# Course Syllabus

| | | |
|---|---|---|
| 1. | **Course ID** | 2110651 |
| 2. | **Course Credit** | 3 Credits |
| 3. | **Course Title** | Digital Image Processing |
| 4. | **Faculty / Department** | Engineering / Computer Engineering |
| 5. | **Semester** | 1 |
| 6. | **Academic year** | 2024 |
| 7. | **Instructor** | Punnarai Siricharoen, PhD |
| 8. | **Pre-requisite** | - |
| 9. | **Course Type** | Approved Elective |
| 10. | **Curriculum** | Computer Science |
| 11. | **Degree** | Master's degree |
| 12. | **Course date/time** | 3 Hours<br>Section 1: Thursday 9-12 (Section 1) Eng4 17-02<br>Section 5: Saturday 9-12 (Section 5) Eng4 18-16 |

**13. Course Description**

Visual perception, digitization and coding of images, converting pictures to discrete (digital) forms; image enhancement; image restoration including improving degraded low-contrast, blurred, or noisy pictures; image compression; data compression used in image processing; image segmentation referred to as first step in image analysis.

**14. Course Information**

14.1. Course Objectives

14.1.1. Be able to understand and describe fundamental image processing techniques

14.1.2. Be able to properly apply techniques for image analysis or image interpretation for the problems with intermediate complexity

14.1.3. Be able to design and implement image processing for image analysis and analyze the interpretation effectively

14.2. Content (see Tentative Schedule)

14.3. Teaching Methods

14.3.1 Onsite (Eng4, 17-02 (Thu), 18-16 (Sat))

14.3.2 myCourseVille course number : 2110651 password : imaging2024 for course materials

14.3.3 Discord (announcement, Q&A)

14.4. Evaluation

14.4.1. Attendance (in-class exercise .5 per week + bonus) 5%

14.4.2. Homework / Exercises 10%

14.4.3. Project 25%

14.4.4. Midterm Exam 30%

14.4.5. Final Exam 30%

*** Your score will be reset to 0 point if any of your work is plagiarized ***

*** Late submission for assignment within 6 hours, receive a deduction of 10% of your mark,
within 24 hours, receive a deduction of 20% of your mark,
NO MARK for the submission after 24 hours due.

14.5 Grading

| Score | Grade |
|---|---|
| >= 85 | A |
| >= 75 | B+ |
| >= 70 | B |
| >= 65 | C+ |
| >= 60 | C |
| >= 55 | D+ |
| >= 50 | D |
| < 50 | F |

**15. Books**

15.1. Books (Reference)

Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison-Wesley

15.2. Additional materials: Related proceedings and Journals

# Tentative Schedule

| Week / Date<br>Sec 1,5 | Topic | Assignment<br>HW (2 weeks due) |
|---|---|---|
| 1 – 8,10/8 | Introduction to image processing and software preparation, Examples of fields using digital image processing | |
| 2 – 15,17/8 | Image Acquisition / Sampling and Quantization / Arithmetic Operations | |
| 3 – 22,24/8 | Histogram and Spatial Filtering | HW #1 Spatial Domain |
| 4 – 29,31/8 | Frequency Domain & Filtering in FD | |
| 5 – 5,7/9 | Wavelet and Multiresolution Processing | |
| 6 – 12,14/9 | Image Restoration | |
| 7- 19,21/9 | Image Compression | HW #2 Frequency & Research paper (Future image processing in your thought?) |
| 8-25-26,28/9 | Midterm exam (1 A4 paper, writing)<br>Sat 28/9/2567 9:00-12:00 (to confirm) | |
| 9- 3,5/10 | Fundamental Image Segmentation I (Online - Graduation Ceremony) | |
| 10 – 10,12/10 | Research discussion / Morphology<br>Feature Extraction and Analysis | |
| 11-17,19/10 | Object Recognition (traditional /modern) | **Project Assignment** |
| 12-24,26/10 | Object Recognition II | |
| 13-31,2/11 | Image Segmentation II | HW #3 Segmentation + Image Augmentation |
| 14-7,9/11 | Object Detection : Paper discussion & applications | |
| 15-14,16/11 | Image Generative Adversarial and other Imaging Applications – discussion (read and discuss) | |
| 16-21,23/11 | Invited Speaker | |
| 17 | - **Final exam** Sat 7/12/2567 9-12<br>- **Project Presentation** - Wed 12/12/2567 – 18 – 21 | |

<table>
<tr><th>Week</th><th>Topic</th></tr>
<tr><td>#1</td><td>Introduction to image processing and software preparation, Examples of fields using digital image processing<br>- Keywords<br>- Dicom image</td></tr>
<tr><td>#2</td><td>Image Acquisition / Sampling and Quantization / Arithmetic Operations</td></tr>
<tr><td>#3</td><td>Histogram and Spatial Filtering<br>-> Topics include complex exponential signals, linear space-invariant systems, <u>2D convolution</u>, and filtering in the spatial domain.<br><br>Homework #1:<br>- create your filter</td></tr>
<tr><td>#4</td><td>Filtering in Frequency Domain<br>-> 2D Fourier transform, sampling, discrete Fourier transform, and filtering in the frequency domain.</td></tr>
<tr><td>#5</td><td>Wavelet and Multiresolution Processing</td></tr>
<tr><td>#6</td><td>Image Restoration<br>Image Recovery I: matrix-vector notation for images, inverse filtering, constrained least squares (CLS), set-theoretic restoration approaches, iterative restoration algorithms, and spatially adaptive algorithms.<br>Image Recovery II: a stochastic perspective. Topics include: Wiener restoration filter, Wiener noise smoothing filter, maximum likelihood and maximum a posteriori estimation, and Bayesian restoration algorithms.<br>Homework#2: Reading a scale</td></tr>
<tr><td>#7</td><td>Image Lossless and Lossy Compression I<br>Lossless: In this module we introduce the problem of image and video compression with a focus on lossless compression. Topics include: elements of information theory, Huffman coding, run-length coding and fax, arithmetic coding, dictionary techniques, and predictive coding.</td></tr>
<tr><td>#8</td><td>Lossy: lossy image compression. Topics include: scalar and vector quantization, differential pulse-code modulation, fractal image compression, transform coding, JPEG, and subband image compression. Additional video compression</td></tr>
</table>

| | |
|---|---|
| #9 | Midterm |
| #10 | Fundamental Image Segmentation I |
| #11 | Fundamental Image Segmentation II<br>- watersheds and K-means algorithms, and other advanced methods.<br>- graph-based image segmentation |
| #12 | Morphology<br>Motion Estimation (basic) Activity find 3D Gradients:<br>https://pythonhosted.org/bob.ip.optflow.hornschunck/py_api.html<br>-การเปลี่ยนแปลงเทียบเฟรมแรก<br>-การเปลี่ยนแปลงเทียบเฟรมก่อนหน้า<br>ทำ Sobel แกนนอน<br><br>**Mini Project: Object Detection without machine learning (Final programming)** |
| #13 | **Quiz #2**<br>Geometric Transformation + Image Registration<br>https://scikit-image.org/docs/stable/auto_examples/registration/plot_register_translation.html<br><br>**Homework**: Data Augmentation (input / output function) |
| #14 | Object Recognition<br>- OCR –> HOG + SVM, LeNet<br>http://amroamroamro.github.io/mexopencv/opencv/svm_hog_ocr_digits_demo.html<br>- Other applications |
| | Image Generative Adversarial and other Imaging Applications – discussion (read and discuss)<br>- Image augmentation<br>- Style transfer<br>- Super-resolution |
| #15 | Review Technology / Project Presentation |
| #16 | Invited Speaker: Asst. Dr. Kitiwat Khamwan (Department of Radiology, Faculty of Medicine)<br>**สรุป 1 page -> mind map (infographics)** |
| #17 | Final Exam |

# Lecture 01 Introduction to Digital Imaging

**Topic**: Introduction

**Instructor**: Punnarai Siricharoen, PhD

**Time**: Wednesday, 11 August 2021, 13-16 pm

**Objectives**:

1. To introduce Digital Imaging vs. Human Vision and its challenges
2. To describe the principal areas of digital imaging and applications
3. To explain about this course
4. To use python for read/write/show an image

**Contents**:

1. Introductory of Imaging vs. Human Vision and Challenges
2. Principal areas of digital imaging and applications
3. About this course
4. Python for read/write/show an image

**Teaching and Learning Methods**:

1. Lecture 01 slide (posted on myCourseVille)
2. Zoom meeting
3. Google Colab for coding python

**Teaching Plan:**

1. Introductory of Imaging vs. Human Vision and Challenges (30 mins)
2. Principal areas of digital imaging and applications (30 mins)
3. Break (10 mins)
4. About this course (30 mins)
5. Python for read/write/show an image (30 mins)
6. In-class Exercises in myCourseVille (20 mins)
7. Questions (10 mins)

**Evaluation:**

- In-class participation
- In-class Exercises in myCourseVille

**Complete the question set**

1 What is the imaging application you found most impressive (for you)?

Compose an answer

Students gave examples of imaging applications

2 What is the only File format is allowed to be checked after submission in myCourseVille?

Pick a choice:

- ( ) .ipynb
- ( ) .word
- (•) .pdf
- ( ) .py

3

```
We read the "kitty.png" image below. Write a code snippet to change the area

x = 100, y = 100 to x = 200, y = 200

to blue color and display it correctly.

import cv2

img = cv2.imread("kitty.png")
```

Compose an answer

```
img[100:201,100:201] = [255,0,0]; plt.imshow(img[:,:,::-1]); plt.show()
```

## Readings:

- Chapter 1, Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley

# Lecture 02 Image Formation, Color Models and Arithmetic Operations

**Topic**: Image Formation, Color Models and Arithmetic Operations

**Instructor**: Punnarai Siricharoen, PhD

**Time**: Wednesday, 18 August 2021, 13-16 pm

**Objectives**:

1. To learn how an image is formed.
2. To understand different color models and their purposes.
3. To be able to use arithmetic operations for processing an image.

**Contents**:

1. Image formation
2. Color models
3. Arithmetic Operations
4. Exercises in Python

**Teaching and Learning Methods**:

1. Lecture 02 slide (posted on myCourseVille)
2. Zoom meeting
3. Google Colab for coding python
4. Discord for Consultation

**Teaching Plan:**

1. Image formation (30 mins)
2. Color models (20 mins)
3. Break (10 mins)
4. Arithmetic Operations (45 mins)
5. Applications (15 mins)
6. Python Exercises (20 mins)
7. In-class Exercises in myCourseVille (20 mins)
8. Questions (10 mins)

## Evaluation:

- In-class participation

- In-class Exercises in myCourseVille

Question List :

**1**

If you want to quantize an image into 4 bins in the following shades: black, dark gray, light gray and white. A pixel intensity of 128 have which of the color shades?

Pick a choice:

- **black**
- **dark gray**
- **light gray**
- **white**

Correct Answer(s) (may be used for autograding)

- light gray

**2**

Which of the following is not the chromatic component of a color model?

Pick a choice:

- **H of HSV**
- **L of CIELAB**
- **A of CIELAB**
- **Grayscale**

inside a course

Correct Answer(s) (may be used for autograding)

- L of CIELAB
- Grayscale

**3**

What happen if we multiply an image with 255?

Pick a choice:

- **An image will be a little brighter**
- **An image will have higher contrast**
- **An image will have no change**
- **An image become white**

Correct Answer(s) (may be used for autograding)

- An image become white

## Readings:

- Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley

- Chapter 2 – Digital Image Fundamentals
- Chapter 3 – 3.2 Some basic gray level transformations
- Chapter 6 – Color Image Processing (6.2, 6.3 Color models)

# Lecture 03 Histogram and Spatial Filtering

**Topic**: Histogram and spatial filtering

**Instructor**: Punnarai Siricharoen, PhD

**Time**: Wednesday, 25 August 2021, 13-16 pm

**Objectives**:

1. To understand how to describe an image using histogram and histogram equalization techniques
2. To understand filtering techniques, e.g., image smoothing or sharpening

**Contents**:

1. Histogram
2. Spatial Filtering
3. Exercises in Python

**Teaching and Learning Methods**:

1. Lecture 03 slide (posted on myCourseVille)
2. Zoom meeting
3. Google Colab for coding python
4. Discord for Consultation

**Teaching Plan:**

9. Histogram (30 mins)
10. Exercise #1 Histogram (15 mins)
11. Break (10 mins)
12. Spatial Filtering (60 mins)
13. Python Exercises (30 mins)
14. In-class Exercises in myCourseVille (15 mins)
15. Questions (10 mins)

**Evaluation:**

- In-class participation
- In-class Exercises in myCourseVille

Question List :

**1**

Which of the following statements is true?

Pick a choice:

- Histogram Equalization enhances image contrast
- Histogram Equalization enhances image blurriness
- Histogram Equalization smoothens image intensity
- All is true

Correct Answer(s) (may be used for autograding)

- Histogram Equalization enhances image contrast

**2**

A histogram of 8-bit image is created for each bin to cover 8 intensity levels. Calculate number of bins of this histogram.

Pick a choice:

- 8
- 32
- 128
- 255
- 256

Correct Answer(s) (may be used for autograding)

- 32

**3**

Which of the following filters when applied on an image can make the image blurred?

Pick a choice:

- First-order derivative filter
- Roberts filter
- Laplacian filter
- box filter

Correct Answer(s) (may be used for autograding)

- box filter

**4**

Which one is not the edge detection filter?

Pick a choice:

- Averaging filter
- Roberts filter
- Laplacian filter
- First-order derivative filter

Correct Answer(s) (may be used for autograding)

- Averaging filter

5

What is the individual value in the box filter with the size of 5x5 ?

Pick a choice:

- 1/25
- 0.11
- 0.25
- 1

Correct Answer(s) (may be used for autograding)

- 1/25

## Readings:

- Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley
    - Chapter 3, Section 3.3-3.8

# Lecture 04 Frequency Domain

**Topic**: Frequency Domain

**Instructor**: Punnarai Siricharoen, PhD

**Time**: Wednesday, 1 and 8 September 2021, 13-16 pm

**Objectives**:

- To understand frequency domain for image analysis
- To be able to describe frequency domain of the image using Fourier transform
- To be able to apply basic filters on an image in frequency domain

**Contents**:

1. Fourier Transform (Frequency Domain)
2. Filtering in Frequency Domain
3. Exercises in Python

**Teaching and Learning Methods**:

1. Lecture 04 slide (posted on myCourseVille)
2. Zoom meeting
3. Google Colab for coding python
4. Discord for Consultation

**Teaching Plan:**

**1 September 2021**

1. Recap and Introduction to Frequency Domain (15 mins)
2. Fourier Transform (30 mins)
3. Exercises by hand (15 mins)
4. Examples of Fourier transform by visualization (30 mins)
5. Break (10 mins)
6. Python Exercises (30 mins)
7. In-class Exercises in myCourseVille (15 mins)
8. Questions (10 mins)

**8 September 2021**

9. Filtering in Frequency Domain – Lowpass filter (20 mins)
10. Highpass filter (20 mins)
11. Convolution Properties (20 mins)

12. Python Examples and exercises (30 mins)

**Evaluation:**

- In-class participation
- In-class Exercises in myCourseVille

1) What domain can be analyzed using Fourier Transform?

(a) Spatial

(b) Frequency Ans

(c) Temporal

(d) None of them

2) Once you (manually) calculate Fourier transform of an image, where is the location of the lowest frequency values?

(a) close to (0,0) Ans

(b) center

(c) bottom left Ans

(d) All of them

4.2 Filtering

- Ideal filter

เพื่อแก้ปัญหาความไม่ต่อเนื่องของภาพหลังจากที่มีการกรองด้วยฟิลเตอร์แบบอุดมคติ ได้มีการออกแบบฟิลเตอร์ท

Two further filters are designed to cope with discontinuity problems in an image after filtered by ideal filter which rapidly attenuate the frequencies from passing to stopping. Practical filters include Butterworth filter and Gaussian filter.

**- Butterworth filter**

Butterworth filter is designed using Laplace transform where the transfer function in frequency domain is defined as follows:

Butterworth filter is frequency response designed using a basic signal in discrete-time Fourier transform below.

$$a^n u[n] \leftrightarrow \frac{1}{1 - ae^{-j\omega}}$$

Where $|a| < 1$

- Gaussian low-pass filter

Gaussian low-pass filter is derived from Gaussian distribution function defined in Eq.XX which has a bell-shape so the frequencies are slowly attenuated from low to high frequency as shown in the Figure XX.

G(x) = (1 / (σ√(2π))) * e^(-(x^2 / (2σ^2)))

In this equation:

- G(x) is the value of the Gaussian function at position x.
- σ (sigma) is the standard deviation of the distribution, which controls the spread or width of the Gaussian curve.
- π is the mathematical constant pi.

Convolution – output is similar to the kernel.

Filtering – output is the correlation of the input and the corresponding kernel.

But if the filter or kernel is symmetric, the output from convolution and filtering should be the same

- Applications of FT
  - Image enhancement – use high-pass filter to increase the sharpness in an image similar to applying second order Laplacian filter on an image and combine with the original image
  - Image compression – remove some high frequencies as they are details in an image but might be less important, especially very high frequency.
  - Machine learning – cancer diagnosis, Fourier transform can be used to identify patterns of an object based on frequencies hiding in the signals
- Fast Fourier transform for fingerprint enhancement
  - Fast Fourier Transform of each homogeneous block is computed; the magnitude of the FFT is taken and raised to some real value *k.*
  - The key for enhancing fingerprint image quality is shown in the figure below where fourier transform is calculated and modified with magnitude of Fourier spectrum raised to some real value k and Butterworth filtering.

- Encryption – watermasking
  - After the calculation of a DFT, the substitution of the bits to be integrated has been performed by modifying the LSBs of the quantized coefficients. This is performed by comparing the parity of the successive coefficients.
  - Watermark was integrated using the mid-band coefficients (mixed time and frequency components of the signal) which allowed us to guarantee imperceptibility while being robust to compression attacks and filtering.
- Compression
  - JPEG image compression algorithms

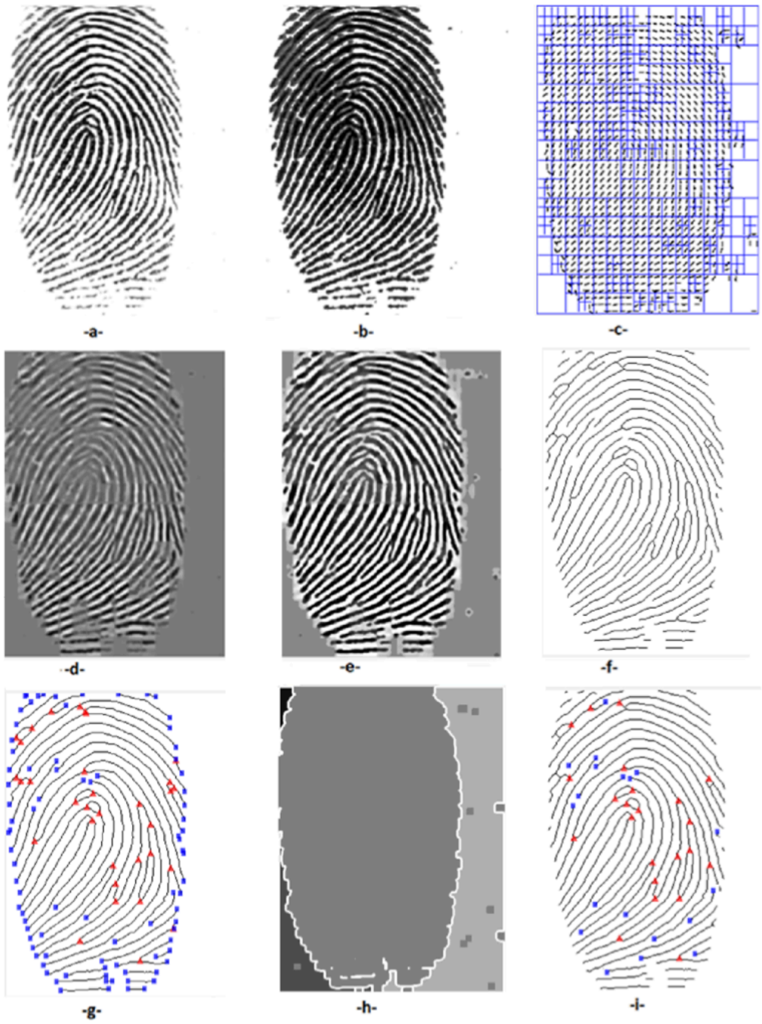
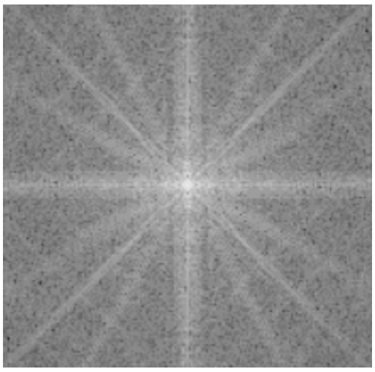



**Figure 2:** a-Original fingerprint image 103_1 of DB1 FVC2004, -b- Image after histogram equalization, -c- Decomposition of the image according to the orientation field, -d- Output after processing $FFT \times |FFT|^k$ with $k$=0.45, and applying a gaussian filter for smoothing, -e-Output after Butterworth filtering and binarizing with adaptive thresholding, -f- Thinned fingerprint, -g- Output with detected minutiae, -h- Boundary of the original image, (the process is shown in section 5.1), -i- Detected minutia after post-processing, Number of bifurcation minutia (in **red** triangles): 23. Number of end minutia (in **blue** squares): 19.

- FFT for pre-processing an image
  - Rotating an image for extract rotation-variant features

3) Consider the Fourier Spectrum (after apply fft2 and fftshift functions) below, what would be the original image?



(a)

(b)

(c) Ans

(d)

**Readings:**

- Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley

  - Chapter 4 Frequency Domain

Different types of noise

1) Salt and Pepper noise

During digitization process, image corruption can occur and shows large values compared with the strength of the image signal, so that value can be digitized into black or white colors. Impulse noise is found in a situations where quick transients, such as faulty switching take place during imaging. For 8-bit image, a and b can be 0 and 255, respectively.

2) Gaussian noise

Gaussian noise is an idealized form of white noise which is caused by random fluctuations and the distribution is show in bell-shape in Figure xx. The representation of the noise is defined by the equation below.

In digital photography, images captured under low-light conditions or with high ISO settings can exhibit Gaussian noise. In medical imaging, such as X-ray, MRI, or CT scans, Gaussian noise can be introduced due to factors like electronic sensor noise or variations in radiation exposure. This noise can affect the clarity and quality of medical images. Images obtained from remote sensing satellites or aerial photography can be affected by Gaussian noise caused by atmospheric interference, sensor imperfections, or electronic noise. Reducing this noise is crucial for accurate interpretation and analysis of the images. During image compression (e.g., JPEG compression), quantization errors can introduce Gaussian noise into the compressed image. Gaussian noise can be intentionally added or present in images during various image editing tasks, such as blurring, sharpening, or resizing. Understanding and managing this noise is essential for maintaining image quality.

4) Speckle noise

Speckle noise is a type of noise commonly encountered in various imaging applications, particularly in radar and ultrasound imaging, but it can also occur in other domains. It appears as a granular or grainy pattern in images, which can degrade image quality and make it difficult to analyze or interpret. Here are some examples of speckle noise in different imaging applications:

Medical Ultrasound Imaging:

Speckle noise is a well-known issue in ultrasound images. It appears as a granular pattern in soft tissue and organs, making it challenging for healthcare professionals to visualize and diagnose medical conditions accurately.

Radar Imaging:

In radar images, speckle noise is caused by the interference of electromagnetic waves. It can obscure important details in the image and affect the ability to detect and track objects, especially in synthetic aperture radar (SAR) imagery.

Satellite and Aerial Imaging:

Speckle noise can occur in images obtained from satellites or aircraft, particularly in synthetic aperture radar (SAR) and interferometric synthetic aperture radar (InSAR) applications. Reducing speckle noise is crucial for accurate Earth observation and remote sensing.

Microscopy:

Speckle noise can be observed in microscopy images, especially in techniques like laser speckle contrast imaging (LSCI) and optical coherence tomography (OCT). It can affect the clarity and contrast of microstructures.

Industrial Inspection:

In industrial applications such as non-destructive testing and quality control, speckle noise can be present in images captured using techniques like laser-based or ultrasonic imaging. Reducing speckle noise is essential for accurate defect detection.

Sonar Imaging:

In underwater sonar imaging, speckle noise can be introduced by reflections and interference of sound waves. Removing speckle noise is critical for improving the quality of underwater sonar images.

3D Printing and Scanning:

Speckle noise can be present in 3D scanning and printing applications, affecting the quality of scanned objects or the appearance of 3D printed surfaces.

4) Periodic noise

5) Other types of noise

Rayleigh Noise:

Medical Imaging: In medical imaging, such as X-ray or MRI, Rayleigh noise can be caused by factors like electronic sensor noise or variations in radiation exposure. It can appear as random fluctuations in pixel values and affect the clarity of the medical images.

Gamma Noise:

Nuclear Medicine: Gamma noise is commonly encountered in nuclear medicine imaging, particularly in single-photon emission computed tomography (SPECT) and positron emission tomography (PET) scans. It is caused by the statistical nature of radioactive decay and can affect the accuracy of image reconstruction.

Exponential Noise:

Fluorescence Microscopy: In fluorescence microscopy, exponential noise can result from factors like fluctuations in fluorescent dye intensity or electronic noise. It can manifest as variations in pixel values and impact the quality of fluorescence images.

Uniform Noise:

Digital Photography: Uniform noise is often observed in digital photographs taken under well-lit conditions. It can be caused by factors like variations in sensor sensitivity and electronic noise. In images, uniform noise appears as a subtle, evenly distributed graininess.

Remote Sensing:

Satellite Imaging: In satellite imagery used for Earth observation and remote sensing applications, various types of noise, including Rayleigh, gamma, exponential, and uniform noise, can be present due to atmospheric interference, sensor limitations, or transmission artifacts. Removing or mitigating this noise is crucial for accurate data analysis.

Industrial Inspection:

Quality Control: In industrial applications involving quality control and non-destructive testing, gamma noise can be encountered in images captured using X-ray or gamma-ray techniques. Uniform noise may also be present in images obtained from machine vision systems used for product inspection.

Astronomy:

Astronomical Imaging: In astrophotography and astronomical imaging, Rayleigh noise can be present due to the random fluctuations in the number of detected photons from celestial objects. This noise affects the clarity of astronomical images and requires advanced noise reduction techniques.

# Lecture 08 Image Segmentation

1. Similarity
2. Discontinuities
3. Combined method

Point, Line and Edge detection

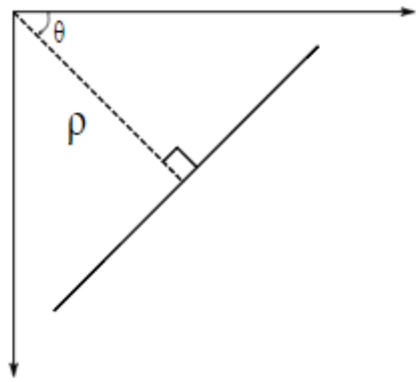
Hough Transform

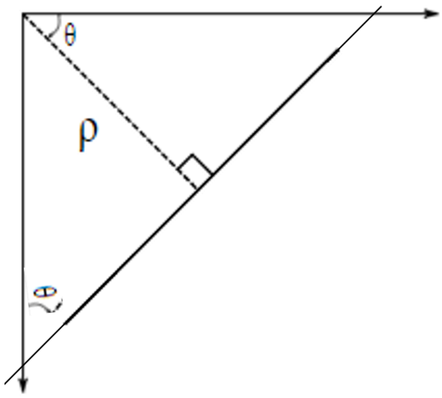
(1) Intro (2) Equation (3) practical

Hough transform is generally used to detect whether a set of pixels lie on curves or lines of some specified objects. Patented by Paul Hough in 1962. The version of the method as it is used today was invented by Richard Duda and Peter Hart in 1972. Hough Line transform is to find every possible line that could fit a set of pixels in an image. For example a linear equation:

$$y = mx + c$$

Representing any line with slope or gradient, m and c is vertical intercept (y-intercept) or the value of y when x equals to zero.




Now, to prove the equation from standard relationships:

Slope, m = $-\frac{\cos\theta}{\sin\theta}$ and $\frac{\rho}{c} = \sin\theta$, so we have

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta}$$

But in the context of the Hough transform, we are more interested in parameter space (r, $\theta$ ) rather than the (m, b) space because the former has advantages in terms of dealing with vertical lines and computational efficiency.

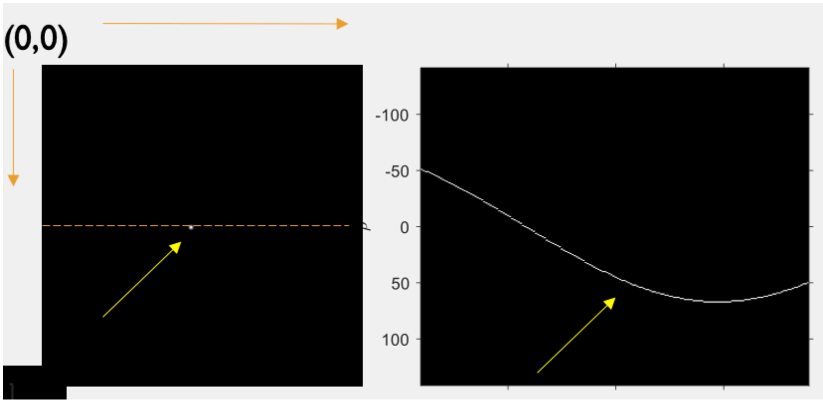
By isolating $\rho$ from our polar equation:

$$\rho = xcos(\theta) + ysin(\theta)$$

Where $\rho$ is the normal (perpendicular) distance from the origin to the closest point on the line. and $\theta$ is the angle between the x-axis and this normal.

This is the equation you see in the Hough transform, and it allows us to represent any line by a point in ($\rho$,$\theta$) space. The Hough transform technique accumulates votes in this parameter space for the potential lines in an image, and peaks in this space correspond to the lines in the original image.

Now let’s take a look at the example,
When you have a point in an image, you can represent in ($\rho$,$\theta$) a number of lines in 360 degrees but 360 degrees contain redundancy, for example, 30 degrees is the same as 210 degrees, so we can consider degrees in the range [0, 180) or [-90, 90). For a point in an image, you can generate accumulator array (H(($\rho$,$\theta$) ) that represent particular line in the right hand side. All value in the curvature line on the right hand side has H($\rho$,$\theta$) = 1 because we are considering only one line first. If we have more points which can vote on the accumulator array H(($\rho$,$\theta$) and the position of the peak gives the $\rho$ and $\theta$ values of the corresponding line.




In the Hough Transform, the parameter space is typically represented using the variables $\rho$ (rho) and $\theta$ (theta). Every point in the ( $\rho$ , $\theta$ ) space corresponds to a line in the image space.

To explain the Hough Transform in terms of its operation, let's break it down step by step:

1 Edge Detection: Before applying the Hough transform, usually an edge detection algorithm like the Canny edge detector is applied to the image. This will give you a binary image where edges are marked with white (or 1) and non-edges with black (or 0).

2 Initialization: Create an accumulator array, H( $\rho$ , $\theta$ ), initialized with zeros. The array's dimensions are determined by the possible ranges of $\rho$ and $\theta$ .

3 **For Each Edge Pixel** in the Image: For every white pixel (x, y) in the edge-detected image, do the following:

a. For each possible value of $\theta$ (from 0 to 180 degrees, or -90 to 90 degrees depending on your convention):

i. Calculate $\rho$ using the equation: $\rho = x\cos(\theta) + y\sin(\theta)$.

ii. Increase the accumulator value for H( $\rho$ , $\theta$ ) by 1.

4 **Thresholding**: After processing all edge pixels, the accumulator will have peaks which correspond to the lines in the image. By selecting a threshold, you can determine which peaks (and corresponding lines) are significant.

5 **Detect Lines**: Peaks in the accumulator array H( $\rho$ , $\theta$ ) represent the lines in the original image. The position of the peak gives the $\rho$ and $\theta$ values of the corresponding line.

To "call" H( $\rho$ , $\theta$ ) essentially means to index into the accumulator array for a specific $\rho$ and $\theta$ value. For example, if you want to know how many votes the line with parameters = 50 $\rho$ =50 and $\theta$ =30 got, you'd check the value at
H(50,30) in the accumulator array.

In practice, programming libraries such as OpenCV have implementations of the Hough Transform that handle these details for you, but understanding the underlying process can be helpful when fine-tuning parameters or when trying to implement the algorithm from scratch.

Thresholding

Thresholding is used to isolate objects from their background. The success of these operations depends very much on choosing an appropriate threshold level. For example of the image below, if the threshold is too low, it can reduce object size, but if the threshold is too high, it can include extraneous background. The automatic Automatic method for choosing a best threshold is needed.

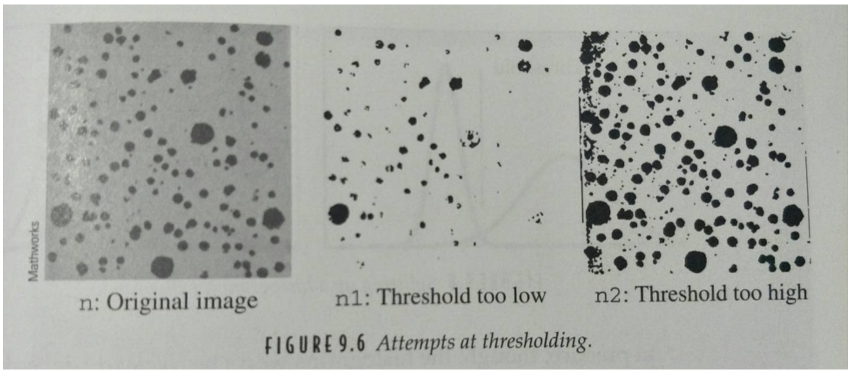


FIGURE 9.6 *Attempts at thresholding.*

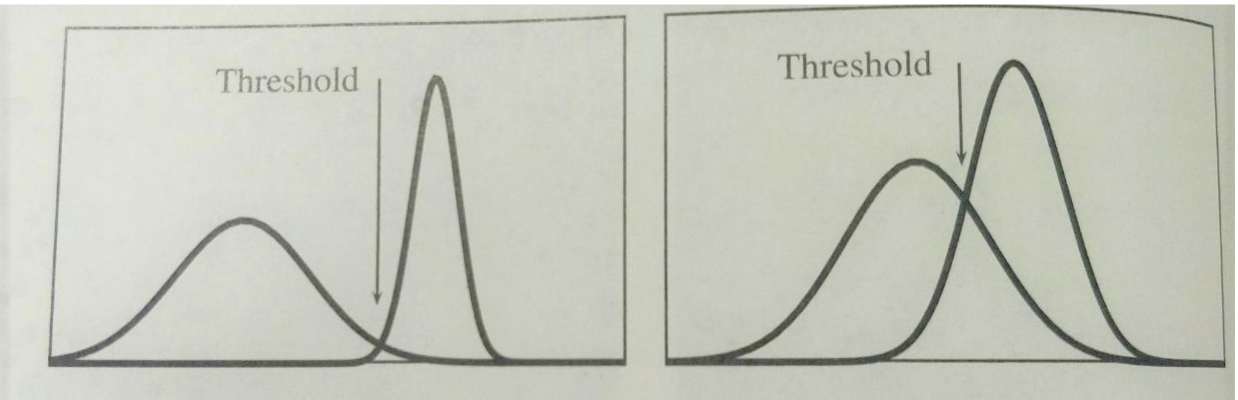


FIGURE 9.8 *Splitting up a histogram for thresholding.*

Global threshold

- The same value is used

for the whole image.

Optimal global threshold

- Based on the shape of the current image histogram. Searching for valleys, Gaussian distribution etc.

Local (or dynamic) threshold

- The image is divided into non-overlapping sections, which are thresholded one by one.

Otsu's Thresholding

Otsu's thresholding is a method to automatically determine an optimal threshold for binarizing an image. It aims to maximize the variance between two classes of pixels (foreground and background) so that they are distinctly separable.

Here's a simplified walkthrough of the method:

1. Histogram Computation:

Start by computing the histogram of the grayscale image. This will give you the probability of occurrence of each gray level in the image. Let's call this p(i) for gray level i.

2. Cumulative Sums:

For each gray level t, compute the cumulative sum up to t for the probabilities. This gives you the probability of a pixel being background (below the threshold) or foreground (above the threshold). These are represented as:

- $\omega_0(t)$ = sum of $p(i)$ for $i = 0$ to $t$ (Background class probability)
- $\omega_1(t)$ = 1 - $\omega_0(t)$ (Foreground class probability)

3. Cumulative Means:

Compute the cumulative mean up to gray level t and then compute the mean level for the entire image. These are represented as:

- $\mu_0(t)$ (Mean brightness of background)
- $\mu_1(t)$ (Mean brightness of foreground)
- $\mu_T$ (Total mean brightness)

4. Maximize Between-class Variance:

The aim is to find a threshold t that maximizes the between-class variance $\sigma_B^2(k)$ which is defined as:

$$\sigma_B^2(t) = \omega_0(t) \times \omega_1(t) \times [\mu_0(t) - \mu_1(t)]^2$$

5. Determine Optimal Threshold:

Iterate through all possible thresholds t and compute $\sigma_B^2(k)$for each. The t which gives the maximum is the Otsu's threshold.

By following these steps, Otsu's method determines a threshold in a way that the "difference" (in terms of brightness) between the background and foreground is maximized.

For a complete mathematical derivation and in-depth understanding, one would need to delve into the formulation of within-class variance and show that maximizing between-class variance is equivalent to minimizing the within-class variance. However, the steps provided above give an intuitive and simplified understanding of the algorithm's process.

MuT is independent bu Mu0 and Mu1 are the function of threshold k

We can rearrange the term and get the between class variance in with one Mu0 and MuT and the find $k\ which\ maximize(\sigma_B^2)$

Between-class variance ($\sigma_B^2(k)$) = $\frac{(\mu_T\omega_0(k) - \mu_0(k))^2}{\omega_0\ (k)(1-\omega_0\ (k))}$

# Lecture 09 Morphology

Morphology in image processing is a technique used to analyze and manipulate the shape and structure of objects within an image. It involves applying a set of mathematical operations to an input image to extract useful information or to modify the image in a way that enhances certain features or removes unwanted details. Morphological operations are primarily used in binary or grayscale images and are especially useful in tasks like image segmentation, noise reduction, object extraction, and feature enhancement.

Some common morphological operations include:

1. Dilation: Dilation is used to expand the boundaries of foreground objects in an image. It involves a sliding structuring element (a small binary matrix) over the image, and for each position of the element, if any of its pixels overlaps with the foreground, the corresponding pixel in the output image is set to the foreground as well.
2. Erosion: Erosion is the opposite of dilation and is used to shrink or erode the boundaries of foreground objects. It is also performed using a structuring element, but the output pixel is set to the background if any of the element's pixels overlap with the background in the input image.
3. Opening: Opening is a sequence of an erosion operation followed by a dilation operation. It is typically used for noise reduction and separating objects that are close to each other.
4. Closing: Closing is a sequence of a dilation operation followed by an erosion operation. It is used for closing small gaps between objects or connecting nearby objects.
5. Morphological gradient: It is the difference between the dilation and erosion of an image and is used to highlight the boundaries of objects in an image.
6. Top-hat and bottom-hat transforms: These operations highlight the bright and dark regions in an image, respectively. The top-hat transform is the difference between the input image and its opening, while the bottom-hat transform is the difference between the closing and the input image.
7. Morphological operations can be very useful in various image processing tasks, including image segmentation, feature extraction, noise removal, and object detection. They are particularly effective when dealing with binary or grayscale images, and the choice of structuring element and the order in which these operations are applied can be customized to suit the specific requirements of a particular image processing task.

$\ominus$ Circled minus operator
$\oplus$ Circled plus operator
$\otimes$ Circled times operator
$\circ$ Circle, open circle operator
$\cdot$ Dot operator

⊛ Hit-or-miss operator (asterisk in a circle, circled asterisk)

$$\odot$$

$$\hat{B} = \{w|w = -b\ for\ b \in B)$$

Reflection of B is written as $\hat{B}$ which is w where w is equal to minus b for all variable b in set B

b = (b1,b2) $\hat{B}$ considers -b or (-b1,-b2)

# Lecture 10 Object Recognition I

Feature engineering is the process of selecting and transforming raw data into a format that is suitable for machine learning algorithms. In the context of machine learning, a "feature" refers to an individual piece of information or attribute that is relevant to a problem or task. Feature engineering involves creating, selecting, and manipulating these features to improve the performance of a machine learning model.

Here are some key aspects of feature engineering:

- Feature Selection: This involves choosing which features to include in your machine learning model. Selecting relevant features and removing irrelevant or redundant ones can help reduce the dimensionality of the data and improve model efficiency.
- Feature Extraction: Feature extraction involves transforming the existing features into a new set of features that may be more informative. This can be achieved through techniques like Principal Component Analysis (PCA), Singular Value Decomposition (SVD), or various mathematical operations.
- Feature Creation: Sometimes, new features are created based on domain knowledge or insights from the data. For example, you might create a feature that calculates the ratio of two existing features, or you might generate new features from text data, such as word counts or sentiment scores.
- Handling Categorical Data: Many machine learning algorithms require numerical data, so categorical features (e.g., colors, categories, or labels) often need to be encoded into numerical representations. This can involve one-hot encoding, label encoding, or more advanced methods like target encoding.
- Dealing with Missing Data: Feature engineering can also involve strategies for handling missing data, such as imputation or creating new features to indicate missing values.
- Scaling and Normalization: It's important to scale or normalize features to bring them to a common scale, which can be particularly important for algorithms sensitive to feature magnitudes, like distance-based methods.
- Feature Transformation: Feature engineering may also involve transforming data to meet the assumptions of the chosen machine learning algorithm. For example, you might apply logarithmic or power transformations to make data more normally distributed.
- Time-Series Features: In time-series data, creating features from historical data points can be crucial for predictive modeling. This might involve calculating moving averages, time lags, or seasonality components.
- Feature engineering is a critical step in the machine learning pipeline, as the quality and relevance of the features can significantly impact the model's performance. Effective feature

engineering requires a good understanding of the problem domain and the characteristics of the data. It often involves an iterative process of experimentation and evaluation to identify the best set of features for a given task.

In machine learning and data science, data is typically divided into three main subsets: the training dataset, the testing dataset, and the validation dataset. Each of these subsets serves a specific role in developing and evaluating machine learning models:

- Training Dataset:

Role: The training dataset is the largest of the three subsets and is used to train the machine learning model. The model learns patterns, relationships, and features from this data to make predictions or classifications.
Process: During the training phase, the model is exposed to examples from the training dataset, and it adjusts its internal parameters based on these examples to minimize the error or loss function.
Size: The training dataset typically comprises 60-80% of the available data.

- Validation Dataset:

Role: The validation dataset is used to fine-tune model hyperparameters and monitor its performance during training. It helps prevent overfitting, which occurs when a model performs well on the training data but poorly on unseen data.
Process: After each training epoch (or a certain number of training steps), the model's performance is evaluated on the validation dataset. This evaluation helps in selecting the best hyperparameters and deciding when to stop training.
Size: The validation dataset is typically smaller than the training dataset, comprising 10-20% of the available data.

- Testing Dataset:

Role: The testing dataset is reserved for the final evaluation of the trained model. It provides an unbiased assessment of the model's performance on unseen data, simulating how the model will perform in the real world.
Process: Once the model has been trained and fine-tuned using the training and validation datasets, it is tested on the testing dataset. The model's performance metrics, such as accuracy, precision, recall, or F1-score, are computed to assess its generalization ability.
Size: The testing dataset should be large enough to provide statistically significant results but should not overlap with the training or validation datasets.
In summary, the training dataset is used to teach the model, the validation dataset is used to optimize hyperparameters and monitor training progress, and the testing dataset is used to evaluate the model's performance in real-world scenarios. Separating data into these three subsets

helps ensure that the machine learning model is robust, generalizes well, and is not just memorizing the training data (overfitting). It is important to keep the testing dataset separate from the training and validation datasets to provide an unbiased assessment of the model's capabilities.

User
what is the blind dataset?
ChatGPT
The term "blind dataset" typically refers to a dataset that is used for an evaluation or test phase in which the model or system being assessed has never seen this data during training or development. The purpose of a blind dataset is to provide an unbiased and objective assessment of the model's performance or the system's capabilities in a real-world scenario.

Key characteristics of a blind dataset include:

Unseen Data: The blind dataset contains examples or samples that the model has not encountered during its training or validation phases. This ensures that the model is tested on data it has never learned from.

Objective Evaluation: By using a blind dataset, researchers or practitioners can assess how well their model generalizes to new, unseen data. This is crucial for understanding how the model will perform in practical applications.

Avoiding Data Leakage: Keeping the blind dataset separate from the training and validation datasets is important to prevent any data leakage or bias that might occur if the model has inadvertently learned from or seen the evaluation data beforehand.

Blind datasets are commonly used in various fields, including machine learning, artificial intelligence, and experimental sciences, to gauge the effectiveness and reliability of models, algorithms, or systems in real-world scenarios. They provide a critical step in the evaluation process to ensure that a model's performance metrics are trustworthy and applicable to new, unseen data.

User
what is loss function and what is the examples and use case?

ChatGPT
A loss function, also known as a cost function or objective function, is a critical component in machine learning and optimization. It is a mathematical function that quantifies the difference between the predicted values generated by a machine learning model and the actual target values in the training dataset. The primary purpose of a loss function is to measure how well or poorly a model is performing, with the goal of minimizing this function during the training process.

Here's an overview of the concept of loss functions, along with examples and use cases:

Purpose of a Loss Function:

Optimization: The primary purpose of a loss function is to serve as a guide for optimization algorithms. Machine learning models aim to find the set of model parameters (weights and biases) that minimize the value of the loss function. These optimized parameters make the model's predictions as close as possible to the true target values.
Examples of Loss Functions:

a. Mean Squared Error (MSE):

Formula: MSE = (1/n) $\Sigma$ (y_true - y_pred)^2
Use Case: Regression problems where the goal is to predict a continuous numerical value (e.g., predicting house prices, stock prices).
b. Binary Cross-Entropy Loss (Log Loss):

Formula: Binary Cross-Entropy = - $\Sigma$ (y_true * log(y_pred) + (1 - y_true) * log(1 - y_pred))
Use Case: Binary classification problems, such as spam detection or medical diagnosis.
c. Categorical Cross-Entropy Loss (Softmax Loss):

Formula: Categorical Cross-Entropy = - $\Sigma$ (y_true * log(y_pred))
Use Case: Multiclass classification problems, where the model needs to assign one of several possible categories to each input (e.g., image classification, sentiment analysis).
d. Hinge Loss (SVM Loss):

Formula: Hinge Loss = max(0, 1 - y_true * y_pred)
Use Case: Support Vector Machines (SVMs) and linear classifiers in classification problems.
e. Huber Loss:

Formula: Huber Loss = $\Sigma$ { |y_true - y_pred| if |y_true - y_pred| <= $\delta$ , 0.5 * $\delta$ * (|y_true - y_pred| - $\delta$ ) if |y_true - y_pred| > $\delta$ }
Use Case: Robust regression, which is less sensitive to outliers compared to Mean Squared Error.

Use Cases of Loss Functions:

Model Training: Loss functions are used during the training process to compute the error or discrepancy between the model's predictions and the actual target values. Gradient descent optimization algorithms then adjust the model's parameters to minimize this loss.

Hyperparameter Tuning: Different loss functions may be suitable for different types of machine learning tasks. Researchers and practitioners often experiment with various loss functions as part of hyperparameter tuning to improve model performance.

Evaluation: Loss functions are also used to evaluate and compare the performance of different models. Lower loss values generally indicate better model performance.

In summary, a loss function quantifies the error between predicted and actual values, and its choice depends on the specific machine learning task. Minimizing the loss function during training leads to models that make more accurate predictions on unseen data.

**Gradient Descent:**
  - **Understanding Words:** Think of this as a person trying to find the lowest point in a valley by feeling the slope of the ground and taking steps downwards.
  - **Example:** Basic gradient descent updates weights by a fraction of the gradient.

- **Stochastic Gradient Descent (SGD):**
  - **Understanding Words:** It's like the person in the valley taking steps one at a time, considering one piece of the terrain (data point) at a time.
  - **Example:** It updates weights by considering one data point at a time.

- **Mini-batch Gradient Descent:**
  - **Understanding Words:** The person takes into account a small group of terrains (data points) to decide the direction to move.
  - **Example:** Updates weights by considering a small batch of data points.

- **Momentum:**
  - **Understanding Words:** Imagine our person in the valley wearing roller skates. Momentum helps to overcome small humps and avoid getting stuck in shallow, non-optimal valleys.
  - **Example:** It considers the previous gradients to keep the momentum.

- **Adam:**

- **Understanding Words:** Adam is like having a GPS that adapts the learning rate besides having momentum. It combines the best of two worlds: Momentum and RMSprop.
- **Example:** It adjusts the learning rates of each parameter, benefiting from adaptive learning rates.

- **RMSprop:**
  - **Understanding Words:** RMSprop changes the learning rate during training. It's like our person adjusting their stride length based on the steepness of the terrain.
  - **Example:** It adapts the learning rates during training, making it easier to find a minimum.

the pros and cons of some common optimizers:

1. Gradient Descent:
Pros:

Simple and easy to understand and implement.
Works well for convex loss functions, ensuring global minimum is reached.
Cons:

Computationally expensive for large datasets as it uses all data points for a single update.
May not be suitable for non-convex loss functions or in presence of saddle points.

2. Stochastic Gradient Descent (SGD):
Pros:

Computationally efficient as it updates weights using a single data point at a time.
Capable of escaping shallow local minima due to its noisy gradient updates.
Cons:

The noisy updates can make the convergence to the minimum slow and erratic.
Requires careful choice of learning rate and might need annealing strategies.

3. Mini-Batch Gradient Descent:
Pros:

Strikes a balance between SGD and gradient descent, making it computationally efficient.
Reduces variance of the updates, leading to more stable convergence.
Cons:

Choice of batch size becomes an additional hyperparameter that needs to be tuned.
Less capable of escaping non-optimal local minima compared to SGD.

4. Momentum:
Pros:

Accelerates convergence by adding a fraction of the previous weight update.
Helps in overcoming small humps and avoid getting stuck in shallow non-optimal valleys.
Cons:

Adds complexity due to momentum hyperparameter which needs to be tuned.
Can overshoot the minimum due to accumulated momentum.

5. Adam:
Pros:

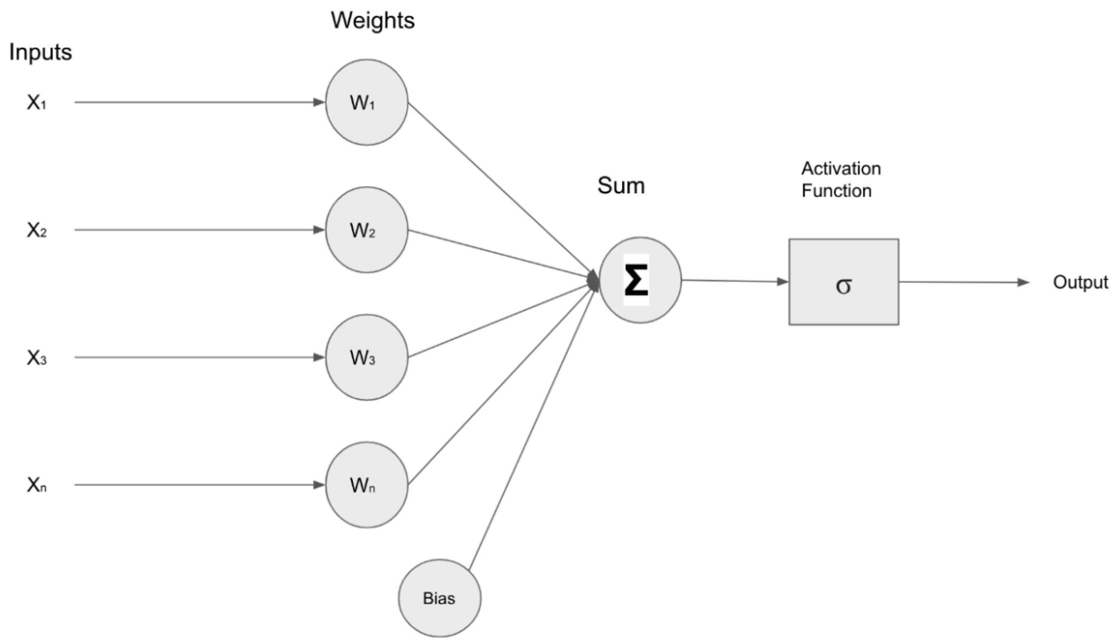
Combines benefits of Momentum and RMSprop, generally leading to fast convergence.
Adaptive learning rates make it suitable for non-stationary objectives and noisy data.
Cons:

Requires more memory for storing moving averages.
Can be sensitive to the choice of hyperparameters, especially the learning rate.

6. RMSprop:
Pros:

Adapts the learning rate during training which can be beneficial for non-stationary objectives.
Often works well even with the default settings.
Cons:

Not as robust as Adam as it lacks momentum, making it slower in convergence.
Hyperparameters like decay factor need tuning.
Each optimizer comes with its own set of advantages and drawbacks. The choice of optimizer can depend on various factors, such as the specific problem, model architecture, and dataset characteristics. Experimentation is often necessary to find the best optimizer for a given neural network model.




Warm starting in the context of training machine learning models, including neural networks, refers to the practice of initializing a model with parameters (weights and biases) from a previously trained model rather than starting with random initialization. This concept is particularly useful in various scenarios such as transfer learning, fine-tuning, and continuous or online learning. Here is a detailed explanation:

Cosine annealing is a learning rate scheduling method used in training deep neural networks. It is applied to adjust the learning rate during training, allowing the model to learn more adaptively. Cosine annealing smoothly varies the learning rate between a maximum and minimum boundary in a cosine function manner.

Vision transformer

When using the Vision Transformer (ViT) for image classification, it is common to only use the class token for the final prediction, even though all tokens, including those representing patches of the image, have been updated during the processing. Here's a simplified explanation of why only the class token is typically used:

1. Focus on the Goal: Specific Objective

* The main goal is to classify the entire image, not its individual parts. The class token is specifically designed to gather and synthesize information from all parts of the image for this global classification task.

2. Efficiency: Simplifying the Output

* Using only the class token simplifies the output of the model. It avoids having to manage and interpret the outputs from all the individual patch tokens, making the model more efficient and straightforward.

3. Special Role: Unique Purpose of Class Token

* The class token has a special role—it starts without being tied to any specific part of the image, allowing it to freely accumulate and represent global information. Patch tokens, on the other hand, start with a focus on specific regions of the image.

4. Training Stability: Consistent Approach

* Using only the class token for classification provides a consistent and stable approach during training, helping the model to learn a clear objective and generalize better to unseen data.

5. Flexibility: Adaptability to Different Tasks

* In some cases, models might indeed use all tokens for tasks like object detection or segmentation where local information and spatial relationships are crucial. But for classification, the global perspective of the class token is usually sufficient and effective.

Simplified Analogy:

Imagine a team working on a project. While every member brings valuable insights, the team leader (class token) has the role of consolidating all information and presenting the final conclusion. This approach keeps the presentation (output) clear, focused, and aligned with the main objective, instead of having every team member present their individual parts, which might complicate and lengthen the presentation.

So, using only the class token in ViT for image classification helps keep the model focused on its main goal, efficient, and stable in its learning process.

# Lecture 12 Segmentation II

**Course Title: Advanced Image Segmentation with Deep Learning**

**Objective**:

To empower students with practical skills and knowledge to implement and optimize deep learning models for image segmentation tasks, focusing on advanced techniques and best practices.

**Hour 1: Introduction to Image Segmentation in Deep Learning**

**Lecture: Concepts and Techniques (30 minutes)**

1. Brief review of image segmentation basics.

Exploration of popular segmentation architectures:

- FCN
- U-Net
- Mask R-CNN
- DeepLabv3+

2. Discussion on loss functions and metrics used in segmentation.

- Pixel-wise Softmax with Cross-Entropy
- Focal loss
- Dice loss

https://www.v7labs.com/blog/semantic-segmentation-guide
https://www.superannotate.com/blog/image-segmentation-for-machine-learning

3. Applications

- Medical imaging
- Self-driving cars
- Scene understanding
- Arial image processing

Segmentation Vs. Object Detection

4. Practice

- Pre-implemented U-Net
- Transfer learning
- Augmentation

**Exercise: Review of a Pre-implemented Model (30 minutes)**

1. Students will analyze a pre-implemented segmentation model (e.g., U-Net) on a sample dataset.
2. Tasks include understanding model architecture, data preprocessing, and output interpretation.

**Hour 2: Practical Implementation of Segmentation Models**

**Lecture: Data Preparation and Augmentation (15 minutes)**

1. Discussion on the importance of data in segmentation tasks.

2. Techniques for data augmentation and preprocessing.

**Exercise: Implementing a Segmentation Model (45 minutes)**

- Students will implement a segmentation model (e.g., U-Net) using a deep learning framework (e.g., TensorFlow, PyTorch).
- Guidance will be given on model configuration, data augmentation strategies, and training techniques.

**Hour 3: Optimization and Evaluation of Segmentation Models**

**Lecture: Model Optimization Techniques (15 minutes)**

- Exploration of techniques for improving model performance:
- Transfer learning
- Hyperparameter tuning
- Regularization methods

**Exercise: Model Evaluation and Optimization (45 minutes)**

- Students will evaluate their models using various metrics (e.g., IoU, Dice coefficient).
- Tasks will include optimizing the model through techniques discussed in the lecture and analyzing the impact of these optimizations.

**Closing and Q&A Session (15 minutes)**

- Recap of key takeaways from the course.
- Open floor for questions, discussion, and feedback.

**Materials and Resources Required:**

- Access to a deep learning environment (Google Colab, Jupyter notebooks).
- Sample datasets for image segmentation tasks.
- Pre-implemented segmentation models and code templates.

**Key Takeaways for Students:**

- In-depth understanding of advanced image segmentation models and techniques.
- Practical experience in implementing, optimizing, and evaluating segmentation models.
- Confidence to undertake image segmentation tasks in real-world projects.

This course is designed to be interactive, balancing theoretical knowledge with practical exercises to ensure students gain hands-on experience and a deep understanding of image segmentation in deep learning.

# Lecture 13 Generative Adversarial Networks

Designing a 3-hour lecture and Python exercise for Generative Adversarial Networks (GANs) in image processing for beginners is an extensive task. Here's a general outline of the lecture and an exercise you can use as a starting point. Feel free to adjust the content and exercises as needed:

Lecture Title: Introduction to Generative Adversarial Networks (GANs) for Image Processing

Duration: 3 hours

Introduction (15 minutes)

Welcome and introduction to the course.
Briefly explain the importance of GANs in image generation and manipulation.
Overview of the agenda for the lecture.
Part 1: Understanding GANs (45 minutes)

Conceptual Understanding

- What are GANs, and why are they important in image processing?
- The architecture of GANs: Generator and Discriminator.
- How GANs learn: the adversarial training process.
- Common applications of GANs in image processing.

Part 2: Building a Simple GAN in Python (1 hour)

- Python Exercise

- Set up the Python environment (using Jupyter Notebook or a similar tool).
- Import necessary libraries (e.g., TensorFlow or PyTorch, NumPy).
- Create a simple GAN architecture:
- Design a generator network.
- Design a discriminator network.
- Define loss functions.
- Implement the training loop:
- Generate synthetic images.
- Train the discriminator.
- Train the generator.
- Visualize the progress and results during training.
- Break (15 minutes)

Part 3: Tips for Training GANs (30 minutes)

- Best Practices and Common Challenges

- Discuss common issues when training GANs (mode collapse, vanishing gradients, etc.).
- Strategies to mitigate these issues.
- Learning rate, batch size, and other hyperparameters.
- Evaluation metrics for GANs.

Part 4: Conditional GANs (30 minutes)

- Advanced Concepts

Introduce conditional GANs and their applications.
Modify the GAN architecture to handle conditional inputs (e.g., labels).
Explain how to generate images with specific attributes.
Part 5: Generating Images with Pretrained GANs (15 minutes)

- Use of Pretrained Models

Discuss the availability of pretrained GAN models.
How to use pretrained GANs to generate images.
Show practical examples using a pretrained GAN model.
Break (15 minutes)

- Part 6: Ethical Considerations and Future Trends (15 minutes)

Ethical and Social Implications

Briefly discuss ethical considerations when working with GANs.
Mention current and future trends in GAN research.
Part 7: Q&A Session (15 minutes)

Open the floor for questions from participants.
Python Exercise (30 minutes)

Advanced Exercise

Challenge the participants to apply what they've learned.
Ask them to train a GAN on their own dataset or use a publicly available one.

Encourage them to experiment with different hyperparameters.
Conclusion and Recap (15 minutes)

Summarize the key takeaways.
Provide resources for further learning.
Thank the participants and encourage them to continue exploring GANs.
Remember that this is a high-level overview, and you may need to adjust the content based on your audience's familiarity with Python and image processing. Be sure to provide ample support during the exercise and encourage participants to ask questions and explore further on their own.

---

*DCGAN*

DCGAN (Deep Convolutional Generative Adversarial Network), pix2pix, and CycleGAN are all types of Generative Adversarial Networks (GANs), but they are designed for different tasks and operate in distinct ways:

Architecture: It is one of the first architectures to modify the standard GAN with deep convolutional networks.
Goal: DCGAN is used for unsupervised learning. It aims to generate new images that are similar to the training dataset without any paired examples.
Key Features:
It uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.
Introduces batch normalization to improve stability.
Uses ReLU activation in the generator and LeakyReLU in the discriminator.
Applications: Typically used for image generation tasks such as creating new human faces, objects, and scenes that resemble the input data.
pix2pix:
Architecture: This is a conditional GAN that works with paired images.
Goal: pix2pix is used for image-to-image translation tasks where the input is a conditioned image and the output is the corresponding transformed image.
Key Features:
It requires paired training data, meaning each input image is directly matched with an output image.
It uses a U-Net-based architecture for the generator.
The discriminator, called PatchGAN, classifies patches of the image as real or fake, rather than the whole image.
Applications: Suitable for tasks like converting satellite photos to maps, black-and-white photos to color, sketches to photographs, etc.

CycleGAN:
Architecture: Like pix2pix, CycleGAN is used for image-to-image translation, but it doesn't require paired images.
Goal: It is designed to learn the translation without paired examples, using unpaired datasets.
Key Features:
Introduces cycle consistency loss to enforce forward and backward consistency, which means the translation from one domain to another and back to the original should result in the original input image.
Also uses PatchGAN for the discriminator.
Uses two generators and two discriminators for both directions of image translation.
Applications: Useful for tasks where paired training data is not available or hard to obtain, such as turning horses into zebras, apples into oranges, or changing seasons in landscapes.
In summary, the main differences lie in the type of data they are trained on (paired vs. unpaired) and the specific tasks they are designed for (image generation vs. image-to-image translation). DCGAN is focused on generating new images from noise, pix2pix requires exact pairs of images to learn translations, and CycleGAN can learn this translation from unpaired datasets, making it more versatile when paired data is not available.

---

*Latent space vector*

The latent space in the context of Generative Adversarial Networks (GANs) refers to an abstract, multidimensional space that encodes compressed representations of the data the network is trained on. Here's a breakdown of its physical meaning or interpretation:

High-Dimensional Representation: Latent space is typically high-dimensional, meaning that it can consist of tens, hundreds, or even thousands of dimensions. Each dimension represents a feature or characteristic that the model has learned during training.

Compression of Information: In latent space, complex data (like images) are represented in a compressed form. This means that each point in the latent space corresponds to a condensed version of the input data, capturing its most essential characteristics as determined by the network.

Continuous and Structured: Latent space is often structured so that similar points represent similar content. For instance, in a GAN trained on images of faces, two points close together in the latent space might correspond to faces with similar features, while points further apart might represent faces that differ significantly.

Interpolation and Sampling: It is possible to interpolate between points in the latent space, which means generating new data samples that combine features from existing ones. This is useful for creating new images that are variations or mixtures of learned data.

Control and Manipulation: If a GAN is well-trained and the latent space is well-structured, it may be possible to control and manipulate specific features of the generated data by adjusting certain dimensions in the latent space. For example, changing one dimension could alter the hair color in generated images of faces.

Vector Arithmetic: Latent spaces often allow for vector arithmetic. For example, by finding the vector direction that represents smiling in a latent space of faces, you can add this vector to any other point in the latent space to generate a new image of a face with a smile.

In physical terms, you can think of the latent space as a kind of "genetic code" for the data. Just as DNA encodes the information needed to generate an organism with certain characteristics, the latent space encodes the information needed to generate data samples with certain features. However, unlike biological DNA, latent space is an abstract mathematical construct without a direct physical manifestation.

*Pix2Pix*

Pix2Pix, short for "Pixel to Pixel," is a type of Generative Adversarial Network (GAN) designed for image-to-image translation tasks. This architecture allows for converting an image from one representation of a scene to another, like turning a sketch into a photorealistic image, converting satellite imagery into maps, or black-and-white photos into color. Here's a breakdown of its architecture and functionality:

Components: Pix2Pix comprises two main components:

Generator (G): The generator takes an input image (like a sketch) and generates a corresponding output image (like a colored image). It typically uses a U-Net based architecture, which is effective in capturing and using fine details from the input image.
Discriminator (D): The discriminator's role is to differentiate between the real images (actual photographs) and the fake images generated by the generator. It uses a patch-based approach, where it tries to classify each patch of the image as real or fake, rather than the entire image at once.
Training Process:

During training, the generator tries to produce images that look as realistic as possible, while the discriminator tries to get better at distinguishing real images from generated ones.
The generator and discriminator are trained simultaneously in a min-max game, where the generator aims to minimize a loss function while the discriminator aims to maximize it.
Loss Functions:

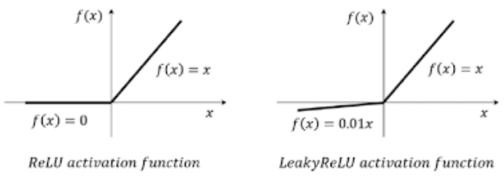
Adversarial Loss: Ensures the generated images are indistinguishable from real images in the dataset.
L1/L2 Loss: Ensures the generated images are close to the target images in terms of pixel values. The L1 loss is often preferred as it encourages less blurring.
Image-to-Image Translation Tasks:

Pix2Pix is used for tasks where the input and output images are paired, meaning for each input image, there's a corresponding ground truth output image. For instance, a sketch paired with its colored version.
Applications:

Pix2Pix has a wide range of applications like photo enhancement, image colorization, style transfer, and more.
Limitations:

It requires paired training data, which can be difficult to obtain for certain applications.
The model may struggle with high-resolution images or extremely detailed translations.
Pix2Pix represents a significant advancement in the field of image processing and has paved the way for many subsequent innovations in image translation and generative models.

*Leaky ReLU*




The use of the LeakyReLU activation function in the discriminator of a Deep Convolutional Generative Adversarial Network (DCGAN) is a strategic choice aimed at improving the performance and stability of the model during training. Here's why LeakyReLU is preferred:

Avoiding Dead Neurons: In a traditional ReLU (Rectified Linear Unit) activation function, if the input to a neuron is negative, the output is zero. This can lead to a problem known as "dying ReLU," where neurons effectively become inactive and stop learning because they consistently output zero. LeakyReLU addresses this issue by allowing a small, non-zero gradient when the unit is not active (i.e., when the input is negative), preventing neurons from dying.

Gradient Flow: By allowing a small gradient when the input is negative, LeakyReLU ensures that there is always some gradient flowing through the network. This helps in the backpropagation process, ensuring that weights can continue to be updated and the model can keep learning. In the context of a GAN, maintaining gradient flow is crucial for the discriminator to effectively learn to distinguish between real and generated images.

Model Stability: In GANs, balancing the training of the discriminator and generator is key to achieving stable training. LeakyReLU helps maintain this balance by avoiding issues with dying neurons in the discriminator. This, in turn, enables the discriminator to provide meaningful feedback to the generator throughout the training process.

Improved Learning Dynamics: The slight slope for negative inputs in LeakyReLU can result in better learning dynamics compared to ReLU. In the adversarial setting of a GAN, where the discriminator and generator are constantly evolving based on each other's performance, this can lead to more robust learning and better overall performance.

Empirical Success: The use of LeakyReLU in the discriminator of DCGANs and other architectures has been empirically shown to improve performance in many cases. The choice is often based on experimental results where LeakyReLU demonstrates superior performance in terms of convergence speed and final model quality.

In summary, the incorporation of LeakyReLU in the discriminator of a DCGAN is mainly to prevent the issue of dead neurons, ensure continuous gradient flow, and improve the overall stability and performance of the adversarial training process.

*Ethical concerns*

Generative Adversarial Networks (GANs) have brought significant advancements in the field of AI and machine learning, especially in tasks related to image generation, enhancement, and translation. However, their capabilities also raise several ethical concerns that need careful consideration:

Deepfakes and Misinformation: One of the most prominent concerns with GANs is their ability to create realistic but fake images, videos, and audio recordings, known as deepfakes. These can be used to spread misinformation, manipulate public opinion, or defame individuals, posing a significant threat to personal reputation, political integrity, and even national security.

Consent and Privacy Violations: GANs can generate realistic images of people who never consented to have their likeness used, including generating images of people in compromising or controversial situations. This raises serious privacy concerns and questions about the ethics of using someone's likeness without their permission.

Bias and Discrimination: Like many AI models, GANs can inherit and amplify biases present in their training data. If the training data is biased, the generated outputs can perpetuate stereotypes and discriminatory practices. This is particularly concerning in applications like facial recognition, where biased data can lead to unfair or prejudiced outcomes.

Artistic and Intellectual Property Rights: GANs can replicate the style of artists or generate new artworks, raising questions about intellectual property rights and the originality of art. Determining the ownership of AI-generated content and the ethical implications of AI mimicking human creativity are ongoing debates.

Economic Impact: The ability of GANs to generate realistic images and videos could impact industries like photography, modeling, and film, potentially displacing jobs. While they can also create new opportunities and efficiencies, the displacement of traditional roles raises concerns about economic impacts and the future of work.

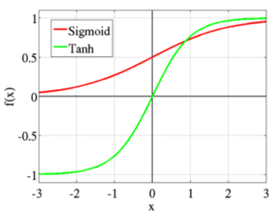
Unintended Harmful Uses: GANs can be used to create highly realistic and convincing synthetic media for malicious purposes such as creating fake evidence, engaging in fraud, or generating harmful content like realistic images of child exploitation.

Legal and Regulatory Challenges: The rapid development of GAN technology poses challenges for legal and regulatory frameworks, which may struggle to keep pace with the ethical implications and potential misuses of this technology.

Psychological Impact: The ability to create realistic fake content can blur the line between reality and fiction, potentially causing psychological impacts on individuals who may find it increasingly difficult to discern truth in the media they consume.

Addressing these ethical concerns requires a multi-faceted approach, including developing robust legal frameworks, ethical guidelines for developers and users, and public awareness and education about the capabilities and limitations of GANs. Moreover, ongoing research into bias mitigation, watermarking synthetic content, and developing detection methods for deepfakes is crucial in managing the ethical implications of GAN technology.

*Tanh*



The use of the hyperbolic tangent (tanh) activation function in the output layer of the generator in Generative Adversarial Networks (GANs) is a design choice that has specific advantages, particularly for image generation tasks:

Output Range: The tanh function has an output range of

[

−

1

,

1

]

$[-1,1]$, which can be more suitable for image data that is typically normalized to be in the range of

[

−

1

,

1

]

$[-1,1]$ or

[

0

,

1

]

$[0,1]$. Using tanh ensures that the output of the generator naturally falls within this range, making it directly compatible with the common image data preprocessing methods.

Centered Data: Since tanh outputs values between -1 and 1, the generated data is centered around zero. Centering the data can lead to better convergence properties during training, as it ensures that the feature values have similar distributions, which can be beneficial for gradient-based optimization methods.

Smoother Gradients: The tanh function provides a smoother gradient than a hard sigmoid function. Smoother gradients can be beneficial during the training process, as they allow for more nuanced updates to the model parameters, which can help in finding better minima during the optimization process.

Empirical Success: The choice of tanh has been empirically found to work well in practice for many types of GANs, especially those used for generating images. The specific reasons for this empirical success can be complex and depend on the particular architecture and application, but it has been observed that GANs using tanh in the generator often produce high-quality images.

It's important to note that while tanh is a common choice, it's not the only option. Some GAN architectures may use different activation functions based on the specific requirements of the task or based on empirical results. For instance, if the data is normalized to a

[

0

,

1

]

[0,1] range, a sigmoid function might be used instead. The choice of activation function is part of the model design process and can depend on various factors including the nature of the data, the architecture of the model, and the specific goals of the task.

Differences between dying ReLU and Gradient vanishing problem

Key Differences

Nature of Problem: Dying ReLU is specific to the ReLU activation function and occurs when neurons stop responding to inputs altogether. Gradient vanishing is a broader problem related to the small gradients and affects the learning process in deep networks.

Impact on Training: Dying ReLU affects individual neurons, leading to a reduction in model capacity, while gradient vanishing affects the entire depth of the network, slowing down or halting the training of early layers.

Solutions: While both problems can be mitigated by choosing appropriate activation functions, the specific strategies differ. Leaky versions of ReLU can prevent dying neurons, while alternative activation functions, better weight initialization methods, and certain network architectures can help with gradient vanishing.

**แผนการสอน**

**หัวข้อ** ...............................................................

**ผู้สอน** ...............................................................

**เวลา** ...............................................................

**วัตถุประสงค์** เพื่อให้นิสิตสามารถ

1. ..........................................................................

2. ......................................................................... .

**เนื้อหา**

1.........................................................................

2.........................................................................

3.........................................................................

**การจัดประสบการณ์การเรียนรู้**

1. บอกวัตถุประสงค์ และบอกเนื้อหา นาที

2. สอนบรรยายเนื้อหาหัวข้อต่างๆ นาที

3. นิสิตซักถาม นาที

**สื่อการสอน**

1. เอกสารประกอบการสอน

2. power point ทั้งภาพนิ่งและ animation

**การประเมินผล**

ตัวอย่างเช่น ข้อสอบ Multiple choice question 5 ตัวเลือก

**หนังสืออ้างอิง**

1. ............................................................................................................

2 ............................................................................................................

3. ............................................................................................................

4 ............................................................................................................

**(หมายเหตุ 1 หัวข้อการสอน เขียน 1 แผนการสอน)**