master-thesis copy

September 5, 2023

Getting Started We will be using TensorFlow and Keras for data augmentation and matplotlib for displaying the images.

[1]:
```
from glob import glob
import pandas as pd
import cv2
from scripts.visualization import Visualization
import matplotlib.pyplot as plt
import numpy as np
```

Reading Data

[2]:
```
# Path to all data
data_dir = './lgg-mri-segmentation/kaggle_3m'

# img size
IMG_SIZE = 512
```

[3]:
```
images_paths = []
masks_paths = glob(f'{data_dir}/*/*_mask*')

for i in masks_paths:
    images_paths.append(i.replace('_mask', ''))

df = pd.DataFrame(data= {'images_paths': images_paths, 'masks_paths':␣
↪masks_paths})
df.head()
```

[3]:
```
                                          images_paths
0  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…  \
1  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
2  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
3  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
4  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…

                                           masks_paths
0  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
1  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
```

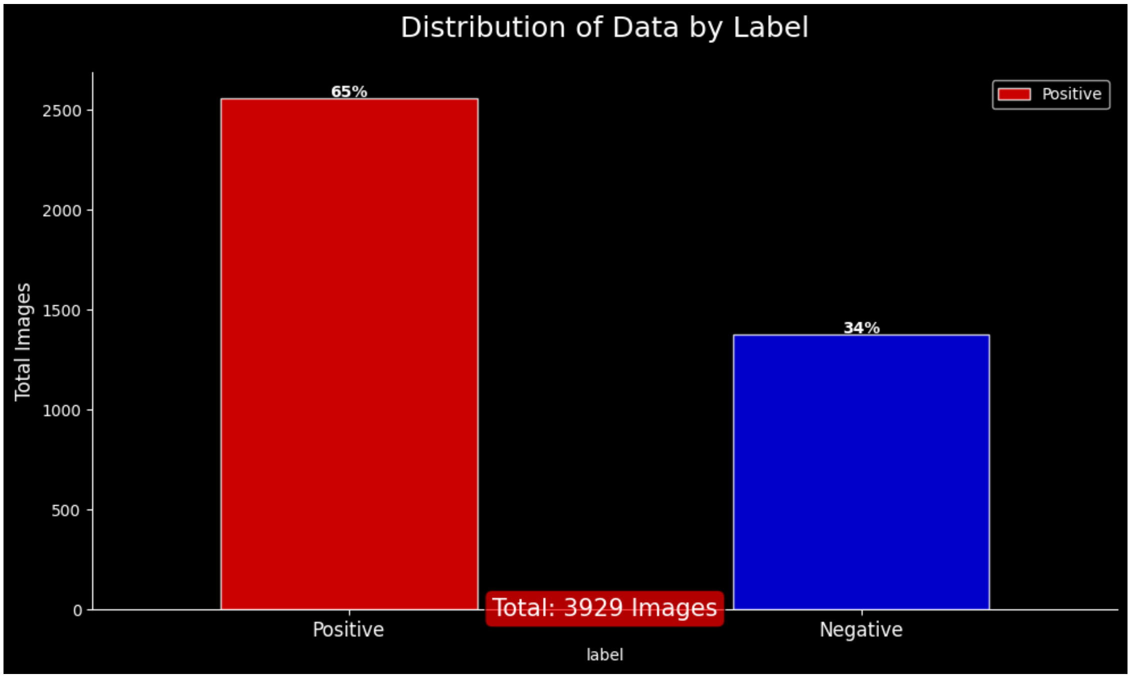
```
2  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
3  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
4  ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
```

[4]:
```
def pos_neg_diagnosis(masks_paths):
    value = np.max(cv2.imread(masks_paths))
    if value > 0 :
        return 1
    else:
        return 0

df['label'] = df['masks_paths'].apply(lambda x: pos_neg_diagnosis(x))
df
```

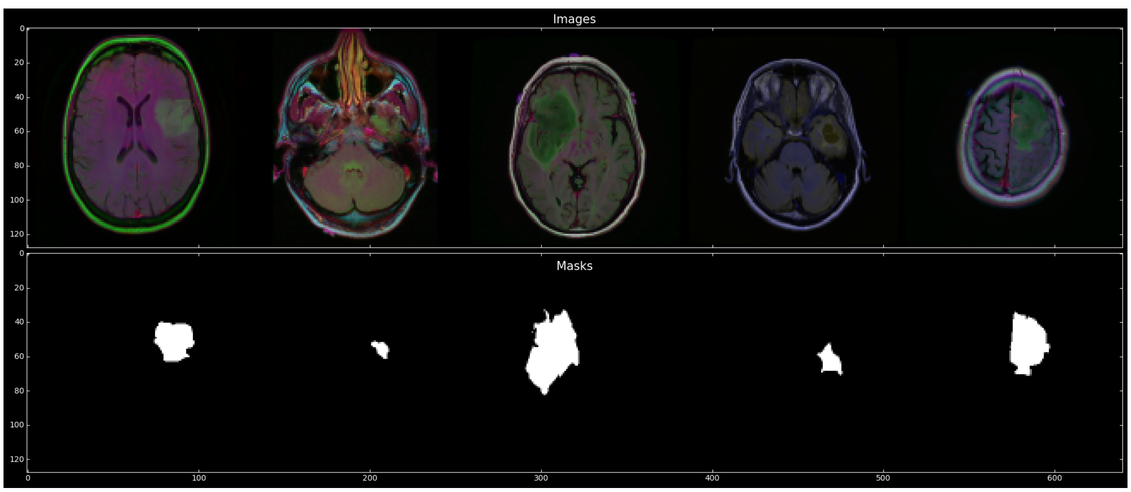
[4]:
```
                                           images_paths  \
0     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
1     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
2     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
3     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
4     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…
…                                                   …
3924  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…
3925  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…
3926  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…
3927  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…
3928  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…

                                            masks_paths  label
0     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…      0
1     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…      0
2     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…      0
3     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…      0
4     ./lgg-mri-segmentation/kaggle_3m/TCGA_CS_6667_…      0
…                                                   …    …
3924  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…      0
3925  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…      0
3926  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…      0
3927  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…      0
3928  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_A60K_…      0

[3929 rows x 3 columns]
```

Data Distribution

[5]:
```
visualization = Visualization(df)
visualization.plot_distribution_grouped_by_label()
```



[6]:
```
visualization = Visualization(df)
visualization.plot_images_and_masks()
```



Data Loading In the code below, we have loaded 80% training, 10% validation, and a 10% test set with labels and metadata.

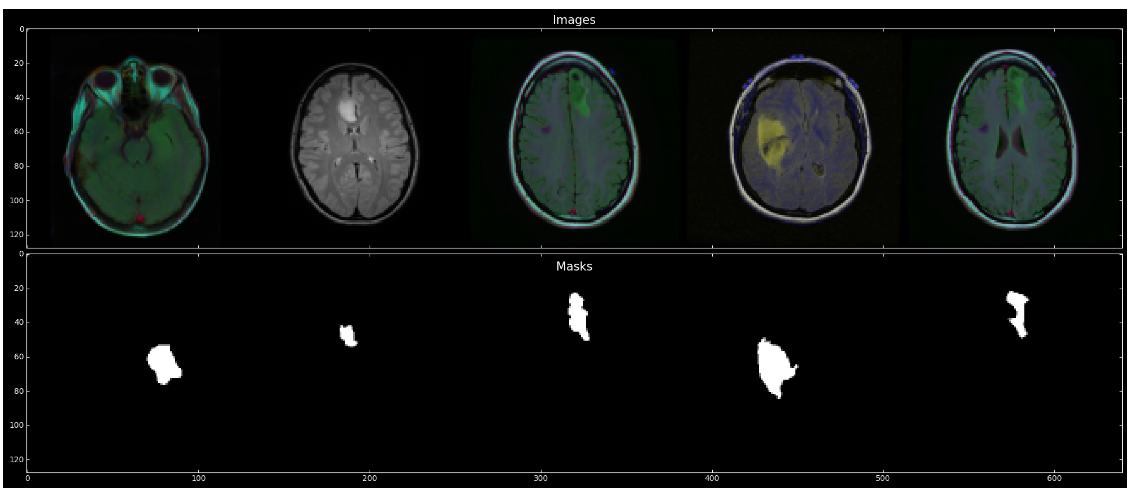
[7]:
```
mask_df = df[df['label'] == 1]
mask_df.shape
```

[7]: (1373, 3)

[8]:
```
from sklearn.model_selection import train_test_split
# Split df into train_df and val_df
train_df, val_df = train_test_split(mask_df, stratify=mask_df.label,␣
 ↪test_size=0.1)
train_df = train_df.reset_index(drop=True)
val_df = val_df.reset_index(drop=True)

# Split train_df into train_df and test_df
val_df, test_df = train_test_split(val_df, stratify=val_df.label, test_size=0.3)
val_df = val_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)

# train_df = train_df[:1000]
print(f"Train: {train_df.shape} \nVal: {val_df.shape} \nTest: {test_df.shape}")
```

```
Train: (1235, 3)
Val: (96, 3)
Test: (42, 3)
```

[9]:
```
visualization = Visualization(train_df)
visualization.plot_images_and_masks()
```



GAN

[10]:
```
train_df
```

[10]:
```
                                         images_paths
0     ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7686_…  \
1     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_A5TR_…
```

```
2     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7299_…
3     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7014_…
4     ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7605_…
…                                               …
1230  ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_A5TW_…
1231  ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_8107_…
1232  ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7608_…
1233  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_7637_…
1234  ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7014_…

                                            masks_paths  label
0     ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7686_…      1
1     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_A5TR_…      1
2     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7299_…      1
3     ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7014_…      1
4     ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7605_…      1
…                                               …    …
1230  ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_A5TW_…      1
1231  ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_8107_…      1
1232  ./lgg-mri-segmentation/kaggle_3m/TCGA_HT_7608_…      1
1233  ./lgg-mri-segmentation/kaggle_3m/TCGA_FG_7637_…      1
1234  ./lgg-mri-segmentation/kaggle_3m/TCGA_DU_7014_…      1

[1235 rows x 3 columns]
```

[11]:
```python
from scripts.brain_mri_dataset import BrainMriDataset
from torch.utils.data import DataLoader
IMG_SIZE = 64
BATCH_SIZE = 26

# train
train_dataset = BrainMriDataset(df=train_df, img_size=IMG_SIZE)
train_dataloader = DataLoader(train_dataset, batch_size=BATCH_SIZE,␣
 ↪num_workers=4, shuffle=True)

# val
val_dataset = BrainMriDataset(df=val_df, img_size=IMG_SIZE)
val_dataloader = DataLoader(val_dataset, batch_size=BATCH_SIZE, num_workers=4,␣
 ↪shuffle=True)

#test
test_dataset = BrainMriDataset(df=test_df, img_size=IMG_SIZE)
test_dataloader = DataLoader(test_dataset, batch_size=BATCH_SIZE,␣
 ↪num_workers=4, shuffle=True)
```

[12]:
```python
images, masks, labels = next(iter(train_dataloader))
```

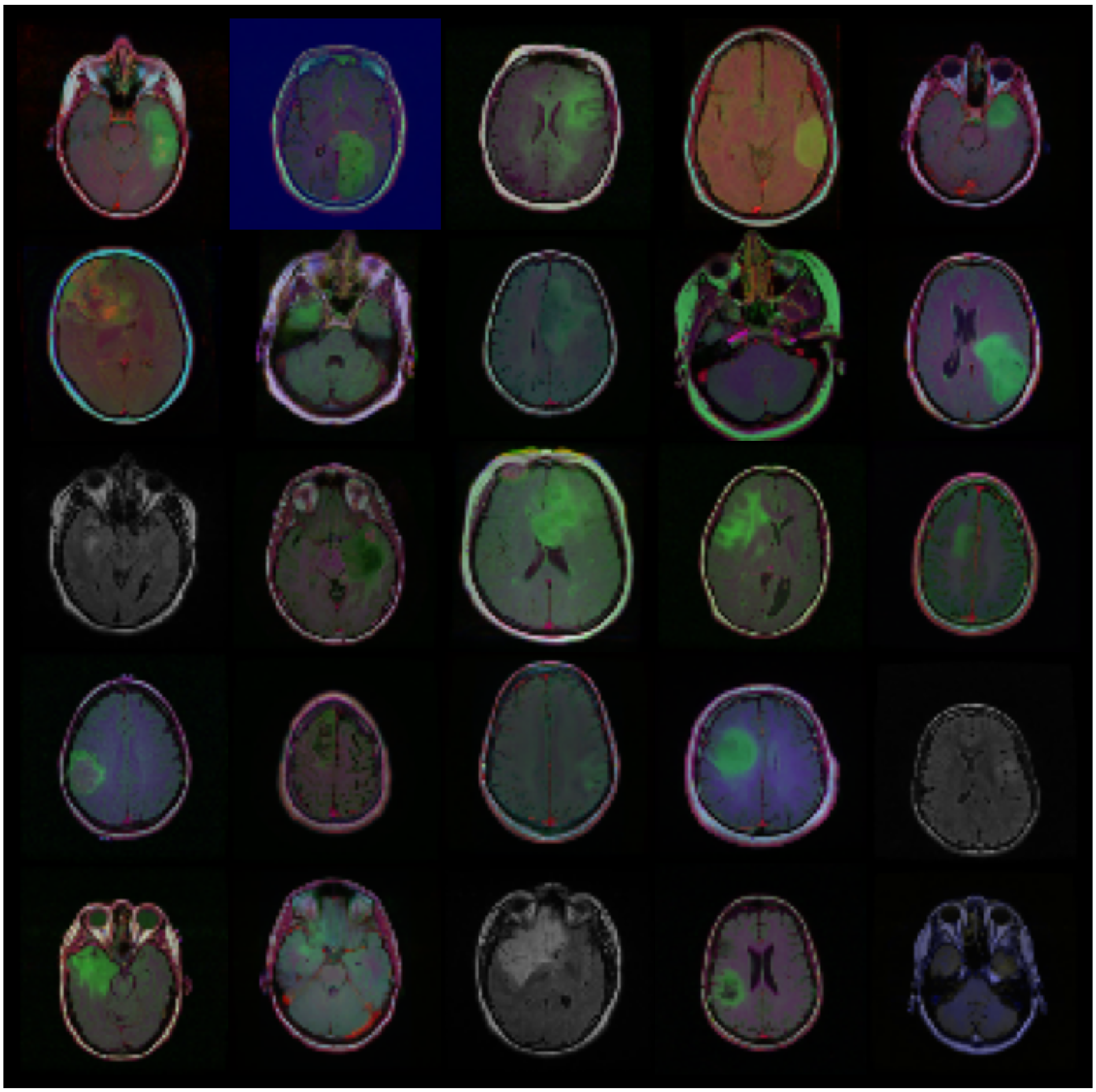
[13]:
```
print(images.shape, masks.shape)
```

```
torch.Size([26, 64, 64, 3]) torch.Size([26, 64, 64, 3])
```

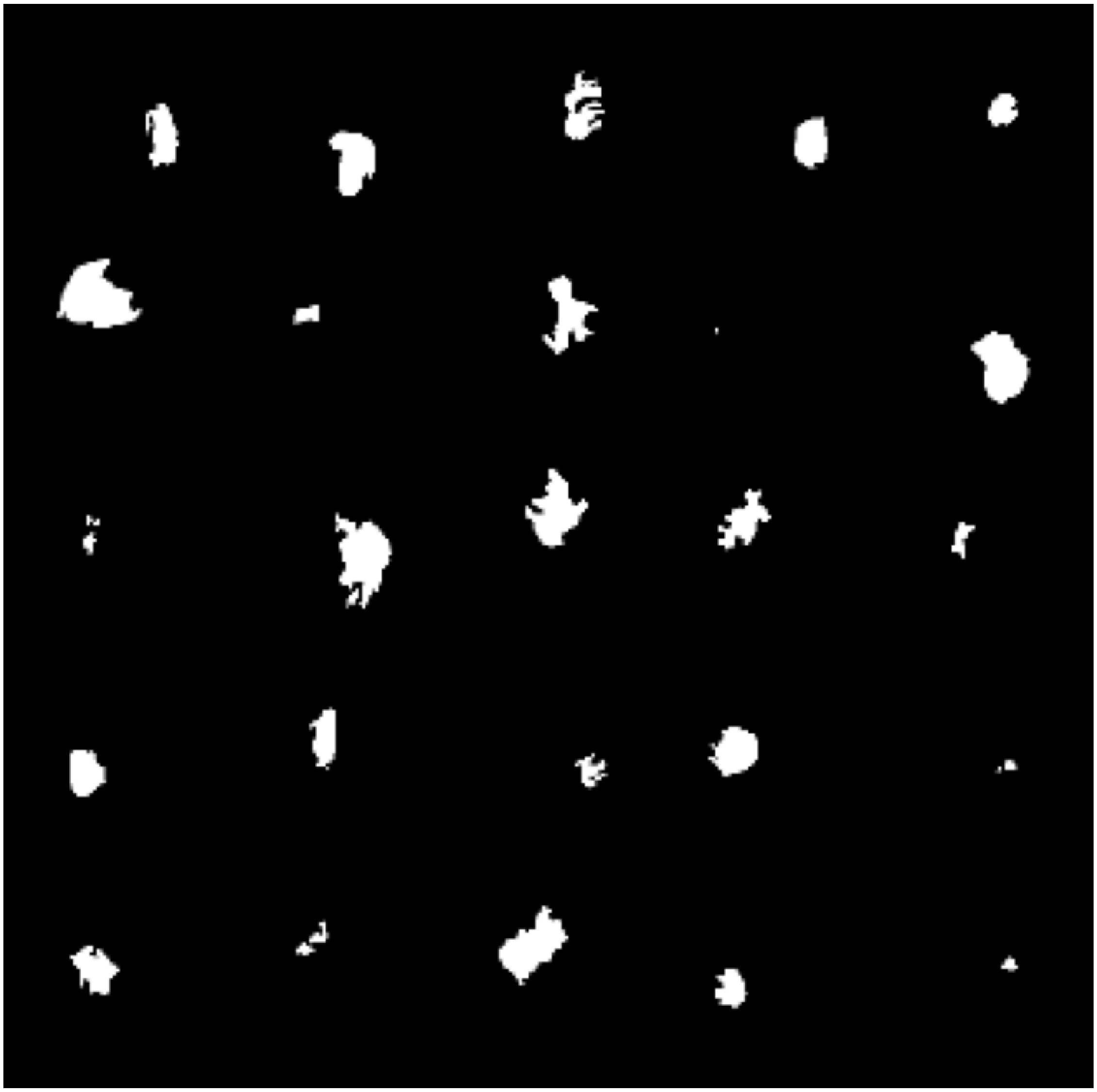
[14]:
```
print(images[0].shape, masks[0].shape)
```

```
torch.Size([64, 64, 3]) torch.Size([64, 64, 3])
```

[15]:
```
visualization = Visualization(train_df)
visualization.plot_images(images)
```



[16]:
```
visualization.plot_images(masks)
```



[17]:
```python
from scripts.gan import SimpleGAN

# Instantiate and train the GAN
gan = SimpleGAN(img_size=IMG_SIZE)
```

```
Metal device set to: Apple M1 Pro
Model: "Discriminator"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Discriminator-Hidden-Layer-  (None, 32, 32, 64)       3136
 1 (Conv2D)
```

```
 Discriminator-Hidden-Layer-  (None, 32, 32, 64)       0
 Activation-1 (LeakyReLU)

 Discriminator-Hidden-Layer-  (None, 16, 16, 128)      131200
 2 (Conv2D)

 Discriminator-Hidden-Layer-  (None, 16, 16, 128)      0
 Activation-2 (LeakyReLU)

 Discriminator-Hidden-Layer-  (None, 8, 8, 256)        524544
 3 (Conv2D)

 Discriminator-Hidden-Layer-  (None, 8, 8, 256)        0
 Activation-3 (LeakyReLU)

 Discriminator-Flatten-Layer  (None, 16384)            0
  (Flatten)

 Discriminator-Flatten-Layer  (None, 16384)            0
 -Dropout (Dropout)

 Discriminator-Output-Layer   (None, 1)                16385
 (Dense)

=================================================================
Total params: 675,265
Trainable params: 0
Non-trainable params: 675,265
_________________________________________________________________
Model: "Generator"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Generator-Hidden-Layer-1 (D  (None, 8192)             827392
 ense)

 Generator-Hidden-Layer-Resh  (None, 8, 8, 128)        0
 ape-1 (Reshape)

 Generator-Hidden-Layer-2 (C  (None, 16, 16, 128)      262272
 onv2DTranspose)

 Generator-Hidden-Layer-Acti  (None, 16, 16, 128)      0
 vation-2 (ReLU)

 Generator-Hidden-Layer-3 (C  (None, 32, 32, 256)      524544
 onv2DTranspose)
```

```
 Generator-Hidden-Layer-Acti  (None, 32, 32, 256)      0
 vation-3 (ReLU)

 Generator-Hidden-Layer-4 (C  (None, 64, 64, 512)      2097664
 onv2DTranspose)

 Generator-Hidden-Layer-Acti  (None, 64, 64, 512)      0
 vation-4 (ReLU)

 Generator-Output-Layer (Con  (None, 64, 64, 3)        38403
 v2D)

=================================================================
Total params: 3,750,275
Trainable params: 3,750,275
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your
model is starting with a freshly initialized optimizer.
WARNING:tensorflow:No training configuration found in the save file, so the
model was *not* compiled. Compile it manually.
Models loaded.
```

[18]:
```python
gan.train(images, epochs=5000, batch_size=128)
```

```
Models are already loaded. Training skipped.
```

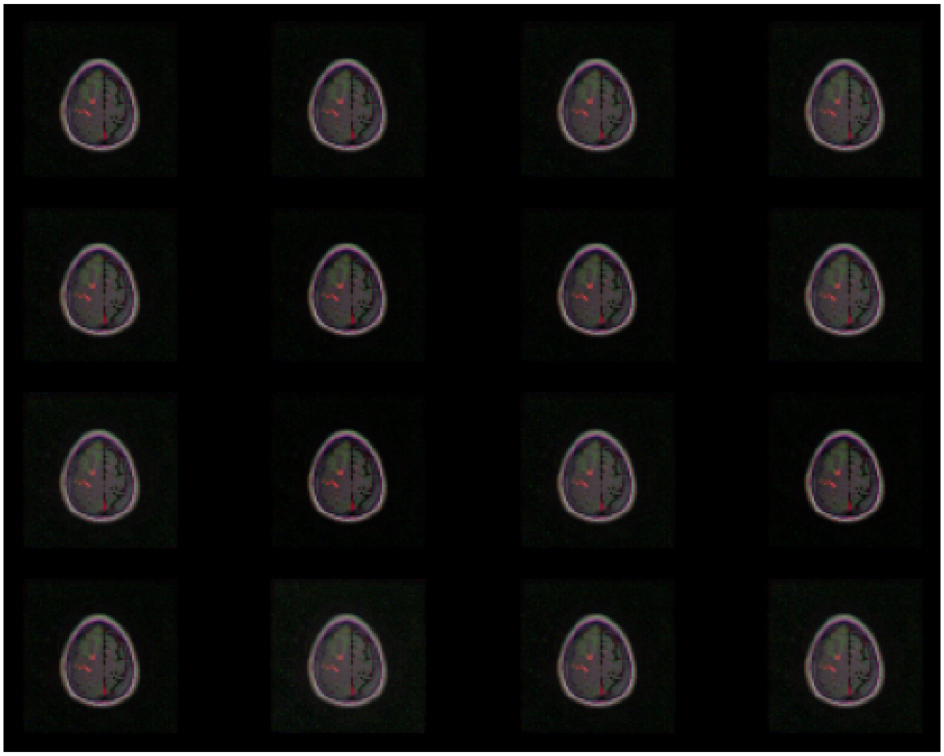
[19]:
```python
# Generate synthetic images
num_images = 16
generated_images = gan.generate_images(num_images)
```

```
1/1 [==============================] - 0s 79ms/step

2023-09-05 09:05:16.054403: W
tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU
frequency: 0 Hz
```

[20]:
```python
def display_images(images):
    fig, axs = plt.subplots(4,4)
    count = 0
    for i in range(4):
        for j in range(4):
            axs[i, j].imshow((images[count] * 0.5) + 0.5)
            axs[i, j].axis('off')
            count += 1
    plt.show()
```

[21]:
```python
# Display a few samples from the dataset
```

[22]:
```
# Display the generated images
display_images(generated_images)
```



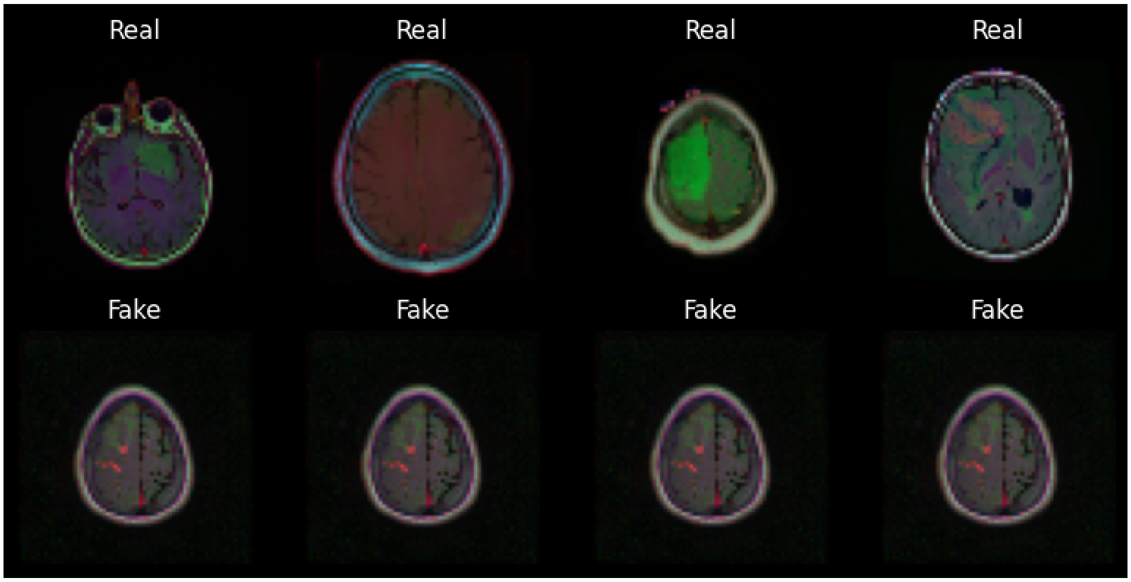
[23]:
```
print(generated_images.shape)
```

```
(16, 64, 64, 3)
```

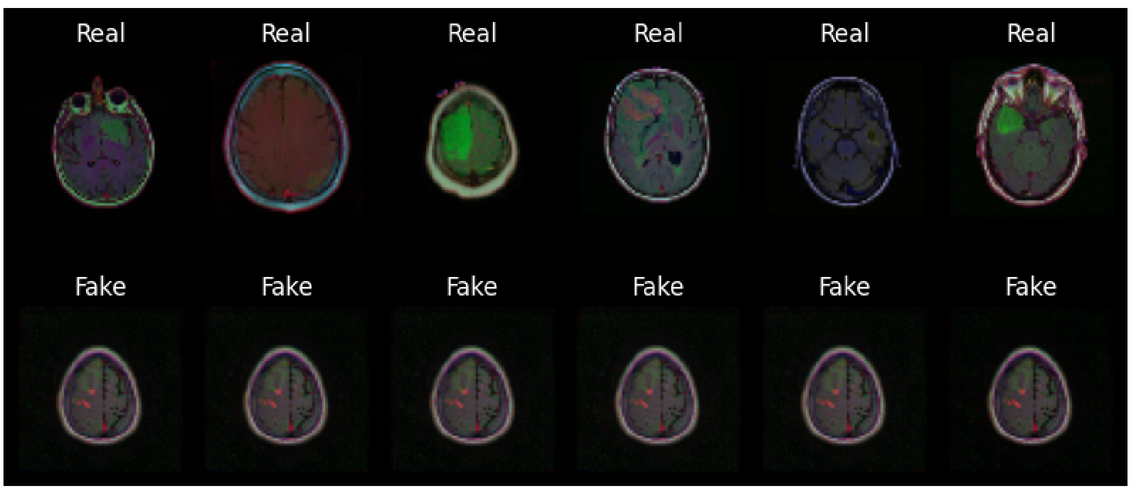
[24]:
```
test_images, test_masks, test_labels = next(iter(test_dataloader))
```

1 Model evaluation

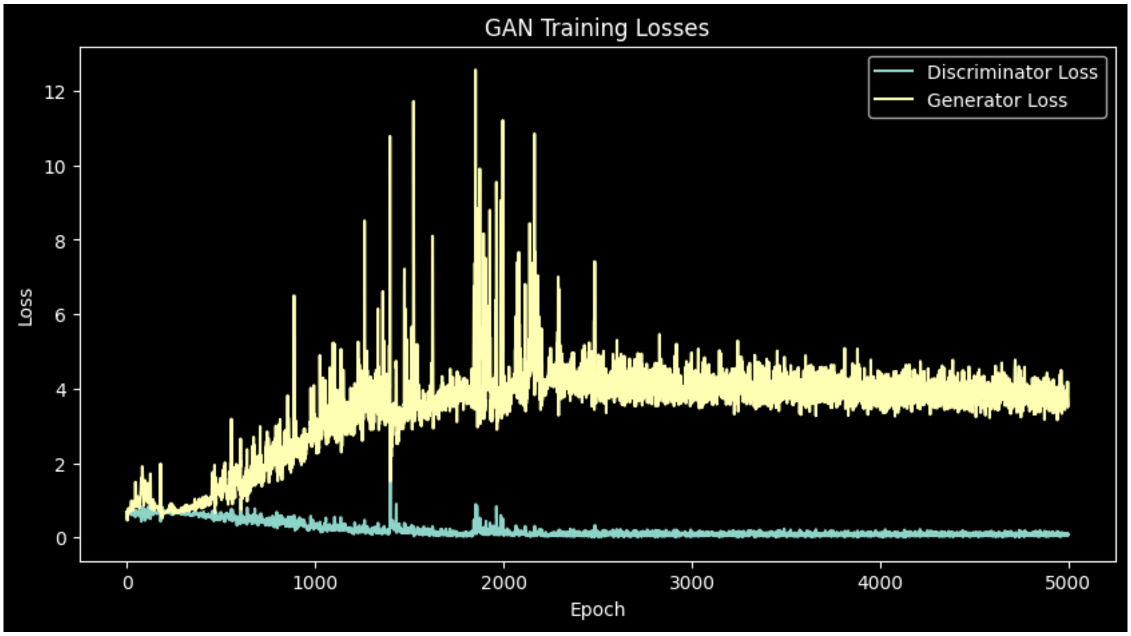
[25]:
```
gan.plot_real_vs_fake(test_images, 1000, 4)
```



[26]:
```
gan.plot_real_vs_fake(test_images, 1000, 6)
```



[27]:
```
gan.plot_losses()
```



2 Brain Cancer Classifier

The fundamental idea underlying this classifier involves a two-step training process. Initially, the model is trained using authentic data, followed by a subsequent training round where both authentic and synthetic data are used. This approach aims to assess whether the classifier's performance exhibits improvement after incorporating synthetic data alongside genuine data.

[28]:
```
from scripts.brain_cancer_classifier import BrainCancerClassifier
```

[29]:
```
classifier = BrainCancerClassifier()
```

Based on real images

[30]:
```
all_data = df
```

[31]:
```
# Split df into train_df and val_df
all_train_df, all_val_df = train_test_split(all_data, stratify=all_data.label,
 ↪test_size=0.1)
all_train_df = all_train_df.reset_index(drop=True)
all_val_df = all_val_df.reset_index(drop=True)

# Split train_df into train_df and test_df
all_val_df, all_test_df = train_test_split(all_val_df, stratify=all_val_df.
 ↪label, test_size=0.3)
all_val_df = all_val_df.reset_index(drop=True)
all_test_df = all_test_df.reset_index(drop=True)
```

```
print(f"Train: {all_train_df.shape} \nVal: {all_val_df.shape} \nTest:␣
↪{all_test_df.shape}")
```

```
Train: (3536, 3)
Val: (275, 3)
Test: (118, 3)
```

[32]:
```
IMG_SIZE = 64
BATCH_SIZE = 26

# train
all_train_dataset = BrainMriDataset(df=all_train_df, img_size=IMG_SIZE)
all_train_dataloader = DataLoader(all_train_dataset, batch_size=BATCH_SIZE,␣
↪num_workers=4, shuffle=True)

# val
all_val_dataset = BrainMriDataset(df=all_val_df, img_size=IMG_SIZE)
all_val_dataloader = DataLoader(all_val_dataset, batch_size=BATCH_SIZE,␣
↪num_workers=4, shuffle=True)

#test
all_test_dataset = BrainMriDataset(df=all_test_df, img_size=IMG_SIZE)
all_test_dataloader = DataLoader(all_test_dataset, batch_size=BATCH_SIZE,␣
↪num_workers=4, shuffle=True)
```

[33]:
```
all_train_images, all_train_masks, all_train_labels =␣
↪next(iter(all_train_dataloader))
```

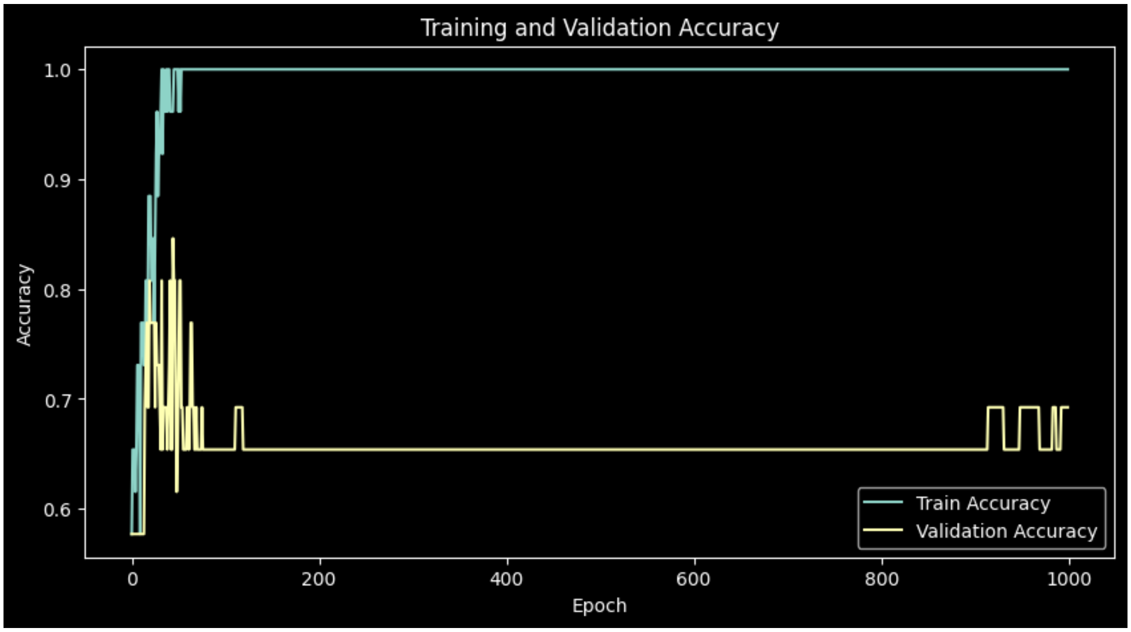
[34]:
```
all_val_images, all_val_masks, all_val_labels = next(iter(all_val_dataloader))
```

[35]:
```
classifier.train(all_train_images, all_train_labels, all_val_images,␣
↪all_val_labels)
```

```
Model saved.
```

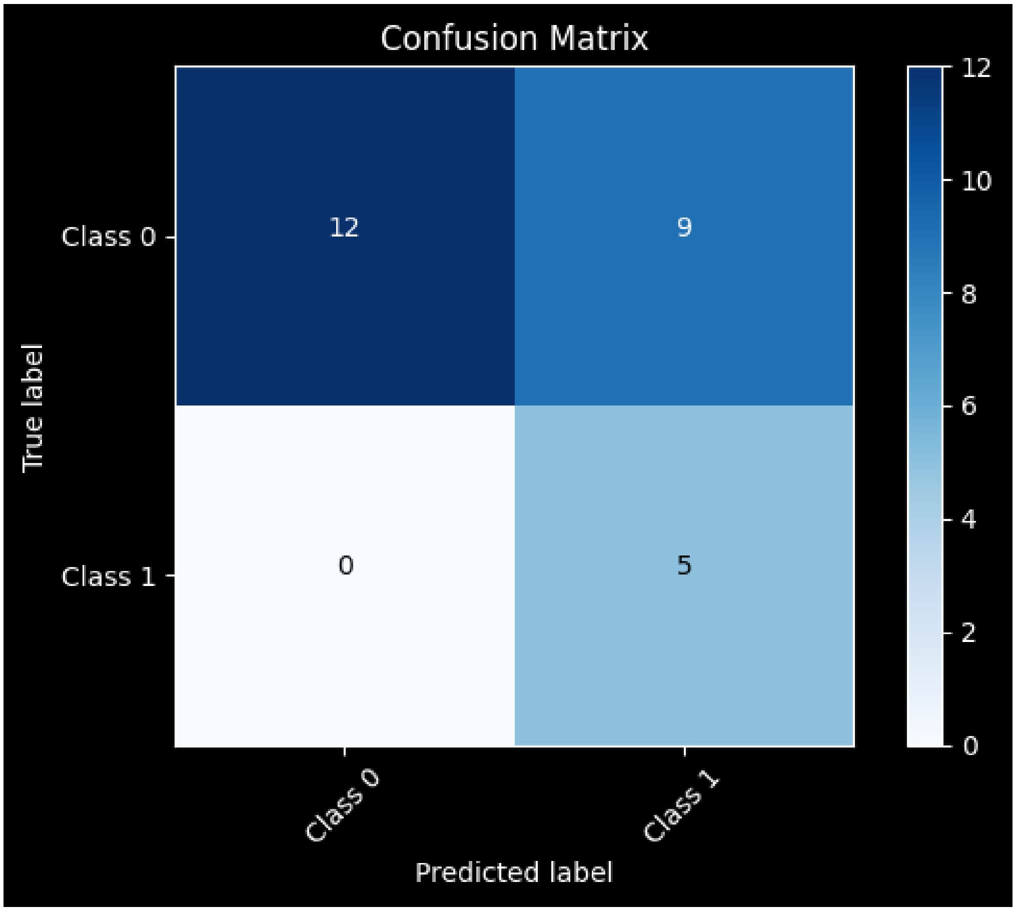
[36]:
```
all_test_images, all_test_masks, all_test_labels =␣
↪next(iter(all_test_dataloader))
```

[37]:
```
classifier.plot_training_history()
```



[38]:
```
classifier.evaluate(all_test_images, all_test_labels)
```

```
1/1 [==============================] - 0s 66ms/step - loss: 3.9830 - accuracy:
0.6538
Test Loss: 3.982983350753784
Test Accuracy: 0.6538462042808533
1/1 [==============================] - 0s 51ms/step
Confusion Matrix:
[[12  9]
 [ 0  5]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.57      0.73        21
           1       0.36      1.00      0.53         5

    accuracy                           0.65        26
   macro avg       0.68      0.79      0.63        26
weighted avg       0.88      0.65      0.69        26
```



[39]:
```
from scripts.brain_mri_dataset import CombinedBrainMriDataset
```

[40]:
```
IMG_SIZE = 64
BATCH_SIZE = 26

# Instantiate the CombinedBrainMriDataset
combined_dataset = CombinedBrainMriDataset(original_df=all_train_df,␣
↪generated_images=generated_images, img_size=IMG_SIZE)

# Create a DataLoader for the combined dataset
combined_dataloader = DataLoader(combined_dataset, batch_size=BATCH_SIZE,␣
↪shuffle=True)
```

Based on fake images

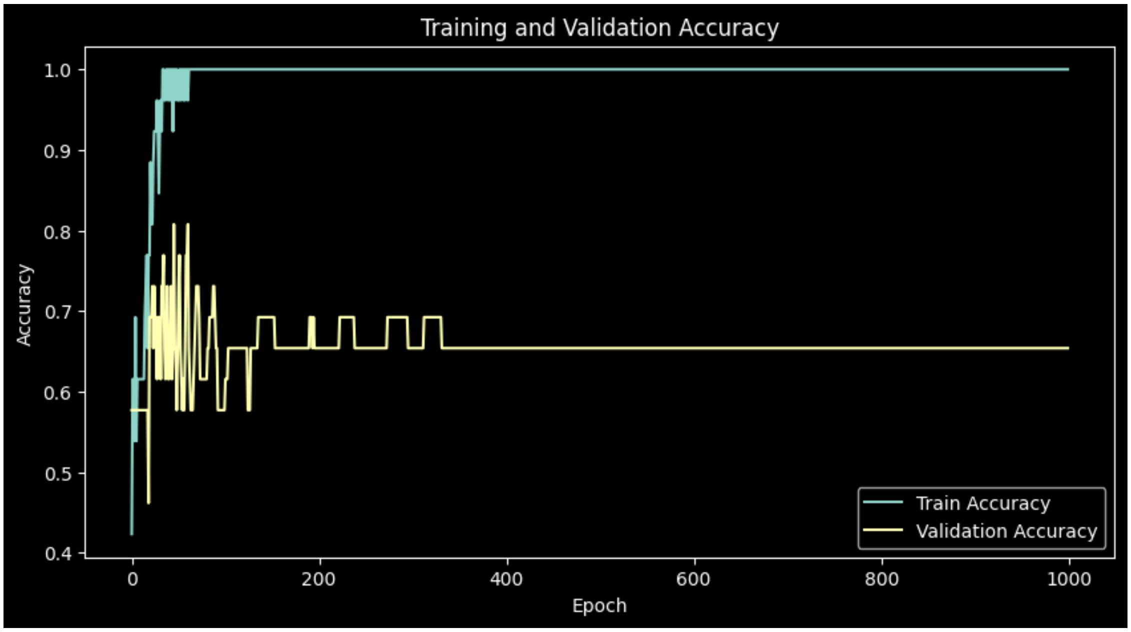
[41]:
```
balanced_train_images, balanced_train_masks, balanced_train_labels =␣
↪next(iter(combined_dataloader))
```

[42]:
```
classifier_with_fake_data = BrainCancerClassifier()
```

[43]:
```
classifier_with_fake_data.train(balanced_train_images, balanced_train_labels,
    all_val_images, all_val_labels)
```

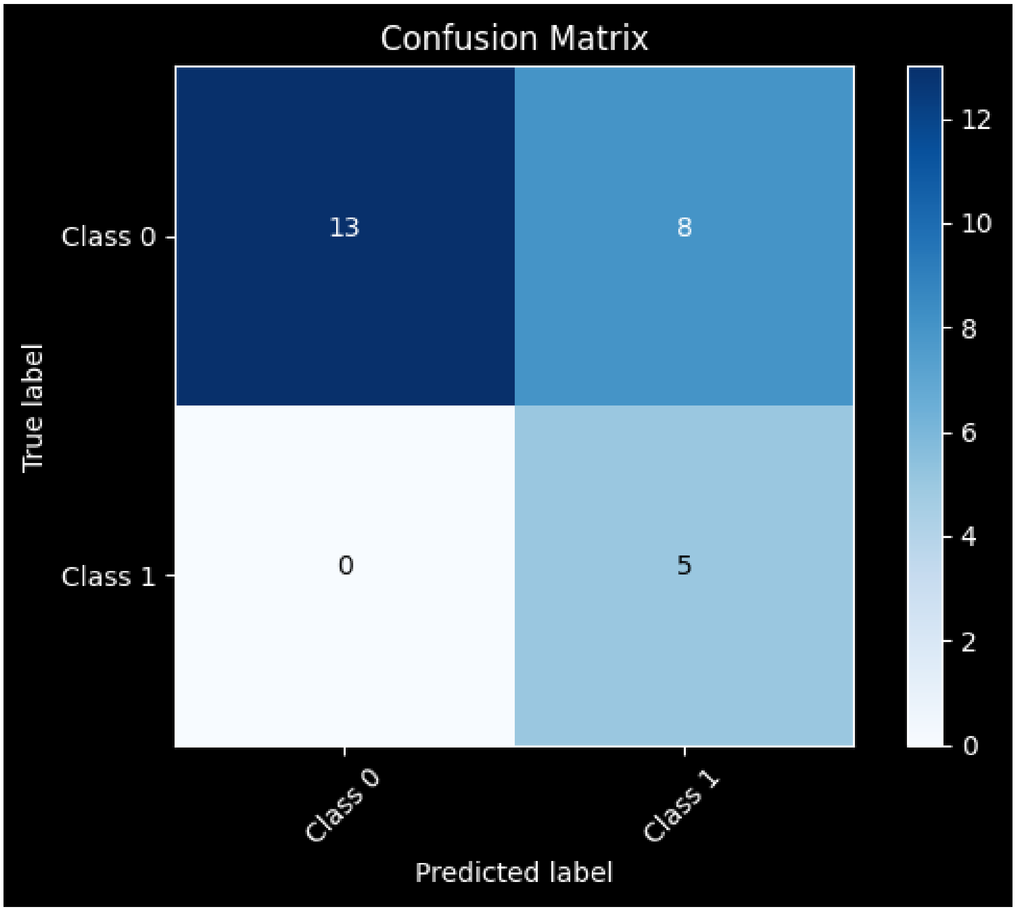
```
Model saved.
```

[44]:
```
classifier_with_fake_data.plot_training_history()
```



[45]:
```
classifier_with_fake_data.evaluate(all_test_images, all_test_labels)
```

```
1/1 [==============================] - 0s 20ms/step - loss: 4.0785 - accuracy:
0.6923
Test Loss: 4.078546524047852
Test Accuracy: 0.692307710647583
1/1 [==============================] - 0s 52ms/step
Confusion Matrix:
[[13  8]
 [ 0  5]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.62      0.76        21
           1       0.38      1.00      0.56         5

    accuracy                           0.69        26
   macro avg       0.69      0.81      0.66        26
weighted avg       0.88      0.69      0.72        26
```


Confusion Matrix
True label
Class 0
Class 1
13
8
0
5
12
10
8
6
4
2
0
Class 0
Class 1
Predicted label

```
[ ]:
```