TELECOM Paris

IP PARIS

Deep Learning for Computer Vision

From pixels to structured scene understanding

Jhony H. Giraldo (Enseignant–Chercheur)
jhony.giraldo@telecom-paris.fr
13 November 2026

Institut Mines-Télécom

Is computer vision relevant? Why are you enrolled?

Is computer vision relevant? Why are you enrolled?

- Yes! Nowadays, there are many successful applications of computer vision:

Healthcare

Detect disease earlier; quantify anatomy; support clinicians.

Mobility

Understand roads, autonomous driving, etc.

Industry

Inspect defects, monitor processes, and enable flexible automation.

Robotics

Perceive geometry, grasp objects, and act safely in the world.

Sport & media

Track players, recover pose, index video, augment live content.

Earth observation

Map land use, measure change, and respond to hazards at scale.

Is computer vision relevant? Why are you enrolled?

- Yes! Nowadays, there are many successful applications of computer vision:

Healthcare

Detect disease earlier; quantify anatomy; support clinicians.

Mobility

Understand roads, autonomous driving, etc.

Industry

Inspect defects, monitor processes, and enable flexible automation.

Robotics

Perceive geometry, grasp objects, and act safely in the world.

Sport & media

Track players, recover pose, index video, augment live content.

Earth observation

Map land use, measure change, and respond to hazards at scale.

- There's a huge industry in computer vision. There are also many researchers working in academia.

Example of applications

Image and Video Generation



Prompt: Generate an image of my students and me in the deep learning for computer vision course on the moon.

Example of applications

Autonomous Vehicles and Transport



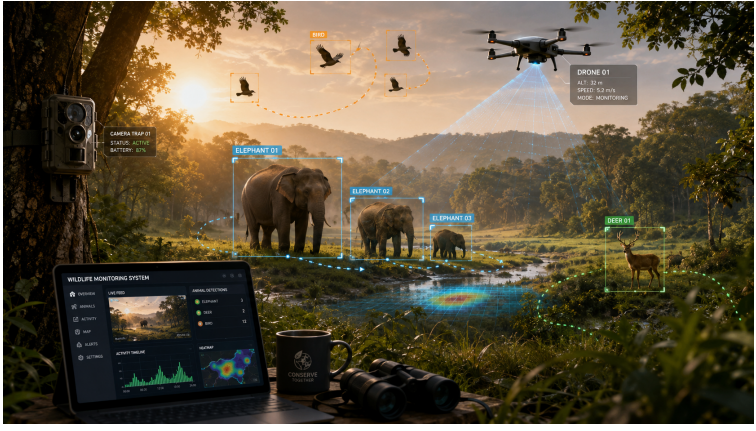
Example of applications

Medical Imaging



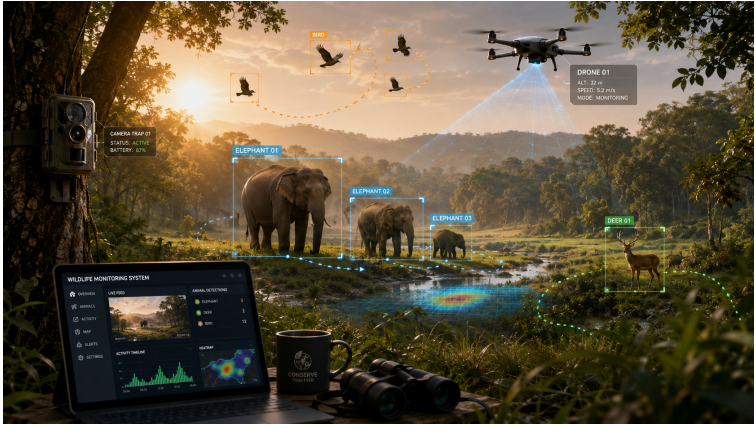
Example of applications

Environmental Applications



Example of applications

Environmental Applications



- There are more applications!

Course objectives

Applications

Choose, train, evaluate, and diagnose modern vision systems.

- Match an output representation to a task.
- Build reproducible PyTorch pipelines.
- Interpret metrics and failure modes.

Fundamentals

Understand why architectures, losses, data, and optimization choices work.

- Connect inductive bias to data efficiency.
- Reason about geometry and scale.
- Read papers beyond their headline result.

Course objectives

Applications

Choose, train, evaluate, and diagnose modern vision systems.

- Match an output representation to a task.
- Build reproducible PyTorch pipelines.
- Interpret metrics and failure modes.

- CV is not just deep learning.
- Deep learning is not plug-and-play.

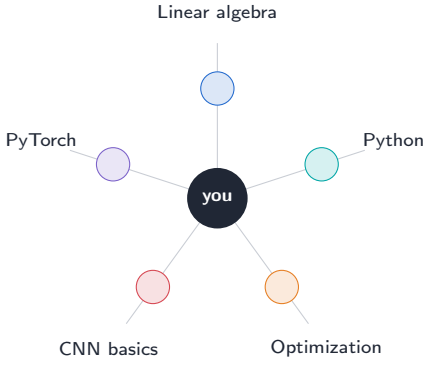
Fundamentals

Understand why architectures, losses, data, and optimization choices work.

- Connect inductive bias to data efficiency.
- Reason about geometry and scale.
- Read papers beyond their headline result.

Prerequisites

- **Linear algebra**: vectors, matrices, tensors, dot products
- **Optimization**: gradients, SGD, regularization
- **Deep learning**: CNN basics, losses, backpropagation
- **Practice**: Python and basic PyTorch workflows



- **We will refresh some concepts needed today**.

Course calendar

Date	Type	Focus	Responsible
13 Nov 2026	CM	CV recap; CNNs/ViTs; detection, segmentation, pose estimation, tracking	Jhony H. Giraldo
20 Nov 2026	TP	From detection to multi-object tracking	Jhony H. Giraldo
27 Nov 2026	CM	Foundation models; contrastive and vision–language learning	Stephan Alaniz
04 Dec 2026	TP	Foundation-models	Jhony H. Giraldo, Stephan Alaniz
11 Dec 2026	CM	Generative foundation models and language models	Stephan Alaniz
18 Dec 2026	CM	Deep learning for 3D: meshes and point clouds	Jhony H. Giraldo
08 Jan 2027	TP	3D deep learning	Jhony H. Giraldo
15 Jan 2027	Eval.	Oral paper discussion / course evaluation	Course team

All sessions are scheduled in the afternoon. Some students will be evaluated on the morning of January 15. CM = cours magistral; TP = travail pratique.

Evaluation

Continuous assessment — 50%

- Two *one-hour written tests*, held during the first half of:
  - *TP 2, December 04*: CM 1 + TP 1 + CM 2.
  - *TP 3, January 08*: TP 2 + CM 3 + CM 4.
- Tests cover both **lecture material and practical sessions**.

Final project and oral exam — 50%

- Work in groups of two on an approved recent research paper.
- Present the problem, method, main results, and limitations.
- Formulate a testable hypothesis inspired by the paper or one of its limitations.
- Keep the hypothesis **feasible** given your time and computational resources.
- Test the hypothesis experimentally and provide evidence for your conclusions.
- Submit a working and reproducible notebook before the final presentation.
- Answer individual questions during the final presentation.

Use of LLMs/agents: You may use them for the final project, but you remain responsible for all presented material and must be able to explain and defend every choice. **We strongly recommend developing the research hypothesis yourselves.**

Expectations and policies

- Limit the use of cell phones during lectures for non-academic purposes.
- Participation in the lectures is important.
- Academic integrity: plagiarism, cheating, and misuse of LLMs will be penalized. You should verify every claim and run every submitted result.
- Examples of misuse of LLMs:
  - Submit/present something you cannot explain.
  - Use invented references, data, or measurements.

You remain responsible for every line, claim, figure, and number you submit.

Resources

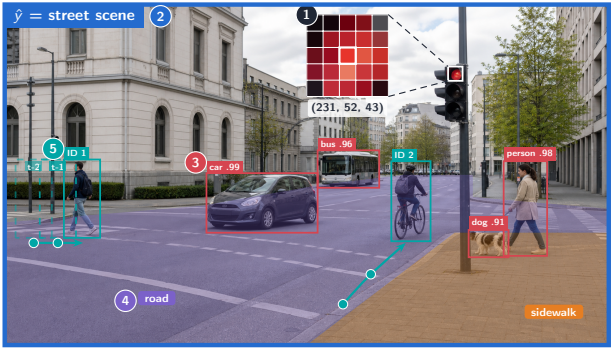
Concepts

- Stanford CS231n: cs231n.stanford.edu
- TUM: Videos
- Goodfellow, Bengio, Courville: Deep Learning

Code and libraries

- torchvision
- Detectron2

Today: beyond a single label



1 pixels $x \in [0, 255]^{H \times W \times 3}$
raw measurements, no meaning

2 class $y \in \{1, \ldots, C\}$
what is the scene? (no *where*)

3 boxes $\{(b_k, c_k, s_k)\}_{k=1}^{K}$
what + where, for every object

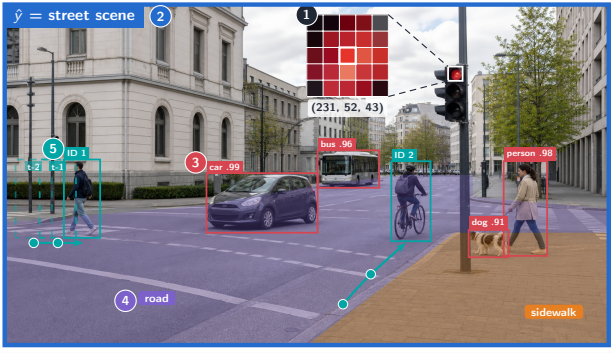
4 masks $M \in \{1, \ldots, C\}^{H \times W}$
which class at every pixel?

5 tracks $\{(t, b_t^i, \mathrm{ID}_i)\}$
who is who, over time?

more structure

From 2 to 5: increasingly **structured** outputs—and costlier supervision.

Today: beyond a single label



1 pixels $x \in [0, 255]^{H \times W \times 3}$
raw measurements, no meaning

2 class $y \in \{1, \ldots, C\}$
what is the scene? (no *where*)

3 boxes $\{(b_k, c_k, s_k)\}_{k=1}^{K}$
what + where, for every object

4 masks $M \in \{1, \ldots, C\}^{H \times W}$
which class at every pixel?

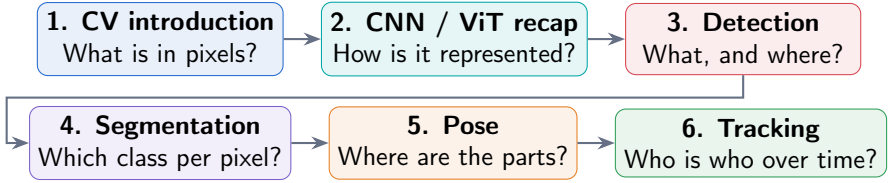
5 tracks $\{(t, b_t^i, \mathrm{ID}_i)\}$
who is who, over time?

more structure

From 2 to 5: increasingly **structured** outputs—and costlier supervision.

- There are too many engineering tricks in computer vision to make things work. I cannot cover all of them in this lecture, but you should be aware of them in practice.

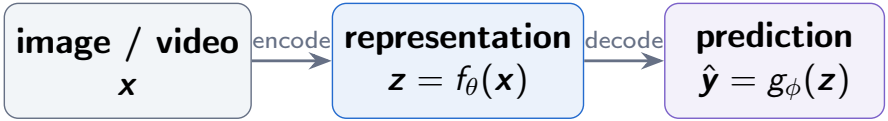
Outline



- The backbone is reusable; the representation, head, loss, metric, and inference must match the task.

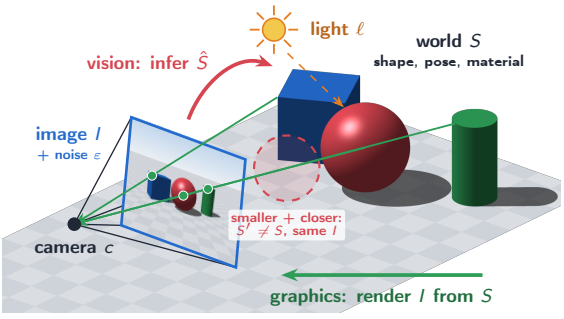
Computer vision recap

From pixels to decisions



- $f_\theta(\cdot)$ is typically a neural network (convolution-based, attention-based, etc.), parametrized by the parameters θ; while $g_\phi(\cdot)$ is another neural network, typically called the prediction head, parametrized by the parameters ϕ.
- **Learning question:** What representation makes the target predictable from finite data?

Computer vision is an inverse problem



Ill-posed

Many worlds can produce nearly the same image:

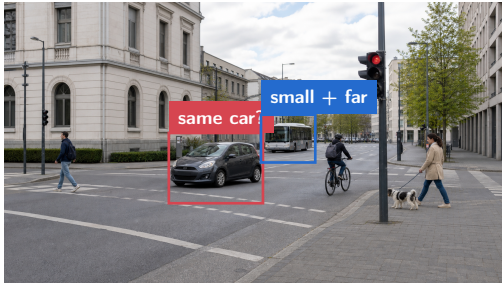
$$I = R(S, \ell, c) + \varepsilon,$$

with lighting ℓ, camera c, and noise or unmodeled effects ε.

Notation. I: observed image
S: 3D world (shape, pose, material)
R: image formation (rendering)
$\hat{S}$: world inferred by vision
S': a different world

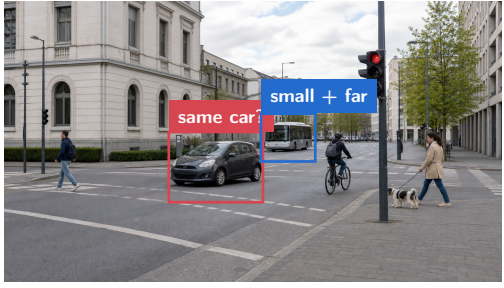
Vision succeeds by combining data with priors.

Why is computer vision hard?



- **Projection**: depth collapses onto the image plane.
- **Appearance**: lighting, weather, texture, sensor noise.
- **Geometry**: viewpoint, deformation, articulation, scale.
- **Visibility**: occlusion, clutter.
- **Semantics**: context and ambiguous categories.
- **Open world**: rare and unseen cases matter.

Why is computer vision hard?



- **Projection**: depth collapses onto the image plane.
- **Appearance**: lighting, weather, texture, sensor noise.
- **Geometry**: viewpoint, deformation, articulation, scale.
- **Visibility**: occlusion, clutter.
- **Semantics**: context and ambiguous categories.
- **Open world**: rare and unseen cases matter.

- Useful invariance must coexist with sensitivity to location, shape, and identity.

Computer-vision task taxonomy

Task	Output unit	Central question
Classification	image	What is present?
Localization	one box + class	Where is the main object?
Detection	set of boxes	What objects, and where?
Semantic segmentation	pixel	Which class per pixel?
Instance segmentation	object mask	Which pixels belong together?
Pose estimation	keypoint / part	Where are articulated parts?
Tracking	trajectory + ID	Which observation is the same entity?

The output space determines annotation cost, architecture, loss, inference, and metric.

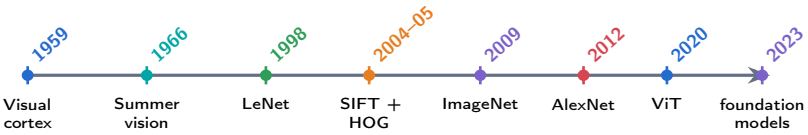
One image, progressively richer annotations



image label | boxes + classes | dense class map | keypoints + limbs | identities over time

more spatial structure, more annotation, more ways to be wrong

A brief history on computer vision



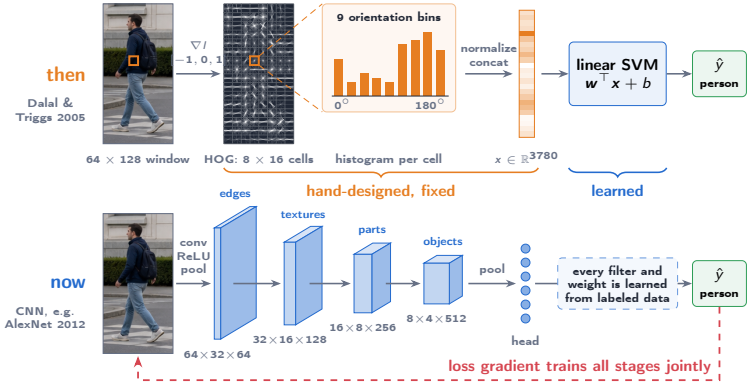
hand-designed features
geometry + invariance

supervised deep learning
data + accelerators

pretraining at scale
transferable representations

Milestones: Hubel & Wiesel (1959/1962); MIT Summer Vision Project (1966); LeCun et al. (1998); Lowe (2004); Dalal & Triggs (2005); Deng et al. (2009); Krizhevsky et al. (2012); Dosovitskiy et al. (2021).

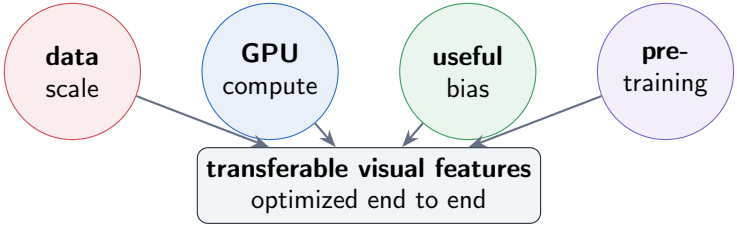
From handcrafted features to learned representations



Notation. I: input image ∇I: image gradient (filter $[-1, 0, 1]$)
$\boldsymbol{x}$: HOG descriptor (3780 numbers) $\boldsymbol{w}, b$: learned SVM weights, bias $\hat{y}$: predicted label

- End-to-end learning replaces hand-designed interfaces, but not the need for good assumptions or evaluation.

Why deep learning changed computer vision



- Convolution encoded locality; modern attention adds adaptive global interaction.
- Large labeled datasets made comparisons credible; large unlabeled datasets widened pretraining.
- Better optimization, normalization, and residual connections made much deeper networks trainable.

Reminder: supervised learning setup

$$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}, \qquad \hat{\boldsymbol{y}}_i = f_\theta(\boldsymbol{x}_i)$$

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i) + \lambda\, \Omega(\theta).$$

Notation. $\boldsymbol{x}_i$: input image i $\boldsymbol{y}_i$: its target (label)
N: number of training examples f_θ: the model
θ: its parameters (weights) $\hat{\boldsymbol{y}}_i$: the prediction
$\mathcal{L}$: loss: how wrong one prediction is
$\Omega(\theta)$: regularizer $\lambda \geq 0$: regularization strength
θ^*: the learned parameters

The loss is optimized on training data; choices are made on validation data; the test set estimates final generalization.

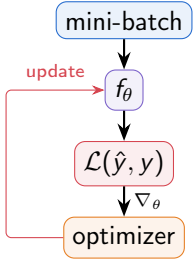


Image classification



Notation. $\boldsymbol{x}$: input image
f_θ: network with weights θ
C: number of classes $\quad \boldsymbol{s}$: class scores (logits)
$\boldsymbol{s}_c$: score of class $c \quad k$: index over all classes
$p(y = c \mid \boldsymbol{x})$: predicted probability of class c
y: true class $\quad \mathcal{L}_{\mathrm{CE}}$: cross-entropy loss

$$\boldsymbol{s} = f_\theta(\boldsymbol{x}) \in \mathbb{R}^C,$$

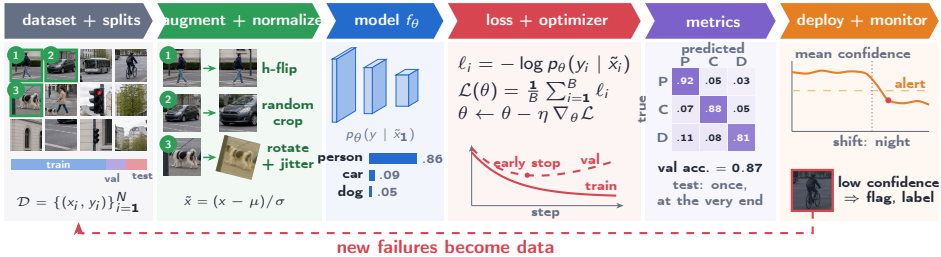
$$p(y = c \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{s}_c}}{\sum_{k=1}^{C} e^{\boldsymbol{s}_k}}.$$

class	probability
street scene	**0.84**
parking lot	0.09
bus station	0.05
other	0.02

Cross-entropy rewards probability on the ground-truth class:

$$\mathcal{L}_{\mathrm{CE}} = -\log p(y \mid \boldsymbol{x}).$$

A machine-learning pipeline for images



Notation. $\mathcal{D}$: dataset of N examples x_i, y_i: image i, its label μ, σ: per-channel mean, std
$\tilde{x}$: normalized image $p_\theta(y \mid \tilde{x})$: predicted class probabilities θ: model weights
ℓ_i: loss of example i B: batch size $\mathcal{L}(\theta)$: batch loss η: learning rate $\nabla_\theta \mathcal{L}$: gradient w.r.t. θ

Before training

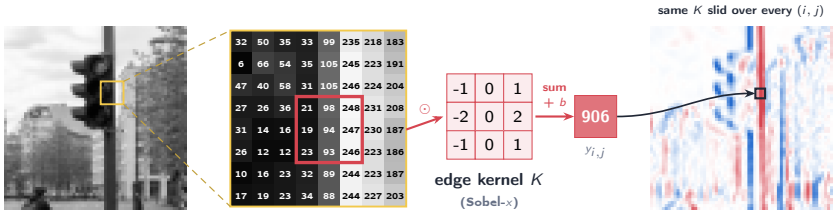
Define the target, split, inspect labels, and establish a simple baseline.

Common leakage

The test/validation data are not truly independent of the training data, so the measured performance looks better than the model's real ability to generalize

Recap: Convolutional Neural Networks

Convolution intuition: a learned local pattern detector



$$y_{i,j} = b + \sum_{u,v,c} K_{u,v,c}\, x_{i+u,j+v,c}$$

red window ($b = 0$):

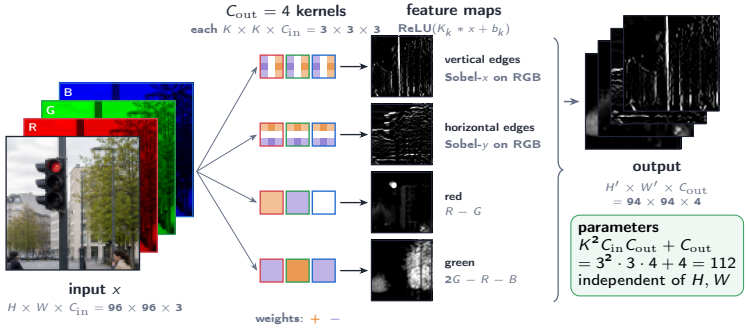
$$-21 + 248 - 2 \cdot 19 + 2 \cdot 247 - 23 + 246 = 906$$

- Same weights at every location.
- High response = local pattern match.
- Training learns K; sliding maps where the pattern occurs.

Notation. x: input image y: output feature map K: kernel weights b: bias
(i, j): output row, column (u, v): offset inside the kernel c: input channel (1 here: gray)

- Convolution is a local linear operator with shared parameters.

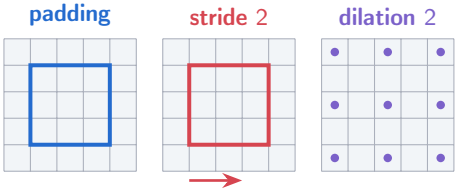
Convolution layer: channels, kernels, feature maps



Notation. x: input image H, W: input height, width H', W': output height, width
$C_{\text{in}}, C_{\text{out}}$: in/out channels K: kernel side K_k, b_k: weights, bias of kernel k $*$: convolution

- Each output channel mixes *all* input channels but only a local spatial neighborhood.
- A 1×1 convolution changes channels without enlarging the spatial receptive field.

Stride, padding, and dilation



For one spatial dimension,

$$H' = \left\lfloor \frac{H + 2P - D(K-1) - 1}{S} + 1 \right\rfloor.$$

Notation. H: input size (one dimension)
H': output size K: kernel size
P: zero padding per side $\lfloor \cdot \rfloor$: round down

- S (stride): sampling step / downsampling
- P: border context and output size
- D (dilation): spacing inside the kernel

These hyperparameters determine resolution, compute, and visible context.

Pooling: invariance versus spatial precision

Max pool 2×2

1	7	2	0
3	4	1	6
2	5	8	1
0	3	2	4

→

7	6
5	8

Average pool 2×2

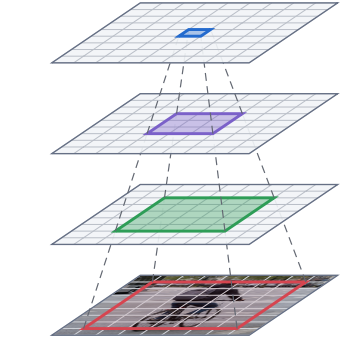
1	7	2	0
3	4	1	6
2	5	8	1
0	3	2	4

→

3.75	2.25
2.50	3.75

- Pooling summarizes a neighborhood and reduces resolution and compute.
- It increases tolerance to small shifts, but exact boundaries and keypoints become harder to recover.
- Modern networks often downsample with a strided convolution instead.

Receptive field: how much context can a feature use?



each step up: 3×3 conv, stride 1

Notation. l: layer index ($l = 0$: input)
k_l: kernel size of layer l s_l: its stride d_l: its dilation
j_l: input pixels between neighbouring units of layer l
r_l: receptive-field size of one unit (input pixels)

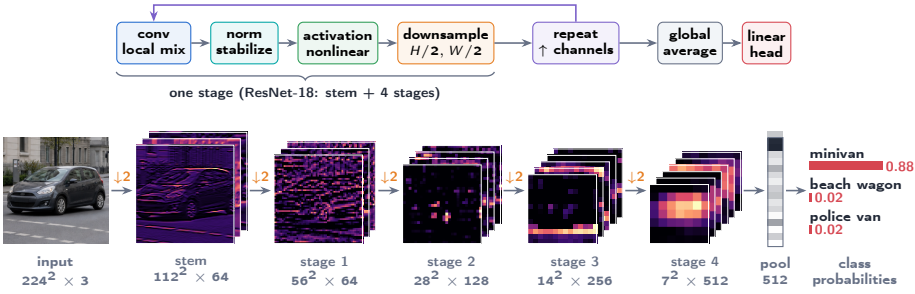
Track the sampling jump j_l and receptive field r_l:

$$j_l = j_{l-1}\, s_l,$$
$$r_l = r_{l-1} + (k_l - 1)\, d_l\, j_{l-1}.$$

Left: $k_l = 3$, $s_l = d_l = 1 \Rightarrow r_l = 1 + 2l$.
With stride $s_2 = 2$: $j_2 = 2$, so
$r_3 = 5 + 2 \cdot 2 = 9$.

- Depth, stride, and dilation expand context.
- Dense prediction needs both broad context and high-resolution detail.

The recurring CNN architecture pattern



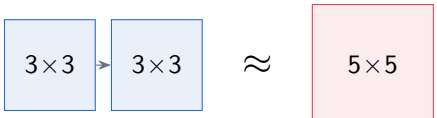
Spatial trade
H, W decrease while receptive field and semantic abstraction increase.

Channel trade
More feature channels preserve diverse evidence as spatial resolution shrinks.

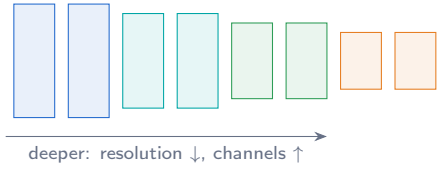
Notation. H, W: feature-map height, width $112^2 \times 64$: $H \times W \times$ channels $\downarrow 2$: halve H and W

- A classifier discards spatial detail at the end; dense tasks tap intermediate maps.

VGG: small filters, deep and simple



Notation. C: channels in and out of each layer
$18C^2$: $2 \times (3 \cdot 3 \cdot C \cdot C)$ weights



- Uniform 3×3 convolutions.
- 16 or 19 weight layers.
- Heavy classifier: VGG-16 has about 138M parameters.

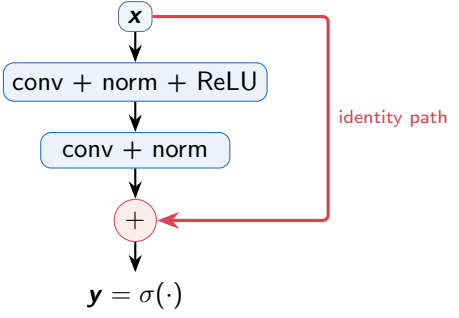
Its clean design became a template, even though later networks are more compute- and parameter-efficient.

Simonyan & Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR, 2015.

ResNet: learn a residual correction

- Intuition: Deeper models were not better than shallow models before ResNet. Deeper models are supersets of shallow models, so they should be at least as good. $\rightarrow$ We have a harder optimization problem.
- We make it easy for the model to learn the identity.

$$\boldsymbol{y} = \sigma(\mathcal{F}(\boldsymbol{x}; \theta) + \boldsymbol{x}) .$$



Notation. $\boldsymbol{x}$: block input $\boldsymbol{y}$: block output
$\mathcal{F}$: residual branch (the conv layers)
θ: their weights σ: activation (ReLU)

- Identity paths carry signal and gradients.
- Very deep networks become trainable and reusable.

He, Zhang, Ren & Sun, "Deep Residual Learning for Image Recognition," CVPR, 2016.

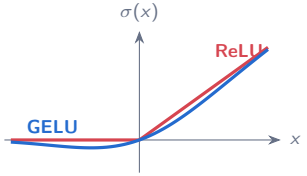
Normalization and activations

Batch normalization (per channel)

$$\hat{x} = \frac{x - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \qquad y = \gamma \hat{x} + \beta.$$

Notation. $\mathcal{B}$: current mini-batch
x: one activation of the channel
$\mu_{\mathcal{B}}, \sigma_{\mathcal{B}}^2$: its mean, variance over $\mathcal{B}$
ϵ: tiny constant (avoids division by 0)
$\hat{x}$: normalized activation
γ, β: learned scale, shift y: output

It improves optimization and permits larger learning rates, but behavior depends on batch statistics and train/eval mode.

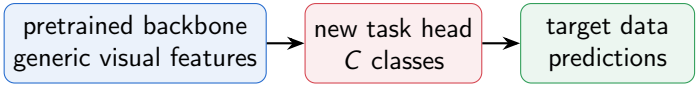


Notation. x: input to the activation
$\sigma(x)$: its output

- ReLU: cheap, sparse, piecewise linear.
- GELU: smooth gating, common in transformers.
- LayerNorm normalizes within a token/example and does not need batch statistics.

Ioffe & Szegedy, "Batch Normalization," ICML, 2015. Hendrycks & Gimpel, "Gaussian Error Linear Units," 2016.

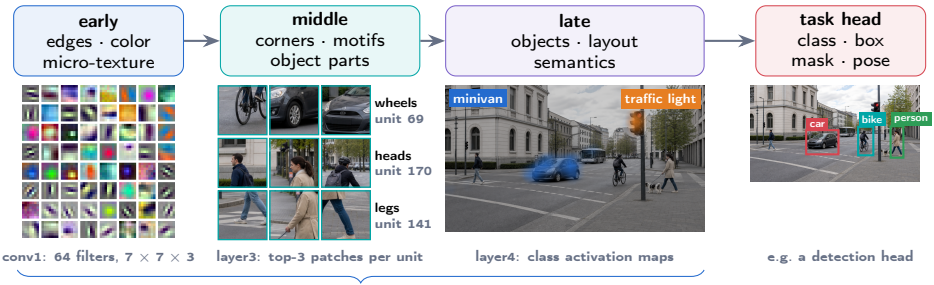
Transfer learning: reuse a pretrained backbone



Strategy	Trainable weights	Best when	Main risk
Linear probe	head only	little data; diagnose representation	under-adaptation
Partial fine-tune	head + late blocks	moderate data / related domain	difficult layer choice
Full fine-tune	all weights	enough data and compute	overfit / forgetting

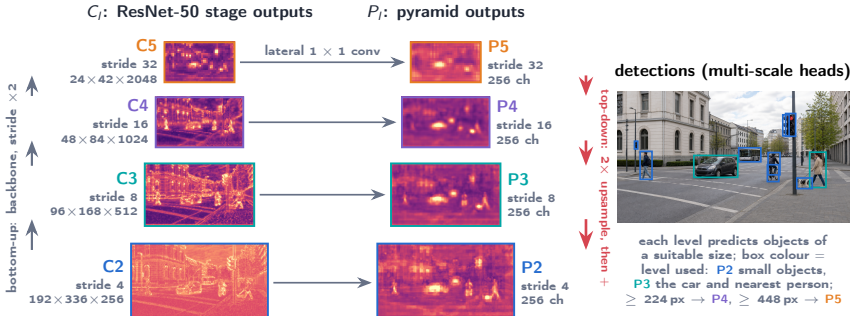
Always adapt preprocessing and normalization to the exact pretrained checkpoint.

A hierarchy of visual features



- Classification can use the final semantic vector.
- Detection and segmentation combine late semantics with earlier spatial detail.
- "Features" are task- and training-dependent, not a universal visual vocabulary.

Feature pyramids: a preview of multi-scale prediction



C_l: backbone output after stage l (stride 2^l). P_l: same stride, 256 channels, enriched top-down:

$$P_5 = \mathrm{conv}_{1\times1}(C_5), \qquad P_l = \mathrm{conv}_{3\times3}\big(\mathrm{conv}_{1\times1}(C_l) + \mathrm{up}_{2\times}(P_{l+1})\big)$$

Notation. l: stage / pyramid level (2–5) $\mathrm{conv}_{k\times k}$: convolution with a $k \times k$ kernel
$\mathrm{up}_{2\times}$: upsample 2× (double H, W) $H \times W \times C$: map height, width, channels

Lin et al., "Feature Pyramid Networks for Object Detection," CVPR, 2017.

CNN strengths: useful inductive bias

Translation equivariance

If T_δ shifts the image, convolution shifts the feature map:

$$K * (T_\delta x) = T_\delta (K * x).$$

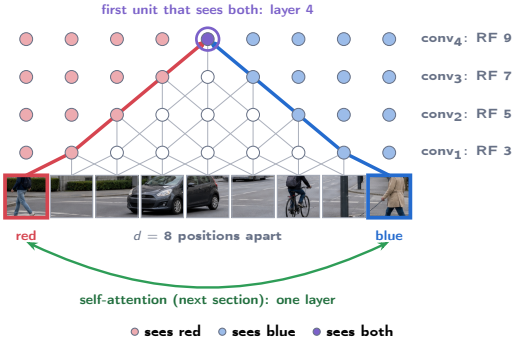
Notation. x: input image
K: convolution kernel $*$: convolution
δ: shift amount (pixels)

Equivariance preserves *where*; pooling can later create partial invariance.

- **Locality**: nearby pixels interact first.
- **Parameter sharing**: the same detector works across locations.
- **Hierarchical composition**: local evidence builds into parts and objects.
- **Data efficiency**: built-in assumptions constrain the hypothesis space.
- **Multi-scale maps**: intermediate layers naturally support dense tasks.

- A good inductive bias says which solutions should be easy to learn.

CNN limitations: global context is indirect



$k \times k$ convs: layers needed $\approx d/(k-1)$, here $8/2 = 4$

Notation. k: kernel size
d: distance between the two inputs
conv_l: conv layer l RF: receptive field (inputs seen)

- Long-range interaction needs depth, large kernels, dilation, or pooling.
- Repeated downsampling discards fine spatial detail.
- Fixed local mixing does not adapt its connectivity to the image.
- Strong locality can favor texture or shortcuts over global shape.

Important: modern CNNs can be very strong, and transformers introduce their own costs.

Recap: Vision Transformers

Attention intuition: content-dependent interaction

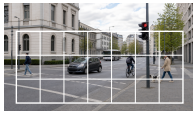


image → 7 × 3 patches
each patch becomes a token
(shown separated below)



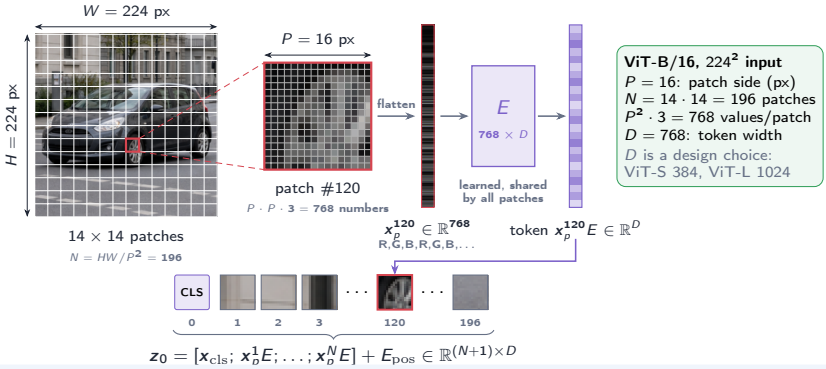
query patch i attends most to the other people, wherever they are

Notation. i: query token (patch) j: any token (incl. i)
$\boldsymbol{v}_j$: value vector: content of token j
α_{ij}: attention weight of i on j
$\boldsymbol{z}'_i$: updated vector of token i N: number of tokens

$$\boldsymbol{z}'_i = \sum_j \alpha_{ij} \boldsymbol{v}_j, \qquad \sum_j \alpha_{ij} = 1.$$

- Weights α_{ij} depend on token content.
- Interaction is global in a standard transformer.
- Geometry must be supplied through position information.
- Full attention costs $\mathcal{O}(N^2)$ token interactions.

From an image to a sequence of patches

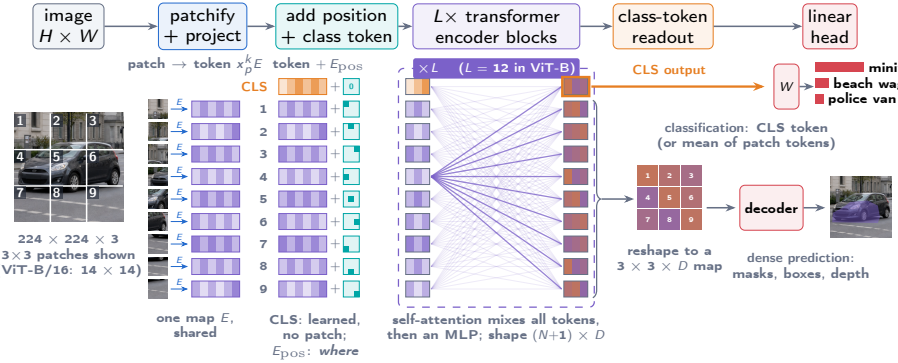


ViT-B/16, 224^2 input
$P = 16$: patch side (px)
$N = 14 \cdot 14 = 196$ patches
$P^2 \cdot 3 = 768$ values/patch
$D = 768$: token width
D is a design choice:
ViT-S 384, ViT-L 1024

Notation. H, W: image height, width $\quad x_p^k$: flattened patch k $\quad E$: learned projection
x_{cls}: learned class token $\quad E_{\text{pos}}$: learned position embeddings $\quad [\cdot;\cdot]$: stack as rows
z_0: transformer input

- Smaller patches preserve detail but increase N and attention cost.
- The class token is a learnable global readout; alternatives include average pooling.

Vision Transformer pipeline

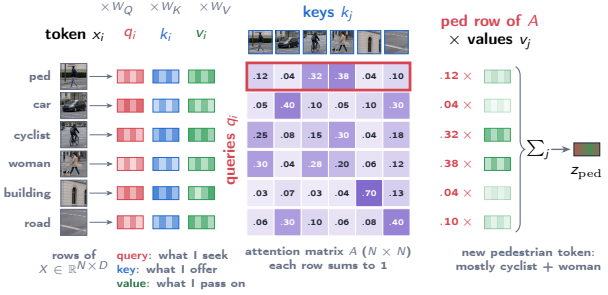


Notation. H, W: image height, width $\boldsymbol{x}_p^k$: flattened patch k E: learned patch projection, shared E_{pos}: learned position embeddings CLS: learned class token L: number of encoder blocks N: number of patch tokens D: token width (numbers per token) W: linear-head weights

Dosovitskiy et al., "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale," ICLR, 2021.

Self-attention mechanics: query, key, value

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \quad A = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big), \quad \mathrm{Attn}(X) = AV$$

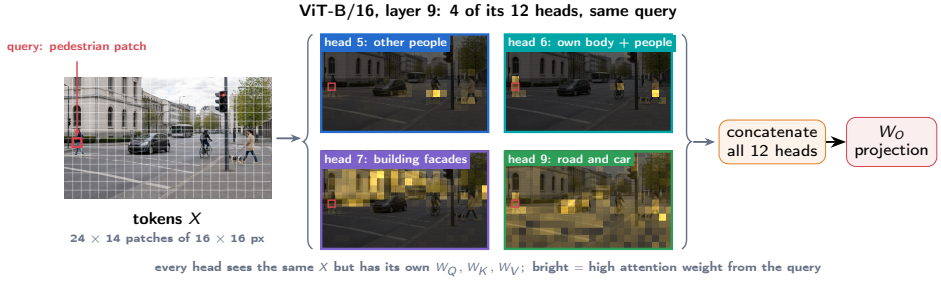


- $q_i \cdot k_j$: compatibility of every (query, key) pair: $QK^\top$ is $N \times N$.
- $1/\sqrt{d_k}$: keeps the scores at a workable scale.
- Row-wise softmax: each query's weights are positive and sum to 1.
- Output row i = weighted mix of the values v_j.

Weights here are illustrative.

Notation. X: input tokens, one row each N: number of tokens D: token width
x_i: token i (row i of X) W_Q, W_K, W_V: learned projection matrices
Q, K, V: queries, keys, values (rows q_i, k_i, v_i) d_k: query/key length $K^\top$: transpose of K
A: attention weights ($N \times N$) $\mathrm{Attn}(X)$: new tokens; z_{ped} is the ped row

Multi-head attention: several relational views

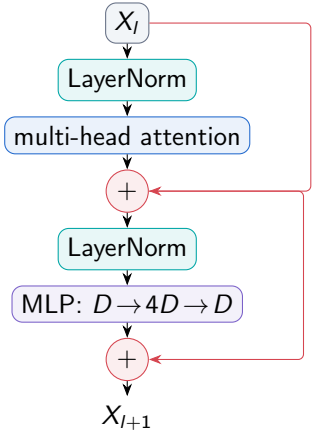


$$\mathrm{MHA}(X) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h) W_O.$$

Notation. X: input tokens h: number of heads (12 here)
head_m: output of head m (own W_Q, W_K, W_V) Concat: place side by side
W_O: learned output projection

- Heads provide separate learned subspaces, not guaranteed human-interpretable roles.

Transformer block: mix tokens, then mix channels



Notation. X_l: tokens into block l X_{l+1}: tokens out
D: token width (channels) $+$: residual add

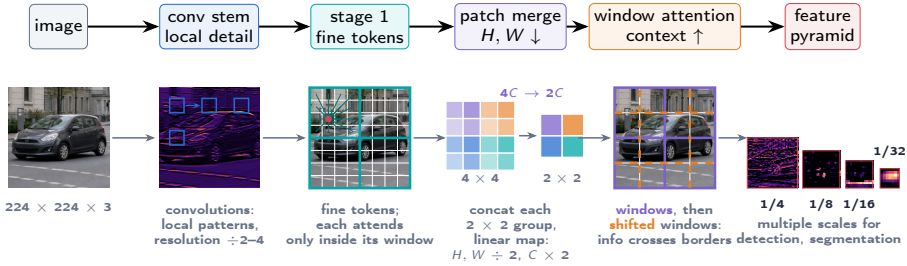
Two complementary operations

Attention mixes information *between tokens*.

MLP mixes information *between channels* independently at each token.

- Residual paths stabilize deep optimization.
- Pre-normalization is common in modern variants.
- Position information prevents permutation ambiguity.

Hybrid and hierarchical vision models



Conv stem
Efficiently captures local patterns and reduces resolution.

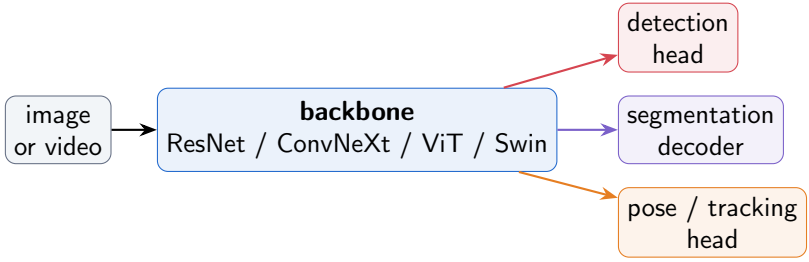
Local windows
Reduce attention cost; shifted windows exchange information.

Hierarchy
Produces multiple scales for detection and segmentation.

Notation. H, W: height, width of the token grid C: channels per token $\downarrow$, ÷2: reduced, halved ↑: increased

Liu et al., "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," ICCV, 2021.

The backbone: a reusable feature extractor



- A backbone outputs feature vectors or maps; the head maps them to a task representation.
- Pretraining can be supervised, self-supervised, or multimodal.

Wooclap

Classification is not enough: where is the object?



A single label discards

- object count,
- object location and scale,
- overlap and relationships,
- uncertainty per object.

image $\longrightarrow$ **set of objects**

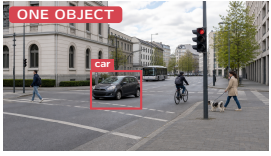
Object detection changes both the prediction space and the evaluation protocol.

Object detection

Object localization vs detection

Localization

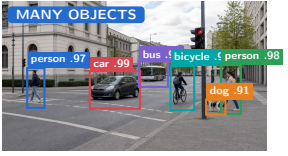
One known object category; predict one box.



$f_\theta(I) = (p,\ b), \qquad b \in \mathbb{R}^4.$

Detection

Unknown categories and unknown object count.



$f_\theta(I) = \{(p_i,\ b_i)\}_{i=1}^{N}.$

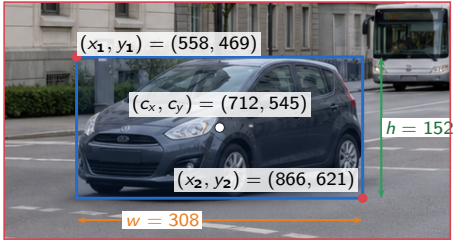
Notation. I: input image f_θ: the network, with weights θ p, p_i: class probabilities b, b_i: box: 4 real numbers ($\mathbb{R}^4$) N: number of objects in the image i: object index

- Detection adds a cardinality problem: the model must decide which candidates correspond to real objects.

Bounding box parameterization



image coordinates:
origin top-left, y **points** *down*
W = **1672**, H = **941 px**



Two equivalent forms

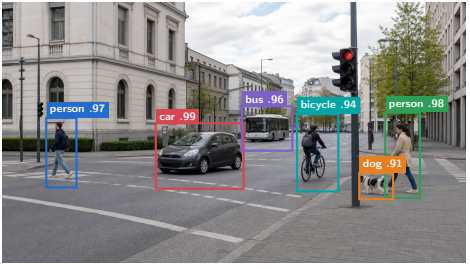
$(x_1, y_1, x_2, y_2) = (558, 469, 866, 621)$
$(c_x, c_y, w, h) = (712, 545, 308, 152)$
$\div(W, H)$: $(0.426, 0.579, 0.184, 0.162)$

- Normalize coordinates by image width and height.
- Enforce $w, h > 0$ through a suitable transform.
- Relative offsets are easier to learn across scales.

Notation. (x_1, y_1): top-left corner (x_2, y_2): bottom-right corner (c_x, c_y): box center
w, h: box width, height W, H: image width, height (all in pixels)

- A box is only four numbers, but their scale and parameterization shape optimization.

Detection output



$$\hat{\mathcal{Y}} = \{(\hat{p}_i, \hat{b}_i)\}_{i=1}^{K}$$

class probabilities $\hat{p}_i \in [0,1]^{C+1}$

box coordinates $\hat{b}_i \in \mathbb{R}^4$

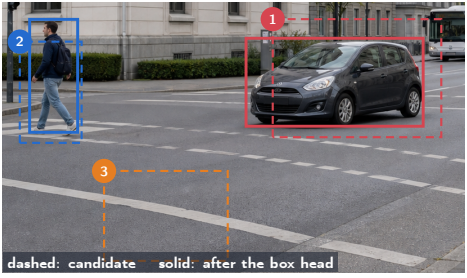
confidence or "no object"

- The target set has variable size N.
- Networks usually emit a fixed budget $K \gg N$.
- Assignment and filtering select detections.

Notation. $\hat{\mathcal{Y}}$: set of predictions $\hat{}$ (hat): predicted value i: prediction index
K: number of predictions (fixed) N: number of true objects
C: number of object classes ($+1$: background)

- Prediction is easy to tensorize; deciding which predictions count is the central difficulty.

Detection as multitask learning



cand.	IoU	class head	box head	loss
1	0.58	car 1.00	refined (solid)	$\mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{box}}$
2	0.64	person 1.00	refined (solid)	$\mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{box}}$
3	0.00	background 1.00	not trained	$\mathcal{L}_{\text{cls}}$ only

Per-candidate objective

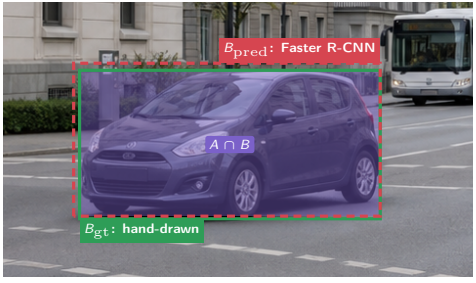
$$\mathcal{L}_i = \mathcal{L}_{\text{cls}}(p_i, c_i) + \lambda\, 1[c_i \neq \varnothing]\mathcal{L}_{\text{box}}(b_i, b_i^*).$$

- Every candidate learns foreground/background.
- Only matched positives receive box supervision.
- Loss scales must be balanced.

Notation. $\mathcal{L}_i$: loss of candidate i $\mathcal{L}_{\text{cls}}$: classification loss $\mathcal{L}_{\text{box}}$: box regression loss
p_i: class probabilities c_i: assigned true class $\varnothing$: background $1[\cdot]$: 1 if true, else 0
b_i: predicted box b_i^*: matched true box λ: loss weight IoU: box overlap (next slide)

Girshick, Fast R-CNN, 2015; Rezatofighi et al., GIoU, 2019.

Intersection over Union (IoU)





IoU $= 0.18$ IoU $= 0.56$ IoU $= 0.89$

dashed: B_{gt} shifted / scaled by hand

Everingham et al., PASCAL VOC, IJCV 2010.

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1].$$

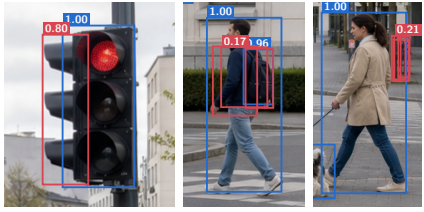
for the car: $\frac{45\,979 \text{ px}^2}{50\,011 \text{ px}^2} = 0.92$

0: disjoint 1: identical

Notation. A, B: two boxes
B_{gt}: true box B_{pred}: predicted box
$|\cdot|$: area (px^2)
$\cap$, $\cup$: intersection, union

- Scale invariant.
- Used for matching and evaluation.

Precision and recall for detection



TP: matched, class correct
FP: duplicate (.80), wrong class, detection of background
FN: no detection of a true object

score $\geq$	TP	FP	FN	precision	recall
0.995	7	0	4	1.00	0.64
0.9	11	0	0	1.00	1.00
0.5	11	1	0	0.92	1.00
0.05	11	10	0	0.52	1.00

11 GT objects; objects $<$ 30 px tall ignored (VOC "difficult")

Match before counting

A prediction is a TP when its class is correct, its IoU $\geq \tau$, and its ground-truth object is not already matched.

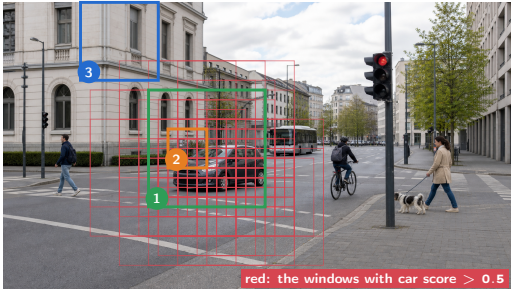
$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- Raise the score threshold: usually precision $\uparrow$, recall $\downarrow$.
- Duplicate boxes count as false positives.
- Missed objects reduce recall.

Notation. P, R: precision, recall FN: missed objects (false negatives) τ: IoU threshold, e.g. 0.5

- *Precision* is how accurate your predictions are, and *Recall* is how good you are at finding all positive objects.
- A detector is evaluated over a confidence sweep, not at one arbitrary threshold.

Why not slide a classifier everywhere?





1611 windows (3 sizes)
= 1611 ResNet-18 passes
= 29.4 s on this CPU

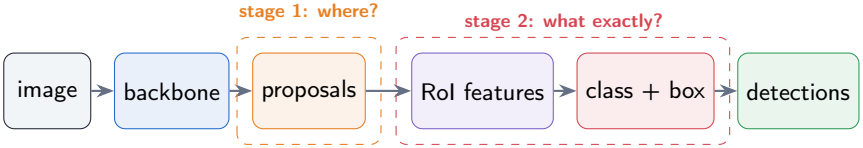
116 fire on *one* car
1383 are background ($< .1$)

- Adjacent windows repeat almost identical computation.
- Object size and aspect ratio are unknown.
- One object activates many overlapping windows.
- Background windows vastly outnumber objects.

Question: Which computation can be shared across all windows?

Two-stage detectors: proposals, then decisions

- In object detection, we have two main types of architectures: two-stage and single-stage detectors.



Strength

Expensive per-object reasoning is applied only to a sparse proposal set.

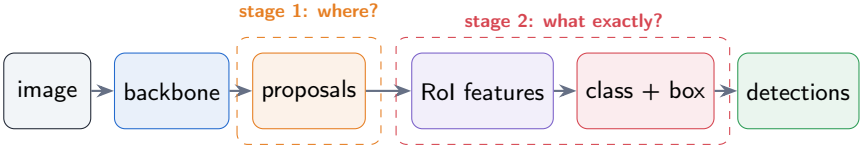
Cost

Proposal generation and per-RoI processing add latency and pipeline complexity.

Girshick et al., R-CNN, 2014; Girshick, Fast R-CNN, 2015.

Two-stage detectors: proposals, then decisions

- In object detection, we have two main types of architectures: two-stage and single-stage detectors.



Strength

Expensive per-object reasoning is applied only to a sparse proposal set.

Cost

Proposal generation and per-RoI processing add latency and pipeline complexity.

- The key part here is the region proposal step.

Girshick et al., R-CNN, 2014; Girshick, Fast R-CNN, 2015.

Region Proposals



Region proposal algorithm

Returns a few thousand **class-agnostic** boxes that probably contain *an* object. It never says *which* class: stage 2 decides.

- Goal: **high recall** with few boxes (thousands, not millions of windows).
- Generic cues only: colour, texture, edges, closed contours.
- Most proposals are background; the detector rejects them.
- Selective search (≈2 s/image, CPU).

Uijlings et al., Selective Search, IJCV 2013; Zitnick and Dollár, EdgeBoxes, ECCV 2014. Here: simplified selective search.

Non-maximum suppression



rank	score	IoU with #1	fate
1	0.96	—	kept
2	0.95	0.99	suppressed ($> \tau$)
3	0.95	0.98	suppressed ($> \tau$)
20	0.82	0.98	suppressed ($> \tau$)
40	0.43	0.95	suppressed ($> \tau$)
59	0.19	0.85	suppressed ($> \tau$)
79	0.06	0.80	suppressed ($> \tau$)

Whole image, all classes: **3,394** raw boxes → **636** after NMS.

1. Select the highest-score box.
2. Remove same-class boxes whose IoU with it exceeds τ_{nms}.
3. Repeat until no candidates remain.

NMS converts redundant local detections into one detection per object.

Notation. τ_{nms}, τ: IoU threshold
score: detector confidence

Neubeck and Van Gool, Efficient NMS, ICPR 2006.

NMS limitations

Overlapping real objects



IoU $= 0.31$ (shaded)

per-class NMS: both kept ✓

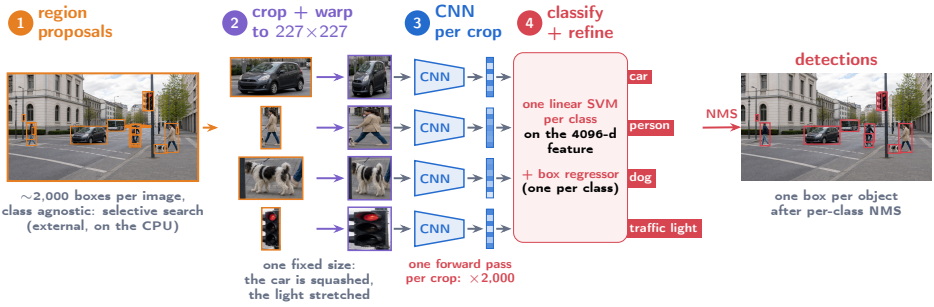
class-agnostic, τ=0.3: **bicycle deleted** ×

Same within one class in crowds (pedestrians, parked cars).

Notation. τ: IoU threshold
IoU: overlap with kept box

- **Crowds:** real objects may overlap strongly.
- **Thresholds:** one τ does not fit every class and scale.
- **Discrete step:** NMS is not learned end to end.

R-CNN: a CNN on every region proposal



Trained in separate stages

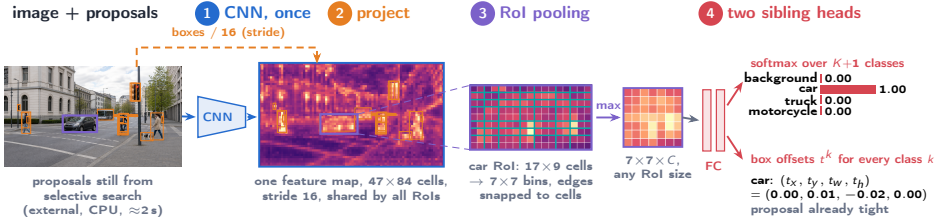
fine-tune the CNN on warped crops → fit one SVM per class on its features (cached on disk) → fit the box regressors: three separate objectives.

Bottleneck: step 3

~2,000 CNN passes per image recompute the same convolutions on overlapping crops: ≈47 s per image with VGG-16 on a GPU.

Girshick, Donahue, Darrell, and Malik, R-CNN, CVPR 2014; timing: Girshick, Fast R-CNN, ICCV 2015.

Fast R-CNN: one CNN pass, then pool every proposal



Trained jointly (given the proposals) with one multitask loss per RoI:

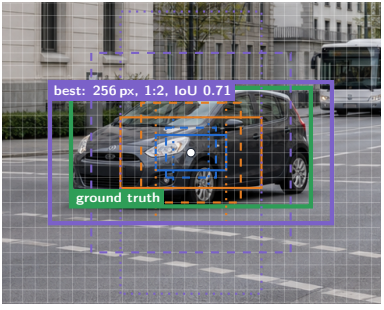
$$L = L_{\text{cls}}(p, u) + \lambda\,[u \geq 1]\,L_{\text{loc}}(t^u, v), \qquad L_{\text{cls}} = -\log p_u, \qquad L_{\text{loc}} = \textstyle\sum_i \text{smooth}_{L_1}(t^u_i - v_i)$$

Notation. p: softmax over $K+1$ classes K: number of object classes C: feature channels
u: true class (0: background) t^u: predicted box offsets for class u v: target offsets
i: one of x, y, w, h $[u \geq 1]$: 1 if object, 0 if background λ: balance weight (paper: 1)

Shared convolutions: ≈0.32 s instead of 47 s per image (VGG-16); no SVMs. **Bottleneck now: the proposals**: selective search takes ≈2 s per image (CPU). We need to make the proposals part learnable!

Girshick, Fast R-CNN, ICCV 2015 (timings: VGG-16, PASCAL VOC, test time excluding proposals).

Anchor boxes



64 128 256 px; $h{:}w$ = 1:2 solid, 1:1 dashed, 2:1 dotted
grid: stride 16 px; this cell owns 9 anchors

whole image: 104 × 59 cells × 9 = 55224 anchors

Notation. a: anchor box c_x^a, c_y^a: anchor center
w_a, h_a: anchor width, height
$\hat{c}_x, \hat{c}_y, \hat{w}, \hat{h}$: predicted center, size
$\hat{b}$: predicted box

Reference geometry

Each location owns boxes with several scales and aspect ratios:
$a = (c_x^a, c_y^a, w_a, h_a)$.

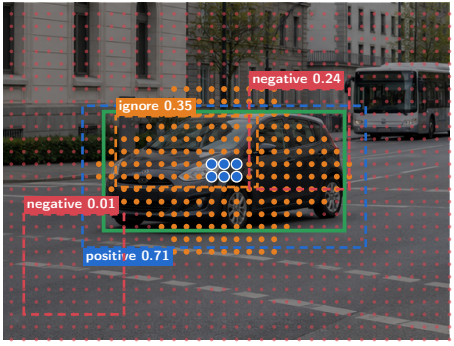
$$\hat{c}_x = c_x^a + t_x\, w_a, \qquad \hat{c}_y = c_y^a + t_y\, h_a,$$
$$\hat{w} = w_a e^{t_w}, \qquad \hat{h} = h_a e^{t_h}.$$

The head predicts objectness, class, and offsets (t_x, t_y, t_w, t_h).

Best anchor $a = (712, 552, 362, 181)$
targets $t = (0.000, -0.039, -0.162, -0.175)$
$\Rightarrow \hat{b}$ = the car, $(712, 545, 308, 152)$

Ren et al., Faster R-CNN, NeurIPS 2015; Liu et al., SSD, ECCV 2016.

Anchor matching



dot = one cell, coloured by its best anchor: IoU $\geq$ .7 .3–.7 $<$ .3

whole image, 11 objects: 55224 anchors
$\rightarrow$ 22 positive, 1607 ignored, 53595 negative

$$y(a,g) = \begin{cases} +1 & \text{IoU}(a,g) \geq \tau_+, \\ 0 & \text{IoU}(a,g) < \tau_-, \\ \perp & \text{otherwise.} \end{cases}$$

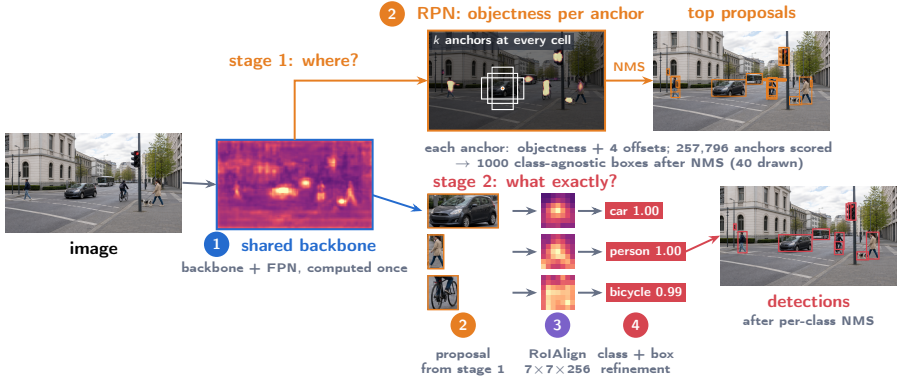
Faster R-CNN RPN: $\tau_+ = 0.7$, $\tau_- = 0.3$

- Thresholds are design choices, not universal constants.
- Ignored ($\perp$) anchors contribute no classification loss.
- Assignment defines the labels seen during training.

Notation. a: an anchor g: a ground-truth box $y(a,g)$: training label of a
$+1$ / 0: object / background $\perp$: ignored, no label τ_+, τ_-: positive, negative IoU thresholds

Ren et al., Faster R-CNN, 2015.

Faster R-CNN: Fast R-CNN + a learned RPN

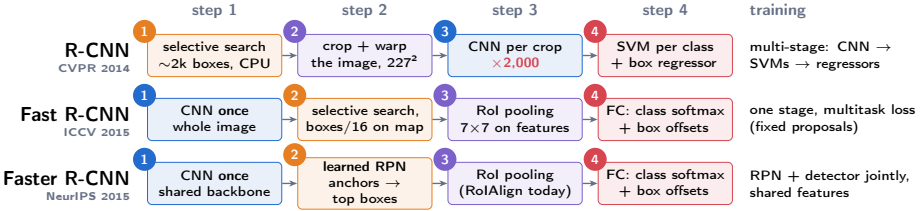


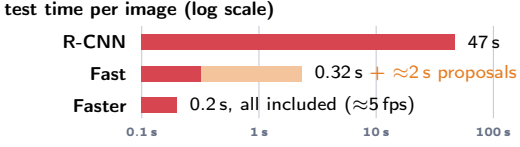
1. RPN and detector reuse the same feature map.
2. The Region Proposal Network is learned: per cell, k anchors → objectness + 4 offsets; NMS keeps the top N (paper: k=9, N=300).
3. Proposals are class agnostic.
4. The second stage classifies and refines each RoI.

Proposals: ≈10 ms, not ≈2 s (≈5 fps).

Ren, He, Girshick, and Sun, Faster R-CNN, NeurIPS 2015.

From R-CNN to Faster R-CNN: what changed





PASCAL VOC 2007 mAP

R-CNN 66.0

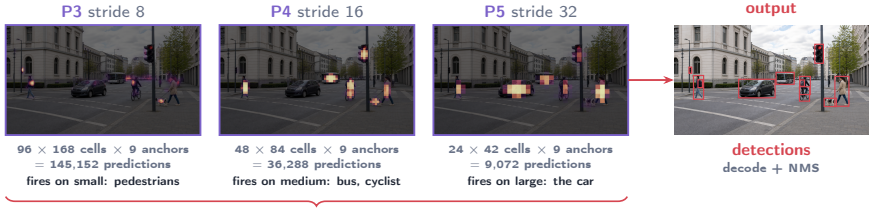
Fast 66.9

Faster 69.9

Each step removed the slowest part: the **CNN** moves from step 3 to step 1 (Fast), then the **proposals** are learned (Faster).

All: VGG-16, VOC 2007 test. Girshick et al., CVPR 2014; Girshick, ICCV 2015 (Table 4); Ren et al., NeurIPS 2015.

One-stage detectors



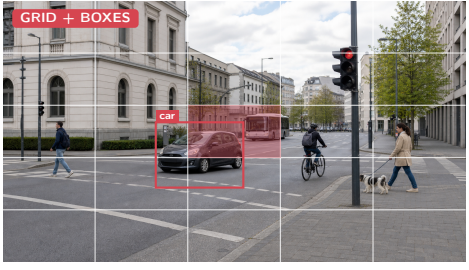
One pass

No explicit proposal/RoI stage: predict densely at several resolutions.

- Low latency and simple batching.
- Multi-scale maps serve different object sizes.
- Dense supervision creates severe class imbalance.

Liu et al., SSD, ECCV 2016; Lin et al., FPN, CVPR 2017; Lin et al., RetinaNet, ICCV 2017.

YOLO intuition



Historical YOLOv1 view

The grid cell containing an object's center is responsible for predicting it.

$$S \times S \times (B \cdot 5 + C)$$

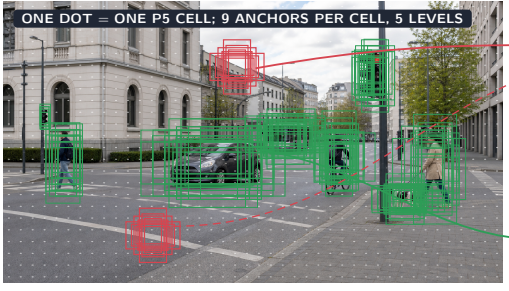
Notation. S: grid size
B: boxes per cell
5: x, y, w, h, confidence C: classes

- One network maps pixels directly to detections.
- Joint reasoning uses global image context.
- Modern variants use different heads and assignments.

Redmon et al., YOLO, CVPR 2016.

Problem with one-stage detectors

- One-stage detectors classify every anchor: here 193,374 anchors densely cover the image → **foreground–background imbalance**.
- Two-stage detectors filter first: ≈2k proposals, then mini-batches sampled with ≈25 % positives.



192,556 negative anchors
mostly easy background (sky, road, walls): **almost no useful signal**

≈ 1 positive for every 669 negatives

288 positive anchors
few, but **rich information** for learning

Lin et al., Focal Loss for Dense Object Detection, ICCV 2017; Liu et al., SSD, ECCV 2016. Counts: RetinaNet anchors on this scene.

Motivation of focal loss

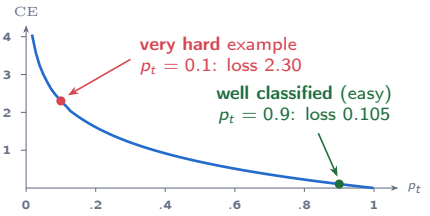
Solution idea: **change the loss function**. Recall the cross-entropy of the true class:

$$\mathrm{CE}(p_t) = -\log p_t, \qquad p_t = \begin{cases} p & \text{if } y = 1, \\ 1 - p & \text{otherwise.} \end{cases}$$

100 hard examples
$\times\, 2.30 = 230$

100,000 easy examples
$\times\, 0.105 \approx 10{,}500$

The easy ones add $\approx 46\times$ more loss, although each is almost correct.



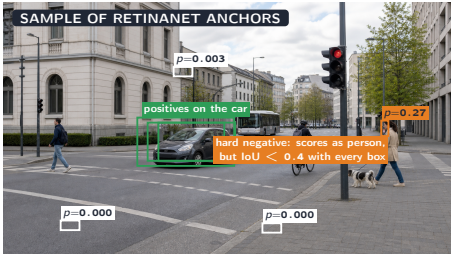
On our scene (RetinaNet anchors): 191,102 easy negatives, each with $p_t \approx 0.9994$, still make **84.2 %** of the total cross-entropy; the positives only 3.6 %.

Notation. p: predicted probability of "object" $y \in \{0, 1\}$: true label
p_t: probability given to the true class hard: p_t small (a mistake) easy: p_t close to 1

Question: Many tiny losses dominate the total loss and its gradient. How can we *down-weight* the examples that are already well classified?

Lin et al., "Focal Loss for Dense Object Detection," ICCV 2017.

Class imbalance and focal loss



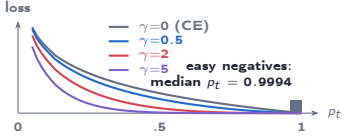
	anchors	share of total loss: CE	focal
positives	288	3.6%	72.0%
hard negatives	1,454	12.1%	27.2%
easy negatives	191,102	84.2%	0.9%

Notation. p_t: prob. of the true class
α_t: class weight $\gamma \geq 0$: focusing parameter
$(1-p_t)^\gamma$: modulating factor

Focal loss

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log p_t$$

The modulating factor suppresses gradients from already-correct examples.

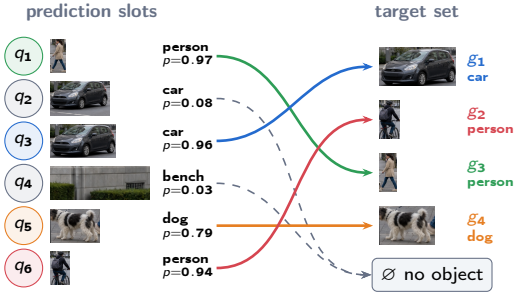


- $\gamma = 0$ recovers weighted cross-entropy.
- Larger γ focuses learning on hard mistakes.
- It enabled accurate dense one-stage detection in RetinaNet.

Lin et al., Focal Loss, ICCV 2017. Shares: RetinaNet (α=.25, γ=2) on this scene.

Modern detectors with ViTs

Modern detection: set prediction



6 slots shown (DETR: $N = 100$); each target gets exactly one slot, the rest learn "no object" ⇒ no duplicates to suppress

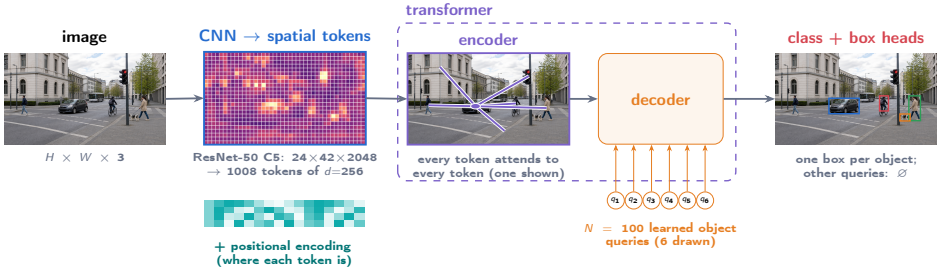
Notation. q_i: prediction slot i
g_j: ground-truth object j ∅: the "no object" class
N: number of slots p: predicted class score

Carion et al., DETR, ECCV 2020.

- Predict a fixed-size unordered set.
- Match each object to exactly one slot.
- Train unmatched slots as "no object."
- Duplicate removal is part of the learning objective.

no anchors no NMS

DETR overview



- Global attention connects distant image regions.
- Learned queries compete for objects in parallel.
- Bipartite set loss trains the complete pipeline.
- Original DETR converges slowly and struggles with small objects.

Notation. H, W: image height and width d: token dimension q_i: learned object query i
$\varnothing$: "no object" prediction

Carion et al., DETR, ECCV 2020. Real input and C5 map; attention and outputs illustrative.

Hungarian matching



rows: prediction slots q_i; columns: targets g_j

green: optimal one-to-one match, total cost -2.8 (the minimum over all 360 assignments); q_2, $q_4 \to \varnothing$

Pairwise matching cost

$$\mathcal{C}(i,j) = -p_i(c_j) + \lambda_1 \|b_i - b_j\|_1 + \lambda_g \mathcal{L}_{\text{GIoU}}$$

Example, q_3 vs. g_1 (car), λ_1=5, λ_g=2:
$-0.96 + 5 \cdot 0.009 + 2 \cdot 0.06 = -0.80$

$$\hat{\sigma} = \arg\min_{\sigma} \sum_{j} \mathcal{C}(\sigma(j), j).$$

Match first; then compute class and box losses on the selected pairs.

Notation. $\mathcal{C}(i,j)$: cost of slot q_i for target g_j
c_j: class of g_j $p_i(c_j)$: slot i's score for c_j
b_i, b_j: predicted, target box λ_1, λ_g: weights
$\mathcal{L}_{\text{GIoU}}$: 1−generalized IoU
σ: assignment $j \mapsto \sigma(j)$
$\hat{\sigma}$: cheapest assignment

Kuhn, 1955; Carion et al., DETR, 2020. Slots: RetinaNet boxes standing in for DETR outputs.

Segmentation

Semantic segmentation



Notation. x: input image f_θ: network, weights θ
z: per-pixel class scores H, W: image height, width
C: number of classes z_{ijc}: score of class c at pixel (i, j)
$\hat{y}_{ij}$: predicted label of pixel (i, j)
$\arg\max_c$: the highest-scoring class

DENSE PREDICTION

Assign one semantic class to **every pixel**:

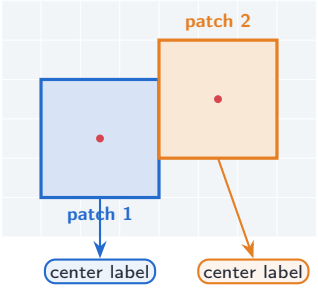
$$z = f_\theta(x) \in \mathbb{R}^{H \times W \times C},$$
$$\hat{y}_{ij} = \arg\max_c z_{ijc}.$$

- Output remains aligned with the input image.
- "Stuff" (road, sky) and "things" (car, person) share one label map.
- Same-class objects are *not* separated into instances.

Classification asks *what*; segmentation asks *what at each location*.

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation," CVPR 2015.

Naive patch classification



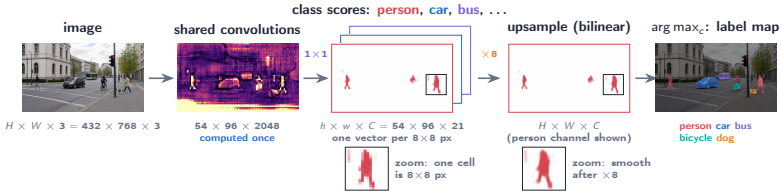
Train a classifier on a crop around each target pixel.

- Neighboring patches overlap almost completely.
- Their convolutional features are recomputed again and again.
- Small patches miss context; large patches multiply the cost.
- A full image needs roughly $H \times W$ classifier calls.

Bottleneck: correct idea, wasteful implementation—shared computation is missing.

Farabet et al., "Learning Hierarchical Features for Scene Labeling," IEEE TPAMI 2013.

Fully convolutional networks



Key conversion

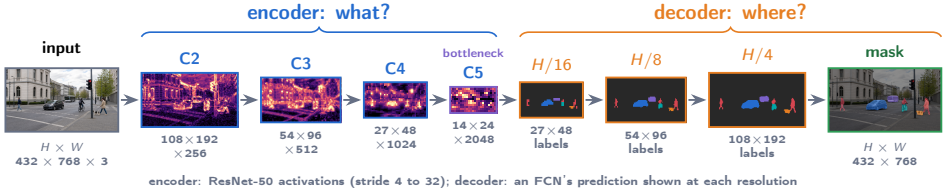
Replace dense classifier layers by convolutions. Spatial score maps emerge in one forward pass.

- accepts variable image sizes;
- shares features across locations;
- trades resolution for receptive field.

Notation. H, W: image height, width h, w: feature-map height, width (8× smaller)
C: number of classes (21) 1×1: per-cell conv ×8: upsampling factor
arg max$_c$: best-scoring class

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation," CVPR 2015.

Encoder–decoder design



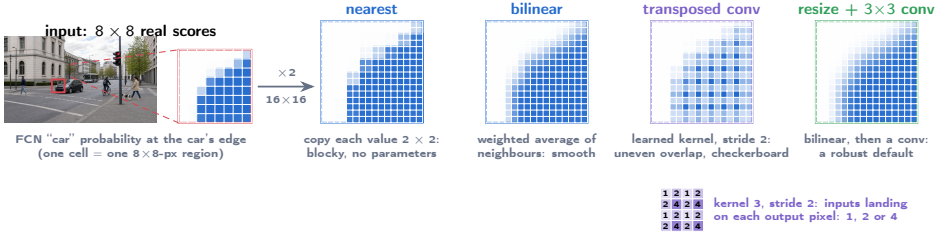
Notation. $H \times W$: input height × width H/s: map s times smaller than the input
C2–C5: ResNet stages (stride 4 to 32) $h \times w \times c$: height × width × channels

Downsampling enlarges the receptive field and builds invariance, but discards precise location.

Upsampling restores spatial resolution, but must reconstruct boundaries from compressed features.

- Segmentation needs both global context and pixel-level localization.

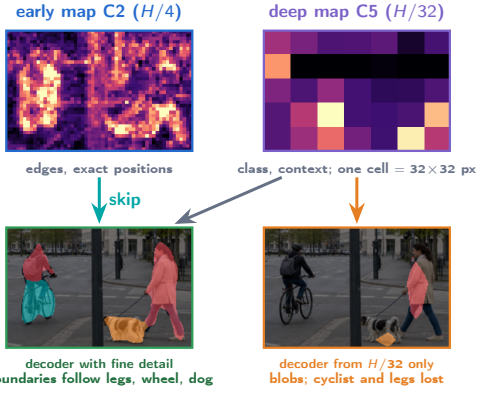
Upsampling methods



Upsampling increases sampling density; it cannot recover detail that the encoder never preserved.

Dumoulin and Visin, "A Guide to Convolution Arithmetic for Deep Learning," 2016; Odena et al., "Deconvolution and Checkerboard Artifacts," 2016.

Skip connections recover detail



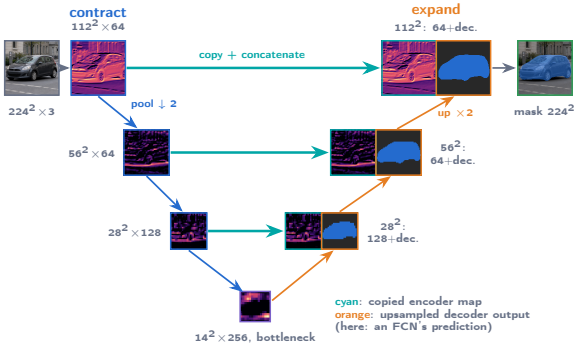
Notation. H: input height
$H/4$, $H/32$: map 4, 32 times smaller

- Early maps retain edges, corners, and accurate coordinates.
- Deep maps encode class and broad context.
- Concatenation or addition lets the decoder use both.
- Feature sizes must be aligned by padding, cropping, or resizing.

Contrast with ResNet: these links bridge scales; residual links mainly ease optimization within a scale.

Long et al., CVPR 2015; Ronneberger, Fischer, and Brox, MICCAI 2015.

U-Net



Notation. $112^2 \times 64$: 112×112 map, 64 channels
+dec.: plus the decoder's channels

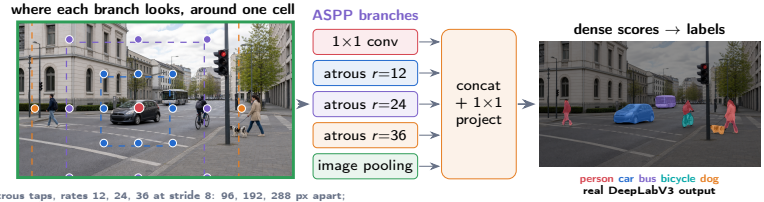
BIOMEDICAL ORIGIN

- Symmetric encoder and decoder.
- Skip features are **concatenated**, not merely added.
- Strong localization with limited annotated data.
- Modern U-Nets often add padding, normalization, and residual blocks.

Enduring idea: pair semantics with same-scale detail.

Ronneberger, Fischer, and Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI 2015.

DeepLab: context at multiple scales



Atrous convolution inserts gaps into the kernel:

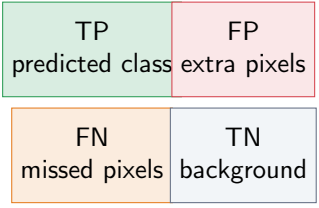
$$y[p] = \sum_k w[k]\, x[p + rk].$$

- Larger field of view without another pooling step.
- Parallel rates capture objects at different scales.
- A light decoder can sharpen boundaries.

Notation. x, y: input, output feature maps p: output position k: kernel tap $(-1, 0, 1)$
$w[k]$: kernel weight r: rate: gap between taps (r=1: ordinary conv)

Chen et al., "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017; Chen et al., "Encoder–Decoder with Atrous Separable Convolution," ECCV 2018.

Segmentation metrics



per class, over all pixels

$$\text{pixel accuracy} = \frac{\sum_c \text{TP}_c}{|\Omega|},$$

$$\text{IoU}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c},$$

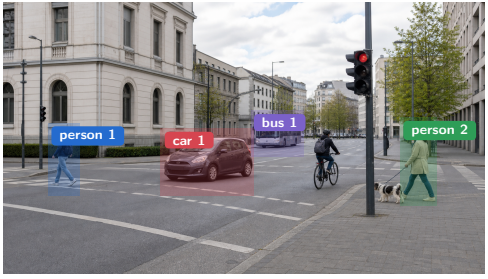
$$\text{mIoU} = \frac{1}{C}\sum_{c=1}^{C} \text{IoU}_c.$$

Notation. c: a class
C: number of classes
TP_c: class-c pixels predicted as c
FP_c: other pixels predicted as c
FN_c: class-c pixels missed
Ω: all pixels; $|\Omega|$ their count

- Accuracy can look excellent when background dominates.
- mIoU gives each class equal weight.

Everingham et al., "The PASCAL Visual Object Classes Challenge," IJCV 2010.

Instance segmentation



Notation. k: instance index K: number of instances
c_k: class s_k: confidence score b_k: box
M_k: binary mask (the object's pixels)

Predict a set of object-level masks:

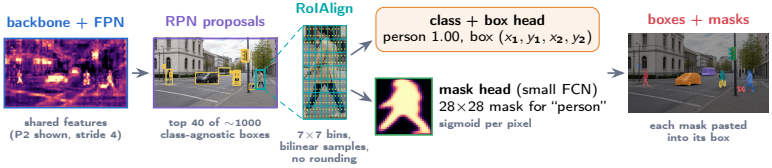
$$\{(c_k, s_k, b_k, M_k)\}_{k=1}^{K}.$$

- Each countable **thing** receives an identity.
- Overlapping objects remain separate.
- Masks capture shape; boxes still help localize proposals.
- "Stuff" is usually left to a semantic branch.

Instance segmentation predicts classes *and individuals*.

Hariharan et al., "Simultaneous Detection and Segmentation," ECCV 2014.

Mask R-CNN



One extra branch

Faster R-CNN supplies proposals, classes, and boxes; a parallel FCN predicts an $m \times m$ mask for each RoI.

- Mask loss is pixel-wise binary cross-entropy.
- Detection and masks share backbone features.

Notation. m: mask side (28 here) RoI: proposal region
$(x_1, y_1), (x_2, y_2)$: top-left, bottom-right box corners

He et al., "Mask R-CNN," ICCV 2017.

Pose estimation and tracking

Pose estimation



Notation. $\mathcal{K}$: keypoints of one person
(x_j, y_j): image position of joint j
J: number of joints (17 in COCO)

Predict a structured set of landmarks for a person or object:

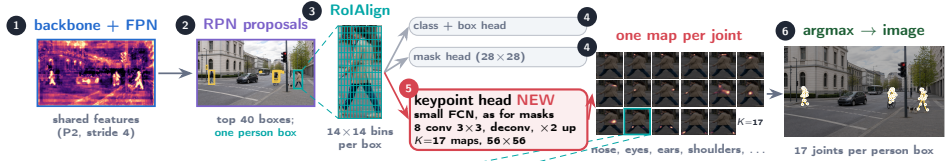
$$\mathcal{K} = \{(x_j, y_j, v_j)\}_{j=1}^{J},$$

where v_j records visibility or confidence.

- **2D / 3D pose:** keypoints in image coordinates or joints in camera/world coordinates.
- A fixed skeleton graph connects related joints.
- Multiple people add a second problem: assigning joints to identities.

Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," CVPR 2014.

Pose = one more head on Mask R-CNN (Keypoint R-CNN)



1–4: Mask R-CNN, unchanged (shared features, proposals, RoIAlign, box + mask heads).

5: the only new part, a keypoint head: a small FCN on the same RoI features that outputs one map per joint.

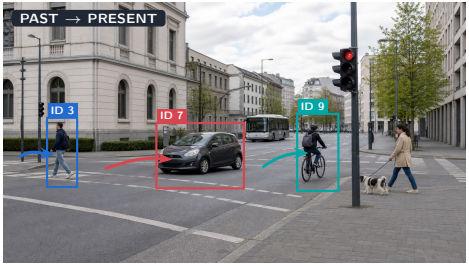
6: decode each map: argmax cell, then box → image.

A **top-down** method: detect people first, then estimate each person's pose inside its box.

Notation. K: joints per person ($= J = 17$, COCO) (x_1, y_1): top-left corner of the person box
w, h: box width, height $(\hat{u}, \hat{v})$: argmax cell (column, row) of a map
$(\hat{x}, \hat{y})$: decoded joint (image pixels)

He, Gkioxari, Dollár, and Girshick, "Mask R-CNN," ICCV 2017 (keypoint head).

Multi-object tracking



Given detections over video, estimate trajectories

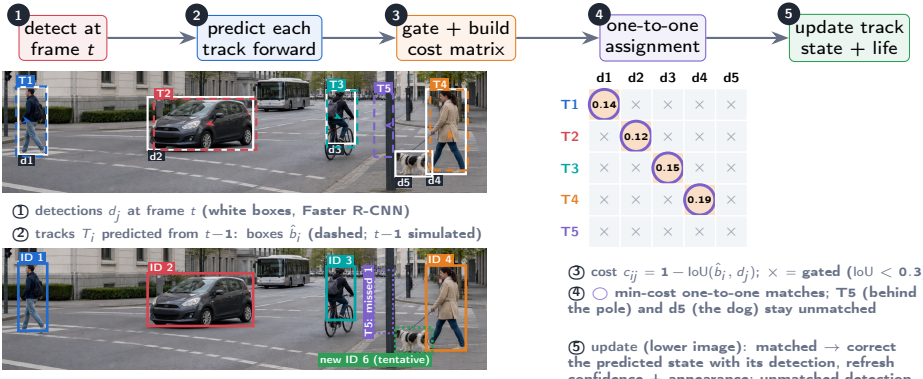
$$\mathcal{T}_i = \{(t, b_t^i, c_i, \mathrm{ID}_i)\}_{t=t_s}^{t_e}.$$

- Localize each object at each time.
- Preserve a stable identity through motion and occlusion.
- Start and terminate tracks as objects enter or leave.
- Diagnose false/missed tracks, fragmentation, and **ID switches**.

Notation. $\mathcal{T}_i$: trajectory of object i
t: frame index b_t^i: its box at frame t
c_i: its class ID_i: its identity label
t_s, t_e: first / last frame of the track

Milan et al., "MOT16: A Benchmark for Multi-Object Tracking," arXiv 2016.

Tracking by detection



① detections d_j at frame t **(white boxes, Faster R-CNN)**

② tracks T_i predicted from $t-1$: boxes $\hat{b}_i$ **(dashed; $t-1$ simulated)**

③ cost $c_{ij} = 1 - \mathrm{IoU}(\hat{b}_i, d_j)$; × = **gated** ($\mathrm{IoU} < 0.3$)

④ ◯ **min-cost one-to-one matches; T5 (behind the pole) and d5 (the dog) stay unmatched**

⑤ **update (lower image): matched → correct the predicted state with its detection, refresh confidence + appearance; unmatched detection → tentative new track; unmatched track → keep briefly, delete after repeated misses**

Bewley et al., "Simple Online and Realtime Tracking," ICIP 2016 (SORT).

Main takeaways

- Vision is an inverse problem. One image is compatible with many 3D worlds, so a model must combine data with prior assumptions.
- Learning is loss minimization; the goal is generalization. Fit on the training set, choose on the validation set, report once on the test set.
- Convolution is a local pattern detector with shared weights. Stacking layers grows the receptive field and builds a hierarchy: edges $\rightarrow$ parts $\rightarrow$ objects.
- ViTs turn an image into patch tokens. Self-attention lets every token use evidence from any other token in one layer, at a cost of $\mathcal{O}(N^2)$ for N tokens; position is supplied by embeddings.
- The backbone is reusable. A pretrained CNN or ViT, often with a feature pyramid, serves every task; mostly the head changes.

Main takeaways

- The output defines the task: label → boxes → masks → keypoints → identities over time. Each needs its own head, loss, and metric.
- Detection scores many candidates (anchors, proposals) and matches them to objects by IoU. R-CNN → Fast → Faster R-CNN: share the CNN, then learn the proposals. One-stage detectors predict densely and need the focal loss; NMS removes duplicates; DETR learns a one-to-one set prediction instead.
- Segmentation is classification at every pixel: an encoder–decoder, with skip connections or atrous convolution to keep both detail and context. Mask R-CNN adds a mask head for each detected object.
- Pose is one more head on Mask R-CNN: one map per joint inside each person box. Tracking links detections across frames by motion, geometry, and appearance.
- Evaluate with task metrics (mAP, mIoU, etc.) and inspect failures by scale, class, and occlusion before adding complexity.

Activity

This lecture is quite important since it lays down the foundation of what follows in the course. Please take the time to reflect on what we learned today, and on a piece of paper, please:

1. Write the most important things you learned today
2. If you have questions about the lecture, you can write them, and I'll try to answer a couple in the following sessions
3. Give me the piece of paper before leaving
4. Please sign it if you want to (participation), and give it to me before leaving

Thank you!

Jhony H. Giraldo: jhony.giraldo@telecom-paris.fr
(https://jhonygiraldo.github.io/)