Diffusion_model

December 16, 2023

[1]:
```
import os
import keras
from keras import layers
import numpy as np
import matplotlib.pyplot as plt

# If you need ops from TensorFlow, import it like this:
from tensorflow.python.ops import math_ops  # You can adjust the path based on␣
 ↪your TensorFlow version

import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow.keras.layers as L
import tensorflow_addons as tfa
import glob, random, os, warnings
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
```

```
C:\Users\nehag\anaconda3\lib\site-
packages\tensorflow_addons\utils\tfa_eol_msg.py:23: UserWarning:

TensorFlow Addons (TFA) has ended development and introduction of new features.
TFA has entered a minimal maintenance and release mode until a planned end of
life in May 2024.
Please modify downstream libraries to take dependencies from other repositories
in our TensorFlow community (e.g. Keras, Keras-CV, and Keras-NLP).

For more information see: https://github.com/tensorflow/addons/issues/2807

  warnings.warn(
C:\Users\nehag\anaconda3\lib\site-
packages\tensorflow_addons\utils\ensure_tf_install.py:53: UserWarning:
Tensorflow Addons supports using Python ops for all Tensorflow versions above or
equal to 2.12.0 and strictly below 2.15.0 (nightly versions are not supported).
 The versions of TensorFlow you are currently using is 2.10.0 and is not
supported.
```

```
Some things might work, some things might not.
If you were to encounter a bug, do not file an issue.
If you want to make sure you're using a tested and supported configuration,
either change the TensorFlow version or the TensorFlow Addons's version.
You can find the compatibility matrix in TensorFlow Addon's readme:
https://github.com/tensorflow/addons
  warnings.warn(
```

[2]:
```python
import os
import cv2
import numpy as np
import re

def load_images_from_folder(folder):
    images = []
    labels = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        if os.path.isfile(img_path):
            img = cv2.imread(img_path)
            if img is not None:
                images.append(img)
                # Extract the label from the filename using a regular expression
                match = re.search(r'\d{4}-\d{2}-\d{2}', filename)
                if match:
                    label = match.group(0)
                    labels.append(label)
    return np.array(images), np.array(labels)

# Replace 'YourDatasetFolder' with the actual path to your dataset folder
dataset_folder = 'images_old'

x_train, y_train = load_images_from_folder(dataset_folder)

# Print the size of the loaded data
print("Size of X_train:", len(x_train))
print("Size of y_train:", len(y_train))
```

```
Size of X_train: 744
Size of y_train: 744
```

```
C:\Users\nehag\AppData\Local\Temp\ipykernel_22688\3386135084.py:20:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  return np.array(images), np.array(labels)
```

[3]:
```
def data_augment(image):
    p_spatial = tf.random.uniform([], 0, 1.0, dtype = tf.float32)
    p_rotate = tf.random.uniform([], 0, 1.0, dtype = tf.float32)

    image = tf.image.random_flip_left_right(image)
    image = tf.image.random_flip_up_down(image)

    if p_spatial > .75:
        image = tf.image.transpose(image)

    # Rotates
    if p_rotate > .75:
        image = tf.image.rot90(image, k = 3) # rotate 270º
    elif p_rotate > .5:
        image = tf.image.rot90(image, k = 2) # rotate 180º
    elif p_rotate > .25:
        image = tf.image.rot90(image, k = 1) # rotate 90º

    return image
```

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Convert date labels to numerical values
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)

# Split the data into training and testing sets
X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(x_train,␣
 ↪y_train_encoded, test_size=0.2, random_state=42)

# Print the shapes of training and testing sets
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train_encoded.shape)
print("Shape of y_test:", y_test_encoded.shape)
```

```
Shape of X_train: (595,)
Shape of X_test: (149,)
Shape of y_train: (595,)
Shape of y_test: (149,)
```

[5]:
```
import numpy as np

# Print the size and shape of X_train
print("Size of X_train:", X_train.shape)
```

```
# Print the unique labels in y_train
print("Unique labels in y_train:", np.unique(y_train))

# Print the number of unique labels in y_train
print("Number of unique labels in y_train:", len(np.unique(y_train)))
```

```
Size of X_train: (595,)
Unique labels in y_train: ['2020-01-01' '2020-01-02' '2020-01-03' '2020-01-04'
'2020-01-05'
 '2020-01-06' '2020-01-07' '2020-01-08' '2020-01-09' '2020-01-10'
 '2020-01-11' '2020-01-12' '2020-01-13' '2020-01-14' '2020-01-15'
 '2020-01-16' '2020-01-17' '2020-01-18' '2020-01-19' '2020-01-20'
 '2020-01-21' '2020-01-22' '2020-01-23' '2020-01-24' '2020-01-25'
 '2020-01-26' '2020-01-27' '2020-01-28' '2020-01-29' '2020-01-30'
 '2020-01-31']
Number of unique labels in y_train: 31
```

[6]:
```
from sklearn.preprocessing import LabelEncoder

# Convert date strings to numeric labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train_encoded)

# Print the unique encoded labels in y_train
print("Unique encoded labels in y_train:", np.unique(y_train_encoded))

# Print the number of unique encoded labels in y_train
print("Number of unique encoded labels in y_train:", len(np.
  ↪unique(y_train_encoded)))
```

```
Unique encoded labels in y_train: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30]
Number of unique encoded labels in y_train: 31
```

[7]:
```
import cv2

# Resize images to (224, 224, 3)
X_train_resized = [cv2.resize(img, (224, 224)) for img in X_train]

# Convert the list to a NumPy array
X_train_resized = np.array(X_train_resized)

# Verify the new shape
print("New shape of X_train:", X_train_resized.shape)
```

```
New shape of X_train: (595, 224, 224, 3)
```

[8]:
```
import cv2

# Resize images to (224, 224, 3)
X_test_resized = [cv2.resize(img, (224, 224)) for img in X_test]

# Convert the list to a NumPy array
X_test_resized = np.array(X_test_resized)

# Verify the new shape
print("New shape of X_train:", X_test_resized.shape)
```

```
New shape of X_train: (149, 224, 224, 3)
```

[9]:
```
print(f"x_train shape: {X_train_resized.shape} - y_train shape:␣
 ↪{y_train_encoded.shape}")
print(f"x_test shape: {X_test_resized.shape} - y_test shape: {y_test_encoded.
 ↪shape}")
```

```
x_train shape: (595, 224, 224, 3) - y_train shape: (595,)
x_test shape: (149, 224, 224, 3) - y_test shape: (149,)
```

[10]:
```
import numpy as np
import pandas as pd

# Assuming you have an array of image IDs
image_ids = y_train_encoded

# Reshape the 4D array into a 2D array
reshaped_array = X_train_resized.reshape(X_train_resized.shape[0], -1)

# Create a DataFrame with the reshaped array
df = pd.DataFrame(reshaped_array, columns=[f'pixel_{i}' for i in␣
 ↪range(reshaped_array.shape[1])])

# Add image IDs to the DataFrame
df['image_id'] = image_ids
```

[11]:
```
x_train=X_train_resized
x_test=X_test_resized
y_train=y_train_encoded
y_test=y_test_encoded
```

[12]:
```
import keras
import tensorflow as tf

import tensorflow.keras
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten,␣
 ↪Conv2D, MaxPooling2D
from tensorflow.keras.layers import Input, Add, Dense, Activation,␣
 ↪ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D,␣
 ↪MaxPooling2D, GlobalMaxPooling2D,MaxPool2D
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras import layers
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.preprocessing import image
from keras.callbacks import ReduceLROnPlateau
```

1 Implementation of the Diffusion architecture

[20]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(595, 224, 224, 3)
(149, 224, 224, 3)
(595,)
(149,)
```

[19]:
```
from tensorflow.keras import layers, models
def build_diffusion_model(input_shape=(224, 224, 3), num_classes=31):
    model = models.Sequential()

    model.add(layers.Conv2D(64, (3, 3), activation='relu',␣
 ↪input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(256, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))

    return model
# Create the model
diffusion_model = build_diffusion_model()

# Define the learning rate
learning_rate = 0.001  # You can adjust this value based on your requirements

# Create an instance of the Adam optimizer with the specified learning rate
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

```
# Compile the model with the Adam optimizer
diffusion_model.compile(optimizer=adam_optimizer,
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy'],
                        run_eagerly=True)



diffusion_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 222, 222, 64)      1792

 max_pooling2d_2 (MaxPooling  (None, 111, 111, 64)     0
 2D)

 conv2d_4 (Conv2D)           (None, 109, 109, 128)     73856

 max_pooling2d_3 (MaxPooling  (None, 54, 54, 128)      0
 2D)

 conv2d_5 (Conv2D)           (None, 52, 52, 256)       295168

 flatten_1 (Flatten)         (None, 692224)            0

 dense_2 (Dense)             (None, 512)               354419200

 dense_3 (Dense)             (None, 31)                15903

=================================================================
Total params: 354,805,919
Trainable params: 354,805,919
Non-trainable params: 0
_________________________________________________________________
```

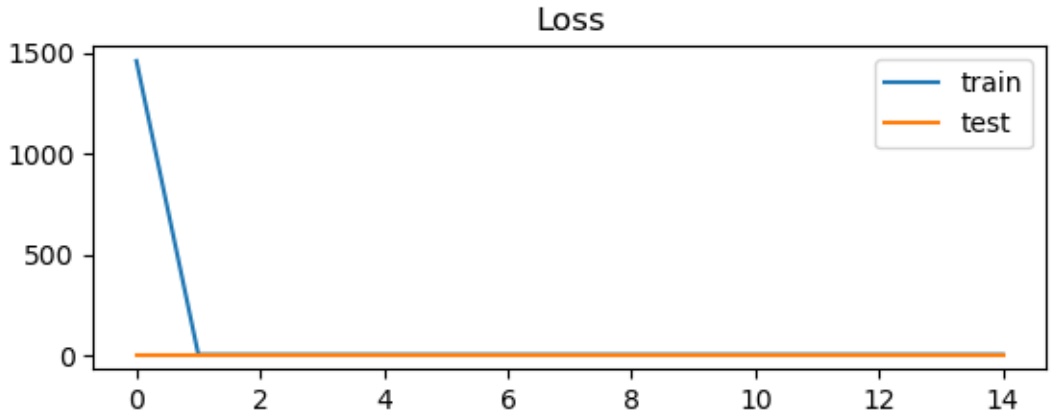
[21]:
```
from tensorflow.keras.callbacks import ReduceLROnPlateau
rlp = ReduceLROnPlateau(factor=0.5, patience=3)  # Adjust parameters as needed

history = diffusion_model.fit(
    x_train, y_train,
    epochs=15,
    callbacks=[rlp],
    validation_split=0.10
```

```
)
```

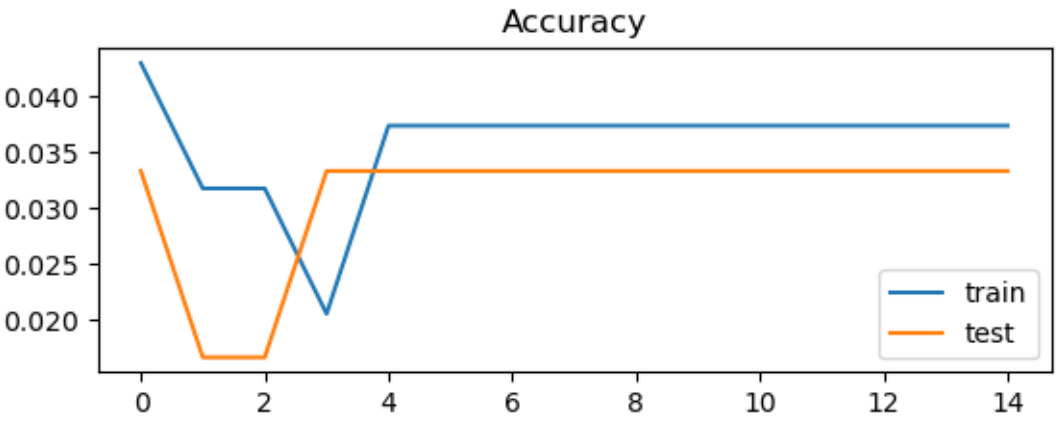
```
Epoch 1/15
17/17 [==============================] - 59s 3s/step - loss: 1459.5288 -
accuracy: 0.0430 - val_loss: 3.4334 - val_accuracy: 0.0333 - lr: 0.0010
Epoch 2/15
17/17 [==============================] - 58s 3s/step - loss: 3.4355 - accuracy:
0.0318 - val_loss: 3.4337 - val_accuracy: 0.0167 - lr: 0.0010
Epoch 3/15
17/17 [==============================] - 59s 3s/step - loss: 3.4341 - accuracy:
0.0318 - val_loss: 3.4340 - val_accuracy: 0.0167 - lr: 0.0010
Epoch 4/15
17/17 [==============================] - 58s 3s/step - loss: 3.4339 - accuracy:
0.0206 - val_loss: 3.4348 - val_accuracy: 0.0333 - lr: 0.0010
Epoch 5/15
17/17 [==============================] - 58s 3s/step - loss: 3.4334 - accuracy:
0.0374 - val_loss: 3.4349 - val_accuracy: 0.0333 - lr: 5.0000e-04
Epoch 6/15
17/17 [==============================] - 58s 3s/step - loss: 3.4333 - accuracy:
0.0374 - val_loss: 3.4352 - val_accuracy: 0.0333 - lr: 5.0000e-04
Epoch 7/15
17/17 [==============================] - 58s 3s/step - loss: 3.4331 - accuracy:
0.0374 - val_loss: 3.4356 - val_accuracy: 0.0333 - lr: 5.0000e-04
Epoch 8/15
17/17 [==============================] - 59s 3s/step - loss: 3.4329 - accuracy:
0.0374 - val_loss: 3.4359 - val_accuracy: 0.0333 - lr: 2.5000e-04
Epoch 9/15
17/17 [==============================] - 58s 3s/step - loss: 3.4328 - accuracy:
0.0374 - val_loss: 3.4359 - val_accuracy: 0.0333 - lr: 2.5000e-04
Epoch 10/15
17/17 [==============================] - 59s 3s/step - loss: 3.4328 - accuracy:
0.0374 - val_loss: 3.4361 - val_accuracy: 0.0333 - lr: 2.5000e-04
Epoch 11/15
17/17 [==============================] - 59s 3s/step - loss: 3.4326 - accuracy:
0.0374 - val_loss: 3.4362 - val_accuracy: 0.0333 - lr: 1.2500e-04
Epoch 12/15
17/17 [==============================] - 59s 3s/step - loss: 3.4326 - accuracy:
0.0374 - val_loss: 3.4362 - val_accuracy: 0.0333 - lr: 1.2500e-04
Epoch 13/15
17/17 [==============================] - 59s 3s/step - loss: 3.4326 - accuracy:
0.0374 - val_loss: 3.4363 - val_accuracy: 0.0333 - lr: 1.2500e-04
Epoch 14/15
17/17 [==============================] - 59s 3s/step - loss: 3.4325 - accuracy:
0.0374 - val_loss: 3.4364 - val_accuracy: 0.0333 - lr: 6.2500e-05
Epoch 15/15
17/17 [==============================] - 59s 3s/step - loss: 3.4325 - accuracy:
0.0374 - val_loss: 3.4364 - val_accuracy: 0.0333 - lr: 6.2500e-05
```

[22]:
```
plt.subplot(211)
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss']
         , label='test')
plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x25406b24280>



[23]:
```
plt.subplot(212)
plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show()
```



[24]:
```
x_test.shape
```

[24]:
```
(149, 224, 224, 3)
```

[25]:
```
y_test.shape
```

[25]:
```
(149,)
```

[26]:
```
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

# Assuming you have your model defined, data loaded (x_test, y_test), and model␣
 ↪trained

# Evaluate the model on the test set
test_loss, test_accuracy = diffusion_model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

# Make predictions on the test set
y_pred_probs = diffusion_model.predict(x_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Convert true labels to class indices if needed
y_true = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_test

# Print classification report
print("Classification Report:")
print(classification_report(y_true, y_pred))

# Calculate and print confusion matrix
conf_mat = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(conf_mat)
```

```
5/5 [==============================] - 4s 812ms/step - loss: 3.4395 - accuracy:
0.0134
Test Loss: 3.4395
Test Accuracy: 0.0134
5/5 [==============================] - 4s 816ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         8
           3       0.00      0.00      0.00         9
```

```
              4       0.00      0.00      0.00         4
              5       0.00      0.00      0.00         3
              6       0.00      0.00      0.00         5
              7       0.01      1.00      0.03         2
              8       0.00      0.00      0.00         8
              9       0.00      0.00      0.00         4
             10       0.00      0.00      0.00         6
             11       0.00      0.00      0.00         2
             12       0.00      0.00      0.00         5
             13       0.00      0.00      0.00         7
             14       0.00      0.00      0.00         4
             15       0.00      0.00      0.00         7
             16       0.00      0.00      0.00         6
             17       0.00      0.00      0.00         9
             18       0.00      0.00      0.00         6
             19       0.00      0.00      0.00         2
             20       0.00      0.00      0.00         4
             21       0.00      0.00      0.00         4
             22       0.00      0.00      0.00         4
             23       0.00      0.00      0.00         3
             24       0.00      0.00      0.00         5
             25       0.00      0.00      0.00         6
             26       0.00      0.00      0.00         3
             27       0.00      0.00      0.00         3
             28       0.00      0.00      0.00         4
             29       0.00      0.00      0.00         3
             30       0.00      0.00      0.00         5

    accuracy                           0.01       149
   macro avg       0.00      0.03      0.00       149
weighted avg       0.00      0.01      0.00       149

Confusion Matrix:
[[0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
 [0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[27]:
```
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

# Assuming you have your model defined, data loaded (x_test, y_test), and model␣
 ↪trained

# Evaluate the model on the test set
test_loss, test_accuracy = diffusion_model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

# Make predictions on the test set
```

```
y_pred_probs = diffusion_model.predict(x_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Convert true labels to class indices if needed
y_true = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_test

# Print classification report
print("Classification Report:")
print(classification_report(y_true, y_pred))

# Calculate and print confusion matrix
conf_mat = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(conf_mat)

# Extract precision, recall, and F1-score from the classification report
precision = classification_report(y_true, y_pred, output_dict=True)['weighted␣
 ↪avg']['precision']
recall = classification_report(y_true, y_pred, output_dict=True)['weighted␣
 ↪avg']['recall']
f1_score = classification_report(y_true, y_pred, output_dict=True)['weighted␣
 ↪avg']['f1-score']

# Calculate accuracy and sensibility
accuracy = np.sum(y_true == y_pred) / len(y_true)
sensibility = recall  # Sensibility is the same as recall in binary␣
 ↪classification

# Print accuracy, sensibility, precision, recall, and F1-score
print(f'Accuracy: {accuracy:.4f}')
print(f'Sensibility: {sensibility:.4f}')
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1_score:.4f}')
```

```
5/5 [==============================] - 4s 818ms/step - loss: 3.4395 - accuracy:
0.0134
Test Loss: 3.4395
Test Accuracy: 0.0134
5/5 [==============================] - 4s 818ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         8
           3       0.00      0.00      0.00         9
```

```
              4       0.00      0.00      0.00         4
              5       0.00      0.00      0.00         3
              6       0.00      0.00      0.00         5
              7       0.01      1.00      0.03         2
              8       0.00      0.00      0.00         8
              9       0.00      0.00      0.00         4
             10       0.00      0.00      0.00         6
             11       0.00      0.00      0.00         2
             12       0.00      0.00      0.00         5
             13       0.00      0.00      0.00         7
             14       0.00      0.00      0.00         4
             15       0.00      0.00      0.00         7
             16       0.00      0.00      0.00         6
             17       0.00      0.00      0.00         9
             18       0.00      0.00      0.00         6
             19       0.00      0.00      0.00         2
             20       0.00      0.00      0.00         4
             21       0.00      0.00      0.00         4
             22       0.00      0.00      0.00         4
             23       0.00      0.00      0.00         3
             24       0.00      0.00      0.00         5
             25       0.00      0.00      0.00         6
             26       0.00      0.00      0.00         3
             27       0.00      0.00      0.00         3
             28       0.00      0.00      0.00         4
             29       0.00      0.00      0.00         3
             30       0.00      0.00      0.00         5

    accuracy                           0.01       149
   macro avg       0.00      0.03      0.00       149
weighted avg       0.00      0.01      0.00       149

Confusion Matrix:
[[0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
 [0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.0134
Sensibility: 0.0134
Precision: 0.0002
Recall: 0.0134
F1 Score: 0.0004
```

```
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
```

```
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nehag\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]: