ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI THU HOẠCH

NHẬN DẠNG THỊ GIÁC VÀ ỨNG DỤNG

ĐỀ TÀI

NHẬN DẠNG CHỮ SỐ VIẾT TAY VỚI TẬP DỮ LIỆU MNIST

GVHD:

TS. LÊ ĐÌNH DUY

TS. NGUYỄN TẤN TRẦN MINH KHANG

HVTH:

TRẦN QUANG KHẢI CH1502007

--- Năm 2017 ---

MỤC LỤC

1. Links

a. Github: https://goo.gl/uQAosj

b. Youtube:

- https://youtu.be/CJoI-Cvs8w4 (MNIST)
- https://youtu.be/ua6GtnA9ZYg (YOLO)

2. Tóm tắt

2.1. Mục tiêu

Báo cáo đồ án cuối kỳ hướng đến việc tổng hợp các kiến thức đã tiếp thu được trong từng bài giảng môn "Nhận dạng thị giác và ứng dụng" của TS. Lê Đình Duy và TS. Nguyễn Tấn Trần Minh Khang. Qua đó, định hướng – phát triển khả năng sử dụng Matlab trong các bài toán về nhận dạng và khả năng phát triển các ý tưởng về lĩnh vực này.

Đồ án giúp em có thể tự tổ chức chương trình rõ ràng hơn qua việc chia nhỏ chương trình thành những tập tin theo chức năng của chúng. Đồ án còn giúp em phát triển khả năng đọc tài liệu, tự tìm hiểu và nghiên cứu qua báo cáo mở rộng qua chủ đề YOLO – Real-Time Object Detections.

Môn học không chỉ cung cấp kiến thức mà còn khơi gợi những ý tưởng trong việc vận dụng nhận dạng vào các lĩnh vực đời sống xã hội từ đó mở ra nhiều hướng mới trong phát triển và nghiên cứu.

2.2. Nội dung

Báo cáo tập trung trình bày về các kết quả đạt được về rút trích các đặc trưng được yêu cầu Pixel Intensity, HoG, LBP, BoW, Deep Features cùng việc sử dụng các phương pháp học máy như KNN, SVM, Deep Learning trong việc huấn luyện và tính độ chính xác của từng mô hình nhận dạng chữ viết tay.

Qua việc thực hiện đồ án, mã nguồn chương trình còn được tổ chức tốt hơn theo từng yêu cầu và nhiệm vụ cụ thể từ đó dễ dàng theo dõi, chỉnh sửa và mở rộng chương trình nếu có điều kiện.

Trong phần báo cáo mở rộng, giới thiệu những nét nổi bật và cách thức thực hiện một chương trình demo mẫu cho quá trình tìm hiểu YOLO.

2.3. Cách thực hiện

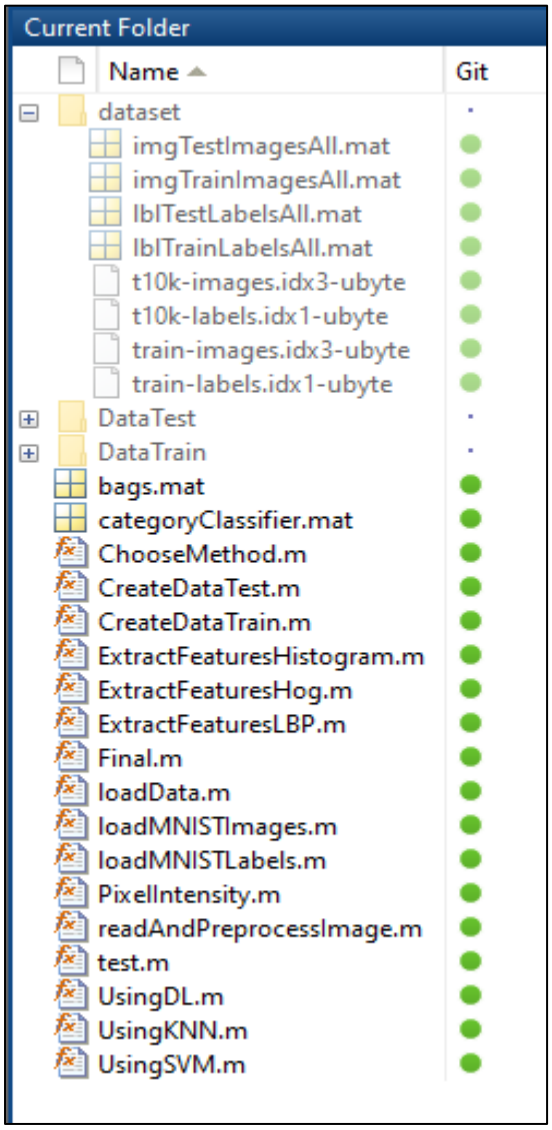
Trước khi thực hiện báo cáo đồ án cuối kỳ, em đã trải qua thời gian xem lại các bài tập đã thực hiện trên lớp và bài tập về nhà được tải lên GitHub để có thể hệ thống lại các phần cần thực hiện và đề ra cấu trúc chương trình cần xây dựng.

Bên cạnh đó, em còn lên trang Matlab để tìm hiểu các tham số trong từng hàm rút trích đặc trưng từ đó đưa ra chiến lược thử nghiệm các tham số và đánh giá kết quả đạt được. Một phần không thể thiếu là việc thực hiện báo cáo mở rộng bằng cách lên trang YOLO để tìm hiểu và xây dựng một mô hình mẫu thử cho việc thể nghiệm mô hình YOLO

Từ những bước thực hiện bên trên, cuối cùng là viết tổng hợp kết quả và báo cáo đồ án cuối kỳ, quay phim và đăng tải.

3. Nội dung chi tiết

3.1. Cấu trúc chương trình



➢ Cấu trúc chương trình gồm thư mục chứa bộ dữ liệu chữ số viết tay được đặt trong thư mục **dataset** gồm bộ dữ liệu test và bộ dữ liệu huấn luyện.

➢ Các thư mục **DataTest** và **DataTrain** chứa bộ dữ liệu chữ số viết tay được sắp xếp vào trong các thư mục tương ứng với nhãn theo đúng định dạng tên ảnh là thứ tự ảnh "**image_xxxxx.jpg**" trong bộ dữ liệu.

➢ **bags.mat** chứa kết quả trung gian trong việc rút trích đặc trưng bag of features.

➢ **categoryClassifier.mat** chứa kết quả trong quá trình xử lý phân loại ảnh huấn luyện vào trong các lớp tương ứng.

➢ **ChooseMethod.m** là phương thức cho phép người dùng lựa chọn các phương pháp huấn luyện dữ liệu gồm KNN, SVM và Deep Learning.

➢ Các tập tin **ExtractFeatures** có nhiệm vụ rút trích các đặc trưng theo Histogram, Hog, LBP nhằm phục vụ qua việc huấn luyện dữ liệu.

➢ **Final.m** chứa chương trình chính của đồ án cuối kỳ.

➢ **loadData.m** là phương thức nạp các bộ dữ liệu chữ số viết tay vào trong chương trình để xử lý, ta có thể nạp dữ liệu bằng cách cung cấp tên bộ dữ liệu cần nạp.

➢ **loadMNISTImages.m** và **loadMNISTLabels.m** là thư viện MNIST để đọc dữ liệu chữ số viết tay.

➢ **CreateDataTest.m** và **CreateDataTrain.m** chứa phương thức tạo ra hai bộ dữ liệu phát sinh từ dữ liệu chữ số viết tay vào trong hai thư mục DataTest và DataTrain.

3.2. Mô tả features – machine learning methods

3.2.1. Dữ liệu thô - Raw (pixel intensity)

Pixel intensity sử dụng đặc trưng histogram để huấn luyện cho bộ dữ liệu với tham số **Bin** được người dùng cung cấp sử dụng **fitcknn** để huấn luyện dữ liệu với các tham số **"NumNeighbors"** (mặc định k = 1) cho phép xác định số lượng láng giềng gần nhất để phân lớp khi dự đoán. Bên cạnh đó cho phép người dùng lựa chọn **"Standardize"** (mặc định false)để chuẩn hoá các dự đoán bằng cách tuỳ chỉnh các cột dữ liệu dự báo theo giá trị trung bình cột và độ lệch chuẩn tương ứng.

Bảng khảo sát với fitcknn

STT	Bin	K	Standardize	Mẫu đúng
1	16	1	false	2635
2	64	1	false	2799
3	128	1	false	2870
4	256	1	false	**3039**
5	16	2	false	2632
6	64	2	false	2777
7	128	2	false	2829
8	256	2	false	2908
9	16	2	true	2583
10	64	2	true	2420
11	128	2	true	2115
12	256	2	true	1674
13	16	3	false	2668
14	64	3	false	2874
15	128	3	false	2829
16	256	3	false	**3039**
17	16	3	true	2667
18	64	3	true	2430
19	128	3	true	2075
20	256	3	true	1651

Bảng khảo sát với SVM

STT	Bin	Mẫu đúng
1	16	2635
2	256	3475
3	144	
4		

3.2.2. Histogram of Oriented Gradients (HOG)

Đặc trưng HoG dựa vào sự phân bố về cường độ và hướng điểm xám trên ảnh với các tham số **"CellSize"** cho biết kích thước ô để tính giá trị HoG (mặc định [8 8]), nếu giá trị nhỏ thì có thể gây mất mát thông tin khi rút trích đặc trưng; **"BlockSize"** cho biết số lượng cell cấu thành một khối (mặc định [2 2]), nếu kích thước quá lớn có thể làm giảm khả năng ngăn chặn những thay đổi sáng cục bộ trên ảnh, bởi vì nếu số lượng điểm ảnh trong một khối lớn, có thể làm mất giá trung bình do đó, giá trị "**BlockSize**" nhỏ có thể hỗ trợ ngăn chặn những thay đổi sáng cục bộ khi rút trích đặc trưng; bên cạnh đó **"NumBins"** cho biết số lượng bin của đặc trưng.

Bảng kết quả sử dụng fitcknn

STT	CellSize	BlockSize	NumBins	K	Standardize	Mẫu đúng
1	8	2	9	1	0	9709
2	8	2	9	2	1	9720
3	8	2	4	1	0	9631
4	8	2	4	2	1	9632
5	4	2	9	1	0	9767
6	4	2	9	2	1	9727
7	4	2	4	1	0	9717
8	2	2	9	1	0	**9772**
9	2	2	9	2	1	9536
10	2	2	4	1	0	9754

Bảng khảo sát với SVM

STT	CellSize	BlockSize	NumBins	Mẫu đúng
1	8	2	9	9794
2	8	2	4	9721
3	4	2	9	**9883**
4	4	2	4	9861
5	2	2	9	9874
6	2	2	4	9852

3.2.3. Local Binary Pattern (LBP)

Đặc trưng LBP dùng để đo độ tương phản cục bộ trong ảnh với các tham số **"NumNeighbors"** để tính toán giá trị LBP cho mỗi pixel của ảnh nguồn có giá trị lớn hơn mặc định (8); **"Radius"** dùng để lựa chọn lân cận của mỗi pixel ảnh nguồn (mặc định 1, giá trị từ 1 → 5); **"CellSize"** cho biết kích thước ô tính giá

trị LBP (mặc định Size(I)) với kích thước lớn hơn 2x Radius và **"Normalization"** để xác định loại dạng chuẩn cho mỗi biểu đồ xám trên ô tính giá LBP, có hai giá trị "L2" hay "None".

Bảng kết quả sử dụng fitcknn (k = 1 & Standardize = 0)

STT	NumNeighbors	Radius	CellSize	Normalization	Mẫu đúng
1	8	1	8	L2	9156
2	8	1	8	None	9252
3	8	1	4	L2	9515
4	8	1	4	None	**9685**
5	8	2	8	L2	9378
6	8	2	8	None	9323
7	16	1	4	L2	9104

Bảng kết quả sử dụng SVM

STT	NumNeighbors	Radius	CellSize	Normalization	Mẫu đúng
1	8	1	8	L2	9612
2	8	1	8	None	9777
3	8	1	4	L2	9830
4	8	1	4	None	**9845**
5	8	2	8	L2	9728
6	8	2	8	None	9839
7	8	1	6	L2	9821
8	8	1	3	L2	9838
9	8	1	3	None	9839
10	16	1	8	L2	9531

3.2.4. Bag of Words (BOW)

Bag of Words coi những thành phần trên ảnh là một đặc điểm hay đặc trưng để phân loại với các tham số **"VocabularySize"** cho biết số lượng visual words trong đối tượng (mặc định 500); **"PointSelection"** cho phép chọn phương pháp xác định vị trí điểm cho rút trích đặc trưng SURF (mặc định Grid, giá trị khác là Detector); **"GridStep"** cho biết kích thước grid step và chỉ có giá trị khi **"PointSelection"** là Grid và không sử dụng *CustomExtractor*.

Do thời gian hạn hẹp nên vấn đề chạy dữ liệu với tuỳ chỉnh tham số gặp khó khăn nên báo cáo chỉ xin trình bày những kiến thức lý thuyết.

Bảng kết quả

STT	VocabularySize	PointSelection	GridStep	Mẫu đúng
1	500	Grid	8	95%
2	500	Detector	8	26%
3	500	Grid	16	93%
	500	Grid	9	95%
4	500	Grid	32	87%
5	500	Detector	8	27%
6	600	Grid	8	95%

3.2.5. Deep Features

Sử dụng các thông số để rút trích đặc trưng như **"layer"** cho biết tầng để lấy đặc trưng; **"OutputAs"** dùng để định dạng đầu ra của Activations như bằng cột (columns), dòng (rows – mặc định) hoặc kênh (channels); **"MiniBatchSize"** cho biết kích thước của mini-batches sử dụng để dự đoán đối tượng (mặc định: 128).

Bảng kết quả sử dụng Deep Learning

STT	Layer	OutputAs	MiniBatchSize	Mẫu đúng
1	fc7	columns	32	
	fc7	rows	128	
2	fc7	rows	256	
3	conv1	rows	128	

4. Báo cáo mở rộng

Chủ đề: Tìm hiểu YOLO – Real-Time Object Detection

Các hệ thống phát hiện (detection) trước đây sử dụng loại các phân loại và định vị để thực hiện bằng cách áp dụng mô mình cho một ảnh ở nhiều vị trí và kích thước khác nhau. Khu vực có trọng số cao nhất được coi như một phát hiện. Còn trong hệ thống YOLO sử dụng phương pháp hoàn toàn khác, YOLO áp dụng một mạng neural duy nhất cho toàn ảnh. Mạng này chia ảnh thành nhiều vùng và dự đoán mỗi vùng với một xác suất. Mỗi vùng có một trọng số gọi là xác suất dự đoán. Mô hình này có một số ưu điểm so với các hệ thống dựa trên phân lớp khác qua việc dự đoán trên toàn bộ ảnh do đó có thể tận dụng thông tin về bố cục để dự đoán chính xác hơn, bên cạnh đó, mạng neural duy nhất không giống như các mạng như R-CNN với việc cần hàng ngàn ảnh. Điều này giúp tiết kiệm thời gian 1000x lần hơn R-CNN và 100x lần khi so sánh với Fast R-CNN. Phiên bản mới nhất YOLO là version 2.

Thử nghiệm YOLO trên môi trường Ubuntu 16.04, việc đầu tiên chúng ta cần tải thư mục darknet về bằng lệnh

```
git clone https://github.com/pjreddie/darknet
cd darknet
make
```

Tiếp theo, chúng ta cần tải gói trọng số tiền xử lý cho mô hình với kích thước khoảng 258 MB bằng lệnh:

```
wget https://pjreddie.com/media/files/yolo.weights
```

Sau khi thực hiện bước trên, cơ bản chúng ta có thể thử nghiệm kiểm tra bằng cách đưa một ảnh đầu vào và kiểm tra YOLO có thể phát hiện được những đối tượng nào trong ảnh nguồn bằng lệnh:

```
./darknet detect cfg/yolo.cfg yolo.weights data/dog.jpg
```

Mặc định, trong YOLO chỉ hiển thị đối tượng khi xác suất từ 25% trở lên, Ta có thể tuỳ chỉnh thông số bằng **`-thresh <value>`** như sau:

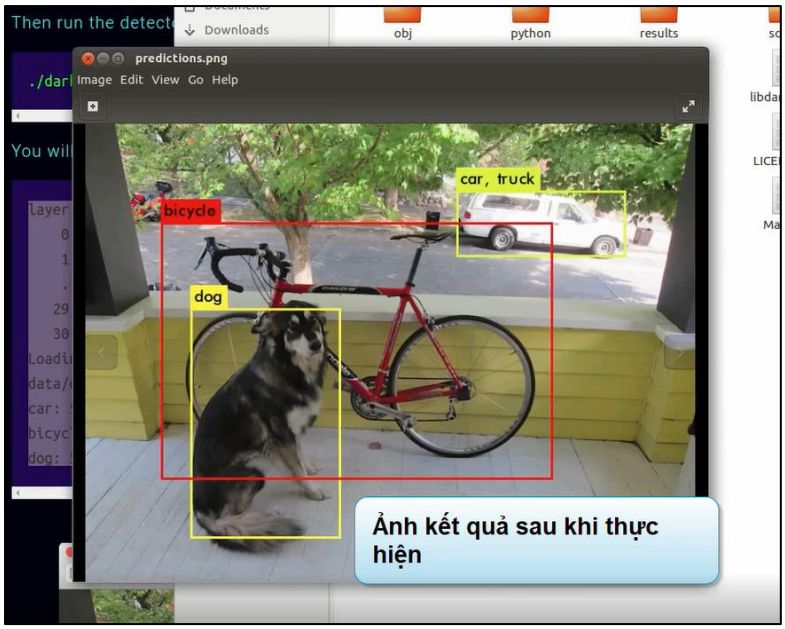
```
./darknet detect cfg/yolo.cfg yolo.weights data/dog.jpg -thresh 0
```

YOLO còn có thể sử dụng được với Webcam bằng cách tích hợp darknet với CUDA và OpenCV qua việc chỉnh thông số trong tập tin Makefile, thiết lập GNU = 1 và OPENCV = 1 bằng lệnh:

```
./darknet detector demo cfg/coco.data cfg/yolo.cfg yolo.weights
```

YOLO sẽ hiển thị số khung mỗi giây (FPS) và dự đoán phân lớp trên ảnh với đường bao bên trên. Tất yếu chúng ta cần webcam kết nối đến máy tính.

```
./darknet detector demo cfg/coco.data cfg/yolo.cfg yolo.weights <video file>
```



Ảnh kết quả sau khi thực hiện chương trình được phát sinh trong thư mục darknet cho viết vùng trên ảnh chứa đối tương nhận dạng (dog, bicycle, car-truck).

5. Kết luận

5.1. Kết quả đạt được

Qua quá trình thực hiện đồ án, em đã hiểu hơn về các khái niệm trong lĩnh vực nhận dạng thị giác và những ứng dụng triển vọng vào nhiều lĩnh vực khác nhau trong đời sống xã hội từ đó thúc đẩy thêm nhiều ý tưởng mới trong việc vận dụng các kiến thức tiếp thu được vào phát triển những phần mềm nhận dạng thị giác trong tương lai. Đồ án thực hiện cơ bản hoàn thành được các yêu cầu sau:

- Cấu trúc tổ chức tập tin chương trình rõ ràng, được phân thành nhiều tập tin với những chức năng và nhiệm vụ riêng từ đó dễ quản lý, dễ dàng chỉnh sửa, mở rộng cho việc phát triển.

- Hiện thực hoá việc tính độ chính xác của thuật toán nhận dạng với nhiều tham số ở mỗi đặc trưng khảo sát (pixcel intensity, HOG, LBP, BoW, Deep Features) và triển khai với các phương pháp học máy như KNN, SVM, Deep Learning (fine-tune).
- Ở mỗi kết quả dữ liệu có đánh giá, tìm ra tham số cho kết quả độ chính xác cao nhất.
- Chương trình thực hiện từng bước rõ ràng có tính tiện ích với người dùng, dễ dàng sử dụng.
- Có tính toán thời gian thực hiện công việc qua việc hiển thị thời gian bắt đầu và kết thúc công việc qua đó có dữ liệu đánh giá chi phí thời gian thực hiện ở mỗi tham số và phương pháp thực hiện.

5.2. Hạn chế

Do thời gian có hạn cùng những yếu tố chủ quan, nên đồ án còn gặp những hạn chế nhất định trong quá trình thực hiện và phát triển:

- Thời gian thực hiện trên một số đặc trưng và phương pháp học máy khá lâu nên chưa thực hiện được toàn bộ đặc trưng. Báo cáo đồ án chỉ dừng ở mức hiện thực chương trình mà chưa hoàn toàn thử nghiệm hết các trường hợp và tham số trong quá trình khảo sát.
- Chưa thực hiện được yêu cầu xây dựng giao diện cho ứng dụng nhận dạng chữ viết tay, tuy đã thực hiện các bước như hướng dẫn.
- Số lượng trường hợp thực hiện còn ít.
- Chưa đủ kiến thức phân tích kết quả một cách tổng quát để đưa ra trường hợp tham số tối ưu.

TÀI LIỆU THAM KHẢO

[1] TS. Lê Đình Duy – TS. Nguyễn Tấn Trần Minh Khang, **Slides bài giảng môn nhận dạng thị giác và ứng dụng**.

[2] https://www.mathworks.com

[3] https://pjreddie.com/darknet/yolo/