Documentation for prototype

Summary Description

This code implements a deep learning pipeline for classifying spectrogram images of radio bursts into three types. It starts by reading .fit.gz files, extracting and preprocessing spectrogram data to remove background noise using median subtraction. The preprocessed spectrograms are resized to (128, 128) and normalized to a range of [0, 1]. Labels are encoded using LabelEncoder. A Convolutional Neural Network (CNN) is defined with three convolutional layers, dropout for regularization, and a softmax output for multi-class classification. The model is trained on the preprocessed data, with a validation split of 20% and checkpointing to save the best model. Training and validation accuracy/loss curves are visualized, and the final model is evaluated on a test set. The approach efficiently handles parallel data loading and preprocessing while demonstrating strong performance in classifying spectrogram data.

Code Cell 1

- `import pandas as pd`: This line imports a library or module.

- `from tensorflow import keras`: Imports specific parts (functions, classes) of a library.

- `from tensorflow.keras import layers`: Imports specific parts (functions, classes) of a library.

- `import tensorflow_hub as hub`: This line imports a library or module.

- `import numpy as np`: This line imports a library or module.

- `import matplotlib.pyplot as plt`: This line imports a library or module.

- `import sys`: This line imports a library or module.

- `from tqdm import tqdm`: Imports specific parts (functions, classes) of a library.

- `import os`: This line imports a library or module.

- `import cv2`: This line imports a library or module.

- `from concurrent.futures import ThreadPoolExecutor`: Imports specific parts (functions, classes) of a library.

- `sys.path.append(r"D:\My Laptop\Me\Programming\Machine Learning\Internships\Egyptian Space Agency\2012")`: Adds a custom path to the system path for importing modules.

- ``: Performs a specific operation in the code.

- `# Now you can import your custom library`: Performs a specific operation in the code.

- `from read_Data import read_fits`: Imports specific parts (functions, classes) of a library.

Code Cell 2

- `os.listdir('radio bursts data')`: Lists the contents of a directory.

Code Cell 3

- `print(len(os.listdir(os.path.join("radio bursts data",'noise and empty')))/2)`: Lists the contents of a directory.

- `print(len(os.listdir(os.path.join("radio bursts data",'rbtype2')))/2)`: Lists the contents of a directory.

- `print(len(os.listdir(os.path.join("radio bursts data",'rbtype3')))/2)`: Lists the contents of a directory.

Code Cell 4

- `# def read_image(full_img_path):`: Performs a specific operation in the code.

- `#     try:`: Performs a specific operation in the code.

- `#         img = cv2.imread(full_img_path)`: Uses OpenCV library for computer vision tasks.

- `#         return img`: Performs a specific operation in the code.

- `#     except Exception as e:`: Performs a specific operation in the code.

- `#         print(f"Warning: Error reading image '{full_img_path}': {e}")`: Prints values or output to the console.

- `#         return None`: Performs a specific operation in the code.

Code Cell 5

- ``: Performs a specific operation in the code.

- `# def read_data(path, verbose=False,shape=(128,128)):`: Performs a specific operation in the code.

- `#     images = []`: Performs a specific operation in the code.

- `#     labels = []`: Performs a specific operation in the code.

- `#     names =[]`: Performs a specific operation in the code.

- `#     for img_type in tqdm(os.listdir(path)):`: Lists the contents of a directory.

- `#         type_path = os.path.join(path, img_type)`: Performs a specific operation in the code.

- `#         img_paths = [os.path.join(type_path, img) for img in os.listdir(type_path) if img.endswith('png')]`: Lists the contents of a directory.

- ``: Performs a specific operation in the code.

- `#         # Use ThreadPoolExecutor for parallel image reading`: Performs a specific operation in the code.

- `#         with ThreadPoolExecutor() as executor:`: Performs a specific operation in the code.

- `#             results = list(executor.map(read_image, img_paths))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#         # Filter out None results and extend images and labels lists`: Performs a specific operation in the code.

- `#         for img, img_path in zip(results, img_paths):`: Performs a specific operation in the code.

- `#             if img is not None:`: Performs a specific operation in the code.

- `#                 img = cv2.resize(img,shape)`: Uses OpenCV library for computer vision tasks.

- `#                 images.append(img)`: Performs a specific operation in the code.

- `#                 labels.append(img_type)`: Performs a specific operation in the code.

- `#                 names.append(img_path)`: Performs a specific operation in the code.

- `#                 if verbose:`: Performs a specific operation in the code.

- `#                     print(f"Read file: {img_path}")`: Prints values or output to the console.

- ``: Performs a specific operation in the code.

- `#         print(f'Finished reading {len(img_paths)} images of type "{img_type}".')`: Prints values or output to the console.

- `#     return np.array(images),np.array(names),np.array(labels)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# # Adjust the verbosity as needed`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 6

- `# images,pathes, labels = read_data('radio bursts data', verbose=False)`: Performs a specific operation in the code.

Code Cell 7

- `# images.shape`: Performs a specific operation in the code.

Code Cell 8

- `# len(pathes)`: Performs a specific operation in the code.

Code Cell 9

- `# labels.shape`: Performs a specific operation in the code.

Code Cell 10

- `# pathes_reshaped = pathes.reshape(-1, 1)`: Performs a specific operation in the code.

- `# labels_reshaped = labels.reshape(-1, 1)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# # Concatenate along axis 1`: Performs a specific operation in the code.

- `# combined = np.concatenate([pathes_reshaped, labels_reshaped], axis=1)`: Performs a specific operation in the code.

- `# print(combined.shape)`: Prints values or output to the console.

Code Cell 11

- `def read_fits_data(path, verbose=False,shape=(128,128)):`: Performs a specific operation in the code.

- `images = []`: Performs a specific operation in the code.

- `labels = []`: Performs a specific operation in the code.

- `freqs = []`: Performs a specific operation in the code.

- `times = []`: Performs a specific operation in the code.

- `names=[]`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `for img_type in tqdm(os.listdir(path)):`: Lists the contents of a directory.

- `type_path = os.path.join(path, img_type)`: Performs a specific operation in the code.

- `img_paths = [os.path.join(type_path, img) for img in os.listdir(type_path) if img.endswith('fit.gz')]`: Lists the contents of a directory.

- ``: Performs a specific operation in the code.

- `# Use ThreadPoolExecutor for parallel file reading`: Performs a specific operation in the code.

- `with ThreadPoolExecutor() as executor:`: Performs a specific operation in the code.

- `results = list(executor.map(read_fits, img_paths))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Filter out None results and extend the lists`: Performs a specific operation in the code.

- `for (img, freq, time), img_path in zip(results, img_paths):`: Performs a specific operation in the code.

- `if img is not None:`: Performs a specific operation in the code.

- `img=cv2.resize(img,shape)`: Uses OpenCV library for computer vision tasks.

- `names.append(img_path)`: Performs a specific operation in the code.

- `images.append(np.array(img))`: Performs a specific operation in the code.

- `freqs.append(freq)`: Performs a specific operation in the code.

- `times.append(time)`: Performs a specific operation in the code.

- `labels.append(img_type)`: Performs a specific operation in the code.

- `if verbose:`: Performs a specific operation in the code.

- `print(f"Read file: {img_path}")`: Prints values or output to the console.

- ``: Performs a specific operation in the code.

- `print(f'Finished reading {len(img_paths)} files of type "{img_type}".')`: Prints values or output to the console.

- ``: Performs a specific operation in the code.

- `return np.array(images), np.array(labels), freqs, times,np.array(names)`: Performs a specific operation in the code.

Code Cell 12

- `images_fits, labels_fits, freqs_fits, times_fits,pathes_fits = read_fits_data('radio bursts data', verbose=False)`: Performs a specific operation in the code.

Code Cell 13

- `images_fits.shape`: Performs a specific operation in the code.

Code Cell 14

- `from collections import Counter`: Imports specific parts (functions, classes) of a library.

- `Counter(labels_fits)`: Performs a specific operation in the code.

Code Cell 15

- `def preprocess_spectrogram(data):`: Performs a specific operation in the code.

- `"""`: Performs a specific operation in the code.

- `Calculate the median over time for each frequency channel`: Performs a specific operation in the code.

- `and subtract median values from each time column to remove background noise.`: Performs a specific operation in the code.

- `"""`: Performs a specific operation in the code.

- `median_values = np.median(data, axis=0)`: Performs a specific operation in the code.

- `processed_data = data - median_values`: Performs a specific operation in the code.

- `return processed_data`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 16

- `preprocessed_spectrogram_data = [preprocess_spectrogram(img) for img in images_fits]`: Performs a specific operation in the code.

- `preprocessed_spectrogram_data = np.array(preprocessed_spectrogram_data)`: Performs a specific operation in the code.

Code Cell 17

- `print(f"Shape of preprocessed spectrogram data: {preprocessed_spectrogram_data.shape}")`: Prints values or output to the console.

Code Cell 18

- `plt.imshow(images_fits[1], aspect='auto', origin='lower')`: Performs a specific operation in the code.

- `plt.title('Noise')`: Performs a specific operation in the code.

- `plt.xlabel('Time')`: Performs a specific operation in the code.

- `plt.ylabel('Frequency')`: Performs a specific operation in the code.

Code Cell 19

- `plt.imshow(images_fits[-1], aspect='auto', origin='lower')`: Performs a specific operation in the code.

- `plt.title('Type 3')`: Performs a specific operation in the code.

- `plt.xlabel('Time')`: Performs a specific operation in the code.

- `plt.ylabel('Frequency')`: Performs a specific operation in the code.

Code Cell 20

- `plt.imshow(images_fits[600], aspect='auto', origin='lower')`: Performs a specific operation in the code.

- `plt.title('Type 2')`: Performs a specific operation in the code.

- `plt.xlabel('Time')`: Performs a specific operation in the code.

- `plt.ylabel('Frequency')`: Performs a specific operation in the code.

Code Cell 21

- `difference = images_fits - preprocessed_spectrogram_data`: Performs a specific operation in the code.

- `plt.imshow(difference[600], aspect='auto', origin='lower', cmap='viridis')`: Performs a specific operation in the code.

- `plt.title('Difference Between Original and Preprocessed')`: Performs a specific operation in the code.

- `plt.xlabel('Time')`: Performs a specific operation in the code.

- `plt.ylabel('Frequency')`: Performs a specific operation in the code.

- `plt.colorbar()`: Performs a specific operation in the code.

- `plt.show()`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 22

- `difference.shape`: Performs a specific operation in the code.

Code Cell 23

- `preprocessed_spectrogram_data.shape`: Performs a specific operation in the code.

Code Cell 24

- ``: Performs a specific operation in the code.

- `import tensorflow as tf`: This line imports a library or module.

- `from tensorflow.keras.models import Sequential,Model,load_model`: Imports specific parts (functions, classes) of a library.

- `from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout , Add,Input`: Imports specific parts (functions, classes) of a library.

- `from tensorflow.keras.optimizers import Adam`: Imports specific parts (functions, classes) of a library.

- `from sklearn.model_selection import train_test_split`: Imports specific parts (functions, classes) of a library.

- `from sklearn.preprocessing import LabelEncoder`: Imports specific parts (functions, classes) of a library.

- `from tensorflow.keras.callbacks import ModelCheckpoint`: Imports specific parts (functions, classes) of a library.

- `from tensorflow.keras.regularizers import l2`: Imports specific parts (functions, classes) of a library.

Code Cell 25

- `preprocessed_spectrogram_data = np.expand_dims(preprocessed_spectrogram_data, axis=-1)  # Shape will be (num_samples, 128, 128, 1)`: Performs a specific operation in the code.

Code Cell 26

- `difference = np.expand_dims(difference, axis=-1)  # Shape will be (num_samples, 128, 128, 1)`: Performs a specific operation in the code.

- `difference.shape`: Performs a specific operation in the code.

Code Cell 27

- `images_fits = np.expand_dims(images_fits, axis=-1)`: Performs a specific operation in the code.

- `images_fits.shape`: Performs a specific operation in the code.

Code Cell 28

- `X_train, X_test, y_train, y_test = train_test_split(`: Performs a specific operation in the code.

- `preprocessed_spectrogram_data, labels_fits, test_size=0.2, random_state=42,shuffle=True`: Performs a specific operation in the code.

- `)`: Performs a specific operation in the code.

Code Cell 29

- `X_train.shape`: Performs a specific operation in the code.

Code Cell 30

- `X_test.shape`: Performs a specific operation in the code.

Code Cell 31

- `label_encoder = LabelEncoder()`: Performs a specific operation in the code.

- `y_train_encoded = label_encoder.fit_transform(y_train)`: Performs a specific operation in the code.

Code Cell 32

- `y_test_encoded = label_encoder.transform(y_test)`: Performs a specific operation in the code.

Code Cell 33

- `y_train_encoded`: Performs a specific operation in the code.

Code Cell 34

- `X_train = X_train / 255.0`: Performs a specific operation in the code.

- `X_test = X_test / 255.0`: Performs a specific operation in the code.

Code Cell 35

- `def create_cnn_model(input_shape, num_classes):`: Performs a specific operation in the code.

- `model = Sequential()`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# First Convolutional Layer`: Performs a specific operation in the code.

- `model.add(Conv2D(32, (3, 3), activation='relu', kernel_regularizer=l2(0.001), input_shape=input_shape))`: Performs a specific operation in the code.

- `model.add(Dropout(0.2))`: Performs a specific operation in the code.

- `model.add(MaxPooling2D((2, 2)))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Second Convolutional Layer`: Performs a specific operation in the code.

- `model.add(Conv2D(64, (3, 3), activation='relu'))`: Performs a specific operation in the code.

- `model.add(Dropout(0.3))`: Performs a specific operation in the code.

- `model.add(MaxPooling2D((2, 2)))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Third Convolutional Layer`: Performs a specific operation in the code.

- `model.add(Conv2D(128, (3, 3), activation='relu'))`: Performs a specific operation in the code.

- `model.add(Dropout(0.5))`: Performs a specific operation in the code.

- `model.add(MaxPooling2D((2, 2)))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Flattening the output from the convolutional layers`: Performs a specific operation in the code.

- `model.add(Flatten())`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Fully Connected Layer`: Performs a specific operation in the code.

- `model.add(Dense(128, activation='relu'))`: Performs a specific operation in the code.

- `model.add(Dropout(0.5))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Output Layer`: Performs a specific operation in the code.

- `model.add(Dense(3, activation='softmax'))`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `return model`: Performs a specific operation in the code.

Code Cell 36

- `# def residual_block(x, filters, kernel_size=(3, 3)):`: Performs a specific operation in the code.

- `#     shortcut = x`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     x = Conv2D(filters, kernel_size, padding='same', activation='relu')(x)`: Performs a specific operation in the code.

- `#     x = Dropout(0.1)(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     x = Conv2D(filters, kernel_size, padding='same', activation='relu')(x)`: Performs a specific operation in the code.

- `#     x = Dropout(0.1)(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Adjust the shortcut to match the output shape`: Performs a specific operation in the code.

- `#     if shortcut.shape[-1] != filters:`: Performs a specific operation in the code.

- `#         shortcut = Conv2D(filters, (1, 1), padding='same')(shortcut)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     x = Add()([x, shortcut])`: Performs a specific operation in the code.

- `#     return x`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# def create_residual_cnn_model(input_shape, num_classes):`: Performs a specific operation in the code.

- `#     inputs = Input(shape=input_shape)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Initial Conv Layer`: Performs a specific operation in the code.

- `#     x = Conv2D(32, (3, 3), padding='same', activation='relu')(inputs)`: Performs a specific operation in the code.

- `#     x = MaxPooling2D((2, 2))(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Residual Block 1`: Performs a specific operation in the code.

- `#     x = residual_block(x, 32)`: Performs a specific operation in the code.

- `#     x = MaxPooling2D((2, 2))(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Residual Block 2`: Performs a specific operation in the code.

- `#     x = residual_block(x, 64)  # Note the increase in filters`: Performs a specific operation in the code.

- `#     x = MaxPooling2D((2, 2))(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Residual Block 3`: Performs a specific operation in the code.

- `#     x = residual_block(x, 128)`: Performs a specific operation in the code.

- `#     x = MaxPooling2D((2, 2))(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Flattening the output from the convolutional layers`: Performs a specific operation in the code.

- `#     x = Flatten()(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Fully Connected Layer`: Performs a specific operation in the code.

- `#     x = Dense(128, activation='relu')(x)`: Performs a specific operation in the code.

- `#     x = Dropout(0.5)(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Output Layer`: Performs a specific operation in the code.

- `#     outputs = Dense(num_classes, activation='softmax')(x)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     # Create the model`: Performs a specific operation in the code.

- `#     model = Model(inputs=inputs, outputs=outputs)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `#     return model`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 37

- `input_shape = X_train.shape[1:]  # e.g., (128, 128, 1)`: Performs a specific operation in the code.

- `num_classes = len(label_encoder.classes_)`: Performs a specific operation in the code.

Code Cell 38

- `model = create_cnn_model(input_shape, num_classes)`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

- `# Compile the model`: Performs a specific operation in the code.

- `model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])`: Performs a specific operation in the code.

Code Cell 39

- `model.summary()`: Performs a specific operation in the code.

Code Cell 40

- `print(f'y train encoded dim ={y_train_encoded.shape}')`: Prints values or output to the console.

- `print(f'x train dim ={X_train.shape}')`: Prints values or output to the console.

- ``: Performs a specific operation in the code.

Code Cell 41

- `num_classes = len(label_encoder.classes_)`: Performs a specific operation in the code.

- `print(f"Number of classes: {num_classes}")`: Prints values or output to the console.

- ``: Performs a specific operation in the code.

Code Cell 42

- `y_train_encoded`: Performs a specific operation in the code.

Code Cell 43

- `y_train`: Performs a specific operation in the code.

Code Cell 44

- `checkpoint = ModelCheckpoint('best_model.keras', monitor='val_accuracy', save_best_only=True)`: Defines or uses machine learning libraries like Keras or TensorFlow.

- `history = model.fit(`: Performs a specific operation in the code.

- `X_train, y_train_encoded,`: Performs a specific operation in the code.

- `epochs=20,`: Performs a specific operation in the code.

- `batch_size=32,`: Performs a specific operation in the code.

- `validation_split=0.2,`: Performs a specific operation in the code.

- `callbacks=[checkpoint]`: Performs a specific operation in the code.

- `)`: Performs a specific operation in the code.

Code Cell 45

- `plt.figure(figsize=(10, 5))`: Performs a specific operation in the code.

- `plt.plot(history.history['accuracy'], label='Train Accuracy')`: Performs a specific operation in the code.

- `plt.plot(history.history['val_accuracy'], label='Validation Accuracy')`: Performs a specific operation in the code.

- `plt.title('Model Accuracy')`: Performs a specific operation in the code.

- `plt.xlabel('Epoch')`: Performs a specific operation in the code.

- `plt.ylabel('Accuracy')`: Performs a specific operation in the code.

- `plt.legend(loc='upper left')`: Performs a specific operation in the code.

- `plt.show()`: Performs a specific operation in the code.

Code Cell 46

- `# Plot training & validation loss values`: Performs a specific operation in the code.

- `plt.figure(figsize=(10, 5))`: Performs a specific operation in the code.

- `plt.plot(history.history['loss'], label='Train Loss')`: Performs a specific operation in the code.

- `plt.plot(history.history['val_loss'], label='Validation Loss')`: Performs a specific operation in the code.

- `plt.title('Model Loss')`: Performs a specific operation in the code.

- `plt.xlabel('Epoch')`: Performs a specific operation in the code.

- `plt.ylabel('Loss')`: Performs a specific operation in the code.

- `plt.legend(loc='upper left')`: Performs a specific operation in the code.

- `plt.show()`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 47

- `y_test_encoded.shape`: Performs a specific operation in the code.

Code Cell 48

- `test_loss, test_accuracy = model.evaluate(X_test, y_test_encoded)  # Use y_test_encoded`: Performs a specific operation in the code.

- `train_loss, train_accuracy = model.evaluate(X_train, y_train_encoded)  # Use y_test_encoded`: Performs a specific operation in the code.

- `print(f'Train Accuracy: {train_accuracy * 100:.2f}%')`: Prints values or output to the console.

- `print(f'Test Accuracy: {test_accuracy * 100:.2f}%')`: Prints values or output to the console.

Code Cell 49

- `from sklearn.metrics import confusion_matrix`: Imports specific parts (functions, classes) of a library.

- `from seaborn import heatmap`: Imports specific parts (functions, classes) of a library.

- ``: Performs a specific operation in the code.

- `heatmap(confusion_matrix(np.argmax(model.predict(X_test),axis=1),y_test_encoded),annot=True)`: Performs a specific operation in the code.

Code Cell 50

- `model = load_model('best_model.keras')`: Defines or uses machine learning libraries like Keras or TensorFlow.

Code Cell 51

- `test_loss, test_accuracy = model.evaluate(X_test, y_test_encoded)  # Use y_test_encoded`: Performs a specific operation in the code.

- `train_loss, train_accuracy = model.evaluate(X_train, y_train_encoded)  # Use y_test_encoded`: Performs a specific operation in the code.

- `print(f'Train Accuracy: {train_accuracy * 100:.2f}%')`: Prints values or output to the console.

- `print(f'Test Accuracy: {test_accuracy * 100:.2f}%')`: Prints values or output to the console.

Code Cell 52

- `# train_accuracies =[]`: Performs a specific operation in the code.

- `# train_losses =[]`: Performs a specific operation in the code.

- `# test_accuracies =[]`: Performs a specific operation in the code.

- `# test_losses =[]`: Performs a specific operation in the code.

Code Cell 53

- `# train_accuracies.append(train_accuracy)`: Performs a specific operation in the code.

- `# train_losses.append(train_loss)`: Performs a specific operation in the code.

- `# test_accuracies.append(test_accuracy)`: Performs a specific operation in the code.

- `# test_losses.append(test_loss)`: Performs a specific operation in the code.

Code Cell 54

- `from sklearn.metrics import confusion_matrix`: Imports specific parts (functions, classes) of a library.

- `from seaborn import heatmap`: Imports specific parts (functions, classes) of a library.

- ``: Performs a specific operation in the code.

- `heatmap(confusion_matrix(np.argmax(model.predict(X_test),axis=1),y_test_encoded),annot=True)`: Performs a specific operation in the code.

Code Cell 55

- `# when i used the difference i got 58% in train and 57% in test`: Performs a specific operation in the code.

- `# when i used the preprocessed spectrogram i got 91.6% in train and 82.14% in test`: Performs a specific operation in the code.

- `# when i used the original data i got 63.97% in train and 62.62% in test`: Performs a specific operation in the code.

- `# when i used the preprocessed spectrogram with batch = 64 i got 95% in train and 82.86% in test`: Performs a specific operation in the code.

- `# when i used the preprocessed spectrogram with split valid = 0.2 i got 95.35% in train and 82.86% in test`: Performs a specific operation in the code.

- ``: Performs a specific operation in the code.

Code Cell 56

- `# model.save_weights('models/processed_spetro_91.weights.h5')`: Performs a specific operation in the code.