

BÁO CÁO CUỐI KỲ MÔN PHƯƠNG PHÁP SỐ CHO KHDL
Đề tài: Xây dựng mạng neural network nhận dạng
thành viên nhóm qua ảnh

Nhóm 12
Trần Ngọc Dẽ -21110057
Dương Thị Kim Liên – 21110119
Lê Trọng An – 21110235
Nguyễn Duy Đạt – 21110262

Ngày 22 tháng 06 năm 2024

Mục lục

- ① Giới thiệu
- ② Cơ sở lý thuyết
- ③ Xây dựng cấu trúc hệ thống nhận dạng
- ④ Kết quả và phân tích
- ⑤ Kết luận và hướng phát triển

Tầm quan trọng của trí tuệ nhân tạo và mạng neural network:

- Ứng dụng rộng rãi: Trí tuệ nhân tạo (AI) và các mạng neural network (NN) đang ngày càng trở thành công cụ không thể thiếu trong nhiều lĩnh vực công nghệ cao, nhất là trong các hệ thống nhận dạng và xử lý ảnh.
- Nhận dạng khuôn mặt: Công nghệ nhận dạng khuôn mặt đã được ứng dụng rộng rãi từ an ninh, giám sát đến các nền tảng giải trí và mạng xã hội, giúp cải thiện đáng kể độ chính xác và tốc độ xử lý.

Mục tiêu của đề tài: Mục tiêu chính của đề tài là phát triển một mô hình mạng neural network có khả năng nhận diện và phân biệt các thành viên trong một nhóm dựa trên ảnh khuôn mặt.

Cơ sở lý thuyết

Giới thiệu chung về mạng nơ-ron nhân tạo (ANN)

Khái niệm cơ bản:

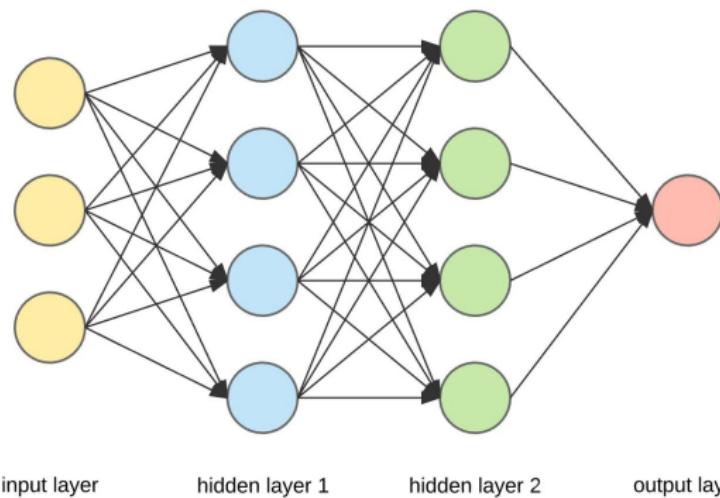
- Mạng nơ-ron nhân tạo (ANN) là một mô hình tính toán được lấy cảm hứng từ hệ thần kinh của não bộ con người. Nó bao gồm các nút, hoặc "nơ-ron", kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó.
- ANN là công cụ chính trong học sâu, giúp máy tính có khả năng học hỏi và đưa ra quyết định dựa trên dữ liệu đầu vào.

Cơ sở lý thuyết

Thành phần cơ bản của ANN

Một mạng nơ-ron thường bao gồm ba lớp: lớp đầu vào (input layer), một hoặc nhiều lớp ẩn (hidden layers), và lớp đầu ra (output layer).

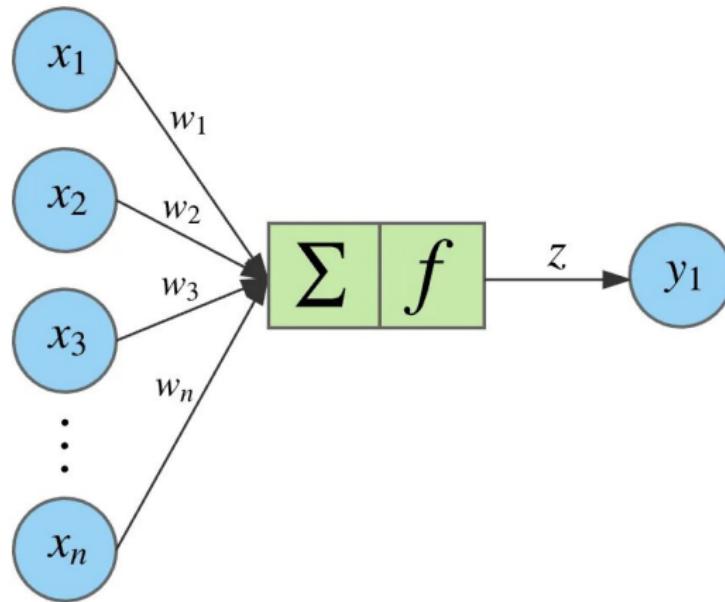
Mỗi nơ-ron trong một lớp nhận đầu vào từ các nơ-ron của lớp trước, xử lý thông tin qua một hàm kích hoạt, và gửi đầu ra đến các nơ-ron của lớp kế tiếp.



Cơ sở lý thuyết

Thành phần cơ bản của ANN

Nếu chúng ta phóng to một trong các nút ẩn hoặc nút đầu ra, chúng ta sẽ thấy như hình bên dưới:



Cơ sở lý thuyết

Thành phần cơ bản của ANN

Phương trình cho một nút cụ thể trông như sau. Tổng có trọng số của các đầu vào của nó được thông qua một hàm kích hoạt phi tuyến. Nó có thể được biểu diễn dưới dạng tích vô hướng vector, trong đó n là số lượng đầu vào cho nút đó.

$$z = f(b + x \cdot w) = f \left(b + \sum_{i=1}^n x_i w_i \right)$$

$$x \in d_{1 \times n}, w \in d_{n \times 1}, b \in d_{1 \times 1}, z \in d_{1 \times 1}$$

Cơ sở lý thuyết

Một số hàm kích hoạt phổ biến

Hàm kích hoạt giúp đưa ra quyết định phi tuyến, làm cho ANNs có thể học được các mô hình phức tạp và không tuyến tính. Dưới đây là một số hàm kích hoạt phổ biến trong các mạng nơ-ron:

- Sigmoid
- Tanh (Hyperbolic Tangent)
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Softmax

Cơ sở lý thuyết

Thuật toán backpropagation

BACKPROPAGATION(*training_examples*, η , n_{in} , n_{out} , n_{hidden})

Each training example is a pair of the form (\vec{x}, \vec{t}) , where \vec{x} is the vector of network input values, and \vec{t} is the vector of target network output values.

η is the learning rate (e.g., .05). n_{in} is the number of network inputs, n_{hidden} the number of units in the hidden layer, and n_{out} the number of output units.

The input from unit i into unit j is denoted x_{ji} , and the weight from unit i to unit j is denoted w_{ji} .

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers (e.g., between -.05 and .05).
- Until the termination condition is met, Do

- For each (\vec{x}, \vec{t}) in *training_examples*, Do

Propagate the input forward through the network:

1. Input the instance \vec{x} to the network and compute the output o_u of every unit u in the network.

Propagate the errors backward through the network:

2. For each network output unit k , calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (\text{T4.3})$$

3. For each hidden unit h , calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{kh} \delta_k \quad (\text{T4.4})$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = \eta \delta_j x_{ji} \quad (\text{T4.5})$$

Cơ sở lý thuyết

Các thuật toán tối ưu

RMSprop Optimizer

Algorithm 2: RMSprop Optimizer

Step 1: Declare the parameters global learning rate ϵ , decay rate ρ , initial parameter θ , small constant δ for numerical stability usually 10^{-6} in the scheme

Step 2: Initialize gradient accumulative variable $r = 0$

Step 3: while stopping criteria not met do

 Step 3.1: Sample a minibatch of m examples from the training set $x^{(i)} \dots x^{(m)} \in X$ with corresponding targets $\{y^{(1)} \dots y^{(i)}\} \in Y$.

 Step 3.2: Using eq.11 compute gradient descent estimate

$$g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \quad (11)$$

 Step 3.3: Accumulate squared gradient using eq.12

$$r \leftarrow r\rho + (1 - \rho)g \odot g \quad (12)$$

 Step 3.4: Compute parameter update as given in eq.13

$$\Delta\theta = -\frac{\epsilon}{\delta + \sqrt{r}} \odot g \quad (13)$$

 Step 3.5: Update objective parameters as shown in eq.14

$$\theta \leftarrow \theta + \Delta\theta \quad (14)$$

end while

Cơ sở lý thuyết

Các thuật toán tối ưu

Adam Optimizer

Algorithm 1: ADAM Optimizer

Step 1: Declare the parameters Objective function $f(\theta)$, stepsize hyperparameter α , exponential decay rates β_1, β_2 for moment estimates, tolerance parameter $\lambda > 0$ for numerical stability and decision rule for declaring convergence of θ_t in the scheme

Step 2: Initialize first moment vector $m_0 \rightarrow \mathbf{0}$, second moment vector $v_0 \rightarrow \mathbf{0}$ and timestep $t \rightarrow 0$

Step 3: while θ_t has not converged do

Step 3.1: update timestep as shown in eq.4

$$t \leftarrow t + 1 \quad (4)$$

Step 3.2: Using eq.5 compute gradient of objective

$$g_t \leftarrow \nabla_{\theta} f_t(\theta_t - 1) \quad (5)$$

Step 3.3: Update first moment estimate and second moment estimate using eq.6 and eq.7

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad (6)$$

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (7)$$

Step 3.4: Using eq.8 and eq.9 compute unbiased first and second moment estimate

$$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t) \quad (8)$$

$$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t) \quad (9)$$

Step 3.5: Update objective parameters using eq.10

$$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \lambda) \quad (10)$$

end while

Step 4: return final parameter θ_t

Xây dựng cấu trúc hệ thống nhận dạng

- Thu thập dữ liệu
- Tiền xử lý dữ liệu
- Xây dựng mô hình
- Huấn luyện và đánh giá

Xây dựng cấu trúc hệ thống nhận dạng

Thu thập dữ liệu

- Nguồn dữ liệu: Thu thập hình ảnh từ các thành viên trong nhóm, mỗi người cung cấp nhiều bức ảnh với các biểu cảm và góc độ khác nhau.
- Đa dạng hóa: Bao gồm cả ảnh từ người lạ để tăng tính chính xác trong việc phân biệt giữa người quen và người lạ.



Hình 7: Minh họa tập dữ liệu ảnh

Xây dựng cấu trúc hệ thống nhận dạng

Tiền xử lý dữ liệu

- Đọc ảnh: Chuyển hình ảnh từ dạng thô sang dữ liệu số có thể xử lý được.
- Chuyển đổi màu sắc: Chuyển hình ảnh màu sang ảnh xám để giảm kích thước dữ liệu mà không ảnh hưởng đến thông tin cần thiết.



Hình 8: Ảnh màu ban đầu



Hình 9: Ảnh xám

Xây dựng cấu trúc hệ thống nhận dạng

Tiền xử lý dữ liệu

- Phát hiện khuôn mặt: Sử dụng thuật toán Haar Cascades để xác định vị trí khuôn mặt trong ảnh.
- Gán nhãn ảnh (Label Images): Gán nhãn cho mỗi hình ảnh để mô hình có thể học và phân biệt các khuôn mặt. Mỗi hình ảnh cần có một nhãn tương ứng để xác định danh tính của người trong ảnh.
- Chuẩn hóa dữ liệu: Điều chỉnh kích thước ảnh về cùng một chuẩn để đảm bảo tính nhất quán khi xử lý.

Xây dựng cấu trúc hệ thống nhận dạng

Xây dựng mô hình

- Mô hình cơ bản: Sử dụng một mạng ANN đơn giản để thử nghiệm cơ bản và hiểu rõ mối quan hệ giữa các đầu vào và đầu ra.
- Cải tiến mô hình: Thêm các lớp ẩn, điều chỉnh các tham số như số lượng nơ-ron, hàm kích hoạt, và thuật toán tối ưu.
- Ứng dụng Batch Normalization và LeakyReLU: Nhằm cải thiện quá trình học và tránh hiện tượng vanishing gradient.
- Chọn Optimizer: So sánh hiệu quả giữa các thuật toán tối ưu như Adam, SGD, và RMSprop.

Xây dựng cấu trúc hệ thống nhận dạng

Huấn luyện và đánh giá

- Chia dữ liệu: Phân chia tập dữ liệu thành tập huấn luyện và tập kiểm thử để đánh giá mô hình một cách khách quan.
- Quá trình huấn luyện: Sử dụng kỹ thuật kiểm định chéo và điều chỉnh các tham số dựa trên độ chính xác và sai số của mô hình.
- Phân tích kết quả: Đánh giá hiệu quả của mô hình thông qua các chỉ số độ chính xác (accuracy) và mất mát (loss).

Kết quả và phân tích

Cơ sở dữ liệu

Dữ liệu đầu vào

- Dataset 1: Nhiệm vụ học tập ở đây liên quan đến việc phân loại hình ảnh khuôn mặt của nhiều người trong các tư thế và biểu cảm khác nhau. Hình ảnh của 4 người trong nhóm đã được thu thập và những người còn lại đặt tên là "unknown". Tổng cộng, 199 hình ảnh sau khi tiền xử lý sang thang độ xám đã được thu thập, mỗi hình ảnh có độ phân giải 100×100 pixel. Giá trị cường độ thang độ xám từ 0 (đen) đến 255 (trắng).
- Dataset 2: Tương tự với dataset 199 ảnh nhưng có sự thay đổi số lượng ảnh. Tổng cộng, 641 hình ảnh thang độ xám, mỗi hình ảnh có độ phân giải 100×100 pixel. Giá trị cường độ thang độ xám từ 0 (đen) đến 255 (trắng).

Kết quả và phân tích

Cơ sở dữ liệu

Dữ liệu đầu vào



Le_Trong_An

Duong_Thi_Kim_Lien

Nguyen_Duy_Dat

Tran_Ngoc_De

unknown

Hình 10: Ảnh xám trong dataset và label được gán

Kết quả và phân tích

Xây dựng các mô hình

Mô hình 1: Basic ANN

```
1 model = Sequential([
2     Dense(512, activation='relu', input_shape=(30000,)),
3     Dense(256, activation='relu'),
4     Dense(128, activation='relu'),
5     Dense(5, activation='softmax')
6 ])
7
8 # Khởi tạo mô hình
9 model.compile(optimizer='adam', loss='categorical_crossentropy',
10                 metrics=['accuracy'])
11
12 # Huấn luyện mô hình
13 model.fit(x_train, y_train, epochs=50, validation_data=(x_test, y_test))
14
15 # Dánh giá mô hình
16 score = model.evaluate(x_test, y_test)
17 print('Test loss:', score[0])
18 print('Test accuracy:', score[1])
```

Kết quả và phân tích

Xây dựng các mô hình

Mô hình 2: ANN with Batch Normalization and LeakyReLU (Adam Optimizer)

```
1 model_adam = Sequential([
2     Dense(512, input_shape=(30000,)),
3     BatchNormalization(),
4     LeakyReLU(alpha=0.1),
5     Dropout(0.5),
6
7     Dense(256),
8     BatchNormalization(),
9     LeakyReLU(alpha=0.1),
10    Dropout(0.5),
11
12    Dense(128),
13    BatchNormalization(),
14    LeakyReLU(alpha=0.1),
15    Dropout(0.5),
16
17    Dense(5, activation='softmax')
18 ])
19
20 # Khởi tạo mô hình
21 optimizer = Adam(learning_rate=0.0001) # Giảm tốc độ học để huấn luyện tốt hơn
22 model_adam.compile(optimizer=optimizer, loss='categorical_crossentropy',
23                      metrics=['accuracy'])
```

Kết quả và phân tích

Xây dựng các mô hình

Mô hình 3: ANN with Batch Normalization and LeakyReLU (SGD Optimizer)

```
1 model_sgd = Sequential([
2     Dense(512, input_shape=(30000,)),
3     BatchNormalization(),
4     LeakyReLU(alpha=0.1),
5     Dropout(0.5),
6
7     Dense(256),
8     BatchNormalization(),
9     LeakyReLU(alpha=0.1),
10    Dropout(0.5),
11
12    Dense(128),
13    BatchNormalization(),
14    LeakyReLU(alpha=0.1),
15    Dropout(0.5),
16
17    Dense(5, activation='softmax')
18 ])
19
20 # Khởi tạo optimizer SGD với momentum
21 optimizer = SGD(learning_rate=0.0001, momentum=0.9)
22 model_sgd.compile(optimizer=optimizer, loss='categorical_crossentropy',
23                     metrics=['accuracy'])
```

Kết quả và phân tích

Xây dựng các mô hình

Mô hình 4: ANN with Batch Normalization and LeakyReLU (RMSprop Optimizer)

```
1 model_rms = Sequential([
2     Dense(512, input_shape=(30000,)),
3     BatchNormalization(),
4     LeakyReLU(alpha=0.1),
5     Dropout(0.5),
6
7     Dense(256),
8     BatchNormalization(),
9     LeakyReLU(alpha=0.1),
10    Dropout(0.5),
11
12    Dense(128),
13    BatchNormalization(),
14    LeakyReLU(alpha=0.1),
15    Dropout(0.5),
16
17    Dense(5, activation='softmax')
18 ])
19
20 # Khởi tạo optimizer RMSprop với momentum
21 optimizer = RMSprop(learning_rate=0.0001, momentum=0.75)
22 model_rms.compile(optimizer=optimizer, loss='categorical_crossentropy',
23                     metrics=['accuracy'])
24
```

Kết quả và phân tích

Kết quả thực nghiệm

Với Dataset 1:

Model	Test Accuracy
Basic ANN (Adam Optimizer)	77.50%
ANN with Batch Normalization and LeakyReLU (Adam)	82.50%
ANN with Batch Normalization and LeakyReLU (SGD)	80.00%
ANN with Batch Normalization and LeakyReLU (RMSprop)	75.00%

Bảng: Comparison of ANN Models with Different Optimizations

Kết quả và phân tích

Kết quả thực nghiệm

Với Dataset 1:

- Mô hình cải tiến với Batch Normalization và LeakyReLU kết hợp với Adam optimizer cho kết quả tốt nhất về độ chính xác.
- Mô hình cơ bản với Adam optimizer cho kết quả trung bình. Mô hình sử dụng SGD optimizer cũng đạt kết quả tốt nhưng không cao bằng mô hình 2.
- Mô hình sử dụng RMSprop optimizer có độ chính xác thấp nhất.

Kết quả và phân tích

Kết quả thực nghiệm

Kết quả chạy mô hình trên tập kiểm thử



4

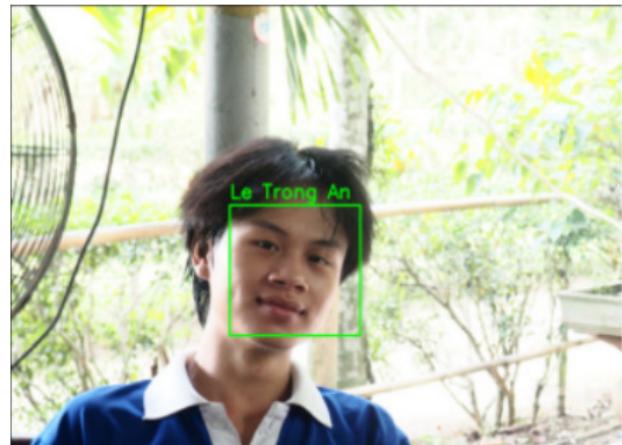
Kết quả và phân tích

Kết quả thực nghiệm

Một số kết quả nhận diện khi chạy mô hình với một ảnh bất kỳ



Hình: Ảnh 1

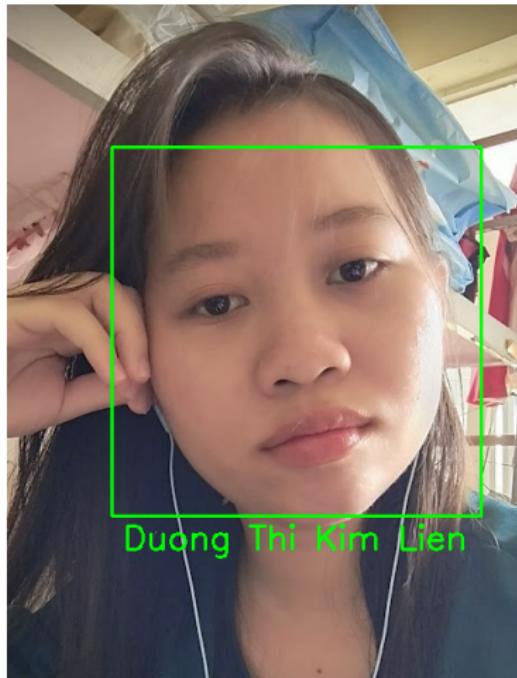


Hình: Ảnh 2

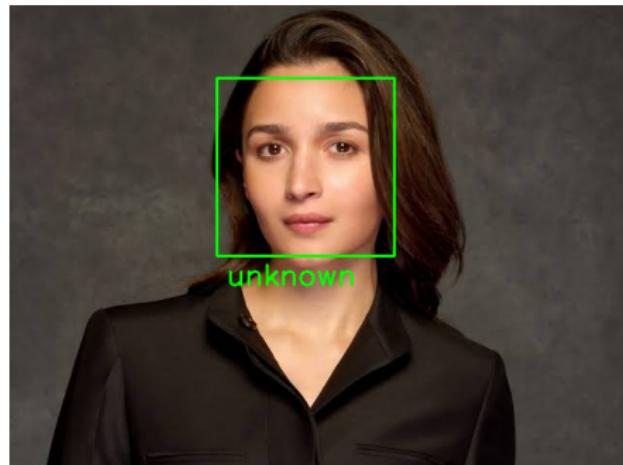
Kết quả và phân tích

Kết quả thực nghiệm

Một số kết quả nhận diện khi chạy mô hình với một ảnh bất kỳ



Hình: Ảnh 3



Hình: Ảnh 4

Kết quả và phân tích

Kết quả thực nghiệm

Với Dataset 2:

Model	Test Accuracy
Basic ANN (Adam Optimizer)	62.79%
ANN with Batch Normalization and LeakyReLU (Adam)	60.47%
ANN with Batch Normalization and LeakyReLU (SGD)	61.24%
ANN with Batch Normalization and LeakyReLU (RMSprop)	60%

Bảng: Comparison of ANN Models with Different Optimizations

Kết quả và phân tích

Kết quả thực nghiệm

Với Dataset 2:

- Mô hình sử dụng Adam Optimizer đạt độ chính xác cao nhất trong số các mô hình được thử nghiệm, đạt 62.79%. Điều này cho thấy Adam Optimizer phù hợp và hiệu quả trong việc tối ưu hóa tham số mô hình.
- Ảnh hưởng của Batch Normalization và LeakyReLU: Các mô hình được cải thiện bằng việc áp dụng Batch Normalization và LeakyReLU không đem lại kết quả tốt hơn so với Basic ANN với Adam Optimizer.
- Sự khác biệt giữa các tối ưu hóa Adam, SGD và RMSprop: Các tối ưu hóa SGD và RMSprop đều cho kết quả tương đối gần nhau, với độ chính xác xấp xỉ 61-62%. Tuy nhiên, chúng không vượt qua được hiệu quả của Adam Optimizer.

Kết luận và hướng phát triển

Kết luận

Đánh giá hiệu quả mô hình nhận diện khuôn mặt:

- Mô hình có khả năng phân biệt và xác định khuôn mặt thành viên trong nhóm.
- Cần tối ưu hóa thêm để giảm sai sót và cải thiện hiệu suất trong điều kiện thực tế.

So sánh tập dữ liệu:

- Tập dữ liệu nhỏ: Có độ chính xác cao hơn.
- Tập dữ liệu lớn: Độ chính xác thấp hơn, do nhiều biến thể và nhiễu hơn.

Cải tiến mô hình

- Batch Normalization và LeakyReLU: Cải thiện đáng kể độ chính xác.
- Adam Optimizer: Hiệu suất tốt trên tập dữ liệu nhỏ (82.50%) nhưng không duy trì ưu thế trên tập dữ liệu lớn.

Kết luận và hướng phát triển

Hướng phát triển

- ① Tăng cường dữ liệu: Tiếp tục thu thập và tăng cường dữ liệu để mô hình có thể học hỏi từ nhiều biến thể khuôn mặt hơn, từ đó cải thiện khả năng tổng quát hóa của mô hình.
- ② Cải thiện cấu trúc mô hình: Thử nghiệm với các kiến trúc mạng sâu hơn và phức tạp hơn, cũng như các kỹ thuật tiên tiến tăng cường khả năng nhận diện.
- ③ Điều chỉnh siêu tham số: Tiến hành điều chỉnh siêu tham số chi tiết hơn để tìm ra các cấu hình tối ưu cho từng loại dữ liệu và từng bài toán cụ thể.

Thank you!