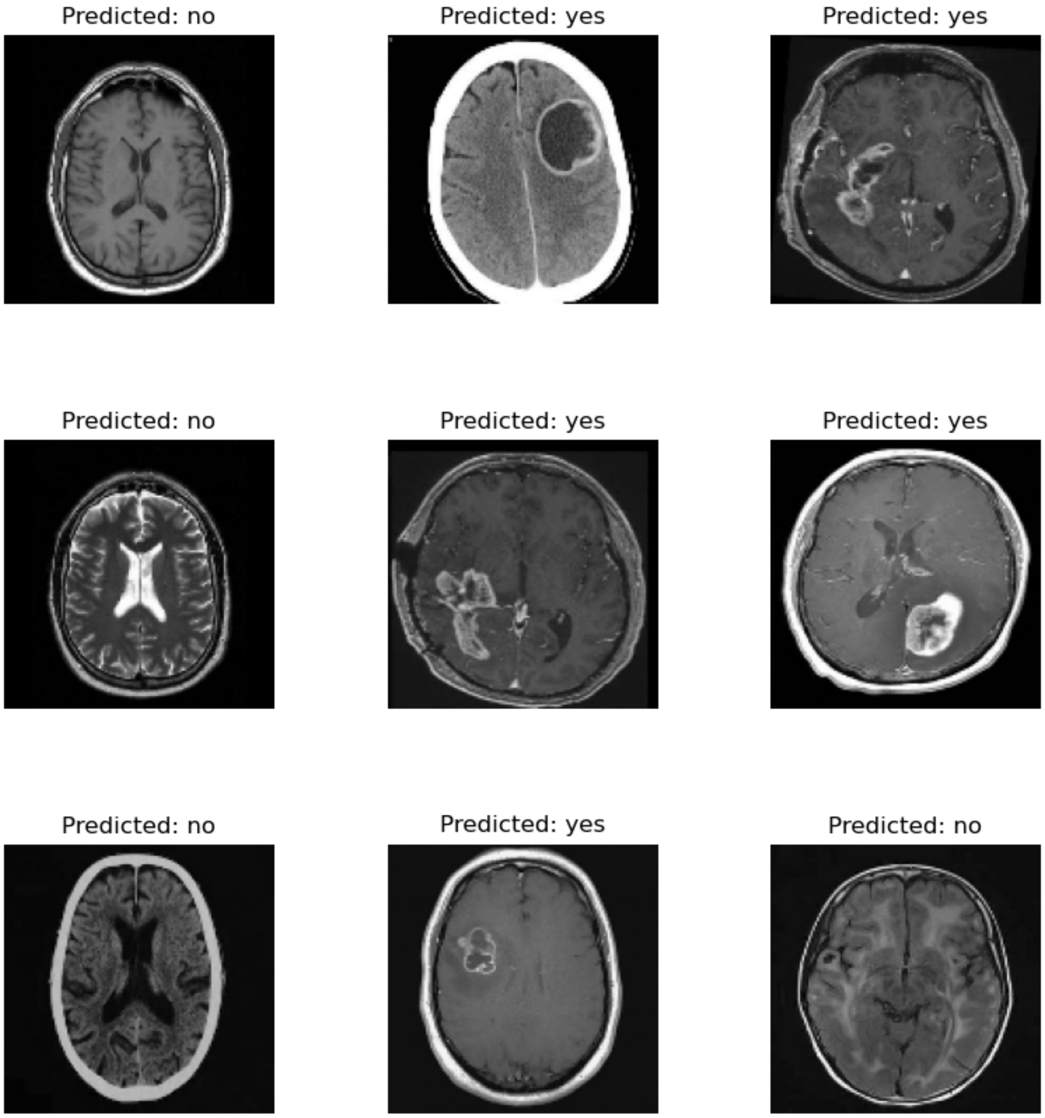
```
2/2 [==============================] - 0s 38ms/step
Image 1: Predicted Label - no
Image 2: Predicted Label - no
Image 3: Predicted Label - yes
Image 4: Predicted Label - yes
Image 5: Predicted Label - no
Image 6: Predicted Label - yes
Image 7: Predicted Label - yes
Image 8: Predicted Label - no
Image 9: Predicted Label - yes
Image 10: Predicted Label - no
Image 11: Predicted Label - no
Image 12: Predicted Label - no
Image 13: Predicted Label - no
Image 14: Predicted Label - no
```

```
Image 15: Predicted Label - no
Image 16: Predicted Label - no
Image 17: Predicted Label - no
Image 18: Predicted Label - no
Image 19: Predicted Label - no
Image 20: Predicted Label - no
Image 21: Predicted Label - no
Image 22: Predicted Label - yes
Image 23: Predicted Label - no
Image 24: Predicted Label - no
Image 25: Predicted Label - no
Image 26: Predicted Label - no
Image 27: Predicted Label - no
Image 28: Predicted Label - no
Image 29: Predicted Label - no
Image 30: Predicted Label - no
Image 31: Predicted Label - no
Image 32: Predicted Label - no
Image 33: Predicted Label - no
Image 34: Predicted Label - yes
Image 35: Predicted Label - no
Image 36: Predicted Label - no
Image 37: Predicted Label - no
Image 38: Predicted Label - no
Image 39: Predicted Label - no
Image 40: Predicted Label - no
Image 41: Predicted Label - yes
Image 42: Predicted Label - no
Image 43: Predicted Label - no
Image 44: Predicted Label - no
Image 45: Predicted Label - no
Image 46: Predicted Label - yes
Image 47: Predicted Label - no
Image 48: Predicted Label - no
Image 49: Predicted Label - no
Image 50: Predicted Label - no
Image 51: Predicted Label - no
Image 52: Predicted Label - yes
Image 53: Predicted Label - yes
Image 54: Predicted Label - no
Image 55: Predicted Label - no
Image 56: Predicted Label - no
Image 57: Predicted Label - no
Image 58: Predicted Label - yes
Image 59: Predicted Label - no
Image 60: Predicted Label - yes
```

Print the images with Predicted labels

In [35]:

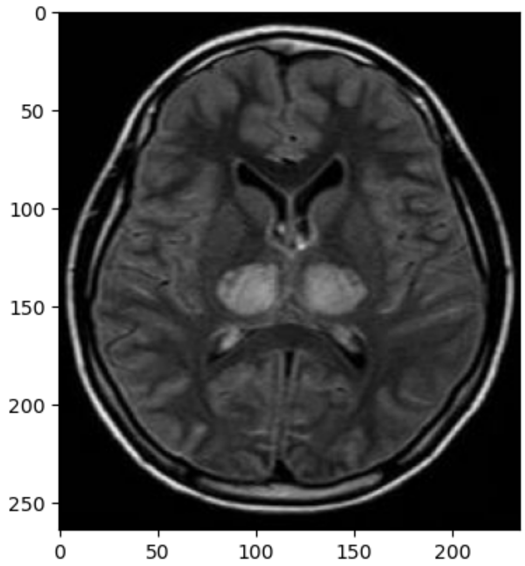
```python
# Plot the images with predicted labels
fig, axs = plt.subplots(3, 3, figsize=(10, 10))
fig.subplots_adjust(hspace=0.5)
counter = 1
for i in range(3):
    for j in range(3):
        axs[i, j].imshow(pred_images[counter])
        axs[i, j].axis('off')
        label = 'yes' if predictions[counter] >= 0.5 else 'no'
        axs[i, j].set_title(f"Predicted: {label}")
        counter += 1
plt.show()
```



In [37]:

```
def names(number):
    if number==0:
        return 'a Tumor'
    else:
        return 'not a tumor'



from matplotlib.pyplot import imshow

# Load the image
img = Image.open("data/pred/pred28.jpg")

# Resize the image to (128, 128) and convert it to a NumPy array
x = np.array(img.resize((128, 128)))

# Reshape the array to match the expected input shape of the model
x = x.reshape(1, 128, 128, 3)

# Make predictions on the image using the model
res = model.predict_on_batch(x)

# Get the index of the predicted class
classification = np.where(res == np.amax(res))[1][0]

# Display the image using matplotlib
imshow(img)

# Print the confidence and the predicted class name
print(str(res[0][classification] * 100) + '% Confidence This Is ' + names(classification))
```

```
100.0% Confidence This Is A Its a Tumor
```



As you can see - after loading the images and running through the model, then plot, we can see a 3x3 of the images the model predicted and classified as.

In [51]:

```
# Convert the predictions to binary labels (0 or 1)
pred_labels = np.round(predictions)

# make prediction on pred_images
y_pred_prob = model.predict(pred_images)
threshold = 0.5
y_pred = (y_pred_prob > threshold).astype(int)


print (classification_report(pred_labels, y_pred))
```

```
2/2 [==============================] - 0s 41ms/step
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        47
         1.0       1.00      1.00      1.00        13

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```

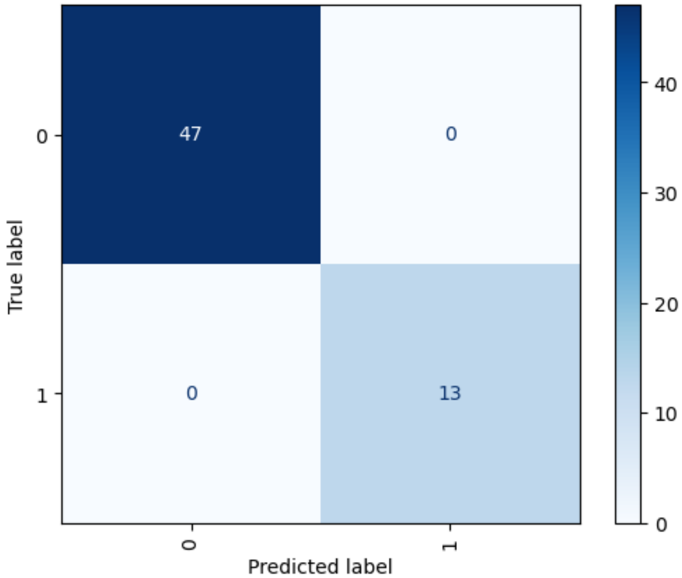
In [52]:

```
test_loss, test_acc = model.evaluate(pred_images, pred_labels)
model_acc_loss(test_acc, test_loss)
```

```
2/2 [==============================] - 0s 43ms/step - loss: 0.0264 - accuracy: 1.0000
Model Accuracy (Test data)
________________________________________

Model Accuracy:       1.0
Test Loss:           0.026415057480335236
________________________________________
```

In [57]:

```
# Confusion Matrix
confused_matrix = confusion_matrix(pred_labels, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = confused_matrix)
disp.plot(cmap='Blues', values_format='d', xticks_rotation='vertical', colorbar=True)
plt.show()
```



In certain scenarios, it is possible for the model to exhibit superior performance on the test dataset when compared to the validation dataset. This discrepancy can be attributed to various factors, including dissimilarities in data distribution, variations in data quality, or the random partitioning of data between the datasets. If the model consistently demonstrates better performance on the test dataset compared to the validation dataset, it indicates a higher likelihood of the model's ability to generalize effectively to new, unseen data. Nevertheless, it remains crucial to ensure that the test dataset accurately represents the real-world data distribution and that the evaluation metrics used provide reliable indicators of the model's performance.

EVALUATION

In the output you provided, the classification report shows that the model has achieved perfect accuracy (1.00) for both classes. The precision, recall, and F1-score are all 1.00 for both classes, indicating perfect performance. The support indicates the number of samples in each class (47 for class 0 and 13 for class 1). Overall, the classification report suggests that the model is performing very well and is able to accurately classify the data

In [44]:

```
test_loss, test_acc = model.evaluate(pred_images, pred_labels)
model_acc_loss(test_acc, test_loss)
```

```
2/2 [==============================] - 0s 44ms/step - loss: 0.0264 - accuracy: 1.0000
Model Accuracy (Test data)
________________________________________

Model Accuracy:       1.0
Test Loss:           0.026415057480335236
________________________________________
```

MOVING FORWARD

After establishing a succesful model, the model was able to predict roughly 99% of 60 unforseen sample MRI images. This means the model is well generalized and can predict new images very well. As the journey to build a sucessful model as come to a close, the work is not yet over. Below, you will see what will be the next steps moving forward:

Model Improvement
The model still has space to improve by using more images to better train and predict Brain Tumor. This in fact can serve as strengthening the model and can serve to be more accurate.

Regular Model Evaluation
It is important the model is monitored regularly to help the model maintain running at peak optimal performance.

Better understand and classify benign vs malignant tumors
As the model develops, one important feature is to expand the models functionality to add more classifiers that can determine what would be considered cancerous from non-cancerous tumors to better aid medical practicioners and encourage better decision making.

Release of Brain Tumor Detection v.1.0.0 *(Work in Progress)*
The next steps that will be taken is to further enhance our end-user interface Brain Tumor Detection that is currently v.1.0.0 and in its (Beta) phase. It has functionality to upload an image from the user local drive, then the user can press the *Detect Tumor* button to get a response as to whether or not the MRI image uploaded in fact contains a tumor.

- End of Document -