HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐỒ ÁN TỐT NGHIỆP

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Trường Điện – Điện tử

Khoa Tự động hoá

NGHIÊN CỨU ỨNG DỤNG MẠNG NƠ-RON LSTM TRONG NHẬN DẠNG CÂU LỆNH ĐIỀU KHIỂN ĐIỀU HOÀ BẰNG TIẾNG NÓI

Lê Đình Khánh
Khanh.LD191905@sis.hust.edu.vn
Giảng viên hướng dẫn: TS. Trần Thị Anh Xuân

Nội Dung

1. Tổng quan đề tài
2. Cơ sở lý thuyết về nhận dạng tiếng nói
3. Xây dựng cơ sở dữ liệu
4. Xây dựng mô hình
5. Thử nghiệm và đánh giá kết quả
6. Kết luận và hướng phát triển tiếp theo

Nội dung

- **Tổng quan về đề tài**
  - **Xử lý tiếng nói và ứng dụng**
  - **Các thách thức**
  - **Mục tiêu và phạm vi đề tài**
- Cơ sở lý thuyết về nhận dạng tiếng nói
- Xây dựng cơ sở dữ liệu
- Xây dựng mô hình
- Thử nghiệm và đánh giá kết quả
- Kết luận và hướng phát triển tiếp theo

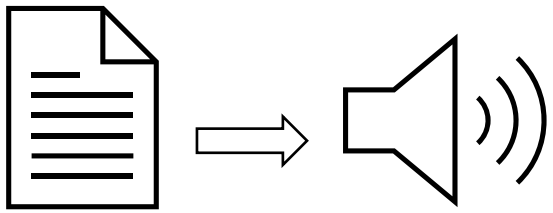
1. Tổng quan đề tài

❑ **Xử lý tiếng nói và ứng dụng**

Sự phát triển mạnh mẽ của Học Sâu, Internet vạn vật (IoT) cùng với xu hướng điều khiển thiết bị không chạm.

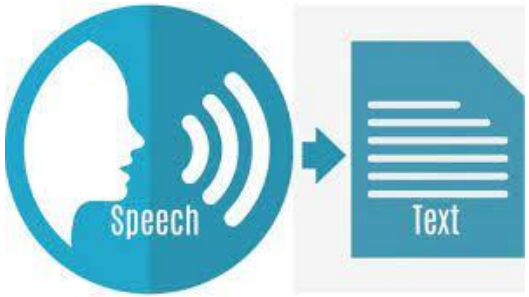
➢ Lĩnh vực xử lý tiếng nói đã có những bước tiến lớn, theo hai hướng:

Tổng hợp tiếng nói
(Text to Speech)



Ứng dụng: - Trợ lý tiếng nói.
- Sách nói.

Nhận dạng tiếng nói
(Speech to Text)



Ứng dụng: - Phiên dịch.
- Điều khiển các thiết bị bằng tiếng nói.

1. Tổng quan đề tài

❑ **Các thách thức**

- Sự đa dạng vùng miền ở Việt Nam.
- Thiếu cơ sở dữ liệu.
- Tác động từ nhiễu.

❑ **Mục tiêu và phạm vi đề tài**

- Xây dựng mô hình ứng dụng mạng nơ-ron LSTM nhận dạng một số câu lệnh thông dụng để điều khiển điều hoà.
- Môi trường sạch, ít nhiễu, giọng miền Trung.
- Chạy thử nghiệm trên máy tính để đánh giá chất lượng của mô hình đề xuất.

Nội dung

- Tổng quan về đề tài
- **Cơ sở lý thuyết về nhận dạng tiếng nói**
  - **Nguyên lý hình thành tiếng nói**
  - **Trích xuất đặc trưng MFCC**
- Xây dựng cơ sở dữ liệu
- Xây dựng mô hình
- Thử nghiệm và đánh giá kết quả
- Kết luận và hướng phát triển tiếp theo

2. Cơ sở lý thuyết về nhận dạng tiếng nói

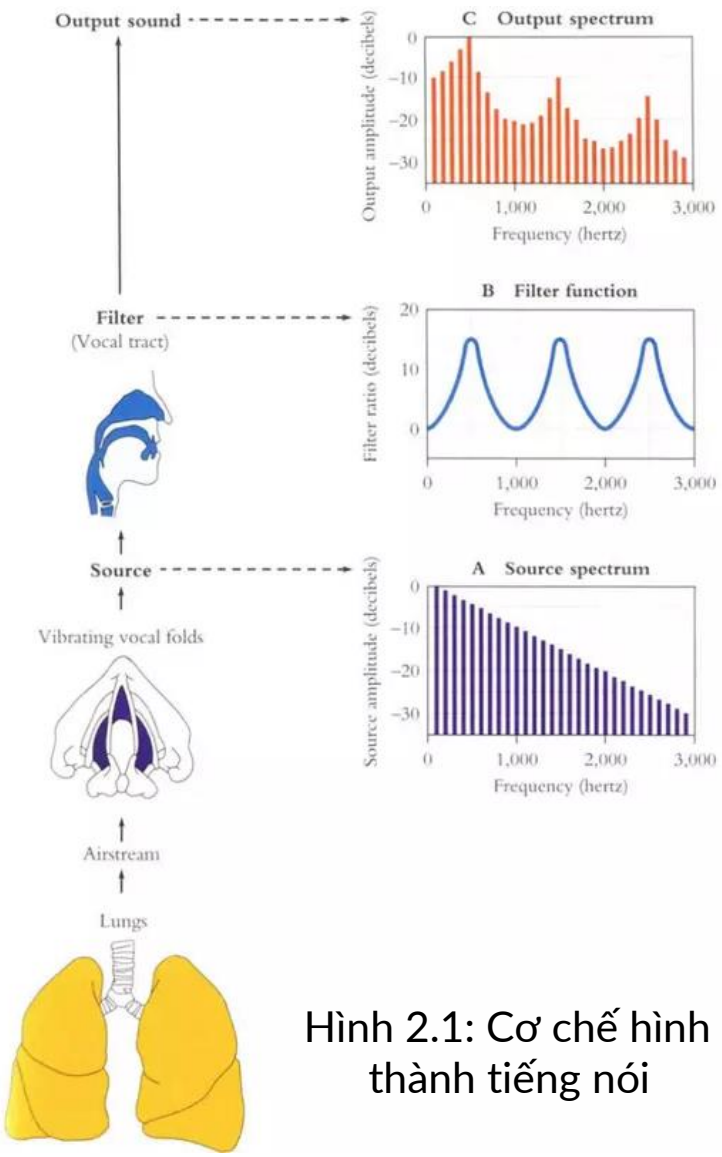
2.1 Nguyên lý hình thành tiếng nói

❑ Nguyên lý

Với đầu vào là luồng không khí từ phổi, qua thanh quản sẽ tạo ra các tần số sóng âm, sau đó các cơ quan có chức năng như bộ lọc tạo thành đầu ra là âm thanh tiếng nói.

❑ Một số đặc điểm

- Tần số cơ bản F0.
- Các "formant" F1, F2, F3 …
- Cơ chế hoạt động của tai: tai người rất nhạy cảm ở âm thanh tần số thấp, kém nhạy cảm ở tần số cao.



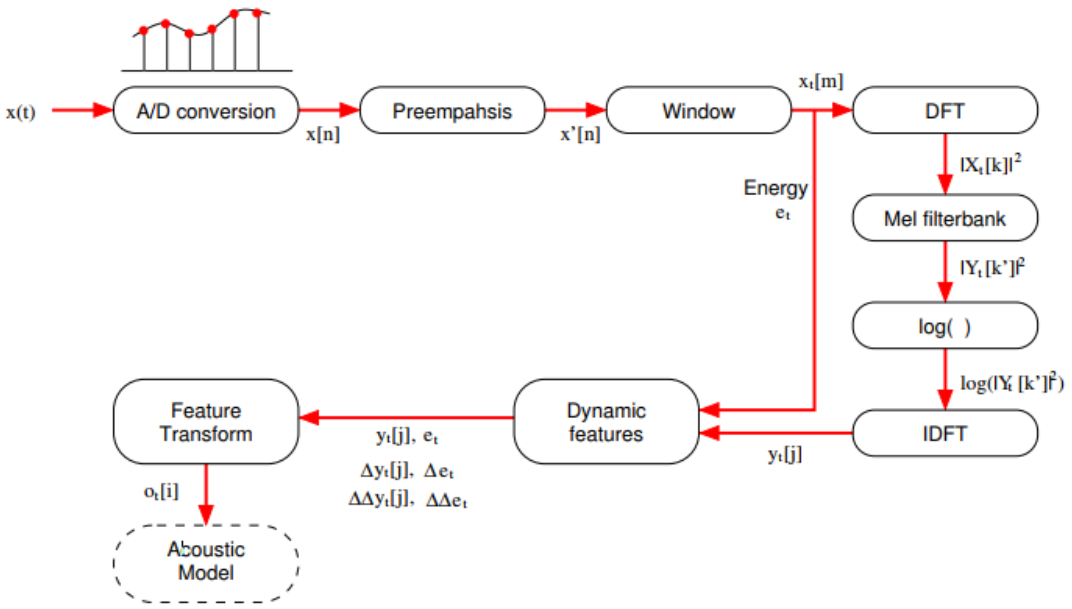
Hình 2.1: Cơ chế hình thành tiếng nói

2.2 Trích xuất đặc trưng MFCC

❑ Luồng xử lý:

- Cắt chuỗi tín hiệu âm thanh thành các đoạn ngắn bằng nhau (25ms) và xếp chồng lên nhau (10ms).
- Mỗi đoạn âm thanh được biến đổi, tính toán để thu được 39 đặc trưng MFCCs, với các tính chất:

• Tính độc lập cao, ít nhiễu.
• Đủ nhỏ để đảm bảo tính toán, đủ thông tin để đảm bảo chất lượng cho các thuật toán nhận dạng.



Hình 2.2: Luồng xử lý âm thanh tạo MFCC

Nội dung

- Tổng quan về đề tài
- Cơ sở lý thuyết về nhận dạng tiếng nói
- **Xây dựng cơ sở dữ liệu**
  - **Cơ sở dữ liệu đề xuất**
  - **Tăng cường dữ liệu âm thanh**
  - **Trích xuất đặc trưng MFCC bằng thư viện Librosa**
- Xây dựng mô hình
- Thử nghiệm và đánh giá kết quả
- Kết luận và hướng phát triển tiếp theo

3. Xây dựng cơ sở dữ liệu

3.1 Cơ sở dữ liệu đề xuất

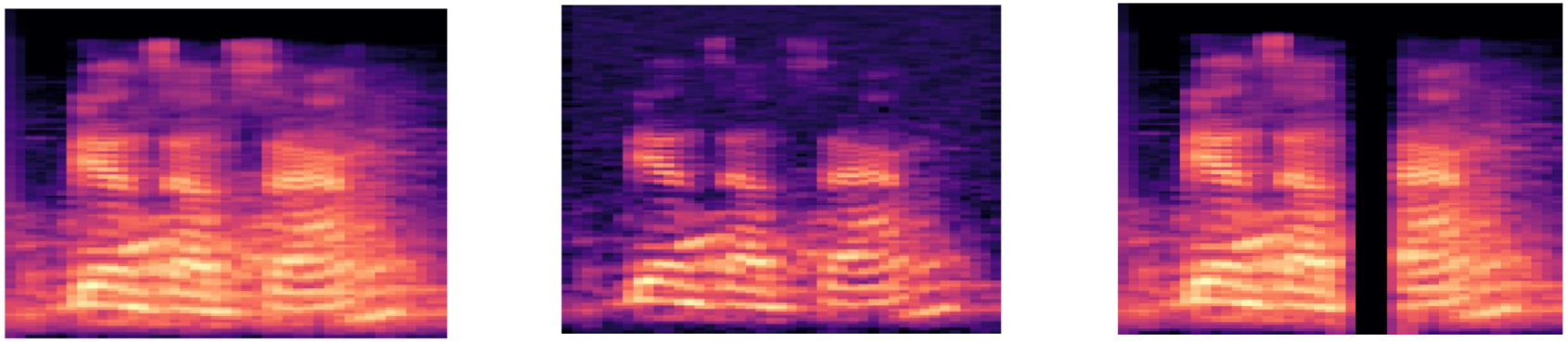
❑ Cơ sở dữ liệu âm thanh
- Câu kích hoạt: Wiki ơi.
- Các câu lệnh: Bật điều hoà, tắt điều hoà, tăng 1 độ, giảm 1 độ, bật 26 độ.
- Nhiễu.

❑ Tập dữ liệu trên được thu:
- 60 người (40 nam, 20 nữ), trong độ tuổi từ 20 – 30.
- Đều là người Hà Tĩnh, Nghệ An.

❑ Thông số:
- 5 mẫu/câu/người.
- Thu: phần mềm Audacity.
- Micrô: micrô trực tiếp của máy tính.
- Độ dài:
+ Câu kích hoạt: 1s.
+ Câu lệnh: 1.5s.
- Chế độ âm thanh: mono (1 kênh).
- Tần số lấy mẫu: 16000 Hz.
- Độ sâu bit: 32-bit float.

3.2 Tăng cường dữ liệu âm thanh

- Thư viện 'Audiomentations': Tạo ra tập dữ liệu mới có số lượng mẫu gấp 12 lần tập dữ liệu ban đầu (khoảng 3600 mẫu/câu).



a) Âm thanh gốc b) Noise injection c) Time masking

Hình 3.1 Spectrogram sau một số hiệu ứng tăng cường

3.3 Trích xuất đặc trưng MFCC bằng thư viện Librosa

- Đầu ra: ma trận có kích thước (x,39).

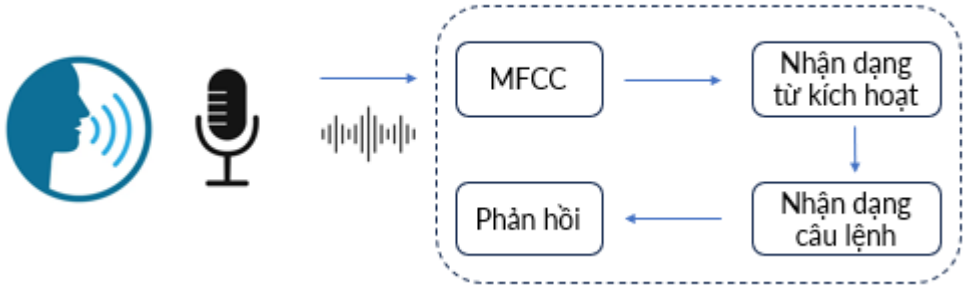
- Mô hình nhận dạng từ kích hoạt với tệp âm thanh dài 1s: x = 66.
- Mô hình nhận dạng câu lệnh với tệp âm thanh dài 1.5s : x = 100.

Nội dung

- Tổng quan về đề tài
- Cơ sở lý thuyết về nhận dạng tiếng nói
- Xây dựng cơ sở dữ liệu
- **Xây dựng mô hình**
  - **Yêu cầu đối với các mô hình**
  - **Mô hình nhận dạng từ kích hoạt đề xuất**
  - **Mô hình nhận dạng câu lệnh đề xuất**
- Thử nghiệm và đánh giá kết quả
- Kết luận và hướng phát triển tiếp theo

4. Xây dựng mô hình

4.1 Yêu cầu đối với các mô hình



Hình 4.1 Cách hoạt động của các mô hình

Yêu cầu	Nhận dạng từ kích hoạt	Nhận dạng câu lệnh
Khả năng biểu diễn	- Ít phức tạp - Đầu ra: phân loại 2 lớp	- Phức tạp - Đầu ra: phân loại nhiều lớp
Kích thước mô hình	- Nhỏ - Yêu cầu ít tài nguyên	- Lớn - Yêu cầu nhiều tài nguyên
Thời gian chạy	- Liên tục	- Không liên tục
Mạng nơ-ron đề xuất	- CNN	- LSTM

Bảng 4.1 Bảng các yêu cầu của hai mô hình

4. Xây dựng mô hình

4.2 Mô hình nhận dạng từ kích hoạt đề xuất

❑ Mạng nơ-ron tích chập CNN:

- Sử dụng các lớp tích chập để trích xuất các đặc trưng từ dữ liệu đầu vào.
- Lớp gộp: giảm kích thước của dữ liệu và giảm độ phức tạp của mô hình.
- Học đặc trưng: các bộ lọc khác nhau trích xuất các đặc trưng quan trọng khác nhau từ dữ liệu đầu vào, giúp cải thiện hiệu suất của mô hình.

❑ Tổng tham số: 132370

```
def create_cnn_model(input_shape):
    model = models.Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation='relu',
                            input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(2, activation='sigmoid'))
    return model
```

Hình 4.2 Mô hình nhận dạng từ kích hoạt đề xuất

4. Xây dựng mô hình

4.3 Mô hình nhận dạng câu lệnh đề xuất

- ❑ Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM):

- Xử lý dữ liệu tuần tự.
- Lưu trữ thông tin ngắn hạn và dài hạn.
- Xử lý chuỗi dài.
- Khả năng học đa tầng.

➢ Phù hợp với xử lý tiếng nói: dữ liệu có tính tuần tự, xử lý chuỗi dài mà không bị ảnh hưởng bởi vấn đề "biến mất gradient".

- ❑ Tổng tham số: 508870

```
def get_sequence_model():
    frame_features = Input(data["x_train"].shape[1:])
    x = LSTM(256, return_sequences=True)(frame_features)
    x = keras.layers.LSTM(128)(x)
    x = keras.layers.Dropout(0.2)(x)
    x = keras.layers.Dense(64, activation="relu")(x)
    x = keras.layers.Dropout(0.2)(x)
    output = keras.layers.Dense(data["y_train"].shape[1],
                                activation="softmax")(x)
    LSTM_model = keras.Model(frame_features, output)
    return LSTM_model
```

Hình 4.3 Mô hình nhận dạng câu lệnh đề xuất

Nội dung

- Tổng quan về đề tài
- Cơ sở lý thuyết về nhận dạng tiếng nói
- Xây dựng cơ sở dữ liệu
- Xây dựng mô hình
- **Thử nghiệm và đánh giá kết quả**
  - **Phân chia cơ sở dữ liệu**
  - **Kết quả mô hình nhận dạng từ kích hoạt**
  - **Kết quả mô hình nhận dạng câu lệnh**
  - **Thực nghiệm**
- Kết luận và hướng phát triển tiếp theo

5. Thử nghiệm và đánh giá kết quả

5.1 Phân chia cơ sở dữ liệu

❑ Cơ sở dữ liệu

- Tập dữ liệu gốc: khoảng 300 mẫu/câu.
- Tập dữ liệu tăng cường: khoảng 3600 mẫu/câu.
- Nhiễu: số lượng mẫu gấp khoảng 2.5 lần số lượng mẫu của một câu: khoảng 9000 mẫu.

❑ Phân chia ra thành 3 tập dữ liệu: Huấn luyện / Đánh giá / Kiểm tra với tỷ lệ tương ứng: 72/18/10.

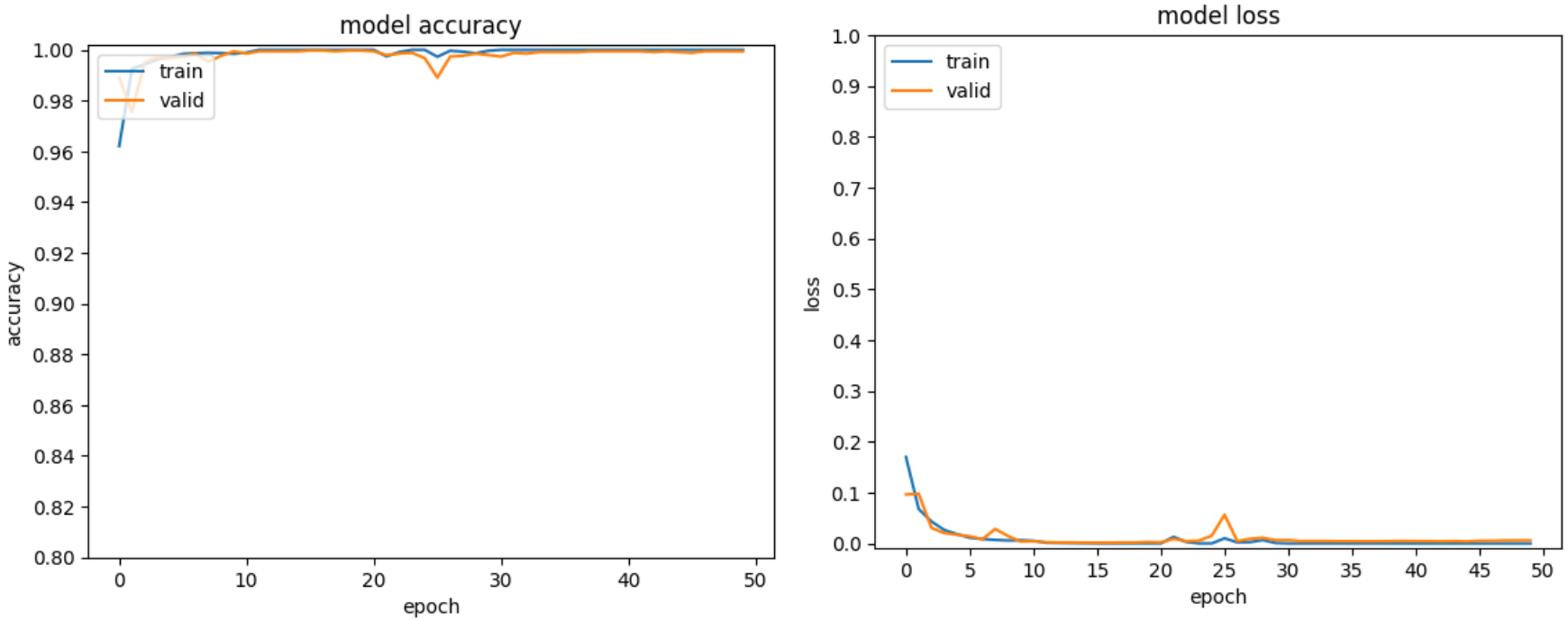
- Tập kiểm tra: các câu được thu từ giọng nói của 6 người, không có trong tập huấn luyện và đánh giá.

❑ Các thông số, tập dữ liệu áp dụng giống nhau với tất cả các quá trình để so sánh kết quả thử nghiệm khách quan nhất.

5. Thử nghiệm và đánh giá kết quả

5.2 Kết quả mô hình nhận dạng từ kích hoạt

- 3387 mẫu tập đánh giá: hàm mất mát: 0.0014; độ chính xác: 99.97%.



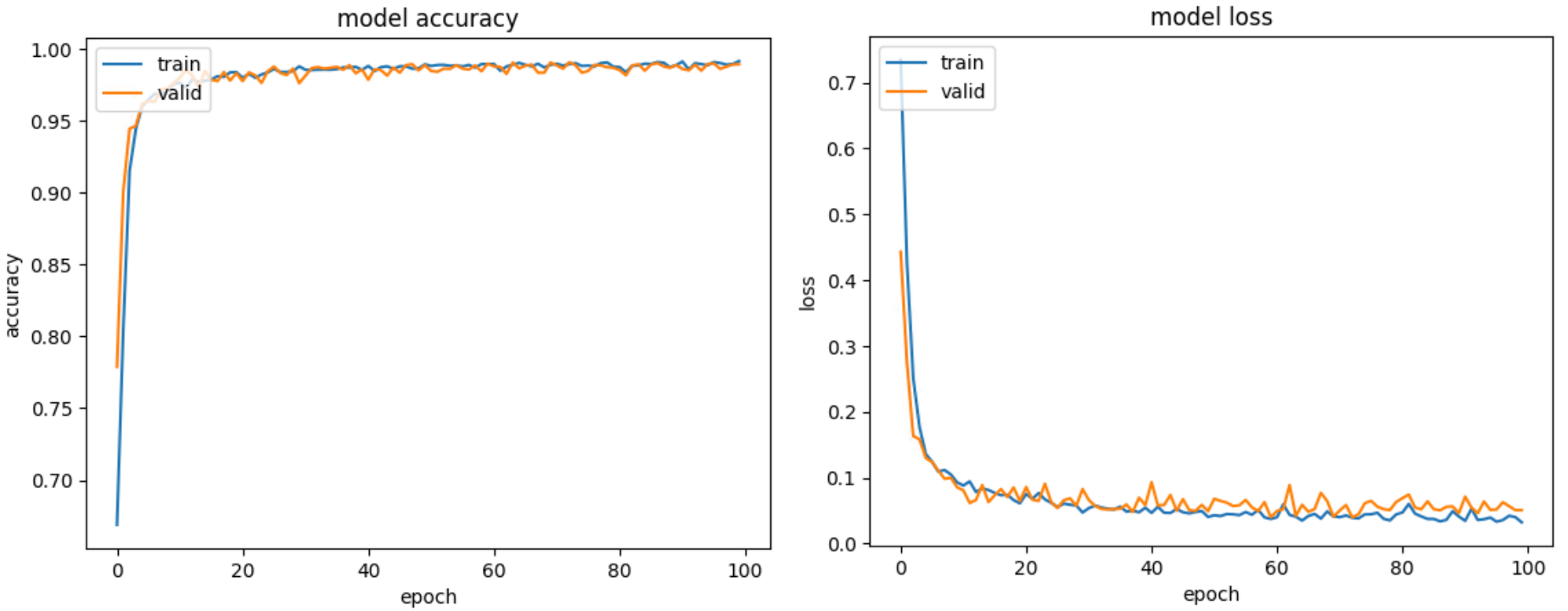
Hình 5.1 Hàm mất mát, độ chính xác của mô hình nhận dạng từ kích hoạt

- 1876 mẫu tập kiểm tra: hàm mất mát: 0.0257; độ chính xác: 99.47%.

5. Thử nghiệm và đánh giá kết quả

5.3 Kết quả mô hình nhận dạng câu lệnh

- 4946 mẫu tập đánh giá: hàm mất mát: 0.0398; độ chính xác: 99.05%.



Hình 5.2 Hàm mất mát, độ chính xác của mô hình nhận dạng câu lệnh

- 2818 mẫu tập kiểm tra: hàm mất mát: 0.1608; độ chính xác: 97.30%.

5. Thử nghiệm và đánh giá kết quả

5.4 Thực nghiệm

❑ Trên bộ dữ liệu thu sẵn

Trên cùng tập dữ liệu tăng cường, tập kiểm tra nhiều nhiễu: mô hình đề xuất cho kết quả tốt nhất.

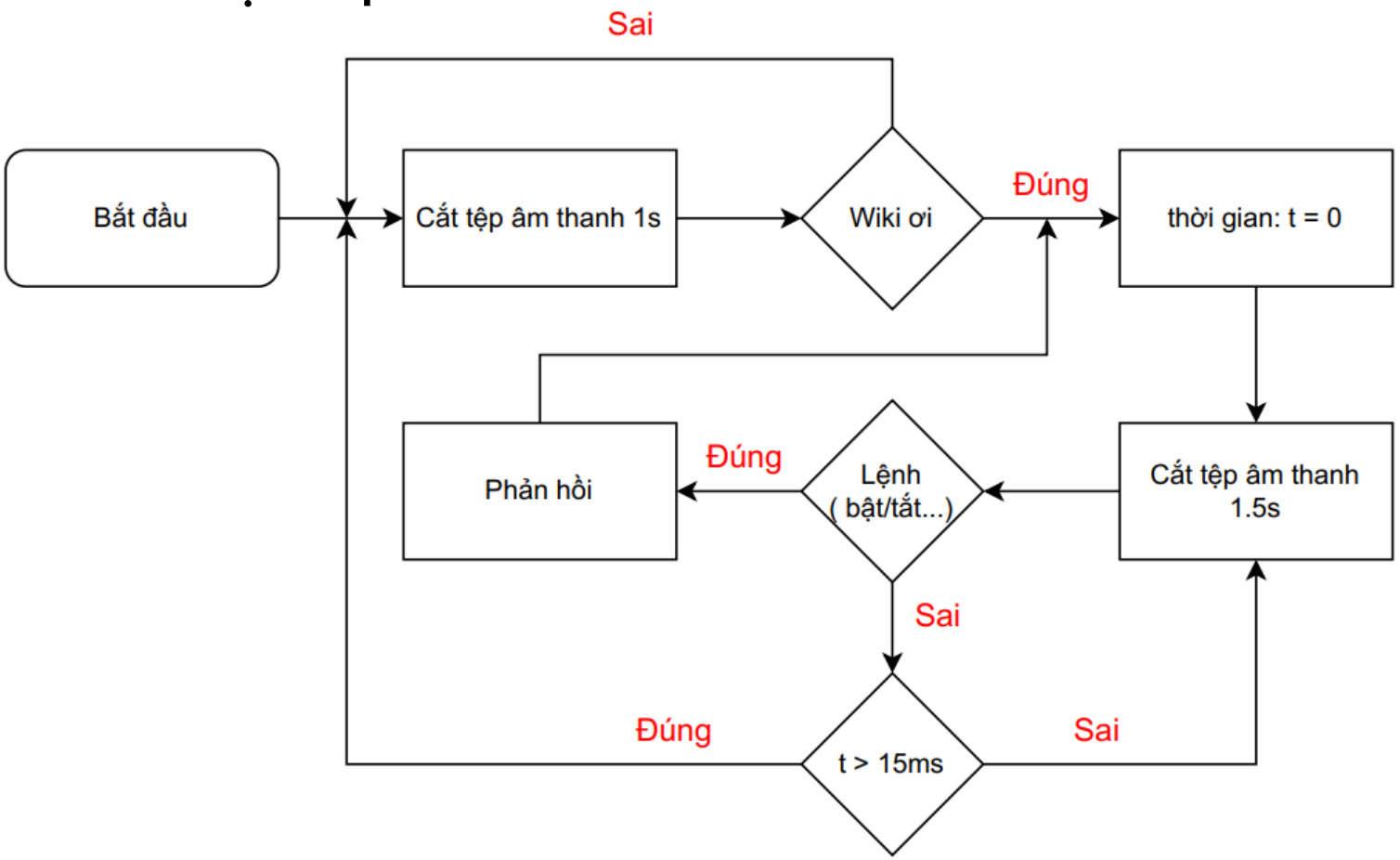
Mô hình	Tổng tham số	Tập đánh giá		Tập kiểm tra	
		Độ chính xác (%)	Hàm mất mát	Độ chính xác (%)	Hàm mất mát
CSVC-Net*	1157006	99.11	0.0517	96.10	0.2591
LSTM**	689414	98.79	0.0646	94.85	0.2574
Mô hình đề xuất	508870	99.05	0.0398	97.30	0.1608

Bảng 5.1 Bảng so sánh với các mô hình nghiên cứu

* : mô hình LSTM của tác giả K. Banuroopaa.
**: mô hình CSVC-Net của tác giả của Arowa Yasmeen.

5.4 Thực nghiệm

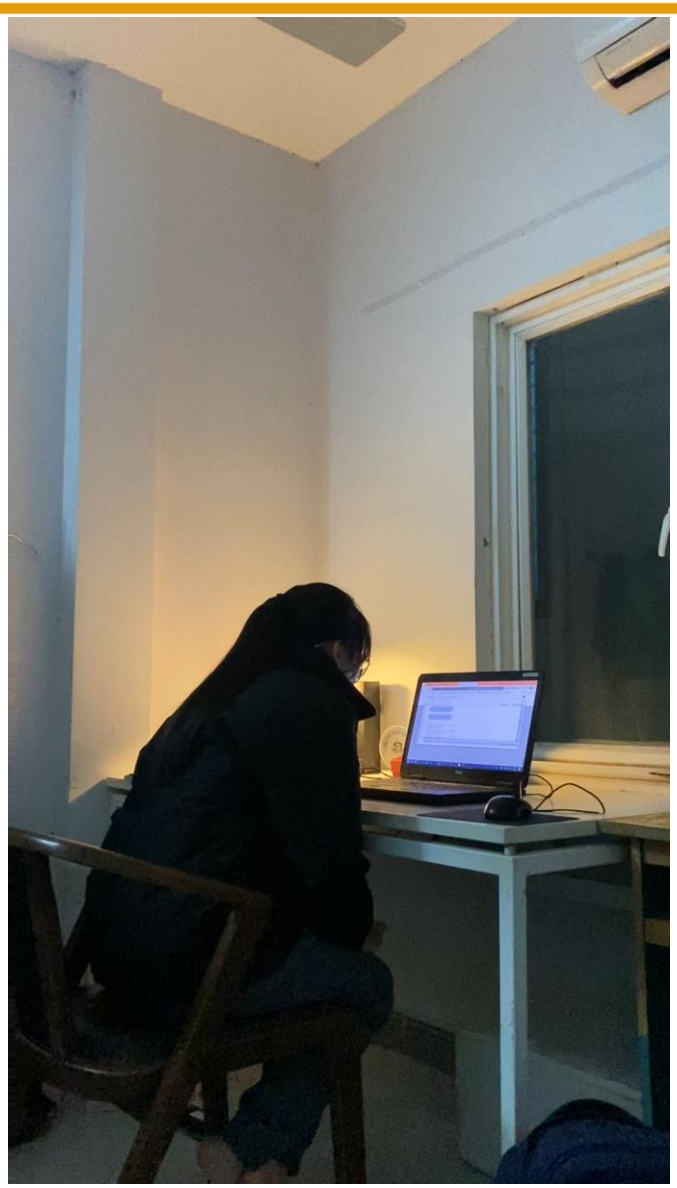
- Trên micrô trực tiếp.



Hình 5.3 Lưu đồ thuật toán

5. Thử nghiệm và đánh giá kết quả

5.5 Thực nghiệm



6. Kết luận và hướng phát triển tiếp theo

❑ Kết luận

Mô hình đề xuất đã đáp ứng được cơ bản mục tiêu đề tài:

- Nghiên cứu ứng dụng mạng nơ-ron LSTM trong nhận dạng câu lệnh điều khiển điều hoà bằng tiếng nói.
- Quá trình thử nghiệm cho thấy:
  - Mô hình có thể hoạt động được trong môi trường thực tế của căn phòng.
  - Tuy nhiên, với môi trường nhiễu và giọng nói đa dạng, vẫn có trường hợp mô hình nhận dạng sai hoặc chưa nhận dạng được. Đây là một trong những yếu tố cần cải thiện của mô hình.

❑ Một số hướng phát triển:

- Đã thu dữ liệu của các câu lệnh để tăng tính ứng dụng của mô hình: tăng/giảm 2 độ, bật 20-32 độ, bật chế độ tự động/sưởi ấm/làm mát.
- Ghi âm thêm câu tạm biệt để thoát khỏi mô hình nhận dạng câu lệnh.
- Phát triển mô hình thành một sản phẩm hoàn thiện.
- Nghiên cứu ứng dụng thêm các loại mạng nơ-ron để nhận dạng chính xác hơn trong mô hình.

Tài liệu tham khảo

[1] T. T. Nguyễn, "Viblo," [Online]. Available: https://viblo.asia/p/kien-thuc-nen-tang-xu-ly-tieng-noi-speech-processing-jvElaAL6lkw.
[2] H. S. a. S. Renals, Speech Signal Analysis, 2015.
[3] Weekly Study Corp, "Weekly Study," [Online]. Available: https://www.weeklystudy.asia/2021/10/gioi-thieu-ve-hoc-sau-deep-learning-su-lien-quan-giua-dl-va-ml.html.
[4] C. P. Van, "Viblo," [Online]. Available: https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2.
[5] Colah, "Colah," [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[6] D. S. P. K. Banuroopaa, "MFCC based hybrid fingerprinting method for audio classification through LSTM," The International Journal of Nonlinear Analysis and Applications (IJNAA), 2022.
[7] F. I. R. S. A. M. H. K. Arowa Yasmeen, "CSVC-Net: Code-Switched Voice Command Classification using Deep CNN-LSTM Network," Conference Paper, 2021.

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !

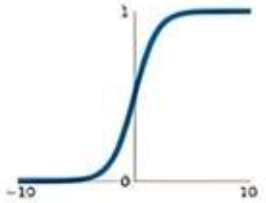
Hàm kích hoạt

Softmax

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

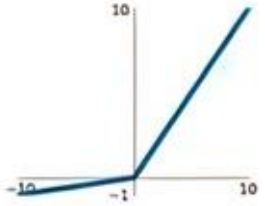
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



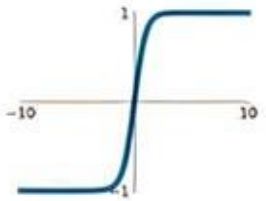
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

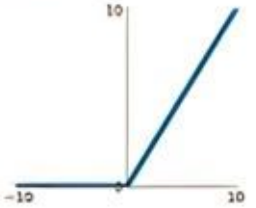


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

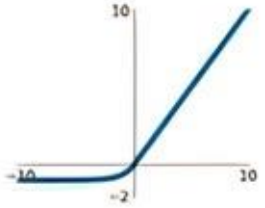
ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Phụ lục

Tăng cường dữ liệu

```
def get_augs(aug):
    augmentations = []
    if aug == 'NoiseInjection':
        augmentations.append(AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.0014, p=1.0))
    elif aug == 'PitchShifting':
        augmentations.append(PitchShift(min_semitones=-2.5, max_semitones=2.5, p=1.0))
    elif aug == 'TimeStretching':
        augmentations.append(TimeStretch(min_rate=1.04, max_rate=1.05, p=1.0))
    elif aug == 'Padding':
        augmentations.append(Padding(min_fraction=0.04,max_fraction=0.05, p=1.0))
    elif aug == 'BandPassFilter':
        augmentations.append(BandPassFilter(min_center_freq=2200, max_center_freq=2400, p=1.0))
    elif aug == 'Gain':
      augmentations.append(Gain(min_gain_in_db=-10, max_gain_in_db=10,p=1.0))
    elif aug == 'TimeMask':
      augmentations.append(TimeMask(min_band_part=0.14, max_band_part=0.16,p=1.0))
    elif aug == 'HighPassFilter':
      augmentations.append(HighPassFilter(min_cutoff_freq=1600, max_cutoff_freq=1800, p=1))
    elif aug == 'ClippingDistortion':
      augmentations.append(ClippingDistortion(min_percentile_threshold =0, max_percentile_threshold=8, p=1))
    elif aug == 'AirAbsorption':
      augmentations.append(AirAbsorption(min_distance=10.0,max_distance=40.0,p=1.0,))
    elif aug == "Trim":
      augmentations.append(Trim(top_db=30.0,p=1.0))
    return augmentations
```

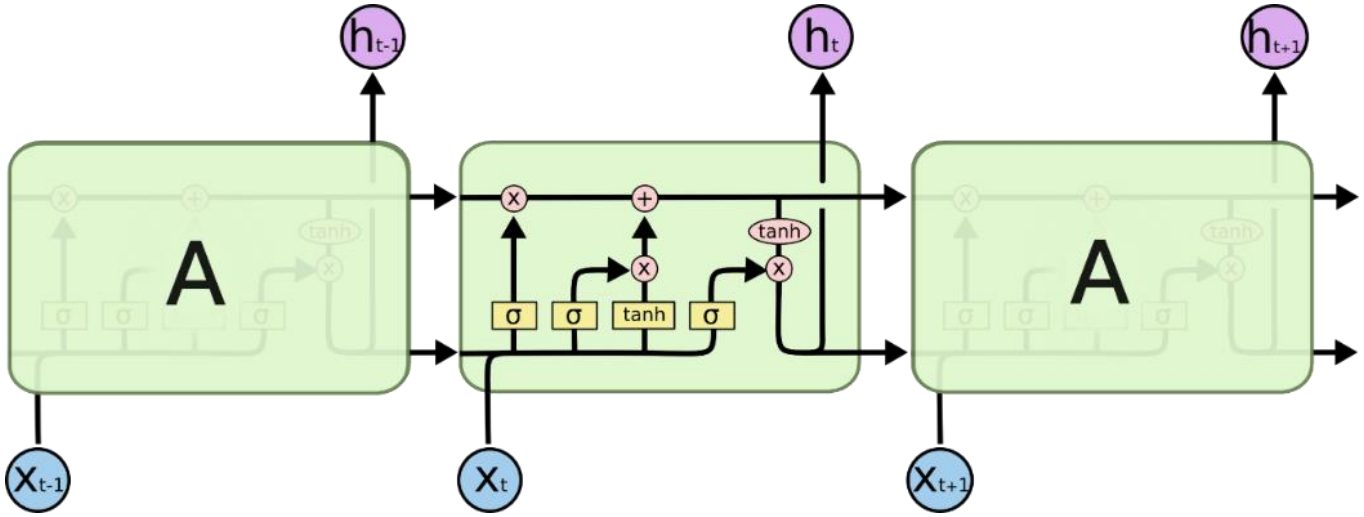
Hàm trích xuất đặc trưng MFCC

```python
def features_extractor(file_name):
    audio,sample_rate  = librosa.load(file_name, sr=16000, mono=True)
    frame_length = int(0.025 * sample_rate)
    hop_length = int(0.015 * sample_rate)
    n_fft = 2 ** int(np.ceil(np.log2(frame_length)))
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_fft=n_fft, hop_length=hop_length, n_mfcc=13)
    desired_size = 100
    mfccs = np.zeros((13,desired_size))
    mfccs[:, :min(desired_size, mfcc.shape[1])] = mfcc[:, :min(desired_size, mfcc.shape[1])]
    delta_mfccs = librosa.feature.delta(mfccs)
    delta2_mfccs = librosa.feature.delta(mfccs, order=2)
    mfccs_features = np.concatenate((mfccs, delta_mfccs, delta2_mfccs))
    mfccs_features = np.transpose(mfccs_features)
    return mfccs_features
```

Phụ lục

LSTM

- Cấu trúc ô nhớ: sử dụng một cấu trúc ô nhớ để lưu trữ thông tin theo thời gian. Thay vì chỉ dựa vào trạng thái ẩn như các mô hình RNN thông thường, LSTM có khả năng duy trì thông tin trong một khoảng thời gian dài, giúp mô hình tránh được vấn đề biến mất gradient khi cần phải trải qua nhiều bước thời gian.
- Cổng quên và cổng cập nhật.



Phụ lục

Thông số huấn luyện

- Binary Crossentropy Loss: $L(\hat{y}, y) = -(y log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$

- Categorical Crossentropy Loss: $L(\hat{y}, y) = -\sum_{i=0}^{N-1} y_i * \log(\hat{y}_i)$

- Accuracy = $\frac{\text{Số mẫu dự đoán đúng}}{\text{Tổng số lượng mẫu}}$

- Learning rate: 0.001

Các bước mô hình nhận dạng từ kích hoạt

```
def create_cnn_model(input_shape):
    model = models.Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation='relu',
                            input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(2, activation='sigmoid'))
    return model
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 37, 16)        160

 max_pooling2d (MaxPooling2  (None, 32, 18, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 30, 16, 64)        9280

 max_pooling2d_1 (MaxPoolin  (None, 15, 8, 64)         0
 g2D)

 flatten (Flatten)           (None, 7680)              0

 dense (Dense)               (None, 16)                122896

 dense_1 (Dense)             (None, 2)                 34

=================================================================
Total params: 132370 (517.07 KB)
Trainable params: 132370 (517.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Phụ lục

Các lớp mô hình nhận dạng câu lệnh

```python
def get_sequence_model():
    frame_features = Input(data["x_train"].shape[1:])
    x = LSTM(256, return_sequences=True)(frame_features)
    x = keras.layers.LSTM(128)(x)
    x = keras.layers.Dropout(0.2)(x)
    x = keras.layers.Dense(64, activation="relu")(x)
    x = keras.layers.Dropout(0.2)(x)
    output = keras.layers.Dense(data["y_train"].shape[1],
                                activation="softmax")(x)
    LSTM_model = keras.Model(frame_features, output)
    return LSTM_model
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 100, 39)]         0

 lstm (LSTM)                 (None, 100, 256)          303104

 lstm_1 (LSTM)               (None, 128)               197120

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 508870 (1.94 MB)
Trainable params: 508870 (1.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```