**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**MÔN : THỰC HÀNH TRÍ TUỆ NHÂN TẠO**

**CHỦ ĐỀ : NHẬN DẠNG CHỮ SỐ VIẾT TAY ÁP DỤNG THUẬT TOÁN**

**K-NN**

Tp.Hồ Chí Mình , Tháng 12 Năm 2022

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**MÔN : THỰC HÀNH TRÍ TUỆ NHÂN TẠO**

**CHỦ ĐỀ : NHẬN DẠNG CHỮ SỐ VIẾT TAY ÁP DỤNG THUẬT TOÁN**

**K-NN**

**Giáo viên hướng dẫn** : Ngô Dương Hà

**Thông tin sinh viên :**

Hoàng Văn Sơn – 2001202224

Đoàn Trường Thành- 2001202246

Mai Nguyễn Phước Yến – 2001200561

Tp.Hồ Chí Mình , Tháng 12 Năm 2022

**MỤC LỤC**

[**Chương 1: TỔNG QUAN 1**](#_Toc123040627)

[**1.1 Giới thiệu 1**](#_Toc123040628)

[**1.2 Mục tiêu đề tài 1**](#_Toc123040629)

[**1.3 Giới hạn đề tài 1**](#_Toc123040630)

[**Chương 2: CƠ SỞ LÝ THUYẾT 2**](#_Toc123040631)

[**2.1. Thuật toán KNN 2**](#_Toc123040632)

[**2.1.1. Định nghĩa 2**](#_Toc123040633)

[**2.1.2. Quy trình làm việc của thuật toán KNN 2**](#_Toc123040634)

[**2.1.3. Ví dụ minh họa 2**](#_Toc123040635)

[**2.1.4. Ví dụ về Knn nhiễu 3**](#_Toc123040636)

[**2.1.5. Ưu điểm, nhược điểm của thuật toán 3**](#_Toc123040637)

[**2.1.6. Khoảng cách trong không gian vector 4**](#_Toc123040638)

[**2.1.7. Định nghĩa 4**](#_Toc123040639)

[**2.1.8. Một số norm thường dùng 4**](#_Toc123040640)

[**2.2. Giới thiệu về python 5**](#_Toc123040641)

[**Đặc điểm 5**](#_Toc123040642)

[**Chương 3: NỘI DUNG THỰC HIỆN 7**](#_Toc123040643)

[**3.1. Các bước mô tả thuật toán của chương trình. 7**](#_Toc123040644)

[**3.2. Giao diện tổng thể của chương trình. 9**](#_Toc123040645)

[**3.3. Thử nghiệm kết quả của chương trình. 9**](#_Toc123040646)

[**1. Nhận dạng số 0. 9**](#_Toc123040647)

[**2. Nhận dạng số 1. 10**](#_Toc123040648)

[**3. Nhận dạng số 2. 11**](#_Toc123040649)

[**4. Nhận dạng số 3. 11**](#_Toc123040650)

[**5. Nhận dạng số 4. 12**](#_Toc123040651)

[**6. Nhận dạng số 5. 12**](#_Toc123040652)

[**7. Nhận dạng số 6. 13**](#_Toc123040653)

[**8. Nhận dạng số 7. 14**](#_Toc123040654)

[**9. Nhận dạng số 8. 14**](#_Toc123040655)

[**10. Nhận dạng số 9. 15**](#_Toc123040656)

[**Chương 4: KẾT QUẢ - KẾT LUẬN 16**](#_Toc123040657)

[**4.1. Kết quả đạt được 16**](#_Toc123040658)

[**1.4 Hạn chế của đề tài 16**](#_Toc123040659)

[**1.5 Hướng phát triển 16**](#_Toc123040660)

[**TÀI LIỆU KHAM KHẢO 17**](#_Toc123040661)

[**PHỤ LỤC 18**](#_Toc123040662)

[**Code chương trình: 18**](#_Toc123040663)

**Lời Cảm Ơn**

Đầu tiên , chúng em xin gửi lời cảm ơn tới thầy Ngô Dương Hà là giảng viên bộ môn Thực Hành Trí Tuệ Nhân Tạo đã mang tới cho chúng em những buổi học hữu ích với những kiến thức đầy ý nghĩa của bộ môn cũng như đã hướng dẫn chúng em hoàn thiện bài tiểu luận một cách tốt nhất.

Tuy vậy , do vốn kiến thức và khả năng tiếp thu còn nhiều hạn chế trước biển rộng kiến thức của bộ môn Thực Hành Trí Tuệ Nhân Tạo . Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài tiểu luận khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy xem xét và góp ý để bài tiểu luận của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn !

\

**DANH MỤC CÁC HÌNH ẢNH**

[Hình 2.1*: Ví dụ minh họa thuật toán KNN*](#_Toc80213563) *2*

[Hình 2.1](#_Toc80213563)*[: Bản đồ minh họa knn nhiễu với k=1](#_Toc80213563) 3*

[Hình 2.3:](#_Toc80213568) *[Norm 1 và norm 2 trong không gian hai chiều](#_Toc80213568)*5

[Hình 3.1:](#_Toc80213575) *[Giao diện của chương trình](#_Toc80213575)* [9](#_Toc80213575)

[Hình 3.2: *Kết quả thử nghiệm nhận dạng số 0*](#_Toc80213576) 9

[Hình 3.3:](#_Toc80213576) *[Kết quả thử nghiệm nhận dạng số 1](#_Toc80213576)* [1](#_Toc80213576)0

[Hình 3.4:](#_Toc80213578) *[Kết quả thử nghiệm nhận dạng số 2](#_Toc80213578)* [1](#_Toc80213578)0

[Hình 3.5:](#_Toc80213579) *[Kết quả thử nghiệm nhận dạng số 3](#_Toc80213579)*[. 1](#_Toc80213579)0

[Hình 3.6:](#_Toc80213580) *[Kết quả thử nghiệm nhận dạng số 4](#_Toc80213580)* [1](#_Toc80213580)1

[Hình 3.7:](#_Toc80213581) *[Kết quả thử nghiệm nhận dạng số 5](#_Toc80213581)* [1](#_Toc80213581)1

[Hình 3.8:](#_Toc80213582) *[Kết quả thử nghiệm nhận dạng số 6](#_Toc80213582)*  [11](#_Toc80213582)

[*Hình 3.9: Kết quả thử nghiệm nhận dạng số 7 12*](#_Toc80213582)

[*Hình 3.10: Kết quả thử nghiệm nhận dạng số 8*](#_Toc80213582) *12*

[*Hình 3.11: Kết quả thử nghiệm nhận dạng số 9*](#_Toc80213582) *12*

# Chương 1: TỔNG QUAN

## Giới thiệu

Hiện tại, thời đại công nghiệp, số hóa đang là xu hướng của toàn thế giới, tuy nhiên đối với đất nước chúng ta thì đó đang là trở ngại rất lớn, việc số hóa vẫn đang còn thực hiện một cách nhỏ lẻ, không đồng nhất và không mang tính cục bộ. Mặc dù, trong công việc thường nhật ta có thể dễ dàng nhìn thấy chữ số viết tay có thể xuất hiện bất kì nơi nào trong cuộc sống của chúng ta như các loại hóa đơn, lưu kí, chứng từ, thư tay, … Trong Công Nghệ Thông Tin như các phần mềm Paint, Power Point, …

Bên cạnh đó, mỗi cá nhân sẽ có các cách viết chữ số khác nhau, làm cho việc xác minh, nhận biết các chữ số sẽ trở nên khó khăn và mất rất nhiều thời gian, bên cạnh đó nó còn giúp lưu trữ thông tin dễ dàng, lâu hơn và giúp tiết kiệm cũng như bảo vệ môi trường

Do đó nhóm chúng em quyết định lựa chọn đề tài “***NHẬN DẠNG SỐ VIẾT TAY DÙNG KNN***” nhằm để nghiên cứu, áp dụng các kiến thức đã được, hiểu rõ hơn thuật toán KNN để tiến hành nhận diện các chữ số viết tay.

## Mục tiêu đề tài

* Hiểu và áp dụng được các thuật toán trong quá trình học.
* Hiểu và vận dụng được thuật toán KNN .
* Nhận dạng được các chữ số viết tay dựa trên thuật toán KNN.

## Giới hạn đề tài

* Chỉ nhận dạng các chữ số viết tay từ 0 đến 9.
* Chất lượng nhận dạng của thuật toán KNN vẫn chưa thể cho ra kết quả đúng ở lần đầu tiên nếu có quá nhiều lớp có số phần tử giống nhau.

# Chương 2: CƠ SỞ LÝ THUYẾT

## 2.1. Thuật toán KNN

### 2.1.1. Định nghĩa

K­nearest neighbor (KNN) là một trong những thuật toán học có giám sát đơn giản nhất trong Machine Learning. Ý tưởng của KNN là tìm ra output của dữ kiệu dựa trên thông tin của những dữ liệu training gần nó nhất.

### 2.1.2. Quy trình làm việc của thuật toán KNN

Bước 1: Xác định tham số K= số láng giềng gần nhất.

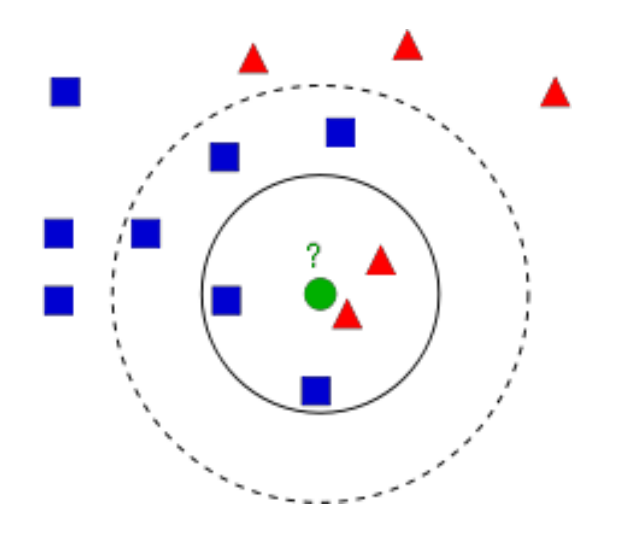
Bước 2: Tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong training data.

Bước 3: Sắp xếp khoảng cách theo thứ tự tăng dần và K láng giềng gần nhất với đối tượng cần phân lớp.

Bước 4: Lấy tất cả các lớp của K láng giềng gần nhất.

Bước 5: Dựa vào phần lớn lớp của K để xác định lớp cho đối tượng cần phân lớp.

### 2.1.3. Ví dụ minh họa



*Hình 2.1. Ví dụ minh họa thuật toán KNN*

Giả sử bài toán được đặt ra: mình mới quen một người bạn, tuy nhiên mình là fan của Us­Uk vậy nên mình cần biết người bạn này có phải là fan của K­Pop hay không. Qua thời gian tìm hiểu mình đã thu thập được một số dữ liệu và đã biểu hiện dưới dạng hình vẽ trên.

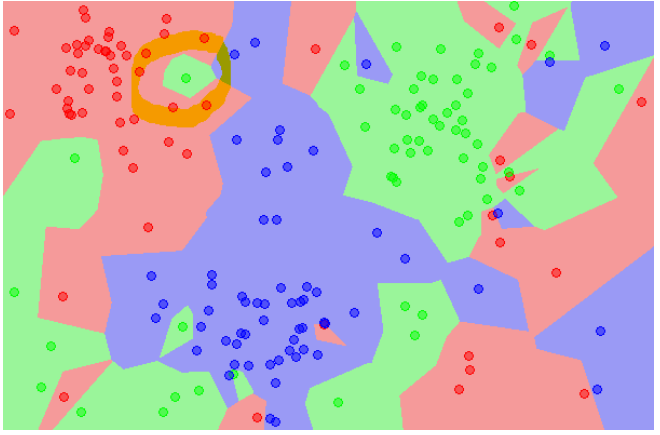
Ta dễ dàng nhìn thấy có hai loại: hình vuông màu xanh biểu diễn cho những người là fan của K­pop, tam giác màu đỏ biểu diễn cho những người không là fan của K­pop, hình tròn màu xanh là người bạn mình muốn biết có phải là fan K­pop hay không, khoảng cách giữa chấm tròn và các điểm còn lại biểu diễn độ thân thiết của bạn đó với những người bạn.

Phương pháp đơn giản nhất để kiểm tra xem bạn đó chơi thân với người bạn nào nhất, tức là tìm xem điểm gần chấm xanh thuộc class nào (hình vuông hay tam giác). Từ hình trên ta dễ dàng nhận thấy điểm gần chấm xanh nhất là hình tam giác màu đỏ, do đó nó sẽ được phân vào lớp tam giác màu đỏ.

Có một vấn đề trong phương pháp trên, xung quanh cấm xanh xuất hiện rất nhiều hình vuông màu xanh nên việc xét điểm gần nhất là chưa khả thi. Vì vậy, ta sẽ xét k điểm gần nhất. Giả sử, ta lấy K=3, dựa theo hình trên ta dễ dàng nhận ra có hai hình tam giác đỏ và một hình vuông xanh có khoảng cách gần chấm xanh nhất, do đó chấm xanh được phân vào lớp tam giác đỏ. Lấy K=7, ta có năm hình vuông xanh và hai hình tam giác đỏ, lúc này chấm xanh được xếp vào lớp hình vuông xanh. Trường hợp lấy K=4, ta nhận thấy sẽ có hai hình vuông xanh và hai hình tam giác đỏ, đây là trường hợp có điểm bằng nhau, với trường hợp này KNN sẽ xử lý bằng cách so sánh tổng khoảng cách của các hình gần nhất với điểm ta đang xét.

Do xuất hiện trường hợp có điểm bằng nhau, vì vậy người ta thường chọn k là số lẻ. Đó cũng là ý tưởng của KNN.

### 2.1.4. Ví dụ về Knn nhiễu



*Hình 2.2. Bản đồ minh họa knn nhiễu với k=1*

Hình trên là bài toán phân lớp với ba lớp: đỏ, lam, lục. Mỗi điểm dữ liệu mới sẽ được gán nhãn theo màu của điểm đó mà nó thuộc về. Trong hình này, chú ý vùng khoanh tròn màu vàng, ta nhận thấy rằng điểm màu lục nằm giữa hai vùng lớn với nhiều dữ liệu đỏ và lam, điểm này rất có thể là nhiễu dẫn đến việc dữ liệu tst nếu rơi vào vùng này sẽ có nhiều khả năng cho kết quả sai lệch.

### 2.1.5. Ưu điểm, nhược điểm của thuật toán

**Ưu điểm:**

* Dễ sử dụng và cài đặt.
* Việc dự đoán kết quả của dữ liệu mới dễ dàng.
* Độ phức tạp tính toán nhỏ.

**Nhược điểm**:

* KNN nhiễu dễ đưa ra kết quả không chính xác khi k nhỏ.
* Cần thời gian lưu training set, khi dữ liệu training và test tăng lên nhiều sẽ mất nhiều thời gian tính toán.

### 2.1.6. Khoảng cách trong không gian vector

Trong không gian một chiều, việc đo khoảng cách giữa hai điểm đã rất quen thuộc: lấy trị tuyệt đối của hiệu giữa hai giá trị đó. Trong không gian hai chiều, tức mặt phẳng, chúng ta thường dùng khoảng cách Euclid để giữa hai điểm đo khoảng cách.

Việc đo khoảng cách giữa hai điểm dữ liệu nhiều chiều, tức hai vector, là rất cần thiết trong Machine Learning. Chúng ta cần đánh giá xem điểm nào là điểm gần nhất của một điểm khác; chúng ta cũng cần đánh giá xem độ chính xác của việc ước lượng; và trong rất nhiều ví dụ khác nữa.

Và đó chính là lý do mà khái niệm norm ra đời. Có nhiều loại norm khác nhau mà các bạn sẽ thấy ở dưới đây:

Để xác định khoảng cách giữa hai vector **y** và **z**, người ta thường áp dụng một hàm số lên vector hiệu **x = y−z**. Một hàm số được dùng để đo các vector cần có một vài tính chất đặc biệt.

### 2.1.7. Định nghĩa

Một hàm số **f ()** ánh xạ một điểm **x** từ không gian nn chiều sang tập số thực một chiều được gọi là norm nếu nó thỏa mãn ba điều kiện sau đây:

­ **F(x) >= 0**. Dấu bằng xảy ra  **x** = 0.

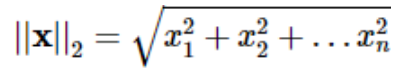
­ **F(**α**x) = |**α**|f(x),** ∀α € R.

­ **F(x1) +f(x2) >= f (x1 + x2)**, ∀**x**1, **x2** € R

### 2.1.8. Một số norm thường dùng

Giả sử các vector **x =** [x1; x2…xn], **y =** [y1; y2…yn].

Nhận thấy khoảng cách Euclid chính là một norm, norm mày thường được gọi là norm 2:

 (2.1)

Với p là một số không nhỏ hơn 1 bất kỳ, hàm số sau đây:

 (2.2)

Được chứng minh thỏa mãn ba ddieuf kiện trên, và được gọi là norm p.

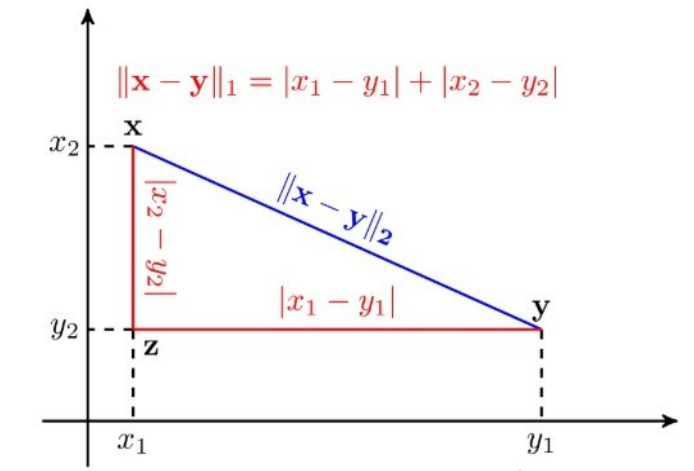
Nhận thấy rằng khi **p**→0 thì biểu thức bên trên trở thành số các phần tử khác 0 của x. Hàm số (2) khi **p**=0 được gọi là giả chuẩn (pseudo­norm) 0. Nó không phải là norm vì nó không thỏa mãn điều kiện 2 và 3 của norm. Giả­chuẩn này, thường được ký hiệu là ||**x**||0, khá quan trọng trong ML vì trong nhiều bài toán, chúng ta cần có ràng buộc “sparse”, tức số lượng thành phần “active” của **x** là nhỏ.

Có một vài giá trị của **p** thường được dùng:

* Khi **p** = 2 chúng ta có norm2 như ở trên.
* Khi **p** = 1 chúng ta có:

||**x**||1 = |x1| + |x2| + |x3| +…|xn| (2.3)

Là tổng các giá trị tuyệt đối của từng phần tử của **x**. Norm 1 thường được dùng như sấp xỉ của norm 0 trong các bài toán có ràng buộc. Dưới đây là một ví dụ so sánh norm 1 và norm 2 trong không gian hai chiều

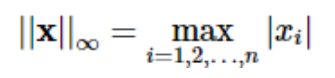


*Hình 2.3. Norm 1 và norm 2 trong không gian hai chiều*

Norm 2 (màu xanh) chính là đường chim bay nối giữa vector **x** và vector **y**.

Khoảng cách norm 1 giữa hai điểm này (màu đỏ) có thể diễn giải như là đường đi từ **x** đến **y** trong một thành phố mà thành phố được tạo hình bàn cờ, chúng ta chỉ có thể đi theo dọc bàn cờ chứ không thể đi theo đường thẳng.

Khi p ­> ∞, ta có norm p chính là trị tuyệt đối của phần tử lớn nhất của vector đó:

 (2.4)

## 2.2. Giới thiệu về python

- Sinh năm 1991, tác giả: Guido Van Rossum

Đặc điểm

- Là ngôn ngữ kịch bản và thông dịch.

- Trong sáng, gần gũi và dễ học.

+ Tăng cường sử dụng từ khóa tiếng Anh, hạn chế các kí hiệu. 

+ Hướng tới sự đơn giản.

\_ Có while bỏ do while.

\_ Có elif bỏ switch – case.

+ Ưu tiên cho việc đọc lại code.

- Đa năng

+ Lập trình web.

+ Ứng dụng desktop, đồ họa, game.

+ Lập trình cho điện thoại.

+ Đặc biệt hiệu quả trong lập trình tính toán khoa học.

- Đa biến hóa

+ Cho phép sử dụng nhiều phương pháp lập trình.

+ Kiểu động, kiểu vịt và kiểu mạnh.

- Python mạnh và nhanh

+ Kèm sẵn pin (Battery included).

+ Viết mã nhanh với số lần gõ phím tối thiểu.

- Hòa hợp tốt với các thứ khác

+ Java -> Jython.

+ .NET -> IronPython, Python for .NET

+ Viết module mở rộng cho C/C++

+ …

- Python chạy mọi nơi

+ Unix

+ Windows

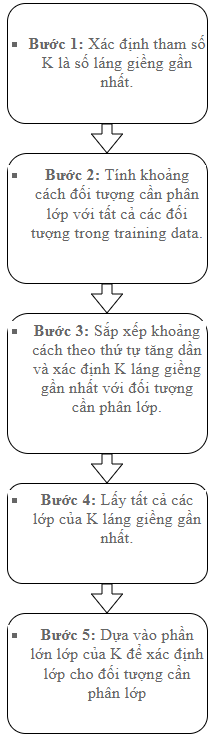
+ Mac

+ Nokia S60

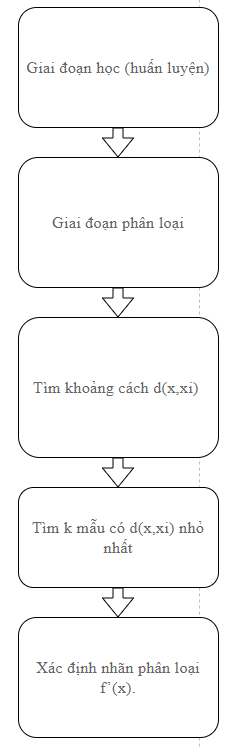
# Chương 3: NỘI DUNG THỰC HIỆN

## 3.1. Các bước mô tả thuật toán của chương trình.

- Các bước phân lớp sử dụng thuật toán KNN.

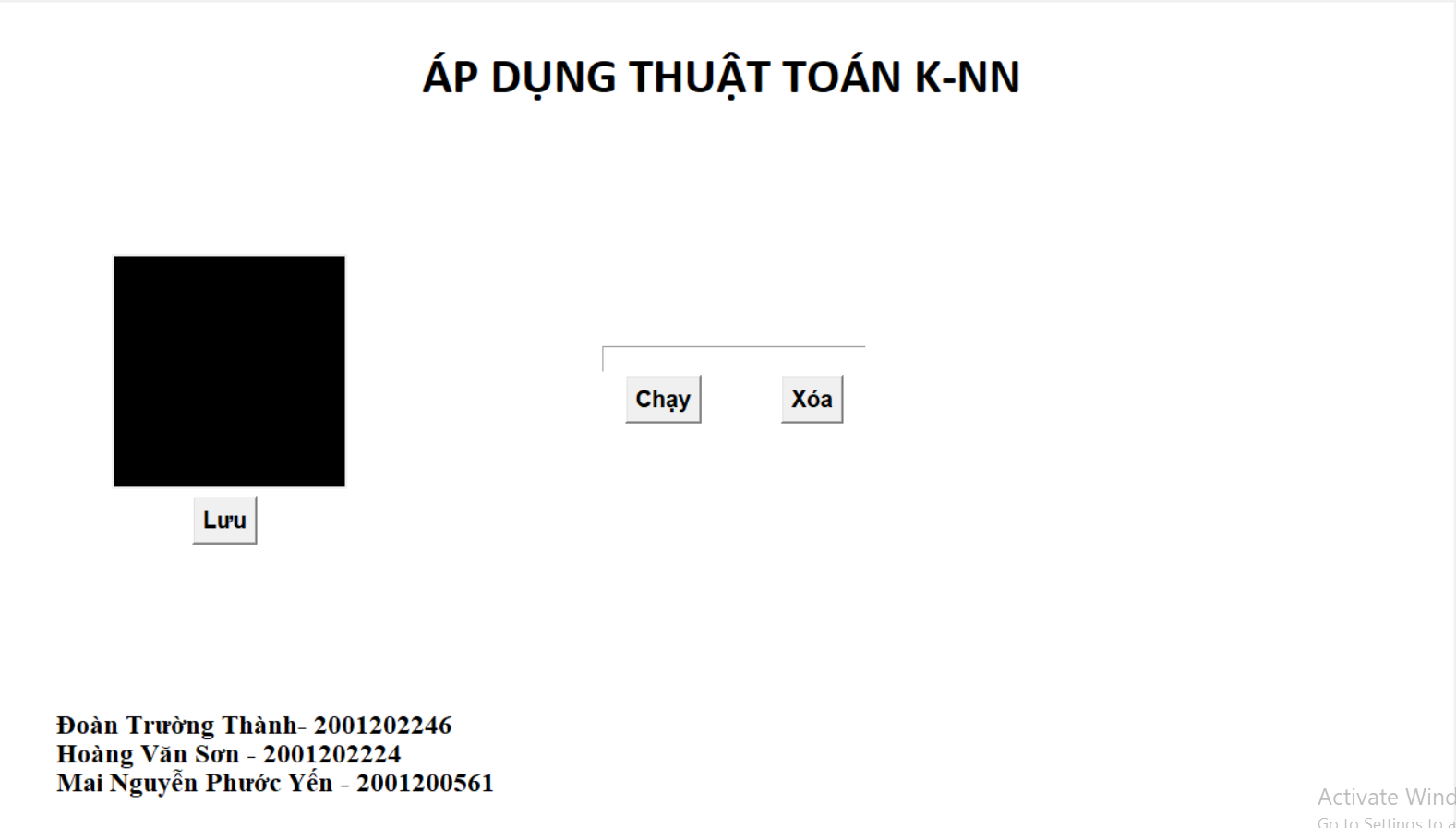


- Các bước mô tả thuật toán của chương trình.



## 3.2. Giao diện tổng thể của chương trình.

Sau khi nghiên cứu và lên ý tưởng, nhóm tác giả đã thiết kế giao diện cho chương trình như hình 3.1.

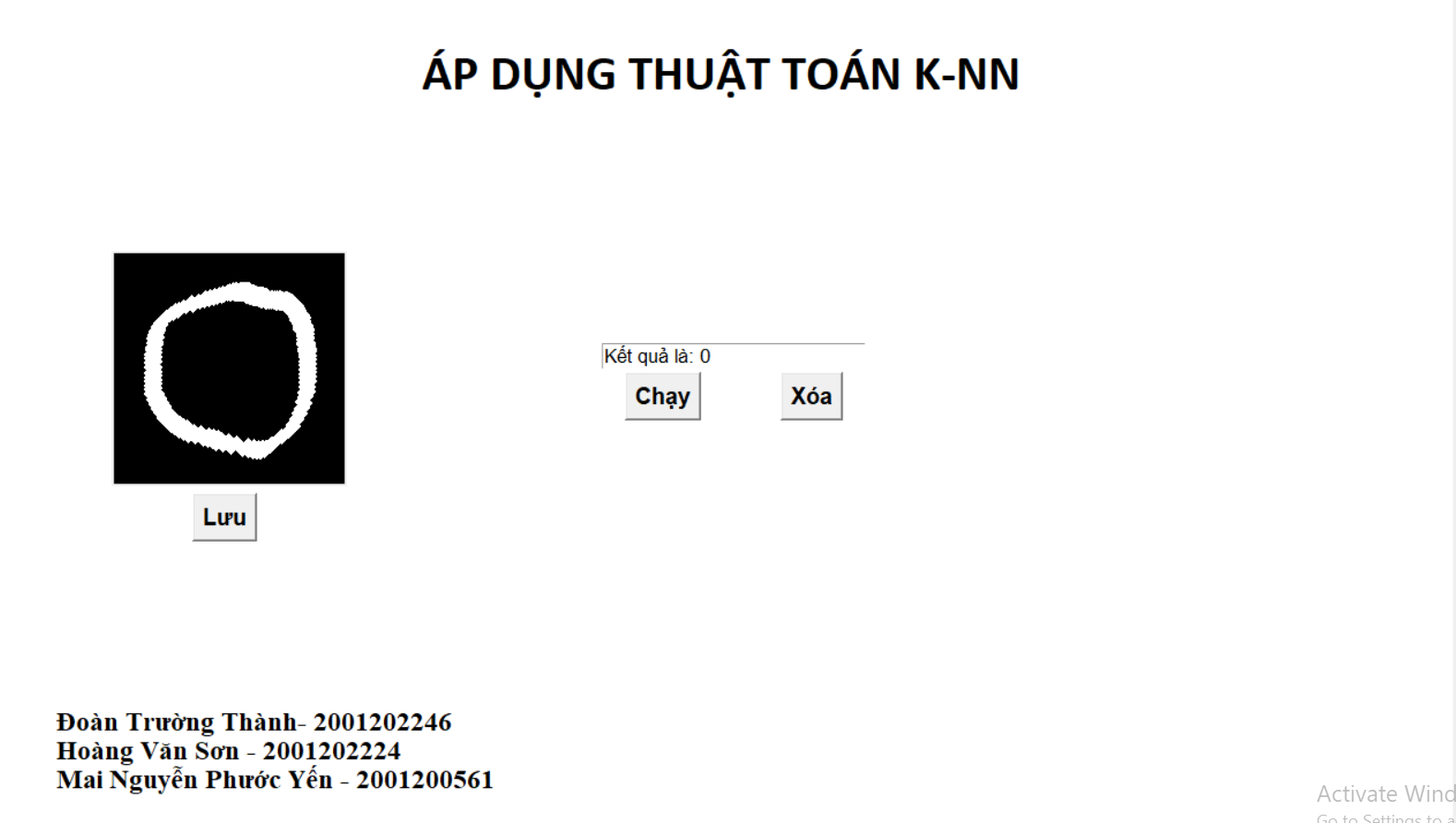


*Hình 3.1. Giao diện của chương trình*

## 3.3. Thử nghiệm kết quả của chương trình.

**1. Nhận dạng số 0.**

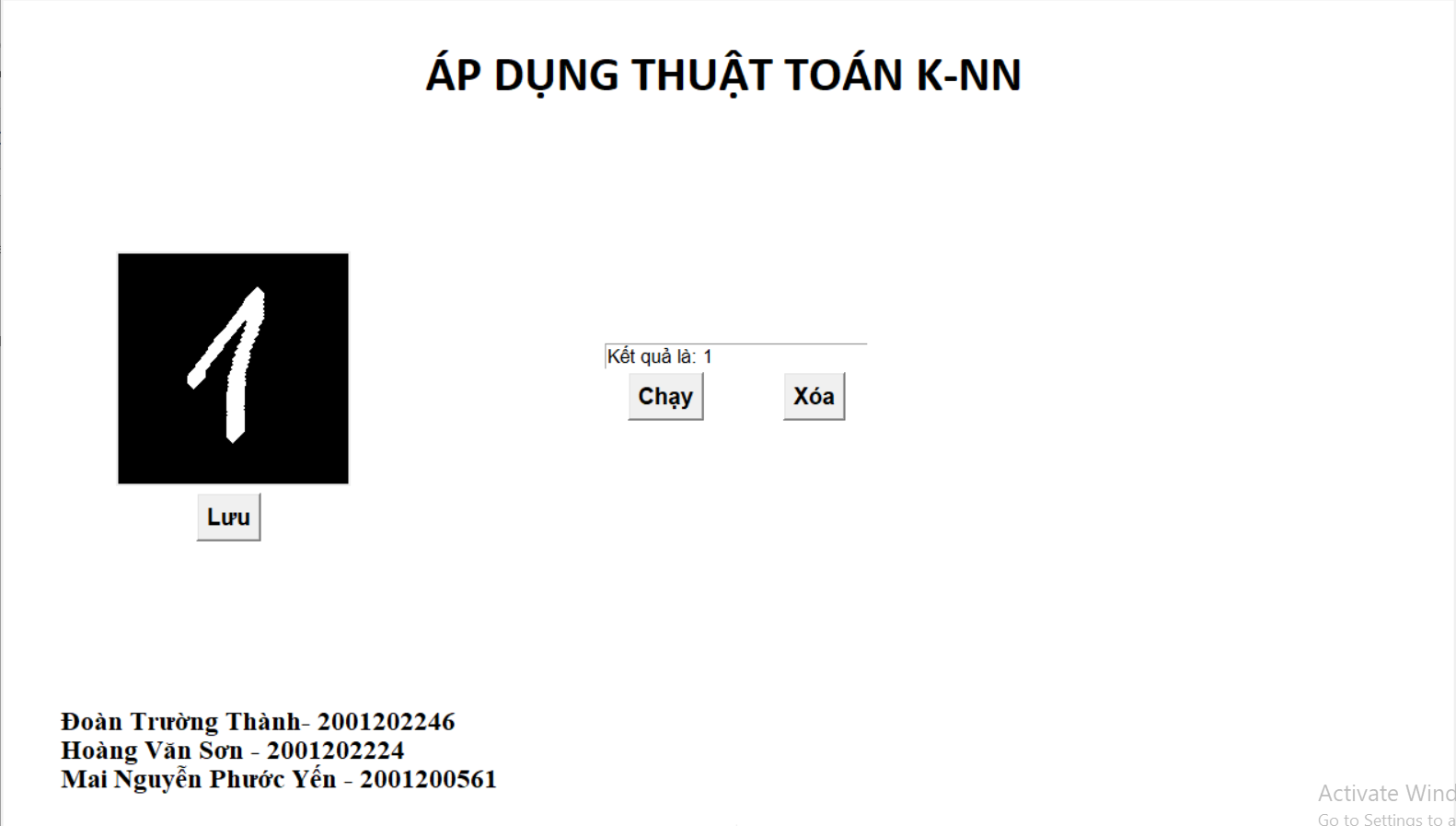
- Kết quả thử nghiệm nhận dạng số 0.



*Hình 3.2. Kết quả thử nghiệm nhận dạng số 0*

1. **Nhận dạng số 1.**

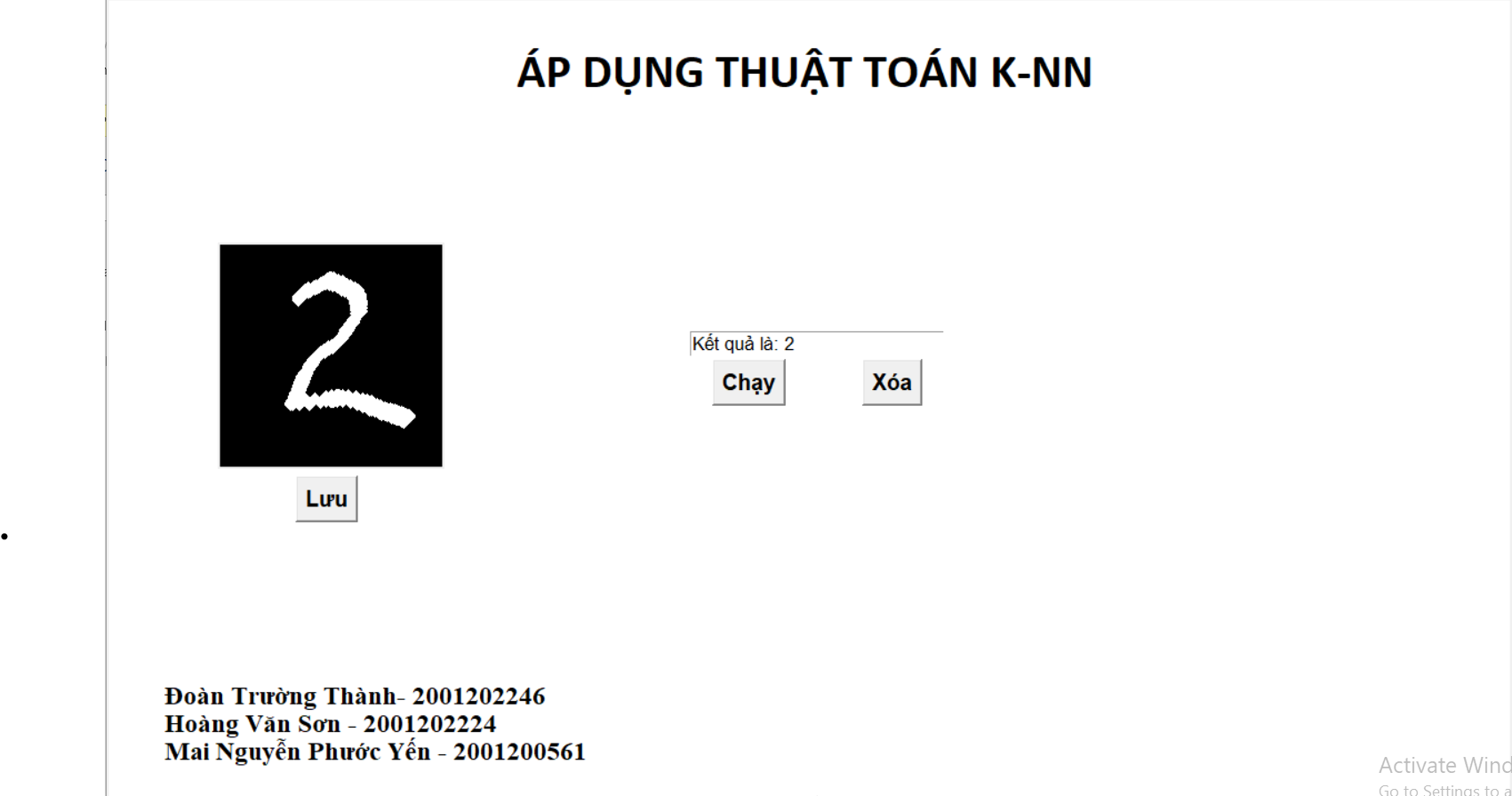
- Kết quả thử nghiệm nhận dạng số 1.



*Hình 3.3. Kết quả thử nghiệm nhận dạng số 1*

1. **Nhận dạng số 2.**

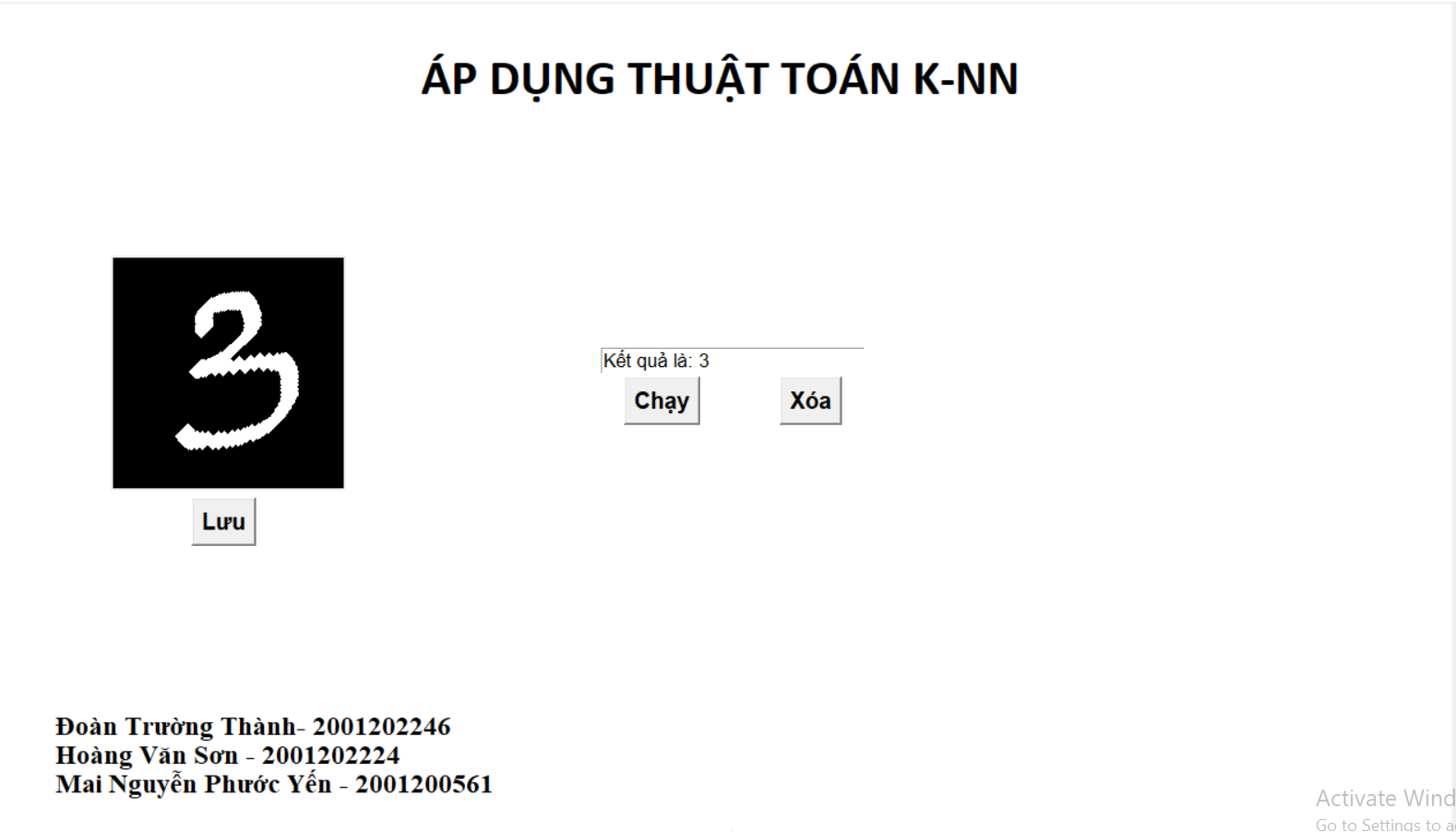
- Kết quả thử nghiệm nhận dạng số 2.



*Hình 3.4. Kết quả thử nghiệm nhận dạng số 2*

1. **Nhận dạng số 3.**

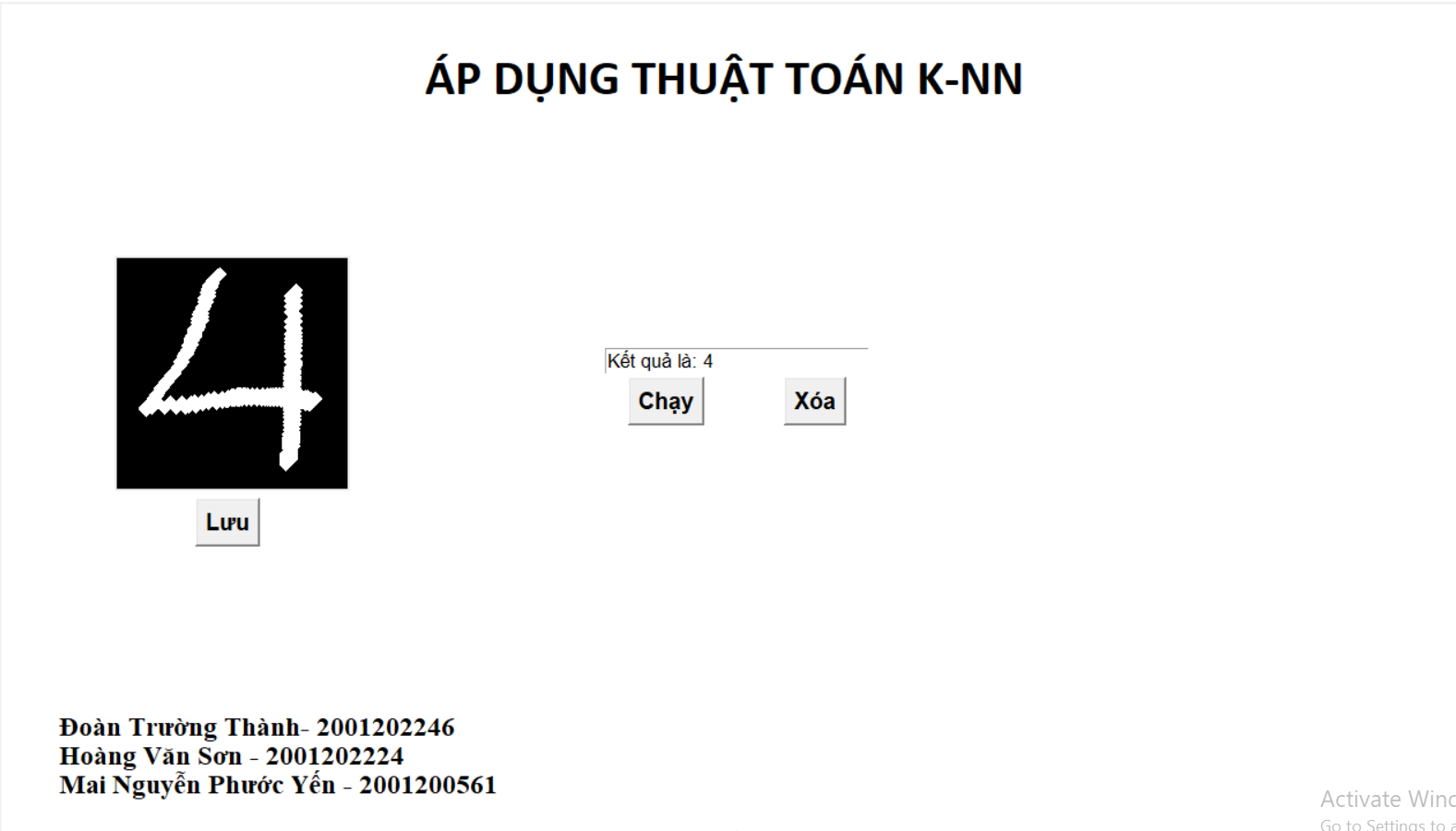
- Kết quả thử nghiệm nhận dạng số 3.



*Hình 3.5. Kết quả thử nghiệm nhận dạng số 3*

1. **Nhận dạng số 4.**

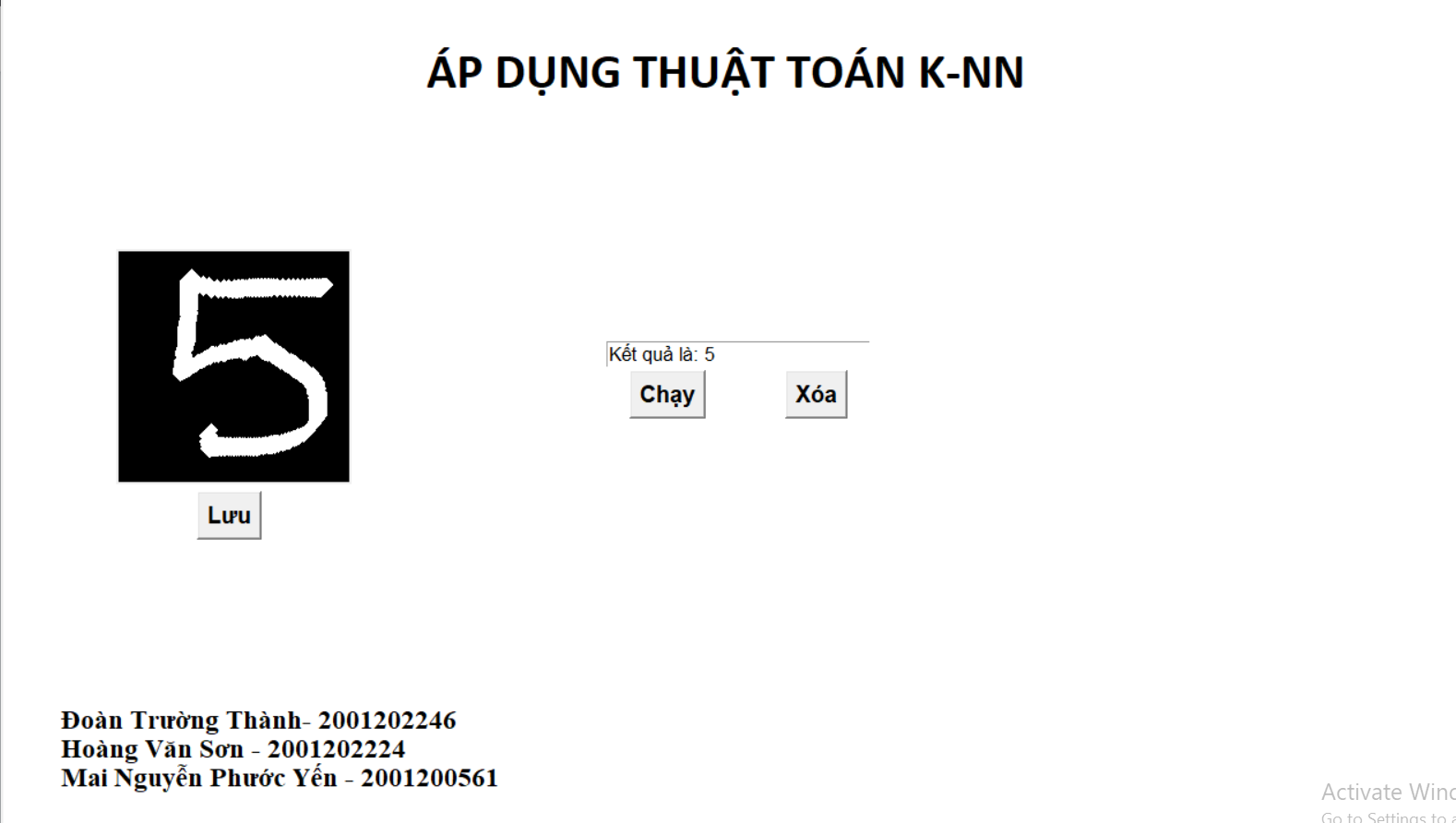
- Kết quả thử nghiệm nhận dạng số 4.



*Hình 3.6. Kết quả thử nghiệm nhận dạng số 4*

1. **Nhận dạng số 5.**

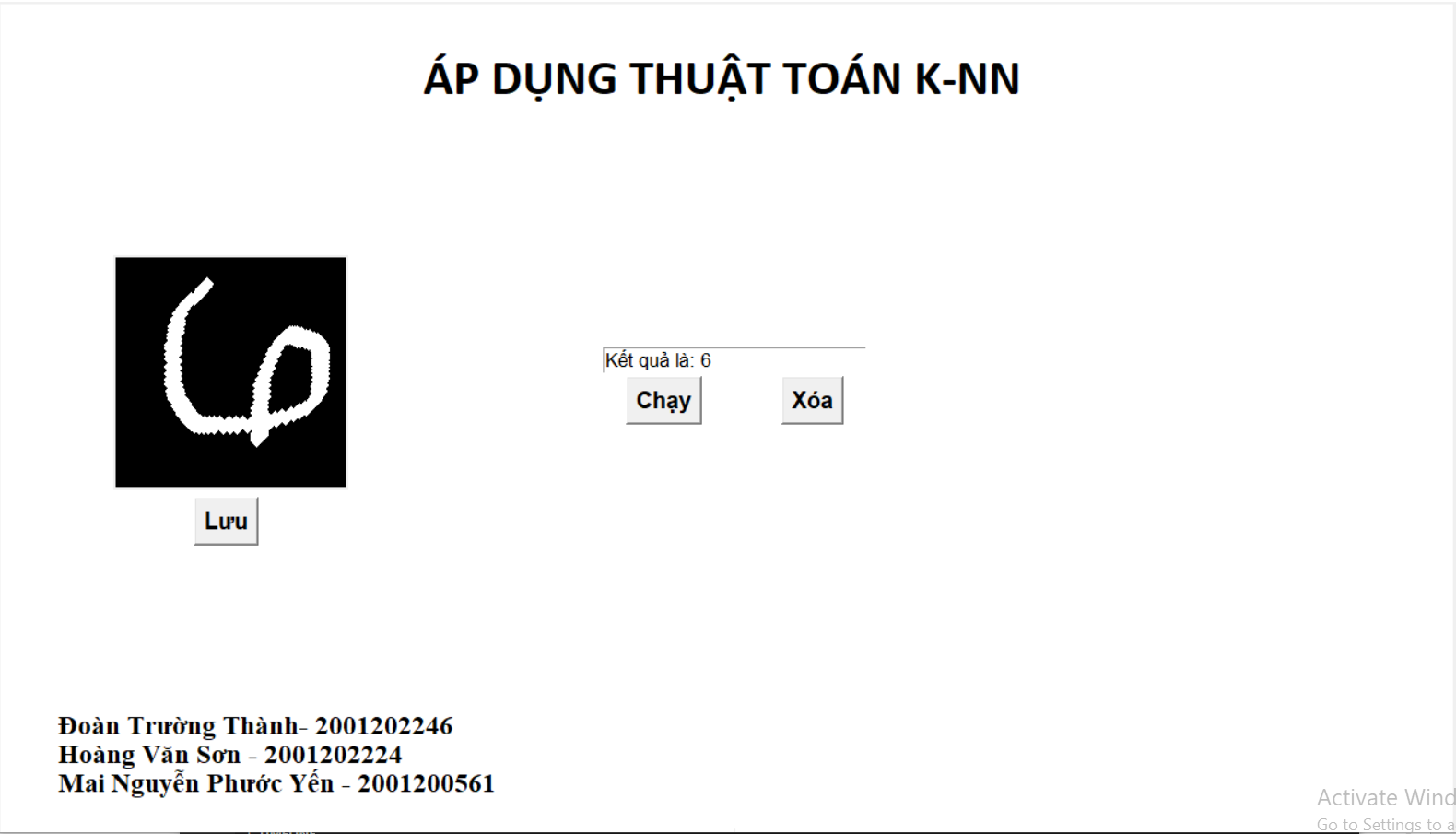
- Kết quả thử nghiệm nhận dạng số 5.



*Hình 3.7. Kết quả thử nghiệm nhận dạng số 5*

1. **Nhận dạng số 6.**

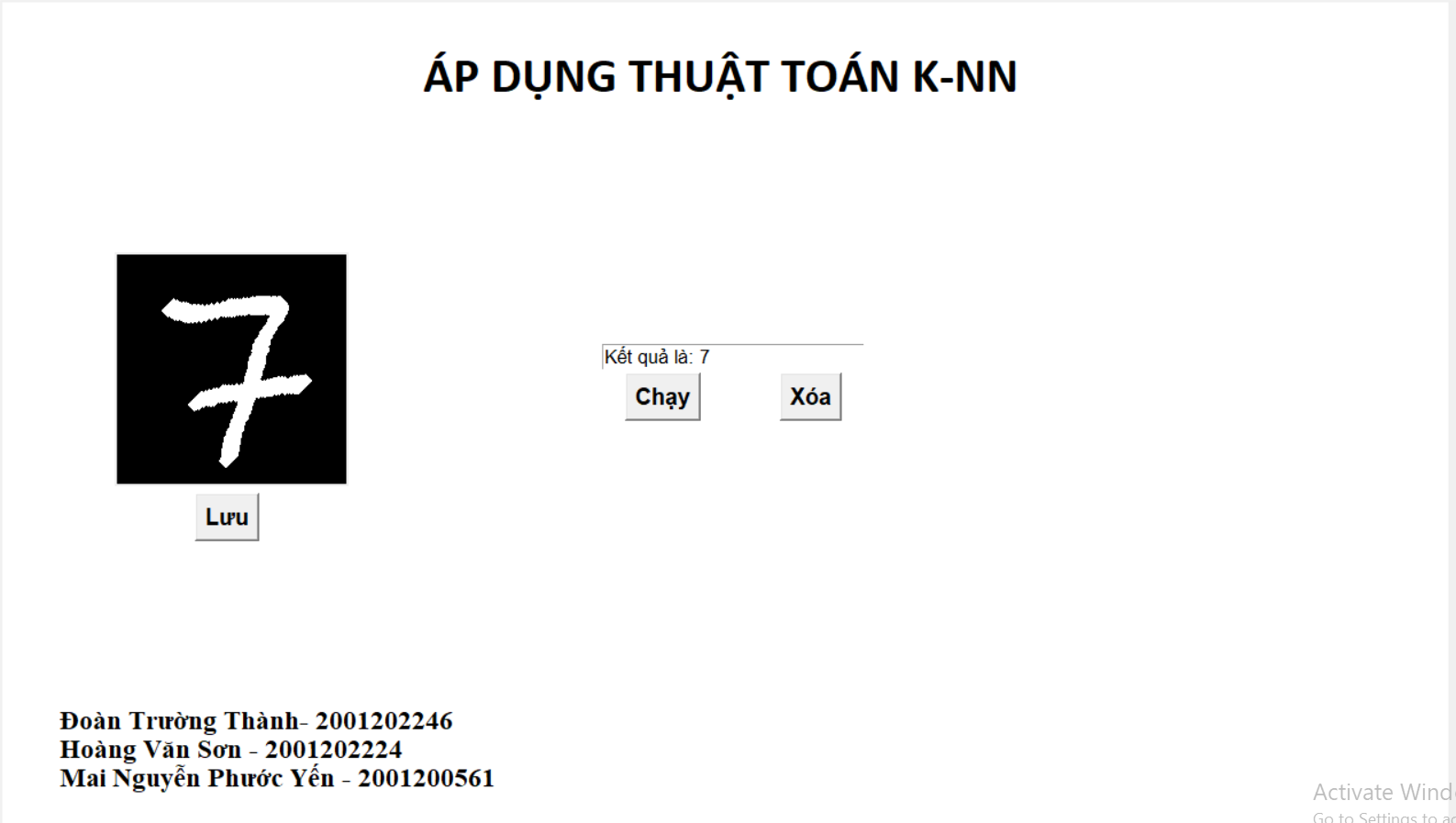
- Kết quả thử nghiệm nhận dạng số 6.



*Hình 3.8. Kết quả thử nghiệm nhận dạng số 6*

1. **Nhận dạng số 7.**

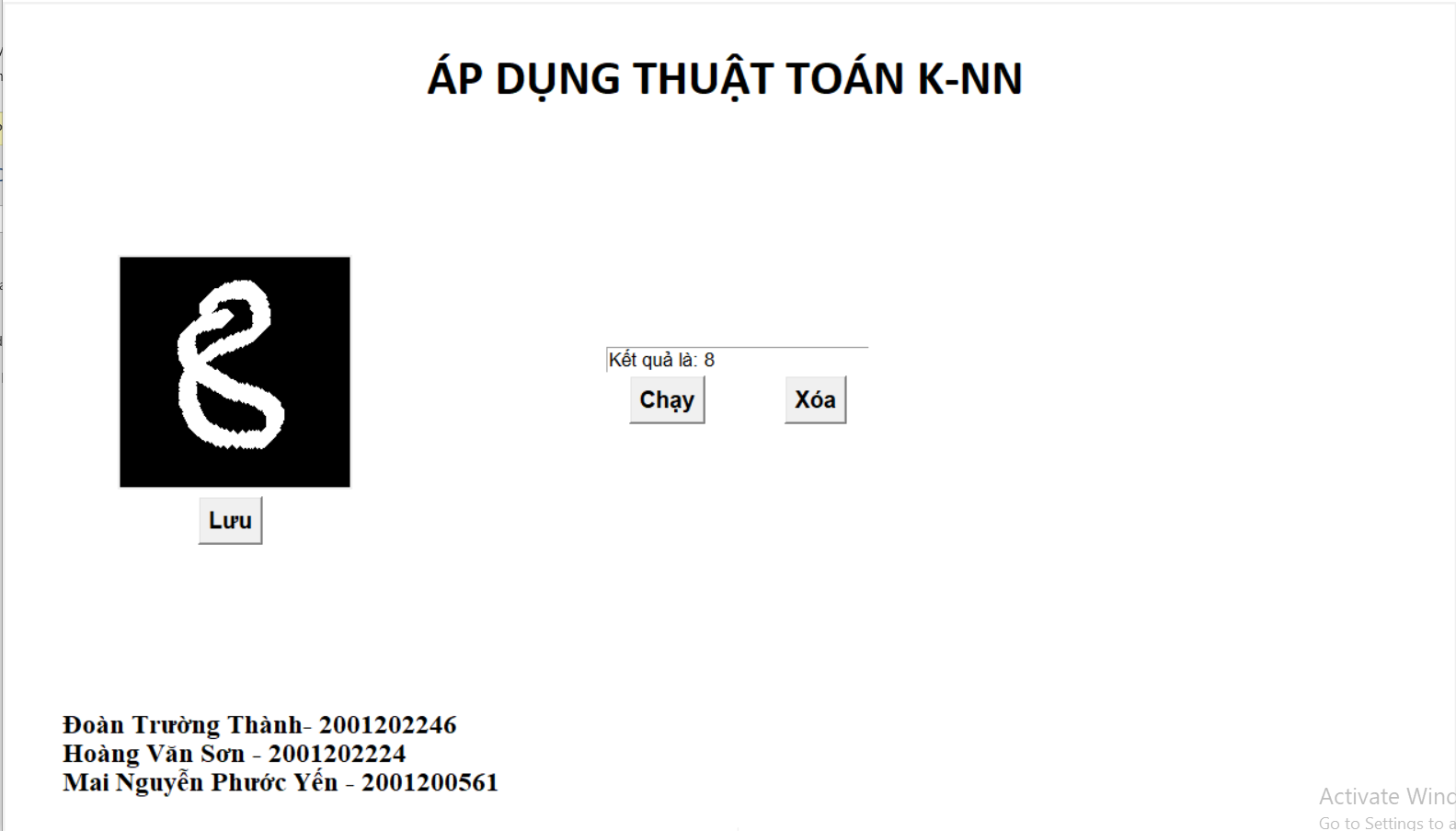
- Kết quả thử nghiệm nhận dạng số 7.



*Hình 3.9. Kết quả thử nghiệm nhận dạng số 7*

1. **Nhận dạng số 8.**

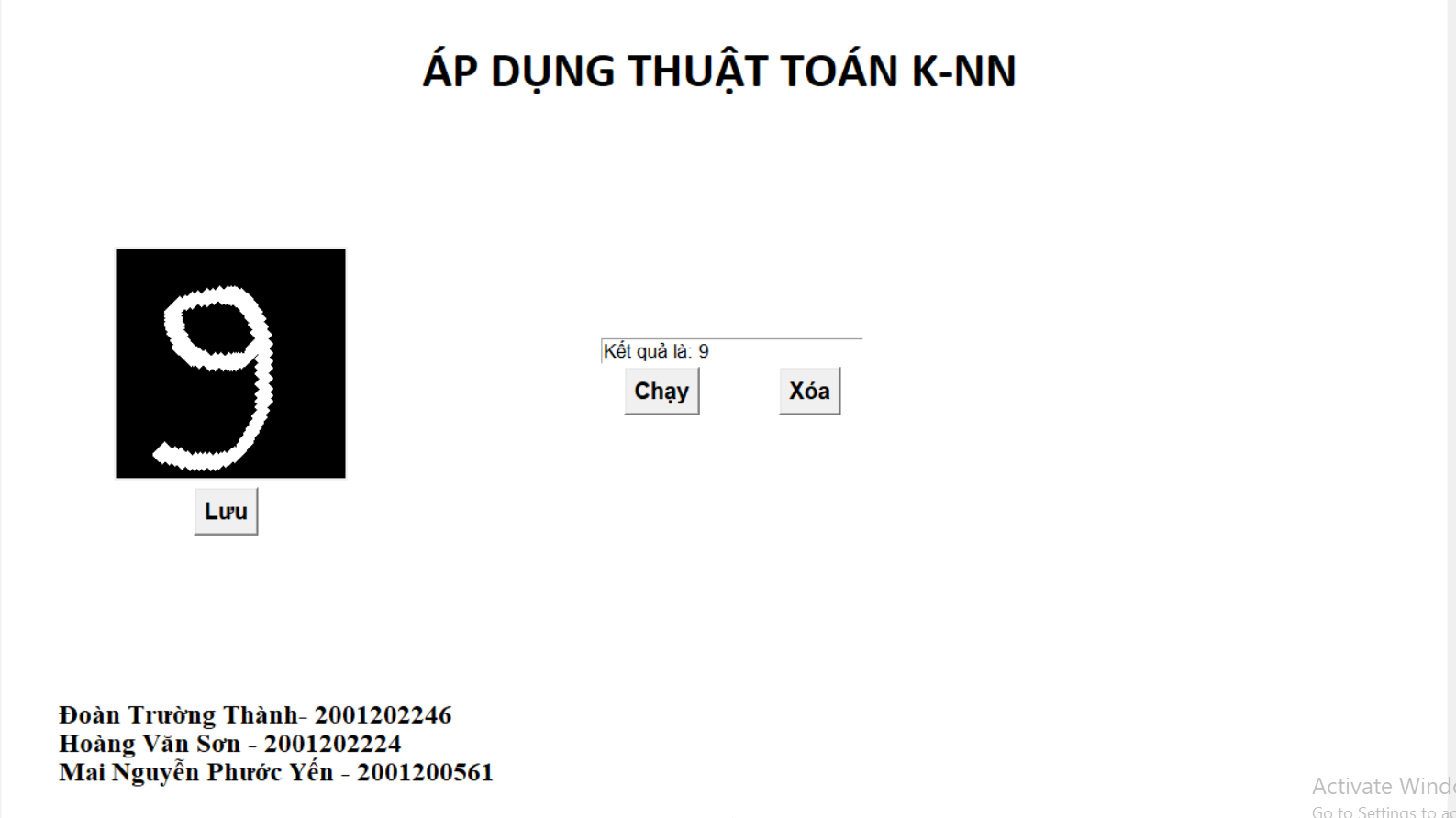
- Kết quả thử nghiệm nhận dạng số 8.



*Hình 3.10. Kết quả thử nghiệm nhận dạng số 8*

1. **Nhận dạng số 9.**

- Kết quả thử nghiệm nhận dạng số 9.



*Hình 3.11. Kết quả thử nghiệm nhận dạng số 9*

**Chương 4: KẾT QUẢ - KẾT LUẬN**

## 4.1. Kết quả đạt được

Qua quá trình tìm hiểu, kiểm nghiệm nhóm đã đạt được kết quả tốt cho trương trình sử dụng thuật toán KNN để nhận diện số viết tay.

- Nhận dạng được các số viết tay từ 0 đến 9.

- Nhận dạng một cách nhanh và chính xác.

## Hạn chế của đề tài

Đề tài mà nhóm đã thực hiện còn một số hạn chế như sau:

* Chỉ nhận dạng được số viết tay từ 0 đến 9
* Để chương trình có thể nhận dạng được một cách chính xác nhất người viết cần viết số mọt cách chính xác gần giống nhiều nhất với mẫu.
* Kết quả nhận dạng đôi lúc còn bị sai lệch.

## Hướng phát triển

Hướng phát triển đề tài của nhóm trong đề tài tiếp theo.

* Nhận dạng được nhiều dạng ký tự hơn, ví dụ: chữ viết, ký tự đặt biệt,...
* Nhận dạng được một cách nhanh chóng và chính xác nhất.
* Kết hợp với cánh tay robot để phối hợp viết và nhận dạng.

# TÀI LIỆU KHAM KHẢO

<https://truyenhinhcapsongthu.net/top/bai-11-nhan-dang-chu-viet-tay-su-dung-knn-phan-1-youtube/PYllhgHmIn05V0>

<https://www.youtube.com/watch?v=bPdlaIZ_NBc>

<https://machinelearningcoban.com/2017/01/08/knn/>

<https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn>

# PHỤ LỤC

**Code chương trình:**

import PIL

from PIL import ImageTk, Image, ImageDraw

from tkinter import \*

from PIL import Image,ImageTk

import numpy as np

import cv2

width = 200

height = 200

white = (255, 255, 255)

black = (0, 0, 0)

def RUN():

    # đọc ảnh train và nhan dạng

    img = cv2.imread('digits.png',0)

    imgNhanDang = cv2.imread('image\_20.png',0)

    # cắt cảnh thàng từng ô nhỏ

    cells = [np.hsplit(row, 100) for row in np.vsplit(img, 50)]

    # chuyển thành ma trận 2 chiều

    x = np.array(cells)

    x2 = np.array(imgNhanDang)

    # tạo du liệu train và du liệu test(nhandang)

    train = x[:,:50].reshape(-1, 400).astype(np.float32)

    test = x2.reshape(-1, 400).astype(np.float32)

    # dán nahxn cho du liệu test

    k = np.arange(10)

    train\_labels = np.repeat(k, 250)[:, np.newaxis]

    # Nhận dạng

    knn = cv2.ml.KNearest\_create()

    knn.train(train, 0, train\_labels)

    kq1, kq2, kq3, kq4 = knn.findNearest(test, 20)

    box.insert(END,"Kết quả là: {}".format(int(kq2)))

def CLEAR():

    box.delete(1.0,END)

    global image1, draw;

    cv.delete("all")

    image1 = PIL.Image.new("RGB", (width, height), black)

    draw = ImageDraw.Draw(image1)

def SAVE():

    filename = "image.png"

    image1.save(filename)

    image = Image.open('image.png')

    new\_image = image.resize((20, 20))

    new\_image.save('image\_20.png')

def paint(event):

    x1, y1 = (event.x - 3), (event.y - 3)

    x2, y2 = (event.x + 3 ), (event.y + 3)

    cv.create\_line(x1, y1, x2, y2, fill="white",width=15)

    draw.line([x1, y1, x2, y2],fill="white",width=15)

root = Tk()

root.geometry("1280x721+90+30")

root.title("NHÓM 16")

background=Image.open("GIAODIEN.png")

render=ImageTk.PhotoImage(background)

img3=Label(root, image=render)

img3.place(x=0, y=0)

cv = Canvas(root, width=width, height=height, bg='black')

cv.place(x=100, y= 240)

image1 = PIL.Image.new("RGB", (width, height), black)

draw = ImageDraw.Draw(image1)

cv.bind("<B1-Motion>", paint)

button\_frame=Frame(root).pack(side=BOTTOM)

save\_button=Button(button\_frame,text="save",font=(("Times New Romen"),15,'bold'),command=SAVE)

# button2=Button(text="Clear",command=Clear)

save\_button.place(x=170, y= 445)

# button2.pack(side = RIGHT)

box=Text(root, width=25, height=1, font=("Times New Romen",13))

box.pack(pady=300)

button\_frame=Frame(root).pack(side=BOTTOM)

run\_button=Button(button\_frame,text="run",font=(("Times New Romen"),15,'bold'),command=RUN)

run\_button.place(x=545, y= 325)

button\_frame=Frame(root).pack(side=BOTTOM)

clear\_button=Button(button\_frame,text="clear",font=(("Times New Romen"),15,'bold'),command=CLEAR)

clear\_button.place(x=680, y= 325)

root.mainloop()