**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN**

**THỰC TẬP TỐT NGHIỆP**

**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG CHẤM CÔNG TỰ ĐỘNG BẰNG NHẬN DIỆN KHUÔN MẶT**

**Gv hướng dẫn: TS. Ngô Đức Vĩnh**

|  |  |
| --- | --- |
| **Họ và tên** | **Mã sinh viên** |
| **Trần Nhữ Vương** | **2017607841** |

**Hà Nội, năm 2022**

**MỤC LỤC**

[**LỜI CẢM ƠN** 4](#_Toc97306987)

[**LỜI MỞ ĐẦU** 5](#_Toc97306988)

[**CHƯƠNG 1: TỔNG QUAN VỀ NHẬN DẠNG** 6](#_Toc97306989)

[**1.1. Đặt vấn đề** 6](#_Toc97306990)

[**1.2. Giới thiệu về nhận dạng** 6](#_Toc97306991)

[**1.3. Quy trình nhận dạng** 6](#_Toc97306992)

[**1.4. Ứng dụng nhận dạng** 6](#_Toc97306993)

[**CHƯƠNG 2: ỨNG DỤNG SVM VÀO BÀI TOÁN NHẬN DẠNG** 7](#_Toc97306994)

[**2.1. SVM là gì ?** 7](#_Toc97306995)

[**2.2. SVM cho 2 lớp dữ liệu** 8](#_Toc97306996)

[**2.4. Ứng dụng trong project** 15](#_Toc97306997)

[**CHƯƠNG 3:TRÍCH XUẤT ĐẶC TRƯNG ẢNH DỰA TRÊN KĨ THUẬT HOG (HISTOGRAM OF ORIENTED GRADIENT)** 17](#_Toc97306998)

[**3.1. Giới thiệu về HOG:** 17](#_Toc97306999)

[**3.2. Thực hiện thuật toán HOG:** 20](#_Toc97307000)

[**3.3. Ứng dụng trong project** 29](#_Toc97307001)

[**CHƯƠNG 4:XÂY DỰNG HỆ THỐNG CHẤM CÔNG TỰ ĐỘNG BẰNG NHẬN DIỆN KHUÔN MẶT** 30](#_Toc97307002)

[**4.1. Mô tả dữ liệu** 30](#_Toc97307003)

[**4.2. Training dữ liệu** 31](#_Toc97307004)

[**4.2. Sử dụng SVM để nhận diện khuôn mặt** 31](#_Toc97307005)

[**4.3. Các giao diện hệ thống** 33](#_Toc97307006)

[**4.3.1. Giao diện Login** 33](#_Toc97307007)

[**4.3.2. Giao diện Hệ thống** 33](#_Toc97307008)

[**4.3.3. Giao diện Quản lý tài khoản** 34](#_Toc97307009)

[**4.3.4. Giao diện Nhận diện** 34](#_Toc97307010)

[**4.3.5. Giao diện Quản lý nhân viên** 35](#_Toc97307011)

[**4.3.6. Giao diện Báo cáo thống kê** 35](#_Toc97307012)

[**CHƯƠNG 5:KẾT LUẬN** 36](#_Toc97307013)

[**TÀI LIỆU THAM KHẢO** 37](#_Toc97307014)

**GHI CHÚ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Tên viết tắt** | **Tên đầy đủ** | **Tên tiếng việt** |
| SVM | Support Vector Machine | Máy vector hỗ trợ |
| HOG | Histogram of oriented gradients | Biểu đồ của gradient định hướng |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| AI | Artificial intelligence | Trí tuệ nhân tạo |
| CNTT |  | Công nghệ thông tin |

# **LỜI CẢM ƠN**

Trước tiên em xin được bày tỏ sự trân trọng và lòng biết ơn đối với thầy giáo TS. Ngô Đức Vĩnh trưởng khoa – Khoa Công nghệ thông tin. Trong suốt thời gian học và làm thực tập tốt nghiệp, thầy đã dành rất nhiều thời gian quý báu để tận tình chỉ bảo, hướng dẫn, định hướng cho em thực hiện thực tập tốt nghiệp.

Em xin được cảm ơn các thầy cô giáo Trường Đại học Công nghiệp Hà Nội đã giảng dạy trong quá trình học tập, thực hành, làm bài tập, giúp em hiểu thấu đáo hơn các nội dung học tập và những hạn chế cần khắc phục trong việc học tập, nghiên cứu và thực hiện bản thực tập tốt nghiệp này.

Em xin cảm ơn các bạn bè và nhất là các thành viên trong gia đình đã tạo mọi điều kiện tốt nhất, động viên, cổ vũ trong suốt quá trình học tập và đồ án tốt nghiệp.

Hà Nội, ngày 25 tháng 3 năm 2022.

Sinh viên

Trần Nhữ Vương

# **LỜI MỞ ĐẦU**

Công nghệ thông tin ngày càng phát triển và có vai trò hết sức quan trọng không thể thiếu trong cuộc sống hiện đại. Con người ngày càng tạo ra những cỗ máy thông minh có khả năng tự nhận biết và xử lí được các công việc một cách tự động, phục vụ cho lợi ích của con người. Trong những năm gần đây, một trong những bài toán nhận được nhiều sự quan tâm và tốn nhiều công sức nhất của lĩnh vực công nghệ thông tin, đó chính là bài toán nhận dạng. Tuy mới xuất hiện chưa lâu nhưng nó đã rất được quan tâm vì tính ứng dụng thực tế của bài toán cũng như sự phức tạp của nó.Bài toán nhận dạng có rất nhiều lĩnh vực như: nhận dạng vất chất(nước, lửa, đất, đá, gỗ..) nhận dạng chữ viết, nhận dạng giọng nói, nhận dang hình dáng, nhận dạng khuôn mặt.. trong đó phổ biến và được ứng dụng nhiều hơn cả là bài toán nhận diện khuôn mặt.Để nhận dạng được khuôn mặt, bước đầu tiên để nhận dạng là phát hiện ra khuôn mặt, điều này thực sự là quan trọng và hết sức khó khăn. Cho đến tận bây giờ, các nhà nghiên cứu vẫn chưa đạt được sự ưng ý trong việc giải quyết các khó khăn của bài toán và cho kết quả hoàn toàn đúng. Tuy nhiên, những gì đã đạt được cũng đủ để chúng ta áp dụng rộng rãi và đem lại những lợi ích to lớn trong cuộc sống.Với sự hấp dẫn của bài toán và những thách thức còn đang ở phía trước, với niềm đam mê công nghệ hiện đại và những ứng dụng thực tế tuyệt với của nó, với khát khao khám phá và chinh phục những chi thức mới mẻ.. tôi đã chọn đề tài nghiên cứu: XÂY DỰNG HỆ THỐNG CHẤM CÔNG TỰ ĐỘNG BẰNG NHẬN DIỆN KHUÔN MẶT làm để tài nghiên cứu cho thực tập tốt nghiệp của mình.

# **CHƯƠNG 1: TỔNG QUAN VỀ NHẬN DẠNG**

## **1.1. Đặt vấn đề**

Ngày nay, với sự phát triển mạnh mẽ của công nghệ đã và đang thúc đẩy nhanh tróng sự phát triển của xã hội. Công nghệ trở thành một phần không thể thiếu trong cuộc sống hằng ngày, nó xuất hiện ở khắp mọi nơi từ trường học, công sở, đến các xí nghiệp hay công ty. Không những thế nó còn thực hiện được nhiều nhiệp vụ đơn giản cũng như phức tạp để phục vụ lợi ích của con người.

Gần đây nhất là sự trỗi dậy thuật ngữ **AI (Artificial intelligence)**. Nó **có thể được định nghĩa như một ngành của khoa học máy tính liên quan đến việc tự động hóa các hành vi thông minh giống con người, được xây dựng trên nền tảng lý thuyết chắc chắn, có ứng dụng thực tiễn cao. Và ngày nay nó đã được ứng dụng khá nhiều vào lĩnh vực như IoT(Internet of Things), Đồ họa, Nhận dạng, Machine Learning, Y học...**

**Có thể thấy rằng việc áp dụng công nghệ vào đời sống đã và đang mang lại lợi ích vô cùng lớn. Và trong khuôn khổ nội dung của báo cáo, em xin nó đến một công nghệ đang được áp dụng rất nhiều trong thực tế đó là “Nhận dạng”.**

## **1.2. Giới thiệu về nhận dạng**

Nhận dạng mẫu là khả [1] năng phát hiện sự sắp xếp các của đặc điểm hoặc dữ liệu mang lại thông tin về một hệ thống hoặc tập dữ liệu nhất định. Trong lĩnh vực công nghệ, một mẫu có thể là chuỗi dữ liệu lặp lại theo thời gian có thể sử dụng để dự đoán xu hướng, xác định các đối tượng cụ thể trong ảnh, sự xuất hiện của các từ hoặc cụm từ để xử lý ngôn ngữ tự nhiên (NLP).

## **1.3. Quy trình nhận dạng**

## **1.4. Ứng dụng nhận dạng**

Nhận dạng là điều cần thiết đối với lĩnh vực CNTT, bao gồm phân tích dữ liệu, nhận dạng sinh trắc học, bảo mật và trí tuệ nhân tạo.

Một số ứng dụng cụ thể:

Phần mềm nhận diện khuôn mặt để quản lý nhân lực.

Phầm mềm nhận diện biển số xe để quản lý xe.

Và còn rất nhiều ứng dụng nữa mà chúng ta có thể áp dụng: kĩ thuật làm phim 3D, nhận dạng trong y học để phát hiện bệnh,...

# **CHƯƠNG 2: ỨNG DỤNG SVM VÀO BÀI TOÁN NHẬN DẠNG**

## **2.1. SVM là gì ?**

SVM (**Support Vector Machine** ) là một [2] thuật toán thuộc nhóm [Supervised Learning](https://1upnote.me/tags/supervised-learning/) (Học có giám sát) dùng để phân chia dữ liệu (Classification) thành các nhóm riêng biệt.

Support Vector Machines (có tài liệu dịch là Máy véctơ hỗ trợ) là một trong số những thuật toán phổ biến và được sử [3] dụng nhiều nhất trong học máy trước khi mạng nơ ron nhân tạo trở lại với các mô hình deep learning. Nó được biết đến rộng rãi ngay từ khi mới được phát triển vào những năm 1990.

Mục tiêu của SVM là tìm ra [3] một siêu phẳng trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Nói theo ngôn ngữ của đại số tuyển tính, siêu phẳng này phải có lề cực đại và phân chia hai bao lồi và cách đều chúng.

Trước khi đi sâu vào tìm hiểu thuật tuấn SVM, chúng ta xem lại một số kiến thức về hình học giải tích:

* Trong không gian 2 chiều, ta biết [4] rằng khoảng cách từ một điểm có toạ độ **(x0,y0)** tới đường thẳng có phương trình **w1x+w2y+b=0** được xác định bởi:

Trong không gian 3 chiều, ta biết [4] rằng khoảng cách từ một điểm có toạ độ **(x0,y0,z0)** tới đường thẳng có phương trình **w1x+w2y+w3z+b=0** được xác định bởi:

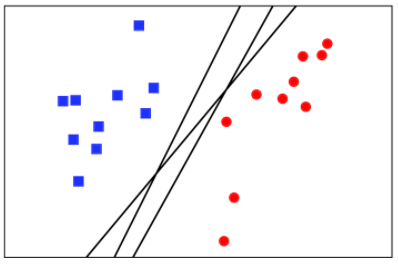
Hơn nữa, nếu ta bỏ dấu trị tuyệt đối ở tử số, chúng ta có thể xác định được điểm đó nằm về phía nào của đường thẳng hay mặt phẳng đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối [4] mang dấu dương nằm về cùng 1 phía (tạm gọi đây là phía dương của đường thẳng), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (gọi là phía âm). Những điểm nằm trên đường thẳng/măt phẳng sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.

Việc này có thể được [4] tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có toạ độ  tới siêu mặt phẳng (hyperplane) có phương trình  được xác định bởi:

Trong đó: với d là số chiều của không gian.

## **2.2. SVM cho 2 lớp dữ liệu**

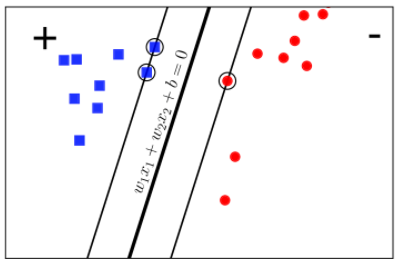
Với 2 lớp dữ liệu, được coi là linearly separable, tức là tồn tại một siêu phẳng phân chia chính xác hai class đó. Việc tìm tìm siêu phẳng phân chia hai class, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại. Chúng ta đã biết rằng, thuật toán PLA có thể làm được việc này nhưng nó có thể cho chúng ta vô số nghiệm như dưới đây:



Hình 2. 1 Các mặt phân cách hai classes linearly separable

Dựa vào hình trên ta thấy rằng có rất nhiều mặt phẳng phân chia , vậy đâu là mặt phẳng tối ưu nhất hay làm cho các lớp dữ liệu cách xa mặt phẳng đó nhất. Trước khi tìm được mặt phẳng ta cùng tìm hiểu khái niệm: margin, soft margin, hard margin.

**Margin:** khoảng cách giữa siêu [2] phẳng (trong trường hợp không gian 2 chiều là đường thẳng) đến 2 điểm dữ liệu gần nhất tương ứng với 2 lớp.



Hình 2. 2 Đường thẳng được chọn làm mặt phân cách

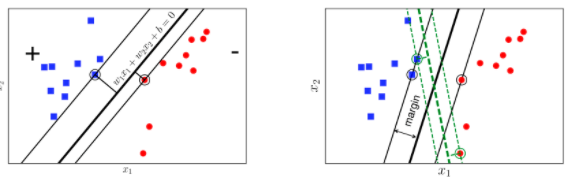
SVM cố gắng tối ưu thuật toán bằng [2] các tìm cách maximize giá trị margin này, từ đó tìm ra siêu phẳng **đẹp nhất** để phân 2 lớp dữ liệu.

Bài toán tìm Margin cực đại là một **Quadratic Programming**, được giải bằng cách giải bài toán đối ngẫu Lagrange (Lagrange dual problem).’

Cách tính Margin:

* Trong không gian 2 chiều:
* Trong không gian nhiều chiều:

Xét hai hình sau:

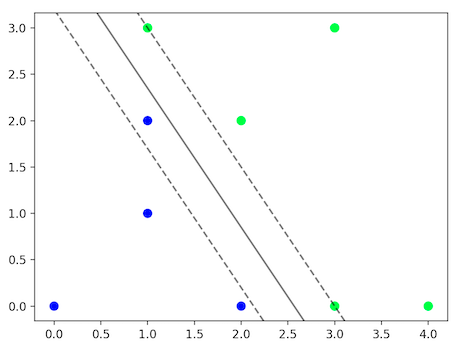


Hình 2. 3 Margin của hai class là bằng nhau và lớn nhất có thể

Nếu ta định nghĩa mức độ hạnh phúc của một class tỉ lệ thuận với khoảng cách gần nhất từ một điểm của class đó tới đường/mặt phân chia,[4] thì ở hình bên trái, class tròn đỏ sẽ không được hạnh phúc cho lắm vì đường phân chia gần nó hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như thế thì mới công bằng. Khoảng cách như nhau này được gọi là margin (lề).

Chúng ta xét tiếp hình bên phải khi khoảng [4] cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, đường nào sẽ làm cho cả hai class hạnh phúc hơn? Rõ ràng đó phải là đường nét liền màu đen vì nó tạo ra một margin rộng hơn.

**Soft margin và hard margin:** để tránh **overfitting**, nhiều khi để muốn có margin cao, ta chấp nhận việc một vài data có thể không được chia chính xác (ví dụ như 1 bóng xanh bị lọt sang vùng của bóng đỏ). Data này được gọi là nhiễu.



Hình 2. 4 Soft margin

Margin trong trường hợp này gọi là **Soft Margin**.  
**Hard Margin** ám chỉ việc tìm được Margin mà không nhiễu (tất cả các data đều thoả mãn sự phân chia).

Với các bái toán thực tế, việc tìm được **Hard Margin** nhiều khi là bất khả thi, vì thế việc chấp nhận sai lệch ở một mức độ chấp nhận được là vô cùng cần thiết.

Trong cài đặt SVM, người ta giới thiệu tham số C*C* với quy ước:

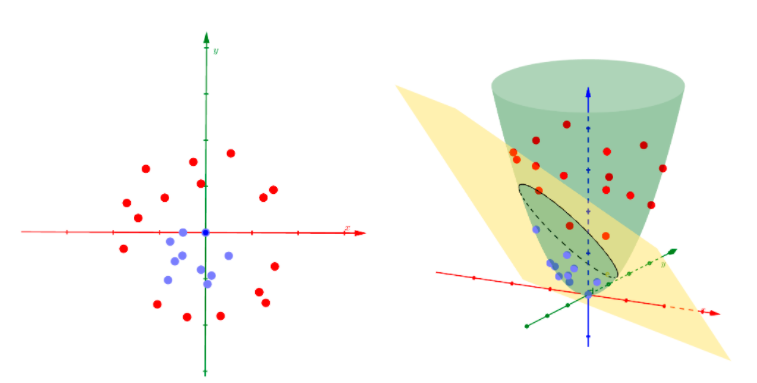
* ***C* = ∞**: không cho phép sai lệch, đồng nghĩa với **Hard Margin**.
* **C lớn**: cho phép sai lệch nhỏ, thu được Margin nhỏ.
* **C nhỏ**: cho phép sai lệch lớn, thu được Margin lớn.

Tuỳ bài toán cụ thể mà ta cần **điểu chỉnh tham số *C* này để thu được kết quả tốt nhất**.

Việc margin***rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn*** [4]vìsự phân chia giữa hai class là rạch ròi hơn*.* Nhưng trên thực tế bộ dữ liệu dữ liệu của chúng ta phức tạp hơn rất nhiều. Để tìm hiểu rõ hơn chúng ta tìm hiểu ở phần sau.

**2.3. Kernel SVM**

Phần trên em đã tìm [5] hiểu SVM trong việc phân chia cho 2 lớp dữ liệu, nhưng trong thực tế lượng dữ liệu của ta lớn hơn rất nhiều vì vậy việc phân chia là vô cùng khó khăn. Nói cách khác dữ liệu là dữ liệu không tuyến tính.



Hình 2. 5Dữ liệu không tuyến tính được ánh xạ sang một không gian khác

Hình bên trái dữ liệu của hai [5] classes là không phân biệt tuyến tính trong không gian hai chiều.

Hình bên phải nếu coi thêm [5] chiều thứ ba là một hàm số của hai chiều còn lại z=x2+y2, các điểm dữ liệu sẽ được phân bố trên 1 parabolic và đã trở nên phân biệt tuyến tính . Mặt phẳng màu vàng là mặt phân chia, có thể tìm được bởi Hard/Soft Margin SVM.

Nói một cách ngắn gọn, Kernel SVM là việc đi tìm một hàm số biến đổi dữ liệu xx từ không gian feature ban đầu thành dữ liệu trong một không gian mới bằng hàm số Φ(x). Trong ví dụ này, hàm Φ() đơn giản là giới thiệu thêm một chiều dữ liệu mới (một feature mới) là một hàm số của các features đã biết. [5] Hàm số này cần thỏa mãn mục đích của chúng ta: trong không gian mới, dữ liệu giữa hai classes là phân biệt tuyến tính hoặc gần như phần biệt tuyến tính. Khi đó, ta có thể dùng các bộ phân lớp tuyến tính thông thường như PLA, Logistic Regression, hay Hard/Soft Margin SVM.

Nếu phải so sánh, ta có thể thấy rằng hàm biến đổi Φ() tương tự như activation functions trong Neural Networks. [5] Tuy nhiên, có một điểm khác biệt ở đây là: trong khi nhiệm vụ của activation function là phá vỡ tính tuyến tính của mô hình, hàm biến đổi Φ() đi biến dữ liệu không phân biệt tuyến tính thành phân biệt tuyến tính. Như vậy là để đạt được mục đích chung, ta có hai cách nhìn khác nhau về cách giải quyết.

Các hàm Φ() thường tạo ra dữ liệu mới có [5] số chiều cao hơn số chiều của dữ liệu ban đầu, thậm chí là vô hạn chiều. Nếu tính toán các hàm này trực tiếp, chắc chắn chúng ta sẽ gặp các vấn đề về bộ nhớ và hiệu năng tính toán. Có một cách tiếp cận là sử dụng các kernel functions mô tả quan hệ giữa hai điểm dữ liệu bất kỳ trong không gian mới, thay vì đi tính toán trực tiếp từng điểm dữ liệu trong không gian mới. Kỹ thuật này được xây dựng dựa trên quan sát về bài toán đối ngẫu của SVM.

**Cơ sở toán học:**

λ=arg subject to:

Trong đó:

* N: số cặp điểm dữ liệu trong tập training.
* xn: feature vector của dữ liệu thứ nn trong tập training.
* yn: nhãn của dữ liệu thứ n, bằng 1 hoặc -1.
* λn: nhân tử Lagrange ứng với điểm dữ liệu thứ n.
* C: hằng số dương giúp cân đối độ lớn của margin và sự hy sinh của các điểm nằm trong vùng không an toàn. Khi C=∞ hoặc rất lớn, Soft Margin SVM trở thành Hard Margin SVM.

Sau khi giải được λ cho bài toán (1), nhãn của một điểm dữ liệu mới sẽ được xác định bởi dấu của biểu thức:

Trong đó:

* ={n:0<λn<C} là tập hợp những điểm nằm trên margin.
* S={n:0<λn} là tập hợp các điểm support.
* là số phần tử của .

Với dữ liệu thực tế, rất khó để có dữ liệu gần phân biệt tuyến tính, [5] vì vậy nghiệm của bài toán (1) có thể không thực sự tạo ra một bộ phân lớp tốt. Giả sử rằng ta có thể tìm được hàm số Φ() sao cho sau khi được biến đổi sang không gian mới, mỗi điểm dữ liệu xx trở thành Φ(x), và trong không gian mới này, dữ liệu trở nên gần phân biệt tuyến tính. Lúc này, hy vọng rằng nghiệm của bài toán Soft Margin SVM sẽ cho chúng ta một bộ phân lớp tốt hơn.

Trong không gian mới, bài toán (1) trở thành:

subject to:

và nhãn của một điểm dữ liệu mới được xác định bởi dấu của biểu thức:

Như đã nói ở [5] trên, việc tính toán trực tiếp Φ(x) cho mỗi điểm dữ liệu có thể sẽ tốn rất nhiều bộ nhớ và thời gian vì số chiều của Φ(x) thường là rất lớn, có thể là vô hạn! Thêm nữa, để tìm nhãn của một điểm dữ liệu mới x, ta lại phải tìm biến đổi của nó Φ(x) trong không gian mới rồi lấy tích vô hướng của nó với tất cả các Φ() với m trong tập hợp support. Để tránh việc này, ta quan sát thấy một điều thú vị sau đây.

Trong bài toán (3) và biểu thức (4), chúng ta không cần tính trực tiếp Φ(x) cho mọi điểm dữ liệu. Chúng ta chỉ cần tính được Φ(x)TΦ(z) dựa trên hai điểm dữ liệu x,z bất kỳ. Kỹ thuật này còn được gọi là **kernel trick**. Những phương [5] pháp dựa trên kỹ thuật này, tức thay vì trực tiếp tính tọa độ của một điểm trong không gian mới, ta đi tính tích vô hướng giữa hai điểm trong không gian mới, được gọi chung là **kernel method**.

Lúc này, bằng cách định nghĩa hàm kernel k(x,z)=Φ(x)TΦ(z), ta có thể viết lại bài toán (3) và biểu thức (4) như sau:

subject to:

và:

Không phải hàm k() bất kỳ nào cũng được [5] sử dụng. Các hàm kerrnel cần có các tính chất:

* Đối xứng: k(x,z)=k(z,x). Điều này dễ nhận ra vì tích vô hướng của hai vector có tính đối xứng.
* Về lý thuyết, hàm kerrnel cần thỏa mãn điều kiện Mercer:

Tính chất này để đảm bảo cho việc hàm mục tiêu của bài toán đối ngẫu (5) là lồi.

Nếu một hàm kerrnel thỏa mãn điều kiện (7), xét cn=ynλn, ta sẽ có:

λTKλ=N∑n=1N∑m=1k(xm,xn)ynymλnλm≥0, ∀λn(8)

với K là một ma trận đối xứng mà phần tử ở hàng thứ nn cột thứ mm của nó được định nghĩa bởi: knm=ynym

Từ (8) ta suy ra K là một ma trận nửa xác định dương. Vì vậy, bài toán tối ưu (5) có ràng buộc là lồi và hàm mục tiêu là một hàm lồi (một quadratic form). Vì vậy chúng ta có thể giải quyết bài toán này một cách hiệu quả.

Trong bài viết này, em sẽ không đi sâu vào việc giải quyết bài toán (5) vì nó hoàn toàn tương tự như bài toán đối ngẫu của [5] Soft Margin SVM. Thay vào đó, em sẽ trình bày các hàm **kernel** thông dụng và hiệu năng của chúng trong các bài toán thực tế. Việc này sẽ được thực hiện thông qua các ví dụ và cách sử dụng thư viện sklearn.

**Một số hàm kernel thông dụng:**

* Linear
* Polynomial
* Radial Basic Function (RBF)
* Sigmoid

Trong phạm vi báo cáo em sử dụng hàm **Polynomial:**

k(x,z)=(r+γxTz)d

Với d là một số dương để chỉ bậc của đa thức. d có thể không là số tự nhiên vì mục đích chính của ta không phải là bậc của đa thức mà là cách tính kernel. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.

**2.4. Ứng dụng trong project**

Hiện tại em đang sử dụng thư viện **sklearn** để giải quyết bài toán phân loại với SVM: **sklearn.svm.svc**

 def \_\_init\_\_(self):

        self.C = 100

        self.gamma = 'scale'

        self.kernel = 'poly'

        self.degree = 3

    def BuiltModel(self, XTrain,YTrain):  # xây dựng thuật toán SVM

        model = SVC(kernel=self.kernel, degree= self.degree, gamma=self.gamma, C=self.C)

        model.fit(XTrain, YTrain)

        return model

Dựa vào mã giải ta thấy rằng, ban đầu ta cần khai báo một vài biến cho thuật toán phân loại (C, gamma, kermel, degree,...) trong hàm **\_\_init()\_\_.**

Sau đó chúng ta xây dựng hàm BuildModel(), sẽ nhận đầu vào là các biến đã khai báo ở hàm \_\_init()\_\_ và hai biến nữa là Xtrain, y\_train, hai biến này ta thu được sau khi trích xuất đặc trưng và lưu vào file có tên là “\*.npz” là một mảng hai chiều đặt tên lần lượt là arr\_0 và arr\_1 do chương trình tự định nghĩa nên nó cũng lần lượt tương ứng với X\_train và y\_train.

**Giải thích một vài thông số:**

|  |  |
| --- | --- |
| **C** | Tham số điều chỉnh. Độ mạnh của quy luật tỷ lệ nghịch với C. |
| **Gamma** | Hệ số nhân cho các kernel ‘rbf’, ‘poly’ and ‘sigmoid’. Mặc định là “scale”, ngoài ra còn có “auto” |
| **Kernel** | Chỉ định loại hạt nhân sẽ được sử dụng trong thuật toán. Nó phải là một trong các giá trị ‘tuyến tính’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ hoặc một trong các nhân khác. Mặc định sẽ là “rbf”. |
| **Degree** | Bậc của hàm nhân đa thức (‘poly’), bị bỏ qua ở các kernel khác. Mặc định bằng 3 |

# **CHƯƠNG 3:TRÍCH XUẤT ĐẶC TRƯNG ẢNH DỰA TRÊN KĨ THUẬT HOG (HISTOGRAM OF ORIENTED GRADIENT)**

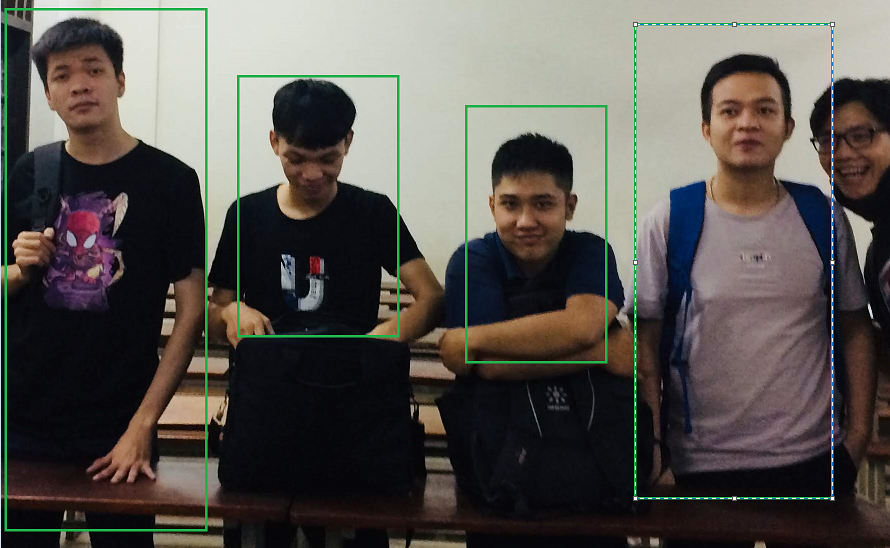
## **3.1. Giới thiệu về HOG:**

*a. Giới thiệu chung:*

HOG là viết tắt của Histogram of Oriented Gradient - một loại “feature descriptor”. Mục đích của “featura descriptor” là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.

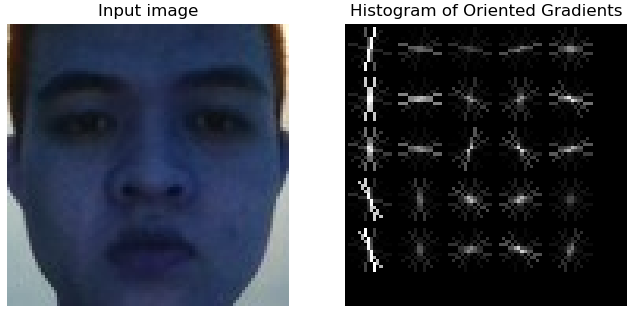
*b. Ứng dụng HOG:*

Nhận diện người (human detection): Lần đầu tiên ứng dụng này được giới thiệu trong bài báo Histograms of Oriented Gradients for Human Detection của Dalal và Trigg[6]. HOG có thể phát hiện được một hoặc nhiều người đi bộ trên cùng một hình ảnh.



Hình 3. 1 Khả năng phát hiện đối tượng của HOG

Nhận diện khuôn mặt (face detection): Thường chúng ta sẽ nghĩ ngay đến thuật toán Haar Cascde Classifier. Tuy nhiên HOG cũng là một thuật toán rất hiệu quả được áp dụng trong bài toán này. Bởi nó có khả năng biểu diễn các đường nét chính của khuôn mặt dựa trên phương và độ lớn gradient thông qua các véc tơ trên mỗi cell như hình mô tả bên dưới:



Hình 3. 2 Biểu diễn đường nét chính của khuôn mặt

Nhận diện các vật thể khác: Ngoài ra còn rất nhiều các trường hợp nhận diện vật thể trên ảnh tĩnh như phương tiện, tín hiệu giao thông, động vật hoặc thậm chỉ là ảnh động từ video.

Tạo feature cho các bài toán phân loại ảnh***:*** Nhiều bài toán phân loại ảnh được xây dựng trên một bộ dữ liệu kích thước nhỏ thì sử dụng các mạng học sâu chưa chắc đã mang lại hiệu quả và dễ dẫn tới overfiting. Nguyên nhân vì dữ liệu ít thường không đủ để huấn luyện cho máy tính nhận tốt các đặc trưng của vật thể. Khi đó sử dụng HOG để tạo đặc trưng sẽ mang lại kết quả tốt hơn. Cụ thể em cũng sẽ thực hiện một ví dụ ở cuối.

*c. Một số thuật ngữ:*

Feature Descriptor: Bộ mô tả đặc trưng, là một phép biến đổi dữ liệu thành các đặc trưng giúp ích cho phân loại hoặc nhận diện vật thể. Các phương pháp có thể kể đến như HOG, SUFT, SHIFT.

Histogram: Là biểu đồ histogram biểu diễn phân phối của các cường độ màu sắc theo khoảng giá trị.

Gradient: Là đạo hàm của véc tơ cường độ màu sắc giúp phát hiện hướng di chuyển của các vật thể trong hình ảnh.

Local cell: Ô cục bộ. Trong thuật toán HOG, một hình ảnh được chia thành nhiều cells bởi một lưới ô vuông. Mỗi cell được gọi là một ô cục bộ.

Local portion: Vùng cục bộ. Là một vùng trước trích suất ra từ ô vuông trên hình ảnh. Trong phần trình bày về thuật toán thì vùng cục bộ còn được gọi là block.

Local normalization: Phép chuẩn hóa được thực hiện trên một vùng cục bộ. Thường là chia cho norm chuẩn bậc 2 hoặc norm chuẩn bậc 1. Mục đích của việc chuẩn hóa là để đồng nhất các giá trị cường độ màu sắc về chung một phân phối. Ta sẽ làm rõ hơn trong phần trình bày thuật toán.

Gradient direction: Phương gradient. Là độ lớn góc giữa véc tơ gradient x và y giúp xác định phương thay đổi cường độ màu sắc hay chính là phương đổ bóng của hình ảnh. Giả sử Gx,Gy lần lượt là giá trị gradient theo lần lượt phương x và y của hình ảnh. Khi đó phương gradient được tính như sau:

Gradient magnitude: Độ lớn gradient. Là chiều dài của véc tơ gradient theo phương x và phương y. Biểu diễn phân phối histogram của véc tơ này theo véc tơ phương gradient sẽ thu được véc tơ mô tả đặc trưng HOG. Độ lớn gradient được tính như sau:

## **3.2. Thực hiện thuật toán HOG:**

Có 5 bước để thực hiện thuật toán HOG:

- Tiền xử lý

- Tính gradient

- Tính vector đặc trưng cho từng ô (cells)

- Chuẩn hóa khối (blocks)

- Tính toán vector HOG

*a. Tiền xử lý:*

* Để ảnh có thể trích xuất ra đặc trưng thường có kích thước cơ bản là 64x128 thì sẽ cho ra kết quả cao nhất.

Có thể để các ảnh ở các dạng khác nhau như :màu,xám,trắng đen…

* Đầu tiên chúng ta cần đưa hình ảnh về kích thước 64x128

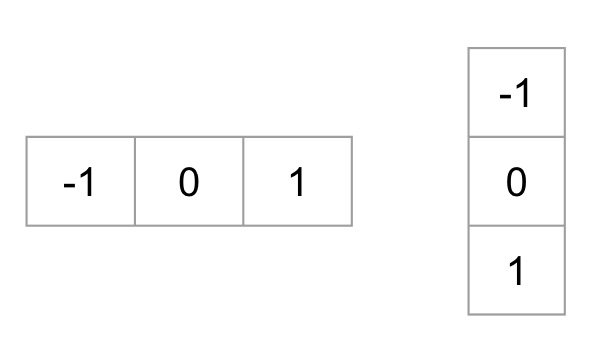


*b. Tính Gradient*

Đây là bước đầu tiên, được thực hiện bằng hai phép nhân chập ảnh gốc với 2 chiều, tương ứng với các toán tử lấy đạo hàm theo hai hướng Ox và Oy. Trong đó, 2 hướng tương ứng đó là:



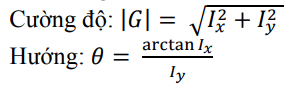
T là phép toán chuyển vị ma trận.

**

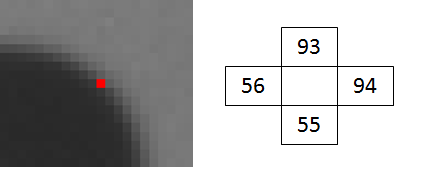
Và nếu bạn có một ảnh input là I, ta sẽ có 2 ảnh đạo hàm riêng theo 2 hướng đó, theo công thức:

https://images.viblo.asia/636c5c91-78ce-4639-9b85-01bacabe1a8c.png

Khi đó, bạn có thể tính được Gradient bao gồm hai thành phần cường độ (Gradient Magnitude) và hướng(Gradient Derection) theo công thức:

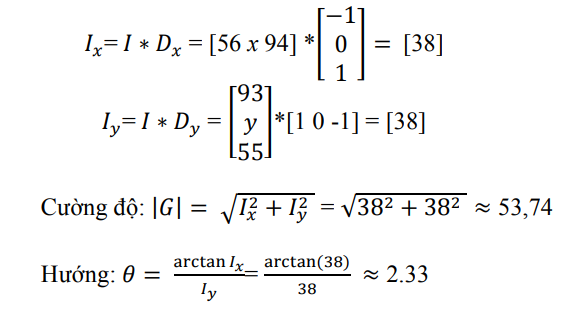


Ví dụ: Giả sử ta có một điểm ảnh như sau:



Hình 3. 5 Ví dụ tính gradient cho từng điểm ảnh

Chúng ta sẽ áp dụng các công thức trên để tính được gradient của điểm ảnh này:

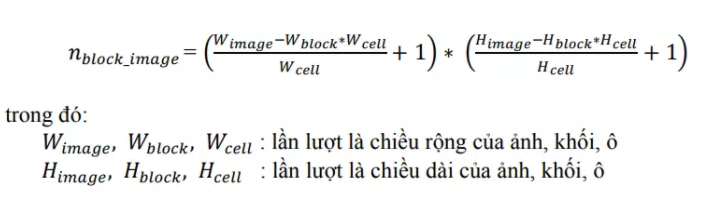


Đối với hình ảnh màu, gradient của ba kênh(red, green và blue) được đánh giá. Độ lớn của gradient tại một điểm ảnh là giá trị lớn nhất của cường độ gradient của ba kênh, và góc là góc tương ứng với gradient tối đa.

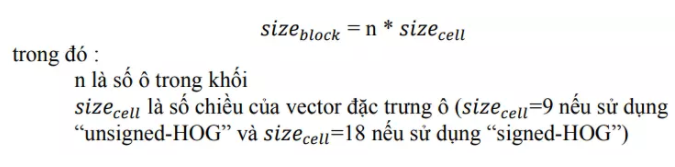
*c. Tính vector đặc trưng cho từng ô (cells):*

Giờ đây ta có thêm một khái niệm mới chen chân vô đó là **cell** (dịch: ô). Một cell được thiết kế là có kích thước 8x8 pixel (đây là siêu tham số, tác giả có tùy chỉnh và chọn 8 là giá trị hợp lý qua các thí nghiệm). Do đó ảnh đầu 64x128, ta sẽ chia mỗi hình ảnh thành các block có kích thước 16x16. Mỗi block sẽ bao gồm 4 cell, mỗi cell có kích thước là 8x8.

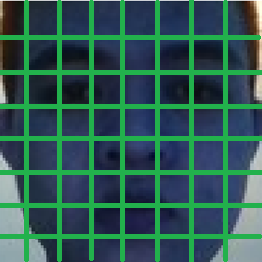
Để tính toán vector đặc trưng cho từng ô (cell), chúng ta cần chia hình ảnh thành các block, mỗi block lại chia đều thành các cell. Để xác định được số block, chúng ta sẽ sử dụng công thức sau:



Sau khi tính toán đặc trưng ô, ta sẽ nối các vector đặc trưng ô để thu được vector đặc trưng khối. Số chiều vector đặc trưng khối tính theo công thức :



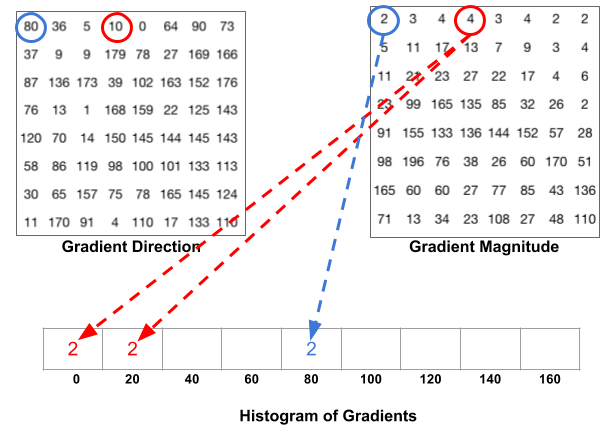
Ví dụ:



Tại mỗi cell, xây dựng một biểu đồ cường độ gradient bằng cách vote các pixel vào biểu đồ. Trọng số vote của mỗi pixel phụ thuộc hướng và cường độ gradient (được tính toán từ bước 2) của pixel đó.

|  |  |
| --- | --- |
|  |  |

Hình 3. Tính độ lớn và phương của gradient



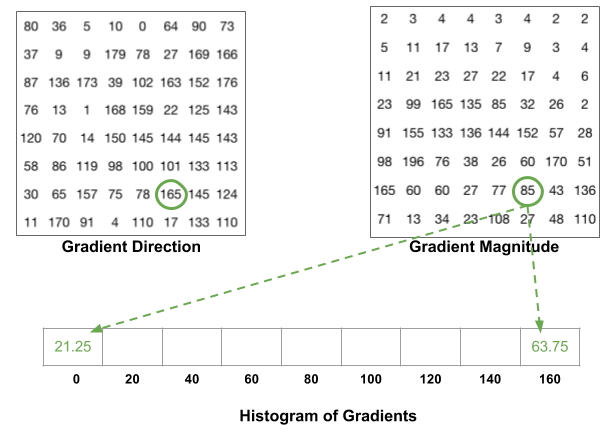
Hình 3. Tiến hành vote vào các đoạn

Đầu mút là các giá trị chia hết cho độ rộng của một bin (chẳng hạn 0, 20, 40,… là những đầu mút bin). Trong trường hợp độ lớn phương gradients không rơi vào các đầu mút, ta sẽ sử dụng linear interpolation để phân chia độ lớn gradient về 2 bins liền kề mà giá trị phương gradient rơi vào.

Ví dụ: giá trị phương gradient bằng x ghép cặp với độ lớn gradient bằng y. x∈[x0,x1] tức là phương gradients rơi vào khoảng giữa bin thứ (l−1) và bin thứ l: . Khi đó tại 2 bins (l−1) và l được điền vào giá trị cường độ theo công thức interpolation:

Gía trị tại bins l−1:

Gía trị tại bins l:

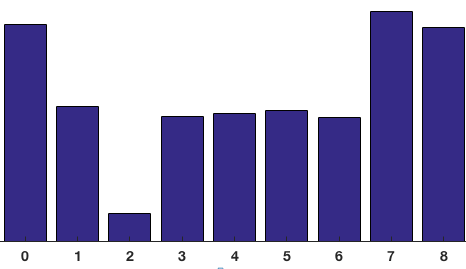


Ví dụ: Với điểm được khoanh tròn bởi hình tròn màu xanh có phương gradient bằng 165 và độ lớn gradient bằng 85. Ta phân chia giá trị về các bins 0 (hoặc 180) và 160 các giá trị theo công thức interpolation bên trên.

*Giá trị bin 8: [(180 – 165) / (180 – 160)] \* 85 = 63.75*

*Giá trị bin 0: [(165 – 160) / (180 – 160)] \* 85 = 21.25*

Kết quả cuối cùng chúng ta thu được là:



Hình 3. Biểu đồ Histogram of Gradient sau khi vote hết các pixel trong một cell kích thước 8x8 vào 9 bin

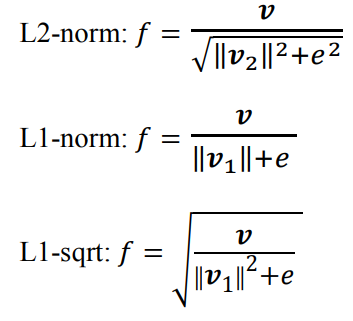
*d. Chuẩn hóa khối (Blocks):*

Đến đây ta cần biết 1 block là gì. Một block gồm nhiều cell, block 2x2 nghĩa là ta có vùng diện tích của 4 cell liền kề –> block này sẽ phủ trên diện tích = 16x16 pixel. Trong quá trình chuẩn hóa, ta sẽ lần lượt chuẩn hóa block 2x2 đầu tiên, rồi dịch block đó sang 1 cell và cũng thực hiện chuẩn hóa cho block này. Như vậy, giữa block đầu tiên và block liền kề đã có sự chồng lấn cell lẫn nhau (2 cell), trong tiếng Anh người ta dùng từ overlap.



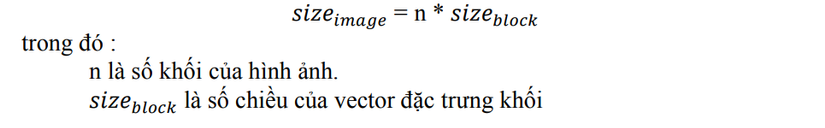
Hình 3. Kích thước của một block (2x2)

Có nhiều phương pháp có thể được dùng để chuẩn hóa khối. Gọi v là vector cần chuẩn hóa chứa tất cả các histogram của mội khối. ‖v(k)‖ là giá trị chuẩn hóa của v theo các chuẩn k=1, 3 và e là một hằng số nhỏ. Khi đó, các giá trị chuẩn hóa có thể tính bằng một trong những công thức sau:



e. Tính toán vector đặc trưng HOG:

Ghép các vector đặc trưng khối sẽ thu được vector đặc trưng R-HOG cho ảnh. Số chiều vector đặc trưng ảnh tính theo công thức :



Ví dụ:



* Với mỗi hình ảnh kích thước 64x128, chia thành các block 16x16 chồng nhau, sẽ có 7 block ngang và 15 block dọc, nên sẽ có 7x15 = 105 blocks.
* Mỗi block gồm 4 cell. Khi áp dụng biểu đồ 9-bin cho mỗi cell, mỗi block sẽ được đại diện bởi một vector có kích thước 36x1.
* Vì vậy, khi nối tất cả các vector trong một block lại với nhau, ta sẽ thu được vector đặc trưng HOG của ảnh có kích thước 105x36x1 = 3780x1.

## **3.3. Ứng dụng trong project**

Em sử dụng thư viện sklearn để làm việc với thuật toán HOG, chi tiết: **sklearn.feature.hog**

**\*Đây là class viết để trích xuất đặc trưng ảnh trong tập ảnh train.**

def \_\_init\_\_(self):

        self.SIZE = (64,128)

        self.SOCHIEU = 3780

        self.pixels\_per\_cell = (8,8)

        self.cells\_per\_block = (2,2)

    '''xử lý tất cả các ảnh trong thư mực chứa ảnh train hoặc test'''

    def ManyFaceFeatures(self,derectory):

        #faces = list()

        try:

            path = list(paths.list\_images(derectory))

            print("Dataset: "+str(len(path))+" images")

            ''' tạo một vector 0 để sau đó gán các đặc trừng khuôn mặt vào vector x này. làm như này sẽ

                ta có được 1 vector đặc trưng phù hợp với yêu cầu đầu vào của model svm mà k cần reshape '''

            x = np.zeros((self.SOCHIEU, len(path))) #tạo một mảng 2 chiều với giá trị 0 với kích thước 3780 x length

            count = 0

            print(len(path))

            for i in range(len(path)):

                print(i)

                try:

                    img = cv2.imread(path[i])

                    box = face\_locations(img) # tìm tất cả các khuôn mặt có trong ảnh

                    if len(box) == 0:

                       continue

                    else:

                        top, right, bottom, left = box[0]

                        face = img[top:bottom, left:right]

                        face\_resize = cv2.resize(face, self.SIZE)

                        # Use HOG to extract features of face images

                        fd, hog\_image = hog(face\_resize, orientations=9, pixels\_per\_cell= self.pixels\_per\_cell,cells\_per\_block= self.cells\_per\_block,visualize=True,transform\_sqrt=True, multichannel=True)

                        x[:, i] = fd

                        count +=1

                        print(str(count) + " images was trained ==> "+ path[i])

                except:

                    print("Error ==> " + path[i])

            labels = [p.split(os.path.sep)[-2] for p in path]

        except:

            print("Error ==> " + path[i])

        return np.asarray(x), np.asarray(labels)

        '''trả về một mảng các đặc trưng của từng bức khuôn mặt train và

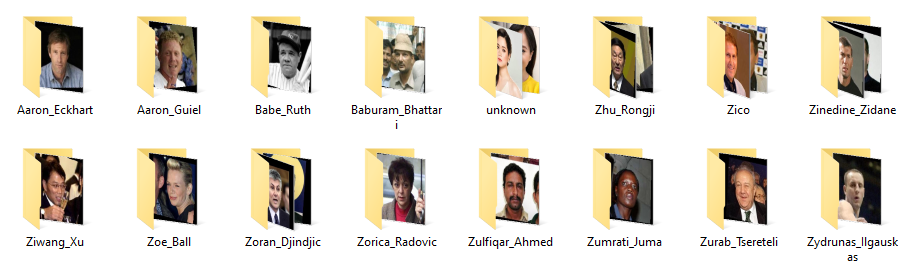
        nhãn tương ứng của từng khuôn mặt train'''

# **CHƯƠNG 4:XÂY DỰNG HỆ THỐNG CHẤM CÔNG TỰ ĐỘNG BẰNG NHẬN DIỆN KHUÔN MẶT**

## **4.1. Mô tả dữ liệu**

Dữ liệu đầu vào là các ảnh khuôn mặt người em lấy từ trang <https://www.kaggle.com/ashwingupta3012/human-faces> .Em đã download và đẩy lên drive https://drive.google.com/file/d/1DdEgdGyPS9AeupejSVdueQAoXcYvVEGE/view?usp=sharing

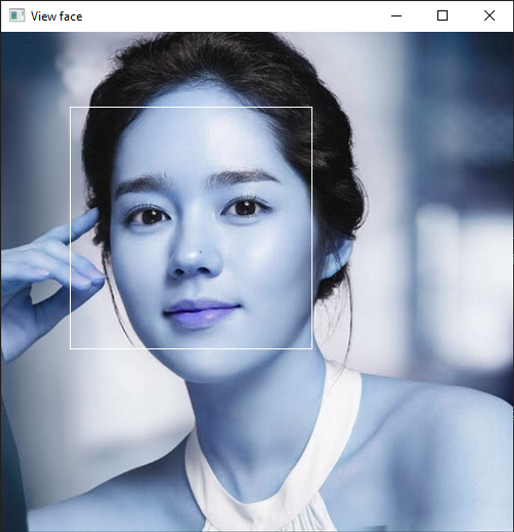
Dữ liệu có dạng:



## **4.2. Training dữ liệu**

Em sử dựng Hog để tìm khuôn mặt trong ảnh,hàm sử dụng : face\_locations của thư viện face\_recognition

Kết quả:



Sau đó sẽ dung hàm hog của thư viện skimage.feature để trích trọn đặc trưng khuôn mặt.Kết quả đặc trưng và nhãn(tương ứng với tên thư mục chứa ảnh) sẽ được lưu và file image\_train\_success.npz



## **4.2. Sử dụng SVM để nhận diện khuôn mặt**

Các bước thực hiện:

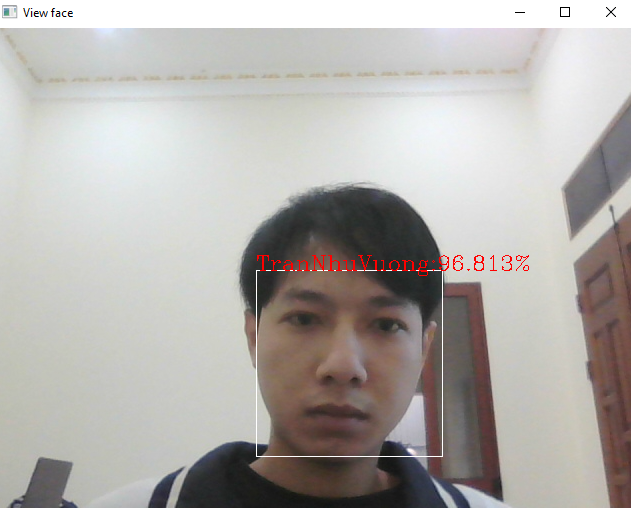
B1: Build hàm SVM từ tệp dữ liệu training \*.npz đã lấy được từ bước training dữ liệu

B2: Dùng hàm face\_locations của thư viện face\_recognition để tìm khuôn mặt trên camera.

B3: Dùng hàm hog của thư viện skimage.feature để trích trọn khuôn mặt vừa tìm được

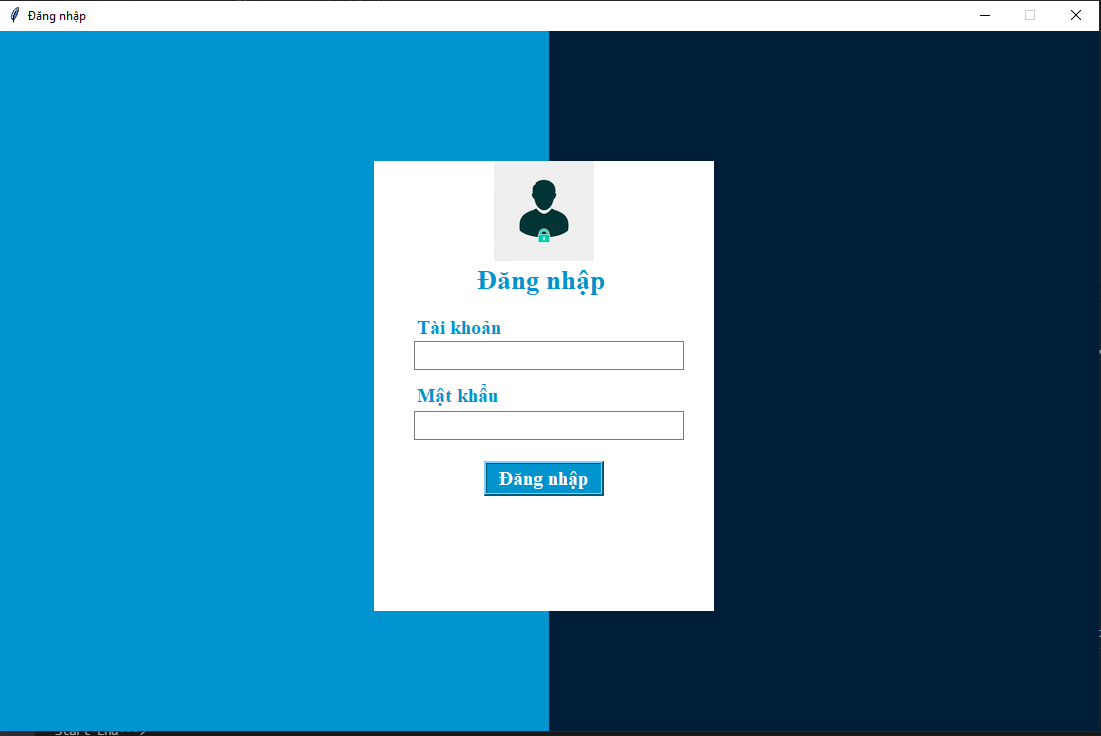
B4: Đưa vector đặc trương khuôn mặt vào hàm SVM để tiến hành nhận diện

Kết quả nhận diện:



## **4.3. Các giao diện hệ thống**

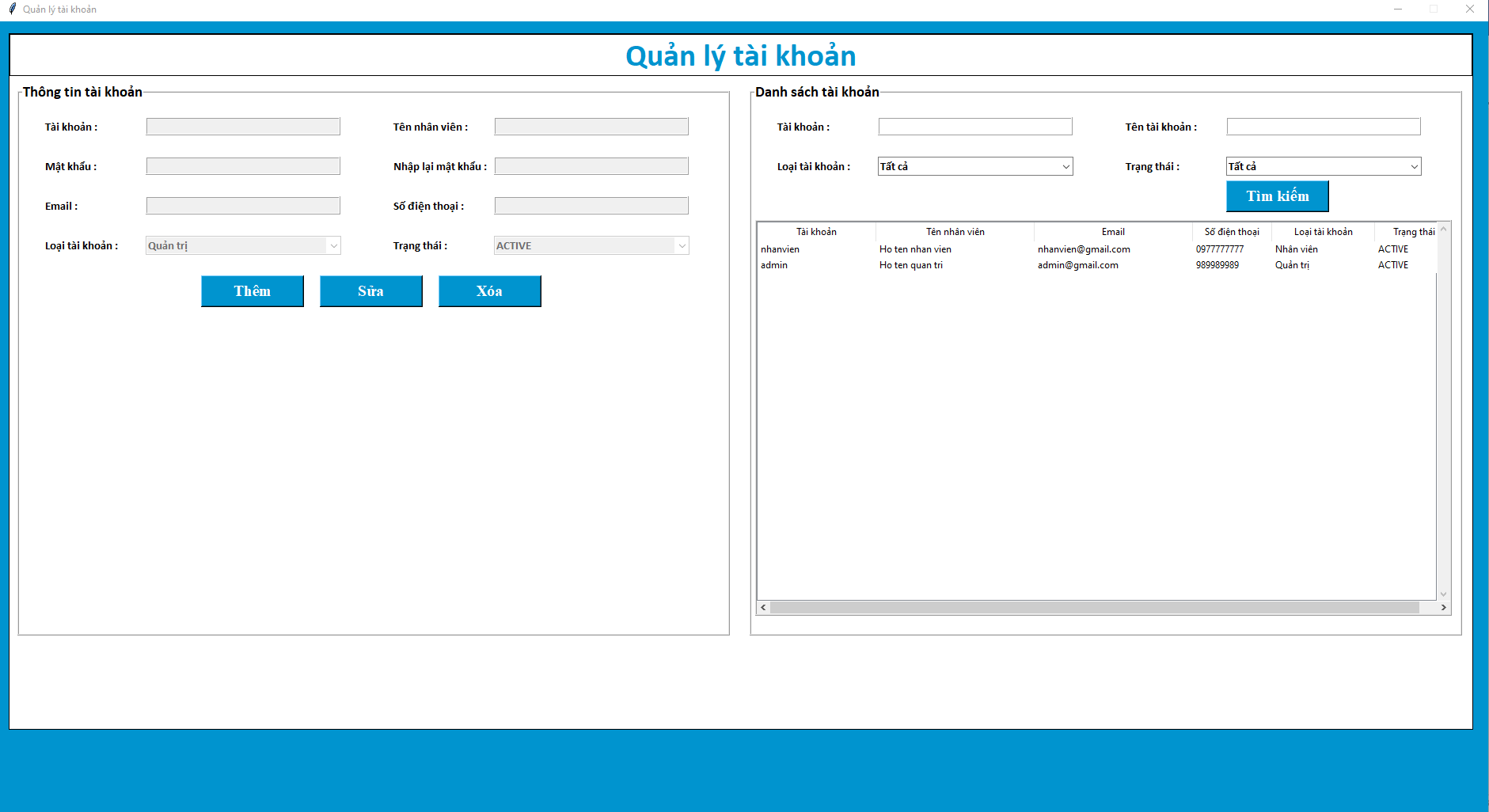
## **4.3.1. Giao diện Login**



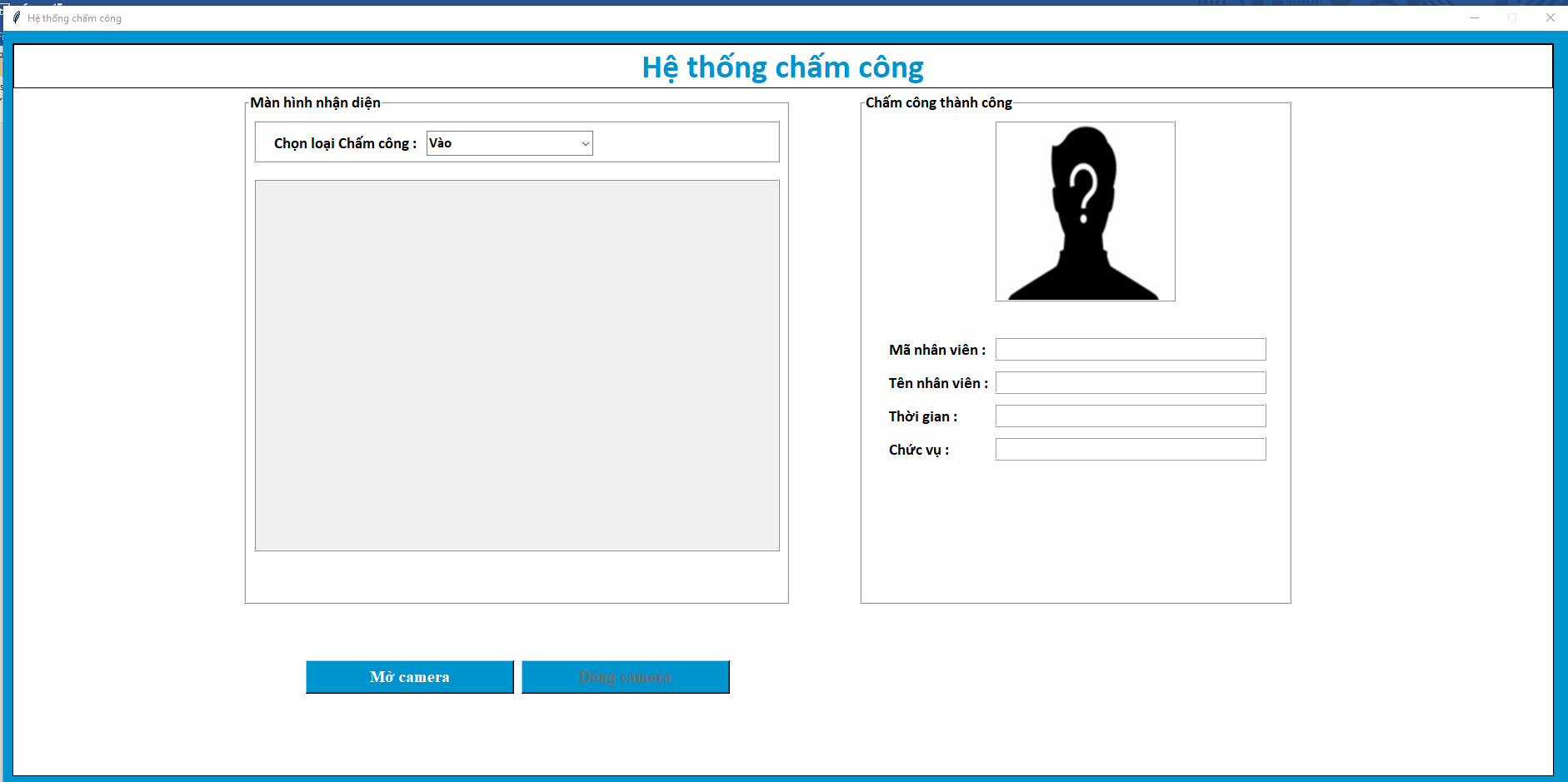
## **4.3.2. Giao diện Hệ thống**



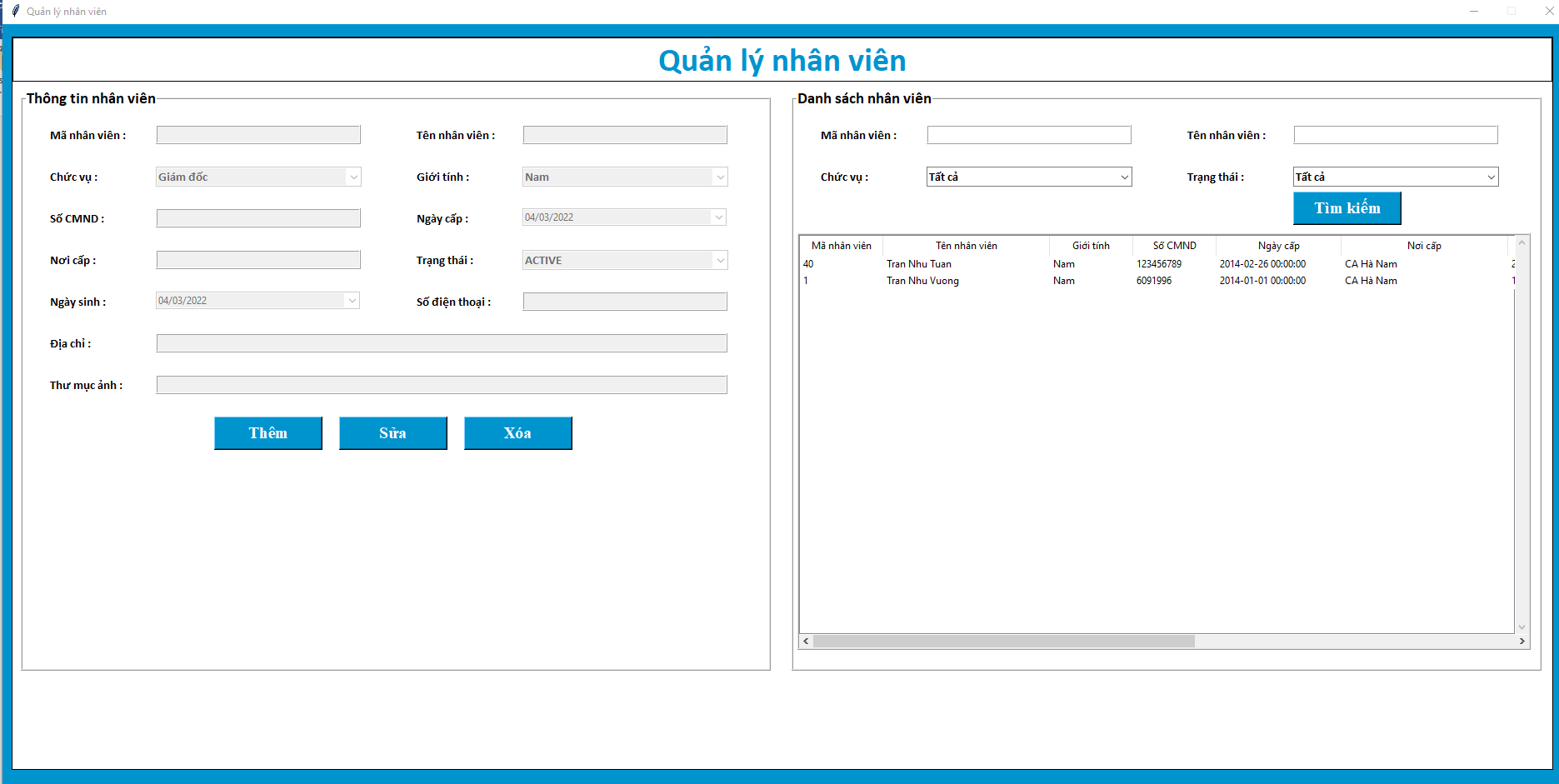
## **4.3.3. Giao diện Quản lý tài khoản**



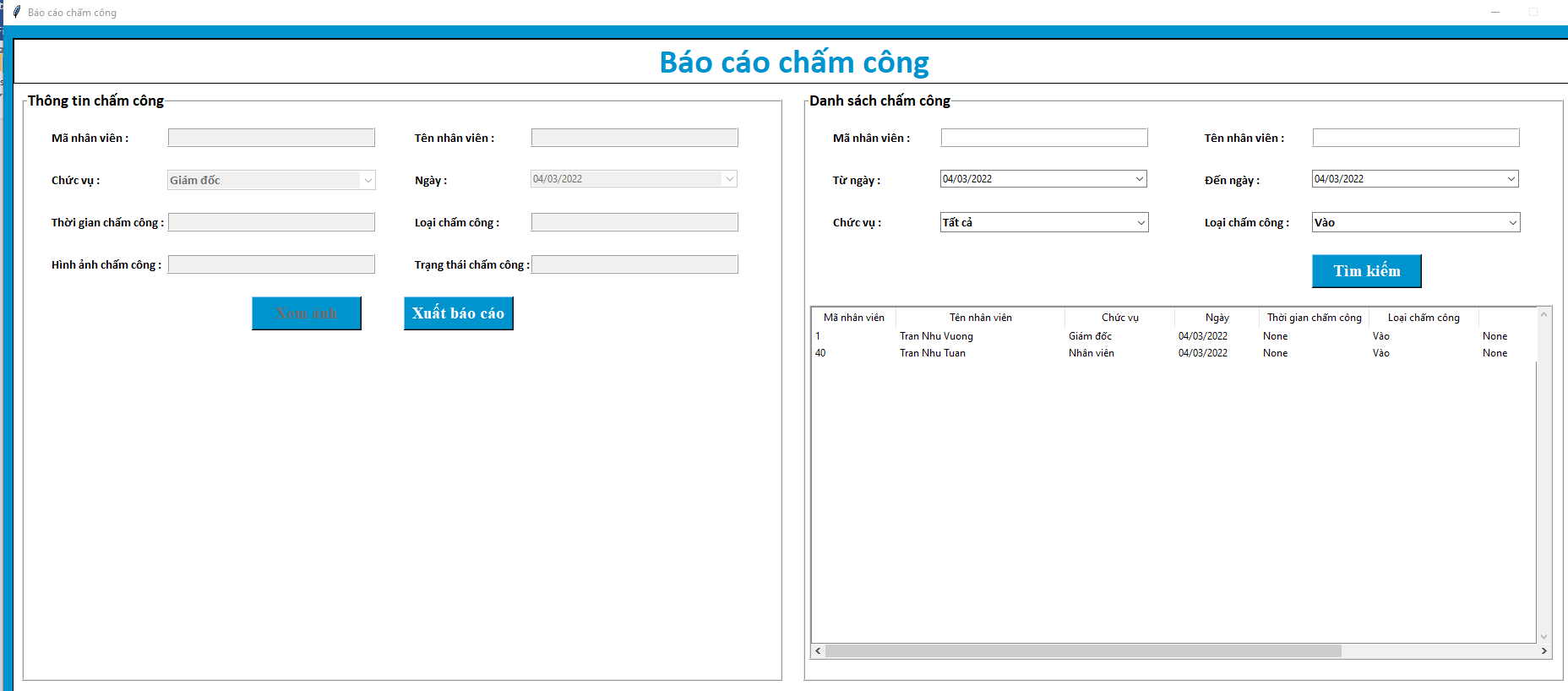
## **4.3.4. Giao diện Nhận diện**



## **4.3.5. Giao diện Quản lý nhân viên**



## **4.3.6. Giao diện Báo cáo thống kê**



# **CHƯƠNG 5:KẾT LUẬN**

***\* Nhận dạng mẫu***

Nhận dạng mẫu [8] (*pattern recognition*) là một ngành thuộc lĩnh vực học máy (*machine learning*). Nói cách khác, nó có thể được xem là việc "cần thực hiện một tác động vào dữ liệu thô mà tác động cụ thể là gì sẽ tùy vào loại của dữ liệu đó" [1]. Như vậy nó là một tập các phương pháp học có giám sát (*supervised learning*).

Nhận dạng mẫu nhằm mục đích phân loại dữ liệu [8] (là các mẫu) dựa trên: hoặc là kiến thức tiên nghiệm (*a priori*) hoặc dựa vào thông tin thống kê được trích rút từ các mẫu có sẵn. Các mẫu cần phân loại thường được biểu diễn thành các nhóm của các dữ liệu đo đạc hay quan sát được, mỗi nhóm là một điểm ở trong một không gian đa chiều phù hợp. Đó là không gian của các đặc tính để dựa vào đó ta có thể phân loại.

Một hệ thống nhận dạng mẫu hoàn thiện gồm một thiết bị cảm nhận (*sensor*) để thu thập các quan sát cần cho việc phân loại hay miêu tả; một cơ chế trích rút đặc trưng (*feature extraction*) để tính toán các thông tin dưới [8] dạng số hay dạng tượng trưng (*symbolic*) từ các dữ liệu quan sát được; và một bộ phân loại (hay lược đồ mô tả) nhằm thực hiện công việc phân loại thực sự (hay miêu tả các quan sát đó) dựa vào các đặc tính đã được trích rút.

Việc phân loại (hay lược đồ mô tả) thường dựa vào sự có sẵn của một tập các mẫu mà đã được phân loại (hay miêu tả) sẵn. Tập các mẫu này được gọi là tập huấn luyện và chiến lược học nhằm phân loại mẫu vào một [8] trong các lớp có sẵn được gọi là học có giám sát. Việc học cũng có thể là không có giám sát, theo nghĩa là hệ thống không được cung cấp các mẫu được đánh nhãn (phân loại) *tiên nghiệm*, mà nó phải tự đưa ra các lớp để phân loại dựa vào tính ổn định trong thống kê của các mẫu.

Việc phân loại (hay lược đồ mô tả) thường dùng một trong các hướng tiếp cận sau: thống kê (hay lý thuyết quyết định), cú pháp (hay cấu trúc). [8]Nhận dạng mẫu dùng thống kê là dựa vào các đặc tính thống kê của các mẫu, chẳng hạn rằng các mẫu được tạo mởi một hệ thống xác suất. Nhận dạng dùng cấu trúc là dựa vào tương quan cấu trúc giữa các mẫu.

***\* Về SVM:***

Ưu điểm*:* Là một kĩ thuật phân lớp khá phổ biến, SVM thể hiện được nhiều ưu điểm trong số đó có việc tính toán hiệu quả trên các tập dữ liệu lớn. Có thể kể thêm một số ưu điểm của phương pháp này như:

* **Xử lý trên không gian số chiều cao**: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn
* **Tiết kiệm bộ nhớ**: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết dịnh
* **Tính linh hoạt** - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

Nhược điểm:

* ***Bài toán số chiều cao***: Trong trường hợp số lượng thuộc tính (**p**) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (**n**) thì SVM cho kết quả khá tồi
* ***Chưa thể hiện rõ tính xác suất***: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm **margin** từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã bàn luận.

***\* Ứng dụng thực tế:***

Hiện tại ứng dụng nhận diện khuôn đang góp một phần không nhỏ trong cuộc sống hằng ngày, nó có thể được tích hợp vào điện thoại để xây dựng chức năng mở khóa.Hay trong hình sự nó được dụng để xác dịnh khuôn mặt của tội phạm, ngoài ra còn rất nhiều lợi ích mà nó đêm lại. Mục đích chính là nhằm tăng năng suất, sức lực cho con người, giúp con người hoàn thành công việc một cách nhanh chóng và chính xác hơn.

***\* Đã làm được:***

* Xây dựng được một chương trình chấm công tự động bằng khuôn mặt người
* Biết được quy trình và cách một chương trình nhận dạng làm việc

***\* Chưa làm được:***

* Chương trình còn nhiều hạn chế về kĩ thuật và dữ liệu
* Độ chính xác chưa cao

# **TÀI LIỆU THAM KHẢO**

[1] Margaret Rouse – [Pattern Recognition](https://whatis.techtarget.com/definition/pattern-recognition)

[2] Nam Doan, 31 , 2018 [Data Science](https://1upnote.me/categories/data-science), [Machine Learning](https://1upnote.me/categories/machine-learning), [Support Vector Machine (SVM) là gì?](https://1upnote.me/post/2018/10/ds-ml-svm/)

[3] [SVM quá khó hiểu! Hãy đọc bài này](https://trituenhantao.io/kien-thuc/svm-qua-kho-hieu-hay-doc-bai-nay/amp/?fbclid=IwAR2NrGP3beRkEz-KzuKLe19y17bh7jKIkSHdoiPEJmVzP5ilt7YCTpXDRJM)

[4] [Support Vector Machine](https://machinelearningcoban.com/2017/04/09/smv/)

[5] [Kernel Support Vector Machine](https://machinelearningcoban.com/2017/04/22/kernelsmv/)

[6] <https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>

[7] <https://www.onlinemathlearning.com/vector-magnitude.html>

[8] [Nhận dạng mẫu](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_m%E1%BA%ABu)

[9] [Support Vector Machine trong học máy - Một cái nhìn đơn giản hơn](https://viblo.asia/p/support-vector-machine-trong-hoc-may-mot-cai-nhin-don-gian-hon-XQZkxoQmewA)

[Histogram of Oriented Gradients](https://www.learnopencv.com/histogram-of-oriented-gradients/?fbclid=IwAR18at-SxmVl7Uqb8_7wjpj5r6T0NzpSK-g9yc8i2QWd8WD_RGfKyVQZ-4U)

[HOG Person Detector Tutorial](http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/?fbclid=IwAR2d9GJgH8ZFgVGAwE0SUMndXUk2LN6kbb88ik1CSJLqyH3LBo4fMCSQXv4)