BÁO CÁO ĐỒ ÁN CUỐI KHÓA

Nhận dạng chữ số viết tay với tập dữ liệu MNIST

A. Tóm tắt các đặc trưng và phương pháp học

1. Đặc trưng Raw Pixel

Sử dụng các pixel nguyên thủy mà không có sự biến đổi.

2. Đặc trưng HOG (Histogram of Oriented Gradient)

- HOG sử dụng hướng của các gradient để làm đặc trưng. Gradient của một ảnh được sử dụng để làm đặc trưng vì độ lớn (magnitude) của gradient là lớn tại các cạnh và góc và cạnh, góc là nơi chứa nhiều thông tin hơn so với các vùng ảnh phẳng.
- **Tính toán đặc trưng HOG**
  - **Bước 1: Tính toán gradient cho ảnh**

    + Đầu tiên trong tính toán HOG là tính gradient ảnh theo chiều ngang (chiều x, ký hiệu G_x) và dọc (chiều y, ký hiêu G_y) bằng cách sử dụng 2 bộ lọc ảnh tương ứng bên dưới

-1	0	1

-1
0
1

Hình 1: *Bộ lọc được sử dụng để tính gradient*

+ Sau đó, tìm độ lớn (magnitude) của gradient bằng công thức:

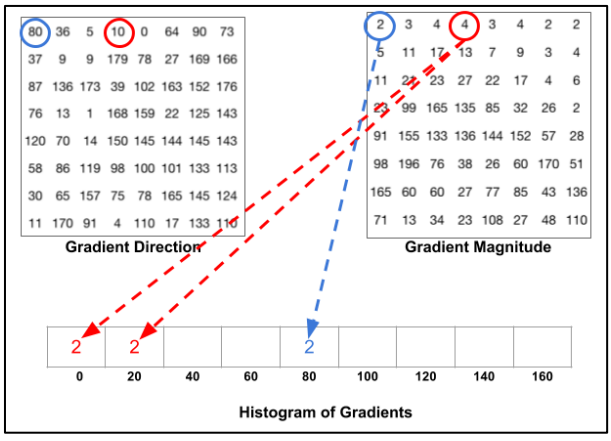
$$|G| = \sqrt{G_x^2 + G_y^2}$$

và hướng của gradient tại mỗi pixel trong ảnh theo công thức:

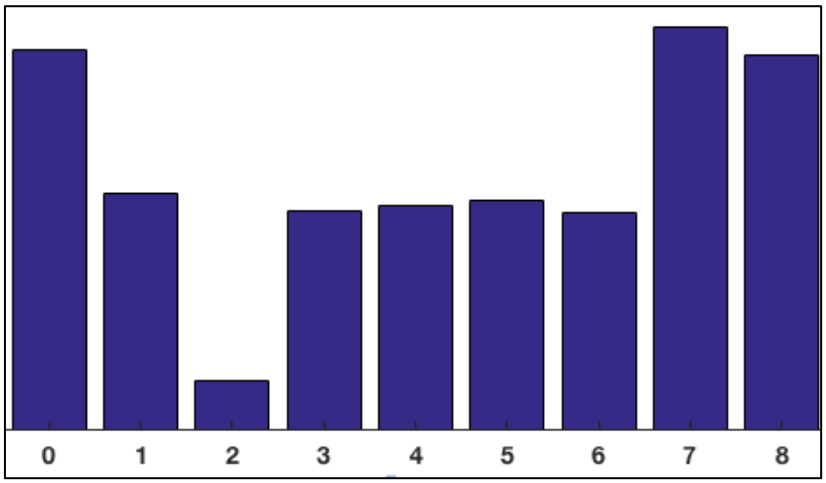
$$\theta = arctan2(Gy, Gx)$$

  - **Bước 2: Tính toán Histogram of Gradient trong các cell 8 x 8**

Ở bước này, ảnh ban đầu được chia thành các ô (cell) có kích thước 8x8. Histogram cần tính thường được chọn là có 9 bin, tương ứng với các góc (angle): 0, 20, 40, 60, 80, 100, 120, 140, 160. Dựa vào độ lớn của gradient (Gradient Magnitude) và hướng của gradient để tính ra histogram như hình bên dưới:



Hình 2: *Minh họa việc xây dựng Histogram of Gradient*

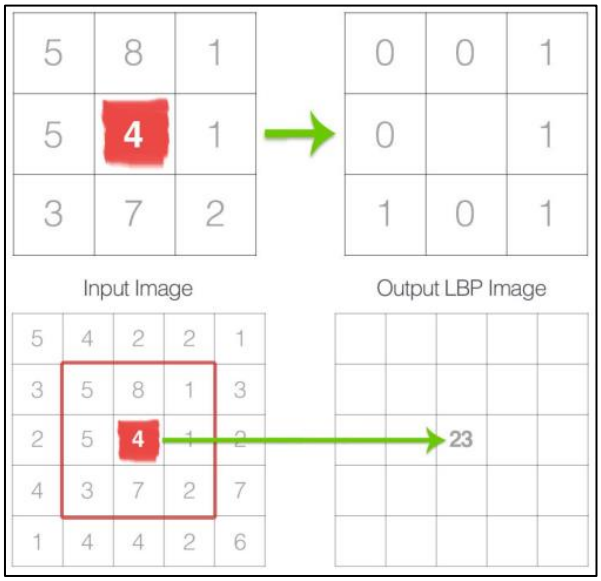


Hình 3: *Histogram sau khi được xây dựng*

- **Bước 3: Chuẩn hóa (Normalization)**
  Việc chuẩn hóa nhằm làm cho histogram không bị ảnh hưởng bởi thay đổi độ sáng. Thực tế cho thấy sử dụng các block có kích thước 2x2 cell hoặc 3x3 cell sẽ tăng độ chính xác cho kết quả.
- **Bước 4: Tính toán vector đặc trưng HOG**
  Để tính ra vector đặc trưng cuối cùng, kết nối tất cả các vector đã chuẩn hóa ở bước 3 lại với nhau.

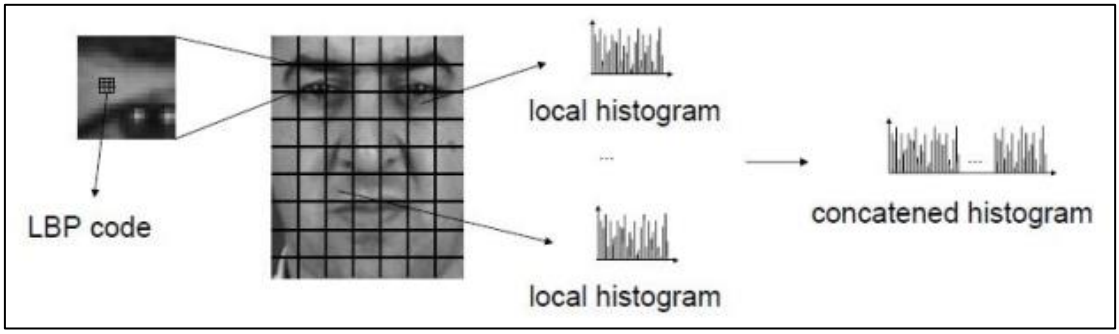
3. **Đặc trưng LBP (Local Binary Pattern):** Đặc trưng này được thực hiện theo các bước sau:

- Chia ảnh đầu vào thành các cell, chẳng hạn: 16x16 pixel
- Đối với mỗi pixel trong cell, so sánh mỗi pixel với 8 giá trị pixel lận cận theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ, nếu giá trị pixel trung tâm lớn hơn giá trị lận cận ta ghi "1", ngược lại ghi "0". Từ đó, ta sẽ được 1 con số nhị phân 8-bit và được chuyển sang 1 con số thực, con số sẽ được ghi tại vị trí pixel trung tâm của cell như minh họa bên dưới:



Hình 4: *Tính toán giá trị tại mỗi pixel với LBP*

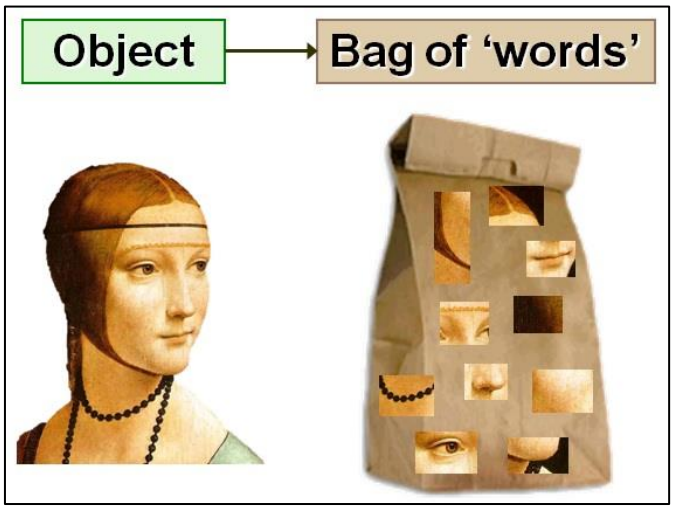
- Tính toán histogram trên mỗi cell vừa thực hiện, chính là số lần xuất hiện của các số.
- Cuối cùng là nối tất cả histogram trên tất cả cell sẽ nhận được vector đặc trưng cần tìm như hình:



Hình 5: *Histogram cuối được tạo bằng cách nối các histogram trên từng cell*

4. Đặc trưng BoW (Bag Of Word)

- BoW là một phương pháp phổ biến cho việc phân lớp ảnh lấy cảm từ mô hình trong xử lý ngôn ngữ tự nhiên.
- Một bag of visual word là một vector thể hiện số lần xuất hiện các từ vựng (vocabulary) của các đặc trưng ảnh cục bộ.

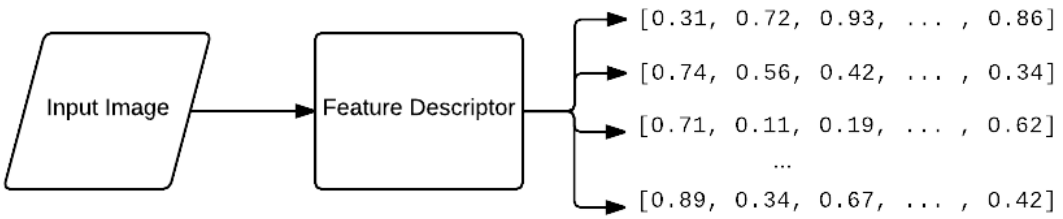


Hình 6: *Ý tưởng của BoW*

- **Xây dựng BoW:**
  - **Bước 1: Trích xuất đặc trưng**

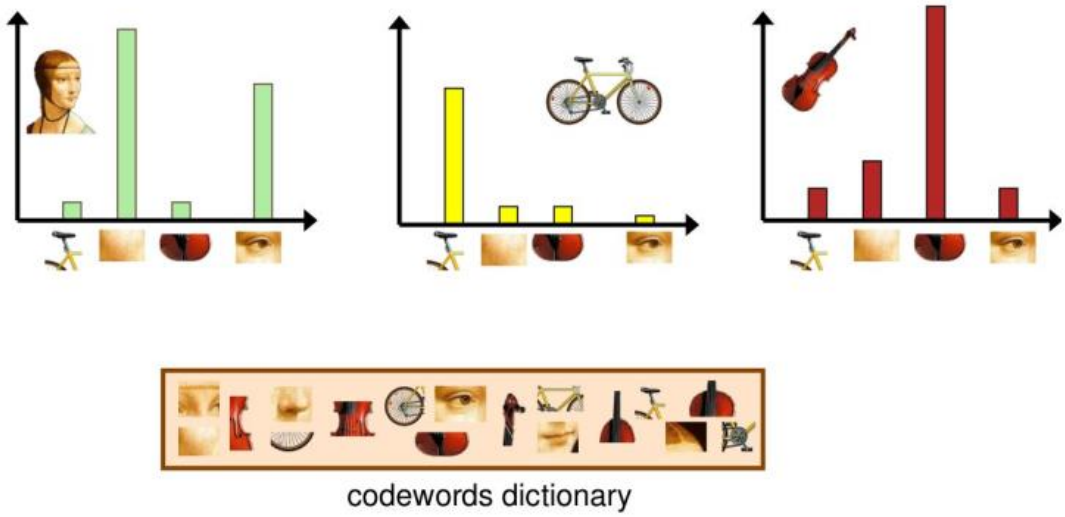
    + Bước đầu tiên trong xây dựng BoW là thực hiện trích xuất đặc trưng bằng các bộ mô tả trích xuất (extracting descriptor) từ mỗi trong tập dữ liệu.

    + Có nhiều cách thực hiện bao gồm: nhận diện keypoint (detecting keypoint) và trích xuất đặc trưng SWIFT từ các vùng nổi bật trong ảnh. Kết quả nhận được là nhiều vector đặc trưng cần tìm.



Hình 7: *Sử dụng các bộ mô tả đặc trưng ta nhận được nhiều vector đặc trưng cho mỗi ảnh*

- **Bước 2: Xây dựng từ vựng (Vocubulary)**
  Từ các vector đặc trưng đã rút trích ở bước 1, chúng ta xây dựng từ vựng bằng cách sử dụng giải thuật gom cụm k-mean. Kết quả các cụm trung tâm được chính là từ điển các từ (dictionary of visual words)
- **Bước 3: Lượng hóa vector**
  Cho một ảnh bất kỳ (từ tập dữ liệu của chúng ta hoặc không), chúng ta có thể lượng hóa và biểu diễn ảnh sử dụng mô hình BoW theo các bước sau:
  + Trích xuất đặc trưng như ở bước 1
  + Từ mỗi vector đã trích xuất, tính toán láng giềng gần nhất của nó trong từ điển đã xây dựng ở bước 2, thường sử dụng khoảng cách Euclidean để làm.
  + Lấy tập các nhãn gần nhất và xây dựng một histogram có chiều dài k (k là số lượng cụm được sinh ra từ k-mean), ở đó giá trị thứ i trong historgram là số lần của từ (word) thứ i.



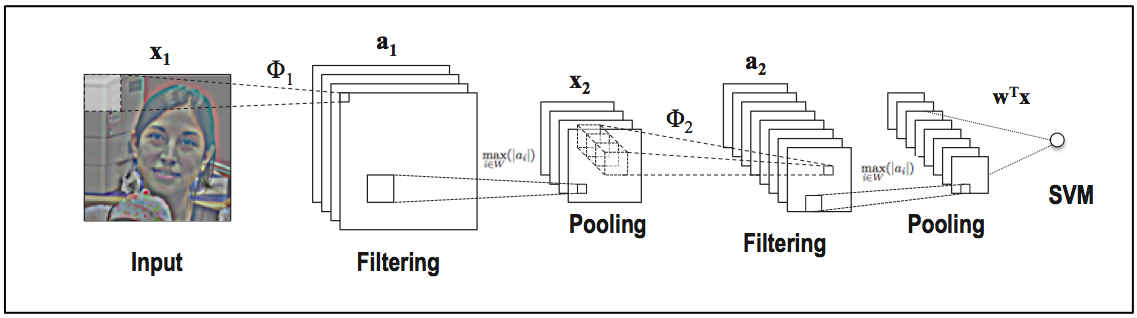
Hình 8: Lượng hóa các ảnh đầu vào thành 1 histogram dựa trên từ điển đã xây dựng

+ Cuối cùng với bag of word đã có, chúng ta biểu diễn tất cả các ảnh trong tập dữ liêu. Sau đó có thể ứng dụng kỹ thuật máy học phổ biến hoặc giải thuật CBIR để phân lớp hoặc truy tìm ảnh.

5. Deep Learning

- Deep Learning là kĩ thuật tiên tiến nhất (the state-of-the-art) trong computer vision hiện nay.

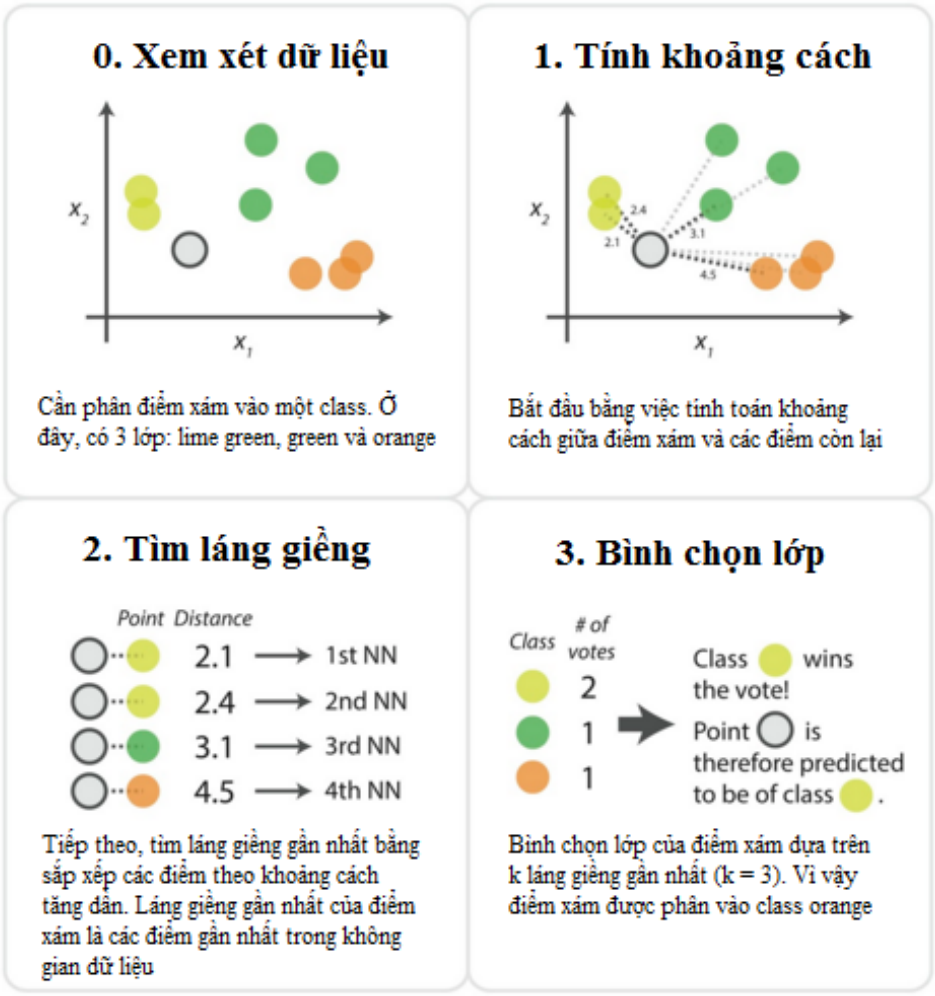
- Deep Learning là một nhánh của Machine Learning. Trong đó, các model là các cấu trúc đồ thị (networks) với nhiều tầng (multiple layers), các model này là phi tuyến tính (non-linear).
- Cả 2 dạng học có giám sát (supervised) và không giám sát (unsupervised) đều được sử dụng để xây dựng mô hình dữ liệu.
- **Ý tưởng:** Deep Learning cho phép các mô hình tính toán được tạo thành từ nhiều lớp (layer) xử lý để học việc biểu diễn dữ liệu với nhiều mức độ trừu tượng. Phương pháp này cải thiện đáng kể trong lĩnh vực nhận dạng giọng nói, nhận dạng đối tượng, do tìm đối tượng và nhiều lĩnh vực khác. Deep Learning phát hiện cấu trúc phức tạp trong tập dữ liệu lớn bằng giải thuật lan truyền ngược để xác định làm thế nào machine thay đổi các tham số nội tại của nó, những cái được sử dụng để tính toán việc biểu diễn trong mỗi lớp (layer) từ sự biểu diễn của lớp (layer) trước đó. Các deep convolution net đưa đến bước đột phá trong xử lý ảnh, giọng nói, video và audio, trong khi mạng hồi quy (recurrent net) chỉ tốt trên các dữ liệu tuần tự như text và giọng nói.



Hình 9: *Cấu trúc tổng quát deep learning*

6. Phương pháp học KNN (K-Nearest Neighbor)

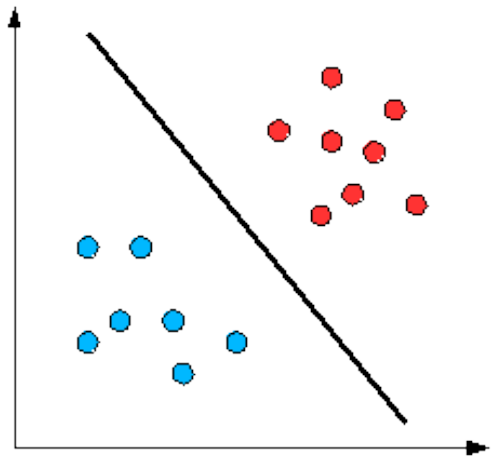
- K-Nearest Neighbors (KNN) là một trong những giải thuật học có giám sát đơn giản nhất trong Machine Learning. K-NN là phương pháp để phân lớp các đối tượng dựa vào khoảng cách gần nhất giữa đối tượng cần xếp lớp (Query point) và tất cả các đối tượng trong Training Data.
- Một đối tượng được phân lớp dựa vào K láng giềng của nó. K là số nguyên dương được xác định trước khi thực hiện thuật toán. Người ta thường dùng khoảng cách Euclidean để tính khoảng cách giữa các đối tượng.
- **Thuật toán K-NN được mô tả như sau:**
  - Xác định giá trị tham số K (số láng giềng gần nhất)
  - Tính khoảng cách giữa đối tượng cần phân lớp (Query Point) với tất cả các đối tượng trong training data (thường sử dụng khoảng các Euclidean)
  - Sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với Query Point
  - Lấy tất cả các lớp của K láng giềng gần nhất đã xác định
  - Dựa vào phần lớn lớp của láng giềng gần nhất để xác định lớp cho Query Point
- **Ví dụ giải thuật K-NN**



Hình 10: *Minh họa KNN*

7. Phương pháp học SVM (Support Vector Machine)

Support Vector Machine (SVM) là một giải thuật máy học có giám sát được sử dụng phổ biến cho các bài toán phân lớp. Trong giải thuật, mỗi hạng mục dữ liệu (data item) được biểu diễn dưới dạng một điểm trong không gian n chiều (n thường là số đặc trưng và giá trị của mỗi đặc trưng là giá trị của một tọa độ cụ thể). Sau đó, dữ liệu được phân lớp bằng cách tìm ra một siêu phẳng (hyper-plane) để tách biệt 2 lớp là tốt nhất.



Hình 11: *Phân lớp với Support Vector Machine*

B. Chức năng của các tập tin, cách chạy chương trình, kết quả thực nghiệm

1. Các tập tin dùng chung:

STT	Tên tập tin	Mô tả
1	train-images.idx3-ubyte train-labels.idx1-ubyte t10k-images.idx3-ubyte t10k-labels.idx1-ubyte	Các tập tin chứa dữ liệu chữ số viết tay mẫu
2	loadMNISTImages.m	Đọc dữ liệu ảnh train hoặc test
3	loadMNISTLabels.m	Đọc dữ liệu nhãn train hoặc test
4	loadData.m	Tập tin chứa một hàm dùng để load dữ liệu train hoặc test
5	CreateDataTrain.m	Tập tin chứa chương trình dùng để phân loại dữ liệu train chữ số viết tay ban đầu vào các thư mục tương ứng từng số từ 0 → 9
6	CreateDataTest.m	Tập tin chứa chương trình dùng để phân loại dữ liệu test chữ số viết tay ban đầu vào các thư mục tương ứng từng số từ 0 → 9
7	categoryClassifier.mat	Một model được xây dựng dựa trên đặc trưng Bag Of Features, được dùng để hỗ trợ chạy chương trình dựa trên đặc trưng Bag Of Features

2. RawKNN.m

- Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng raw pixel với giải thuật KNN (sử dụng các giá trị thiết lập mặc định của KNN)
- Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **RawKNN**
- **Kết quả thực nghiệm:**

  + Distance = ‘euclidean’

K	Kết quả
1	9691
2	9612
3	9706

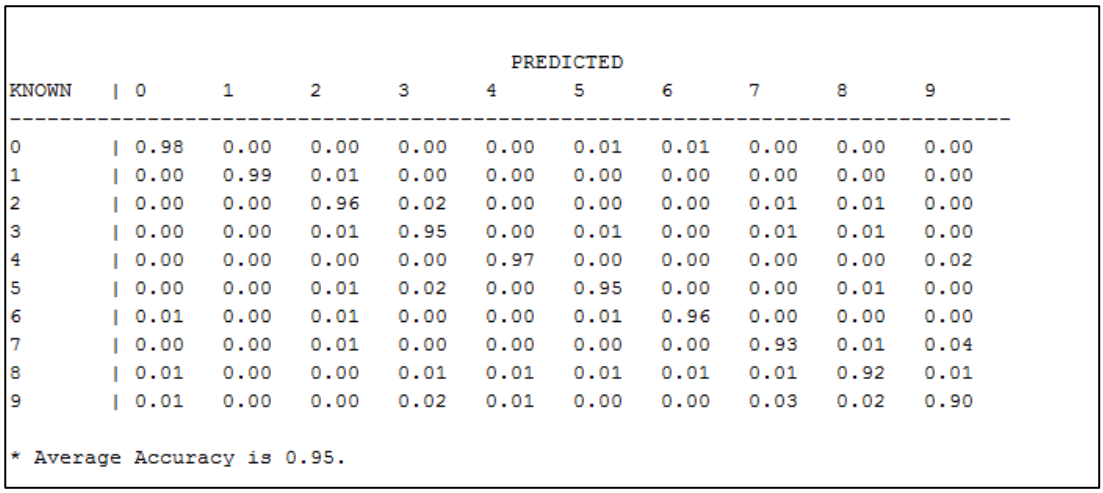
+ Distance = ‘cityblock’

K	Kết quả
1	9631
2	9528
3	9631

3. **RawSVM.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng raw pixel với giải thuật SVM (sử dụng các giá trị thiết lập mặc định của SVM)
   - Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **RawSVM**
   - **Kết quả: 9438 mẫu đúng**
4. **HogKNN.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng HOG với giải thuật KNN (sử dụng các giá trị thiết lập mặc định của HOG và KNN)
   - Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **HogKNN**
   - **Kết quả thực nghiệm: 9709 mẫu đúng**
5. **HogSVM.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng HOG với giải thuật SVM (sử dụng các giá trị thiết lập mặc định của HOG và SVM)
   - Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **HogSVM**
   - **Kết quả thực nghiệm: 9794 mẫu đúng**
6. **LBPKNN.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng LBP với giải thuật KNN (sử dụng các giá trị thiết lập mặc định của LBP và KNN)
   - Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **LBPKNN**
   - **Kết quả thực nghiệm: 6697 mẫu đúng**
7. **LBPSVM.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng LBP với giải thuật SVM (sử dụng các giá trị thiết lập mặc định của LBP và SVM)
   - Chạy chương trình: Mở file trong MATLAB cùng với các function và dữ liệu cần thiết, trong cửa sổ Command Window gõ: >> **LBPSVM**

- **Kết quả thực nghiệm: 6785 mẫu đúng**

8. **Bag_Of_Features.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng đặc trưng Bag Of Word
   - Chạy chương trình:
     - Bước 1: Chạy file CreateDataTrain.m để phân loại số trong dữ liệu train vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
     - Bước 2: Chạy file CreateDataTest.m để phân loại số trong dữ liệu test vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
     - Bước 3: Chạy file **Bag_Of_Features.m** để kết xuất kết quả
   - **Kết quả thực nghiệm:**



	PREDICTED									
KNOWN	0	1	2	3	4	5	6	7	8	9
0	0.98	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00
1	0.00	0.99	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.96	0.02	0.00	0.00	0.00	0.01	0.01	0.00
3	0.00	0.00	0.01	0.95	0.00	0.01	0.00	0.01	0.01	0.00
4	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.02
5	0.00	0.00	0.01	0.02	0.00	0.95	0.00	0.00	0.01	0.00
6	0.01	0.00	0.01	0.00	0.00	0.01	0.96	0.00	0.00	0.00
7	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.93	0.01	0.04
8	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.92	0.01
9	0.01	0.00	0.00	0.02	0.01	0.00	0.00	0.03	0.02	0.90

* Average Accuracy is 0.95.

9. **DeepLearningSVM.m**
   - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng kỹ thuật Deep Learning và phương pháp học SVM
   - Chạy chương trình:
     - Bước 1: Chạy file CreateDataTrain.m để phân loại số trong dữ liệu train vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
     - Bước 2: Chạy file CreateDataTest.m để phân loại số trong dữ liệu test vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
     - Bước 3: Chạy file **DeepLearningSVM.m** để kết xuất kết quả
   - **Kết quả thực nghiệm: 9886 mẫu đúng**

10. **DeepLearningKNN.m**
    - Mô tả: Tập tin này định nghĩa chương trình kết xuất ra số lượng chữ số viết tay được dự đoán đúng dùng kỹ thuật Deep Learning và phương pháp học KNN
    - Chạy chương trình:
      - Bước 1: Chạy file CreateDataTrain.m để phân loại số trong dữ liệu train vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
      - Bước 2: Chạy file CreateDataTest.m để phân loại số trong dữ liệu test vào các thư mục tương ứng từ 0 → 9 (nếu chưa có)
      - Bước 3: Chạy file **DeepLearningKNN.m** để kết xuất kết quả
    - **Kết quả thực nghiệm: 9791 mẫu đúng**

C. Bảng tổng hợp kết quả

Bảng sau thể hiện sự so sánh trong kết quả dự đoán đúng các mẫu trong tập dữ liệu test chữ số viết tay trên các đặc trưng: Raw pixel, HOG, LBP, Deep Learning dựa trên 2 phương pháp học: KNN và SVM, sử dụng các thiết lập mặc định trong MATLAB.

	Raw pixel	**HOG**	**LBP**	**BoW**	**Deep Learning**
KNN	9691	9709	6697	x	9719
SVM	9438	9794	6785	95000	9886

x: chưa có kết quả