CSC420 —— Final project Report

Mohammad Javaad Akhtar, 1002551838

April 2018

1 Introduction

Fashion industry is one of the largest industries in the world. One's choice of clothing is closely related to one's socio-identity mainly indicating one's status, wealth, fashion sense and culture. An algorithm related to vision to recognize pieces of clothing have a wide variety of potential impacts that can range from a better social understanding to improved person identification [**1**]. One of the main challenges are identifying the pieces of clothes a person can be wearing, from hats to shirts to shoes. Parsing such information from a 2D-image is challenging and can be very costly to the fashion industry. However, producing an algorithm and training a neural network to understand the human and the piece of clothes they are wearing, as interesting as it sounds, brings a ton of difficulties and challenges with it. Computer Vision is one of my favourite topics in the field of computer science, and when combined with a real life scenario, it makes it more interesting. The whole idea of parsing clothes not only looks challenging but also a type of challenge that will give me more better insight of the world of super pixels and neural networks. I used various methods and techniques i learned from this course to complete this project. I tried various techniques ranging from, initially tried to distinguish the person and background using the combination of Canny Edge detector and Harrison ford corner detector to using Convolutional Neural Network and super pixel. From using as basic approach like calculating pixels and gradients and keeping a certain thresh hold to color the region to training a Neural network with combination of super pixels. It's been a long, yet educational ride.

2 Methods and Results

We have learned a lot of methods regarding computer vision and how computers interact with visuals and it's surrounding. For this project, I used 2 method. The first method is not precise in determining the person and the background. However, the second method is more precise and more accurate. I will discuss both of the methods.

Method 1:Canny Edge Detector
The first method is something i call a Noobies approach. The implementation is relatively simple and concise however comes with a downsize of not being accurate. The accuracy rate is very low. This approach uses the Canny edge detector combined with Harrison Ford corners detector and some manipulation of the output. The method starts by finding the canny edge of the picture by using the edge function:

```
edge(img, 'canny', 0.3, 0.3)
```

Once we have the edges, now we have to find the harrison ford corner detector. We use code from A2 to find the Harrison Ford corner detector. Once we have both matrices and data, now we start computing the data. So for that, i use matrix manipulation.Soon i realized this method was flawed, so i stopped working on it.
Result of Method 1:
The result from Method 1 were very disappointing. However it was not surprising that the results were way off than the real image. This is the code for the canny where Q1a is from A2. Q1a is modified to work with the threshold for the required output:

```
function canny()
addpath(genpath('./images'));
addpath(genpath('./labels'));
images = dir('images/*.jpg');


for all= 1:5
    img =  rgb2gray(double(imread(images(all).name))/255) ;
    % img =  double(imread('002.png'))/255 ;
    f1 = edge(img, 'canny', 0.3, 0.3);
    [x,y] = Q1a(imread(images(all).name));
    figure;
    imshow(f1);
    [l,w] = size(f1);
    count=0;
    for i = 1:w
        for j = 1:l-1
            count = count + 1;
            if f1(x(i),y(j)) == 0
```

```
                f1(x(j),y(i)) =1;
            end
            if f1(j,i) == 1
                f1(j+3,i) = 1;
            end
        end
    end
     imwrite(f1,images(all).name, 'jpg');
end

end
```

canny.m

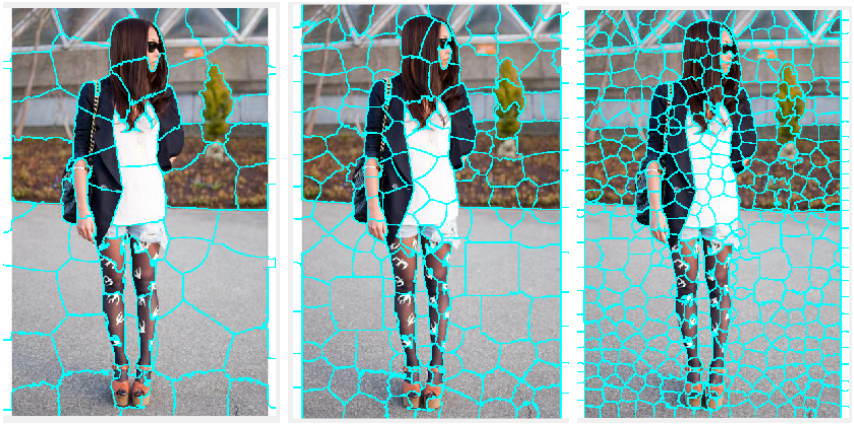
And this is the output of the canny m. From right to left: real image, truth image, canny output image:

We can see that the canny function does recognize human to a very small extent and is somewhat able to distinguish the person with the background. However, we can observe it's not that precise when it comes to predicting which one is human and which one is background. The reason it fails so terribly is because of the implemented algorithm. The algorithm only take into account the first half of the canny edge output, and then predict the other half based on the first half which leads to wrong result and leads to looking like a white box by the foot side of the human. That's the reason the foot is non-existent in the canny edge detector. We can conclude that resemblance from the output of the canny.m function and the truth image is pretty uncanny.

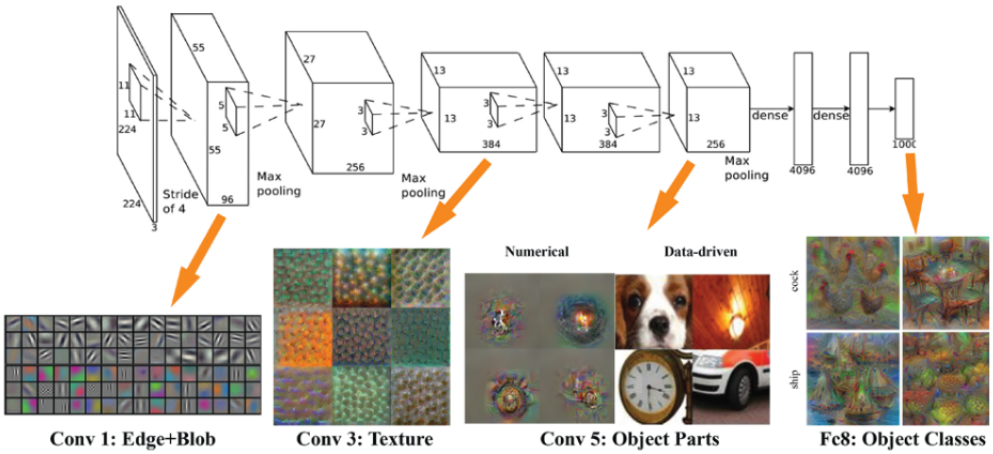
Method 2: Super Pixels and CNN

Implementing this method was the best part of this project. Not only using

this method increased accuracy, but also helped me learn in depth about super pixels, super pixel manipulation and Convolutional neural networks. This approach is at somepoint still incomplete as i was not able to finish part (b) of this project due to time shortage. Let's first talk about super pixel. Super pixel is the core part of this project. I start by finding the super pixel of the picture. However first i needed to make sure how may super pixels i want.



Too many super pixels unnecessary increases the computation and makes the image hard to deal with (and more clutter). Too less super pixels does not divides all the region properly. Once we get the right super pixels, we can further do the computation. Now we have to go through each of the super pixels to compute the center of the super pixel. Once we have the center, we use a certain threshold to keep the super pixel which are fit for features extractions and disregard the one's that are not. Once we have the features, we label each super pixel 1 or 0 depending on whether that region is of a human or of a back ground. And we store the label (that is 1 or 0) of each image in an array and features in other array for further use and computations.



Now we start with CNN (Convolutional Neural Network). This was the most challenging part of the assignment. In order to achieve the goal to integrate Convolutional Neural Network with the super pixels was a hard task. First i need to build some sort of Convolutional Neural Network. To do this i need to understand in depth how the Convolutional Neural Network works. Luckily, project 6 of CS4476 Georgia Institute of Technology teaches you too how to build your own Convolutional Neural Network. The starter Code provided by the site has only 25% accuracy. It's our job to understand the CNN well enough to make changes in order to increase the accuracy from 25% to 85%. Fortunately i was able to achieve that accuracy by completing the task provided by that site. However i wasn't able to fully integrate it with the super pixel implementation i did. For my CNN, i have multiple layers. The first layer is the "weights" which are the filters being learned. These filters are initialized as random numbers from a Gaussian distribution.There is 9 by 9 spatial resolution, span 1 filter depth and there are 10 filters. The network also learns a bias or constant offset to associate with the output of the each filter. The second layer the max pooling layer. This layer will decrease the the spatial resolution by a factor of 7 according to the stride parameter. However, same as previous layer the filter depth remains same. The next layer is the non-linearity. Any features map from the max pooling will be set to 0 if they are negative. Finally the last layer, where the real stuff happens. The last layer is the convolution. The filter learned at this payer operates on the rectified, subsampled, maxpooled filter responses from the first layer [2]. At the top of all these, we have a others layer. These will be the "loss" layer.

Learning about Convolutional Neural Network gave me insight to many important concepts of deep learning, one of the major ones is "jittering" when we do not have sufficient data. The Convolutional Neural Network i made is trained with jitter, and using mirroring technique and left-right flip, thus this increased the accuracy of the Convolutional Neural Network by 10%. Another important concept i had to implement in my Neural network was adding the dropout reg-

ularization. This prevents a unit in one layer from relying too strongly on a single unit in the previous layer. After applying dropout regularization and increasing the number of training epochs, i was able to achieve 55% test accuracy. Various other technique were used to make my Convolutional Neural Network more faster, deeper and more accurate like adding a batch normalization layer after the convolutional layer.

Results of method 1: Result of this method were very astonishing. The accuracy can be seen pretty high. First we need to specify whether we want to specify how many segmentation do we want and which part do we want to compute.

```
function extractFeatures(param)
images = dir('./images/*.jpg');
persons = dir('./labels/*person.png');
truths = load('truths.mat');
cloths = dir('labels/*clothes.png');
labels = {};
features = [];

%If the user doesnt tells which part it wants to solve, a or b
if nargin < 1
     error("Please choose the parameter to be either 'a' or 'b'");
end
%param decides whether we want to compute part a of the project or
    part b.
for i = 1:2
    if param == 'a'
        backgroundAndPerson(i,persons,images,truths.truths,labels,
            features);
    elseif param == 'b'
        clothsDetector(i,cloths,images,truths.truths,labels,
            features);
    else
        error("Please choose the parameter to be either 'a' or 'b'"
            );
    end
end
end
```

extractFeatures.m

The user needs to specify which part it want to be labelled. If user passes:

```
    extractFeatures('a');
```

This will call the function "backgroundAndPerson" which does the major computation.

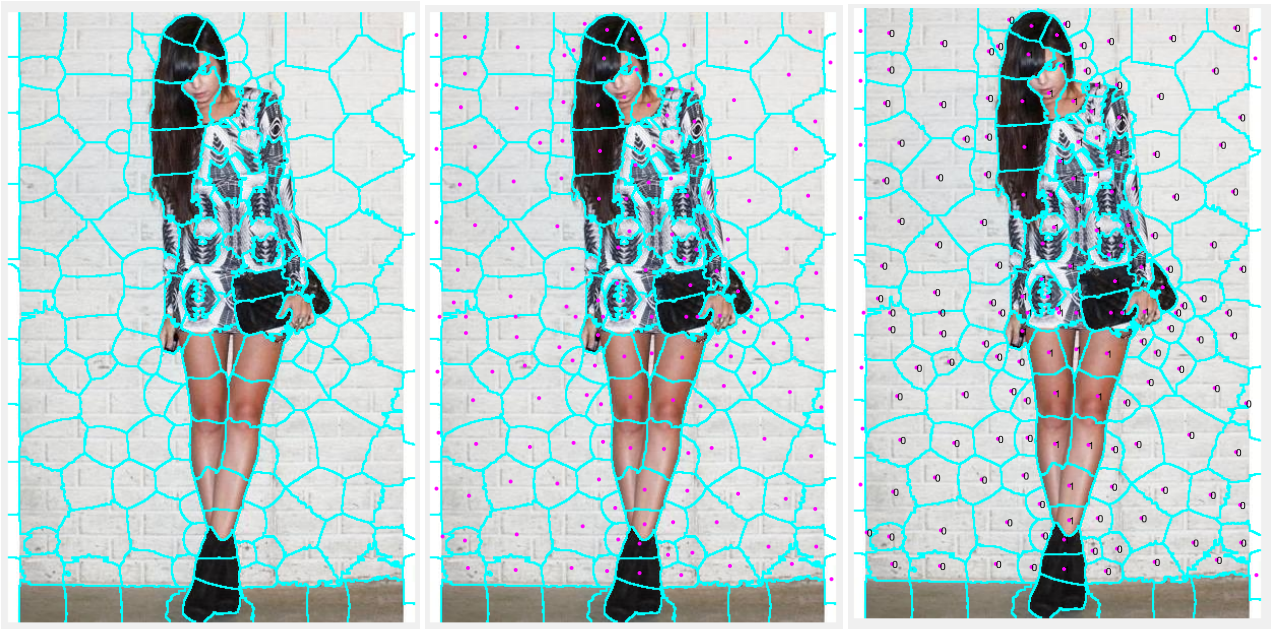
```
function  backgroundAndPerson(i,persons,images,truth,label_image,
    features)
    % This function computes the background and person and store
        the labels
```

```
3       % and featuers for further use for CNN. i is the number of the
            current
4       % image, persons is the directory of labels/*person.jpg, truths
             is the
5       % truth extracted from fashionista_v0.2.1, label_image is where
            the
6       % labels of the final computed image is going to be stored and
            features
7       % is where the features of the function is going to be stored.
8       current = imread(images(i).name);
9       truths = truth(i).pose.point;
10      labelimage = imread(persons(i).name);
11
12      %Computing the super pixels and Convert label matrix to cell
            array of
13      %linear indices which is obtained by the super pixels
14      [SP, ~] = superpixels(current,170);
15      [rSp,~] = size(SP);
16      SP_index = label2idx(SP);
17      counter = length(SP_index);
18      BW = boundarymask(SP);
19      %Showing the image
20      imshow(imoverlay(current,BW,'cyan'),'InitialMagnification',67);
21      hold on;
22
23      for j = 1:counter
24
25          %We are finding the center of the each segment of the super
                pixels
26          %for labeling purposes.
27          diffX = max(mod(SP_index{j}, rSp)) - min(mod(SP_index{j},
                rSp));
28          diffY = (max(SP_index{j}/rSp) + 1) - (min(SP_index{j}/rSp)
                + 1);
29          centerX = ceil(min(mod(SP_index{j}, rSp)) + diffX/2);
30          centerY = ceil((min(SP_index{j}/rSp) + 1) + diffY/2);
31
32          % Plotting the points on the image.
33          plot(centerY,centerX,'m.','MarkerSize',10);
34
35           %Filtering out the super pixels which are below the
                threshold as
36           %described in the report.
37          if centerX-11 <= 0 || centerX+12 > 600 || centerY-12 <= 0
                || centerY+11 > 400
38              continue;
39          end
40
41          % Extracting the features from the image, so it can
                helpfull for
42          % fruther use when using the CNN
43          featuresExtract = extractLBPFeatures(rgb2gray(current(
                centerX-11: centerX+12, centerY-12: centerY+11, 1:3)));
44  %         [featuresExtract, ~] = extractHOGFeatures(current, [
        centerX centerY], 'CellSize', [8 8]);
45          if isempty(featuresExtract) == 0
46
```

```
47              %labelling each super pixel segmentation with 0 if it
                    is a
48              %background and 1 if it is a forground from the
                    knowledge of
49              %the truth image from the label
50              text(centerY,centerX,int2str(labelimage(centerX,centerY
                    )));
51              features = [features; featuresExtract];
52              label_image = [label_image; labelimage(centerX,centerY)
                    ];
53          end
54      end
55      % Person and background distinguisher
56      labels = [];
57      points
            =[[1,2];[2,3];[3,4];[4,5];[5,6];[7,8];[8,9];[9,10];[10,11];[11,12];[13,14];
             [3,9]; [3,13]; [3,10]; [4,9]; [4,13];
            [4,10];[10,14];[9,14]];
58      tempX= [];
59      tempY = [];
60
61      for x = 1:length(points)
62          [x_p,y_p]=findLines([truths(points(x,1),1),truths(points(x
                ,1),2)],[truths(points(x,2),1),truths(points(x,2),2)
                ],100);
63          tempX = [tempX ,round(x_p)];
64          tempY = [tempY , round(y_p)];
65      end
66
67      for x = 1:length(tempX)
68          if ~isnan(tempY(x)) && ~isnan(tempX(x))
69             labels = [labels SP(tempY(x), tempX(x))];
70          end
71      end
72      labels = unique(labels);
73      outputImage = zeros(600, 400);
74
75      display(labels)
76      % labels all the super pixels that crosses the lines
77      for y = 1:600
78          for x = 1 : 400
79              if any( labels == SP(y,x))
80                  outputImage(y,x)= 1;
81              end
82          end
83      end
84      imagesc(outputImage);
85      imwrite(outputImage,images(i).name, 'jpg');
86      display(labels)
87
88  end
```

backgroundAndPerson.m

Here is a step by step results that is produced by that function: From right to left: Super pixel of the image, center of the super pixel, center with label 1 if person and 0 if label is a background:



The labelling helps the Convolutional Neural Network as well as me in order to write the image. The label where the super pixel segmentation is 1, we color that area 1 and same goes for background. Here is the output from the backgroundAndPerson function.From right to left: real image, truth image, backgroundAndPerson output image:

Here is an output from a different image:

However now if the user passes:

```
extractFeatures('b');
```

This is will call the function clothsDetector which labels the different type of

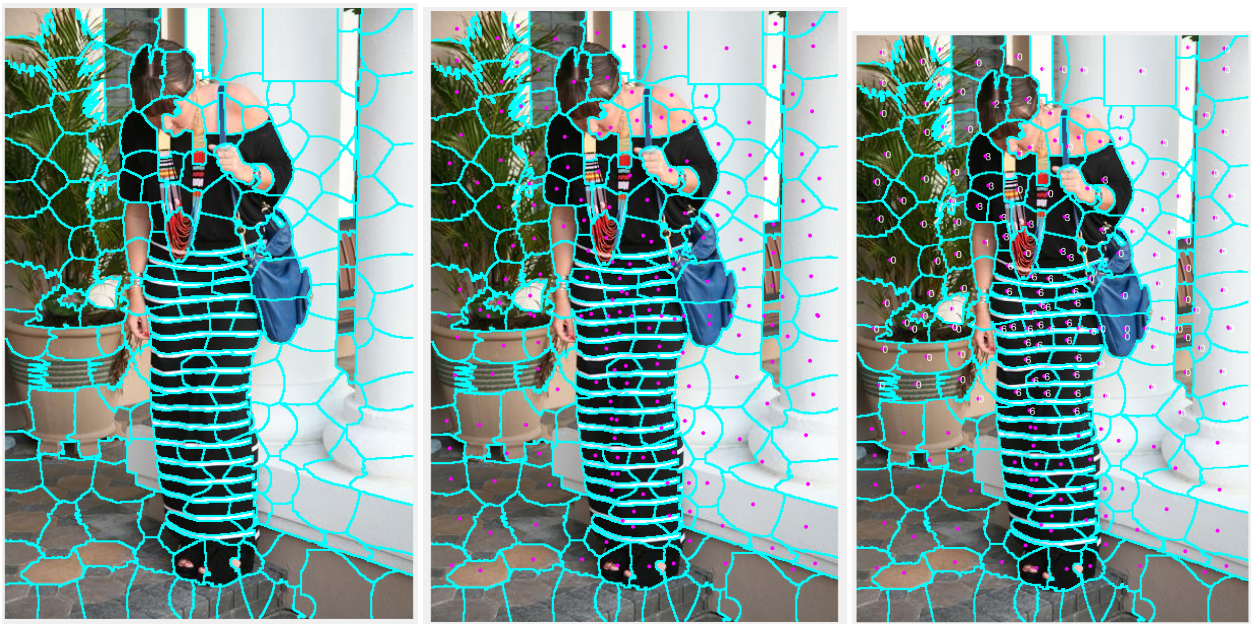
clothing.

```
function clothsDetector(i,cloths,images,truth,labels,features)
    % This function computes the different kind of clothings and
        store the labels
    % and featuers for further use for CNN. i is the number of the
        current
    % image, persons is the directory of labels/*person.jpg, truths
         is the
    % truth extracted from fashionista_v0.2.1, label_image is where
        the
    % labels of the final computed image is going to be stored and
        features
    % is where the features of the function is going to be stored.

    current = imread(images(i).name);
    truths = truth(i).pose.point;
    labelimage = imread(cloths(i).name);
    A = current;
    A = imresize(A,0.5);
    Alab = rgb2lab(A);
    %Computing the super pixels and Convert label matrix to cell
        array of
    %linear indices which is obtained by the super pixels
    [SP, N] = superpixels(current,200,'isInputLab',true);
    [rSp,~] = size(SP);
    SP_index = label2idx(SP);
    counter = length(SP_index);
    BW = boundarymask(SP);
%     imshow(imoverlay(current,BW,'cyan'),'InitialMagnification
    ',67);
%     hold on;

    for j = 1:counter

        %We are finding the center of the each segment of the super
            pixels
        %for labeling purposes.
        diffX = max(mod(SP_index{j}, rSp)) - min(mod(SP_index{j},
            rSp));
        diffY = (max(SP_index{j}/rSp) + 1) - (min(SP_index{j}/rSp)
            + 1);
        centerX = ceil(min(mod(SP_index{j}, rSp)) + diffX/2);
        centerY = ceil((min(SP_index{j}/rSp) + 1) + diffY/2);

        % Plotting the points on the image.
%         plot(centerY,centerX,'m.','MarkerSize',10);

        %Filtering out the super pixels which are below the
            threshold as
        %described in the report.
        if centerX-11 <= 0 || centerX+12 > 600 || centerY-12 <= 0
            || centerY+11 > 400
            continue;
        end

```

```
43          % Extracting the HOG features from the image, so it can
                helpfull for
44          % fruther use when using the CNN
45          [featuresExtract, ~] = extractHOGFeatures(current, [centerX
                centerY], 'CellSize', [8 8]);
46          if isempty(featuresExtract) == 0
47              %labelling each super pixel segmentation with 0 if it
                    is a
48              %background and other labels if it is a differnt piece
                    of clothings from the knowledge of
49              %the truth image from the label
50  %             text(centerY,centerX,int2str(labelimage(centerX,
    centerY)),'Color','white','FontSize',14);
51              distances = zeros(1,14);
52              for m = 1:14
53                  distances(m) = sqrt( (centerX - truths(m,1))^2  + (
                        centerY - truths(m,2))^2 );
54              end
55              features = [features; featuresExtract];
56              labels = [labels; labelimage(centerX,centerY)];
57          end
58      end
59
60      %coloring the region of the super pixels so we can distinguigh
            certain
61      %elements
62      meanColor = zeros(N,3);
63      [m,n] = size(SP);
64      for  i = 1:N
65              meanColor(i,1) = mean(Alab(SP_index{i}));
66              meanColor(i,2) = mean(Alab(SP_index{i}+m*n));
67              meanColor(i,3) = mean(Alab(SP_index{i}+2*m*n));
68      end
69      numColors = 4;
70      [idx,cmap] = kmeans(meanColor,numColors,'replicates',2);
71      cmap = lab2rgb(cmap);
72      Lout = zeros(size(A,1),size(A,2));
73      for i = 1:N
74          Lout(SP_index{i}) = idx(i);
75      end
76  %     imshow(label2rgb(Lout))
77      imshow(Lout,cmap)
78
79  end
```

clothsDetector.m

Here is a step by step results that is produced by that function: From right to left: Super pixel of the image, center of the super pixel, center with label values: 0,1,2,3,4,5,6 depending on its category in that is background, skin, hair, t-shirt, shoes, pants, dress and 0 if it is background:



The label where the super pixel segmentation is 0: background, 1: skin, 2: hair, 3: t-shirt, 4: shoes, 5: pants, 6: dress.
Following is the result to an attempt to color certain regions of the super pixels:

Clearly, we can see it's a failed attempt. The color in the image does not corresponds to color values we want. Part a of the project is completed and running, however part b needs a lot of improvements.

3 Main Challenges:

This project came with a lot of challenges. From as simple challenge as setting up the host machine for this project to writing a whole convolutional Neural network to trying to integrate it with the super pixel segmentation has been a hassle. However, with every challenge comes a benefit, the benefit of learning and improving one's skill. This what happened with this project, with every step i gain more and more insight about the world of deep learning and super pixels. One of the major hassle was running the code on the host machine. Many time a code provided on the CSC420 website sometimes does not runs due to lack of proper libraries in the host machine. Another major challenge that i tackled was understanding how the super pixel works and how to manipulate them. It took me an approximate of 2 weeks to fully understand the super pixel segmentation and it's manipulation. I had to read a lot of articles and published papers regarding super pixels segmentation in order to grasp the idea of super pixel good enough in order to start this assignment. However, the hardest challenge i faced was understanding Deep learning and Convolutional Neural network. The remaining time spent was on understanding and creating Neural Network. Unfortunately it did not give me enough time to fully integrate it with the super pixel segmentation extractor i made. As a person who never took courses like Machine Learning or any Computer vision course, i had to take it upon myself to understand the Convolutional Neural network in order to finish this project. Learning and understanding Convolutional Neural network in complete depth was hard enough, creating it was another challenge. Once I have taken my time and after a significant amount of reasearch, once i felt comfortable with the idea of Convolutional Neural network, it was time to make one. Luckily, project 6 of CS4476 Georgia Institute of Technology teaches you too how to build your own Convolutional Neural Network [2]. The whole idea behind that project 6 of CS4476 is to teach student in depth how Convolutional Neural network works and how one can make it more efficient and accurate. I have to complete project 6 of CS4476 in order to not only fully understand Convolutional Neural network but also to have that i made by my own hand. Once after completing project 6 of CS4476, i had to tweak the Convolutional Neural network so it can work with my super pixel segmentation and feature extractions. That project 6 starts with giving you a simple Convolutional Neural network with 25% accuracy and makes you go through the hassle of understanding the Convolutional Neural network so you can increase it's efficiency and accuracy. I was able to increase the accuracy of the Convolutional Neural network to more than 70%. Due to lack of time and resources (as i am working alone for this pair project), i was not able to finish the task of integrating the Convolutional Neural network i made with the super pixel segmentation and feature extraction i made. However,

even tho i was not able to finish the task, i believe this project of CSC420 completed it's objective by giving me an in depth understanding of super pixel and Convolutional Neural networks.

4 Conclusion and future work

This project can only be described as a roller coaster ride. It came with thrilling challenges and exciting new ideas. This project came a long way since it was started and with each step, it made me understand in depth more and more about the world of computer vision. This project has a lot of potential in term of real life use and implementation. I had made a super pixel segmentation extractor which extracts data from the truths and it's corresponding image. Then used the knowledge of Deep learning, made a Comvolutional Neural Network so it can learn about the images. Due to time shortage, i was not able to fully integrate the two main part of the project, convolutional Neural Network and Super pixel segmentation. In the future, i believe both can be integrated in order to identify the cloths and person and backgrounds.

5 References

SuperPixel and Segmentation:

1. Yamaguchi, Kota, et al. "Parsing clothing in fashion photographs." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.

1.1. Mori, Greg. "Guiding model search using segmentation." Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 2. IEEE, 2005.

1.2. Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods." IEEE transactions on pattern analysis and machine intelligence 34.11 (2012): 2274-2282.

CNN:

2. https://www.cc.gatech.edu/ hays/compvision/proj6/

2.1. Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." IEEE transactions on neural networks 8.1 (1997): 98-113.

2.2. http://www.robots.ox.ac.uk/ vgg/practicals/cnn/index.html