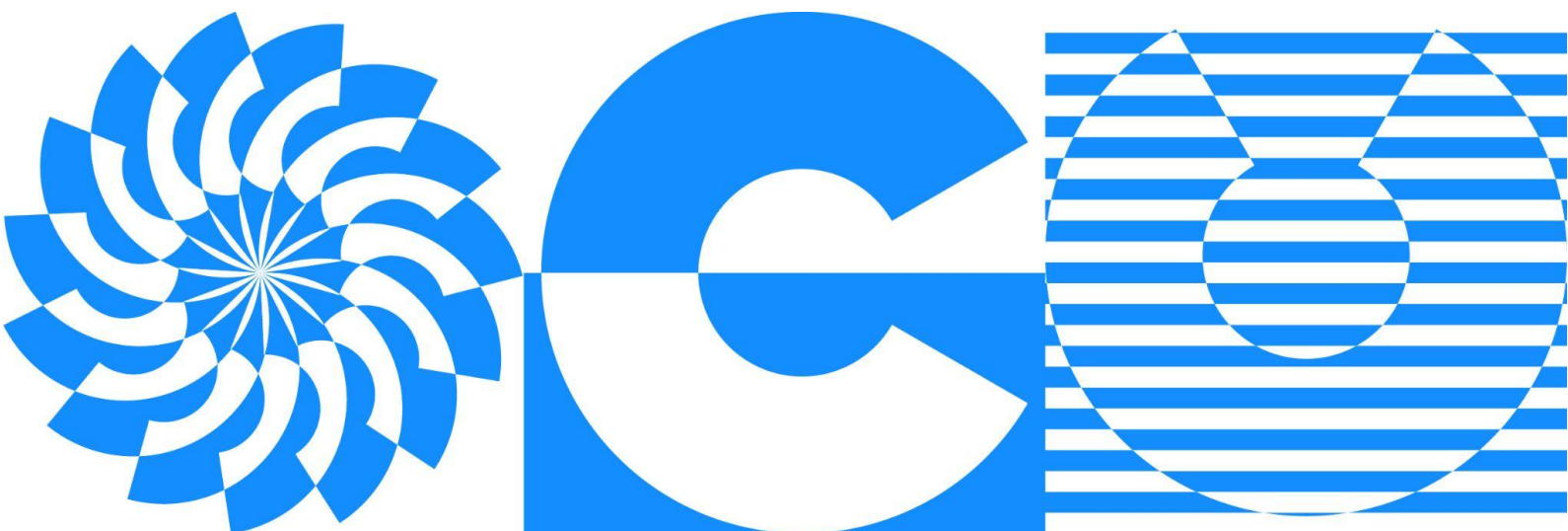


A place where legacy creates future.



Program

CVDL Master

[Detailed Curriculum](#)

Index

Course 1 – MOCV

Mastering OpenCV with Python

|

Course 2 – CVIP

Fundamentals of Computer Vision & Image Processing

|

Course 3 – DLPT

Deep Learning with PyTorch

|

Course 4 – DLTk

Deep Learning with TensorFlow & Keras

|

Course 5 – DLAP

Computer Vision and Deep Learning Applications

|

Course 6 – GENAI

Mastering Generative AI for Art

Mastering OpenCV with Python

1. **Getting Started with Images**

- 1.1. Image Basics (Basics of Image Formation and Image Formats)
- 1.2. Reading, Display and Writing Images in OpenCV
- 1.3. Color space conversion and different color spaces
- 1.4. Basic image manipulation (resizing, cropping, annotating, creating a Region of Interest)

2. **Basic Image Operations**

- 2.1. Mathematical operations on images (brightness and contrast)
- 2.2. Image thresholding, bitwise operations and masking
- 2.3. Image blending and the alpha channel
- 2.4. Creating Digital signatures using alpha blending

3. **Histograms and Color Segmentation**

- 3.1. Image histograms and enhancement using Histogram equalization
- 3.2. Color segmentation on images
- 3.3. Deforestation analysis using Color Segmentation
- 3.4. Satellite Imagery analysis using GeoTIFF Images

4. **Video Processing and Analysis**

- 4.1. Reading and Writing videos using OpenCV
- 4.2. Motion Detection analysis using Background Subtraction

5. **Contours and Shape Analysis**

- 5.1. Finding and Drawing Contours
- 5.2. Intruder detection using Contour Analysis

6. **Human Computer interaction(HCI) using PyAutoGUI**

- 6.1. Keyboard and Mouse Controls using PyAutoGUI
- 6.2. Playing Online Games using PyAutoGUI

7. **Building and Deploying Apps with Streamlit**

- 7.1. Building a Face detection application using streamlit

7.2. Deploying streamlit apps on Streamlit cloud

7.3. App Deployment on Heroku

8. **Image Filtering and Enhancement**

8.1. Image Filtering using Convolution Operations

8.2. Image Blurring and sharpening using convolutions

8.3. Edge Detection using Sobel Filters and Canny algorithm

8.4. Artistic Renderings using Image Filters

9. **Hough Transforms**

9.1. Detecting Lines using Hough Lines

9.2. Lane detection using using Hough Transforms from videos

10. **Image Restoration Techniques**

10.1. Noise Reduction using Median and Bilateral Filters

10.2. Image Inpainting for Image Restoration

10.3. Building a streamlit application on image restoration using Inpainting.

11. **Image Registration Techniques**

11.1. Affine Transforms and Homography

11.2. Image Alignment using Homography and Feature Matching

11.3. Building Virtual billboards and Creating Panoramas

12. **Augmented Reality**

12.1. Overview of ArUco markers

12.2. Application: AR using ArUco Markers

13. **Deep Learning using OpenCV**

13.1. Introduction to OpenCv's DNN Module.

13.2. Image classification using OpenCV DNN Module

13.3. Web application for Image Classification

14. **Face and Landmark Detection**

14.1. Face Detection using DNN Module

14.2. Face Blurring

14.3. Facial Landmarks Detection

- 14.4. Building a Real-time Blink Detection application
 - 15. **Object Detection**
 - 15.1. Object Detection using MobileNet SSD, YOLOv4 and YOLOv5
 - 15.2. Building a Social Distance Monitoring Application
 - 16. **Object Tracking**
 - 16.1. Introduction on Object Tracking Models in OpenCV
 - 16.2. Comparison across multiple tracking models.
 - 17. **Human Pose Estimation**
 - 17.1. Human Pose Estimation using MediaPipe
 - 17.2. Sports Analytics using MediaPipe
 - 18. **Human Segmentation**
 - 18.1. Creating selfie-style photos using Person Segmentation through MediaPipe
 - 19. **Text Detection and OCR**
 - 19.1. Text Detection using EAST and Differentiable Binarization (DB)
 - 19.2. OCR on Natural Images
 - 19.3. Language Translation using OCR
 - 20. **Super Resolution**
 - 20.1. Image Super Resolution techniques using OpenCV
 - 21. **Deploying Web Applications on Cloud Services**
 - 21.1. Building web applications using streamlit
 - 21.2. Deploying web applications using AWS, GCP, and Azure
-

Fundamentals of Computer Vision & Image Processing

Module 1

Getting Started with OpenCV

|

Module 2

Video IO and GUI

|

Module 3

Binary Image Processing

|

Module 4

Image Enhancement and Filtering

|

Module 5

Advanced Image Processing and Computational Photography

|

Module 6

Geometric Transforms and Image Features

|

Module 7

Image Segmentation and Recognition

|

Module 8

Video Analysis and Object Tracking

|

Module 9

Deep Learning with OpenCV

1 Getting Started with OpenCV

1.1 Introduction to Computer Vision

- 1.1.1 Image Processing VS Computer Vision
- 1.1.2 Problems in Computer Vision

1.2 Introduction to images

- 1.2.1 How are images formed?
- 1.2.2 Digital image
- 1.2.3 Image as a matrix
- 1.2.4 Manipulating pixels
- 1.2.5 Displaying and saving an image
- 1.2.6 Display utility functions
- 1.2.7 Color image
- 1.2.8 Image channels
- 1.2.9 Splitting and merging channels
- 1.2.10 Manipulating color pixels
- 1.2.11 Images with Alpha channel

1.3 Basic image operations

- 1.3.1 How to create new images?
- 1.3.2 Cropping an image section
- 1.3.3 Copying a region to another in an image
- 1.3.4 Resizing an image
- 1.3.5 Creating an image mask

1.4 Mathematical operations on images

- 1.4.1 Datatype conversion
- 1.4.2 Contrast enhancement
- 1.4.3 Brightness enhancement

1.5 Sunglass filter: A simple application

- 1.5.1 Load images
- 1.5.2 Use naive replacement
- 1.5.3 Use arithmetic operations

1.6 Bitwise operations

- 1.6.1 Different bitwise operations

1.7 Image annotations

- 1.7.1 Draw a line over an image
- 1.7.2 Draw a circle over an image
- 1.7.3 Draw a rectangle over an image
- 1.7.4 Draw an ellipse over an image
- 1.7.5 Draw text over an image

| | |
|---------------------|---------------------------------|
| Assignment 1 | Build a QR Code Detector |
|---------------------|---------------------------------|

2 Video IO & GUI

2.1 Video IO using HighGUI

- 2.1.1 Video IO jargon
- 2.1.2 Read and display video
- 2.1.3 Properties of video capture
- 2.1.4 How to write a video

2.2 Callback Functions

- 2.2.1 What are Callback Functions?

2.3 Keyboard as input device

- 2.3.1 How to take input from keyboard

| | |
|---------------------|---|
| Assignment 2 | Perform Image Annotation Using Mouse |
| Assignment 3 | Enhance GUI features with Sliders |

3 Binary Image Processing

3.1 Thresholding

- 3.1.1 What is Thresholding?
- 3.1.2 Thresholding in OpenCV

3.2 Erosion / Dilation

- 3.2.1 Overview on Erosion and Dilation
- 3.2.2 Erosion and Dilation in OpenCV

3.3 Opening and Closing

- 3.3.1 Overview on Opening and Closing
- 3.3.2 Opening and Closing on OpenCV

3.4 Connected Component Analysis

- 3.4.1 What is Connected Component Analysis?
- 3.4.2 Connected Component Analysis in OpenCV

3.5 Contour Analysis

- 3.5.1 What are Contours?
- 3.5.2 Contour Analysis in OpenCV

3.6 Blob detection

- 3.6.1 Blob detection in OpenCV

| | |
|---------------------|--|
| Assignment 4 | Implement Different Morphological Operations |
| Assignment 5 | Build a Coin Detection application using Contours |

4 Image Enhancement & Filtering

4.1 Color spaces

- 4.1.1 RGB color space
- 4.1.2 HSV color space
- 4.1.3 Other color spaces
- 4.1.4 Application: Finding dominant color in an image
- 4.1.5 Application: Desaturation Filter

4.2 Color transforms

- 4.2.1 Histogram Equalization
- 4.2.2 Advanced Histogram Equalization (CLAHE)
- 4.2.3 Color adjustment using curves

4.3 Image filtering

- 4.3.1 Introduction to image filtering
- 4.3.2 What is Convolution?
- 4.3.3 Convolution in OpenCV

4.4 Image smoothing

- 4.4.1 Box Blur
- 4.4.2 Gaussian Blur
- 4.4.3 Median Blur
- 4.4.4 Median Blur in OpenCV
- 4.4.5 Bilateral filtering
- 4.4.6 Bilateral Blur in OpenCV
- 4.4.7 Comparison: Median VS Bilateral

4.5 Image gradients

- 4.5.1 Introduction to image gradients
- 4.5.2 First Order Derivative Filters
- 4.5.3 Why is smoothing important before gradient?
- 4.5.4 Second Order Derivative Filters
- 4.5.5 Application: Sharpening Filter
- 4.5.6 Canny Edge Detection
- 4.5.7 Canny Edge Detection in OpenCV

| | |
|---------------------|--|
| Assignment 6 | Convert Your Images Into Different Color Spaces |
| Assignment 7 | Implement Camera Autofocus using Image Analysis |

5 Advanced Image Processing & Computational Photography

5.1 Hough transforms

- 5.1.1 What is a hough transform?
- 5.1.2 HoughLine: How to detect a line in an image?
- 5.1.3 HoughCircle: How to detect a circle in an image?

5.2 High Dynamic Range imaging

- 5.2.1 What is High Dynamic Range (HDR) imaging?
- 5.2.2 HDR in OpenCV

5.3 Seamless cloning

- 5.3.1 What is seamless cloning?
- 5.3.2 Seamless cloning in OpenCV
- 5.3.3 Application: Face Blending

5.4 Image inpainting

- 5.4.1 What is image inpainting?

| | |
|------------------|---|
| Project 1 | <ul style="list-style-type: none"> • Create Your Own Instagram Filter • Blemish Removal From Face • Chroma Keying |
|------------------|---|

6 Geometric Transforms & Image Features

6.1 Geometric transforms

- 6.1.1 Affine transform
- 6.1.2 Homography
- 6.1.3 Geometric transforms in OpenCV

6.2 Image features

- 6.2.1 Image feature: ORB
- 6.2.2 ORB feature in OpenCV

6.3 Feature matching

- 6.3.1 Different feature matching algorithms in OpenCV
- 6.3.2 RANSAC

6.4 Application: Image alignment

- 6.5 Application: Creating panoramas
- 6.6 Application: Finding known objects using OpenCV

| | |
|---------------------|---|
| Assignment 8 | Create Panorama from Multiple Images |
| Assignment 9 | Feature Matching-Based Image Alignment |
| Project 2 | Implement a Document Scanner Application |

7 Image Segmentation & Recognition

- 7.1 **Image segmentation using GrabCut**
 - 7.1.1 GrabCut theory
 - 7.1.2 GrabCut in OpenCV
- 7.2 **Introduction to AI**
 - 7.2.1 Basic overview of AI
- 7.3 **Image classification**
 - 7.3.1 Histogram of Oriented Gradients (HOG)
 - 7.3.2 Support Vector Machines (SVM)
 - 7.3.3 Eyeglass Classifier in OpenCV
- 7.4 **Object detection**
 - 7.4.1 Pedestrian detection in OpenCV
 - 7.4.2 Face detection using HAAR Cascade
 - 7.4.3 Face detection in OpenCV

| | |
|------------------|--|
| Project 3 | Create Your Own Selfie Application <ul style="list-style-type: none"> • Skin Smoothing Filter • Sunglass Filter |
|------------------|--|

8 Video Analysis

- 8.1 Motion estimation using Optical Flow
 - 8.1.1 What is Optical Flow?
 - 8.1.2 Lucas-Kanade Optical Flow
- 8.2 Application: Video stabilization
- 8.3 Object tracking
 - 8.3.1 Different object tracking algorithms
- 8.4 Different object trackers in OpenCV



- 8.4.1 Object tracking in OpenCV
- 8.4.2 Comparison of different trackers
- 8.5 Multiple object tracking
 - 8.5.1 How to track multiple objects in OpenCV
- 8.6 Kalman filtering
 - 8.6.1 Kalman Filter Tracker
- 8.7 MeanShift and CamShift
 - 8.7.1 Tracking using MeanShift and CamShift

| | |
|------------------|---|
| Project 4 | Implement fusion of Detection + Tracking |
|------------------|---|

9 Deep Learning with OpenCV

- 9.1 Image classification
 - 9.1.1 Image classification using Caffe and TensorFlow
 - 9.2 Object detection
 - 9.2.1 Single Shot Multibox Detector (SSD)
 - 9.2.2 You Only Look Once Detector (YOLO)
 - 9.3 Face detection
 - 9.3.1 SSD-Based Face Detector
 - 9.4 Human pose estimation
 - 9.4.1 OpenPose
-

Course 3 - DLPT

Deep Learning with PyTorch

Module 1

Getting Started with PyTorch

|

Module 2

Neural Networks

|

Module 3

Convolutional Neural Networks

|

Module 4

Deep Neural Networks

|

Module 5

Best Practices in Deep Learning

|

Module 6

Object Detection

|

Module 7

Single Stage Object Detectors

|

Module 8

Segmentation

|

Module 9

Pose Estimation

|

Module 10

Bonus Module: Generative Adversarial Networks (GANs)

1 Getting Started with PyTorch

1.1 Introduction to Artificial Intelligence

- 1.1.1 History of AI
- 1.1.2 Applications of AI
- 1.1.3 AI in Computer Vision
- 1.1.4 AI terminology
- 1.1.5 Why is Deep Learning so popular?

1.2 NumPy refresher

1.3 Introduction to PyTorch

- 1.3.1 Why use PyTorch?
- 1.3.2 Overview of PyTorch
- 1.3.3 PyTorch basics

1.4 What is inside an ML algorithm?

- 1.4.1 Machine Learning pipeline
- 1.4.2 Solving ML problems
- 1.4.3 Gradient Descent
- 1.4.4 Gradient Descent for Optimization

| | |
|---------------------|---|
| Assignment 1 | Implement ReLu, Softmax & Neuron Using PyTorch |
| Assignment 2 | Implement Gradient Descent For Two Variables |

2 Neural Networks

2.1 Feature Vectors 1-D to N-D

- 2.1.1 Feature Vectors & Normalization

2.2 Basics of Neural Networks

- 2.2.1 What is a Neural Network?
- 2.2.2 Loss Functions for Regression
- 2.2.3 Loss Functions for Classifications
- 2.2.4 Types of Activation Functions
- 2.2.5 How does the network learn?
- 2.2.6 Demystifying Neural Networks

2.3 Binary Classification using Perceptrons

2.4 PyTorch NN module

- 2.4.1 Introduction to PyTorch NN Module
- 2.4.2 MLP using Functional API
- 2.4.3 MLP using Sequential API

2.5 Image Classification using Multilayer Perceptron

- 2.5.1 MLP Classifier for Handwritten Digits (MNIST)

3 Convolutional Neural Networks

3.1 Convolution operation

- 3.1.1 What is a Convolution operation?
- 3.1.2 CNN building blocks
- 3.1.3 Layers in CNN

3.2 How to implement LeNet using PyTorch?

- 3.2.1 How to implement LeNet?
- 3.2.2 Implementing LeNet using PyTorch
- 3.2.3 LeNet with Batch Normalization
- 3.2.4 Effects of Batch Normalization

3.3 Evaluation of Classification Performance

- 3.3.1 Performance metrics for Classification
- 3.3.2 How to implement Classification metrics?

3.4 Introduction to Torchvision

- 3.4.1 Torchvision overview
- 3.4.2 Datasets
- 3.4.3 What are the different Transforms used to train a network?
- 3.4.4 Different models in Torchvision

3.5 Important CNN architectures

- 3.5.1 Different CNN architectures
- 3.5.2 Pre-trained models in Torchvision
- 3.5.3 Pre-trained Classification models in Torchvision

4 Deep Neural Networks

4.1 Optimization

- 4.1.1 What are Optimizers?
- 4.1.2 Learning Rate Decay methods
- 4.1.3 LR Scheduler

4.2 Training Deep Neural Networks

- 4.2.1 Step 1: Data understanding
- 4.2.2 Step 2: Data preparation
- 4.2.3 Step 3: Check training pipeline
- 4.2.4 Step 4: Train the model
- 4.2.5 Step 5: Improve the model

4.2.6 Check training pipeline

4.3 How to add robustness to a model?

4.3.1 Bias-variance tradeoff

4.3.2 How to prevent Overfitting?

4.3.3 Training with Regularization

4.4 Data loader with ImageFolder

4.4.1 How to load custom datasets in PyTorch?

| | |
|---------------------|-------------------------------------|
| Assignment 5 | Implement the Adam Optimizer |
|---------------------|-------------------------------------|

| | |
|------------------|---|
| Project 1 | Implement An Image Classifier From Scratch |
|------------------|---|

5 Best Practices in Deep Learning

5.1 Troubleshooting training with TensorBoard

5.1.1 TensorBoard overview

5.1.2 TensorBoard dashboard

5.1.3 Logging using TensorBoard

5.1.4 Sharing TensorBoard logs

5.2 Leverage pre-trained models

5.2.1 CNN architectures (Recap)

5.2.2 Fine-tuning and Transfer Learning

5.2.3 Fine-tuning using ResNet

5.3 How to structure your project for scale?

5.3.1 Introduction to py_modules package

5.3.2 Motivation of Trainer Pipeline

5.3.3 Hands-on Trainer Pipeline

5.4 Introduction to PyTorch Lightning

5.4.1 PyTorch Lightning overview

5.4.2 Inference on Production (ONNX)

5.4.3 Transfer Learning with Lightning

| | |
|------------------|---|
| Project 2 | Kaggle Competition On Image Classification |
|------------------|---|

6 Object Detection

6.1 Overview of Object Detection

6.1.1 Introduction to Object Detection

6.2 Evaluation metrics

- 6.2.1 Evaluation metrics for Object Detection
- 6.2.2 Compute evaluation metrics
- 6.3 Traditional algorithms in Object Detection**
 - 6.3.1 Different traditional algorithms
 - 6.3.2 Implement Non-Maximum Suppression
- 6.4 Two-Stage Object Detectors**
 - 6.4.1 Introduction to Two-Stage Object Detectors
 - 6.4.2 Faster RCNN using Torchvision
 - 6.4.3 Understanding faster RCNN
 - 6.4.4 Fine-tuning of faster RCNN
 - 6.4.5 Fine-tuning training of faster RCNN

7 Single Stage Object Detectors

- 7.1 You Only Look Once (YOLO)**
 - 7.1.1 Introduction to YOLO
- 7.2 Single Stage Multibox Detector (SSD)**
 - 7.2.1 Introduction to SSD
 - 7.2.2 SSD with PyTorch Hub
- 7.3 RetinaNet**
 - 7.3.1 Introduction to RetinaNet
 - 7.3.2 RetinaNet with Detectron2
- 7.4 How to create a custom Single State Multibox Detector (SSD)?**
 - 7.4.1 Detector NN architecture
 - 7.4.2 Generating Anchor Boxes
 - 7.4.3 Matching predictions with ground truth
 - 7.4.4 Loss Function
 - 7.4.5 Experiment

| | |
|---------------------|---|
| Assignment 6 | Implementing Focal Loss for Object Detection |
|---------------------|---|

| | |
|------------------|---|
| Project 3 | Automatic Number Plate Detection |
|------------------|---|

8 Segmentation

- 8.1 Semantic Segmentation architecture**
 - 8.1.1 Types of Semantic Segmentation architectures
 - 8.1.2 Dilated Convolution
 - 8.1.3 Transposed Convolution
 - 8.1.4 Fully Convolutional Network (FCN)
 - 8.1.5 U-Net



- 8.1.6 SegNet
- 8.1.7 DeepLab
- 8.2 Evaluation metrics for Semantic Segmentation**
 - 8.2.1 Dice Coefficient metrics
- 8.3 LinkNet architecture**
 - 8.3.1 Introduction to LinkNet architecture
- 8.4 Soft-Dice Loss**
 - 8.4.1 Introduction to Soft-Dice Loss
- 8.5 FCN and DeepLab using Torchvision**
 - 8.5.1 FCN and DeepLabV3 using Torchvision
- 8.6 U-Net for MRI Abnormality Segmentation**
- 8.7 Train your model from scratch**
- 8.8 Instance Segmentation**
 - 8.8.1 Instance Segmentation using Mask RCNN

| | |
|---------------------|--|
| Assignment 7 | LinkNet Architecture With VGG16 |
|---------------------|--|

| | |
|------------------|--|
| Project 4 | Kaggle Competition On Semantic Segmentation |
|------------------|--|

9 Pose Estimation

- 9.1 DensePose**
 - 9.1.1 Introduction to DensePose
 - 9.1.2 DensePose inference
 - 9.1.3 DensePose training
- 9.2 Create your own gym trainer**
 - 9.2.1 Automated Squat Counter

| | |
|------------------|-------------------------------------|
| Project 5 | Create An App Of Your Choice |
|------------------|-------------------------------------|

10 Generative Adversarial Networks (GANS)

- 10.1 GANs**
 - 10.1.1 Introduction to GANs
 - 10.1.2 Vanilla GAN using Fashion MNIST
 - 10.1.3 DCGAN using Flickr Faces
 - 10.1.4 CGAN using Fashion MNIST

Course 4 - DLTk

Deep Learning with TensorFlow & Keras

Module 1

Getting Started with Tensorflow and Keras

|

Module 2

Neural Networks

|

Module 3

Convolutional Neural Networks

|

Module 4

Best Practices In Deep Learning

|

Module 5

Object Detection

|

Module 6

Image Segmentation

|

Module 7

Pose Estimation

|

Module 8

Bonus: Generative Adversarial Networks (GANs)

1 Getting Started with TensorFlow and Keras

- 1.1 Introduction to Artificial Intelligence
- 1.2 AI in Computer Vision
- 1.3 AI terminology
- 1.4 NumPy refresher
- 1.5 Introduction to TensorFlow & Keras
- 1.6 What is inside an ML algorithm?
- 1.7 Gradient-based Optimization techniques in ML

| | |
|---------------------|---|
| Assignment 1 | Implement Leaky ReLu, Softmax & Convolution Using TensorFlow |
| Assignment 2 | Implement Gradient Descent For Two Variables |

2 Neural Networks

- 2.1 Feature Vectors & Normalization
- 2.2 Basics of Neural Networks
- 2.3 Understanding Backpropagation in a Neural Network
- 2.4 Loss Functions for Regression and Classification Tasks.
- 2.5 Mathematical foundation for Regression & Classification
- 2.6 Implementing a single neuron to model Linear Regression
- 2.7 Binary Classification using Perceptrons
- 2.8 Image Classification using Multilayer Perceptron
- 2.9 Model Complexit, Generalizaation and handling Overfitting
- 2.10 Understand and implement the building blocks like the different Activation Functions, Loss Functions, Hidden Layers

| | |
|---------------------|--|
| Assignment 3 | Implement A Multi-Layer Perceptron For Apparel Classification |
|---------------------|--|

3 Convolutional Neural Networks

- 3.1 The Convolution operation
- 3.2 CNN building blocks and Layers
- 3.3 Evaluation metrics for Classification Performance
- 3.4 Important CNN architectures like VGGNet and ResNet
- 3.5 Leverage pre-trained models using Transfer Learning and Fine-tuning
- 3.6 Building custom models from scratch using your own data
- 3.7 Handling Overfitting through Data augmentations

| | |
|--------------|---|
| Assignment 4 | Conversion from Sequential API to Functional API. |
| Assignment 5 | Implement a CNN For Image Classification On CIFAR10/100 Dataset |

| | |
|-----------|--|
| Project 1 | Implement An Image Classifier From Scratch |
|-----------|--|

4 Best Practices in Deep Learning

- 4.1 Advanced Optimizers in Keras
- 4.2 Training Deep Neural Networks
- 4.3 Handling Data in TensorFlow using TF Data, Sequence Class, and TF Records
- 4.4 Learning Rate Schedulers
- 4.5 Model Interpretability using GradCAM

| | |
|--------------|---------------------------|
| Assignment 6 | Implement AdamW Optimizer |
|--------------|---------------------------|

| | |
|-----------|---|
| Project 2 | Kaggle Competition For Sports Item Classification |
|-----------|---|

5 Semantic Segmentation

- 5.1 Problem formulation, custom layers and Loss Functions associated with Segmentation, like **Dilated & Transposed Convolution, Dice Loss, Pixel Accuracy**, etc.
- 5.2 Semantic Segmentation Datasets and Architectures
- 5.3 Evaluation metrics for Semantic Segmentation
- 5.4 Custom DataLoader for Segmentation Datasets
- 5.5 Fully Convolutional Networks (**FCNs**) and **UNet** training for Binary Segmentation on **Road Data**
- 5.6 UNet and **DeepLab** v3+ Training on Multi-class segmentation on **CamVid** Data.
- 5.7 Improving **DeepLab** v3+ on Segmentation of **Underwater Imagery (SUIM)**

| | |
|--------------|--------------------------|
| Assignment 7 | Implementation of PSPNet |
|--------------|--------------------------|

| | |
|-----------|---|
| Project 3 | Kaggle Competition On Semantic Segmentation Of FloodNet Dataset |
|-----------|---|

6 Object Detection

- 6.1 Evolution from traditional Object Detection algorithms to state-of-the-art Deep Learning-based methods
- 6.2 Exploring Object detection Datasets.
- 6.3 Revisiting Image Classification, and comparing it with Object Detection.
- 6.4 Two-Stage & Single Stage Object Detectors
- 6.5 Problem formulation, custom layers and Loss Functions used in Object Detection like Anchors, NMS, IoU, etc.
- 6.6 Learn about Object Detection models like RCNN, SSD, RetinaNet, YOLO, EfficientDet
- 6.7 Evaluation metrics for Object Detection: Precision, Recall, Average Precision(AP) and mean average Precision (mAP)
- 6.8 Using the TensorFlow Object Detection (TFOD) API
- 6.9 Fine-tuning of Object Detection Models available on TFOD API on a subset of Pascal VOC data.
- 6.10 Building a Custom SSD Model with FPN and training it on PenFudanPed Dataset

| | |
|---------------------|---|
| Assignment 8 | Encoding And Decoding Of Ground Truths For Anchor Box Implementation |
|---------------------|---|

| | |
|------------------|--|
| Project 4 | Build A Custom Object Detection Model For Detecting Safety Kits In Construction Sites |
|------------------|--|

7 Pose Estimation using MediaPipe

- 7.1 Real-time Posture analysis using MediaPipe Pose
- 7.2 Drowsy Driver Detection using MediaPipe

8 Generative Adversarial Networks (GANs)

- 8.1 Introduction to GANs
- 8.2 Vanilla GAN using Fashion MNIST
- 8.3 DCGAN using Flickr Faces
- 8.4 CGAN using Fashion MNIST

Computer Vision and Deep Learning Applications

Module 1

Facial Landmark Detection

|

Module 2

Applications of Facial Landmarks

|

Module 3

Snapchat Filters

|

Module 4

Face Recognition

|

Module 5

Introduction to Deep Learning with CNNs

|

Module 6

State-of-the-art Object Detection

|

Module 7

Text Detection And OCR

|

Module 8

Deploy Applications On Cloud

1 Facial Landmark Detection

- 1.1 Different face processing applications**
- 1.2 Facial Landmark Detection using Dlib**
 - 1.2.1 Introduction to Dlib
 - 1.2.2 Facial Landmark Detection using Dlib
- 1.3 Application: Face Alignment**
- 1.4 Improving Facial Landmark Detector**
 - 1.4.1 Improving speed
 - 1.4.2 Improving stability
 - 1.4.3 Landmark stabilization in OpenCV
 - 1.4.4 Optical Flow
 - 1.4.5 Lucas-Kanade Optical Flow
- 1.5 Facial Landmark Detection Theory**
 - 1.5.1 Machine Learning basics
 - 1.5.2 Research paper review
- 1.6 Train a custom Facial Landmark Detector**
 - 1.6.1 How to train a custom Landmark Detector?

2 Applications of Facial Landmarks

- 2.1 Alpha Blending**
 - 2.1.1 Alpha Blending in OpenCV
- 2.2 Image Warping**
 - 2.2.1 Affine Transform
 - 2.2.2 Geometric Transform
 - 2.2.3 Triangle Warping
- 2.3 Delaunay Triangulation**
 - 2.3.1 Theory of Delaunay Triangulation
- 2.4 Face Averaging**
 - 2.4.1 Face Averaging using OpenCV
- 2.5 Face Morphing**
 - 2.5.1 Face Morphing using OpenCV
- 2.6 Application: Bug Eyes**
 - 2.6.1 Create a Big Eye app
- 2.7 Head Pose Estimation**
 - 2.7.1 Head Pose Estimation in OpenCV

Assignment 1

Build an Automatic Smile Detection Application

3 Snapshot Filters

3.1 Face Swap

- 3.1.1 How can Face Swapping be achieved?
- 3.1.2 Seamless Cloning
- 3.1.3 Seamless Cloning in OpenCV
- 3.1.4 Face Swapping in a video

3.2 Application: Beard Filter

3.3 Application: Aging Filter

3.4 Non-linear Deformations

- 3.4.1 Moving Least Squares
- 3.4.2 Application: Happify and Fatify filters on face

| | |
|-----------|------------------------------------|
| Project 1 | Build a Virtual Makeup Application |
|-----------|------------------------------------|

4 Face Recognition

- 4.1 Introduction to Face Recognition
- 4.2 Eigen Faces
- 4.3 Fisher Faces
- 4.4 Local Binary Patterns Histograms
- 4.5 Face Recognition API in OpenCV
- 4.6 Deep Learning-based Face Recognition [To be updated with latest models]

| | |
|-----------|---|
| Project 2 | DoppelGanger: Find Celebrity Look-Alike |
|-----------|---|

5 Introduction to Deep Learning

5.1 Basics of Neural Networks

- 5.1.1 What is a Neural Network?
- 5.1.2 How do Neural Networks learn?

5.2 Introduction to Keras [To be updated to The PyTorch framework]

- 5.2.1 Deep Learning frameworks
- 5.2.2 The Keras Framework
- 5.2.3 Linear Regression using Keras

5.3 Convolutional Neural Network

- 5.3.1 What is a Convolutional Neural Network (CNN)?
- 5.3.2 Example: Image Classification using CNN
- 5.3.3 Data Augmentation (Python)

5.4 Transfer Learning and Fine-tuning and Logging

- 5.4.1 Weights and Biases Logging
- 5.4.2 Transfer Learning
- 5.4.3 Fine-Tuning

| | |
|---------------------|--|
| Assignment 2 | Improve CNN Performance during Training |
|---------------------|--|

6 Object Detection

- 6.1 Object Detection overview**
 - 6.1.1 What is Object Detection?
 - 6.1.2 Traditional Object Detection pipeline
- 6.2 YOLOv3 using DarkNet [To be updated to YOLOv8 using Ultralytics]**
 - 6.2.1 Inference using YOLOv3
 - 6.2.2 Fine-Tuning using YOLOv3
- 6.3 YOLO-NAS [Will be added as part of the course update]**
 - 6.3.1 Inference using YOLO-NAS
 - 6.3.2 Fine-Tuning using YOLO-NAS
- 6.4 RT-DETR [Will be added as part of the course update]**
 - 6.4.1 Inference using RT-DETR
 - 6.4.2 Fine-Tuning using RT-DETR

| | |
|------------------|--|
| Project 3 | Train A Face Mask Detector using YOLO |
|------------------|--|

7 Text Detection & OCR

- 7.1 Overview of OCR**
 - 7.1.1 What is OCR?
 - 7.1.2 The OCR pipeline
 - 7.1.3 Challenges
 - 7.1.4 Datasets and competitions
- 7.2 Graphic Text Recognition using Tesseract**
 - 7.2.1 What is Tesseract?
 - 7.2.2 Introduction to OCR using Tesseract
 - 7.2.3 Tesseract OCR failure cases
 - 7.2.4 Improving Tesseract OCR failures
- 7.3 Text Detection**
 - 7.3.1 Text Recognition using EAST
 - 7.3.2 Text Recognition using CRAFT (Python)
- 7.4 Modified pipeline for scene Text Recognition using Tesseract (Python)**



7.5 Inference using Transformer OCR (TrOCR)

7.6 Fine-Tuning Transformer OCR (TrOCR)

7.7 Case Study - ALPR (YOLO NAS + TrOCR)

| | |
|---------------------|--|
| Assignment 3 | Build an app to perform OCR On Invoices |
|---------------------|--|

8 Deploy Applications on Cloud

8.1 Create a web application using Flask

- 8.1.1 What is Flask?
- 8.1.2 A minimal Flask application
- 8.1.3 Using HTML templates
- 8.1.4 A complete Flask application

8.2 Deploy a web application on Heroku [Paas]

- 8.2.1 What is Heroku?
- 8.2.2 How to create an account on Heroku?
- 8.2.3 Prepare application for deployment
- 8.2.4 How to deploy using Heroku CLI?
- 8.2.5 How to deploy using Heroku website?

8.3 Deploy a web application on Google GCP [Iaas]

- 8.3.1 What is Google Cloud Platform (GCP)?
- 8.3.2 Create account on Google Cloud Platform
- 8.3.3 Create and configure a VM instance
- 8.3.4 Setup VM and deploy app
- 8.3.5 Change firewall settings and check deployment

| | |
|---------------------|--------------------------------------|
| Assignment 4 | Deploy Your Web App On Heroku |
|---------------------|--------------------------------------|

[Explore Other Courses](#)