```
!pip install torchtext==0.17.0
```

```
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
Requirement already satisfied: torchtext==0.17.0 in /usr/local/lib/python3.11/dist-packages (0.17.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from torchtext==0.17.0) (4.67.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from torchtext==0.17.0) (2.3
Collecting torch==2.2.0 (from torchtext==0.17.0)
  Using cached torch-2.2.0-cp311-cp311-manylinux1_x86_64.whl.metadata (25 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from torchtext==0.17.0) (2.0.2)
Requirement already satisfied: torchdata==0.7.1 in /usr/local/lib/python3.11/dist-packages (from torchtext==0.17
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torchtext
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dist-packages (from torch==
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torchtext==0
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torchtext
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torchtext==(
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torchtext==(
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.11/dist-packages (from torc
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.11/dist-packages (from tor
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.11/dist-packages (from tor
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.11/dist-packages (from t
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in /usr/local/lib/python3.11/dist-packages (from torch==
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from torch
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.2.0->torc
Requirement already satisfied: urllib3>=1.25 in /usr/local/lib/python3.11/dist-packages (from torchdata==0.7.1->
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.11/dist-packages (from nvidia-cus
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->torchtext
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->tor
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch==2
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->torch=
Using cached torch-2.2.0-cp311-cp311-manylinux1_x86_64.whl (755.5 MB)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
Installing collected packages: torch
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This
torchaudio 2.6.0+cu124 requires torch==2.6.0, but you have torch 2.2.0 which is incompatible.
torchvision 0.21.0+cu124 requires torch==2.6.0, but you have torch 2.2.0 which is incompatible.
torchtune 0.6.1 requires torchdata==0.11.0, but you have torchdata 0.7.1 which is incompatible.
Successfully installed torch
```

```
import torch
import torchtext
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

device= torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Device availble for runing:")
print(device)
```

```
Device availble for runing:
cpu
```

```
def my_function():
        import seaborn as sns
        # ... rest of the code ...
```

```
# Install/upgrade necessary packages before importing
!pip install numpy
!pip install matplotlib
!pip install librosa

# Now import the libraries
```

```
import pandas as pd
import numpy as np
import os
import shutil
import seaborn as sns  # seaborn imported before pyplot
import librosa
import librosa.display
import tensorflow as tf
from IPython.display import Audio
import warnings
import matplotlib.pyplot as plt  # pyplot imported after seaborn


warnings.filterwarnings('ignore')
%matplotlib inline
```

```
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->m
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
Requirement already satisfied: librosa in /usr/local/lib/python3.11/dist-packages (0.11.0)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.11/dist-packages (from librosa) (3.0.1
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (0.60.0)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from librosa) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.15.3)
Requirement already satisfied: scikit-learn>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.5.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (4.4.2
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.11/dist-packages (from librosa) (0.13
Requirement already satisfied: pooch>=1.1 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.8.2)
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.11/dist-packages (from librosa) (0.5.0.post
Requirement already satisfied: typing_extensions>=4.1.1 in /usr/local/lib/python3.11/dist-packages (from librosa
Requirement already satisfied: lazy_loader>=0.1 in /usr/local/lib/python3.11/dist-packages (from librosa) (0.4)
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.1.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from lazy_loader>=0.1->libr
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.11/dist-packages (from pooch>=1.1->
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.11/dist-packages (from pooch>=1.1->lib
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-lear
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.11/dist-packages (from soundfile>=0.12.1->lib
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0->soundfile>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->p
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.1
WARNING: Ignoring invalid distribution ~orch (/usr/local/lib/python3.11/dist-packages)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
```

```
paths = []
labels = []
for dirname, _, filenames in os.walk("/content/drive/MyDrive/TESS"):
```

```
for dirname, _, filenames in os.walk('/content/drive/MyDrive/TESS'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
    if len(paths) == 2800:
        break
print('Dataset is Loaded')
```

```
Dataset is Loaded
```

```
len(paths)
```

```
2800
```

```
paths[:5]
```

```
['/content/drive/MyDrive/TESS/YAF_sad/YAF_bath_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_bar_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_base_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_back_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_bought_sad.wav']
```

```
paths[2:6]
```

```
['/content/drive/MyDrive/TESS/YAF_sad/YAF_base_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_back_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_bought_sad.wav',
 '/content/drive/MyDrive/TESS/YAF_sad/YAF_bean_sad.wav']
```

```
labels[2:6]
```

```
['sad', 'sad', 'sad', 'sad']
```

```
labels[:5]
```

```
['sad', 'sad', 'sad', 'sad', 'sad']
```

```
df = pd.DataFrame()
df['speech'] = paths
df['label'] = labels
df.head()
```

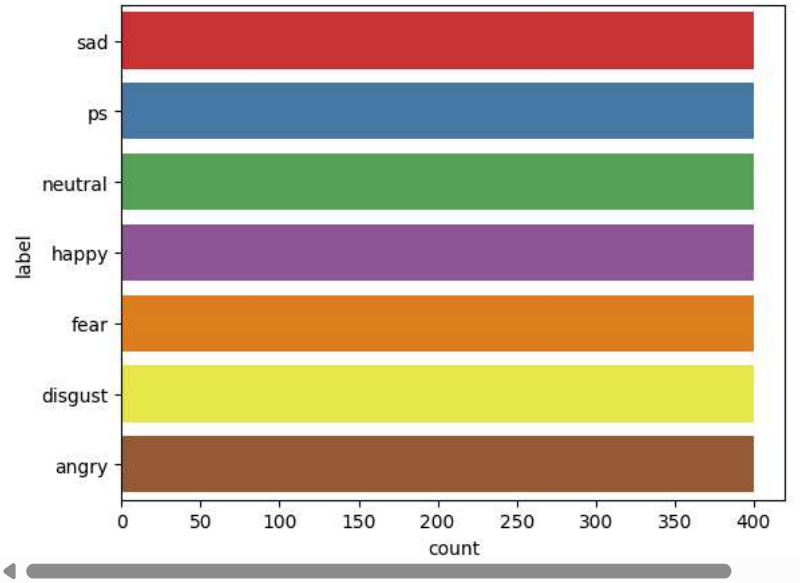
	speech	label
0	/content/drive/MyDrive/TESS/YAF_sad/YAF_bath_s...	sad
1	/content/drive/MyDrive/TESS/YAF_sad/YAF_bar_sa...	sad
2	/content/drive/MyDrive/TESS/YAF_sad/YAF_base_s...	sad
3	/content/drive/MyDrive/TESS/YAF_sad/YAF_back_s...	sad
4	/content/drive/MyDrive/TESS/YAF_sad/YAF_bought...	sad

```
df['label'].value_counts()
```

label	count
sad	400
ps	400
neutral	400
happy	400
fear	400
disgust	400
angry	400

dtype: int64

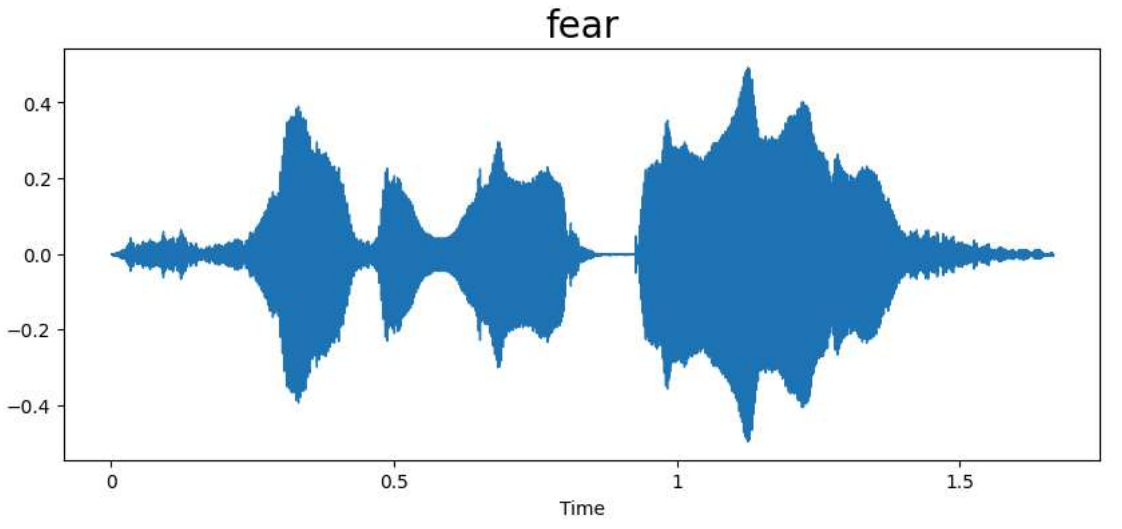
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame containing the 'label' column
sns.countplot(data=df, y='label', palette='Set1')
plt.show()
```



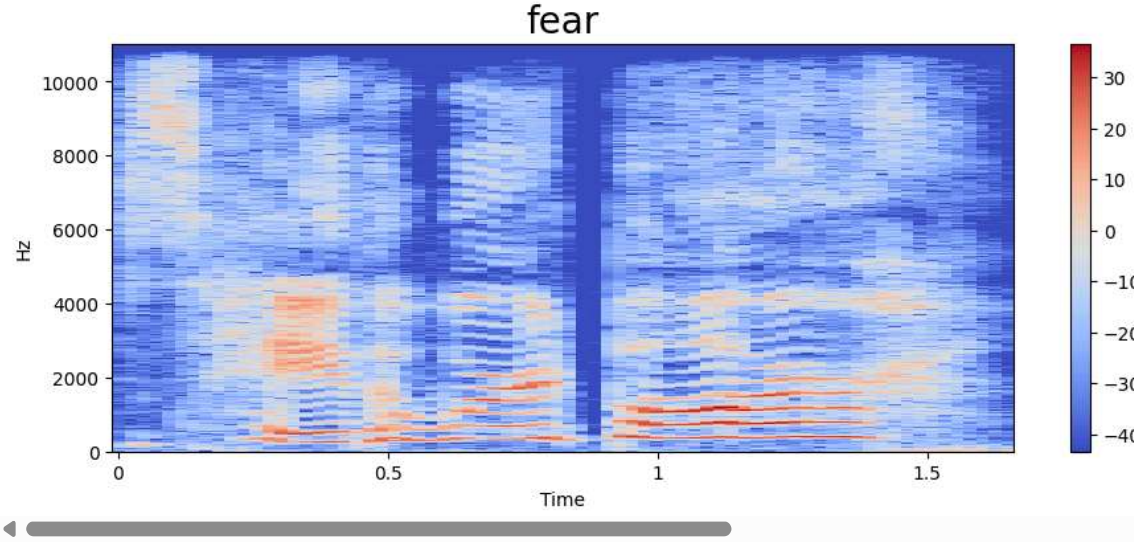

```
def waveplot(data, sr, emotion):
    plt.figure(figsize=(10,4))
    plt.title(emotion, size=20)
    librosa.display.waveshow(data, sr=sr)
    plt.show()

def spectogram(data, sr, emotion):
    x = librosa.stft(data)
    xdb = librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(11,4))
    plt.title(emotion, size=20)
    librosa.display.specshow(xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()
```

```
emotion = 'fear'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
```

```
spectogram(data, sampling_rate, emotion)
Audio(path)
```



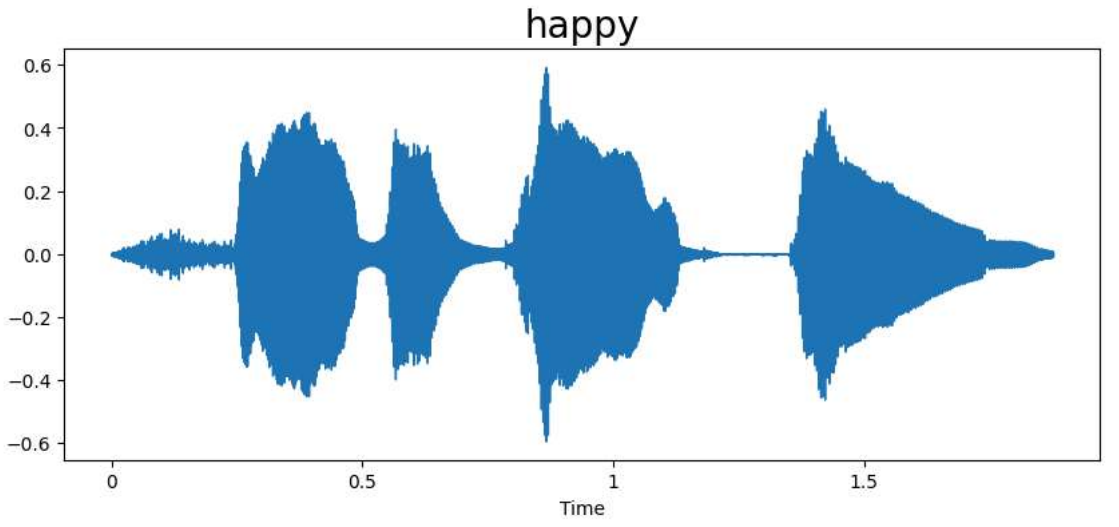
0:00 / 0:01



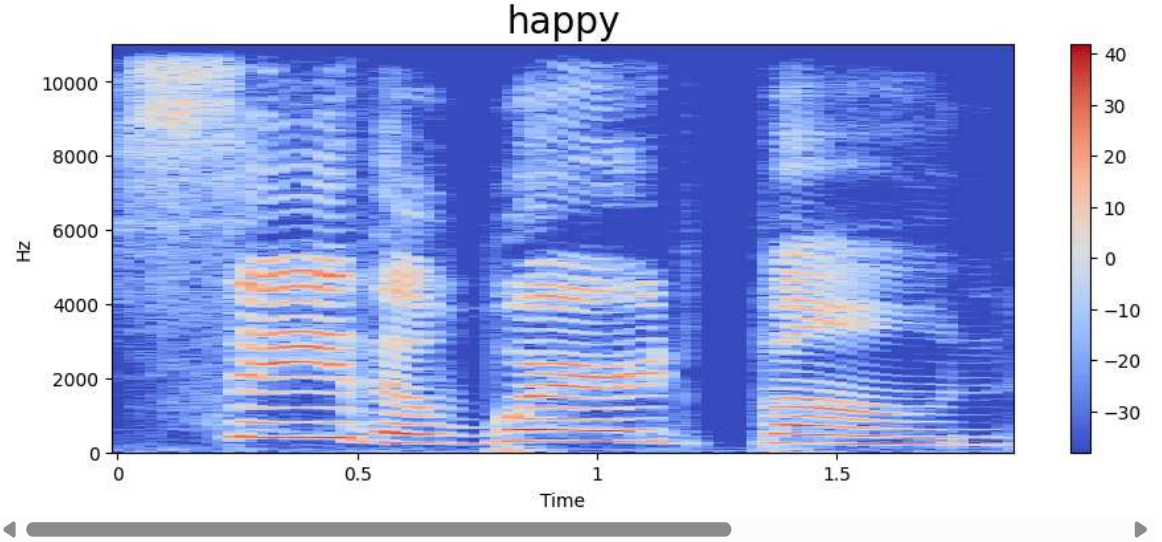

```
print(sampling_rate)
```

```
22050
```

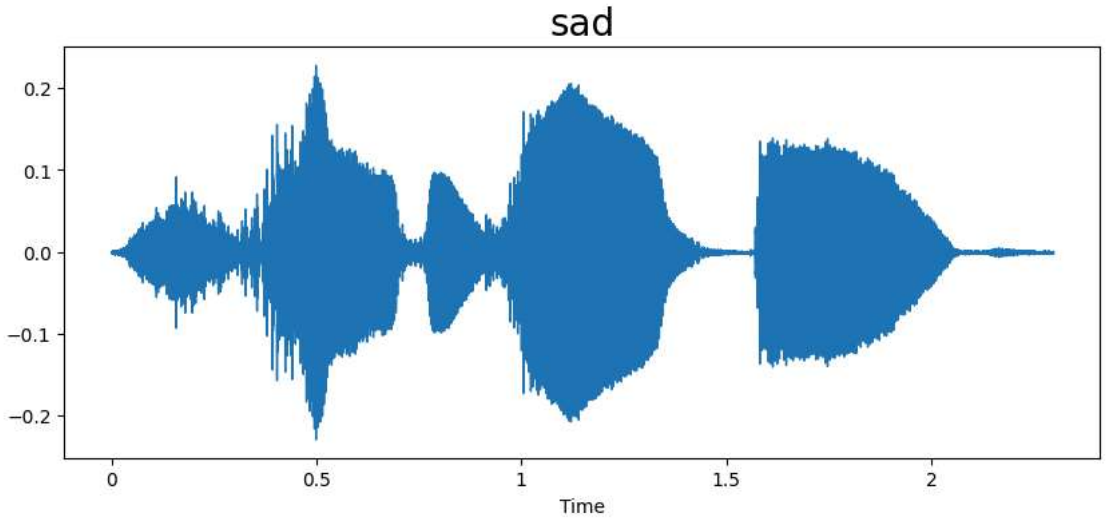
```
emotion = 'happy'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



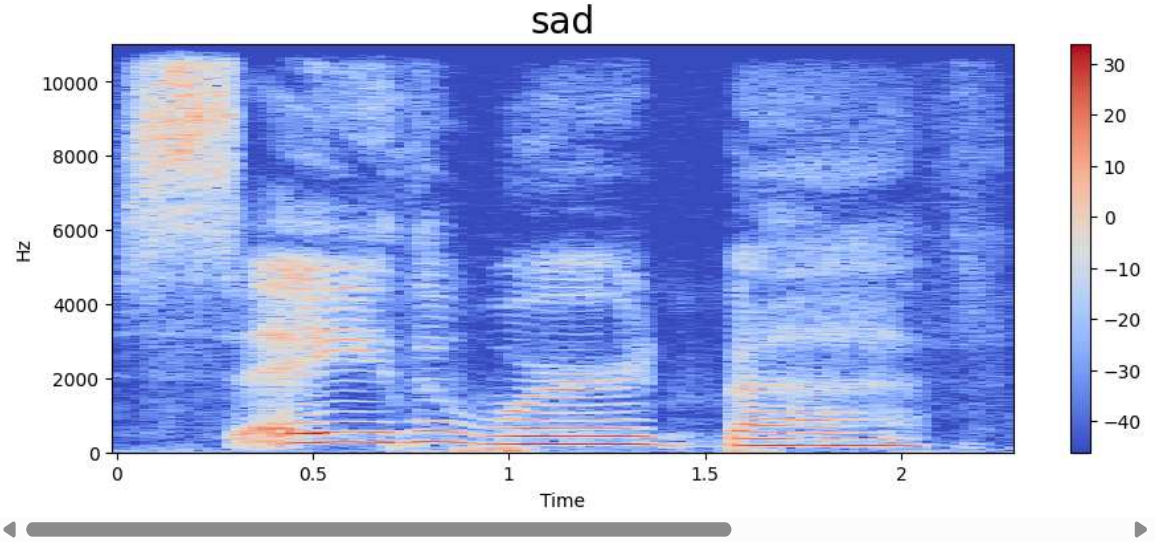
0:00 / 0:01



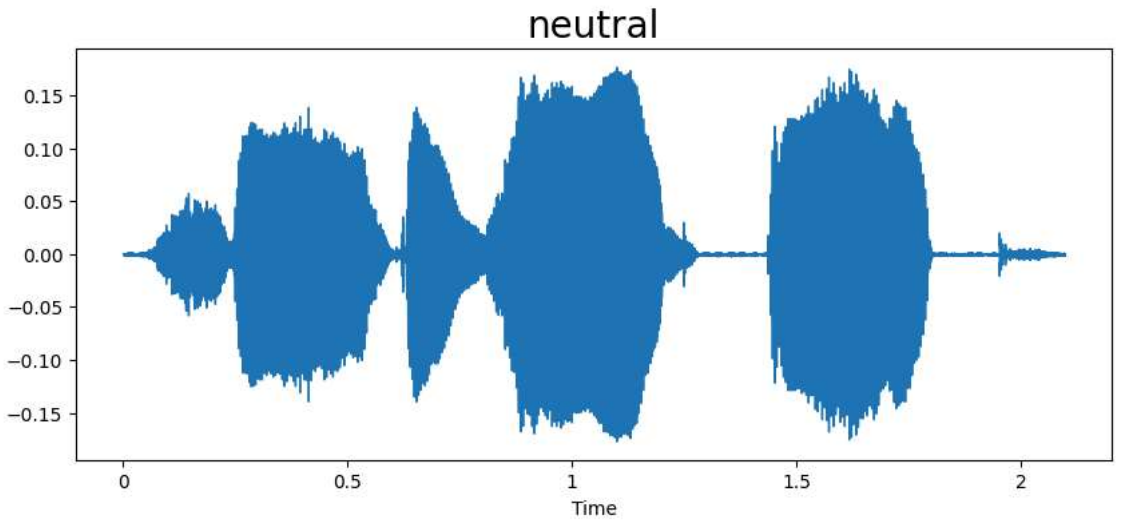

```
emotion = 'sad'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



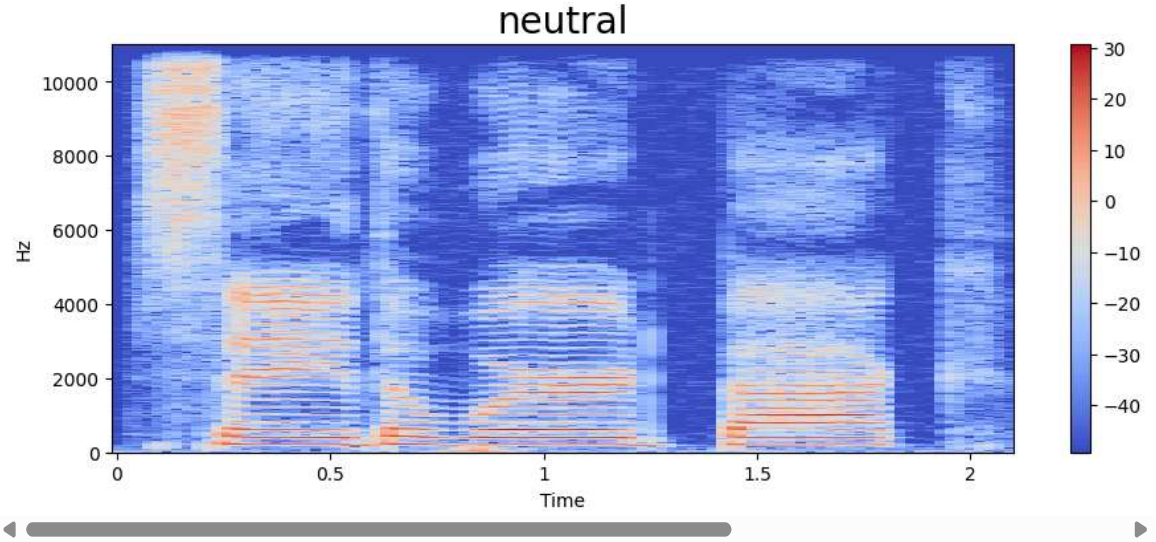
0:00 / 0:02



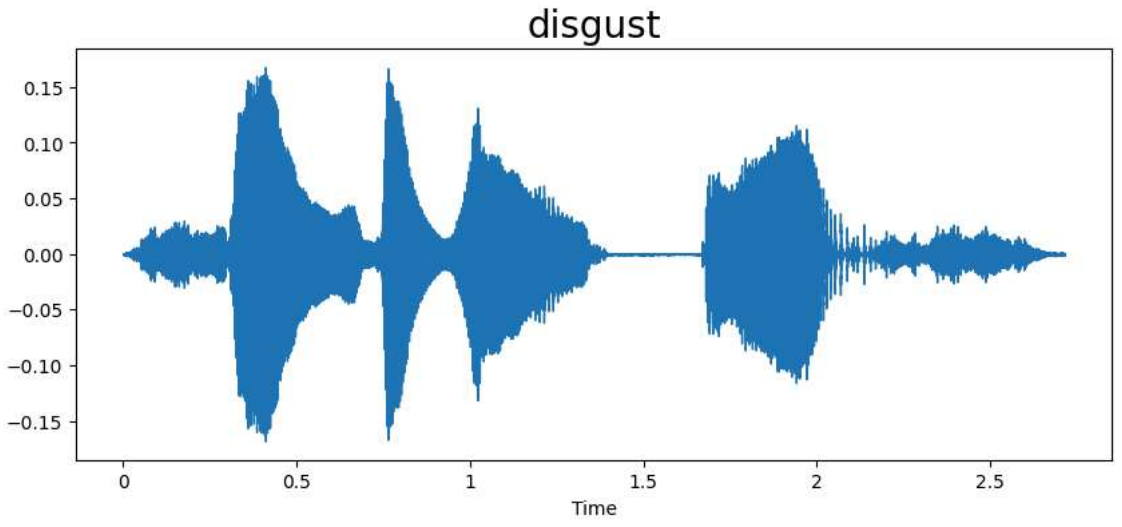

```
emotion = 'neutral'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



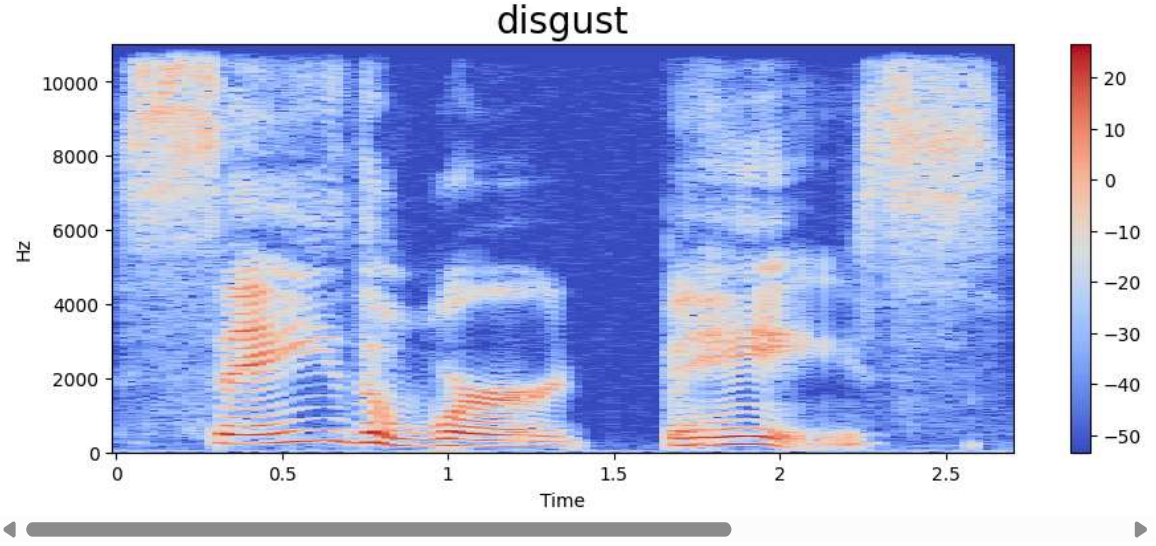
0:00 / 0:02



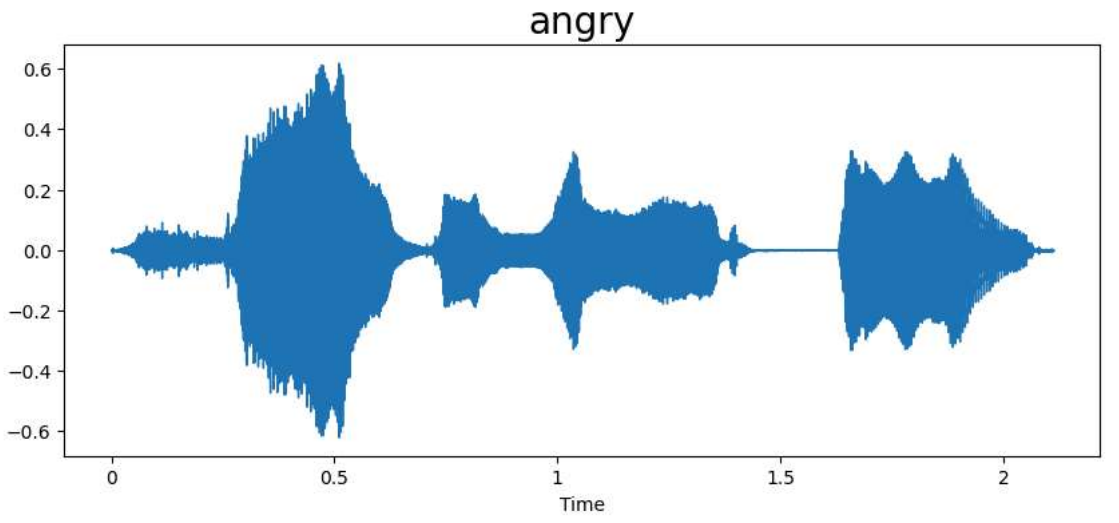

```
emotion = 'disgust'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



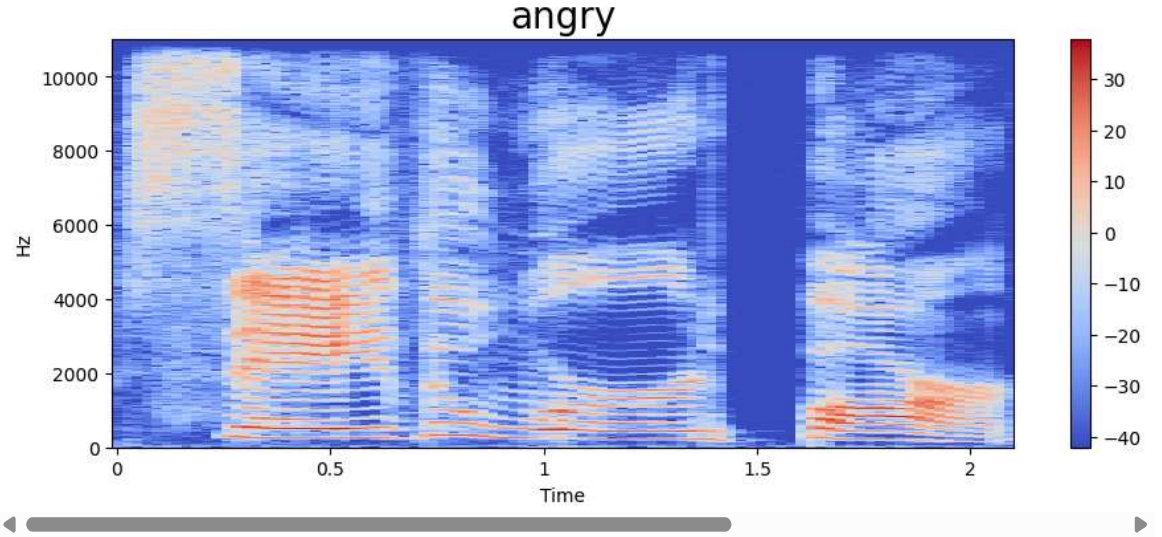
0:00 / 0:02



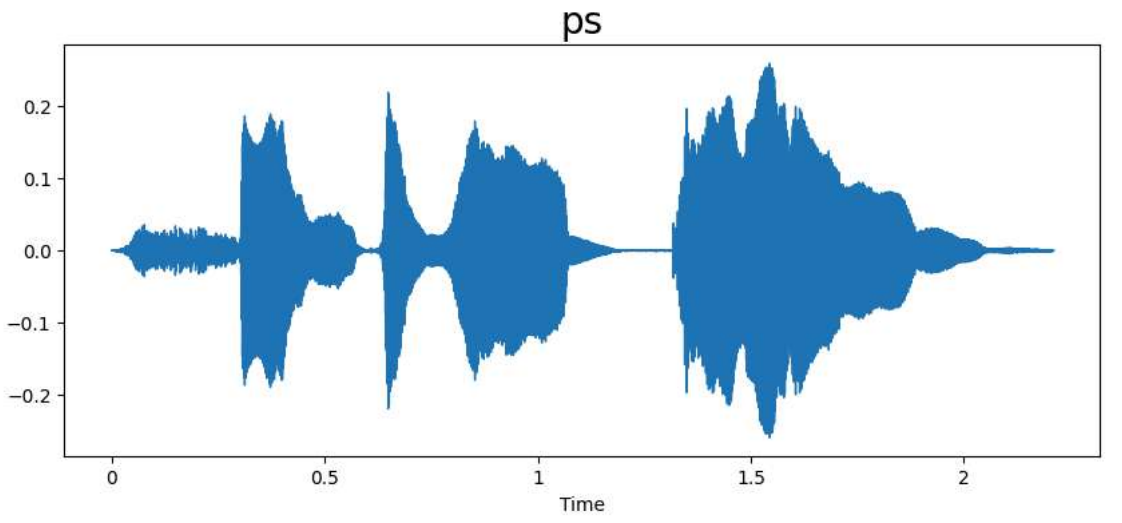

```
emotion = 'angry'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



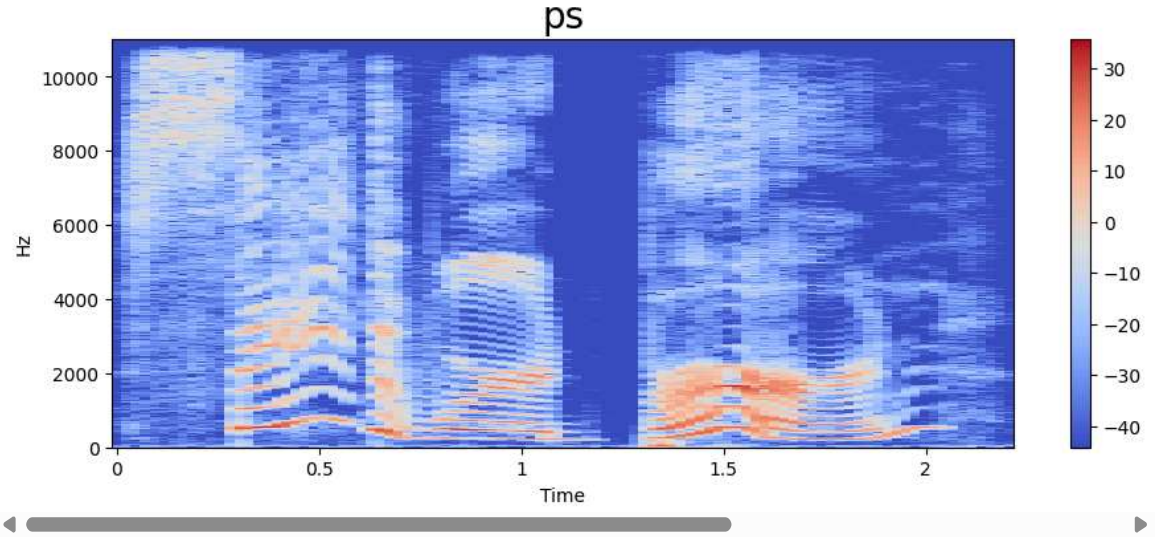
0:00 / 0:02




```
emotion = 'ps'
path = np.array(df['speech'][df['label']==emotion])[0]
data, sampling_rate = librosa.load(path)
waveplot(data, sampling_rate, emotion)
spectogram(data, sampling_rate, emotion)
Audio(path)
```



0:00 / 0:02




```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3)
    mfccs = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)

    # Feature Scaling and Normalization
    mfccs = librosa.util.normalize(mfccs, axis=0)  # Normalize each MFCC frame

    return mfccs


# Apply scaling using StandardScaler
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_mfcc = df['speech'].apply(lambda x: extract_mfcc(x))
X = np.array(X_mfcc.tolist())
X_scaled = scaler.fit_transform(X)
```

```
extract_mfcc(df['speech'][0])
```

```
array([-1.0000000e+00,  1.6540602e-01,  4.3245755e-02,  6.8328097e-02,
       -2.2012913e-03,  8.2453499e-03, -9.7525558e-03, -2.6535735e-02,
        1.1242810e-02, -1.5065876e-02, -1.4571326e-02, -6.3275066e-03,
       -1.5172775e-02,  2.5929630e-02, -2.9504875e-02, -3.5805996e-03,
       -1.8929054e-03, -4.3387120e-03, -1.9237502e-02, -8.9127533e-03,
       -2.3615330e-02, -1.5722446e-02, -1.2458775e-02,  5.8494275e-03,
        5.2592938e-04,  1.4868785e-02,  3.0491126e-03, -1.3387287e-03,
       -6.1408132e-03,  8.5776690e-03,  2.3705829e-03,  3.7431289e-02,
        1.6041353e-02,  1.4511839e-02, -1.3998607e-03, -2.0004266e-03,
        8.3303032e-03,  1.3856082e-02,  2.1118641e-02,  1.6593169e-02],
      dtype=float32)
```

```
X = X_scaled  # Replace X with X_scaled
X.shape
```

```
(2800, 40)
```

```
X = np.expand_dims(X, -1)
X.shape
```

```
(2800, 40, 1)
```

```
print(X.shape)
```

```
(2800, 40, 1)
```

```
# Reshape X for temporal context
num_samples = X.shape[0]
num_timesteps = 40  # Experiment with this value
num_features = X.shape[1] // num_timesteps
X_reshaped = X.reshape(num_samples, num_timesteps, num_features)
```

```
print(X_reshaped.shape)
```

```
(2800, 40, 1)
```

```
mfcc_df = pd.DataFrame(X_mfcc.tolist())
mfcc_df['label'] = df['label']
```

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
y = enc.fit_transform(df[['label']])
```

```
y = y.toarray()
```

```
y.shape
```

```
(2800, 7)
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
import numpy as np

# 1. Split into train and test sets with stratification
X_train, X_test, y_train, y_test = train_test_split(
    X_reshaped, y, test_size=0.2, stratify=y, random_state=42
)

# 2. Split train set further into train and validation sets with stratification
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.25, stratify=y_train, random_state=42
```

```
)  # 0.25 x 0.8 = 0.2

# Create the LSTM model with more layers and adjustments for robustness
model = Sequential([
    LSTM(256, return_sequences=True, input_shape=(num_timesteps, num_features)),  # First LSTM layer with recurrent_
    Dropout(0.4),
    LSTM(128, return_sequences=True),  # Second LSTM layer with recurrent_dropout
    Dropout(0.4),
    LSTM(64, return_sequences=False),  # Third LSTM layer with recurrent_dropout
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),  # First dense layer
    Dropout(0.4),
    Dense(64, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),  # Second dense layer
    Dropout(0.4),
    Dense(7, activation='softmax')  # Output layer
])

# Early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0005)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.summary()
model.save('lstm_model.keras')
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 40, 256)	264,192
dropout (Dropout)	(None, 40, 256)	0
lstm_1 (LSTM)	(None, 40, 128)	197,120
dropout_1 (Dropout)	(None, 40, 128)	0
lstm_2 (LSTM)	(None, 64)	49,408
dropout_2 (Dropout)	(None, 64)	0
dense (Dense)	(None, 128)	8,320
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_4 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455

Total params: 527.751 (2.01 MB)

```
import os
print(os.getcwd())
```

```
/content
```

```
print(X_reshaped.shape)
```

```
(2800, 40, 1)
```

```
print(X_train.shape)
print(y_train.shape)
```

```
(1680, 40, 1)
(1680, 7)
```

```
print(X_test.shape)
print(y_test.shape)
```

```
(560, 40, 1)
(560, 7)
```

```
print(X_val.shape)
print(y_val.shape)
```

```
(560, 40, 1)
(560, 7)
```

```
import numpy as np
from tensorflow.keras.callbacks import EarlyStopping

# 2. Train the model
history = model.fit(
    X_train,
    y_train,
    batch_size=48,
    epochs=48,  # You can adjust the number of epochs
    validation_data=(X_val, y_val),
    callbacks=[early_stopping]
)

# 3. Evaluate the model on the validation set
test_loss, test_accuracy = model.evaluate(X_val, y_val)
print("Mean accuracy of the trained model on the validation data: {:.2f}%".format(test_accuracy * 100))
```

```
Epoch 1/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 18s 353ms/step - accuracy: 0.2371 - loss: 3.4836 - val_accuracy: 0.4357 - val_loss:
Epoch 2/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 338ms/step - accuracy: 0.4300 - loss: 2.7612 - val_accuracy: 0.6143 - val_loss:
Epoch 3/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 19s 286ms/step - accuracy: 0.5692 - loss: 2.2307 - val_accuracy: 0.8250 - val_loss:
Epoch 4/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 331ms/step - accuracy: 0.7335 - loss: 1.7325 - val_accuracy: 0.8929 - val_loss:
Epoch 5/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 329ms/step - accuracy: 0.8324 - loss: 1.3798 - val_accuracy: 0.9429 - val_loss:
Epoch 6/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 329ms/step - accuracy: 0.9063 - loss: 1.1356 - val_accuracy: 0.9536 - val_loss:
Epoch 7/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 329ms/step - accuracy: 0.9161 - loss: 1.0113 - val_accuracy: 0.9268 - val_loss:
Epoch 8/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 331ms/step - accuracy: 0.9269 - loss: 0.9102 - val_accuracy: 0.9375 - val_loss:
Epoch 9/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 328ms/step - accuracy: 0.8873 - loss: 0.9315 - val_accuracy: 0.9393 - val_loss:
Epoch 10/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 331ms/step - accuracy: 0.9245 - loss: 0.7873 - val_accuracy: 0.9500 - val_loss:
Epoch 11/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 331ms/step - accuracy: 0.9336 - loss: 0.7233 - val_accuracy: 0.9357 - val_loss:
Epoch 12/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 352ms/step - accuracy: 0.9473 - loss: 0.6571 - val_accuracy: 0.9554 - val_loss:
Epoch 13/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 331ms/step - accuracy: 0.9464 - loss: 0.6019 - val_accuracy: 0.9357 - val_loss:
Epoch 14/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 343ms/step - accuracy: 0.9517 - loss: 0.5830 - val_accuracy: 0.9554 - val_loss:
Epoch 15/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 19s 283ms/step - accuracy: 0.9520 - loss: 0.5170 - val_accuracy: 0.9571 - val_loss:
Epoch 16/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 330ms/step - accuracy: 0.9734 - loss: 0.4414 - val_accuracy: 0.9464 - val_loss:
Epoch 17/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 332ms/step - accuracy: 0.9612 - loss: 0.4322 - val_accuracy: 0.9607 - val_loss:
Epoch 18/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 335ms/step - accuracy: 0.9725 - loss: 0.3970 - val_accuracy: 0.9625 - val_loss:
Epoch 19/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 10s 287ms/step - accuracy: 0.9700 - loss: 0.3773 - val_accuracy: 0.9643 - val_loss:
Epoch 20/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 324ms/step - accuracy: 0.9746 - loss: 0.3517 - val_accuracy: 0.9482 - val_loss:
Epoch 21/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 341ms/step - accuracy: 0.9697 - loss: 0.3630 - val_accuracy: 0.9696 - val_loss:
Epoch 22/48
```

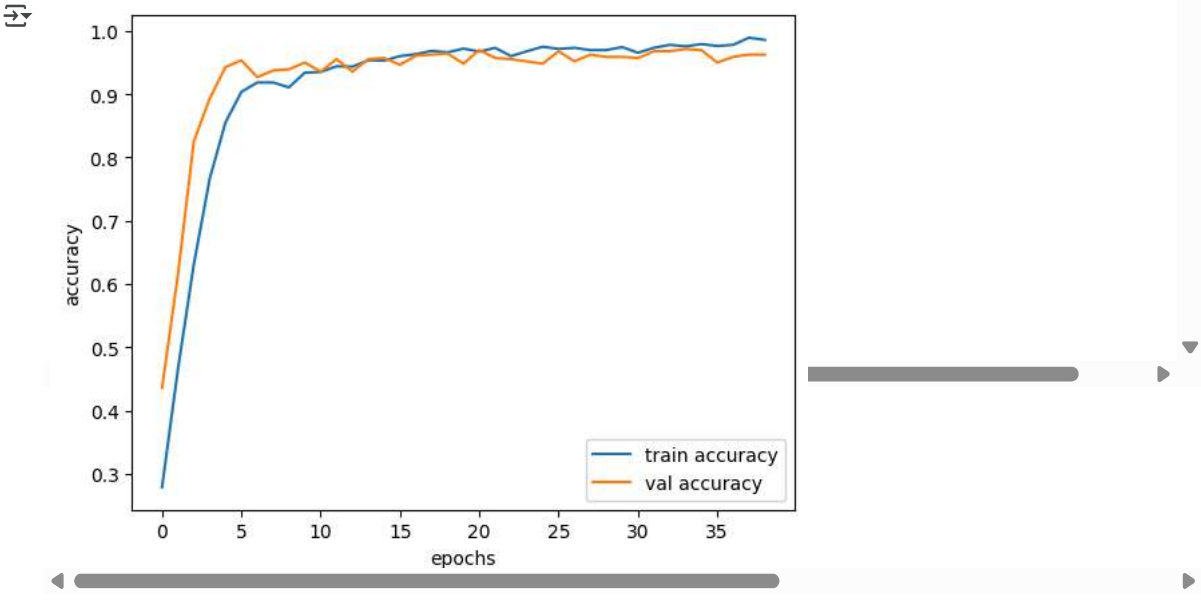
```
35/35 ━━━━━━━━━━━━━━━━━━━━ 12s 341ms/step - accuracy: 0.9704 - loss: 0.3576 - val_accuracy: 0.9571 - val_loss:
Epoch 23/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 342ms/step - accuracy: 0.9585 - loss: 0.3425 - val_accuracy: 0.9554 - val_loss:
Epoch 24/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 19s 310ms/step - accuracy: 0.9715 - loss: 0.3062 - val_accuracy: 0.9518 - val_loss:
Epoch 25/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 333ms/step - accuracy: 0.9738 - loss: 0.2867 - val_accuracy: 0.9482 - val_loss:
Epoch 26/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 319ms/step - accuracy: 0.9789 - loss: 0.2661 - val_accuracy: 0.9679 - val_loss:
Epoch 27/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 21s 340ms/step - accuracy: 0.9725 - loss: 0.2722 - val_accuracy: 0.9518 - val_loss:
Epoch 28/48
35/35 ━━━━━━━━━━━━━━━━━━━━ 20s 323ms/step - accuracy: 0.9652 - loss: 0.2912 - val_accuracy: 0.9625 - val_loss:
Epoch 29/48
```

```
model.save('lstm_model.keras')
```

```
print(X_train.shape)
print(y_train.shape)
print(X_val.shape)
print(y_val.shape)
```

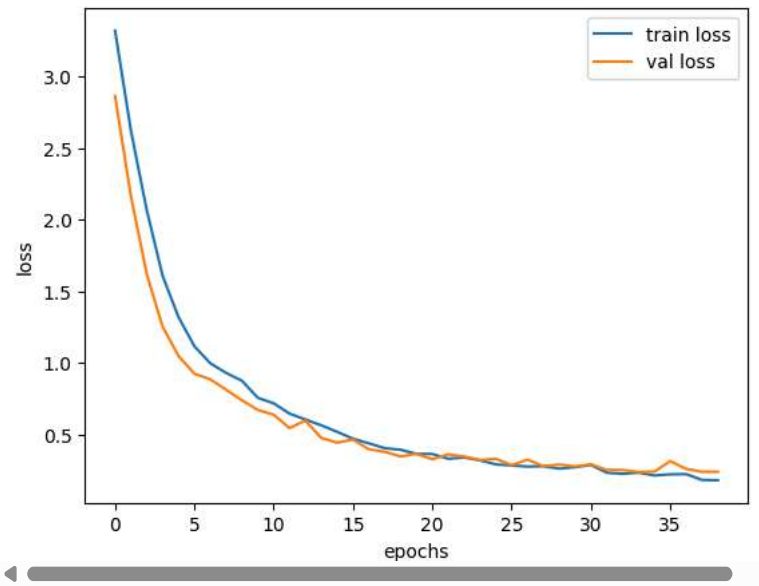
```
(1680, 40, 1)
(1680, 7)
(560, 40, 1)
(560, 7)
```

```
epochs = list(range(39))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, label='train accuracy')
plt.plot(epochs, val_acc, label='val accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```




```
epochs = list(range(39))
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(epochs, loss, label='train loss')
plt.plot(epochs, val_loss, label='val loss')
```

```
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

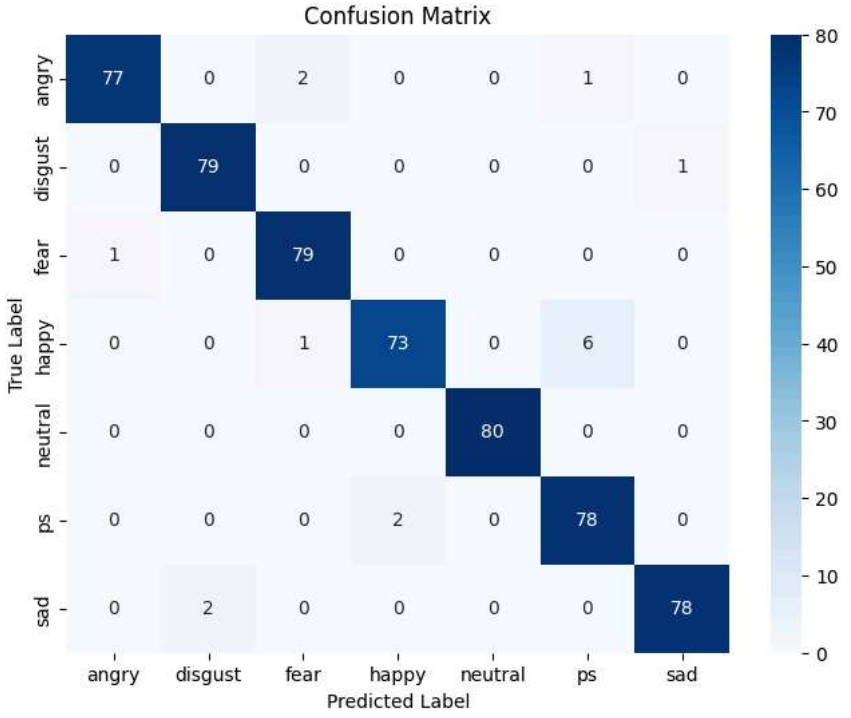



```
from sklearn.metrics import classification_report
predictions = model.predict(X_val)

class_names= np.argmax(predictions, axis=1)
new_Val = np.argmax(y_val, axis=1)

report = classification_report(new_Val, class_names)
print(report)
# In[30]:
from sklearn.metrics import confusion_matrix


import matplotlib.pyplot as plt


from sklearn.metrics import confusion_matrix
predictions = model.predict(X_val, steps=len(X_val), verbose=0)
y_true=np.argmax(new_Val, axis=-1)

cm = confusion_matrix(new_Val, class_names)
print(cm)
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 3s 134ms/step
              precision    recall  f1-score   support

           0       0.99      0.96      0.97        80
           1       0.98      0.99      0.98        80
           2       0.96      0.99      0.98        80
           3       0.97      0.91      0.94        80
           4       1.00      1.00      1.00        80
           5       0.92      0.97      0.95        80
           6       0.99      0.97      0.98        80

    accuracy                           0.97       560
   macro avg       0.97      0.97      0.97       560
weighted avg       0.97      0.97      0.97       560

[[77  0  2  0  0  1  0]
 [ 0 79  0  0  0  0  1]
 [ 1  0 79  0  0  0  0]
 [ 0  0  1 73  0  6  0]
 [ 0  0  0  0 80  0  0]
```

```
       [ 0  0  0  2  0 78  0]
       [ 0  2  0  0  0  0 78]]
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

# Assuming you have already calculated 'predictions', 'class_names', and 'new_Ytest'

# Plotting the confusion matrix
cm = confusion_matrix(new_Val, class_names)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=enc.categories_[0], yticklabels=enc.categories_[0])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# Printing the classification report
report = classification_report(new_Val, class_names)
print(report)
```




```
              precision    recall  f1-score   support

           0       0.99      0.96      0.97        80
           1       0.98      0.99      0.98        80
           2       0.96      0.99      0.98        80
           3       0.97      0.91      0.94        80
           4       1.00      1.00      1.00        80
           5       0.92      0.97      0.95        80
           6       0.99      0.97      0.98        80

    accuracy                           0.97       560
   macro avg       0.97      0.97      0.97       560
weighted avg       0.97      0.97      0.97       560
```

```
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Load the saved model
model = tf.keras.models.load_model('lstm_model.keras')

# Assuming you have X_test and y_test from your train_test_split
predictions = model.predict(X_test)

# Convert predictions to class labels
predicted_labels = np.argmax(predictions, axis=1)
true_labels = np.argmax(y_test, axis=1)

# Calculate accuracy
accuracy = accuracy_score(true_labels, predicted_labels)
print(f"Accuracy: {accuracy}")

# Generate classification report
report = classification_report(true_labels, predicted_labels)
```