Fundamentals of Deep Learning Computer Vision (1) Discriminative Models

An-Che Liang

Data Science Study Group, November 2025

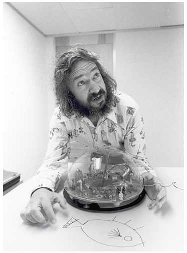
Acknowledgments

Portions of this material are adapted from the following course:

- NTU EEE5053: Computer Vision (Spring 2023)
- NTU CommE5052: Deep Learning for Computer Vision (Fall 2025)

What is Computer Vision?

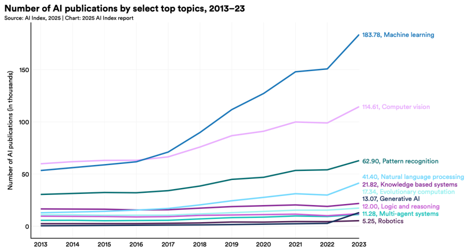
Computer Vision is a field of computer science that focuses on enabling computers to interpret and understand visual information in a manner similar to humans. The origins of this field can be traced back to a 1966 undergraduate summer research project at MIT, led by Seymour Papert, which aimed to develop computer programs capable of recognizing patterns in images.



Seymour Papert — Source: Wikipedia

What is Computer Vision? (Cont.)

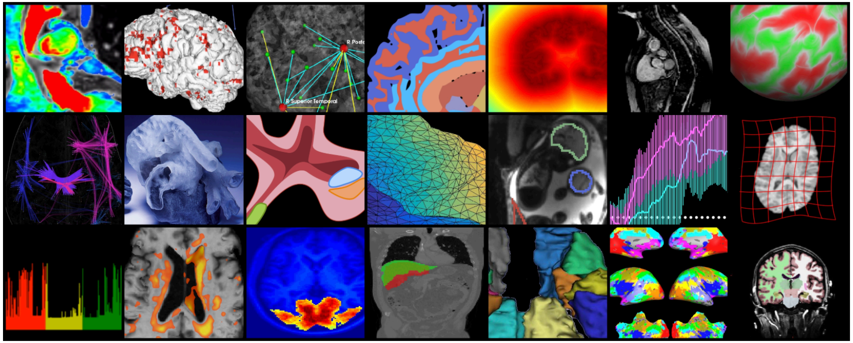
After decades of development, computer vision has become a prominent field within artificial intelligence and computer science. Common tasks in computer vision include image classification, object detection, image segmentation, image understanding, and image generation. In this lecture, we will focus on **deep learning methods** for computer vision.



Source: AI Index Report 2025

Computer Vision Applications

Applications of computer vision span a wide range of domains, including medical imaging, self-driving cars, surveillance, retail, agriculture, and many more.



Source: MIT CSAIL

Computer Vision Applications (Cont.)

Full Self-Driving (Supervised) — Source: Tesla

Computer Vision Applications (Cont.)

Figure 1

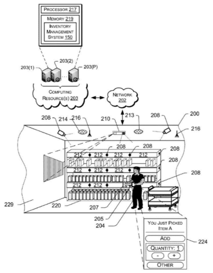
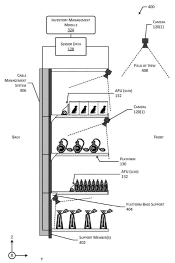
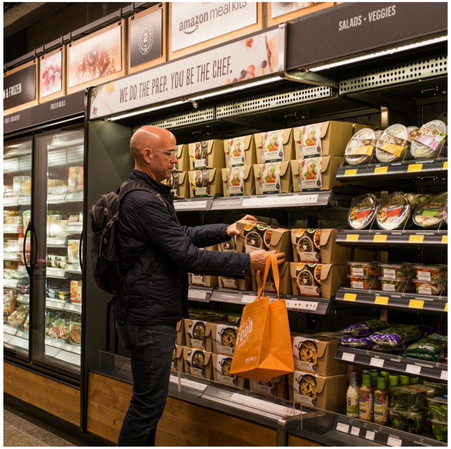


Figure 2





Amazon Go — Source: Harvard Digital Design Institute

Discriminative Models and Generative Models

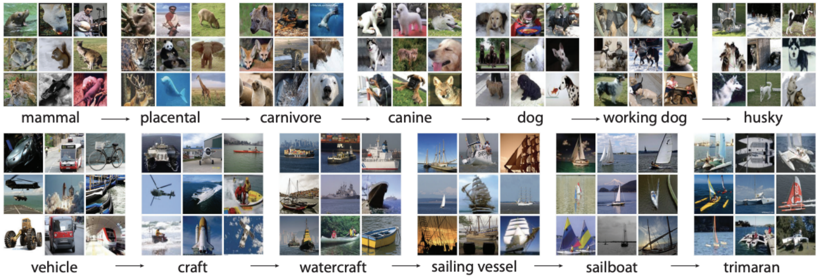
Discriminative Models focus on learning the decision boundary between classes. They model the conditional probability $p(y|x)$, directly predicting the label y given an image x. These models are widely used for tasks such as image classification, object detection, and semantic segmentation.

Generative Models aim to model how the data is generated. They learn the joint distribution $p(x, y)$ or the data distribution $p(x)$, enabling them to *generate* new images, reconstruct missing information, or model complex visual structures. Common applications include image synthesis, denoising, inpainting, and representation learning.

Let's begin by focusing on discriminative models.

ImageNet

ImageNet is a large-scale dataset containing 1,000 object categories and more than two million images. It was developed by Fei-Fei Li and her team at Princeton and was first released at CVPR 2009. Since 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been held annually.


mammal → placental → carnivore → canine → dog → working dog → husky
vehicle → craft → watercraft → sailing vessel → sailboat → trimaran

ImageNet — Source: roboflow

Convolution

Convolution is a mathematical operation that describes how one function responds to or interacts with another. In traditional signal processing, it characterizes how a filter modifies or reacts to an input signal.
For two continuous functions $f(t)$ and $g(t)$, the convolution is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau.$$

For discrete sequences $x[n]$ and $h[n]$, the convolution is given by:

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k].$$

Convolution on 2D Image

In the case of a 2D image, given an input image I and a convolution kernel K, the 2D convolution operation is defined as:

$$(I * K)(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(x - m,\ y - n)\, K(m, n).$$

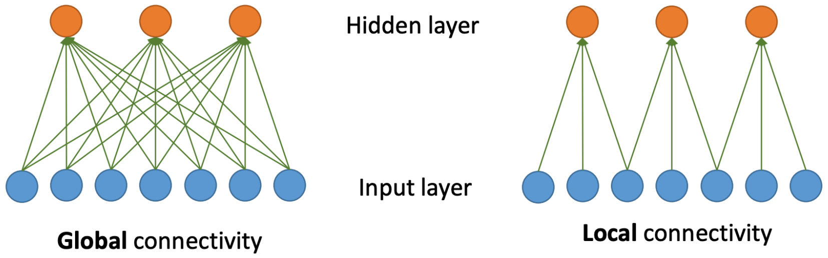
For finite image and kernel sizes, this is typically written as:

$$(I * K)(x, y) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} I(x - i,\ y - j)\, K(i, j),$$

where H and W denote the kernel height and width.

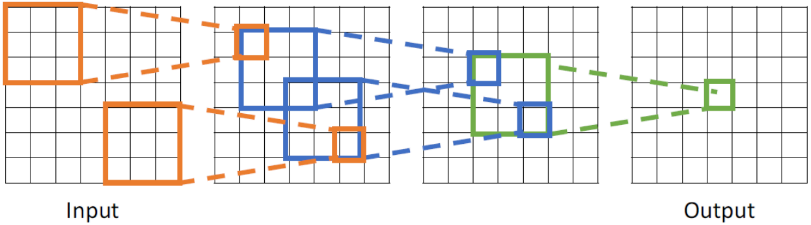
Convolutional Layer

Compared with a fully connected layer, a convolutional layer restricts each neuron to receive input only from a local neighborhood of pixels. This notion of *locality* is essential in image processing, as nearby pixels tend to be highly correlated and form meaningful visual patterns such as edges, textures, and shapes.



Convolutional Layer (Cont.)

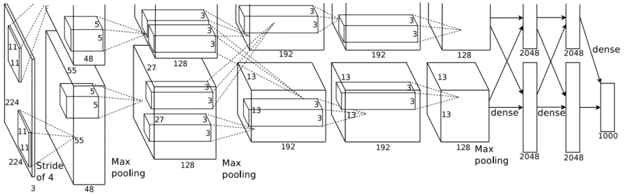
By stacking multiple convolutional layers, a model can combine low-level features (such as edges and corners) to form higher-level representations (such as objects or human faces). This hierarchical feature extraction is learned automatically, eliminating the need for manually engineered features.



AlexNet

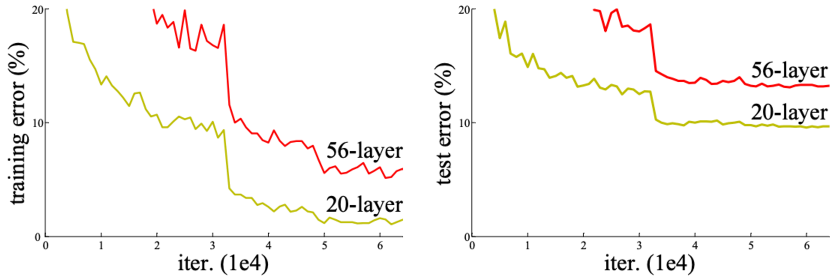
AlexNet is a 7-layer convolutional neural network with roughly 60 million trainable parameters, developed by Alex Krizhevsky, Ilya Sutskever (co-founder of OpenAI), and Geoffrey Hinton.

It achieved a top-5 error rate of 16%, dramatically outperforming the second-place result of 26%. By winning ILSVRC in 2012, AlexNet rekindled widespread interest in convolutional neural networks and marked a pivotal breakthrough in deep learning for computer vision.



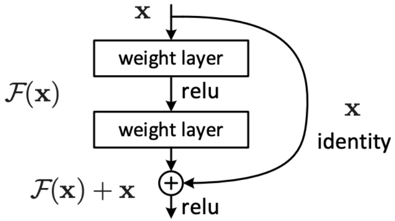
ResNet

In theory, stacking more layers allows a model to represent increasingly complex functions. However, in practice, very deep neural networks often suffer from issues such as exploding or vanishing gradients, which make training unstable or ineffective.



ResNet

ResNet is a family of deep convolutional neural networks that introduces residual blocks to address the previously mentioned training difficulties.



Instead of learning a full transformation, each residual block learns only the *residual* (the difference between the input and the desired output), which greatly stabilizes training and enables much deeper architectures.

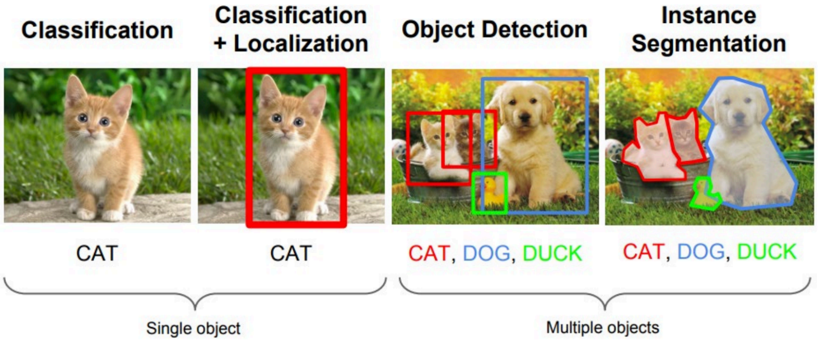
Lab 1: Image Classification

In Lab 1, you will use `torchvision.models.resnet18` to perform image classification. Follow the steps below:

1. Start a Google Colab instance with a T4 GPU (or use a local machine with a dedicated GPU). Verify that PyTorch is installed and CUDA is available.
2. Download the images here.
3. Load the pretrained ResNet-18 model weights.
4. Apply the appropriate image transformations to obtain the correct input dimensions.
5. Run inference on the images.
6. Output the top-5 predicted classes for each image. Class labels can be found here here
7. Check whether the predicted classes are correct.

From Image Classification to Object Detection

After learning how to classify images containing a single object, the next natural step is to determine *where* objects are located within an image.



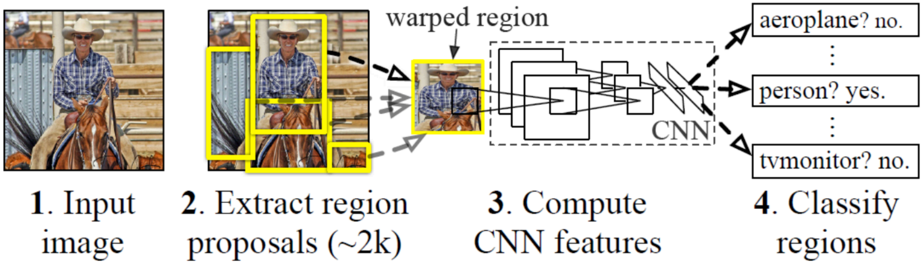
R-CNN

R-CNN (Region-based Convolutional Neural Network) is a two-stage object detection model. It first *(1) proposes candidate regions* that may contain objects, and then *(2) classifies each proposed region* to determine whether it contains an object and, if so, which class it belongs to.

In step (1), a preprocessing algorithm called *selective search* is used to generate region proposals. In step (2), each proposed region is passed individually through a CNN for feature extraction and classification. This pipeline is computationally expensive because the CNN must process every region separately (often around 2,000 regions per image).

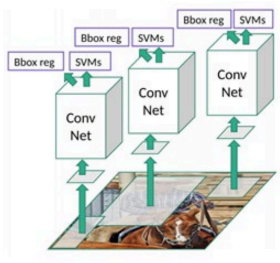
Subsequent models such as **Fast R-CNN** and **Faster R-CNN** significantly improved the efficiency of this approach by reducing redundant computation and introducing learnable region proposal mechanisms.

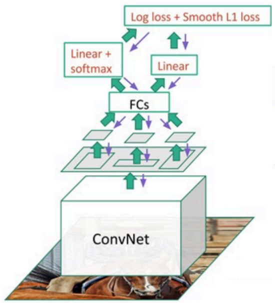
R-CNN (Cont.)

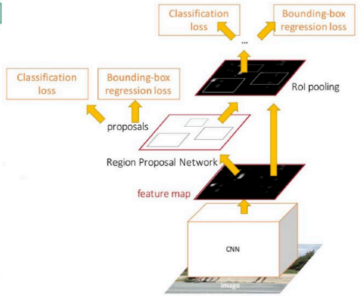


R-CNN Model Architecture

R-CNN (Cont.)



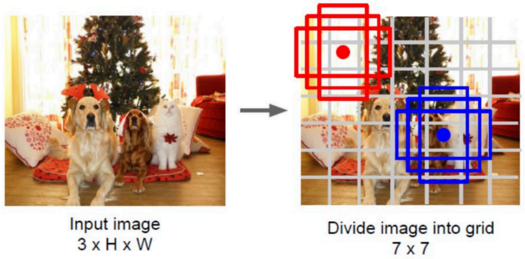




R-CNN, Fast R-CNN & Faster R-CNN

YOLO

YOLO (You Only Look Once) is a one-stage object detection model that divides an image into an $S \times S$ grid. For each grid cell, the model predicts B bounding boxes with associated confidence scores, along with C class probabilities. These outputs are combined into a single tensor of shape: $S \times S \times (B \cdot 5 + C)$, where each bounding box contributes 5 values (center x, center y, width w, height h, and confidence conf).

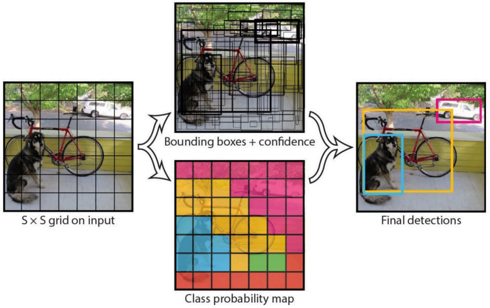


YOLO (Cont.)

The confidence score is computed as

$$\text{conf} = \Pr(\text{object}) \times \text{IoU} \times \Pr(\text{class} \mid \text{object}),$$

where IoU is the intersection-over-union between the predicted bounding box and the ground-truth box.



Lab 2: Object Detection

In Lab 2, you will use `ultralytics.YOLO` to perform image classification. Follow the steps below:

1. Start a Google Colab instance with a T4 GPU (or use a local machine with a dedicated GPU).
2. Download the images here.
3. Load the pretrained YOLOv11-nano model weights.
4. Run inference on the images, then annotate each detected object with its predicted class label and bounding box.
5. Check whether the predicted objects are correct.
6. (Optional) Use OpenCV to perform real-time object detection using the built-in camera on your laptop.

CLIP

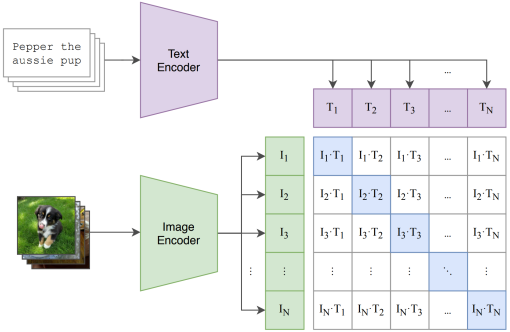
Image classification and object detection models have achieved near-human performance with low latency suitable for real-time applications. However, they still face two major limitations:

1. They rely on a fixed, predefined set of classes.
2. They require labor-intensive labeled datasets, making it difficult to train on internet-scale data.

OpenAI introduced **CLIP** (Contrastive Language–Image Pre-training) in 2021 to address these challenges by learning directly from large collections of image–text pairs.

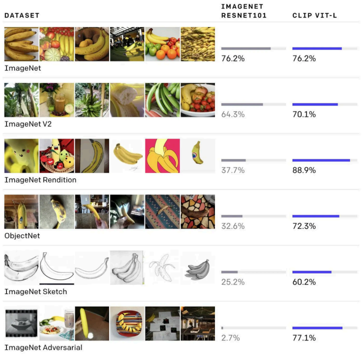
CLIP (Cont.)

During training, CLIP is given a batch of N image–text pairs and is optimized to identify the correct associations among the $N \times N$ possible pairings. The image encoder and text encoder are trained jointly to *maximize* the cosine similarity of the N correct pairs while *minimizing* the similarity of the remaining $N(N-1)$ incorrect pairings.


Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ⋮ IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
IN·T1 IN·T2 IN·T3 ... IN·TN

CLIP (Cont.)

After being trained on 400 million image–text pairs, CLIP demonstrates strong zero-shot capabilities—similar to GPT-2 and GPT-3. It can accurately classify images and recognize categories it has never been explicitly trained on.

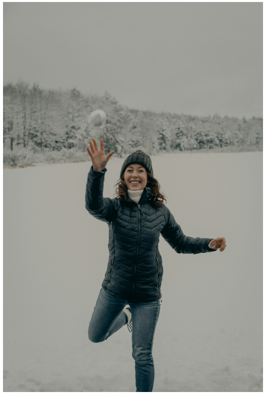


DATASET	IMAGENET RESNET101	CLIP VIT-L
ImageNet	76.2%	76.2%
ImageNet V2	64.3%	70.1%
ImageNet Rendition	37.7%	88.9%
ObjectNet	32.6%	72.3%
ImageNet Sketch	25.2%	60.2%
ImageNet Adversarial	2.7%	77.1%

Lab 3: Open Word Image Classification

In Lab3, you will use Google's SigLIP 2 model with the `transformers` library to perform open word image classification.

1. Start a Google Colab instance with a T4 GPU (or use a local machine with a dedicated GPU).
2. Load the pretrained `google/siglip2-so400m-patch14-384` model weights.
3. Run inference on the image using the following class sets:
   - [spring, summer, autumn, winter]
   - [human, cat, dog, horse]
   - [happy, sad, angry, scared]
4. Check whether the predicted classes are correct.



Remarks

There are many important concepts we did not cover in this lecture. Here are several suggested readings for further exploration:

1. SegNet: A Deep Convolutional Encoder–Decoder Architecture for Image Segmentation
2. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
3. BLIP: Bootstrapping Language–Image Pre-training for Unified Vision–Language Understanding and Generation
4. Visual Instruction Tuning

And many more!