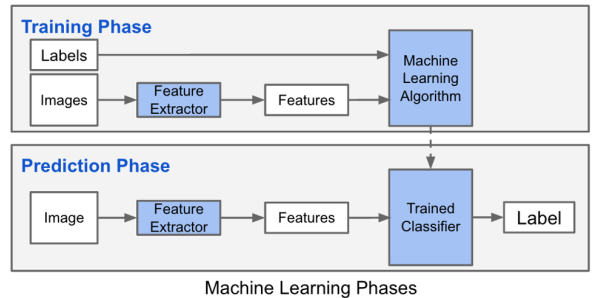
Yêu cầu: viết chương trình để tính độ chính xác của thuật toán nhận dạng với các tham số

**I/ Feature: raw (pixel intensity), HOG, LBP, BoW, deep learning.**

Mô hình hệ thống:



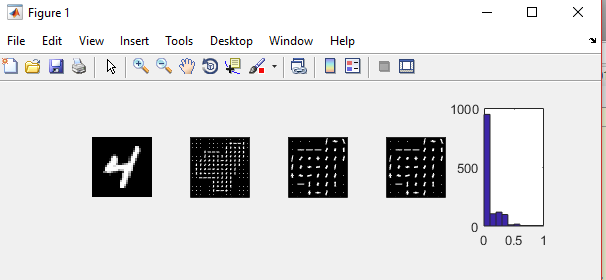
* Nạp dữ liệu train cho bài toán nhận dạng chữ số viết tay
* Lấy đặc trưng **LBP** cho từng ảnh trong dữ liệu train
* Xây dựng một model từ tập dữ liệu đặt trưng **LBP**
* Nạp dữ liệu test của bài toán nhận dạng chữ số viết tay
* Lấy đặc trưng **LBP** cho từng ảnh trong tập dữ liệu test.
* Sử dụng model để predict cho tập dữ liệu train
* Đếm số lượng mẫu đúng và xuất kết quả

1. function Baitap012()
2. imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');
3. lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');
4. imgI1D = imgTrainAll(:,1);
5. imgI2D = reshape(imgI1D,28,28);
6. featureVector = extractLBPFeatures(imgI2D);
7. nSize = length(featureVector);
8. nTrainData = size(imgTrainAll,2);
9. featuresDataTrain = zeros(nSize,nTrainData);
10. for i = 1:nTrainData
11. imgI1D = imgTrainAll(:,i);
12. imgI2D = reshape(imgI1D,28,28);
13. featuresDataTrain(:,i) = extractLBPFeatures(imgI2D);
14. end
15. Mdl = fitcknn(featuresDataTrain', lblTrainAll);
16. imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');
17. lblTestAll = loadMNISTLabels('t10k-labels.idx1-ubyte');
18. imgI1D = imgTestAll(:,1);
19. imgI2D = reshape(imgI1D,28,28);
20. featureVector = extractLBPFeatures(imgI2D);
21. nSize = length(featureVector);
22. nTestData = size(imgTestAll,2);
24. featuresDataTest = zeros(nSize,nTestData);
25. for i = 1:nTestData
26. imgI1D = imgTestAll(:,i);
27. imgI2D = reshape(imgI1D,28,28);
28. featuresDataTest(:,i) = extractLBPFeatures(imgI2D);
29. end
30. lblResult = predict(Mdl,featuresDataTest');
31. nResult = (lblResult == lblTestAll);
32. nCount = sum(nResult);
33. fprintf('\nSo luong mau dung: %d\n',nCount);
34. end

So sánh với thuật giải KNN

1. function BaiTap013()
2. %% Load Data Train
3. strData = 'train-images.idx3-ubyte';
4. strDataLabel = 'train-labels.idx1-ubyte';
5. [imgDataTrain,lblDataTrain] = loadData(strData,strDataLabel);
6. %% Extract Features
7. featuresDataTrain = extractLBPFeatures(imgDataTrain);
8. %% Build model KNN
9. Mdl = fitcknn(featuresDataTrain', lblDataTrain);
10. %% Load Data Test
11. strData = 't10k-images.idx3-ubyte';
12. strDataLabel = 't10k-labels.idx1-ubyte';
13. [imgDataTest,lblActualDataTest] = loadData(strData,strDataLabel);
14. %% Extract Features
15. featuresDataTest = ExtractFeaturesHist(imgDataTest);
16. %% Save Result
17. lblResult = predict(Mdl,featuresDataTest');
18. nResult = (lblResult == lblActualDataTest);
19. nCount = sum(nResult);
20. fprintf('\nSo luong mau dung: %d\n',nCount);
21. end

So luong mau dung: 5439



**Độ chính xác của thuật toán KNN so với Nearest Neighbors**

function Q8\_Advance(k)

imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll' , lblTrainAll, 'NumNeighbors', k);

imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');

lblTestAll = loadMNISTLabels('t10k-labels.idx1-ubyte');

lblResult = predict(Mdl,imgTestAll');

result = (lblResult == lblTestAll);

nCount = sum(result);

accuracy = nCount\*100/size(imgTestAll, 2);

fprintf('\nDo chinh xac cua thuat toan knn voi %d nearest neighbors la %d\n', k, accuracy);

end

Kết quả

>> Q8\_Advance(1)

Do chinh xac cua thuat toan knn voi 1 nearest neighbors la 9.691000e+01

* **BoF**
* Nạp dữ liệu huấn luyện dữ liệu train của bài toán nhận dạng chữ viết tay
* Xây dựng Bag-of-Features(BoF)
* Xây dựng mô hình phân lớp từ tập dữ liệu đặc trưng BoF.
* Phân lớp ảnh mới
* Đánh giá kết quả phân lớp: đếm số lượng mẫu đúng và xuất kết quả.

1. rootFolder = fullfile('DataTrain');
2. categories = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
3. imds = imageDatastore(fullfile(rootFolder, categories), 'LabelSource', 'foldernames');
4. tbl01 = countEachLabel(imds);
5. minSetCount = min(tbl01{:,2});
6. imds = splitEachLabel(imds, minSetCount, 'randomize');
7. tbl02 = countEachLabel(imds);
8. load('bag.mat');
9. img = readimage(imds, 1);
10. featureVector = encode(bag, img);
11. figure
12. bar(featureVector)
13. title('Visual word occurrences')
14. xlabel('Visual word index')
15. ylabel('Frequency of occurrence')
16. load('categoryClassifier.mat');
17. rootFolder = fullfile('DataTest');
18. categories = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
19. imds = imageDatastore(fullfile(rootFolder, categories), 'LabelSource', 'foldernames');
20. tbl01 = countEachLabel(imds);
21. confMatixTest = evaluate(categoryClassifier, imds);
22. mean(diag(confMatixTest));

Kết quả:

>> BagOfFeatures

>> BagOfFeatures

Evaluating image category classifier for 10 categories.

-------------------------------------------------------

\* Category 1: 0

\* Category 2: 1

\* Category 3: 2

\* Category 4: 3

\* Category 5: 4

\* Category 6: 5

\* Category 7: 6

\* Category 8: 7

\* Category 9: 8

\* Category 10: 9

\* Evaluating 10000 images...done.

\* Finished evaluating all the test sets.

\* The confusion matrix for this test set is:

PREDICTED

KNOWN | 0 1 2 3 4 5 6 7 8 9

--------------------------------------------------------------------------------

0 | 0.97 0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.01 0.00

1 | 0.00 0.99 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

2 | 0.01 0.00 0.95 0.01 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 0.00 0.03 0.00 0.95 0.00 0.00 0.01 0.00

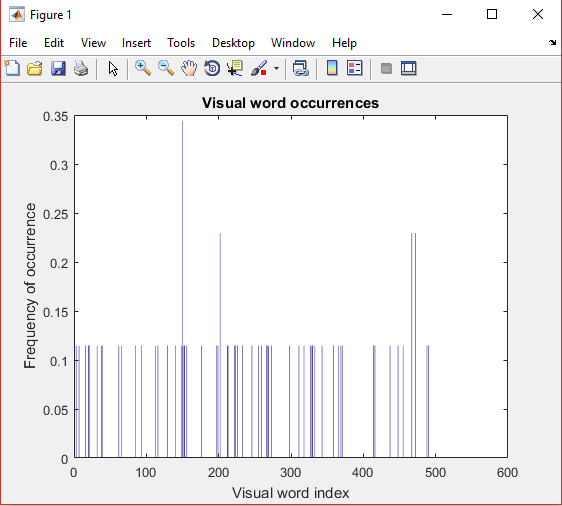
6 | 0.01 0.00 0.01 0.00 0.00 0.01 0.97 0.00 0.00 0.00

7 | 0.00 0.01 0.01 0.01 0.01 0.00 0.00 0.92 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 0.01 0.00 0.01 0.01 0.00 0.00 0.02 0.01 0.92

\* Average Accuracy is 0.95.

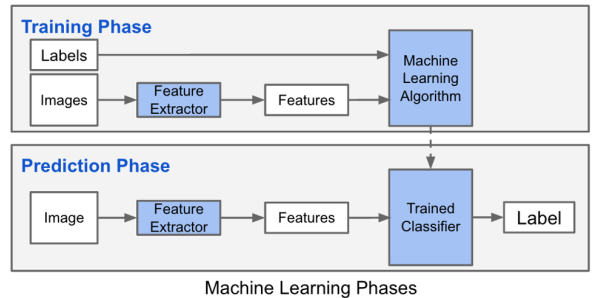


**II. Machine learning methods: KNN, SVM, DL (fine-tune).**

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay.
* Xây dựng một modle từ tập dữ liệu train bằng thuất toán KNN
* Kiểm tra kết quả dự đoán chính xác.

**1/Machine learning methods: KNN**

Mô hình hệ thống:



* Nạp dữ liệu train và dữ liệu test cho bài toán nhận dạng chữ số viết tay

1. function Recognition001\_Digits()
2. fprintf('\n Load du lieu train');
3. imgTrainAll= loadMNISTImages('./train-images.idx3-ubyte');
4. lblTrainAll= loadMNISTLabels('./train-labels.idx1-ubyte');
5. fprintf('\n Load du lieu Test');
6. imgTestAll=loadMNISTImages('./t10k-images.idx3-ubyte');
7. lblTestAll=loadMNISTLabels('./t10k-labels.idx1-ubyte');
8. fprintf('\n Ket thuc .\n');
9. end

* Số lượng ảnh train và ảnh test:

1. function Recognition002\_Digits()
2. fprintf('\n Load du lieu train');
3. imgTrainAll= loadMNISTImages('./train-images.idx3-ubyte');
4. lblTrainAll= loadMNISTLabels('./train-labels.idx1-ubyte');
5. fprintf('\n Load du lieu Test');
6. imgTestAll=loadMNISTImages('./t10k-images.idx3-ubyte');
7. lblTestAll=loadMNISTLabels('./t10k-labels.idx1-ubyte');
8. nTrainImages=size(imgTrainAll,2);
9. nTrainLabels=size(lblTrainAll,1);
10. nTestImages=size(imgTestAll,2);
11. nTestLabels=size(lblTestAll,1);
12. nSizeofImage =size(imgTrainAll,1);
13. fprintf('\n So luong anh train: [%d].',nTrainImages);
14. fprintf('\n So luong nhan anh train: [%d].l',nTrainLabels);
15. fprintf('\n So luong anh test: [%d].', nTestImages);
16. fprintf('\n So luong nhan anh test: [%d].',nTestLabels);
17. fprintf('\n Kich thuoc cua mot anh: [%d].',nSizeofImage);
18. end

Kết quả:

Load du lieu train

Load du lieu Test

So luong anh train: [60000].

So luong nhan anh train: [60000].l

So luong anh test: [10000].

So luong nhan anh test: [10000].

Kich thuoc cua mot anh: [784].

* Xây dựng một modle từ tập dữ liệu train bằng thuất toán KNN

1. function Q5(n)
2. imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');
3. lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');
4. Mdl = fitcknn(imgTrainAll' , lblTrainAll);
5. imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');
6. imgTest = imgTestAll(:, n);
7. lblPredictTest = predict(Mdl, imgTest');
8. fprintf('Ket qua nhan dang anh thu %d trong tap test la: %d\n', n, lblPredictTest);
9. end

Kết quả:

Ket qua nhan dang anh thu 1 trong tap test la: 7

Ket qua nhan dang anh thu 1000 trong tap test la: 9

Ket qua nhan dang anh thu 10000 trong tap test la: 6

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **1** | **1000** | **10000** |
| **Kết quả** |  |  |  |

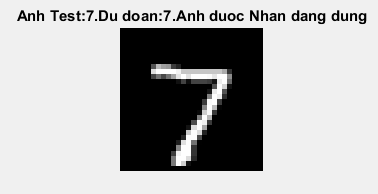
* Ảnh được dự đoán nhận dạng đúng:

1. function Q6(n)
2. imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');
3. lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');
4. Mdl = fitcknn(imgTrainAll' , lblTrainAll);
6. imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');
7. lblTestAll = loadMNISTLabels('t10k-labels.idx1-ubyte');
8. imgTest = imgTestAll(:, n);
9. lblPredictTest = predict(Mdl, imgTest');
10. lblImageTest = lblTestAll(n);
12. figure;
13. img2D = reshape(imgTest, 28, 28);
14. imshow(img2D);
15. strLabelImage = 'Anh Test:';
16. strLabelImage = [strLabelImage, num2str(lblImageTest), '.'];
17. strLabelImage = [strLabelImage, 'Du doan:'];
18. strLabelImage = [strLabelImage, num2str(lblPredictTest), '.'];
19. if(lblPredictTest == lblImageTest)
20. strLabelImage = [strLabelImage, 'Anh duoc Nhan dang dung'];
21. else
22. strLabelImage = [strLabelImage, 'Anh bi nhan dang sai.'];
23. end
24. title(strLabelImage);
25. end

Kết quả

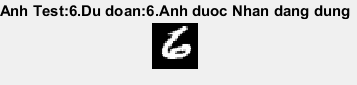
Q6(1)

Ket qua nhan dang anh thu 1 trong tap test la: 7



Q6(10000)

Ket qua nhan dang anh thu 10000 trong tap test la: 6



* Kiểm tra kết quả dự đoán có đúng hay không?

1. function Q7(n)
2. fprintf('\n Load du lieu train');
3. imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');
4. lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');
6. Mdl = fitcknn(imgTrainAll', lblTrainAll);
8. fprintf('\n Load du lieu test');
9. imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');
10. lblTestAll = loadMNISTLabels('t10k-labels.idx1-ubyte');
12. numLabel = size(lblTestAll, 1);
13. result = 0;
15. for i=1:numLabel
16. if(lblTestAll(i) == n)
17. imgTest = imgTestAll(:, i);
18. if(predict(Mdl, imgTest') ~= lblTestAll(i))
19. result = result + 1;
20. end
21. end
22. end
23. fprintf('\nSo luong anh co label %d bi nhan dang sai la: %d\n', n, result);
24. end

>> Q7(1)

Load du lieu train

Load du lieu test

So luong anh co label 1 bi nhan dang sai la: 6

**Lập bảng confusion matrix**

1. function Q7\_Advance()
2. imgTrainAll = loadMNISTImages('train-images.idx3-ubyte');
3. lblTrainAll = loadMNISTLabels('train-labels.idx1-ubyte');
4. Mdl = fitcknn(imgTrainAll', lblTrainAll);
5. imgTestAll = loadMNISTImages('t10k-images.idx3-ubyte');
6. lblTestAll = loadMNISTLabels('t10k-labels.idx1-ubyte');
7. matrix = zeros(10, 10);
8. lblImage = unique(lblTestAll);
9. for val = 1:numel(lblImage)

n = lblImage(val);

fprintf('Anh so %d\n', n);

indices = find(lblTestAll==n);

for idx = 1:numel(indices)

nNumber = indices(idx);

imgTest = imgTestAll(:, nNumber);

lblPredictTest = predict(Mdl, imgTest');

matrix(n+1,lblPredictTest+1) = matrix(n+1,lblPredictTest+1) + 1;

end

1. end
2. csvwrite('Confusion\_matrix.csv', matrix);
3. end

Q7\_Advance

Anh so 0

* Hàm tính độ chính xác của thuật toán KNN với các tham số khác nhau.

1. function Q8\_Advance(k)
2. imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');
3. lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');
5. Mdl = fitcknn(imgTrainAll' , lblTrainAll, 'NumNeighbors', k);
7. imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');
8. lblTestAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');
10. lblResult = predict(Mdl,imgTestAll');
11. result = (lblResult == lblTestAll);
12. nCount = sum(result);
14. accuracy = nCount\*100/size(imgTestAll, 2);
15. fprintf('\nDo chinh xac cua thuat toan knn voi %d nearest neighbors la %d\n', k, accuracy);
16. end

Kết quả:

>> Q8\_Advance(1)

Do chinh xac cua thuat toan knn voi 1 nearest neighbors la 9.691000e+01

**2/Machine learning methods: SVM**

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay
* Xây dựng một model từ tập dữ liệu train bằng thuật giải SVM.
* Nạp dữ liệu test của bài toán nhận dạng chữ viết tay.
* Sử dụng model để dự đoán cho tập dữ liệu test.
* Đếm số lượng mẫu đúng và xuất kết quả.

1. function BaiTap26()
2. strData = 'train-images.idx3-ubyte';
3. strDataLabel = 'train-labels.idx1-ubyte';
4. strDataTest = 't10k-images.idx3-ubyte';
5. strDataLabelTest = 't10k-labels.idx1-ubyte';
6. [imgDataTrain,lblDataTrain] = loadData(strData,strDataLabel);
7. [imgDataTest,lblDataTest] = loadData(strDataTest, strDataLabelTest);
8. Mdl = fitcecoc(imgDataTrain', lblDataTrain);
9. lblResult = predict(Mdl, imgDataTest');
10. nResult = (lblResult == lblDataTest);
11. nCount = sum(nResult);
12. fprintf('\nSo luong mau dung: %d\n', nCount);
13. end

**Số lượng mẫu đúng:**

So luong mau dung: 9438

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay
* Lấy đặc trưng histogram cho từng ảnh trong dữ liệu train
* Xây dựng một modle từ tập dữ liệu đặt trưng histogram bằng thuật giải SVM
* Đếm số lượng mẫu đúng và xuất kết quả:

1. function BaiTap27()
2. strData = 'train-images.idx3-ubyte';
3. strDataLabel = 'train-labels.idx1-ubyte';
4. strDataTest = 't10k-images.idx3-ubyte';
5. strDataLabelTest = 't10k-labels.idx1-ubyte';
6. [imgDataTrain,lblDataTrain] = loadData(strData,strDataLabel);
7. [imgDataTest,lblDataTest] = loadData(strDataTest, strDataLabelTest);
9. featureTrain\_hist = ExtractFeaturesHistogram(imgDataTrain);
10. featureTest\_hist = ExtractFeaturesHistogram(imgDataTest);
12. Mdl = fitcecoc(featureTrain\_hist', lblDataTrain);
13. lblResult = predict(Mdl, featureTest\_hist');
15. nResult = (lblResult == lblDataTest);
16. nCount = sum(nResult);
18. fprintf('\nSo luong mau dung: %d\n', nCount);
19. end

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay
* Lấy đặc trưng HOG cho từng ảnh trong dữ liệu train.
* Xây dựng một model từ tập dữ liệu đặc trưng HOG bằng thuật giải SVM.
* Nạp dữ liệu test của bài toán nhận dạng chữ viết tay.
* Sử dụng model để predict cho tập dữ liệu test.
* Đếm số lượng mẫu đúng và xuất kết quả.

Tập dữ liệu đặc trưng HOG

1. function [featuresDataTrain] = ExtractFeaturesHOG(imgDataTrain, cellSize)
2. imgI1D = imgDataTrain(:,1);
3. imgI2D = reshape(imgI1D,28,28);
4. featureVector = extractHOGFeatures(imgI2D, 'CellSize', cellSize);
5. nSize = length(featureVector);
6. nTrainData = size(imgDataTrain,2);
7. featuresDataTrain=zeros(nSize,nTrainData);
9. for i = 1:nTrainData
10. imgI1D = imgDataTrain(:,i);
11. imgI2D = reshape(imgI1D,28,28);
12. featuresDataTrain(:,i) = extractHOGFeatures(imgI2D, 'CellSize', cellSize);
13. end
14. end

Xây dựng một model từ tập dữ liệu đặc trưng HOG bằng thuật giải SVM

1. function BaiTap27()
2. strData = 'train-images.idx3-ubyte';
3. strDataLabel = 'train-labels.idx1-ubyte';
4. strDataTest = 't10k-images.idx3-ubyte';
5. strDataLabelTest = 't10k-labels.idx1-ubyte';
6. [imgDataTrain,lblDataTrain] = loadData(strData,strDataLabel);
7. [imgDataTest,lblDataTest] = loadData(strDataTest, strDataLabelTest);
8. featureTrain\_hist = ExtractFeaturesHistogram(imgDataTrain);
9. featureTest\_hist = ExtractFeaturesHistogram(imgDataTest);
10. Mdl = fitcecoc(featureTrain\_hist', lblDataTrain);
11. lblResult = predict(Mdl, featureTest\_hist');
12. nResult = (lblResult == lblDataTest);
13. nCount = sum(nResult);
14. fprintf('\nSo luong mau dung: %d\n', nCount);
15. end

Đếm số lượng mẫu đúng và xuất kết quả.

1. function BaiTap28(cellSize)
2. strData = 'train-images.idx3-ubyte';
3. strDataLabel = 'train-labels.idx1-ubyte';
4. strDataTest = 't10k-images.idx3-ubyte';
5. strDataLabelTest = 't10k-labels.idx1-ubyte';
6. [imgDataTrain,lblDataTrain] = loadData(strData,strDataLabel);
7. [imgDataTest,lblDataTest] = loadData(strDataTest, strDataLabelTest);
8. featureTrain\_hog = ExtractFeaturesHOG(imgDataTrain, cellSize);
9. featureTest\_hog = ExtractFeaturesHOG(imgDataTest, cellSize);
10. Mdl = fitcecoc(featureTrain\_hog', lblDataTrain);
11. lblResult = predict(Mdl, featureTest\_hog');
12. nResult = (lblResult == lblDataTest);
13. nCount = sum(nResult);
14. fprintf('\nSo luong mau dung: %d\n', nCount);
15. end

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay.
* Nạp model alexnet (net)
* Sử dụng model(net) để trích chọn đặc trưng của dữ liệu datatrain.
* Xây dựng một model bằng cách sử dụng hàm fitcecoc từ tập dữ liệu đặt trưng này.
* Nạp dữ liệu datatest của bài toán nhận dạng chữ viết tay
* Sử dụng model (net) để trích chọn đặc trưng của dữ liệu datatest
* Sử dụng model để predict cho tập dữ liệu test
* Đếm số lượng mẫu đúng và xuất kết quả

1. function BaiTap29()
2. rootFolderTrain = fullfile('../VRA\_Buoi07/DataTrain');
3. category = {'0','1','2','3','4','5','6','7','8','9'};
4. imdsDataTrain = imageDatastore(fullfile(rootFolderTrain, category), 'LabelSource', 'foldernames');
5. featuresDataTrain = loadFeatures('../../VRA\_Features/featuresTrain-Alex-Number.mat', 'features');
6. lblDataTrain = imdsDataTrain.Labels;
7. classifier = fitcecoc(featuresDataTrain, lblDataTrain, 'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'columns');
8. rootFolderTest = fullfile('../VRA\_Buoi07/DataTest');
9. imdsDataTest = imageDatastore(fullfile(rootFolderTest, category), 'LabelSource', 'foldernames');
10. featuresDataTest = loadFeatures('../../VRA\_Features/featuresTest-Alex-Number.mat', 'features');
11. lblDataTest = imdsDataTest.Labels;
12. lblResult = predict(classifier, featuresDataTest');
13. nResult = (lblDataTest == lblResult);
14. nCount = sum(nResult);
15. fprintf('\nSo luong mau dung: %d\n', nCount);
16. end

* Nạp dữ liệu train của bài toán nhận dạng chữ viết tay.
* Nạp model alexnet (net)
* Sử dụng model(net) để trích chọn đặc trưng của dữ liệu datatrain.
* Xây dựng một model bằng cách sử dụng hàm **fitcknn** từ tập dữ liệu đặt trưng này.
* Nạp dữ liệu datatest của bài toán nhận dạng chữ viết tay
* Sử dụng model (net) để trích chọn đặc trưng của dữ liệu datatest
* Sử dụng model để predict cho tập dữ liệu datatest
* Đếm số lượng mẫu đúng và xuất kết quả

1. function BaiTap30()
2. rootFolderTrain = fullfile('../VRA\_Buoi07/DataTrain');
3. category = {'0','1','2','3','4','5','6','7','8','9'};
4. imdsDataTrain = imageDatastore(fullfile(rootFolderTrain, category), 'LabelSource', 'foldernames');
5. featuresDataTrain = loadFeatures('../../VRA\_Features/featuresTrain-Alex-Number.mat', 'features');
6. lblDataTrain = imdsDataTrain.Labels;
7. classifier = fitcknn(featuresDataTrain', lblDataTrain);
8. rootFolderTest = fullfile('../VRA\_Buoi07/DataTest');
9. imdsDataTest = imageDatastore(fullfile(rootFolderTest, category), 'LabelSource', 'foldernames');
10. featuresDataTest = loadFeatures('../../VRA\_Features/featuresTest-Alex-Number.mat', 'features');
11. lblDataTest = imdsDataTest.Labels;
12. lblResult = predict(classifier, featuresDataTest');
13. nResult = (lblDataTest == lblResult);
14. nCount = sum(nResult);
15. fprintf('\nSo luong mau dung: %d\n', nCount);
16. end

**Tương tự như trên:** Xây dựng một model bằng cách sử dụng hàm **fitcnb** từ tập dữ liệu đặt trưng này.

1. function BaiTap31()
2. rootFolderTrain = fullfile('../VRA\_Buoi07/DataTrain');
3. category = {'0','1','2','3','4','5','6','7','8','9'};
4. imdsDataTrain = imageDatastore(fullfile(rootFolderTrain, category), 'LabelSource', 'foldernames');
5. featuresDataTrain = loadFeatures('../../VRA\_Features/featuresTrain-Alex-Number.mat', 'features');
6. lblDataTrain = imdsDataTrain.Labels;
7. classifier = fitcnb(featuresDataTrain', lblDataTrain);
8. rootFolderTest = fullfile('../VRA\_Buoi07/DataTest');
9. imdsDataTest = imageDatastore(fullfile(rootFolderTest, category), 'LabelSource', 'foldernames');
10. featuresDataTest = loadFeatures('../../VRA\_Features/featuresTest-Alex-Number.mat', 'features');
11. lblDataTest = imdsDataTest.Labels;
12. lblResult = predict(classifier, featuresDataTest');
13. nResult = (lblDataTest == lblResult);
14. nCount = sum(nResult);
15. fprintf('\nSo luong mau dung: %d\n', nCount);
16. end

**Tương tự như trên:** Xây dựng một model bằng cách sử dụng hàm **fitcdiscr** từ tập dữ liệu đặt trưng này.

1. function BaiTap32()
2. rootFolderTrain = fullfile('../VRA\_Buoi07/DataTrain');
3. category = {'0','1','2','3','4','5','6','7','8','9'};
4. imdsDataTrain = imageDatastore(fullfile(rootFolderTrain, category), 'LabelSource', 'foldernames');
5. featuresDataTrain = loadFeatures('../../VRA\_Features/featuresTrain-Alex-Number.mat', 'features');
6. lblDataTrain = imdsDataTrain.Labels;
7. classifier = fitcdiscr(featuresDataTrain', lblDataTrain);
8. rootFolderTest = fullfile('../VRA\_Buoi07/DataTest');
9. imdsDataTest = imageDatastore(fullfile(rootFolderTest, category), 'LabelSource', 'foldernames');
10. featuresDataTest = loadFeatures('../../VRA\_Features/featuresTest-Alex-Number.mat', 'features');
11. lblDataTest = imdsDataTest.Labels;
12. lblResult = predict(classifier, featuresDataTest');
13. nResult = (lblDataTest == lblResult);
14. nCount = sum(nResult);
15. fprintf('\nSo luong mau dung: %d\n', nCount);
16. end