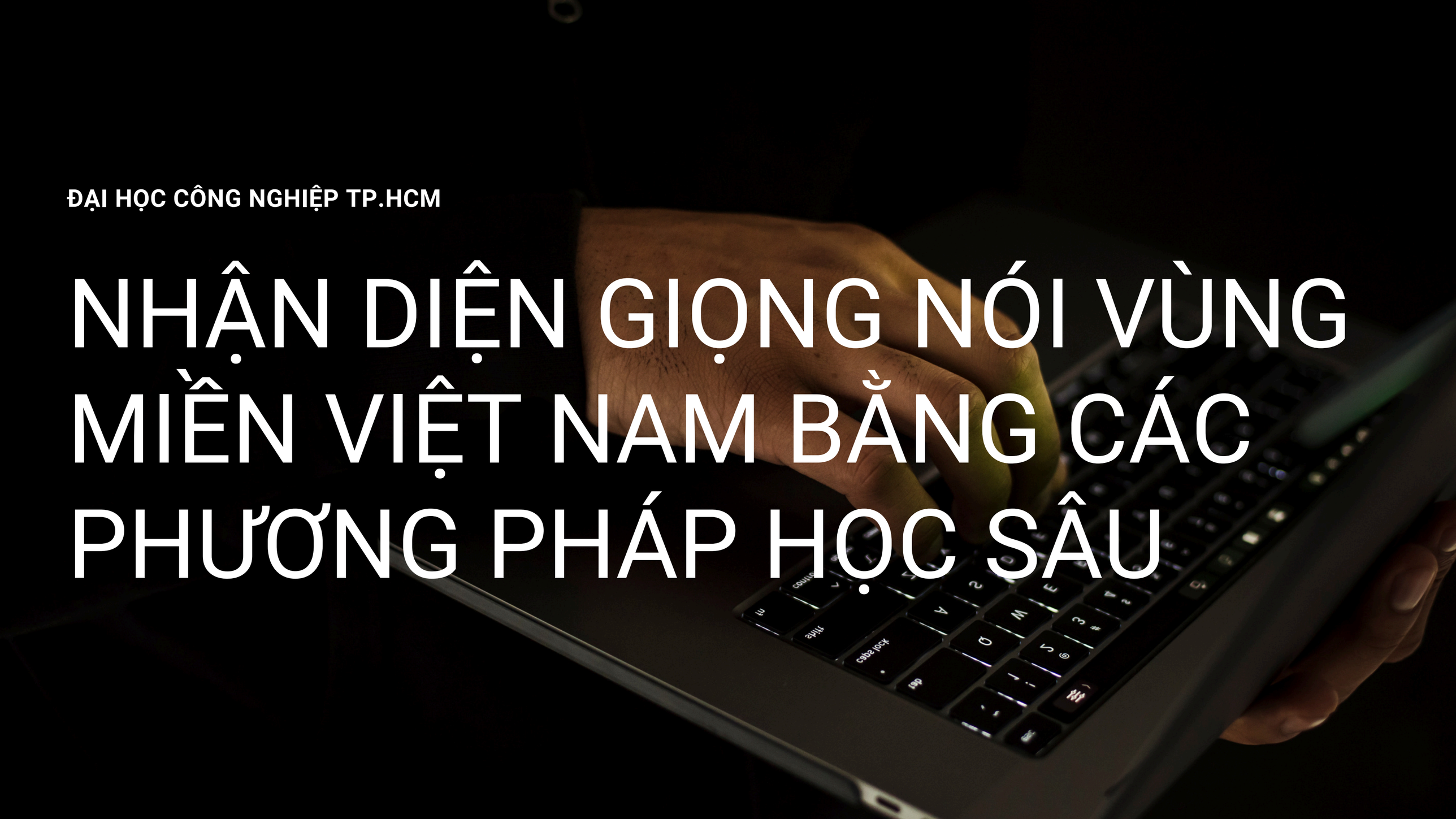
ĐẠI HỌC CÔNG NGHIỆP TP.HCM

NHẬN DIỆN GIỌNG NÓI VÙNG MIỀN VIỆT NAM BẰNG CÁC PHƯƠNG PHÁP HỌC SÂU



MỤC LỤC

Tóm tắt vấn đề

Nhận diện giọng nói các vùng miền là gì?

- Nhận diện giọng nói vùng miền là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói.

- Với sự đa dạng văn hóa giữa các vùng miền thì nhận diện giọng nói từng vùng miền là một thách thức không nhỏ.

Hướng giải quyết vấn đề

01 Thu thập dữ liệu

02 Đọc - trích xuất đặc trưng và xử lý dữ liệu

03 Biểu đồ tương quan

04 Sử dụng mô hình học sâu

05 So sánh và đưa ra mô hình tốt nhất

Thực hiện giải quyết vấn đề

Thu thập dữ liệu

Chúng em sử dụng tập dữ liệu VIVOS là kho ngữ liệu tiếng Việt miễn phí bao gồm 15 giờ ghi âm giọng nói được chuẩn bị cho nhiệm vụ Nhận dạng giọng nói tự động. Ngữ liệu được biên soạn bởi AILAB, phòng thí nghiệm khoa học máy tính của Đại học Khoa học Tự nhiên - ĐHQG-HCM, do GS. Vũ Hải Quân làm trưởng phòng. Nhóm đã sử dụng bộ dữ liệu này và label lại các tệp âm thanh theo vùng miền Việt Nam, tập dữ liệu của nhóm bao gồm 46 speaker với 6,297 tệp âm thanh.

Do chúng em không tìm được dữ liệu sẵn có để giải quyết vấn đề trên, vậy nên phương pháp đề xuất của chúng em là tìm tập dữ liệu tiếng việt đã có sẵn và đánh label lại vùng miền cho các tập dữ liệu đó.

Thu thập dữ liệu

Khi đã tìm được bộ dữ liệu thì nhóm em tiến hành đánh label lại cho tất cả các file để phân loại từng miền. Sau khi ngồi nghe và đánh nhãn cho từng file âm thanh. Có tổng cộng 46 người được dùng làm bộ dữ liệu này và chúng em đã phân loại được ra như thế này

```
mienbac_speakers = ["VIVOSSPK01", "VIVOSSPK14", "VIVOSSPK17", "VIVOSSPK22", "VIVOSSPK27", "VIVOSSPK30", "VIVOSSPK41", "VIVOSSPK44"]
mientrung_speakers = ["VIVOSSPK12", "VIVOSSPK13", "VIVOSSPK21", "VIVOSSPK26", "VIVOSSPK33", "VIVOSSPK37", "VIVOSSPK39"]
miennam_speakers = ["VIVOSSPK02", "VIVOSSPK03", "VIVOSSPK04", "VIVOSSPK05", "VIVOSSPK06", "VIVOSSPK07", "VIVOSSPK08", "VIVOSSPK09", "VIVOSSPK10", "VIVOSSPK11", "VIVOSSPK15",
                    "VIVOSSPK16", "VIVOSSPK18", "VIVOSSPK19", "VIVOSSPK20", "VIVOSSPK23",  "VIVOSSPK25", "VIVOSSPK28", "VIVOSSPK29", "VIVOSSPK31", "VIVOSSPK32", "VIVOSSPK34",
                    "VIVOSSPK35", "VIVOSSPK36", "VIVOSSPK38", "VIVOSSPK40", "VIVOSSPK42", "VIVOSSPK43", "VIVOSSPK45", "VIVOSSPK46"]
```

Mỗi một người thì có khoảng hơn 100 file âm thanh. Kết quả sau khi tổng hợp được số lượng thì như sau:

+ Miền Nam: 3000 file
+ Miền Bắc: 2147 file
+ Miền Trung: 1150 file

Đọc dữ liệu

```
# Định nghĩa các đường dẫn tới thư mục chứa dữ liệu âm thanh
path_1 = r"D:\HK7\DL\CK\data"

# Khởi tạo danh sách để lưu dữ liệu
data = []

for root, dirs, files in os.walk(path_1):  # Duyệt qua cả thư mục con
    for filename in files:
        if filename.endswith(".wav"):  # Kiểm tra định dạng .wav
            filepath = os.path.join(root, filename)  # Đường dẫn đầy đủ
```

- Duyệt thư mục chứa các file .wav bằng os.walk.
- Lọc file theo định dạng .wav.

Trích xuất đặc trưng

- Dùng librosa để tính toán các đặc trưng:
- Chroma STFT (Tần số sắc thái).
- RMSE (Năng lượng tín hiệu).
- Spectral Centroid (Trung tâm phổ tần số).
- Spectral Bandwidth (Độ rộng băng thông phổ).
- Spectral Rolloff (Tần số cắt phổ).
- MFCC (Hệ số Mel Frequency Cepstral).

```
y, sr = librosa.load(filepath, mono=True, duration=10)
chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
rmse = librosa.feature.rms(y=y)
spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)

# Tính giá trị trung bình của các đặc trưng
features = {
    "filename": filename,
    "chroma_stft": np.mean(chroma_stft),
    "rmse": np.mean(rmse),
    "spectral_centroid": np.mean(spec_cent),
    "spectral_bandwidth": np.mean(spec_bw),
    "rolloff": np.mean(rolloff)
}
for i, mfcc_coef in enumerate(mfcc):
    features[f"mfcc{i+1}"] = np.mean(mfcc_coef)
```

Xử lý dữ liệu

Dán nhãn theo vùng miền dựa trên thư mục cha

- MienBac → North.
- MienNam → Southern.
- MienTrung → Central.

```
# Phân loại nhãn dựa trên thư mục cha
if "MienBac" in root:
    features["label"] = "North"
elif "MienNam" in root:
    features["label"] = "Southern"
elif "MienTrung" in root:
    features["label"] = "Central"
else:
    features["label"] = "Unknow"
```

Xử lý dữ liệu

```
df['label']=df['label'].replace(['North'],1)
df['label']=df['label'].replace(['Southern'],2)
df['label']=df['label'].replace(['Central'],3)
```

Encoding

Chuyển đổi dữ liệu cột label từ dạng text sang dạng numberic mục đích là để dữ liệu tương thích với mô hình.

Xử lý dữ liệu

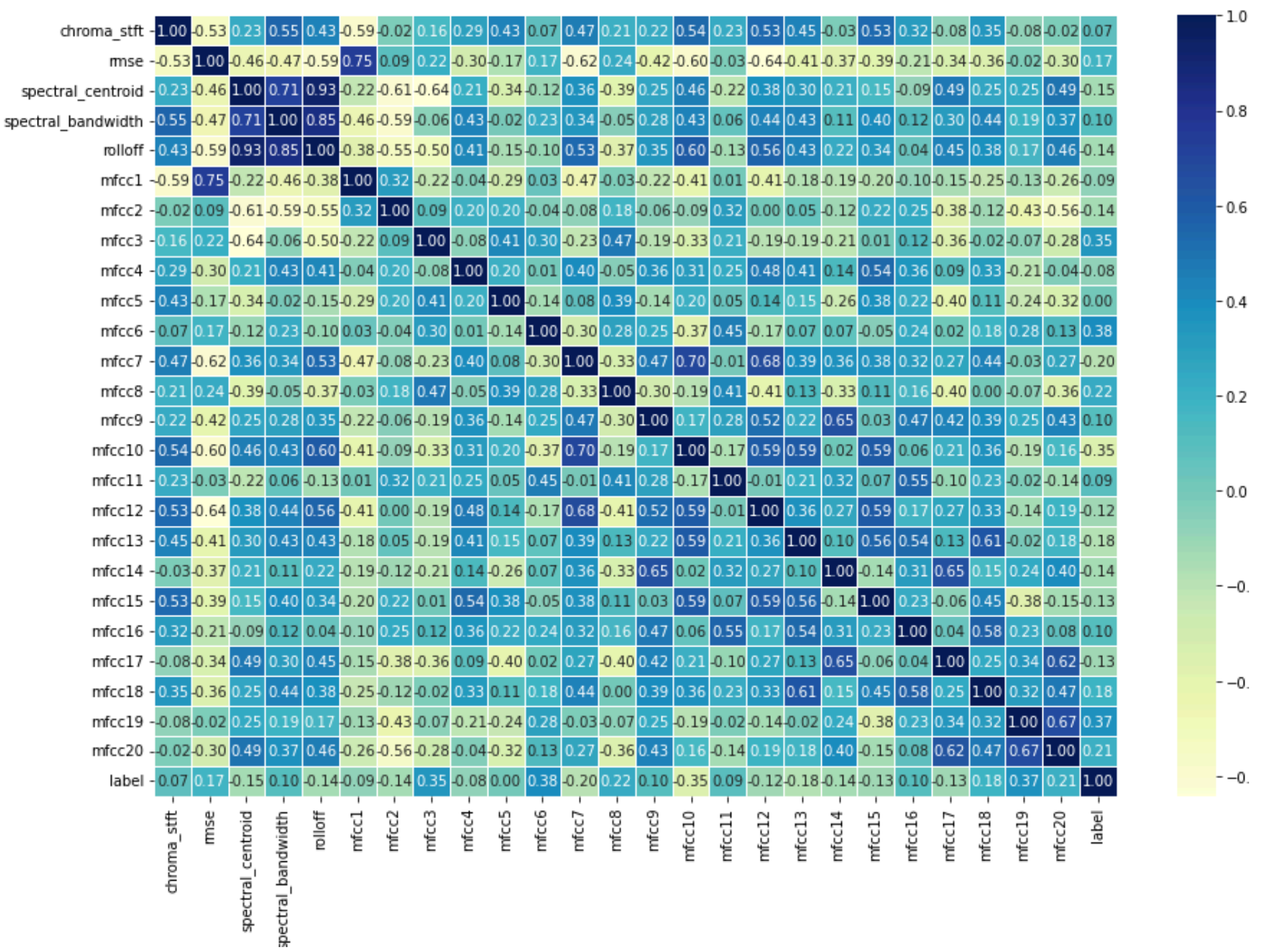
```
# Under-sampling
min_count = df['label'].value_counts().min()
df_balance = df.groupby('label').sample(n=min_count, random_state=42)
print(df_balance['label'].value_counts())
```

Xử lý dữ liệu mất cân bằng

Thực hiện under-sampling để cân bằng số lượng mẫu giữa các nhãn trong tập dữ liệu

Biểu đồ tương quan

Loại bỏ cột filename và dùng .corr() để tính ma trận tương quan và trực quan hóa bằng thư viện plt



Các mô hình học sâu

MLP (Multi-Layer Perceptron)

- Chuẩn hóa dữ liệu và one-hot encoding.
- Xây dựng mạng với 2 lớp ẩn và lớp đầu ra softmax.
- Huấn luyện mô hình với categorical_crossentropy và adam optimizer.

```
# Chuyển đổi y_train, y_test thành one-hot encoding
y_train_onehot = to_categorical(y_train)
y_test_onehot = to_categorical(y_test)

# MLP Model
mlp_model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(y_train_onehot.shape[1], activation='softmax')
])

mlp_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Huấn luyện MLP
mlp_model.fit(X_train_scaled, y_train_onehot, epochs=10, batch_size=32, verbose=0)

# Đánh giá MLP
mlp_train_acc = mlp_model.evaluate(X_train_scaled, y_train_onehot, verbose=0)[1]
mlp_test_acc = mlp_model.evaluate(X_test_scaled, y_test_onehot, verbose=0)[1]
y_pred_mlp = tf.argmax(mlp_model.predict(X_test_scaled), axis=1).numpy()
mlp_f1 = f1_score(y_test, y_pred_mlp, average='macro')

m = 'MLP'
print("Model:", m)
print("Train Accuracy:", mlp_train_acc)
print("Test Accuracy:", mlp_test_acc)
print("F1 Score:", mlp_f1)
model_acc.append([m, mlp_train_acc, mlp_test_acc, mlp_f1])
```

Các mô hình học sâu

CNN (Convolutional Neural Network)

- Lớp tích chập (Conv2D): Kernel (1, 1), 32 filters.
- MaxPooling2D: Kernel (2, 1) để giảm chiều cao.
- Dropout: 30% tại lớp Conv2D, 50% tại lớp Dense.
- Flatten: Chuyển từ 2D sang 1D.
- Fully Connected Layers:
- 128 nút (ReLU).
- Lớp đầu ra (Softmax).

```python
# CNN Model
cnn_model = Sequential([
    Conv2D(32, (1, 1), activation='relu', input_shape=(X_train_scaled.shape[1], 1, 1)),
    MaxPooling2D((2, 1)),  # Giảm chiều cao, giữ nguyên chiều rộng
    Dropout(0.3),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(y_train_onehot.shape[1], activation='softmax')
])

cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Reshape dữ liệu cho CNN
X_train_cnn = X_train_scaled.reshape(X_train_scaled.shape[0], X_train_scaled.shape[1], 1, 1)
X_test_cnn = X                                                    X_test_scaled.shape[1], 1, 1)
# Huấn luyện C
cnn_model.fit(X_train_cnn, y_train_onehot, epochs=10, batch_size=32, verbose=0)
# Đánh giá CNN
cnn_train_acc = cnn_model.evaluate(X_train_cnn, y_train_onehot, verbose=0)[1]
cnn_test_acc = cnn_model.evaluate(X_test_cnn, y_test_onehot, verbose=0)[1]
y_pred_cnn = tf.argmax(cnn_model.predict(X_test_cnn), axis=1).numpy()
cnn_f1 = f1_score(y_test, y_pred_cnn, average='macro')
m = 'CNN'
print("Model:", m)
print("Train Accuracy:", cnn_train_acc)
print("Test Accuracy:", cnn_test_acc)
print("F1 Score:", cnn_f1)
model_acc.append([m, cnn_train_acc, cnn_test_acc, cnn_f1])
```

(variable) X_train_cnn: ndarray[Any, Any]

Các mô hình học sâu

LSTM

-Dữ liệu X_train_scaled và X_test_scaled được reshape thành dạng (samples, timesteps, features) để phù hợp với đầu vào của LSTM.
-LSTM layer: Lớp LSTM với 128 đơn vị (neurons), kích hoạt relu, nhận đầu vào có kích thước (1, số đặc trưng).
-Dense layer:

- Lớp ẩn với 64 đơn vị, kích hoạt relu.
- Lớp đầu ra với số neuron bằng số lớp phân loại, kích hoạt softmax.

```python
#LSTM Model

# Chuyển đổi dữ liệu đầu vào thành dạng 3D cho LSTM (samples, timesteps, features)
# Ví dụ, giả sử mỗi mẫu có 1000 bước thời gian (timesteps) và 13 đặc trưng (features)
X_train_scaled = X_train_scaled.reshape(X_train_scaled.shape[0], 1, X_train_scaled.shape[1])
X_test_scaled = X_test_scaled.reshape(X_test_scaled.shape[0], 1, X_test_scaled.shape[1])
# Xây dựng mô hình LSTM
model = Sequential()
# LSTM layer
model.add(LSTM(units=128, activation='relu', input_shape=(X_train_scaled.shape[1], X_train_scaled.shape[2]), return_sequences=False))
# Dropout để tránh overfitting
model.add(Dropout(0.2))
# Lớp Dense đầu ra
model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=y_train_onehot.shape[1], activation='softmax'))  # Số lớp output = số lớp phân loại
# Biên dịch mô hình
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Huấn luyện mô hình
history = model.fit(X_train_scaled, y_train_onehot, epochs=10, batch_size=32, validation_data=(X_test_scaled, y_test_onehot))
# Đánh giá mô hình
train_acc = history.history['accuracy'][-1]
test_acc = history.history['val_accuracy'][-1]
# Tính F1 score
y_pred = np.argmax(model.predict(X_test_scaled), axis=-1)
y_true = np.argmax(y_test_onehot, axis=-1)
f1 = f1_score(y_true, y_pred, average='weighted')
# In kết quả
m = 'LSTM'
print("Model:", m)
print("Train Accuracy:", train_acc)
print("Test Accuracy:", test_acc)
print("F1 Score:", f1)
# Lưu các kết quả vào danh sách
model_acc.append([m, train_acc, test_acc, f1])
```

Kết quả

	Model	Train_Acc	Test_Acc	F1 score
0	MLP	0.959058	0.953623	0.953467
1	CNN	0.912681	0.895652	0.895202
2	LSTM	0.972101	0.972464	0.972427

1. MLP:
   - Độ chính xác huấn luyện cao (95.91%) và độ chính xác kiểm tra tương đối cao (95.36%).
   - F1 Score khá tốt (95.35%), cho thấy mô hình có khả năng phân loại chính xác.
2. CNN:
   - Độ chính xác huấn luyện (91.27%) và kiểm tra (89.57%) thấp hơn MLP.
   - F1 Score (89.52%) cho thấy mô hình hoạt động ổn nhưng không bằng MLP.
3. LSTM:
   - Độ chính xác huấn luyện và kiểm tra đều rất cao (97.21% và 97.25%).
   - F1 Score đạt 97.24%, cho thấy đây là mô hình mạnh mẽ nhất trong việc phân loại âm thanh.

Kết luận:

- LSTM là mô hình tốt nhất trong ba mô hình, với độ chính xác và F1 Score cao nhất trên cả dữ liệu huấn luyện và kiểm tra.
- MLP cũng cho kết quả tốt, nhưng kém hơn so với LSTM.
- CNN có hiệu suất thấp nhất trong ba mô hình, nhưng vẫn hoạt động khá ổn.

Sử dụng

Hàm thu thập âm thanh trực tiếp

```python
import sounddevice as sd
from scipy.io.wavfile import write

def record_audio(output_file, duration=5, sampling_rate=44100):
    """
    Ghi âm trực tiếp từ micro và lưu vào file WAV.
    - output_file: Tên file WAV lưu.
    - duration: Thời gian ghi âm (giây).
    - sampling_rate: Tần số mẫu (Hz).
    """
    print("Bắt đầu ghi âm... Nói đi nào!")
    audio_data = sd.rec(int(duration * sampling_rate), samplerate=sampling_rate, channels=1, dtype='int16')
    sd.wait()  # Đợi ghi âm xong
    write(output_file, sampling_rate, audio_data)  # Lưu file WAV
    print(f"Ghi âm xong! File được lưu tại: {output_file}")

# Gọi hàm để ghi âm
record_audio(r"D:\HK7\DL\CK\data_test\giong_noi.wav", duration=5)  # Ghi âm 5 giây
```

Sử dụng

Hàm xử lý file âm thanh vừa lưu và tiến hành dự đoán kết quả.

```python
import librosa
import numpy as np

# Hàm tiền xử lý file âm thanh
def preprocess_audio(file_path, scaler):
    """
    Tiền xử lý âm thanh:
    - Load âm thanh.
    - Trích xuất đặc trưng MFCC.
    - Chuẩn hóa đặc trưng.
    """
    y, sr = librosa.load(file_path, sr=None, duration=10)  # Load âm thanh (10 giây)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=20)  # Trích xuất MFCC (20 coefficients)
    mfcc_mean = np.mean(mfcc.T, axis=0)  # Lấy trung bình của MFCC
    scaled_mfcc = scaler.transform([mfcc_mean])  # Chuẩn hóa đặc trưng
    return scaled_mfcc

# Hàm dự đoán miền
def predict_region(file_path, model, scaler, label_mapping):
    """
    Dự đoán miền của giọng nói từ file âm thanh:
    - `file_path`: Đường dẫn tới file âm thanh.
    - `model`: Mô hình MLP đã huấn luyện.
    - `scaler`: Bộ chuẩn hóa đã sử dụng trong quá trình huấn luyện.
    - `label_mapping`: Mapping từ nhãn số sang tên miền.
    """
    # Tiền xử lý file âm thanh
    processed_audio = preprocess_audio(file_path, scaler)

    # Dự đoán
    prediction = model.predict(processed_audio)
    predicted_label = np.argmax(prediction, axis=1)[0]  # Lấy nhãn dự đoán
    region = label_mapping[predicted_label]  # Chuyển nhãn sang tên miền
    return region

# Mapping nhãn sang tên miền
label_mapping = {
    1: "Miền Bắc",
    2: "Miền Trung",
    3: "Miền Nam"
}

# Ví dụ sử dụng
file_path =r"D:\HK7\DL\CK\data_test\mien_nam_giong_noi.wav" # Đường dẫn tới file âm thanh
predicted_region = predict_region(file_path, mlp_model, scaler, label_mapping)
print(f"Giọng nói này thuộc miền: {predicted_region}")
```

THANK YOU FOR WATCHING!