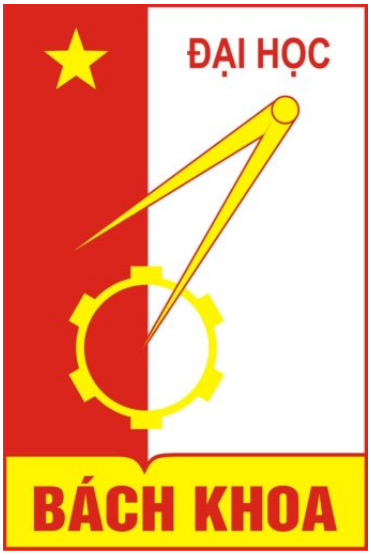
Hanoi University of Science and Technology
School of Information and Communication Technology



Graduation Research 1

Basic Image Principles, Operations and Neural Network in Computer Vision

Student: Nguyen Dinh Hai Nam - 20143040 - ICT k59

Instructor: Ph.D Nguyen Thi Oanh

June 17, 2018

Contents

1 Basic concepts:

1.1 Images

1.1.1 Definition:

- An image is a 2D signal (x,y)
- Mathematical point of view:
  - An image is a matrix of numbers representing a signal.
  - Several tools exist to manipulate that signal.
- Humanity point of view:
  - An image contains many semantic information.
  - It needs to be interpreted beyond the value of numbers

1.1.2 Classification:

There are 3 main types of images

- Gray level: representing images in black and white channels where distribution of histogram in range [0...255]
- Binary images: each pixel of image has the value is 1 or 0
- RGB images: representing images as a combination of 3 color channels Red-Green-Blue, distribution of histogram in each color channel in is in range [0...255]

1.1.3 Properties:

- Image sampling is limited by the capacity of the sensor, which is the number of pixels.
- Image quantization: is limited by the quantity of tones defined for the interval.
- Image representation:
  - Matrix of size MxN
  - Each value in the matrix is an integer values in range [0...255]
- Image resolution:
  - Spatial resolution: The smallest visible element.
  - Gray level resolution: The smallest visible color change.
- Image luminance: is defined the mean of all gray levels in the image
- Image contrast: is defined by standard deviation of the gray levels or the variation between the min and the max gray level

1.2 Edge

1.2.1 Definition:

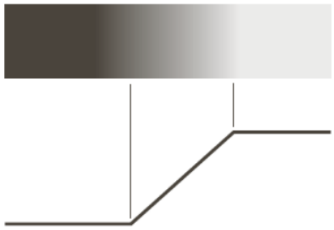
An edge is a frontier(boder) separating two objects in an image.

- A discontinuity in the image.
- Sometimes also referred as "*contours*"

1.2.2 Classifications:

- Step edge:

- Ramp edge:



- Roof edge:

Otherwise, edges can be affected by noise:

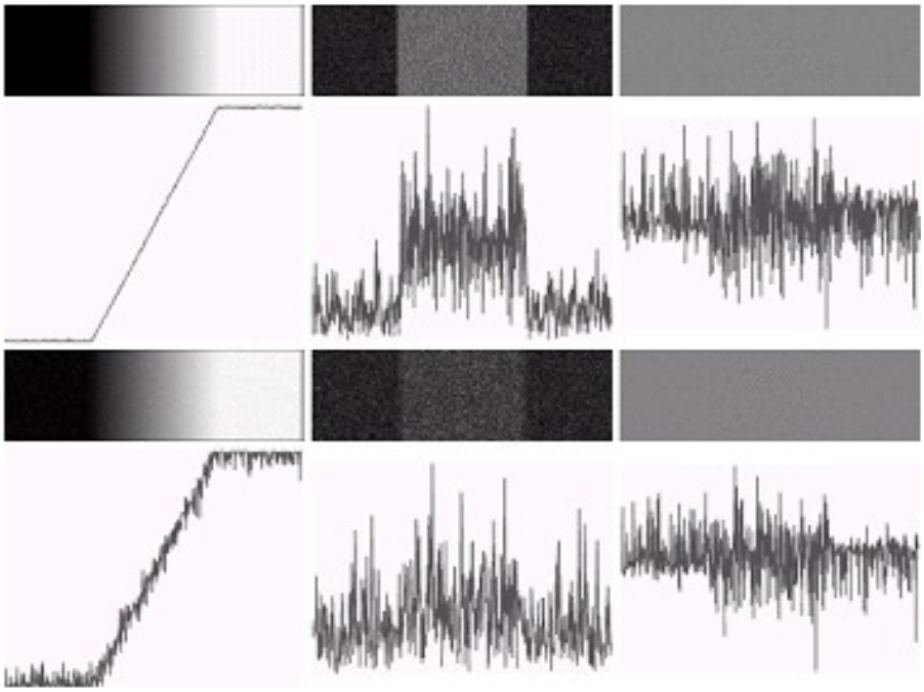


Figure 1: Edges with noise

1.2.3 Image derivative and edges:

1.2.3.1 Image derivative:

The first derivative (*gradient*) of the image is the base operator to measure edges in the image:

$$| \nabla f | \equiv \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2$$

- Image histogram:

- First derivative $f'(x)$:

- $| f'(x) |$ & threshold:

- $| f'(x) | <$ threshold:

1.2.4 Filter/Kernel for detecting edges by convolution operation:

- Roberts filter:

1	0
0	-1

0	1
-1	0

- Sobel filter:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

- Prewitt:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

and etc...

1.2.5 Image gradient:

Derivative in X coordinate denoted as Gx, derivative in X coordinate denoted as Gy

1.2.5.1 Magnitude:

Gradient intensity for each pixel:

$$|\,G\,| \;=\; \sqrt{Gx^2 + Gy^2} \;\approx\; |\,Gx\,|\;|\,Gy\,|$$

1.2.5.2 Direction:

Main gradient funtion for each pixel

$$\theta = arctan(Gy/Gx)$$

1.2.6 Second image derivative

Another approach to find edges in images is to use the *second image derivative, for this, we use the Laplacian as an operator*

$$\nabla^2 I = \frac{\partial I}{\partial x^2} + \frac{\partial I}{\partial y^2}$$

1.2.6.1 Laplacian using convolution:

- Many discrete approximations exist for the Laplacian:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -8 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- One sole convolution matrix
- Rotation symmetric

1.2.7 Canny filter:

Optimal filter for edge detection: filtering the image in several steps.

1.2.7.1. Steps:

1. Apply a Gaussian low-pass filter to remove noise on the image.
2. Apply Sobel filter to the image and compute the gradient intensity and gradient direction in the image.
3. Non-maxima suppression: if the gradient magnitude of a pixel (x,y) is inferior to the one of its 2 neighbors along the gradient direction, then set this magnitude for (x,y) to zero.
4. Edge thressholding (hysteresis)
   (a) Use two thresholds: a threshold high (Sh) and a threshold low (Sb)
   (b) For each pixel in the gradient magnitude:
      i. if $magnitude(x, y) < Sb$, then set this pixel to zero
      ii. If $magnitude(x, y) > Sh$, the set this pixel to edge pixel.
      iii. If $Sb < magnitude(x, y) \leq Sh$, then the pixel is set to edge pixel if it is connected to another edge pixel

1.3 Segmentation and Texture:

1.3.1 Segmentation:

1.3.1.1 Definition:

Segmentation aims to split an image into several parts which should correspond to object in the image

1.3.1.2 Goals:

Extract, separate elements in the image for applying a specific processing afterward or interpreting the image content.

- Discontinuities, edges: sudden changes, borders (frontier) between regions...
- Homogeneous zones, regions: which has same colors, texture, intensity.

1.3.1.3 Methods:

1. Thresholding
   (a) Histogram thresholding
   (b) Global thresholding
   (c) Multi-thresholding
   (d) Global automatic thresholding

(e) Adaptive thresholding

2. K-means algorithm
3. Split-and-merge
4. Watershed

1.3.2 Textures:

1.3.2.1 Definition:

Texture can be defined as a region with variation of intensity or as a spatial organization of pixels.

1.3.2.2 Methods:

1. First order statistics: statistics on histogram
2. Co-occurence matrices: searching patterns
3. Frequential analysis: Gabor filter

The most difficult is to find a good presentation for each texture.

1.3.3 Color:

1.3.3.1 Introduction:

- Each pixel contains information from a spectral bandwidth.
- We obtain color images, for example, by taking 3 bands from visible spectrum.
- Some devices exist to acquire signal from more bands (X-ray, infrared, radio, ...)

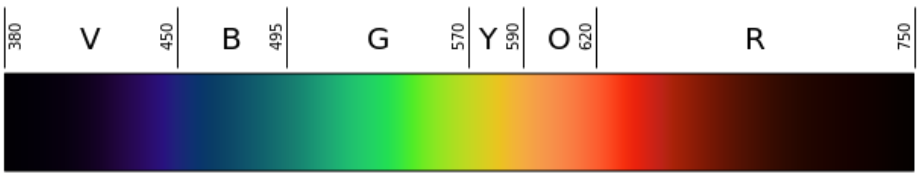


Figure 2: Color spectrum

1.3.3.2 Primary colors:

- RGB: Color representation using primary colors Red-Green-Blue:Additive scheme for displaying on a screen
- CMY: Color representation using primary colors Cyan-Magenta-Yellow
  - Subtractive scheme for printing on paper
  - We subtract from white instead of adding to black like in RGB
  - CMY = 1 - RGB

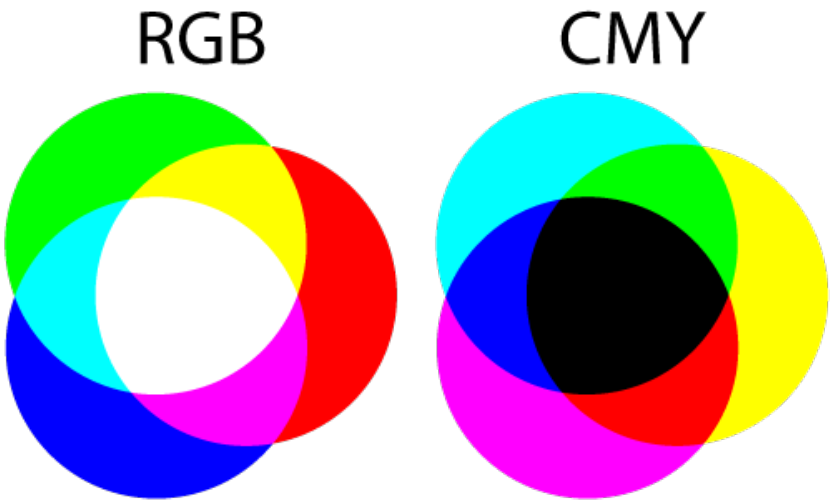


Figure 3: RGB - CMY color space

1.3.3.3 Color spaces

- There are many different spaces to represent colors.
- RGB is the most common color representation in computer.
- There are three types of color spaces:
  - Purely physic approach
  - Purely visual approach
  - Physic approach but corrected by psychometry

1.3.3.4 HSV representation:

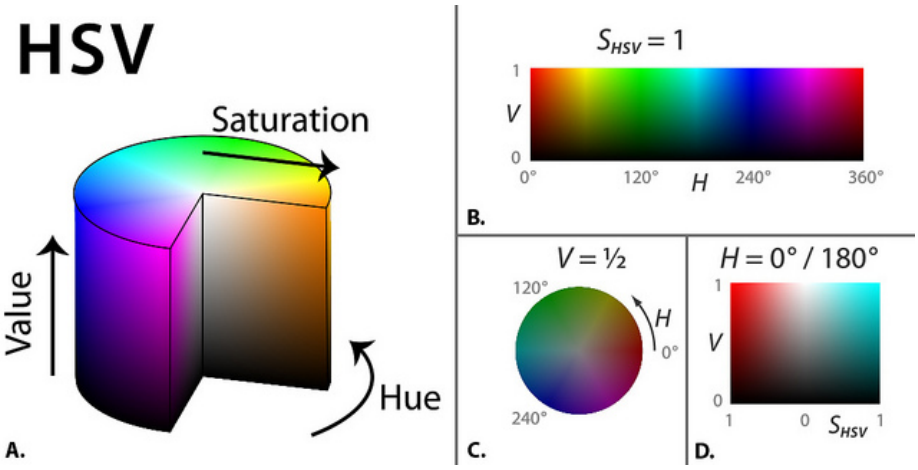


Figure 4: HSV color space

- H-Hue is coded as an angle between 0 and 360°
- S-Saturation is coded as a radius between 0 and 1, S = 0 ∼ gray, S = 1 ∼ pure color
- V-Value = MAX(Red, Green, Blue)

2 Operations and Transformations:

2.1 Operations:

2.1.1 Contrast enhancement:

- Linear Transformation: Enhance the dynamic range by linearly stretching the original gray levels to the range of target.
- Piecewise Linear Transformation: Linear stretching with k segments.
- Non-linear Transformation:
  - Non-linear functions with a fixed form.
  - Fewer parameters to adjust.
  - Satisfying: $0 = f_{min} \leq g \leq f_{max} = L - 1$
  - Examples:
    - Logarithmic transformation: stretch dark region, suppress bright region
      $$g = b.log(af + 1)$$
    - Exponential transformation: expand bright region
      $$g = b(e^{af} - 1)$$
    - Power law: $g = af^k$
      $k = 2$: square law, similar to exponential, $k = \frac{1}{3}$: cubic root, similar to logarithmic

- Histogram Equalization: Transform an image with arbitrary histogram to one with a flat histogram
  - Suppose f has PDF $p_F(f), 0 \leq f \leq 1$
  - Transform function (continuous): $g(f) = \int_0^f p_F(t)dt$

2.1.2 Image operators

- Logical operators: AND, OR, XOR, NOT
- Addition, Subtraction, Multiplication: add, subtract, multiply two pixels in two images with the corresponding position with upperbound 255 and lowerbound 0 into a new values of new image.

2.1.3 Interpolation

- Nearest Neighbor Interpolation:
  Interpolation uses the values of the nearby translated pixel for the output pixel values.
- Bilinear Interpolation:
  The block uses the weighted average of two translated pixel values for each output pixel value.
- Bicubic Interpolation:
  The block uses the weighted average of four translated pixel values for each output pixel value.

2.2 Transformations

2.2.1 Fourier Transforms:

- Fourier transforms applied in Image Processing is Discrete Fourier Transforms because spatial domain from image is discrete
- The output of the transformation represents the image in the frequency domain.

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+uy)}dudv$$

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{j2\pi(ux+uy)}dxdy$$

  where u and v are spatial frequencies
- Fourier transformation produces a complex number valued output image. We use the real part to represent geometric structure of the spatial domain image. However, we need the real and the imaginary part to re-transform the image.
  - $F(u,v)$ is complex in general

$$F(u,v) = F_R(u,v) + jF_i(u,v)$$

  - $|\ F(u,v)\ |$ is the magnitude spectrum
  - $\arctan(F_i(u,v)/F_R(u,v))$ is the phase angle spectrum
- High frequencies: The points far from from FT center.
- Low frequencies: The points near from FT center
- Applications: Image analysis, image filtering, image reconstruction and image compression

2.2.2 Hough transforms:

- Hough transformation can be used to detect lines circles of used to detect other parametric curves.
- It can give robust detection under noise and partial occlusion.
- Transform line in $x-y$ space to point in $m-c$ space:

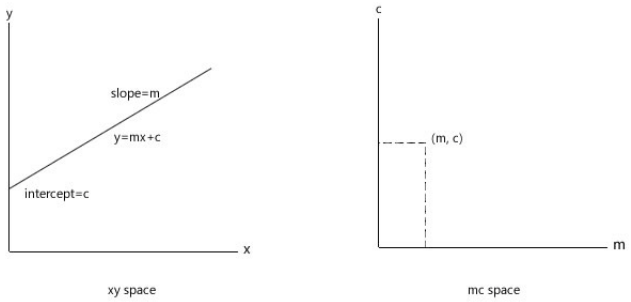


Figure 5: Line in xy space to point in mc space

- Transform point in $x-y$ space to line in $m-c$ space:

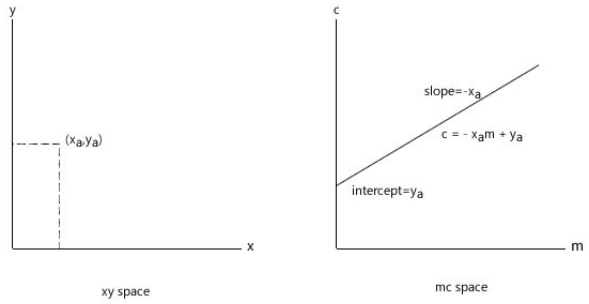


Figure 6: Point in xy space to line in mc space

- For Hough transform, taken an edge detected image, for every point that is non-black, draw lines in the $m-c$ space. Obviously, some lines will intersect. These intersections mark are the parameters of the line.

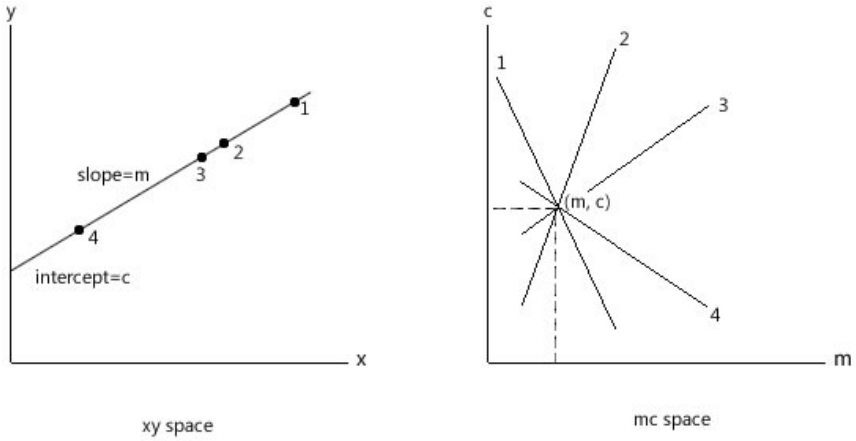


Figure 7: Lines and points in xy space to points and lines in mc space

3 SIFT & SURF:

3.1 SIFT

3.1.1 Description:

The scale invariant feature transform, SIFT [17], extracts a set of descriptors from an image. The extracted descriptors are invariant to image translation, rotation and scaling (zoom-out). SIFT descriptors have also proved to be robust to a wide family of image transformations, such as slight changes of viewpoint, noise, blur, contrast changes, scene deformation, while remaining discriminative enough for matching purposes

3.1.2 Algorithm:

1. Compute the Gaussian scale-space defined as function $L(x, y, \sigma)$ with the variable-scale Gaussian $G(x, y, \sigma)$ and the input image $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

   where $*$ is convolution operation in x and y and $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

2. compute the Difference of Gaussians(DoG)

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

3. Find candicate keypoints (3D discrete extrema of DoG) by comparing a pixel to its 26 neighbors in 3x3 regions at the current and adjacent scales.

4. Refine candicate keypoints location with sub-pixel precision.

5. Filter unstable keypoints due to noise.

6. Filter unstable keypoints lying on edge using Taylor expansion of scale-space function.

7. Assign a reference orientation to each keypoint based on scale-space gradient and orientation.

8. Build the keypoints d;escriptor $(x, y, \sigma, \theta, \mathbf{f})$

3.2 SURF

3.2.1 Introduction:

In computer vision, speeded up robust features (SURF) is a patented local feature detector and descriptor. It can be used for tasks such as object recognition, image registration, classification or 3D reconstruction. It is partly inspired by the scale-invariant feature transform (SIFT) descriptor

3.2.2 Algorithm:

To detect interest points, SURF uses an integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image. Its feature descriptor is based on the sum of the Haar wavelet response around the point of interest. These can also be computed with the aid of the integral image.

SURF descriptors have been used to locate and recognize objects, people or faces, to reconstruct 3D scenes, to track objects and to extract points of interest.

3.3 Comparison:

1. Change of Viewpoint:

Viewpoint Variation	SIFT	SURF
Number of keypoint of img1:	2058	2951
Number of keypoint of img2:	1692	2431
Time elapsed (s):	0.12	0.24

2. Salt-pepper noise with rate of 0.5:

Noisy	SIFT	SURF
Number of keypoint of img1:	1422	2399
Number of keypoint of img2:	33495	59142
Time elapsed (s):	0.77	01.09

3. Angle Variation of 45 °:

Angle Variation - 45*	SIFT	SURF
Number of keypoint of img1:	1422	2399
Number of keypoint of img2:	7107	8601
Time elapsed (s):	01.06	0.54

4. Angle Variation of 90 °:

Angle Variation - 90*	SIFT	SURF
Number of keypoint of img1:	1422	2399
Number of keypoint of img2:	6560	8476
Time elapsed (s):	0.56	0.43

5. Angle Variation of 135 °:

Angle Variation - 135*	SIFT	SURF
Number of keypoint of img1:	1422	2399
Number of keypoint of img2:	7135	8602
Time elapsed (s):	01.04	0.55

6. Angle Variation of 180 °:

Angle Variation - 180*	SIFT	SURF
Number of keypoint of img1:	1422	2399
Number of keypoint of img2:	6840	8502
Time elapsed (s):	0.55	0.43

4 Neural Network in Computer Vision:

4.1 Neural Network:

4.1.1 Definitions:

An Artificial Neural Network is an information processing paradigm that is inspired by the way biological nervous system, such as brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones

4.1.2 Architecture:

Neural Network as neurons in graphs. Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons.

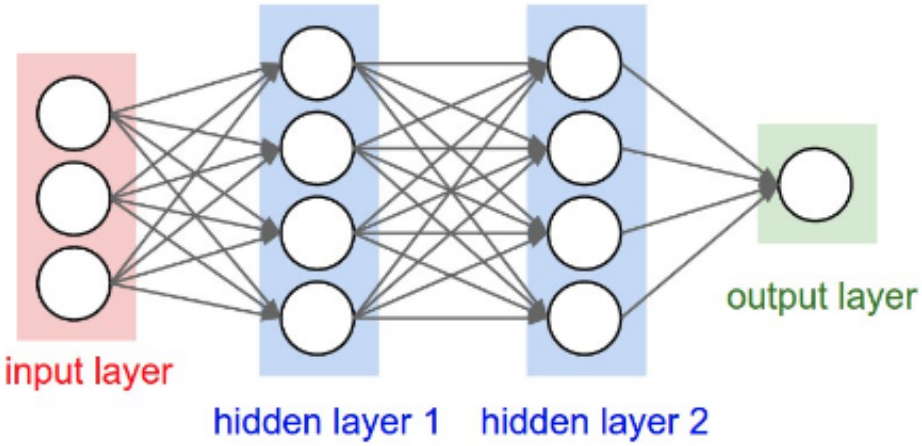


Figure 8: Neural Network with 2 hidden layers

1. Input layer: this is the first layer of a neural network. It is used to provide the input data or features to the network

2. Output layer. Unlike all layers in a Neural Network, the output layer neurons most commonly do not have an activation function (or you can think of them as having a linear identity activation function). This is because the last output layer is usually taken to represent the class scores (e.g. in classification), which are arbitrary real-valued numbers, or some kind of real-valued target (e.g. in regression).

3. Hidden layer: A feedforward network applies a series of functions to the input. By having multiple hidden layers, we can compute complex functions by cascading simpler functions. The number of hidden layers is termed as the depth of the neural network. In general, deeper networks can learn more complex functions.

4.1.3 Activation Functions

1. Sigmoid functions: It maps the input (x axis) to values between 0 and 1.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

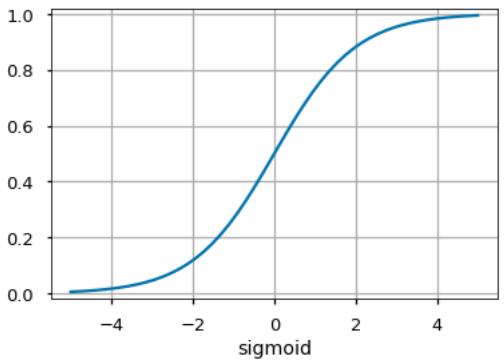


Figure 9: sigmoid function

In practice, the sigmoid non-linearity has recently fallen out of favor and it is rarely ever used. It has two major drawbacks:

- Sigmoid saturates and kills gradients: when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. Therefore, if the local gradient is very small, it will effectively "kill" the gradient and almost no signal will flow through the neuron to its weights and recursively to its data.
- Sigmoid outputs are not zero-centered:This has implications on the dynamics during gradient descent, because if the data coming into a neuron is always positive (e.g. $x > 0$ elementwise in $f = wTx + b)$), then the gradient on the weights w will during backpropagation become either all be positive, or all negative (depending on the gradient of the whole expression f). This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights.

2. Tanh functions:It is similar to the sigmoid function butmaps the input to values between -1 and 1.

$$\tanh(x) = 2\sigma(2x) - 1.$$

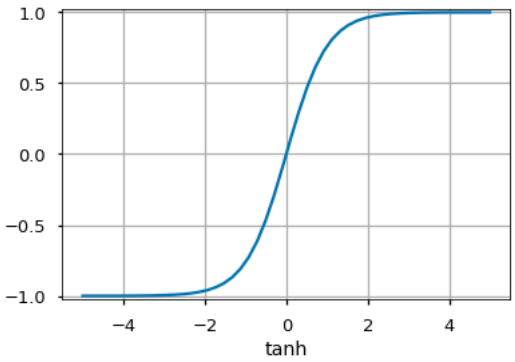


Figure 10: tanh function

Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity.

3. Rectified Linear Unit(ReLU): It allows only positive values to pass through it. The negative values are mapped to zero.

$$f(x) = \max(0, x)$$

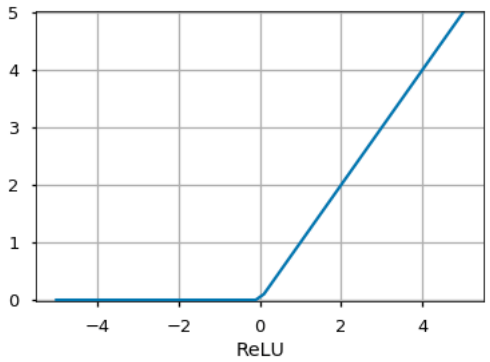


Figure 11: ReLU function

(a) Advantages:

- It was found to greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid/tanh functions.
- Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.

(b) Disadvantages:

- ReLU units can be fragile during training and can "die". For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again. If this happens, then the gradient flowing through the unit will forever be zero from that point on

4.1.4 Optimization:

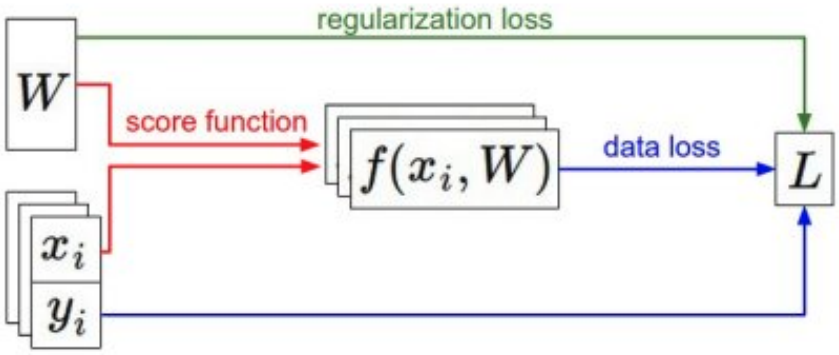


Figure 12: Optimization

The dataset of pairs of (x, y) is given and fixed. The weights start out as random numbers and can change. During the forward pass the score function computes class scores, stored in vector f. The loss function contains two components: The data loss computes the compatibility between the scores f and the labels y. The regularization loss is only a function of the weights. During Gradient Descent, we compute the gradient on the weights (and optionally on data if we wish) and use them to perform a parameter update during Gradient Descent.

4.1.5 Backpropagation:

A way of computing gradients of expressions through recursive application of chain rule.Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

4.2 Convolution Neural Network:

4.2.1 Definition:

Convolutional Neural Networks are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply

4.2.2 Why Convolution Neural Network in Computer Vision:

Input images for processing in Computer Vision usually consist of 3 channel colors with size MxN pixels, for example in CIFAR10 data set each image is size of 32x32x3 $\approx$ 3072 weights. Clearly, full connectivity in Regular Neural Nets is wasteful and the huge number of parameters would quickly lead to overfitting and cause very costly computational. While Convolutional Neural Networks

take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner

4.2.3 Architecture:

- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.
- RELU layer will apply an element-wise activation function, such as the max(0,x) thresholding at zero.
- POOL layer will perform a down-sampling operation along the spatial dimensions (width, height).
- FC (i.e. fully-connected) layer will compute the class scores. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

5 Projects:

5.1 Environments, dataset, frameworks:

5.1.1 Environments:

- OS: Linux - Ubuntu 16.04
- Programing Language: Python (3.5) - an interpreted high-level programing language, close to human language.

5.1.2 Dataset:

- CIFAR10: a dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images used for Classification.
- Object images: high-resolution image of object to identify the fraud on it.

5.1.3 Frameworks:

- Anaconda: Free, open-source packages management and deployment aimed distribution of Python.
- Scientific Python distributions: Numpy, Scipy, Matplotlib, ipython, jupyter, Pandas, SciKit learn.
- Pytorch: a python package that provides two high-level features:
  - Tensor computation (like Numpy) with strong GPU acceleration
  - Deep Neural Networks built on a tape-based autodiff system.

- OpenCV-Python: (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

5.2 Image Classification with CIFAR10 dataset:

5.2.1 Self-made convolution network:

A simple Convolution Network with 3 CONV layers with MaxPooling layers, 3 FC layers and the the ReLu activation function was made to understand roles of each layer in CNN and how to implement a CNN in Pytorch.

- Models:
  - conv layer 1: $Conv2d(3, 12, kernel\_size = (5, 5), stride = (1, 1), padding = (2, 2))$
  - pool layer: $MaxPool2d(kernel\_size = 2, stride = 2, padding = 0, dilation = 1, ceil\_mode = False)$
  - conv layer 2: $Conv2d(12, 18, kernel\_size = (5, 5), stride = (1, 1), padding = (2, 2))$
  - pool layer: $MaxPool2d(kernel\_size = 2, stride = 2, padding = 0, dilation = 1, ceil\_mode = False)$
  - conv layer 3: $Conv2d(18, 26, kernel\_size = (3, 3), stride = (1, 1))$
  - pool layer: $MaxPool2d(kernel\_size = 2, stride = 2, padding = 0, dilation = 1, ceil\_mode = False)$
  - fc layer 1: $Linear(in\_features = 234, out\_features = 200, bias = True)$
  - fc layer 2: $Linear(in\_features = 200, out\_features = 84, bias = True)$
  - fc layer 3: $Linear(in\_features = 84, out\_features = 10, bias = True)$

  With the ReLU activation function
- Train loss: 1.547 and Accuracy: 45.96%
- Test loss: 1.335 and Accuracy: 54.740%

5.2.2 Residual Network (a.k.a ResNet):

5.2.2.1 Why Residual Network?

When Microsoft Research released Deep Residual Learning for Image Recognition in 2015, these networks led to 1st-place winning entries in all five main tracks of the ImageNet and COCO 2015 competitions, which covered image classification, object detection, and semantic segmentation. The robustness of ResNets has since been proven by various visual recognition tasks and by non-visual tasks involving speech and language.

Deep networks are hard to train because of the notorious vanishing or exploding gradient problem - as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

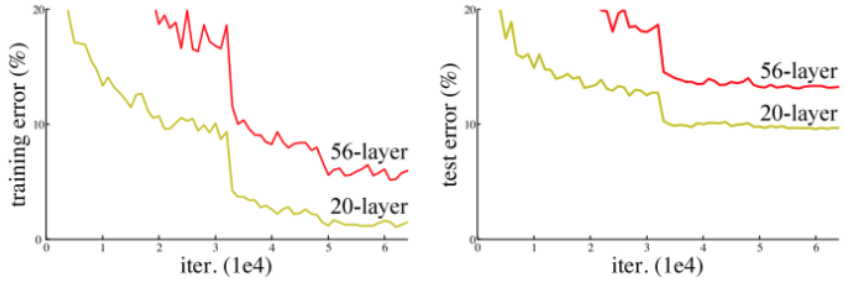


Figure 13: Increasing network depth leads to worse performance

The core idea of ResNet to tackle vanishing the above problems is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure:

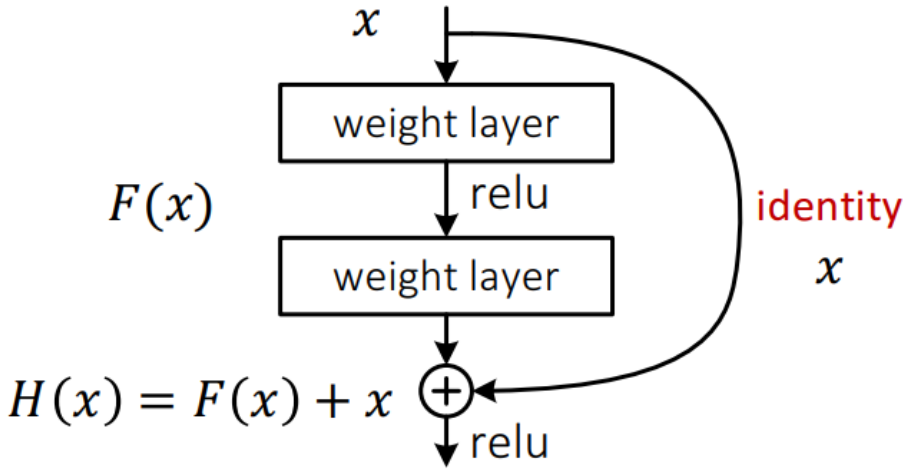


Figure 14: A residual block

Skip connections which allows you to take the activation from one layer and suddenly feed it to another layer even much deeper in the neural network. And using that, ResNet enables to train very, very deep networks.

Instead of learning a direct mapping of $x- > y$ with a function $H(x)$. Let define the residual function using $F(x) = H(x) - x$, which can be reframed into $H(x) = F(x) + x$, where $F(x)$ and x represent the stacked non-linear layers and the identity function respectively. The author's hypothesis is that it is easy to optimize the residual mapping function $F(x)$ than to optimize the original, unreferenced mapping $H(x)$.

The authors of [2] argue that stacking layers shouldn't degrade the network performance, because we could simply stack identity mappings (layer that doesn't do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlaying mapping. And the residual block above explicitly allows it to do precisely that.

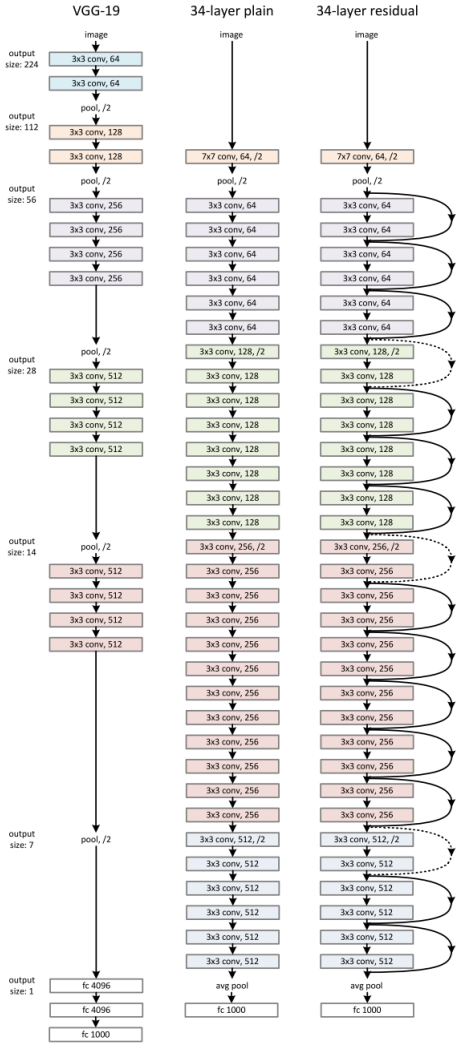


Figure 15: ResNet Architecture

Furthermore, in ResNet, the authors proposed using Batch Normalization which also helps the network to solve vanishing gradients problem.

- Address the problem of vanishing/exploding gradients
- Increase learning speed and solve many other problems
- Each activation in every iteration each layer is normalized to have zero mean and variance 1 over a minibatch
- Integrated into back-propagation algorithm

Batch Normalization:

- **Input:**Value of x over a mini-batch $B = \{x_{1...m}\}$. Parameters to be learned: γ, β
- **Output:**$\{y_i = BN_{\gamma,\beta}(x_i)\}$
  - Mini-batch mean: $\mu_B \leftarrow \frac{1}{m}\Sigma_{i=1}^{m} x_i$
  - Mini-batch variance: $\sigma_B^2 \leftarrow \frac{1}{m}\Sigma_{i=1}^{m}(x_i - \mu_B)^2$
  - Normalize: $\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
  - Scale and shift: $y_i \leftarrow \gamma\hat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$

5.2.2.2 ResNet 34:

To execute classification on CIFAR10 dataset, I would like to use ResNet 34 architecture in Residual Network framework. Due to availability on my own hardware to develop and debug and the experimental errors on the ImageNet data set.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Figure 16: Top-1 error(%, 10-crop testing) on ImageNet validation.

Intuitively, We have the architecture of the ResNets including ResNet 34.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 17: ResNets architecture

5.2.2.3 Implementation and Results:

The code of this project has been public on Github:

`https://github.com/t3min4l/pytorch-cifar10`

Result with 150 epochs, learning-rate: 0.1:
Best accuracy on Test set: 87.98% while loss is 0.35

5.2.3 Fraud detection on objects image:

5.2.3.1 Propose idea:

1. Find contour of the objects
2. Find minReact contain the contour
3. Find centroid of contour, and apply perspective transform
4. Draw midperpendiculars of the contour and split images in half
5. Detect differences by using function 'compare_ssim' from 'skimage'

5.2.3.3 Execution:

1. Find contour of the object:
   (a) Convert color from RGB to gray

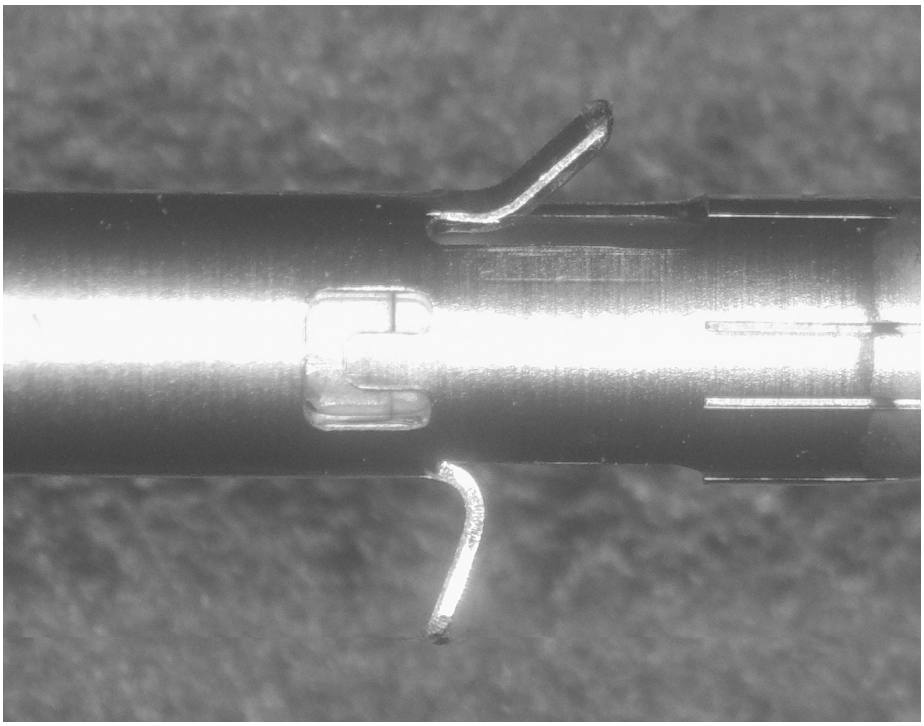


Figure 18: Convert image to gray scale

(b) Apply Canny edge detector

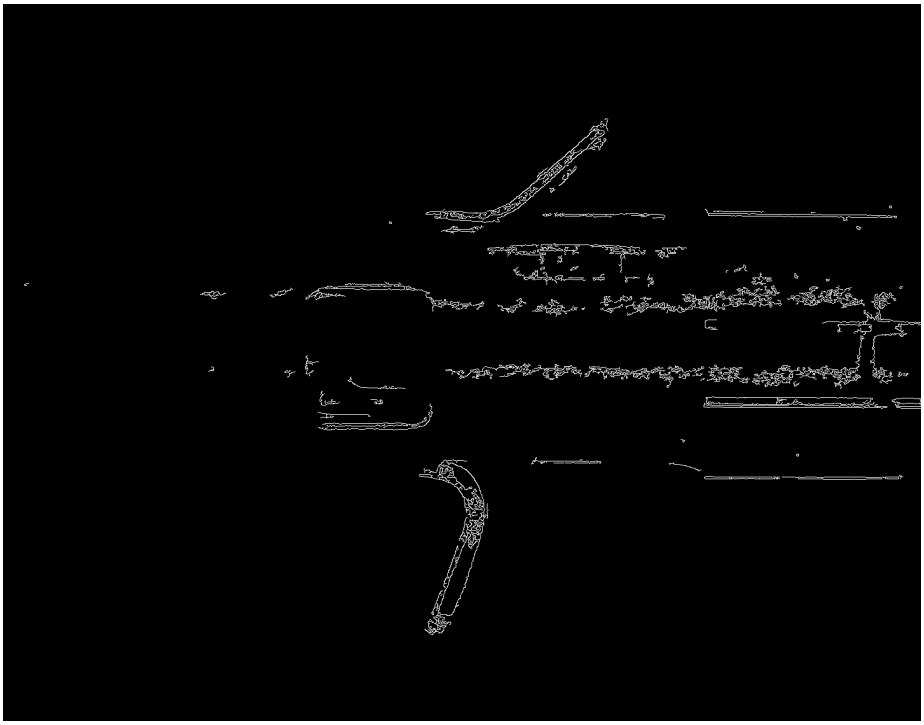


Figure 19: Edges detected by Canny algorithm

(c) Apply Morphological transformation

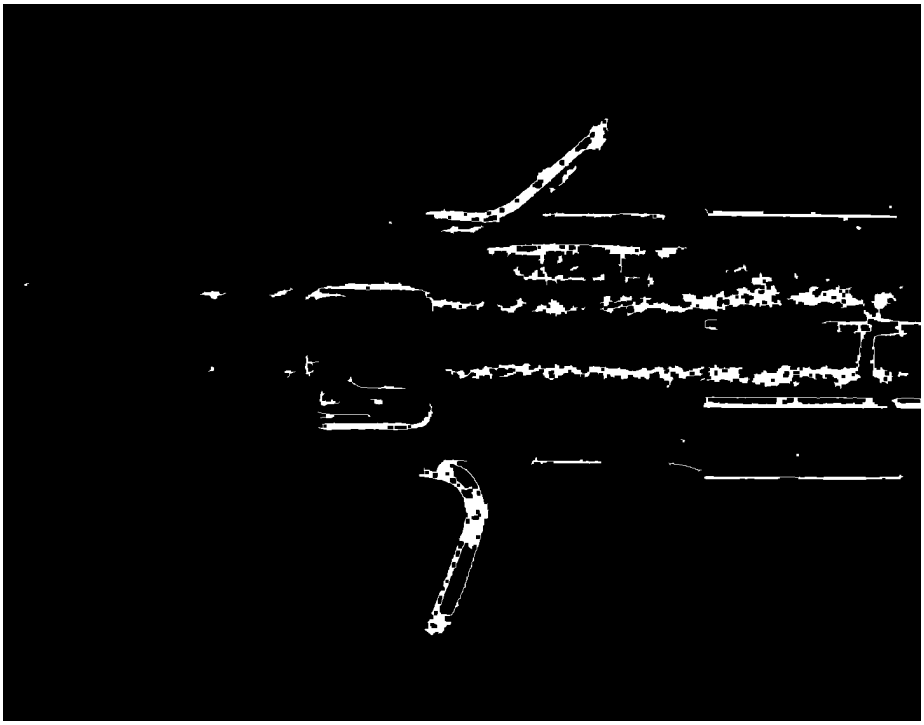


Figure 20: Morphological gradients transformation

(d) Find contours and draw them on the image

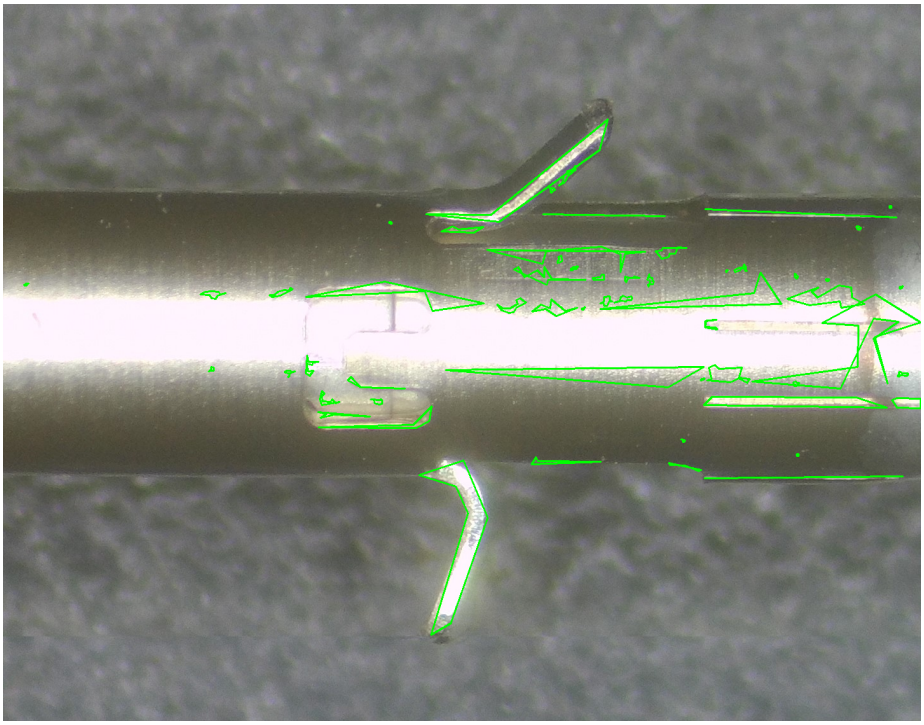


Figure 21: Contours drew on image

Due to the failure of finding contours of the whole object, I can not perform the remaining steps which I have proposed. The main reasons which lead to the failure are firstly, the background to likely salt-pepper noise with high ratio, secondly, the color the object is also like the color of the background, so that when applying Median filter or Gaussian filter then applying Canny edge detector, the result lost a lot of segmentations, and thirdly the angle of the light in the source input image is not optimal.
To improve the result, I would like to propose some ideas:

- Subtraction the background to extract only object
- Take other input image with black or white color and better angle of light.

Reference

[1] http://is.hust.edu.vn/ oanhnt/GRK54/IPCV/

[2] https://arxiv.org/abs/1512.03385

[3] http://cs231n.stanford.edu/

[4] https://www.pyimagesearch.com/opencv-tutorials-resources-guides/