Synthetic Image Character Recognition

JAZIB ALI

BCSF14M508

UMAR BIN KHALID

BCSF14M519

26 January 2018

Table of Contents

[Introduction 1](#_Toc504667912)

[Literature Review 1](#_Toc504667913)

[Implementation 1](#_Toc504667914)

[Results 1](#_Toc504667915)

[Conclusion 1](#_Toc504667916)

[References 1](#_Toc504667917)

[Scripts 1](#_Toc504667918)

Table of Figures

*Figure 1 A 9-bin Histogram \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4*

*Figure 2 Histogram of Gradients \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5*

*Figure 3 Result 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5*

*Figure 4 Result 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6*

# Introduction

The main objective of this project was to recognize characters from synthetic character images of Chars74K dataset. For this purpose, we first calculated the **histogram of oriented gradients** for our images. We broke our dataset into two parts. The first part, containing 700 images, was used for training purpose. In the second part, containing rest of the images, was used for testing purpose. We used the hog of our images from training dataset to train the K nearest neighbor classifier.

# Literature Review

For the understanding and implementation of HOG, we took help from the article on “Histogram of Oriented Gradients” by Satya Mallick from 2016 [1].

For the understanding of K nearest neighbor classifier algorithm we took help from Wikipedia article on “K nearest neighbor classifier” [2].

# Implementation

For implementation,

* We first divided our dataset into two variables named traindata and testdata.
* We calculated HOG for our traindata.
* We created a cell vector to classify data calculated above.
* We trained a nearest neighbor classifier table using FITCKNN [3].
* We calculated HOG for our testdata.
* We used the KNN classification to identify our testdata.

**Steps to Calculate Histogram of Oriented Gradients for synthetic Images.**

1. Computing Gradients

Computation of Gradients of an image in both x and y directions. This method requires filtering the intensity data of the image with the following filter kernels:

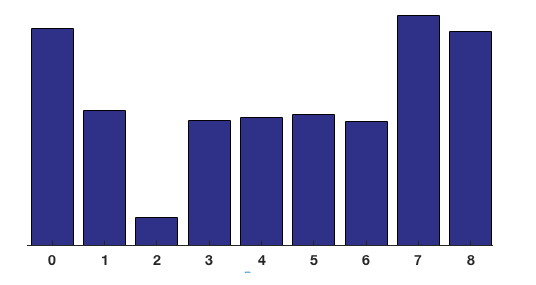
[-1,0,1] and [-1,0,1]T

When the gradients are computed it automatically removed a lot of non-essential information but highlighted outlines.

1. Binning of orientations

In this step orientation that is to say the computed phase angles are put in 9 bins. The bins are from 0 to 160 with a gap of 20 between each bin.

9-bin histogram which can be stored as vector of 9 numbers.

   
 Figure 1 A 9-bin Histogram.

1. Calculate Histogram of Oriented Gradient

In this step, the image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells.

The following figure illustrates the process. We are looking at magnitude and direction of the gradient of an 8×8 patch. A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. Let’s first focus on the pixel encircled in blue. It has an angle (direction) of 80 degrees and magnitude of 2. So it adds 2 to the 5th bin. The gradient at the pixel encircled using red has an angle of 10 degrees and magnitude of 4. Since 10 degrees is half way between 0 and 20, the vote by the pixel splits evenly into the two bins.

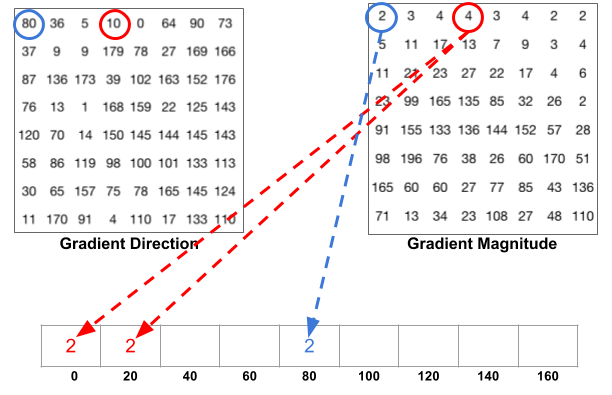


Figure 2 Histogram of Gradients.

We get a feature vector by concatenating histogram of gradients of every cell in our image.

# Results



Figure 3 Result 1

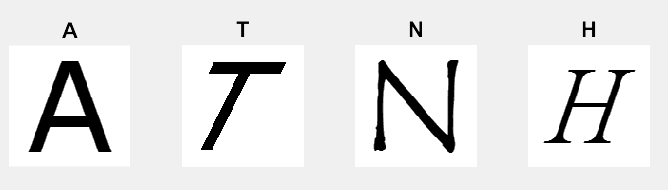


Figure 4 Result 2

# Conclusion

In a nut shell, we implemented a small level of machine learning concept in which we recognize characters from synthetic character images using the K nearest neighbor classifier algorithm. The KNN classification table we trained gives results close to 88% accuracy.

# References

[1] “Histogram of Oriented Gradients” by Satya Mallick from 2016 [Online]. Available: <https://www.learnopencv.com/histogram-of-oriented-gradients/>

[2] “K nearest neighbor classifier” on Wikipedia [Online]. Available:   
<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>   
  
[3] <https://www.mathworks.com/help/stats/fitcknn.html>

# Scripts

**Main.m**

figure  
subplot(1,4,1);  
I = testdata{21};  
imshow(I);  
hog = HogH(I);  
class = predict(Mdl,hog);  
title(class);  
  
subplot(1,4,2);  
I = testdata{6000};  
imshow(I);  
hog = HogH(I);  
class = predict(Mdl,hog);  
title(class);  
  
subplot(1,4,3);  
I = testdata{4322};  
imshow(I);  
hog = HogH(I);  
class = predict(Mdl,hog);  
title(class);  
  
subplot(1,4,4);  
I = testdata{2312};  
imshow(I);  
hog = HogH(I);  
class = predict(Mdl,hog);  
title(class);

**HogFeat.m**

function [feature] = HogH(im)  
  
if size(im,3)==3  
 im=rgb2gray(im);  
end  
im=double(im);  
  
rows=size(im,1);  
cols=size(im,2);  
  
  
  
[magnitude,angle] = imgradient(im,'central');  
angle=imadd(angle,90);  
  
  
feature=[];  
  
for i = 1:8:rows-7  
 for j=1:8:cols-7  
  
  
 magA = magnitude(i : i+7 , j : j+7);  
 angleA = angle(i : i+7 , j : j+7);  
 histr =zeros(1,9);  
  
 for p=1:8  
 for q=1:8  
  
 alpha= angleA(p,q);  
  
 if alpha>0 && alpha<=20  
 histr(1)=histr(1)+ magA(p,q)\*(20-alpha)/20;  
 histr(2)=histr(2)+ magA(p,q)\*(alpha-0)/20;  
 elseif alpha>20 && alpha<=40  
 histr(2)=histr(2)+ magA(p,q)\*(40-alpha)/20;  
 histr(3)=histr(3)+ magA(p,q)\*(alpha-20)/20;  
 elseif alpha>40 && alpha<=60  
 histr(3)=histr(3)+ magA(p,q)\*(60-alpha)/20;  
 histr(4)=histr(4)+ magA(p,q)\*(alpha-40)/20;  
 elseif alpha>60 && alpha<=80  
 histr(4)=histr(4)+ magA(p,q)\*(80-alpha)/20;  
 histr(5)=histr(5)+ magA(p,q)\*(alpha-60)/20;  
 elseif alpha>80 && alpha<=100  
 histr(5)=histr(5)+ magA(p,q)\*(100-alpha)/20;  
 histr(6)=histr(6)+ magA(p,q)\*(alpha-80)/20;  
 elseif alpha>100 && alpha<=120  
 histr(6)=histr(6)+ magA(p,q)\*(120-alpha)/20;  
 histr(7)=histr(7)+ magA(p,q)\*(alpha-100)/20;  
 elseif alpha>120 && alpha<=140  
 histr(7)=histr(7)+ magA(p,q)\*(140-alpha)/20;  
 histr(8)=histr(8)+ magA(p,q)\*(alpha-120)/20;  
 elseif alpha>140 && alpha<=160  
 histr(8)=histr(8)+ magA(p,q)\*(160-alpha)/20;  
 histr(9)=histr(9)+ magA(p,q)\*(alpha-140)/20;  
 elseif alpha>=160 && alpha<=180  
 histr(1)=histr(1)+ magA(p,q)\*(alpha-160)/20;  
 histr(9)=histr(9)+ magA(p,q)\*(180-alpha)/20;  
 elseif alpha>170 && alpha<=180  
 histr(9)=histr(9)+ magA(p,q)\*(190-alpha)/20;  
 histr(1)=histr(1)+ magA(p,q)\*(alpha-170)/20;  
 end  
  
  
 end  
 end  
 feature=[feature histr];  
  
 end  
end  
end

**Classification.m**

A = cellfun(@HogH,traindata,'UniformOutput',false);  
  
  
c = length(traindata);  
double\_matrix = cell2mat(A');  
alphab = ('A':'Z');  
char\_array = char(ones(c,1) \* 'A');  
for itr=1:26  
 char\_array(((itr-1) \* (c/26)) + 1: ((itr-1) \* (c/26)) + (c/26)) = alphab(itr);  
end  
  
  
t1 = num2cell(char\_array);  
  
Mdl = fitcknn(double\_matrix,t1.');