2_CNN_spectrogram_classifier

April 23, 2023

[4]:
```
!pip install pydub
!pip install librosa
!pip install numba==0.49.0
!pip install llvmlite==0.32.1
```

```
Requirement already satisfied: pydub in c:\users\hocke\anaconda3\lib\site-
packages (0.25.1)
Requirement already satisfied: librosa in c:\users\hocke\anaconda3\lib\site-
packages (0.8.1)
Requirement already satisfied: soundfile>=0.10.2 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (0.10.3.post1)
Requirement already satisfied: numba>=0.43.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (0.53.1)
Requirement already satisfied: packaging>=20.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (20.9)
Requirement already satisfied: decorator>=3.0.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (5.0.6)
Requirement already satisfied: joblib>=0.14 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (1.0.1)
Requirement already satisfied: scipy>=1.0.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (1.6.2)
Requirement already satisfied: resampy>=0.2.2 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (0.2.2)
Requirement already satisfied: audioread>=2.0.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (2.1.9)
Requirement already satisfied: pooch>=1.0 in c:\users\hocke\anaconda3\lib\site-
packages (from librosa) (1.5.2)
Requirement already satisfied: scikit-learn!=0.19.0,>=0.14.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (0.24.1)
Requirement already satisfied: numpy>=1.15.0 in
c:\users\hocke\anaconda3\lib\site-packages (from librosa) (1.20.1)
Requirement already satisfied: llvmlite<0.37,>=0.36.0rc1 in
c:\users\hocke\anaconda3\lib\site-packages (from numba>=0.43.0->librosa)
(0.36.0)
Requirement already satisfied: setuptools in c:\users\hocke\anaconda3\lib\site-
packages (from numba>=0.43.0->librosa) (52.0.0.post20210125)
Requirement already satisfied: pyparsing>=2.0.2 in
c:\users\hocke\anaconda3\lib\site-packages (from packaging>=20.0->librosa)
```

```
(2.4.7)
Requirement already satisfied: requests in c:\users\hocke\anaconda3\lib\site-
packages (from pooch>=1.0->librosa) (2.25.1)
Requirement already satisfied: appdirs in c:\users\hocke\anaconda3\lib\site-
packages (from pooch>=1.0->librosa) (1.4.4)
Requirement already satisfied: six>=1.3 in c:\users\hocke\anaconda3\lib\site-
packages (from resampy>=0.2.2->librosa) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\hocke\anaconda3\lib\site-packages (from scikit-
learn!=0.19.0,>=0.14.0->librosa) (2.1.0)
Requirement already satisfied: cffi>=1.0 in c:\users\hocke\anaconda3\lib\site-
packages (from soundfile>=0.10.2->librosa) (1.14.5)
Requirement already satisfied: pycparser in c:\users\hocke\anaconda3\lib\site-
packages (from cffi>=1.0->soundfile>=0.10.2->librosa) (2.20)
Requirement already satisfied: chardet<5,>=3.0.2 in
c:\users\hocke\anaconda3\lib\site-packages (from requests->pooch>=1.0->librosa)
(4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\hocke\anaconda3\lib\site-packages (from requests->pooch>=1.0->librosa)
(1.26.4)
Requirement already satisfied: idna<3,>=2.5 in
c:\users\hocke\anaconda3\lib\site-packages (from requests->pooch>=1.0->librosa)
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\hocke\anaconda3\lib\site-packages (from requests->pooch>=1.0->librosa)
(2020.12.5)
Collecting numba==0.49.0
  Using cached numba-0.49.0-cp38-cp38-win_amd64.whl (2.1 MB)
Collecting llvmlite<=0.33.0.dev0,>=0.31.0.dev0
  Using cached llvmlite-0.32.1-cp38-cp38-win_amd64.whl (13.6 MB)
Requirement already satisfied: numpy>=1.15 in c:\users\hocke\anaconda3\lib\site-
packages (from numba==0.49.0) (1.20.1)
Requirement already satisfied: setuptools in c:\users\hocke\anaconda3\lib\site-
packages (from numba==0.49.0) (52.0.0.post20210125)
Installing collected packages: llvmlite, numba
  Attempting uninstall: llvmlite
    Found existing installation: llvmlite 0.36.0

ERROR: Cannot uninstall 'llvmlite'. It is a distutils installed project and thus
we cannot accurately determine which files belong to it which would lead to only
a partial uninstall.

Collecting llvmlite==0.32.1
  Using cached llvmlite-0.32.1-cp38-cp38-win_amd64.whl (13.6 MB)
Installing collected packages: llvmlite
  Attempting uninstall: llvmlite
    Found existing installation: llvmlite 0.36.0

ERROR: Cannot uninstall 'llvmlite'. It is a distutils installed project and thus
```

we cannot accurately determine which files belong to it which would lead to only a partial uninstall.

[5]:
```
import numpy as np
import os
import matplotlib.pyplot as plt
import librosa
import random
import shutil
from pydub import AudioSegment
from matplotlib.backends.backend_agg import FigureCanvasAgg
from tensorflow.keras.optimizers import Adam
from keras.layers import Input, Dense, Activation, BatchNormalization, Flatten,␣
↪Conv2D, MaxPooling2D, Dropout
from keras.models import Model
from keras.initializers import glorot_uniform
from keras.preprocessing.image import ImageDataGenerator
import keras.backend as K

# COLAB = True
COLAB = False
```

[6]:
```
if COLAB:
    from google.colab import drive
    drive.mount('/content/gdrive')
    try:
        os.makedirs('/content/gdrive/MyDrive/Data/spectrograms')
        os.makedirs('/content/gdrive/MyDrive/Data/audio_samples')
    except FileExistsError:
        pass
else:
    try:
        os.makedirs('./spectrograms')
        os.makedirs('./audio_samples')
    except FileExistsError:
        pass
```

[7]:
```
genres = 'blues classical country disco pop hiphop metal reggae rock'
genres = genres.split()
```

[8]:
```
if COLAB:
    for g in genres:
        path1 = os.path.join('/content/gdrive/MyDrive/Data/audio_samples',␣
↪f'{g}')
        try:
            os.makedirs(path1)
        except FileExistsError:
```

```
            pass
        path = os.path.join(
            '/content/gdrive/MyDrive/Data/spectrograms', f'{g}')
        try:
            os.makedirs(path)
        except FileExistsError:
            pass
else:
    for g in genres:
        path1 = os.path.join('./audio_samples', f'{g}')
        try:
            os.makedirs(path1)
        except FileExistsError:
            pass
        path = os.path.join('./spectrograms', f'{g}')
        try:
            os.makedirs(path)
        except FileExistsError:
            pass
```

[44]:
```
i = 0
for g in genres:
    j = 0
    print(f"{g}")
    if COLAB:
        directory = '/content/gdrive/MyDrive/Data/genres_original'
    else:
        directory = './genres_original'
    for filename in os.listdir(os.path.join(directory, f"{g}")):
        song = os.path.join(f'{directory}/{g}', f'{filename}')
        j += 1
        for w in range(0, 10):
            i += 1
            t1 = 3 * w * 1000
            t2 = 3 * (w + 1) * 1000
            newAudio = AudioSegment.from_wav(song)
            new = newAudio[t1:t2]
            if COLAB:
                new.export(
                    f'/content/gdrive/MyDrive/Data/audio_samples/{g}/{g}{j}{w}.
 ↪wav', format="wav")
            else:
                new.export(
                    f'./audio_samples/{g}/{g}{j}{w}.wav', format="wav")
```

```
blues
classical
```

```
country
disco
pop
hiphop
metal
reggae
rock
```

[45]:
```
if COLAB:
    directory = '/content/gdrive/MyDrive/Data/audio_samples'
else:
    directory = './audio_samples'
for g in genres:
    j = 0
    print(g)
    for filename in os.listdir(os.path.join(directory, f"{g}")):
        song = os.path.join(f'{directory}/{g}', f'{filename}')
        j = j+1
        y, sr = librosa.load(song, duration=3)
        mels = librosa.feature.melspectrogram(y=y, sr=sr)
        fig = plt.Figure()
        canvas = FigureCanvasAgg(fig)
        p = plt.imshow(librosa.power_to_db(mels, ref=np.max))
        if COLAB:
            plt.savefig(
                f'/content/gdrive/MyDrive/Data/spectrograms/train/{g}/{g}{j}.
 ↪png')
        else:
            plt.savefig(f'./spectrograms/train/{g}/{g}{j}.png')
```

```
blues
classical
country
```

[9]:
```
# Split data into testing and training
directory = './spectrograms/train/'
print(directory)
if COLAB:
    directory = "/content/gdrive/MyDrive/Data/spectrograms/train/"
for g in genres:
    filenames = os.listdir(os.path.join(directory, f"{g}"))
    random.shuffle(filenames)
    test_files = filenames[0:100]
    for f in test_files:
        shutil.move(f"{directory}{g}/{f}", f"{directory[:-6]}test/{g}")
```

```
./spectrograms/train/
```

[10]:
```
train_directory = directory
train_data_generator = ImageDataGenerator(rescale=1./255)
train_generator = train_data_generator.flow_from_directory(train_directory,␣
↪target_size=(
    288, 432), color_mode="rgba", class_mode='categorical', batch_size=128)

validation_directory = f"{directory[:-6]}test/"
validation_data_generator = ImageDataGenerator(rescale=1./255)
validation_generator = validation_data_generator.
↪flow_from_directory(validation_directory, target_size=(
    288, 432), color_mode='rgba', class_mode='categorical', batch_size=128)
```

```
Found 7405 images belonging to 9 classes.
Found 900 images belonging to 9 classes.
```

[11]:
```
def cnn(input_shape=(288, 432, 4), classes=9):
    def step(dim, X):
        X = Conv2D(dim, kernel_size=(3, 3), strides=(1, 1))(X)
        X = BatchNormalization(axis=3)(X)
        X = Activation('relu')(X)
        return MaxPooling2D((2, 2))(X)
    X_input = Input(input_shape)
    X = X_input
    layer_dims = [8, 16, 32, 64, 128, 256]
    for dim in layer_dims:
        X = step(dim, X)

    X = Flatten()(X)
    X = Dropout(rate=0.3)(X)
    X = Dense(classes, activation='softmax',
              name=f'fc{classes}', ␣
↪kernel_initializer=glorot_uniform(seed=9))(X)
    model = Model(inputs=X_input, outputs=X, name='cnn')
    return model

def f1_score(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2 * (precision * recall) / (precision + recall + K.epsilon())
    return f1_val
```

[12]:
```
model = cnn(input_shape=(288, 432, 4), classes=9)
opt = Adam(learning_rate=0.00005)
```

```python
model.compile(optimizer=opt, loss='categorical_crossentropy',␣
↪metrics=['accuracy', f1_score])
model.summary()
```

```
Model: "cnn"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 288, 432, 4)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 286, 430, 8)       296
_________________________________________________________________
batch_normalization (BatchNo (None, 286, 430, 8)       32
_________________________________________________________________
activation (Activation)      (None, 286, 430, 8)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 143, 215, 8)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 141, 213, 16)      1168
_________________________________________________________________
batch_normalization_1 (Batch (None, 141, 213, 16)      64
_________________________________________________________________
activation_1 (Activation)    (None, 141, 213, 16)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 70, 106, 16)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 68, 104, 32)       4640
_________________________________________________________________
batch_normalization_2 (Batch (None, 68, 104, 32)       128
_________________________________________________________________
activation_2 (Activation)    (None, 68, 104, 32)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 34, 52, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 50, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 32, 50, 64)        256
_________________________________________________________________
activation_3 (Activation)    (None, 32, 50, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 25, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 23, 128)       73856
_________________________________________________________________
batch_normalization_4 (Batch (None, 14, 23, 128)       512
_________________________________________________________________
activation_4 (Activation)    (None, 14, 23, 128)       0
```

```
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 11, 128)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 5, 9, 256)         295168
_________________________________________________________________
batch_normalization_5 (Batch (None, 5, 9, 256)         1024
_________________________________________________________________
activation_5 (Activation)    (None, 5, 9, 256)         0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 4, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
fc9 (Dense)                  (None, 9)                 18441
=================================================================
Total params: 414,081
Trainable params: 413,073
Non-trainable params: 1,008
_________________________________________________________________
```

[13]:

```
history = model.fit(train_generator, epochs=100,␣
↪validation_data=validation_generator)
```

```
Epoch 1/100
58/58 [==============================] - 329s 6s/step - loss: 2.2779 - accuracy:
0.2211 - f1_score: 0.1139 - val_loss: 2.2322 - val_accuracy: 0.1111 -
val_f1_score: 0.0000e+00
Epoch 2/100
58/58 [==============================] - 325s 6s/step - loss: 1.8132 - accuracy:
0.3441 - f1_score: 0.2636 - val_loss: 2.3169 - val_accuracy: 0.1111 -
val_f1_score: 0.0000e+00
Epoch 3/100
58/58 [==============================] - 330s 6s/step - loss: 1.5687 - accuracy:
0.4336 - f1_score: 0.3524 - val_loss: 2.4569 - val_accuracy: 0.1111 -
val_f1_score: 0.0000e+00
Epoch 4/100
58/58 [==============================] - 325s 6s/step - loss: 1.4188 - accuracy:
0.4852 - f1_score: 0.4210 - val_loss: 2.6208 - val_accuracy: 0.1111 -
val_f1_score: 0.1424
Epoch 5/100
58/58 [==============================] - 327s 6s/step - loss: 1.3068 - accuracy:
0.5268 - f1_score: 0.4710 - val_loss: 2.5900 - val_accuracy: 0.1122 -
val_f1_score: 0.1857
Epoch 6/100
58/58 [==============================] - 327s 6s/step - loss: 1.2142 - accuracy:
0.5633 - f1_score: 0.5131 - val_loss: 2.3346 - val_accuracy: 0.1856 -
```

```
val_f1_score: 0.1551
Epoch 7/100
58/58 [==============================] - 330s 6s/step - loss: 1.1401 - accuracy:
0.5930 - f1_score: 0.5426 - val_loss: 2.0113 - val_accuracy: 0.2678 -
val_f1_score: 0.2310
Epoch 8/100
58/58 [==============================] - 328s 6s/step - loss: 1.0617 - accuracy:
0.6277 - f1_score: 0.5851 - val_loss: 1.6777 - val_accuracy: 0.3722 -
val_f1_score: 0.3346
Epoch 9/100
58/58 [==============================] - 330s 6s/step - loss: 1.0065 - accuracy:
0.6438 - f1_score: 0.6030 - val_loss: 1.2580 - val_accuracy: 0.5889 -
val_f1_score: 0.4676
Epoch 10/100
58/58 [==============================] - 332s 6s/step - loss: 0.9573 - accuracy:
0.6646 - f1_score: 0.6364 - val_loss: 1.1040 - val_accuracy: 0.6356 -
val_f1_score: 0.5319
Epoch 11/100
58/58 [==============================] - 330s 6s/step - loss: 0.9016 - accuracy:
0.6839 - f1_score: 0.6498 - val_loss: 0.9994 - val_accuracy: 0.6900 -
val_f1_score: 0.5717
Epoch 12/100
58/58 [==============================] - 332s 6s/step - loss: 0.8529 - accuracy:
0.6998 - f1_score: 0.6795 - val_loss: 0.9557 - val_accuracy: 0.6944 -
val_f1_score: 0.6521
Epoch 13/100
58/58 [==============================] - 328s 6s/step - loss: 0.8148 - accuracy:
0.7149 - f1_score: 0.6975 - val_loss: 0.8725 - val_accuracy: 0.7233 -
val_f1_score: 0.7400
Epoch 14/100
58/58 [==============================] - 330s 6s/step - loss: 0.7689 - accuracy:
0.7325 - f1_score: 0.7168 - val_loss: 0.8582 - val_accuracy: 0.7189 -
val_f1_score: 0.6705
Epoch 15/100
58/58 [==============================] - 325s 6s/step - loss: 0.7428 - accuracy:
0.7429 - f1_score: 0.7290 - val_loss: 0.9195 - val_accuracy: 0.7133 -
val_f1_score: 0.7054
Epoch 16/100
58/58 [==============================] - 324s 6s/step - loss: 0.6986 - accuracy:
0.7608 - f1_score: 0.7467 - val_loss: 0.8240 - val_accuracy: 0.7333 -
val_f1_score: 0.7383
Epoch 17/100
58/58 [==============================] - 322s 6s/step - loss: 0.6716 - accuracy:
0.7700 - f1_score: 0.7565 - val_loss: 0.8120 - val_accuracy: 0.7289 -
val_f1_score: 0.7273
Epoch 18/100
58/58 [==============================] - 321s 6s/step - loss: 0.6372 - accuracy:
0.7830 - f1_score: 0.7691 - val_loss: 0.7964 - val_accuracy: 0.7400 -
```

```
val_f1_score: 0.7667
Epoch 19/100
58/58 [==============================] - 331s 6s/step - loss: 0.6162 - accuracy:
0.7883 - f1_score: 0.7790 - val_loss: 0.7720 - val_accuracy: 0.7533 -
val_f1_score: 0.7708
Epoch 20/100
58/58 [==============================] - 387s 7s/step - loss: 0.5739 - accuracy:
0.8070 - f1_score: 0.7923 - val_loss: 0.7397 - val_accuracy: 0.7578 -
val_f1_score: 0.7849
Epoch 21/100
58/58 [==============================] - 327s 6s/step - loss: 0.5522 - accuracy:
0.8180 - f1_score: 0.8049 - val_loss: 0.7729 - val_accuracy: 0.7444 -
val_f1_score: 0.6822
Epoch 22/100
58/58 [==============================] - 313s 5s/step - loss: 0.5343 - accuracy:
0.8190 - f1_score: 0.8122 - val_loss: 0.7688 - val_accuracy: 0.7544 -
val_f1_score: 0.7349
Epoch 23/100
58/58 [==============================] - 381s 7s/step - loss: 0.5112 - accuracy:
0.8297 - f1_score: 0.8193 - val_loss: 0.7202 - val_accuracy: 0.7656 -
val_f1_score: 0.7578
Epoch 24/100
58/58 [==============================] - 387s 7s/step - loss: 0.4869 - accuracy:
0.8402 - f1_score: 0.8333 - val_loss: 0.7109 - val_accuracy: 0.7756 -
val_f1_score: 0.7240
Epoch 25/100
58/58 [==============================] - 386s 7s/step - loss: 0.4608 - accuracy:
0.8536 - f1_score: 0.8450 - val_loss: 0.6889 - val_accuracy: 0.7722 -
val_f1_score: 0.7784
Epoch 26/100
58/58 [==============================] - 387s 7s/step - loss: 0.4411 - accuracy:
0.8555 - f1_score: 0.8494 - val_loss: 0.6999 - val_accuracy: 0.7689 -
val_f1_score: 0.7506
Epoch 27/100
58/58 [==============================] - 386s 7s/step - loss: 0.4207 - accuracy:
0.8633 - f1_score: 0.8560 - val_loss: 0.6714 - val_accuracy: 0.7767 -
val_f1_score: 0.8037
Epoch 28/100
58/58 [==============================] - 388s 7s/step - loss: 0.3983 - accuracy:
0.8714 - f1_score: 0.8665 - val_loss: 0.6723 - val_accuracy: 0.7733 -
val_f1_score: 0.7826
Epoch 29/100
58/58 [==============================] - 388s 7s/step - loss: 0.3784 - accuracy:
0.8793 - f1_score: 0.8752 - val_loss: 0.6791 - val_accuracy: 0.7822 -
val_f1_score: 0.7682
Epoch 30/100
58/58 [==============================] - 389s 7s/step - loss: 0.3587 - accuracy:
0.8886 - f1_score: 0.8855 - val_loss: 0.6563 - val_accuracy: 0.7800 -
```

```
val_f1_score: 0.7916
Epoch 31/100
58/58 [==============================] - 391s 7s/step - loss: 0.3438 - accuracy:
0.8920 - f1_score: 0.8871 - val_loss: 0.6534 - val_accuracy: 0.7811 -
val_f1_score: 0.7747
Epoch 32/100
58/58 [==============================] - 387s 7s/step - loss: 0.3305 - accuracy:
0.8974 - f1_score: 0.8939 - val_loss: 0.6383 - val_accuracy: 0.7956 -
val_f1_score: 0.8160
Epoch 33/100
58/58 [==============================] - 382s 7s/step - loss: 0.3059 - accuracy:
0.9098 - f1_score: 0.9041 - val_loss: 0.6653 - val_accuracy: 0.7833 -
val_f1_score: 0.7938
Epoch 34/100
58/58 [==============================] - 383s 7s/step - loss: 0.3030 - accuracy:
0.9083 - f1_score: 0.9020 - val_loss: 0.6461 - val_accuracy: 0.7889 -
val_f1_score: 0.7252
Epoch 35/100
58/58 [==============================] - 386s 7s/step - loss: 0.2904 - accuracy:
0.9133 - f1_score: 0.9069 - val_loss: 0.6532 - val_accuracy: 0.7644 -
val_f1_score: 0.7957
Epoch 36/100
58/58 [==============================] - 388s 7s/step - loss: 0.2700 - accuracy:
0.9240 - f1_score: 0.9187 - val_loss: 0.6415 - val_accuracy: 0.7856 -
val_f1_score: 0.8082
Epoch 37/100
58/58 [==============================] - 384s 7s/step - loss: 0.2564 - accuracy:
0.9253 - f1_score: 0.9227 - val_loss: 0.6037 - val_accuracy: 0.8022 -
val_f1_score: 0.8220
Epoch 38/100
58/58 [==============================] - 386s 7s/step - loss: 0.2439 - accuracy:
0.9305 - f1_score: 0.9277 - val_loss: 0.6113 - val_accuracy: 0.7833 -
val_f1_score: 0.7840
Epoch 39/100
58/58 [==============================] - 388s 7s/step - loss: 0.2397 - accuracy:
0.9315 - f1_score: 0.9269 - val_loss: 0.6106 - val_accuracy: 0.7967 -
val_f1_score: 0.8081
Epoch 40/100
58/58 [==============================] - 387s 7s/step - loss: 0.2213 - accuracy:
0.9422 - f1_score: 0.9365 - val_loss: 0.6496 - val_accuracy: 0.7822 -
val_f1_score: 0.7817
Epoch 41/100
58/58 [==============================] - 389s 7s/step - loss: 0.2116 - accuracy:
0.9454 - f1_score: 0.9418 - val_loss: 0.5996 - val_accuracy: 0.8067 -
val_f1_score: 0.7429
Epoch 42/100
58/58 [==============================] - 387s 7s/step - loss: 0.1955 - accuracy:
0.9484 - f1_score: 0.9451 - val_loss: 0.5891 - val_accuracy: 0.7967 -
```

```
val_f1_score: 0.7976
Epoch 43/100
58/58 [==============================] - 386s 7s/step - loss: 0.1882 - accuracy:
0.9531 - f1_score: 0.9500 - val_loss: 0.6170 - val_accuracy: 0.7911 -
val_f1_score: 0.7879
Epoch 44/100
58/58 [==============================] - 384s 7s/step - loss: 0.1864 - accuracy:
0.9517 - f1_score: 0.9460 - val_loss: 0.5882 - val_accuracy: 0.8022 -
val_f1_score: 0.8103
Epoch 45/100
58/58 [==============================] - 382s 7s/step - loss: 0.1815 - accuracy:
0.9537 - f1_score: 0.9504 - val_loss: 0.5834 - val_accuracy: 0.8100 -
val_f1_score: 0.8091
Epoch 46/100
58/58 [==============================] - 382s 7s/step - loss: 0.1669 - accuracy:
0.9594 - f1_score: 0.9565 - val_loss: 0.5781 - val_accuracy: 0.8100 -
val_f1_score: 0.8083
Epoch 47/100
58/58 [==============================] - 383s 7s/step - loss: 0.1644 - accuracy:
0.9595 - f1_score: 0.9567 - val_loss: 0.5620 - val_accuracy: 0.8089 -
val_f1_score: 0.7844
Epoch 48/100
58/58 [==============================] - 385s 7s/step - loss: 0.1570 - accuracy:
0.9616 - f1_score: 0.9604 - val_loss: 0.6075 - val_accuracy: 0.8033 -
val_f1_score: 0.8069
Epoch 49/100
58/58 [==============================] - 384s 7s/step - loss: 0.1876 - accuracy:
0.9475 - f1_score: 0.9447 - val_loss: 0.6899 - val_accuracy: 0.7578 -
val_f1_score: 0.7082
Epoch 50/100
58/58 [==============================] - 385s 7s/step - loss: 0.1469 - accuracy:
0.9630 - f1_score: 0.9596 - val_loss: 0.6081 - val_accuracy: 0.8000 -
val_f1_score: 0.8032
Epoch 51/100
58/58 [==============================] - 383s 7s/step - loss: 0.1435 - accuracy:
0.9642 - f1_score: 0.9636 - val_loss: 0.5724 - val_accuracy: 0.8056 -
val_f1_score: 0.8003
Epoch 52/100
58/58 [==============================] - 383s 7s/step - loss: 0.1320 - accuracy:
0.9699 - f1_score: 0.9686 - val_loss: 0.5888 - val_accuracy: 0.7944 -
val_f1_score: 0.7930
Epoch 53/100
58/58 [==============================] - 383s 7s/step - loss: 0.1196 - accuracy:
0.9766 - f1_score: 0.9755 - val_loss: 0.5692 - val_accuracy: 0.8133 -
val_f1_score: 0.8220
Epoch 54/100
58/58 [==============================] - 382s 7s/step - loss: 0.1153 - accuracy:
0.9754 - f1_score: 0.9738 - val_loss: 0.5742 - val_accuracy: 0.8089 -
```

```
val_f1_score: 0.8011
Epoch 55/100
58/58 [==============================] - 384s 7s/step - loss: 0.1186 - accuracy:
0.9749 - f1_score: 0.9722 - val_loss: 0.6050 - val_accuracy: 0.8100 -
val_f1_score: 0.8353
Epoch 56/100
58/58 [==============================] - 384s 7s/step - loss: 0.1144 - accuracy:
0.9750 - f1_score: 0.9735 - val_loss: 0.5600 - val_accuracy: 0.8111 -
val_f1_score: 0.7860
Epoch 57/100
58/58 [==============================] - 383s 7s/step - loss: 0.1038 - accuracy:
0.9801 - f1_score: 0.9778 - val_loss: 0.5567 - val_accuracy: 0.8056 -
val_f1_score: 0.7877
Epoch 58/100
58/58 [==============================] - 385s 7s/step - loss: 0.0986 - accuracy:
0.9814 - f1_score: 0.9796 - val_loss: 0.5822 - val_accuracy: 0.8100 -
val_f1_score: 0.8067
Epoch 59/100
58/58 [==============================] - 385s 7s/step - loss: 0.0934 - accuracy:
0.9828 - f1_score: 0.9811 - val_loss: 0.5759 - val_accuracy: 0.8122 -
val_f1_score: 0.8371
Epoch 60/100
58/58 [==============================] - 382s 7s/step - loss: 0.0902 - accuracy:
0.9842 - f1_score: 0.9825 - val_loss: 0.5882 - val_accuracy: 0.8144 -
val_f1_score: 0.7759
Epoch 61/100
58/58 [==============================] - 383s 7s/step - loss: 0.0836 - accuracy:
0.9856 - f1_score: 0.9848 - val_loss: 0.5547 - val_accuracy: 0.8178 -
val_f1_score: 0.8120
Epoch 62/100
58/58 [==============================] - 383s 7s/step - loss: 0.0788 - accuracy:
0.9877 - f1_score: 0.9866 - val_loss: 0.5482 - val_accuracy: 0.8044 -
val_f1_score: 0.8040
Epoch 63/100
58/58 [==============================] - 384s 7s/step - loss: 0.0779 - accuracy:
0.9866 - f1_score: 0.9856 - val_loss: 0.5660 - val_accuracy: 0.8111 -
val_f1_score: 0.8414
Epoch 64/100
58/58 [==============================] - 383s 7s/step - loss: 0.0767 - accuracy:
0.9873 - f1_score: 0.9868 - val_loss: 0.5620 - val_accuracy: 0.8122 -
val_f1_score: 0.8382
Epoch 65/100
58/58 [==============================] - 384s 7s/step - loss: 0.0753 - accuracy:
0.9874 - f1_score: 0.9862 - val_loss: 0.5586 - val_accuracy: 0.8122 -
val_f1_score: 0.7793
Epoch 66/100
58/58 [==============================] - 383s 7s/step - loss: 0.0767 - accuracy:
0.9864 - f1_score: 0.9852 - val_loss: 0.5793 - val_accuracy: 0.8111 -
```

```
val_f1_score: 0.7754
Epoch 67/100
58/58 [==============================] - 383s 7s/step - loss: 0.0724 - accuracy:
0.9874 - f1_score: 0.9859 - val_loss: 0.5363 - val_accuracy: 0.8222 -
val_f1_score: 0.8149
Epoch 68/100
58/58 [==============================] - 386s 7s/step - loss: 0.0661 - accuracy:
0.9892 - f1_score: 0.9885 - val_loss: 0.5517 - val_accuracy: 0.8300 -
val_f1_score: 0.8493
Epoch 69/100
58/58 [==============================] - 384s 7s/step - loss: 0.0652 - accuracy:
0.9901 - f1_score: 0.9892 - val_loss: 0.5605 - val_accuracy: 0.8244 -
val_f1_score: 0.8493
Epoch 70/100
58/58 [==============================] - 383s 7s/step - loss: 0.0635 - accuracy:
0.9903 - f1_score: 0.9893 - val_loss: 0.5599 - val_accuracy: 0.8056 -
val_f1_score: 0.8068
Epoch 71/100
58/58 [==============================] - 384s 7s/step - loss: 0.0615 - accuracy:
0.9905 - f1_score: 0.9896 - val_loss: 0.5870 - val_accuracy: 0.8144 -
val_f1_score: 0.8422
Epoch 72/100
58/58 [==============================] - 382s 7s/step - loss: 0.0592 - accuracy:
0.9908 - f1_score: 0.9902 - val_loss: 0.5451 - val_accuracy: 0.8322 -
val_f1_score: 0.8503
Epoch 73/100
58/58 [==============================] - 386s 7s/step - loss: 0.0554 - accuracy:
0.9920 - f1_score: 0.9915 - val_loss: 0.5510 - val_accuracy: 0.8211 -
val_f1_score: 0.8455
Epoch 74/100
58/58 [==============================] - 384s 7s/step - loss: 0.0528 - accuracy:
0.9923 - f1_score: 0.9920 - val_loss: 0.5473 - val_accuracy: 0.8322 -
val_f1_score: 0.8386
Epoch 75/100
58/58 [==============================] - 385s 7s/step - loss: 0.0505 - accuracy:
0.9927 - f1_score: 0.9929 - val_loss: 0.5461 - val_accuracy: 0.8356 -
val_f1_score: 0.8318
Epoch 76/100
58/58 [==============================] - 384s 7s/step - loss: 0.0491 - accuracy:
0.9937 - f1_score: 0.9935 - val_loss: 0.5564 - val_accuracy: 0.8222 -
val_f1_score: 0.8069
Epoch 77/100
58/58 [==============================] - 384s 7s/step - loss: 0.0479 - accuracy:
0.9931 - f1_score: 0.9926 - val_loss: 0.5331 - val_accuracy: 0.8200 -
val_f1_score: 0.8470
Epoch 78/100
58/58 [==============================] - 384s 7s/step - loss: 0.0503 - accuracy:
0.9920 - f1_score: 0.9911 - val_loss: 0.5960 - val_accuracy: 0.8178 -
```

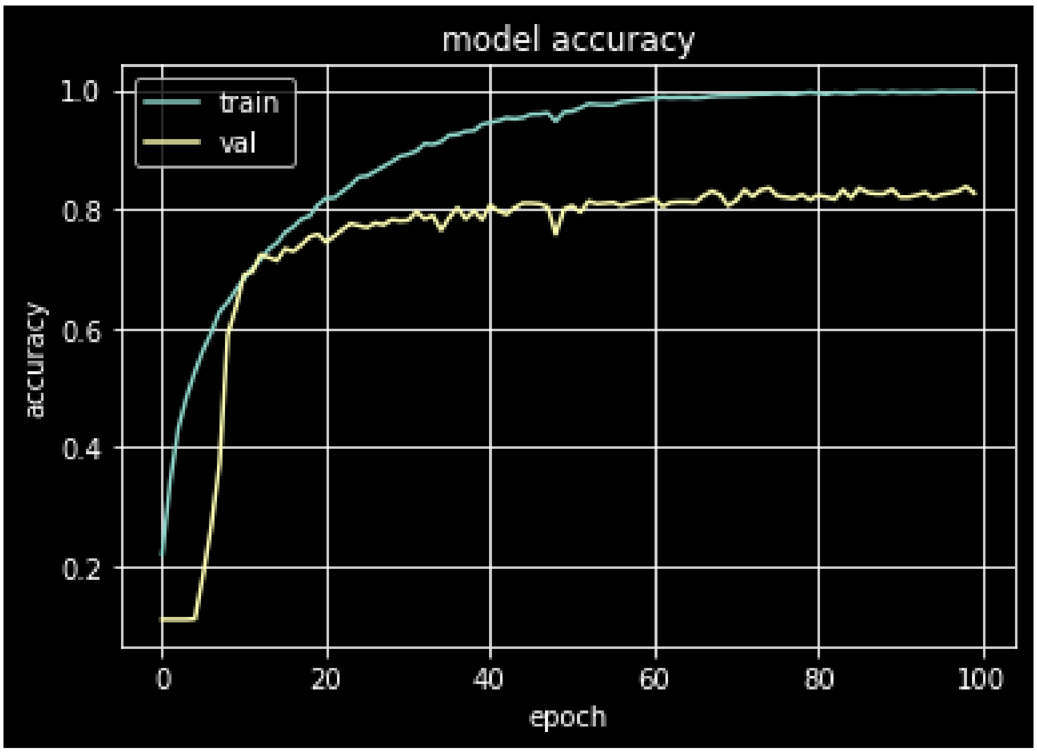
```
val_f1_score: 0.8407
Epoch 79/100
58/58 [==============================] - 384s 7s/step - loss: 0.0438 - accuracy:
0.9943 - f1_score: 0.9943 - val_loss: 0.5440 - val_accuracy: 0.8244 -
val_f1_score: 0.8313
Epoch 80/100
58/58 [==============================] - 384s 7s/step - loss: 0.0424 - accuracy:
0.9958 - f1_score: 0.9950 - val_loss: 0.5433 - val_accuracy: 0.8156 -
val_f1_score: 0.8266
Epoch 81/100
58/58 [==============================] - 384s 7s/step - loss: 0.0432 - accuracy:
0.9942 - f1_score: 0.9933 - val_loss: 0.5736 - val_accuracy: 0.8233 -
val_f1_score: 0.8500
Epoch 82/100
58/58 [==============================] - 386s 7s/step - loss: 0.0489 - accuracy:
0.9923 - f1_score: 0.9917 - val_loss: 0.5771 - val_accuracy: 0.8200 -
val_f1_score: 0.8104
Epoch 83/100
58/58 [==============================] - 385s 7s/step - loss: 0.0410 - accuracy:
0.9954 - f1_score: 0.9951 - val_loss: 0.5685 - val_accuracy: 0.8167 -
val_f1_score: 0.8427
Epoch 84/100
58/58 [==============================] - 386s 7s/step - loss: 0.0418 - accuracy:
0.9946 - f1_score: 0.9939 - val_loss: 0.5192 - val_accuracy: 0.8322 -
val_f1_score: 0.8021
Epoch 85/100
58/58 [==============================] - 394s 7s/step - loss: 0.0391 - accuracy:
0.9937 - f1_score: 0.9933 - val_loss: 0.5591 - val_accuracy: 0.8189 -
val_f1_score: 0.8163
Epoch 86/100
58/58 [==============================] - 384s 7s/step - loss: 0.0371 - accuracy:
0.9961 - f1_score: 0.9958 - val_loss: 0.5472 - val_accuracy: 0.8356 -
val_f1_score: 0.8586
Epoch 87/100
58/58 [==============================] - 385s 7s/step - loss: 0.0345 - accuracy:
0.9962 - f1_score: 0.9962 - val_loss: 0.5395 - val_accuracy: 0.8278 -
val_f1_score: 0.7652
Epoch 88/100
58/58 [==============================] - 384s 7s/step - loss: 0.0336 - accuracy:
0.9962 - f1_score: 0.9961 - val_loss: 0.5359 - val_accuracy: 0.8256 -
val_f1_score: 0.8212
Epoch 89/100
58/58 [==============================] - 383s 7s/step - loss: 0.0350 - accuracy:
0.9949 - f1_score: 0.9951 - val_loss: 0.5396 - val_accuracy: 0.8256 -
val_f1_score: 0.7923
Epoch 90/100
58/58 [==============================] - 385s 7s/step - loss: 0.0316 - accuracy:
0.9966 - f1_score: 0.9961 - val_loss: 0.5570 - val_accuracy: 0.8333 -
```

```
val_f1_score: 0.8280
Epoch 91/100
58/58 [==============================] - 385s 7s/step - loss: 0.0327 - accuracy:
0.9954 - f1_score: 0.9951 - val_loss: 0.5780 - val_accuracy: 0.8200 -
val_f1_score: 0.8482
Epoch 92/100
58/58 [==============================] - 385s 7s/step - loss: 0.0320 - accuracy:
0.9955 - f1_score: 0.9957 - val_loss: 0.5417 - val_accuracy: 0.8200 -
val_f1_score: 0.8439
Epoch 93/100
58/58 [==============================] - 371s 6s/step - loss: 0.0297 - accuracy:
0.9959 - f1_score: 0.9955 - val_loss: 0.5561 - val_accuracy: 0.8233 -
val_f1_score: 0.7838
Epoch 94/100
58/58 [==============================] - 317s 5s/step - loss: 0.0349 - accuracy:
0.9953 - f1_score: 0.9952 - val_loss: 0.5281 - val_accuracy: 0.8278 -
val_f1_score: 0.8515
Epoch 95/100
58/58 [==============================] - 329s 6s/step - loss: 0.0286 - accuracy:
0.9958 - f1_score: 0.9957 - val_loss: 0.5718 - val_accuracy: 0.8189 -
val_f1_score: 0.8444
Epoch 96/100
58/58 [==============================] - 389s 7s/step - loss: 0.0275 - accuracy:
0.9970 - f1_score: 0.9971 - val_loss: 0.5710 - val_accuracy: 0.8244 -
val_f1_score: 0.8488
Epoch 97/100
58/58 [==============================] - 390s 7s/step - loss: 0.0283 - accuracy:
0.9964 - f1_score: 0.9959 - val_loss: 0.5621 - val_accuracy: 0.8267 -
val_f1_score: 0.8302
Epoch 98/100
58/58 [==============================] - 374s 6s/step - loss: 0.0279 - accuracy:
0.9966 - f1_score: 0.9964 - val_loss: 0.5251 - val_accuracy: 0.8300 -
val_f1_score: 0.7951
Epoch 99/100
58/58 [==============================] - 322s 6s/step - loss: 0.0271 - accuracy:
0.9966 - f1_score: 0.9964 - val_loss: 0.5297 - val_accuracy: 0.8378 -
val_f1_score: 0.8635
Epoch 100/100
58/58 [==============================] - 322s 6s/step - loss: 0.0290 - accuracy:
0.9966 - f1_score: 0.9962 - val_loss: 0.5392 - val_accuracy: 0.8267 -
val_f1_score: 0.8343
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-13-83d7ad36eced> in <module>
      1 history = model.fit(train_generator, epochs=100,␣
 ↪validation_data=validation_generator)
```

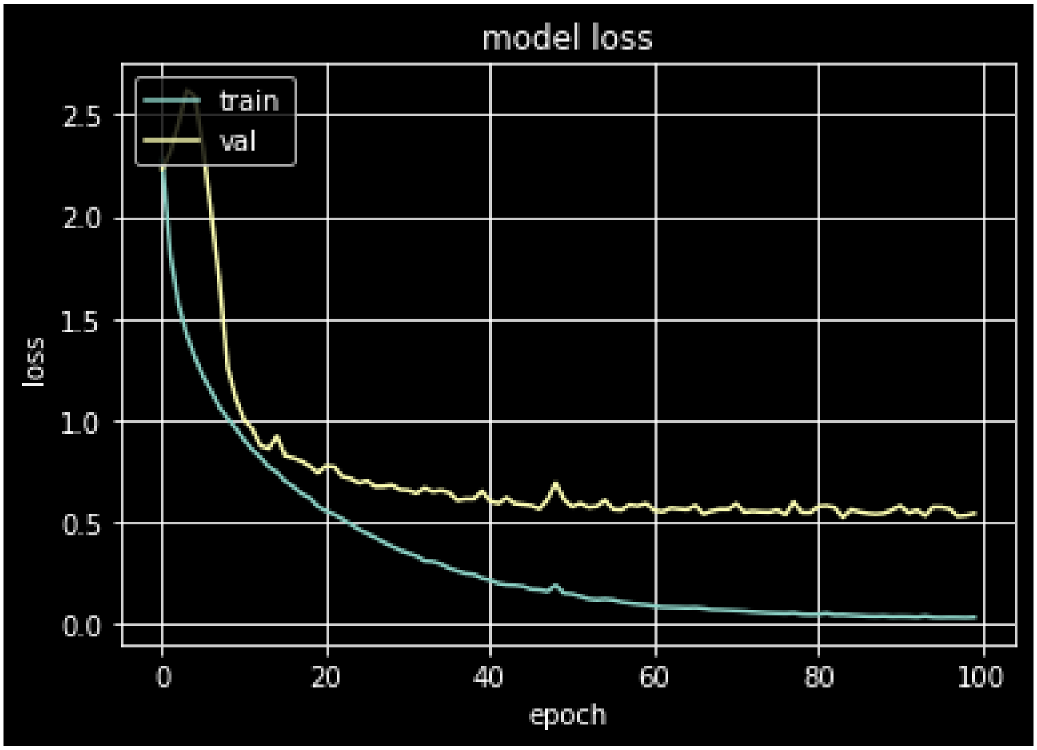
```
----> 2 plt.plot(history.history['acc'])
      3 plt.plot(history.history['val_acc'])
      4 plt.title('model accuracy')
      5 plt.ylabel('accuracy')

KeyError: 'acc'
```

[17]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.grid()
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



[18]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.grid()
plt.legend(['train', 'val'], loc='upper left')
```

```
plt.show()
```



[20]:
```
model.save('CNNModelWeights.h5', save_format='h5')
```

[ ]: