TRƯỜNG ĐẠI HỌC PHENIKAA
KHOA CÔNG NGHỆ THÔNG TIN

Deep Learning

BÀI TẬP LỚN

Recurrent Neural Network for Sign Language Recognition

Giảng viên hướng dẫn: TS. Nguyễn Văn Tới
Sinh viên thực hiện: Hứa Phương Nam - 20010745
Nguyễn Đức Hải - 21010560

HÀ NỘI, 10/2023

Contents

1 Lời mở đầu **4**

2 Giới thiệu **5**

3 Recurrent Neural Network **6**
3.1 Ý tưởng và sự ra đời 6
3.2 Cấu trúc RNN cơ bản 7
3.2.1 Ghi nhớ trạng thái 7
3.2.2 Hàm mất mát 9
3.2.3 Backpropagation 9
3.3 Nhận xét 10
3.3.1 Điểm mạnh và điểm yếu 10

4 Long-short Term Memory **11**
4.1 Mô hình LSTM 11
4.2 Backpropagation 12

5 Tập dữ liệu LSA64 **13**

6 Quá trình huấn luyện **15**
6.1 Tiền xử lí và thiết kế mô hình 15
6.1.1 Tiền xử lí 15
6.2 Thiết kế mô hình 15
6.2.1 Kết quả 16

7 Công việc tiếp theo **17**

8 Kết luận **18**

9 Phân công công việc **19**

List of Figures

2.1 Phân loại các chủ đề lớn trong Sign Language Recognition 5
3.1 Các dạng cấu trúc RNN dựa trên bài toán 7
3.2 Trạng thái ẩn của RNN 8

3.3 Mô hình RNN cho bài toán 8
4.1 Mô hình LSTM. Lưu ý rằng phép nhân trong hình là phép nhân ma trận bình thường (elementwise multiplication) 12
4.2 Backpropagation trong LSTM 12
5.1 Danh sách các ký hiệu và bàn tay (cột H). Cột H cho chúng ta biết ký hiệu này được thực hiện bởi tay phải hoặc cả 2 bàn tay 13
5.2 Một vài khung hình từ LSA64 14
6.1 Thông tin mô hình 15
6.2 Mô hình thực hiện dự đoán ký hiệu trong video. Như ta thấy, ký hiệu được mô hình dự đoán có tỉ lệ cao nhất là "Opaque" (có nghĩa là "mờ đục"), trùng khớp với label thực của video 16

List of Tables

1 Lời mở đầu

Ngôn ngữ ký hiệu (Sign Language) đóng vai trò quan trọng trong việc giao tiếp của cộng đồng người điếc và câm trên khắp thế giới. Để tạo ra các hệ thống hỗ trợ giao tiếp cho người sử dụng ngôn ngữ ký hiệu, việc nhận dạng và hiểu rõ các biểu hiện ngôn ngữ ký hiệu là một thách thức quan trọng. Báo cáo này tập Recurrent Neural Network (RNN) và các biến thể của nó trong việc nhận dạng ngôn ngữ ký hiệu.

Trong báo cáo này, chúng tôi sẽ tập trung vào giới thiệu tác vụ nhận dạng ngôn ngữ ký hiệu, thảo luận và giải thích về RNN cũng như những cải tiến của nó so với mạng neuron sâu thông thường; những phương pháp cải tiến và biến thể của RNN. Mô hình RNN sẽ được ứng dụng huấn luyện trên một tập dữ liệu Sign Language tiêu biểu là LSA64 (Argentinian Sign Language). Sau đó, chúng tôi sẽ báo cáo và nhận xét các kết quả thu được.

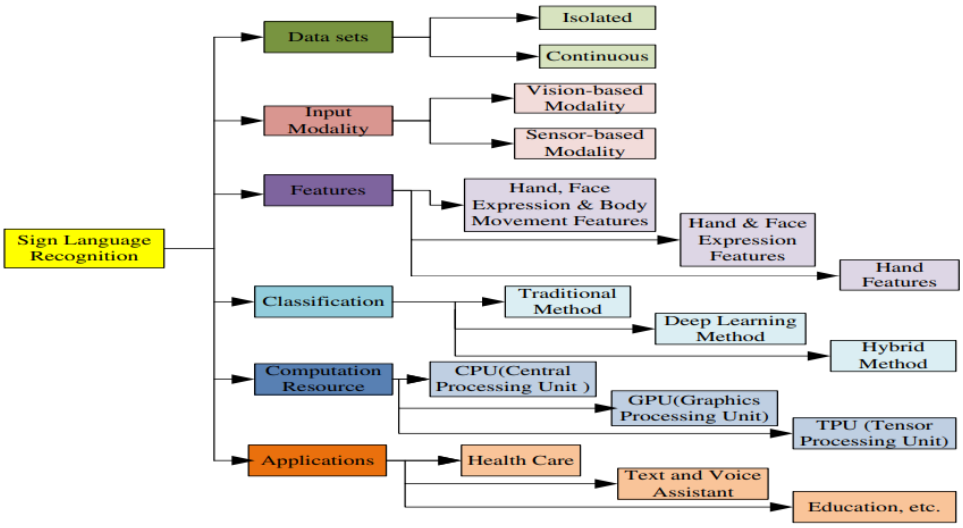

Figure 2.1: Phân loại các chủ đề lớn trong Sign Language Recognition

2 Giới thiệu

Theo số liệu của tổ chức WHO (World Health Organization), khoảng 466 triệu người trên khắp thế giới gặp khó khăn hoặc không thể nghe / nói, trong đó 80% số người này không được giáo dục đọc / viết. Những phương thức giao tiếp bằng cử chỉ, ký hiệu được sử dụng để bình thường hóa cuộc sống của phần dân số này, do đó đóng một vai trò vô cùng quan trọng.

Ngôn ngữ cử chỉ rất khác biệt so với ngôn ngữ nói. Mỗi một ký hiệu bao gồm các cử chỉ của tay, mặt và cơ thế đặc trưng. Sign Language Recognition (nhận dạng ngôn ngữ cử chỉ, ký hiệu - SLR) là một tác vụ phức tạp và nhiều khó khăn, đã thu hút sự chú ý của giới nghiên cứu AI kể từ khi xuất hiện. Hiện nay, rất nhiều các mô hình AI tiên tiến và các tập dữ liệu dành cho SLR đã xuất hiện.

Các tập dữ liệu dành cho SLR thường nằm dưới dạng hình ảnh (dữ liệu cô lập) hoặc video (dữ liệu dạng chuỗi). Các kỹ thuật phân loại và nhận dạng được chia thành 3 loại: phương pháp truyền thống, deep learning và kết hợp (hybrid).

Trong báo cáo này, chúng ta sẽ nghiên cứu về cách ứng dụng Recurrent Neural Network - một phương pháp học sâu - vào SLR.

3 Recurrent Neural Network

3.1 Ý tưởng và sự ra đời

Sự phát triển của AI đã đưa tới các bài toán yêu cầu xử lí với đầu vào hoặc đầu ra là chuỗi, tỉ dụ như video. Giả sử có một bài toán như sau: cần phân loại hành động của người trong video, đầu vào là video kéo dài 30 giây, tốc độ 1 FPS (1 khung ảnh mỗi giây), đầu ra là phân loại hành động.

Rõ ràng mạng CNN (convolutional neural network) bình thường không thể xử lí được bài toán này, bởi lẽ ảnh đầu vào của chúng ta là 30 ảnh được sắp xếp theo thứ tự và thứ tự đó có ý nghĩa.

- Nếu ta đảo lộn thứ tự của các ảnh, nội dung video có thể sẽ bị thay đổi.
- Một ảnh duy nhất không thể bao hàm nội dung của cả video.

Chúng ta cần một mô hình mới có thể giải quyết được những bài toán với đầu vào ở dạng chuỗi. Ý tưởng đầu tiên mà các nhà nghiên cứu đưa ra chính là bằng cách nào đó giúp mô hình hiện tại "nhớ" được điều gì xảy ra ở những iteration trước và có thể sử dụng lại những thông tin đó.

Ý tưởng về RNN đã xuất hiện từ trước khi AI thực sự trở thành một ngành khoa học nghiên cứu. Mô hình Ising (1925), được phát minh bởi hai nhà vật lý Wilhelm Lenz và Ernst Ising là cấu trúc RNN đầu tiên trong lịch sử. Tuy nhiên, mô hình toán học này không thể học. Vào năm 1972, nhà nghiên cứu Shun'ichi Amari đã cải tiến mô hình Ising để có thể học được. Mô hình mới được đặt tên là mạng Hopfield. Báo cáo kỹ thuật đầu tiên gọi RNN là "recurrent net(work)s" là *Learning internal representations by error propagation* của David Rumelhart và Geoffrey Hinton.

Sự xuất hiện của RNN được xem là khởi đầu cho nhiều lĩnh vực sử dụng Deep Learning.

- **Speech to text**: chuyển đổi giọng nói sang văn bản.
- **Sentiment analysis**: phân tích bình luận.
- **Machine translation**: dịch tự động giữa các ngôn ngữ.
- **Video recognition**: nhận diện hành động trong video.
- **Medical applications**: Các ứng dụng vào dữ liệu dạng chuỗi trong y tế.

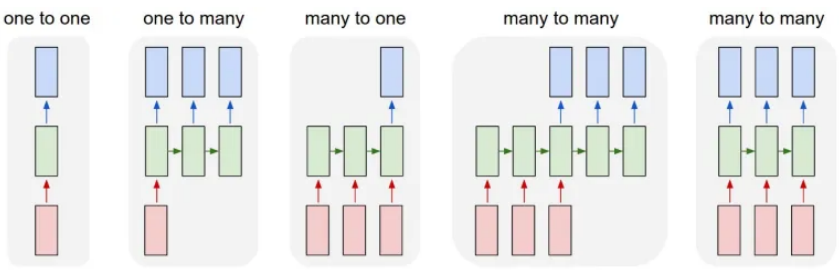

Figure 3.1: Các dạng cấu trúc RNN dựa trên bài toán

3.2 Cấu trúc RNN cơ bản

RNN có rất nhiều biến thể. Cấu trúc của RNN cũng sẽ thay đổi tùy thuộc vào bài toán mà mô hình cần xử lí. Hình 3.1 miêu tả sự thay đổi đầu vào và đầu ra của RNN tùy thuộc yêu cầu bài toán.

- One to one: mẫu bài toán cơ bản cho NN và CNN cơ bản, gồm 1 input và 1 output duy nhất.
- One to many: mẫu bài toán có 1 input nhưng nhiều output. Ví dụ tiêu biểu là tóm tắt nội dung ảnh.
- Many to one: mẫu bài toán có nhiều input nhưng chỉ có 1 output. Ví dụ tiêu biểu là phân loại hành động trong video.
- Many to many: mẫu bài toán có nhiều input và output. Ví dụ tiêu biểu là dịch tự động.

Tuy nhiên, chúng tôi sẽ chỉ phân tích những đặc điểm cơ bản nhất mô hình RNN cơ bản nhất.

3.2.1 Ghi nhớ trạng thái

RNN duy trì một trạng thái ẩn được cập nhật mỗi khi xử lí một đầu vào mới, gọi là s. Cụ thể, trạng thái s_t sẽ có đầu vào là x_t và s_{t-1} (đầu ra của trạng thái trước đó), đầu ra sẽ là $s_t = f_{param}(s_{t-1}, x_t)$ với f_{param} là một hàm truy hồi có các tham số $param$ không đổi. f_{param} thường là hàm tanh hoặc ReLU.

Lưu ý rằng ở mỗi timestep, hàm hồi quy và tập các tham số đều không đổi.

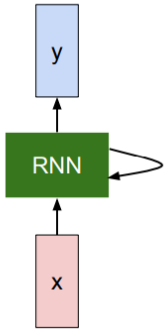

Figure 3.2: Trạng thái ẩn của RNN

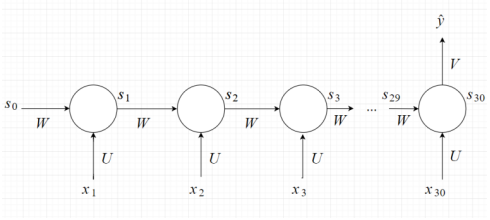

Figure 3.3: Mô hình RNN cho bài toán

Quay trở lại với công thức truy hồi. Ta sẽ xử lí bài toán được nêu ra ở mục 3.1.

$$s_t = f(U * x_t + W * s_{t-1})$$
$$\hat{y} = g(V * s_{30})$$

Lúc này,

- x_i là vector có kích thước n $\times$ 1.
- s_i là vector có kích thước m $\times$ 1.
- $\hat{y}$ là vector có kích thước d $\times$ 1.
- U là ma trận có kích thước m $\times$ n.
- W là ma trận có kích thước m $\times$ m.
- V là ma trận có kích thước d $\times$ m.
- f là hàm tanh hoặc ReLU.

- Chỉ có một đầu ra cho bài toán nên ta đặt nó ở state cuối cùng, tức s_{30}. s_{30} sẽ học được thông tin từ tất cả các đầu vào, tức các frame ảnh. g sẽ là hàm kích hoạt, trong trường hợp này là softmax vì ta đang giải quyết bài toán phân loại.

3.2.2 Hàm mất mát

Hàm mất mát của cả mô hình sẽ bằng tổng mất mát của các đầu ra ở từng timestep. Tuy nhiên, bài toán ta đang giải quyết chỉ có 1 đầu ra và đang dùng bài toán phân loại, ta sẽ chọn categorical cross entropy loss.

$$L = \sum_{i=1}^{C} y_i \times log(\hat{y}_i) \tag{3.1}$$

Ở đây C là số loại hành động.

3.2.3 Backpropagation

Có ba tham số ta cần phải tìm là W, U và V. Để thực hiện gradient descent, ta cần tính $\frac{\partial L}{\partial U}$, $\frac{\partial L}{\partial V}$, $\frac{\partial L}{\partial W}$.

Tương đối đơn giản khi tính đạo hàm V: $\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial V}$

Tuy nhiên với U và W lại khác. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial W}$. Do $s_{30} = f(W * s_{29} + V * x_{30})$ nên ta có thể áp dụng công thức tách đạo hàm hàm hợp $(f(x) * g(x))' = f'(x) * g(x) + f(x) * g'(x)$. Ta có $\frac{\partial s_{30}}{\partial W} = \frac{\partial s'_{30}}{\partial W} + \frac{\partial s_{30}}{\partial s_{29}} * \frac{\partial s_{29}}{\partial W}$. Tương tự trong biểu thức của s_{29} sẽ có s_{28} phụ thuộc vào W, s_{28} có s_{27} phụ thuộc vào W nên ta có công thức cuối cùng:

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{30} \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial s_i} * \frac{\partial s'_i}{\partial W}$$

$$\frac{\partial s_{30}}{\partial s_i} = \prod_{j=i}^{29} \frac{\partial s_{j+1}}{\partial s_j}$$

Giả sử hàm f của chúng ta là hàm tanh, $s_t = tanh(U * x_t + W * s_{t-1})$. Lúc này $\frac{\partial s_t}{\partial s_{t-1}} = (1 - s_t^2) * W$, suy ra $\frac{\partial s_{30}}{\partial s_i} = W^{30-i} * \prod_{j=i}^{29}(1 - s_j^2)$. Với $s_j < 1$, $W < 1$ thì tại những trạng thái đầu tiên, $\frac{\partial s_{30}}{\partial s_i} \approx 0$ hay $\frac{\partial L}{\partial W} \approx 0$, xuất hiện **vanishing gradient**.

Chúng ta có thể thực hiện forward passing và backpropagation trên các đoạn của chuỗi đầu vào thay vì toàn bộ chuỗi đầu vào đó. Phương pháp này gọi là truncated backpropagation.

3.3 Nhận xét

3.3.1 Điểm mạnh và điểm yếu

Điểm mạnh của RNN

- Có thể xử lí chuỗi đầu vào với độ dài tùy ý.
- Trên lý thuyết, việc tính toán ở bước t có thể sử dụng thông tin từ bất kỳ bước nào trước đó.
- Kích thước mô hình không tăng lên theo kích thước chuỗi dầu vào.

Điểm yếu của RNN

- Tính toán truy hồi rất chậm
- Trên thực tế, rất khó tiếp cận thông tin từ các bước cách xa hiện tại.
- Dễ bị ảnh hưởng bởi vanishing gradient.

Vấn đề vanishing gradient đòi hỏi một cấu trúc mô hình tiên tiến hơn, hiệu quả hơn, có thể thật sự nhớ được cả các thông tin ở những trạng thái xa hơn.

4 Long-short Term Memory

Long-short term memory (LSTM) được hai nhà nghiên cứu Sepp Hochreister và Juergen Schmidhuber giới thiệu vào năm 1995, là một sự cải thiện đáng kể so với mô hình RNN thông thường.

Ý tưởng đằng sau kiến trúc LSTM chính là tạo ra một phương thức nội tại giúp mạng neuron của chúng ta biết khi nào thì nên nhớ và khi nào thì nên quên các thông tin liên quan. Nói cách khác, mô hình có thể học một cách hiệu quả những thông tin cần thiết sau này và loại bỏ nó khi đã không còn cần thiết. Ví dụ, một LSTM có thể xử lí câu "Việt Nam, dưới sự dẫn dắt của Đảng và Nhà nước, đã trải qua hai cuộc chiến tranh vệ quốc vĩ đại và giành được độc lập" bằng cách ghi nhớ sắc thái từ ngữ của chủ ngữ "Việt Nam", hiểu rằng thông tin này quan trọng cho những từ như "trải qua" và "giành được" cũng như biết rằng nó không còn cần thiết sau từ "giành được".

LSTM sau khi xuất hiện đã phá vỡ nhiều kỷ lục về độ chính xác và tốc độ trong các cuộc thi về trí tuệ nhân tạo, cũng như tạo ra các đột phá trong các lĩnh vực như nhận diện giọng nói. Một trong những "hậu bối" của LSTM là ResNet đã trở thành nền tảng cho lĩnh vực Deep Learning hiện tại, là bài báo khoa học về mạng neuron được trích dẫn nhiều nhất thế kỉ 21.

4.1 Mô hình LSTM

Ở trạng thái thứ t của mô hình LSTM.

- Đầu ra c_t, h_t, ta gọi c là cell state, h là hidden state.
- Đầu vào s_{t-1}, h_{t-1}, x_t. Trong đó, x_t là đầu vào ở trạng thái thứ t của mô hình, còn s_{t-1} và h_{t-1} là cell state và hidden state của trạng thái trước.
- Forget gate (dùng để quên thông tin): $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$, b_f là hệ số bias.
- Input gate $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$, b_i là hệ số bias.
- Output gate $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$, b_o là hệ số bias.
- $0 < f_t, i_t, o_t < 1$, các hệ số W, U giống như RNN.
- $\tilde{c}_t = tanh(U_c * x_t + W_c * h_{t-1} + b_c)$, bước này giống hệt như khi tính s_t trong RNN.

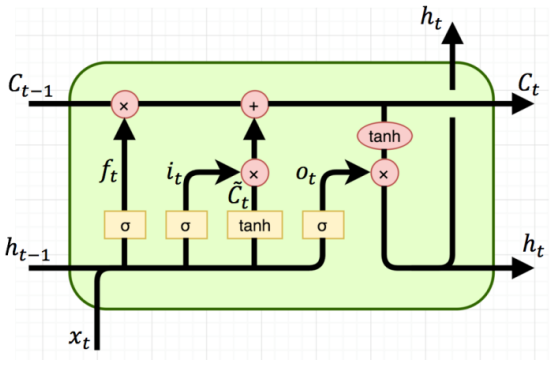

Figure 4.1: Mô hình LSTM. Lưu ý rằng phép nhân trong hình là phép nhân ma trận bình thường (elementwise multiplication)

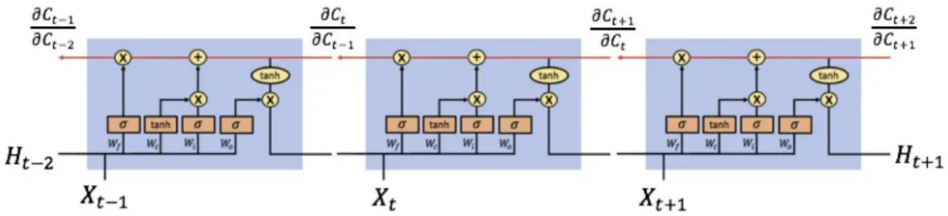

Figure 4.2: Backpropagation trong LSTM

- $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$. Forget gate sẽ quyết định xem cần lấy bao nhiêu thông tin từ cell state trước còn input gate sẽ quyết định lấy bao nhiêu thông tin từ đầu vào và hidden state của trạng thái trước đó.

- $h_t = o_t * tanh(c_t)$. Output gate sẽ quyết định xem phải lấy bao nhiêu thông tin từ cell state để trở thành đầu ra của hidden state. Ngoài ra h_t cũng được sử dụng để tính ra kết quả y_t của trạng thái thứ t.

4.2 Backpropagation

Thành phần chính gây vanishing gradient trong RNN cơ bản là $\frac{\partial s_{t+1}}{\partial s_t} = (1 - s_t^2) * W$, trong đó $s_t, W < 1$. Tương tự trong LSTM ta sẽ quan tâm tới $\frac{\partial c_{t+1}}{\partial c_t} = f_t$. Do $0 < f_t < 1$ nên LSTM vẫn bị vanishing gradient. Tuy nhiên, do mô hình mang thông tin trên cell state nên ít khi cần phải quên đi giá trị của cell cũ, nên $f_t \approx 1$, ít bị vanishing gradient ảnh hưởng hơn.

ID	Name	H	ID	Name	H	ID	Name	H	ID	Name	H
01	Opaque	R	17	Call	R	33	Hungry	R	49	Yogurt	R
02	Red	R	18	Skimmer	R	34	Map	B	50	Accept	B
03	Green	R	19	Dinner	R	35	Coin	R	51	Thanks	R
04	Yellow	R	20	Sweet milk	R	36	Music	B	52	Shut down	R
05	Bright	R	21	Milk	R	37	Shirt	R	53	Appear	R
06	Light-blue	R	22	Water	R	38	None	R	54	To land	B
07	Colors	R	23	Food	R	39	Name	R	55	Catch	R
08	Pink	R	24	Argentina	R	40	Patience	R	56	Help	B
09	Women	R	25	Uruguay	R	41	Perfume	R	57	Dance	R
10	Enemy	R	26	Country	R	42	Deaf	R	58	Bathe	B
11	Son	R	27	Last name	R	43	Trap	R	59	Buy	R
12	Man	R	28	Where	R	44	Rice	B	60	Copy	B
13	Away	R	29	Mock	R	45	Barbecue	R	61	Run	R
14	Drawer	R	30	Birthday	R	46	Candy	R	62	Realize	R
15	Born	R	31	Breakfast	R	47	Chewing-gum	R	63	Give	R
16	Learn	R	32	Photo	B	48	Spaghetti	B	64	Find	R

Figure 5.1: Danh sách các ký hiệu và bàn tay (cột H). Cột H cho chúng ta biết ký hiệu này được thực hiện bởi tay phải hoặc cả 2 bàn tay

5 Tập dữ liệu LSA64

LSA64 là tập dữ liệu về ngôn ngữ ký hiệu cho tiếng Argentina, bao gồm 3200 video khác nhau. 10 nhân vật khác nhau lặp lại 5 lần một tập hợp 64 loại ký hiệu khác nhau, được chọn trong những ký hiệu thường được sử dụng nhất trong từ vựng của ngôn ngữ ký hiệu Argentina.

Tập dữ liệu được chia thành 2 phần. Ở phần thứ nhất, 23 ký hiệu chỉ sử dụng 1 bàn tay được ghi lại. Ở phần thứ hai có thêm 41 ký hiệu (22 ký hiệu sử dụng cả 2 bàn tay và 19 ký hiệu chỉ sử dụng 1 bàn tay).

Trong cả hai phần dữ liệu, những người thực hiện đều mặc quần áo màu đen trên hậu cảnh là một bức tường màu trắng. Ánh sáng trong phần thứ nhất là ánh sáng tự nhiên, trong phần thứ hai là ánh sáng nhân tạo trong phòng kín. Những người thực hiện đều đeo găng tay màu để hỗ trợ tác vụ cắt ghép ảnh (segmentation). Camera sử dụng cho quá trình quay video là Sony HDR-CX420, độ phân giải mỗi video là 1920 x 1080.

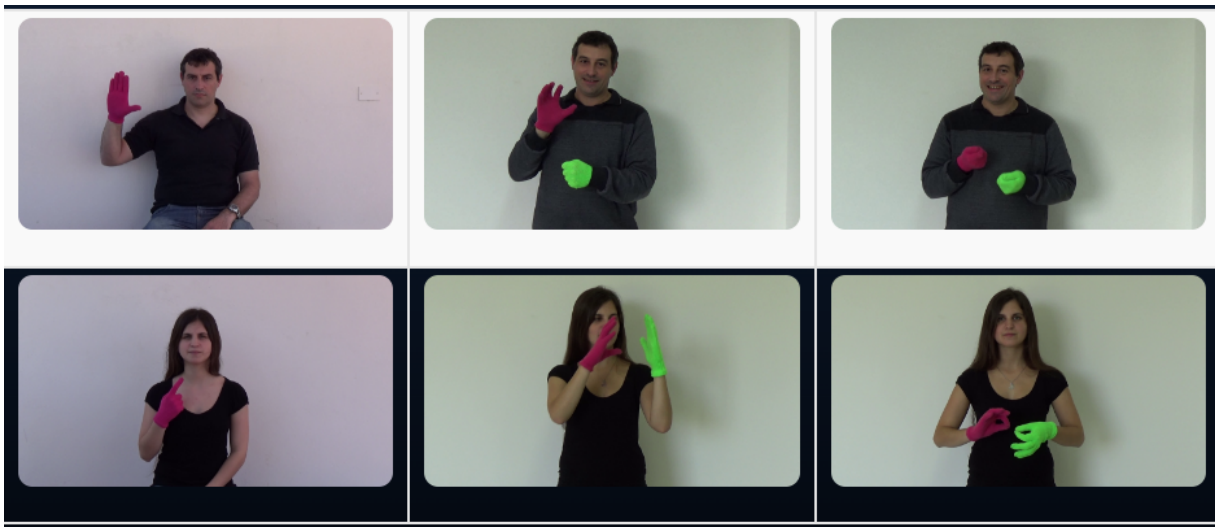

Figure 5.2: Một vài khung hình từ LSA64

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_5 (InputLayer)        [(None, 50, 2048)]        0         []

 input_6 (InputLayer)        [(None, 50)]              0         []

 gru (GRU)                   (None, 50, 64)            405888    ['input_5[0][0]',
                                                                  'input_6[0][0]']

 dropout (Dropout)           (None, 50, 64)            0         ['gru[0][0]']

 gru_1 (GRU)                 (None, 128)               74496     ['dropout[0][0]']

 dropout_1 (Dropout)         (None, 128)               0         ['gru_1[0][0]']

 dense (Dense)               (None, 128)               16512     ['dropout_1[0][0]']

 dense_1 (Dense)             (None, 4)                 516       ['dense[0][0]']

=================================================================================================
Total params: 497412 (1.90 MB)
Trainable params: 497412 (1.90 MB)
```

Figure 6.1: Thông tin mô hình

6 Quá trình huấn luyện

6.1 Tiền xử lí và thiết kế mô hình

6.1.1 Tiền xử lí

Do số lượng video tương đối lớn, mỗi lần trích xuất khung ảnh từ 1 video ta sẽ chỉ lấy 10 khung ảnh. Ngoài ra, các khung ảnh sẽ được cắt bớt để lấy được phần trung tâm.

Các nhãn (label) của video đều là dạng chuỗi (string). Chúng ta sẽ biến đổi các nhãn này về dạng số học bằng hàm **keras.layers.StringLookup**.

6.2 Thiết kế mô hình

Trước khi xử dụng RNN-LSTM để huấn luyện mô hình, chúng ta sẽ sử dụng một mô hình khác để trích xuất các đặc trưng từ các khung ảnh video. Mô hình mà chúng tôi chọn là InceptionV3. Sau đó, các đặc trưng dưới dạng chuỗi sẽ được đưa vào mô hình RNN-LSTM.

Mô hình mà chúng tôi sử dụng sẽ là Gate Recurrent Unit (GRU, một dạng LSTM không có output gate). Đặc biệt, chúng tôi cung cấp cho GRU tín hiệu về việc nên hay không nên sử dụng thông tin ở từng timestep, gọi là các "mask". Cấu trúc mô hình như sau.
Các siêu tham số và thông tin cần thiết khác được đặt như sau:

- Kích thước hình ảnh: 224×224
- Kích thước batch dữ liệu để huấn luyện: 128

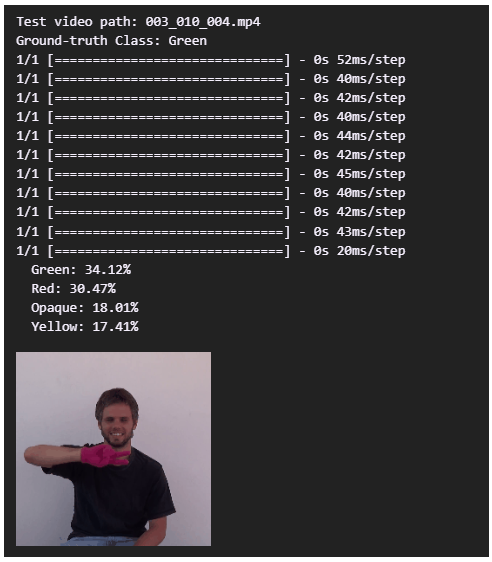

Figure 6.2: Mô hình thực hiện dự đoán ký hiệu trong video. Như ta thấy, ký hiệu được mô hình dự đoán có tỉ lệ cao nhất là "Opaque" (có nghĩa là "mờ đục"), trùng khớp với label thực của video

- Số epoch huấn luyện: 300
- Tốc độ học: 0.0001
- Độ dài tối đa của dữ liệu dạng chuỗi được truyền vào mô hình RNN-LSTM: 50
- Số lượng đặc trưng được trích xuất từ mỗi khung hình: 2048
- Hàm mất mát: categorical crossentropy
- Optimizer: Adam
- Phương thức regularization: early stopping

6.2.1 Kết quả

Mô hình của chúng tôi đạt được độ chính xác 72.5% trên tập dữ liệu test. Tuy nhiên, thời gian để mô hình InceptionV3 trích xuất đặc trưng và thời gian huấn luyện tương đối lâu, có thể lên đến 1 giờ nếu như chỉ sử dụng CPU.

7 Công việc tiếp theo

Độ chính xác còn hạn chế là một thử thách lớn trong tương lai. Chúng tôi sẽ xem xét hiệu suất của mô hình RNN hiện tại và thực hiện các biện pháp để tối ưu hóa nó. Các công việc tiếp theo có thể bao gồm:

- Fine-tuning mô hình: Điều chỉnh siêu tham số của mô hình để cải thiện hiệu suất. Điều này có thể bao gồm việc điều chỉnh tỷ lệ học, số lớp ẩn, số đơn vị trong mỗi lớp, và số lượng thời điểm trong chuỗi.
- Tinh chỉnh dữ liệu: Tạo thêm dữ liệu bằng cách áp dụng các kỹ thuật tăng cường dữ liệu, chẳng hạn như xoay, thu phóng, hoặc sửa đổi hình ảnh ngôn ngữ ký hiệu để tạo sự đa dạng và cải thiện khả năng tổng quát của mô hình.
- Tối ưu quá trình huấn luyện: Tối ưu hóa quá trình huấn luyện bằng cách sử dụng các kỹ thuật để giảm dư overfitting và kiểm soát việc huấn luyện qua các epoch.

8 Kết luận

Trong đề tài này, chúng tôi đã tiến hành nghiên cứu và phát triển một hệ thống sử dụng mạng nơ-ron hồi quy (RNN) để dự đoán ngôn ngữ ký hiệu, góp một phần nhỏ trong việc áp dụng trí tuệ nhân tạo vào việc giúp cộng đồng người khiếm thính giao tiếp một cách hiệu quả.

Kết quả cuối cùng của dự án này cho thấy sự hiệu quả của trí tuệ nhân tạo trong việc nâng cao cuộc sống của người khiếm thính bằng cách cung cấp phần "lõi" công nghệ cho một công cụ mạnh mẽ giúp họ tương tác và giao tiếp. Chúng tôi tin rằng việc phát triển và cải tiến mô hình dự đoán ngôn ngữ ký hiệu sẽ tiếp tục đóng vai trò quan trọng trong tương lai và là một sự cống hiến đáng kể đối với lĩnh vực trí tuệ nhân tạo phục vụ con người nói chung.

9 Phân công công việc

- Nguyễn Đức Hải: tìm hiểu bài toán và mô hình, soạn thảo báo cáo, viết và chỉnh sửa code.
- Hứa Phương Nam: tìm hiểu bài toán và mô hình, chỉnh sửa báo cáo.

References

[1] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation 9*, 1997.

[2] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by backpropagating errors," *Nature*, 1986.

[3] F. Ronchetti, F. Quiroga, C. Estrebou, L. Lanzarini, and A. Rosete, "LSA64: An Argentinian Sign Language Dataset," *XXII Congreso Argentino de Ciencas de la Computación (CACIC)*, 2016.

[4] "LSTM description." https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57 Accessed: 2023-10-20.

[5] "LSTM backpropagation description." https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577. Accessed: 2023-10-20.

[6] J. Schmidhuber, "Annotated History of Modern AI and Deep Learning," tech. rep., Dalle Molle Institute for Artificial Intelligence Research, 2022.