TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO

PROJECT III

ĐỀ TÀI: Hệ thống nhận dạng giọng nói (Speech-to-text)

Giảng viên hướng dẫn:

TS. Nguyễn Tuấn Dũng

Sinh viên thực hiện:

Nguyễn Đức Quyết – 20177021

Hà Nội, ngày 04 tháng 12 năm 2020

Mục lục

I. Nhận xét của giáo viên

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

II. Lời mở đầu

Với sự hướng dẫn tận tình của thầy **Nguyễn Tuấn Dũng**, đã hỗ trợ em rất nhiều để hoàn thành báo cáo môn học. Tuy nhiên trong quá trình tìm hiểu, học tập bản thân em còn nhiều thiếu sót. Em rất mong nhận được sự đóng góp của thầy. Em xin chân thành cảm ơn.

Nhận diện giọng nói đang xâm nhập vào cuộc sống hiện đại. Nó được cài đặt trong những chiếc điện thoại, điều khiển trò chơi hay những chiếc đồng hồ thông minh. Chỉ với khoảng $50, bạn có thể có Amazon Echo Dot - một chiếc hộp thần kỳ cho phép bạn đặt pizza, nhận thông tin dự báo thời tiết hoặc thậm chí mua những vật dụng - chỉ bằng cách đưa ra mệnh lệnh.

Speech-to-text (STT) hay còn được xem là speech reconigtion – nhận dạng giọng nói là 1 lĩnh vực con trong NLP – xử lý ngôn ngữ tự nhiên, trong dó STT phát triển lý thuyết và công nghệ nhận dạng và chuyển dữ liệu ngôn ngữ nhận dạng âm thanh thành ký tự.

III. Nội dung chi tiết

1. Lý thuyết

Tín hiệu âm thanh là gì?

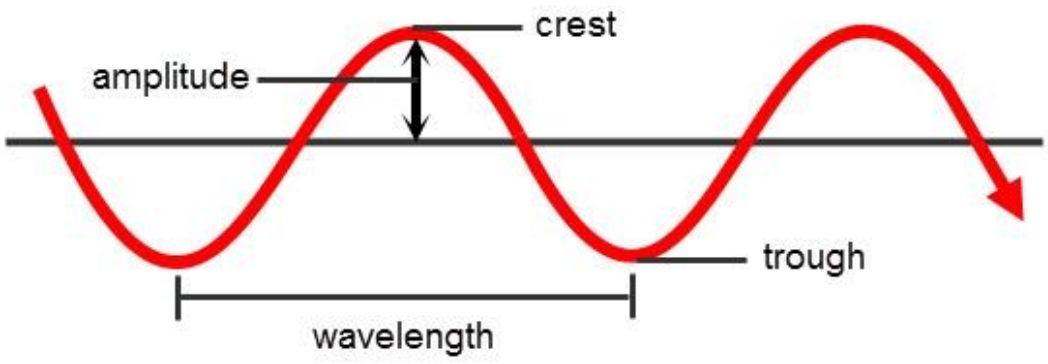
Điều này khá trực quan - bất kỳ đối tượng nào rung đều tạo ra sóng âm thanh. Bạn đã bao giờ nghĩ về cách chúng ta có thể nghe thấy giọng nói của ai đó? Đó là do sóng âm thanh. Hãy nhanh chóng hiểu quá trình đằng sau nó.

Khi một vật thể rung lên, các phân tử không khí dao động đến và đi từ vị trí nghỉ ngơi của chúng và truyền năng lượng của nó đến các phân tử lân cận. Điều này dẫn đến việc truyền năng lượng từ phân tử này sang phân tử khác, từ đó tạo ra sóng âm.

Âm thanh được truyền dưới dạng sóng, có thể lưu trữ âm thanh được ghi lại bằng độ cao của sóng âm sau các khoảng thời gian bằng nhau, phương thức này được gọi là lấy mẫu – **sampling**.

Thông số của tín hiệu âm thanh

- **Biên độ:** Biên độ đề cập đến sự dịch chuyển tối đa của các phân tử không khí từ vị trí còn lại
- **Đỉnh và máng:** Đỉnh là điểm cao nhất trong sóng trong khi máng là điểm thấp nhất
- **Bước sóng: Khoảng** cách giữa 2 đỉnh hoặc máng liên tiếp được gọi là bước song
- **Chu kỳ:** Mỗi tín hiệu âm thanh đi qua trong các hình thức của chu kỳ. Một chuyển động lên hoàn toàn và chuyển động xuống của tín hiệu tạo thành một chu kỳ
- **Tần số:** Tần số đề cập đến tốc độ thay đổi tín hiệu trong một khoảng thời gian



Một số thách thức

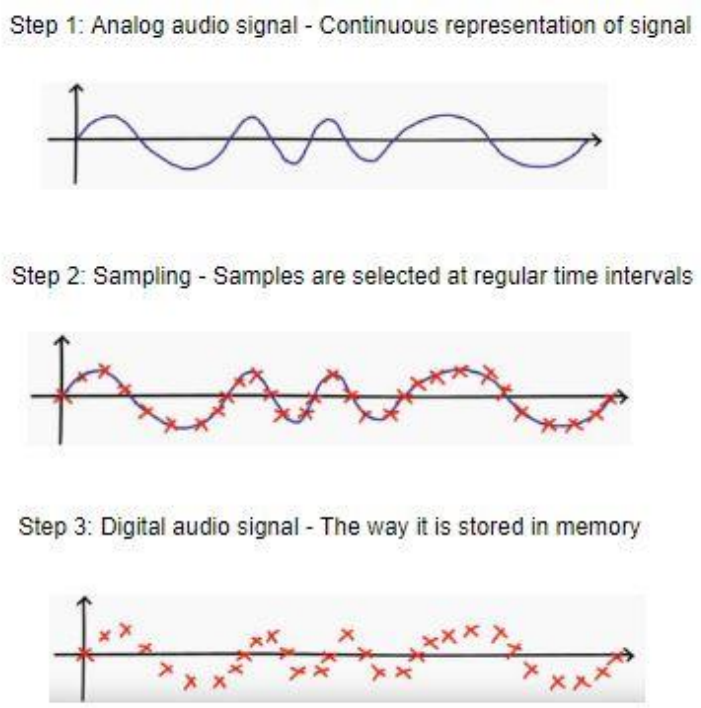
Cùng một nội dung nhưng khác nhau về âm điệu, cao độ, âm lượng,…

Cùng một từ nhưng có thể phát âm khác nhau tùy theo vùng, lãnh thổ, cùng nói một ngôn ngữ**.**

Chuyển âm thanh thành số

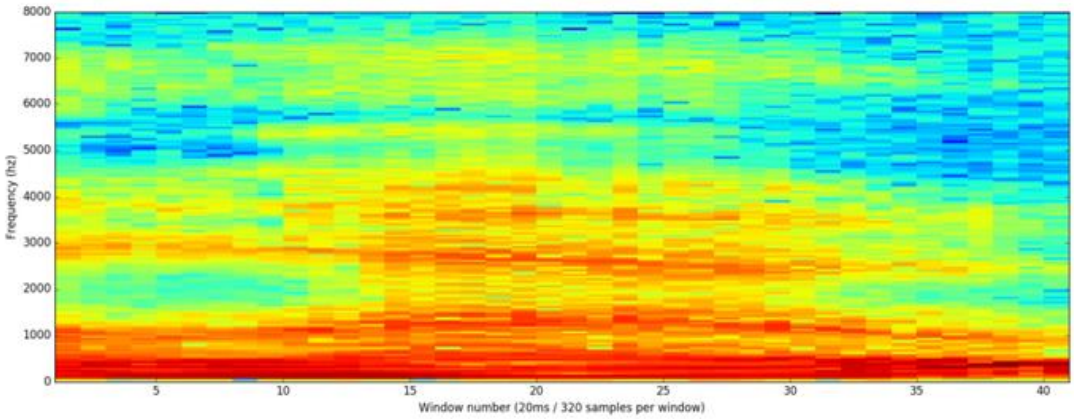
Bước đầu tiên trong nhận diện giọng nói là chúng ta cần truyền sóng âm vào máy tính.

Sóng âm có **một chiều** dữ liệu. Ở mỗi thời điểm, chúng có một giá trị cao độ. Để chuyển sóng âm thành số, chúng ta chỉ cần ghi lại cao độ(sampling).



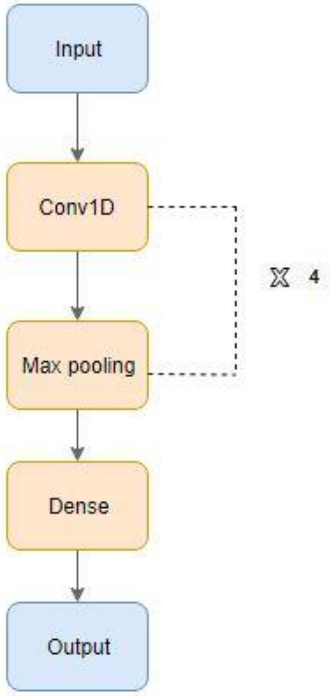
Do sóng âm rất phức tạp, kể cả những đoạn ghi âm ngắn cũng là một mớ hỗn độn vì chứa âm thấp, âm trung và cả âm cao. Để máy tính có thể dễ dàng hiểu và học được các đặc trưng của sóng âm, ta sử dụng biến đổi Fourier transform trong toán

học thành biểu đồ quang phổ giúp chúng ta thực sự nhìn thấy âm thanh và cấu trúc cao độ của nó. Và lấy kết quả này là đặc trưng mà ta truyền vào mạng nơron.



2. Giải pháp

Mô hình CNN: Convolutional neural network – mạng nơ-ron tích chập, là một trong những mô hình Deep Learning tiên tiến. giúp ta xây dựng được những hệ thống thông minh với độ chính xác cao. Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như *ReLu, Tanh* để kích hoạt.

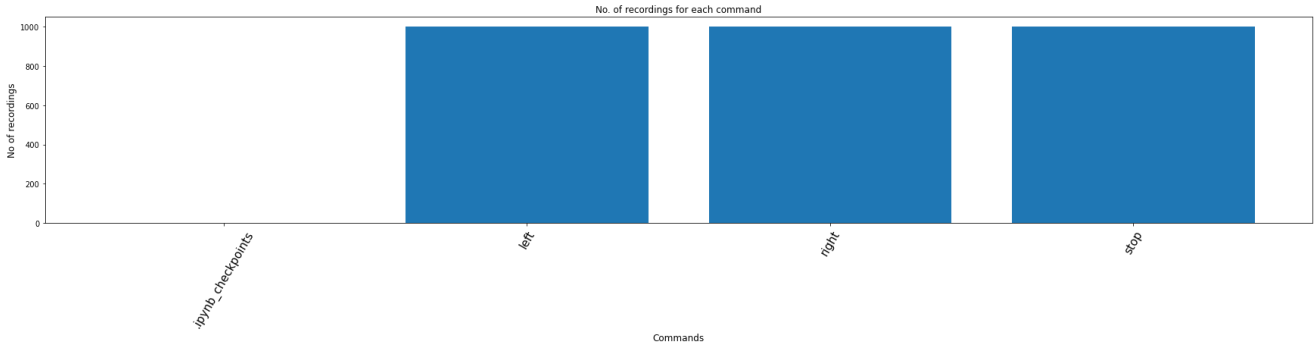


3. Source code

Tranning model sử dụng công cụ Jupyter Notebook.

3.1 Dataset

Sử dụng bộ dữ liệu giọng nói bao gồm 65.000 bản ghi của Tensorflow. Mỗi bản ghi âm có độ dài 1 giây bởi hàng ngàn người khác nhau. Trong khuôn khổ bản cáo cáo này, em sử dụng 3 từ bao gồm “left”, “right”, “stop”. Mỗi từ sử dụng 1000 bản ghi.



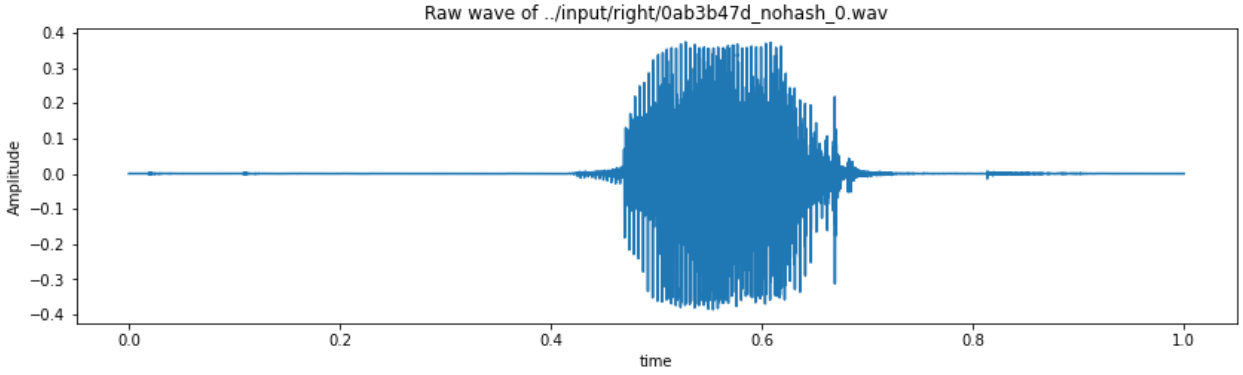
3.2 Tiền xử lý dữ liệu

Sử dụng thư viện **Librosa** và thư viện **SciPy** của Python để xử lý tín hiệu âm thanh.

```
import librosa
import os
import IPython.display as idp
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
import warnings
warnings.filterwarnings("ignore")
```

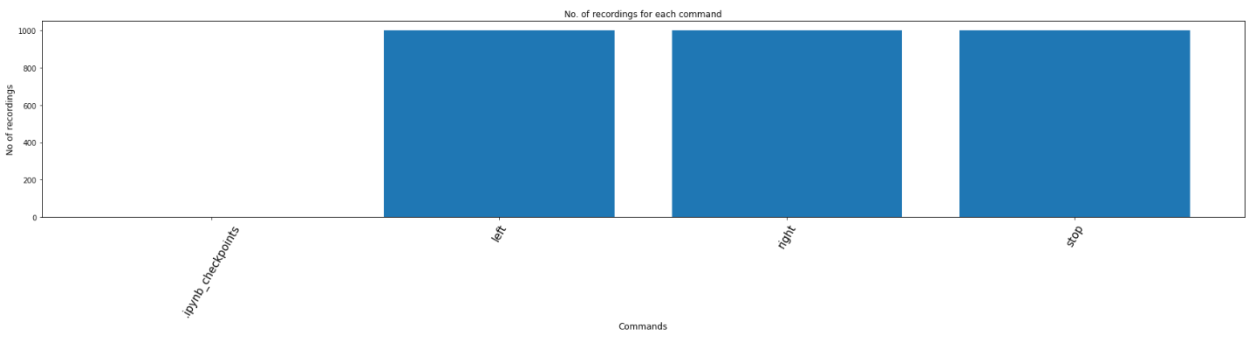
Tiền xử lý dữ liệu.

```
# processing audio file
train_audio_path = "./input/"
samples, sample_rate = librosa.load(train_audio_path+"right/0ab3b47d_nohash_0.wav" , sr=16000)
fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + '../input/right/0ab3b47d_nohash_0.wav')
ax1.set_xlabel('time')
ax1.set_ylabel('Amplitude')
ax1.plot(np.linspace(0, sample_rate/len(samples), sample_rate), samples)
```



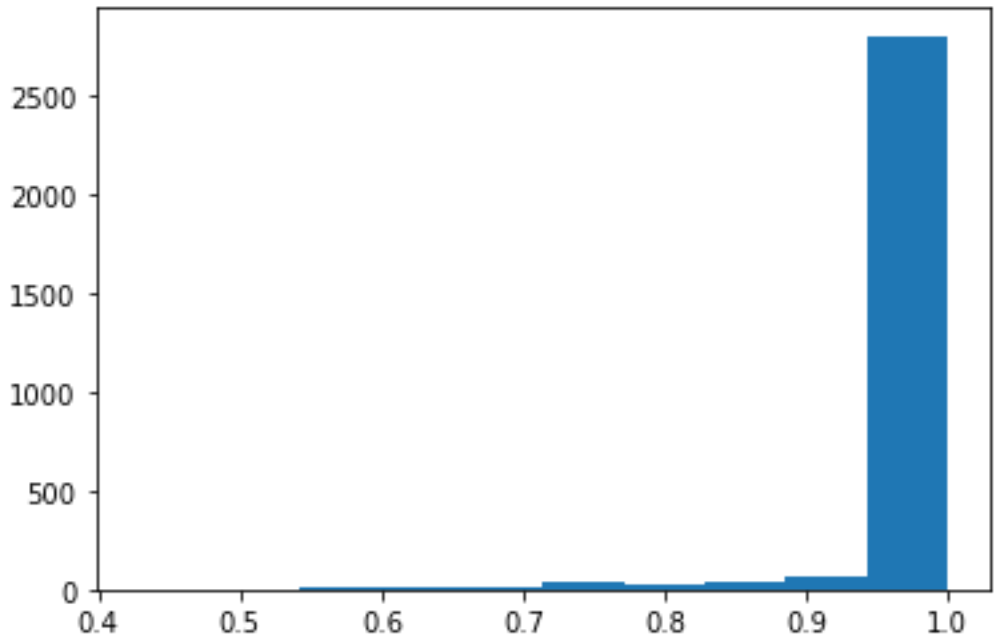
Lấy thông tin về bộ dữ liệu

```
labels=os.listdir(train_audio_path)

#find count of each label and plot bar graph
no_of_recordings=[]
for label in labels:
    print(label)
    waves = [f for f in os.listdir(train_audio_path + label) if f.endswith('.wav')]
    no_of_recordings.append(len(waves))


print(no_of_recordings)
# plot
plt.figure(figsize=(30,5))
index = np.arange(len(labels))
plt.bar(index, no_of_recordings)
plt.xlabel('Commands', fontsize=12)
plt.ylabel('No of recordings', fontsize=12)
plt.xticks(index, labels, fontsize=15, rotation=60)
plt.title('No. of recordings for each command')
plt.show()

labels=["left","right","stop"]
```



Tìm hiểu về các bản ghi âm

```
duration_of_recordings=[]
for label in labels:
    waves = [f for f in os.listdir(train_audio_path + '/'+ label) if f.endswith('.wav')]
    for wav in waves:
        sample_rate, samples = wavfile.read(train_audio_path + '/' + label + '/' + wav)
        duration_of_recordings.append(float(len(samples)/sample_rate))

plt.hist(np.array(duration_of_recordings))
```



Tiền xử lý sóng âm thanh

Trong phần khám phá dữ liệu trước đó, chúng tôi đã thấy rằng thời lượng của một vài bản ghi âm nhỏ hơn 1 giây và tỷ lệ lấy mẫu (***sampling***) quá cao. Vì vậy, chúng ta hãy đọc sóng âm thanh và sử dụng các bước dưới đây tiền xử lý để đối phó với điều này.

Dưới đây là hai bước chúng ta sẽ làm theo:

- Lấy mẫu lại
- Loại bỏ các mẫu ngắn hơn dưới 1 giây

```
train_audio_path = './input/'

all_wave = []
all_label = []
for label in labels:
    waves = [f for f in os.listdir(train_audio_path + '/'+ label) if f.endswith('.wav')]
    for wav in waves:
        samples, sample_rate = librosa.load(train_audio_path + '/' + label + '/' + wav, sr = 16000)
        samples = librosa.resample(samples, sample_rate, 8000)
        if(len(samples)== 8000) :
            all_wave.append(samples)
            all_label.append(label)
```

Mã hóa nhãn

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
le = LabelEncoder()
y=le.fit_transform(all_label)

classes= list(le.classes_)
classes

y=np_utils.to_categorical(y, num_classes=len(labels))
y
```

Chia bộ dữ liệu training set và validation set: (80-20)

```
from sklearn.model_selection import train_test_split
x_tr, x_val, y_tr, y_val = train_test_split(np.array(all_wave),
                                            np.array(y),
                                            stratify=y,
                                            test_size = 0.2,
                                            random_state=777,
                                            shuffle=True)
```

3.3 Mô hình

```
from keras.layers import Dense, Dropout, Flatten, Conv1D, Input, MaxPooling1D
from keras.models import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras import backend as K
K.clear_session()

inputs = Input(shape=(8000,1))

#First Conv1D layer
conv = Conv1D(8,13, padding='valid', activation='relu', strides=1)(inputs)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Second Conv1D layer
conv = Conv1D(16, 11, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Third Conv1D layer
conv = Conv1D(32, 9, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Fourth Conv1D layer
conv = Conv1D(64, 7, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Flatten layer
conv = Flatten()(conv)

#Dense Layer 1
conv = Dense(256, activation='relu')(conv)
conv = Dropout(0.3)(conv)

#Dense Layer 2
conv = Dense(128, activation='relu')(conv)
conv = Dropout(0.3)(conv)

outputs = Dense(len(labels), activation='softmax')(conv)

model = Model(inputs, outputs)
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 8000, 1)]         0
_________________________________________________________________
conv1d (Conv1D)              (None, 7988, 8)           112
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 2662, 8)           0
_________________________________________________________________
dropout (Dropout)            (None, 2662, 8)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2652, 16)          1424
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 884, 16)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 884, 16)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 876, 32)           4640
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 292, 32)           0
_________________________________________________________________
dropout_2 (Dropout)          (None, 292, 32)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 286, 64)           14400
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 95, 64)            0
_________________________________________________________________
dropout_3 (Dropout)          (None, 95, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 6080)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1556736
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
=================================================================
Total params: 1,610,595
Trainable params: 1,610,595
Non-trainable params: 0
```

Xác định loss function là: *categorical cross-entropy* cho vấn đề đa phân loại:

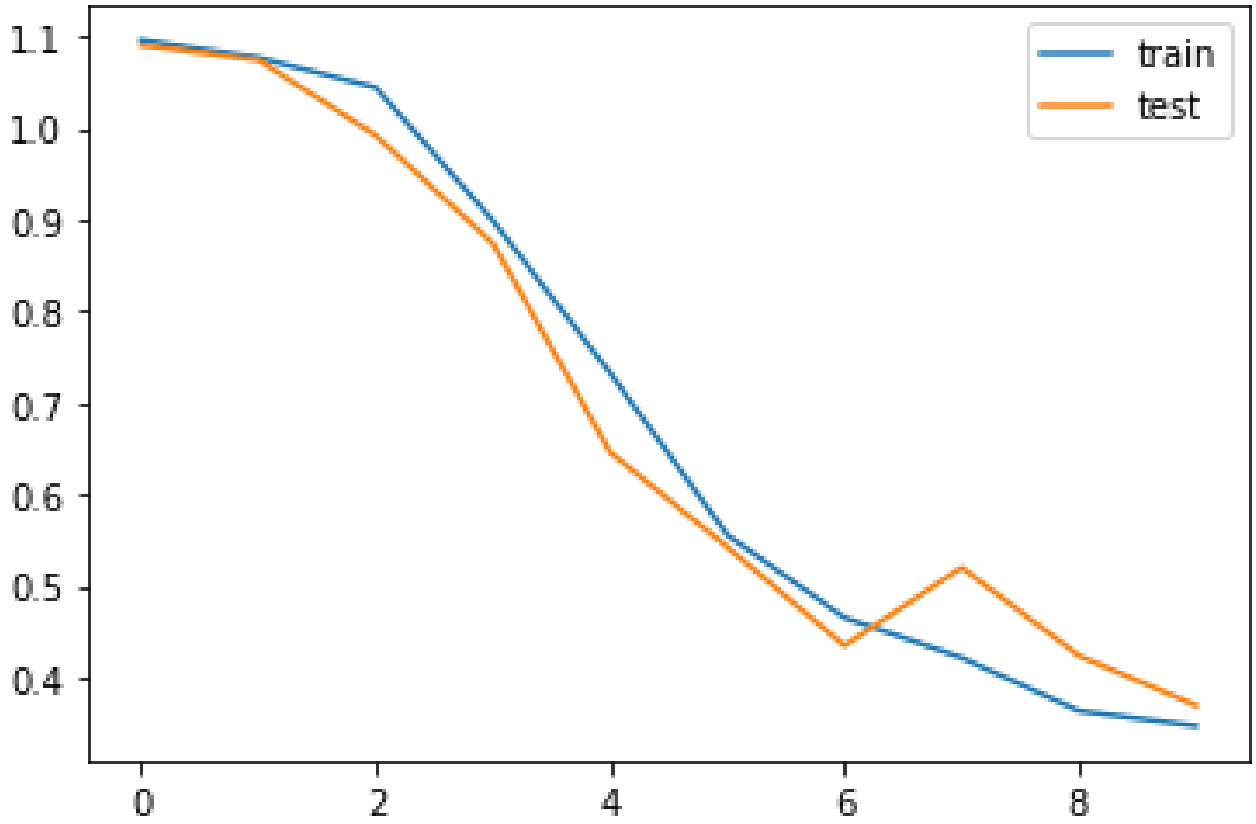
```
1  model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
1  history=model.fit(x_tr, y_tr ,epochs=10, callbacks=[es,mc], batch_size=32, validation_data=(x_val,y_val))
```

3.4 Đánh giá

```
Epoch 10/10
69/69 [==============================] - ETA: 0s - loss: 0.3482 - accuracy: 0.8657WARNING:tensorflow:Can save best model only w
ith val_acc available, skipping.
69/69 [==============================] - 11s 161ms/step - loss: 0.3482 - accuracy: 0.8657 - val_loss: 0.4345 - val_accuracy: 0.
8597
```

```python
from matplotlib import pyplot
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```



3.5 Nhận xét

Giá trị **accuracy** chưa được cao, đạt : **86,57%**

3.6 Phương hướng phát triển