**CV Libraries / Tools**

* OpenCV -Image/video processing, basic CV tasks (Reading/writing images, filtering, edge detection, transformations)
* RoboFlow -Dataset management, augmentation, labeling (Object detection dataset creation, exporting formats)
* PyTorch / TensorFlow -Deep learning framework for CV (CNNs, Transformers, GANs)
* VLMs (Vision-Language Models) -Models that connect vision & text (CLIP, BLIP, OpenCLIP)
* Pillow / scikit-image -Image manipulation and processing (Resize, crop, rotate, filters)

**Image Preprocessing**

* Resizing / Cropping -Adjust image size or focus area (Resize to model input, random crop)
* Normalization / Scaling -Standardize pixel values (Divide by 255, mean-std normalization)
* Color Space Conversion -Transform images (RGB → Gray, RGB → HSV)
* Noise Removal / Filtering -Improve image quality (Gaussian blur, median filter)
* Histogram Equalization -Enhance contrast (CLAHE -Contrast Limited Adaptive HE)

**Data Augmentation**

* Geometric -Transformations that preserve labels (Flip, rotation, scaling, translation)
* Color / Style -Change visual appearance (Brightness, contrast, hue, saturation jitter)
* Synthetic Data -Generate artificial data (GAN-based augmentation, CutMix, MixUp)
* Advanced CV Augmentation -Bounding-box aware transformations (Mosaic, Random Erasing)

**Core CV Tasks**

* Image Classification -Label whole image (CNNs, Vision Transformers)
* Object Detection -Detect & localize objects (YOLOv5/YOLOv8, Faster R-CNN, SSD, RetinaNet)
* Semantic Segmentation -Pixel-wise classification (U-Net, DeepLabv3, SegFormer)
* Instance Segmentation -Detect & segment individual objects (Mask R-CNN, Detectron2)
* Image Captioning -Describe image in text (Encoder-Decoder, VLMs)
* Optical Flow / Tracking -Motion estimation & tracking (Lucas-Kanade, Farneback, DeepSort)
* Image Generation -Generate realistic images (GANs, Diffusion Models)
* Image Enhancement / Restoration -Improve or restore images (Super-resolution, Denoising, Inpainting)
* Few-Shot / One-Shot / Zero-Shot Learning -Learning with limited or no labeled examples (Zero-shot: CLIP, Flamingo; One-shot/Few-shot: Meta-learning, Prototypical Networks, BLIP, Prompting with VLMs)

**YOLO Notes / Tips**

* YOLO Specifics -Single-stage real-time object detector (YOLOv5 → PyTorch, YOLOv8 → Ultralytics, Tiny versions for edge, export to ONNX/TensorRT)

**Evaluation Metrics**

* Classification -Evaluate prediction accuracy (Accuracy, Top-k Accuracy, F1-score)
* Detection -Evaluate localization & classification (mAP -mean Average Precision, IoU)
* Segmentation -Evaluate pixel-wise prediction (IoU, Dice coefficient, Pixel Accuracy)
* Tracking -Evaluate multi-object tracking (MOTA, MOTP, ID Switches)

**Specialized Tools / Techniques**

* Pretrained Models -Use models trained on large datasets (ResNet, EfficientNet, CLIP, DINOv2)
* Vision-Language Alignment -Match images & text (CLIP, BLIP, Flamingo)
* Edge / Real-time Deployment -Optimized inference for devices (ONNX, TensorRT, OpenVINO, TFLite)
* Explainability -Understand model predictions (Grad-CAM, Saliency maps)

**Tips / Best Practices**

* Dataset Management -Balance classes, clean labels (RoboFlow, Data Augmentation, Stratified Split)
* Performance -Optimize training & inference (Mixed precision, batch size tuning, model pruning)
* Robustness -Handle real-world variations (Augmentations, adversarial training, domain adaptation)