# 📘 Computer Vision Terms (Starter Glossary)

A concise collection of key Computer Vision and Deep Learning concepts —

perfect for building a strong foundation in your "Zero to Hero" journey 👁️‍🗨️

---

### 🧩 Core Concepts

- \*\*Pixel:\*\* The smallest unit of a digital image.

- \*\*Channel:\*\* A component of color representation (e.g., R, G, B).

- \*\*Resolution:\*\* The number of pixels in an image (width × height).

- \*\*Feature:\*\* Information extracted from data to describe patterns or objects.

- \*\*Weight:\*\* A learnable parameter in a neural network that determines how much influence an input has on the output.

- \*\*Bias:\*\* An additional parameter that shifts the activation function to better fit data.

---

### 🔍 Classical Computer Vision

- \*\*Convolution:\*\* A mathematical operation used to apply filters on images.

- \*\*Edge Detection:\*\* Identifying boundaries in an image (e.g., Sobel, Canny).

- \*\*HOG (Histogram of Oriented Gradients):\*\* A descriptor for object detection.

- \*\*SIFT / SURF / ORB:\*\* Feature detection and description algorithms.

- \*\*L1 Distance (Manhattan Distance):\*\* Sum of absolute differences between vectors.

- \*\*L2 Distance (Euclidean Distance):\*\* Square root of the sum of squared differences between vectors.

---

### 🧠 Deep Learning

- \*\*CNN (Convolutional Neural Network):\*\* A neural network architecture for processing images.

- \*\*Pooling:\*\* Downsampling feature maps to reduce spatial dimensions (e.g., MaxPooling).

- \*\*Activation Function:\*\* Introduces non-linearity to neural networks (e.g., ReLU, Sigmoid, Tanh).

- \*\*Batch Normalization:\*\* Normalizes layer inputs to stabilize and accelerate training.

- \*\*Loss Function:\*\* A mathematical function that measures how far the model’s predictions are from the true values (e.g., Cross-Entropy, MSE).

- \*\*Regularization:\*\* Techniques that prevent overfitting by penalizing large weights (e.g., L1, L2, Dropout).

- \*\*Backpropagation:\*\* The algorithm that calculates how much each weight contributed to the error, allowing the model to update them.

- \*\*Gradient Descent:\*\* Optimization method that updates weights in the opposite direction of the gradient to minimize loss.

- \*\*Learning Rate:\*\* A hyperparameter that controls how big each weight update step should be.

| **Adım** | **Ne Oluyor** | **Ne Buluyor** |
| --- | --- | --- |
| 1️⃣ Forward Pass | Ağı şimdiki ağırlıklarla çalıştır | Tahmin (ŷ) ve Loss |
| 2️⃣ Backward Pass | Loss’u geri doğru türevle | Her ağırlık için gradyan (∂L/∂w) |
| 3️⃣ Update | Gradient descent uygula | Yeni ağırlıkları hesapla |