



# Нейронні мережі

Лекція 7: Комп'ютерний зір II

Кочура Юрій Петрович  
[iuriy.kochura@gmail.com](mailto:iuriy.kochura@gmail.com)  
[@y\\_kochura](https://twitter.com/y_kochura)

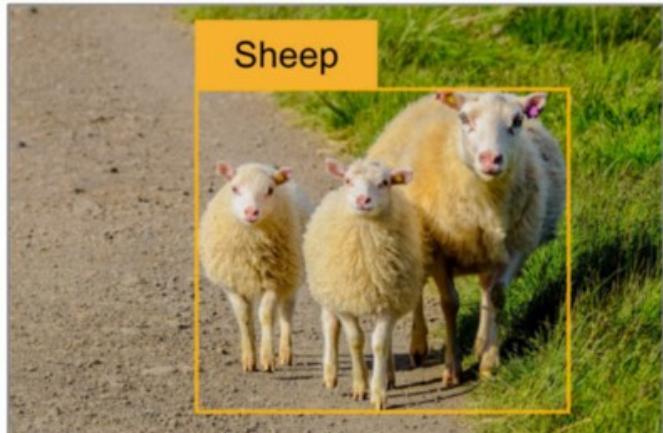
# Сьогодні

Як створювати нейронні мережі для деяких просунутих задач комп'ютерного зору:

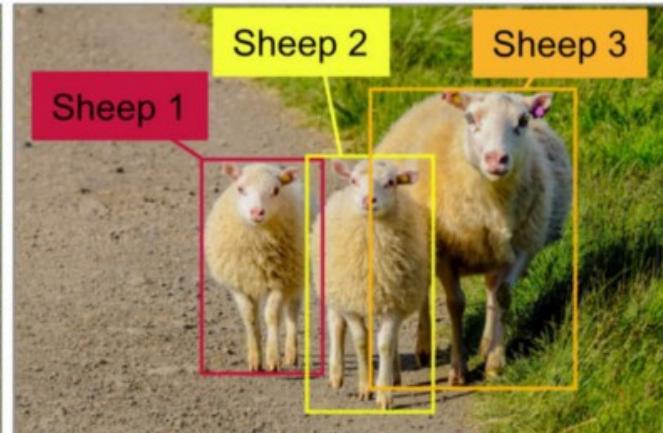
- Класифікація

- Детекція об'єктів

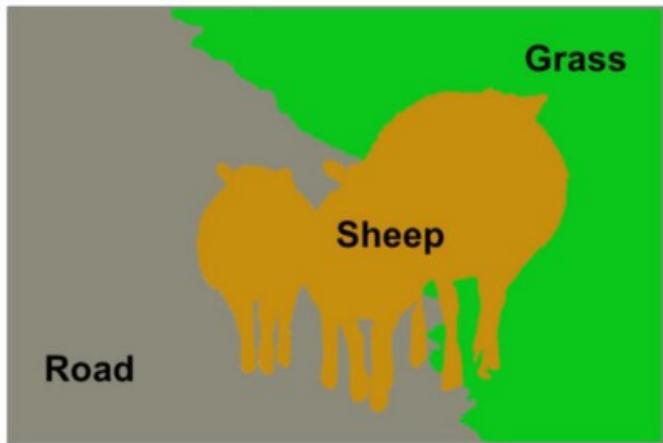
- Сегментація



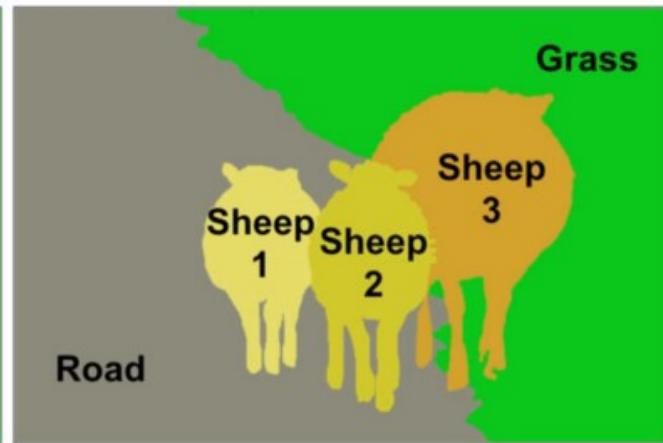
**Classification + Localization**



**Object Detection**



**Semantic Segmentation**



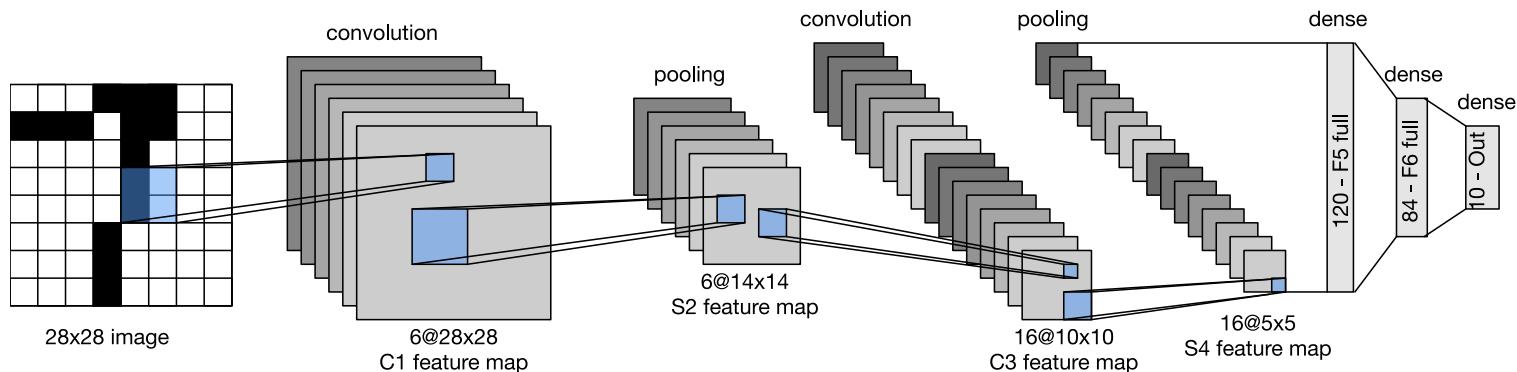
**Instance Segmentation**

# Класифікація

Поради з використання ConvNet для класифікації зображень

# Згорткові нейронні мережі

- Згорткові нейронні мережі поєднують згорткові шари, шари агрегації (pooling) та повнозв'язні шари.
- Забезпечують передові результати для просторово структурованих даних, таких як зображення, звук або текст.



# Класифікація

- Активація у вихідному шарі – це **Softmax**-активація, яка формує вектор ймовірнісних оцінок

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_C] \in \Delta^C$$

$$\sum_{i=1}^C \hat{y}_i = 1$$

$$\hat{y}_i = P(Y = i | \mathbf{x}) = \text{softmax}(z_i) = \frac{\exp^{z_i}}{\sum_j^C \exp^{z_j}},$$

де  $C$  – кількість класів;

- Функція втрат – перехресна ентропія:  $\mathcal{L} = -\sum_{i=1}^C y_i \log(\hat{y}_i)$

# Три способи покращити дані

Нестача даних – найбільше обмеження для ефективності моделей глибокого навчання.

1. Збір та анотація додаткових даних зазвичай є дорогим і трудомістким завданням.



# Три способи покращити дані

Нестача даних – найбільше обмеження для ефективності моделей глибокого навчання.

1. Збір додаткових даних зазвичай є дорогим і трудомістким завданням.
2. Створення синтетичних даних може бути складним завдання і не відображати реальний розподіл.

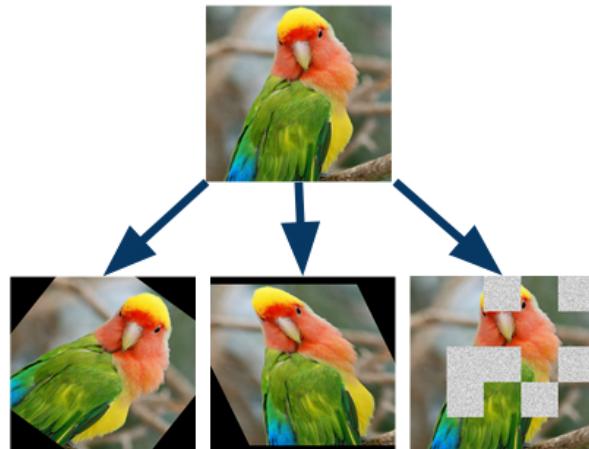


Мерилін Монро в азійському, оригінальному та африканському стилях

# Три способи покращити дані

Нестача даних – найбільше обмеження для ефективності моделей глибокого навчання.

1. Збір додаткових даних зазвичай є дорогим і трудомістким завданням.
2. Створення синтетичних даних може бути складним завдання і не відображати реальний розподіл.
3. Доповнення даних за допомогою базових перетворень – простий та ефективний підхід, але пошук хорошої стратегії для доповнення даних може вимагати численних експериментів.



## Геометричні перетворення



## Перетворення кольору



## Шум /перекриття



## Погода





$$\text{Error Reduction} = \frac{E_{\text{old}} - E_{\text{new}}}{E_{\text{old}}} \cdot 100\% = \frac{0.143 - 0.058}{0.143} \cdot 100\% = 59.4\%$$

# Попередньо навчені моделі

- Навчання моделі «з нуля» може тривати дні або навіть тижні.
- Багато моделей, навчених на великих наборах даних, доступні у відкритому доступі. Їх можна використовувати як: **екстрактори ознак** або як основу для розумної ініціалізації при донаавченні.
- Такі моделі слід розглядати як універсальні, багаторазові активи.

# Передавальне навчання

- Беремо попередньо навчену мережу, видаляємо останній (або кілька останніх) шарів і використовуємо решту як фіксований екстрактор ознак.
- На основі цих ознак навчаємо нову модель для нової задачі.
- Часто показує кращі результати, ніж ручне створення ознак або навчання моделі лише на нових даних.

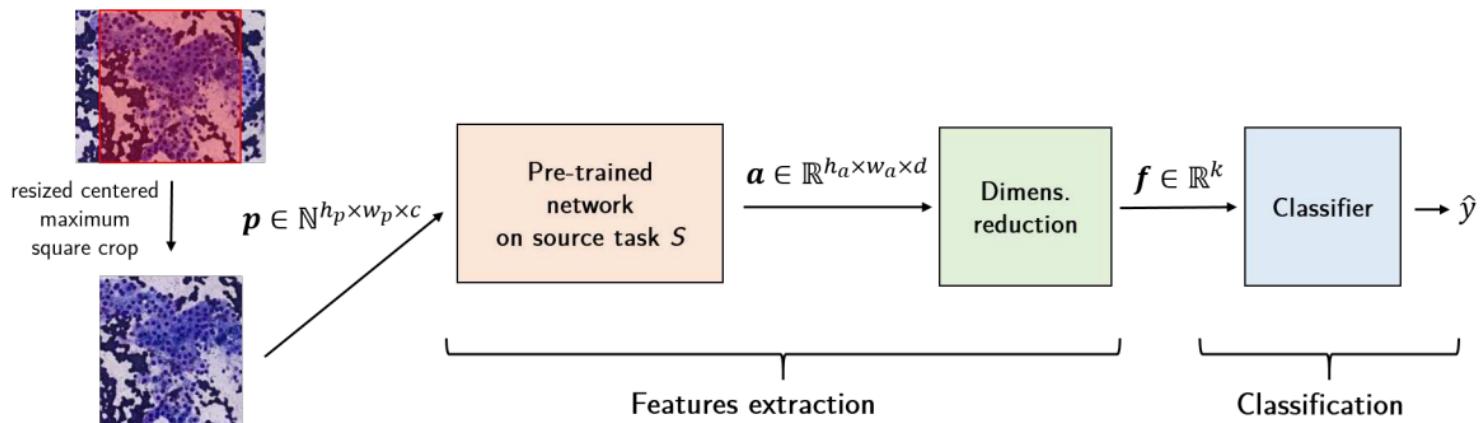
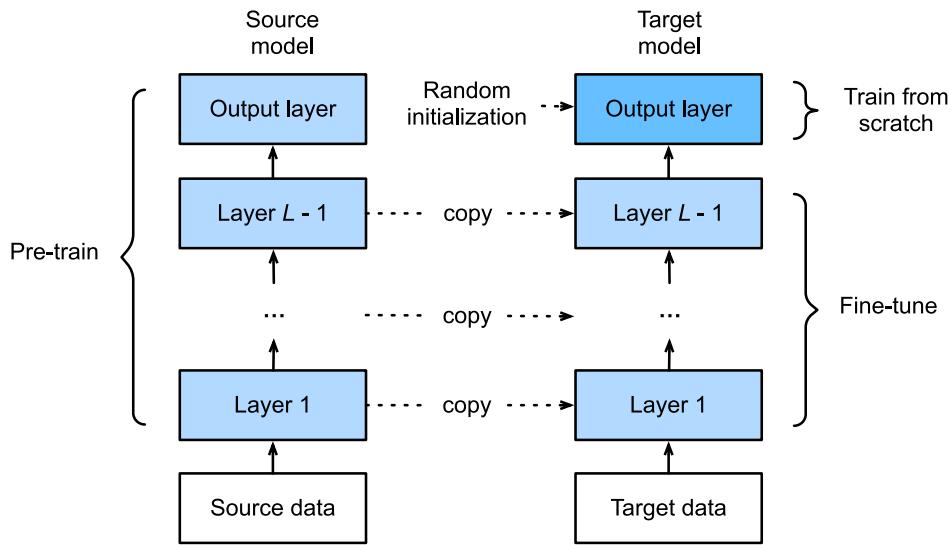


Figure 2. Feature extraction from pre-trained convolutional neural networks



## Тонке налаштування

Ми не лише використовуємо попередньо навчену модель для нової задачі, але й дозволяємо моделі адаптуватися до нових даних, змінюючи її ваги під час навчання.

Моделі, які попередньо навчені на **ImageNet**: передані/тонко налаштовані мережі зазвичай працюють краще навіть тоді, коли вхідні зображення належать до іншої доменної області (наприклад, біомедичні зображення, супутникові зображення або картини).

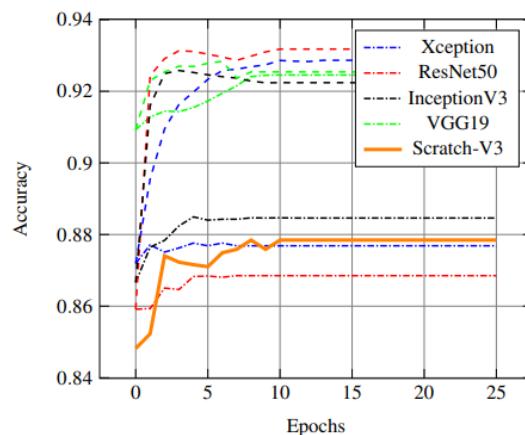


Fig. 2: Comparison between the fine tuning approach versus the off the shelf one when classifying the material of the heritage objects of the Rijksmuseum dataset. We observe how the first approach (as reported by the dashed lines) leads to significant improvements when compared to the latter one (reported by the dash-dotted lines) for three out of four neural architectures. Furthermore, we can also observe how training a DCNN from scratch leads to worse results when compared to fine-tuned architectures which have been pre-trained on ImageNet (solid orange line).

# Детекція об'єктів

Найпростіша стратегія для переходу від класифікації зображень до детекції об'єктів – це класифікація локальних ділянок на різних масштабах і в різних місцях.



Parsing at fixed scale



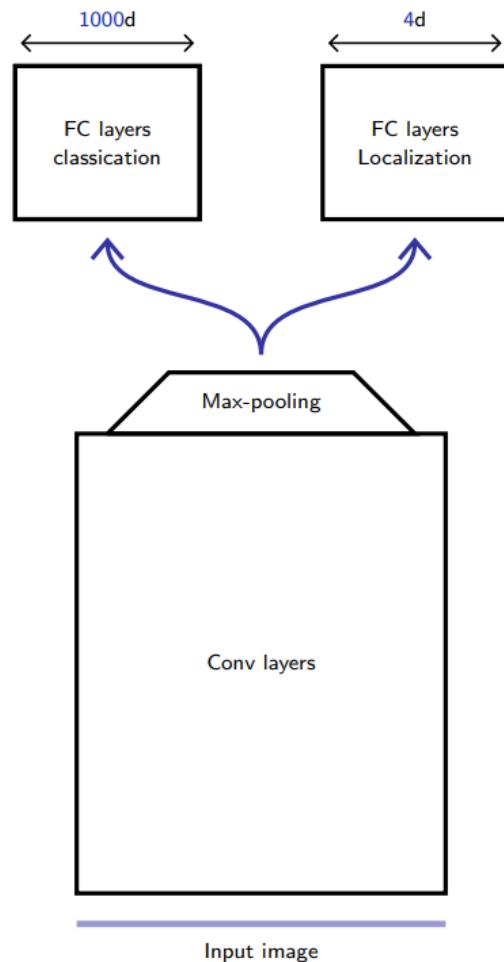
Final list of detections

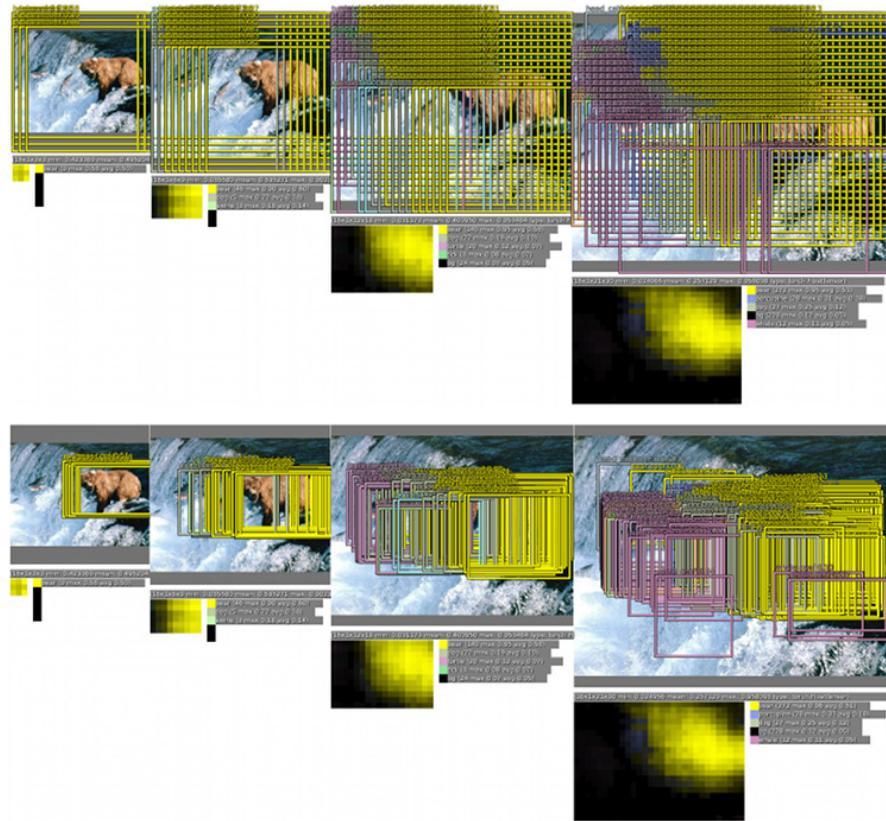
Підхід зі слайдінгом вікна оцінює класифікатор на великій кількості позицій та масштабів.

Цей підхід зазвичай є обчислювально витратним, оскільки ефективність безпосередньо залежить від роздільної здатності та кількості вікон, поданих до класифікатора (чим більше вікон, тим краще, але й тим дорожче).

# OverFeat

Складність підходу зі слайдінгом вікна була знижена в пionерській мережі OverFeat (Sermanet et al, 2013) шляхом додавання блоку регресії для передбачення обмежувальної рамки об'єкта ( $x, y, w, h$ ).





Для кожної позиції та масштабу:

- блок класифікатора виводить клас та ймовірність (верхній рис.);
- блок регресії передбачає місцезнаходження об'єкта (нижній рис.).

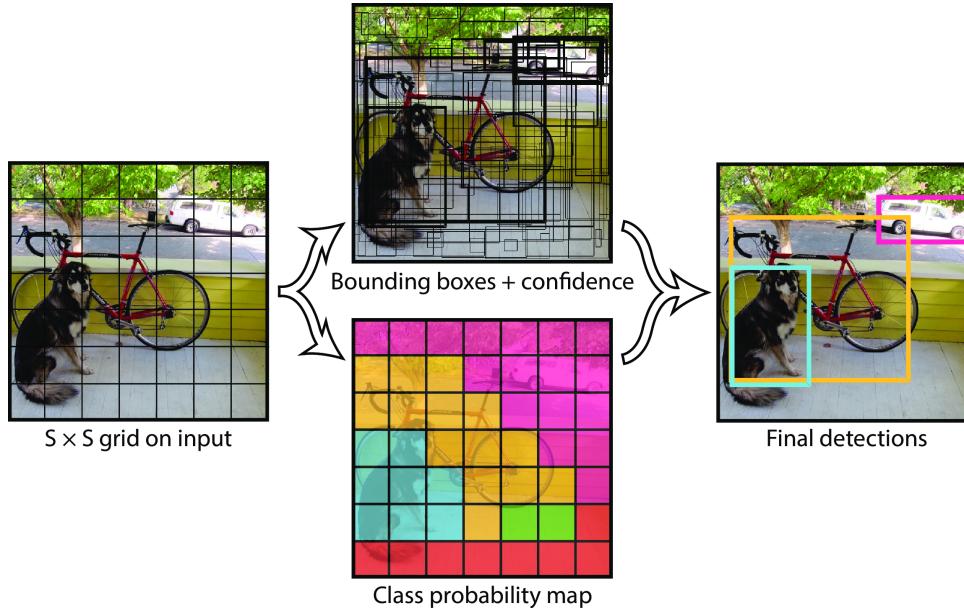


Ці прямокутні рамки в кінцевому підсумку об'єднуються за допомогою **Non-Maximum Suppression**, щоб отримати остаточні передбачення для меншої кількості об'єктів.

Архітектура OverFeat має кілька недоліків:

- це роз'єднана система (2 окремі блоки з відповідними втратами, спеціальний процес об'єднання);
- орієнтована на точне визначення місця (локалізації) об'єкта на зображенні, а не на виявленні самого об'єкта;
- не враховує загальний контекст зображення при прийнятті рішень, через це потрібно виконувати значне постоброблення для того, щоб зробити детекцію об'єктів більш узгодженою та точною.

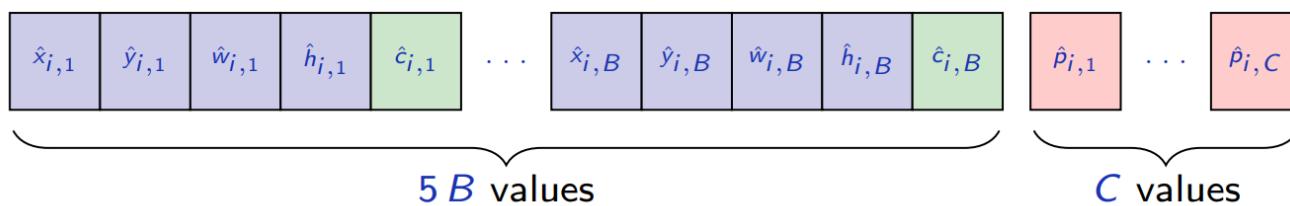
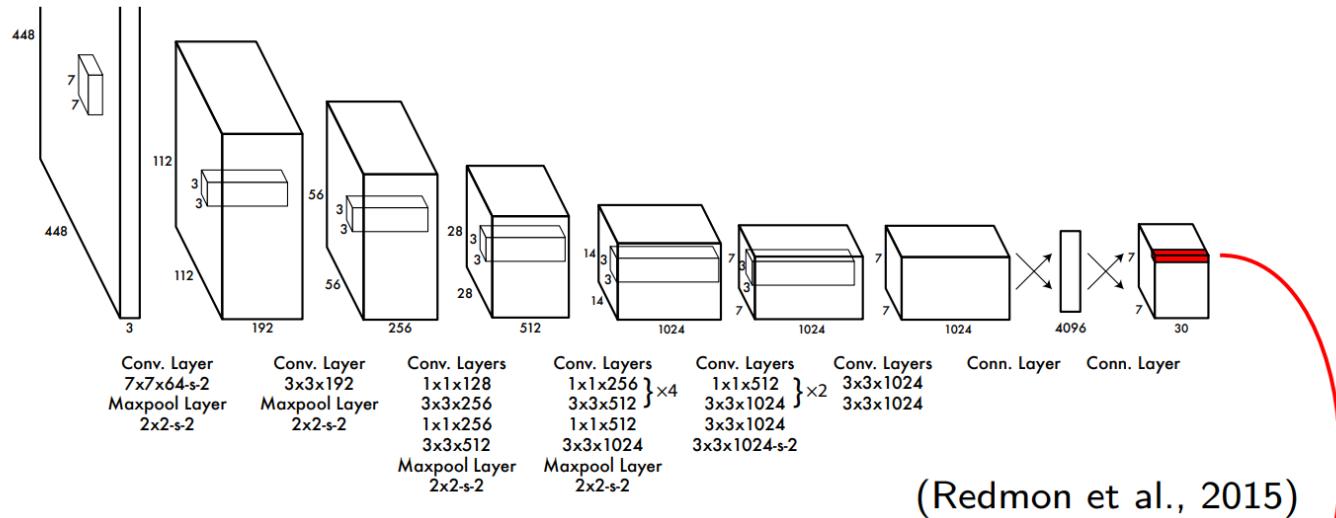
# YOLO (You Only Look Once)



YOLO (Redmon et al., 2015) розглядає задачу детекції об'єктів як задачу регресії.

Зображення розбивається на сітку розміру  $S \times S$ . Дляожної клітинки сітки передбачаються такі значення:  $B$  обмежувальні рамки, ймовірність дляожної з цих рамок та  $C$  класова ймовірність. Ці прогнози кодуються як тензор розміру  $S \times S \times (5B + C)$ .

Для  $S = 7, B = 2, C = 20$ , мережа передбачає вектор розміру 30 для кожної клітинки.



Мережа виконує передбачення оцінок класів та регресій обмежувальних рамок, і незважаючи на те, що вихід формується повнозв'язними шарами, він зберігає **2D** структуру.

- На відміну від методів ковзаючого вікна, YOLO аналізує зображення як єдине ціле.
- Модель обробляє усе зображення як на етапі навчання, так і під час тестування, тому неявно враховує контекст про класи та їхній зовнішній вигляд.

Під час навчання YOLO припускає, що в кожній клітинці сітки  $S \times S$  може бути щонайбільше один об'єкт (а точніше – центр об'єкта). Для кожного зображення визначаються: індекс клітинки  $i = 1, \dots, S \times S$ , індекс передбаченої рамки  $j = 1, \dots, B$  та індекс класу  $c = 1, \dots, C$ :

- $1_i^{\text{obj}}$  дорівнює 1, якщо в клітинці  $i$  є об'єкт, і 0 – інакше;
- $1_{i,j}^{\text{obj}}$  дорівнює 1, якщо в клітинці  $i$  є об'єкт і передбачена рамка  $j$  є тією, що найкраще його описує (має найбільше перекриття), і –0 – інакше;
- $p_{i,c}$  дорівнює 1, якщо в клітинці  $i$  є об'єкт класу  $c$ , і 0 – інакше;
- $x_i, y_i, w_i, h_i$  – координати анатованої обмежувальної рамки навколо об'єкта (визначаються тільки якщо  $1_i^{\text{obj}} = 1$ ; координати та розміри – задані відносно меж клітинки);
- $c_{i,j}$  – це IoU між передбаченою рамкою та реальним об'єктом.

Навчання починається з обчислення  $1_{i,j}^{\text{obj}}$  та  $c_{i,j}$  для зображення, після чого виконується один крок мінімізації функції втрат:

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=1}^{S \times S} \sum_{j=1}^B 1_{i,j}^{\text{obj}} \left( (x_i - \hat{x}_{i,j})^2 + (y_i - \hat{y}_{i,j})^2 + (\sqrt{w_i} - \sqrt{\hat{w}_{i,j}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_{i,j}})^2 \right) \\
 & + \lambda_{\text{obj}} \sum_{i=1}^{S \times S} \sum_{j=1}^B 1_{i,j}^{\text{obj}} (c_{i,j} - \hat{c}_{i,j})^2 + \lambda_{\text{noobj}} \sum_{i=1}^{S \times S} \sum_{j=1}^B (1 - 1_{i,j}^{\text{obj}}) \hat{c}_{i,j}^2 \\
 & + \lambda_{\text{classes}} \sum_{i=1}^{S \times S} 1_i^{\text{obj}} \sum_{c=1}^C (p_{i,c} - \hat{p}_{i,c})^2
 \end{aligned}$$

де  $\hat{p}_{i,c}$ ,  $\hat{x}_{i,j}$ ,  $\hat{y}_{i,j}$ ,  $\hat{w}_{i,j}$ ,  $\hat{h}_{i,j}$  та  $\hat{c}_{i,j}$  – це значення, які генерує нейромережа як свій вихід.

Навчання YOLO залежить від багатьох інженерних рішень, що добре ілюструє складність практичного застосування глибокого навчання:

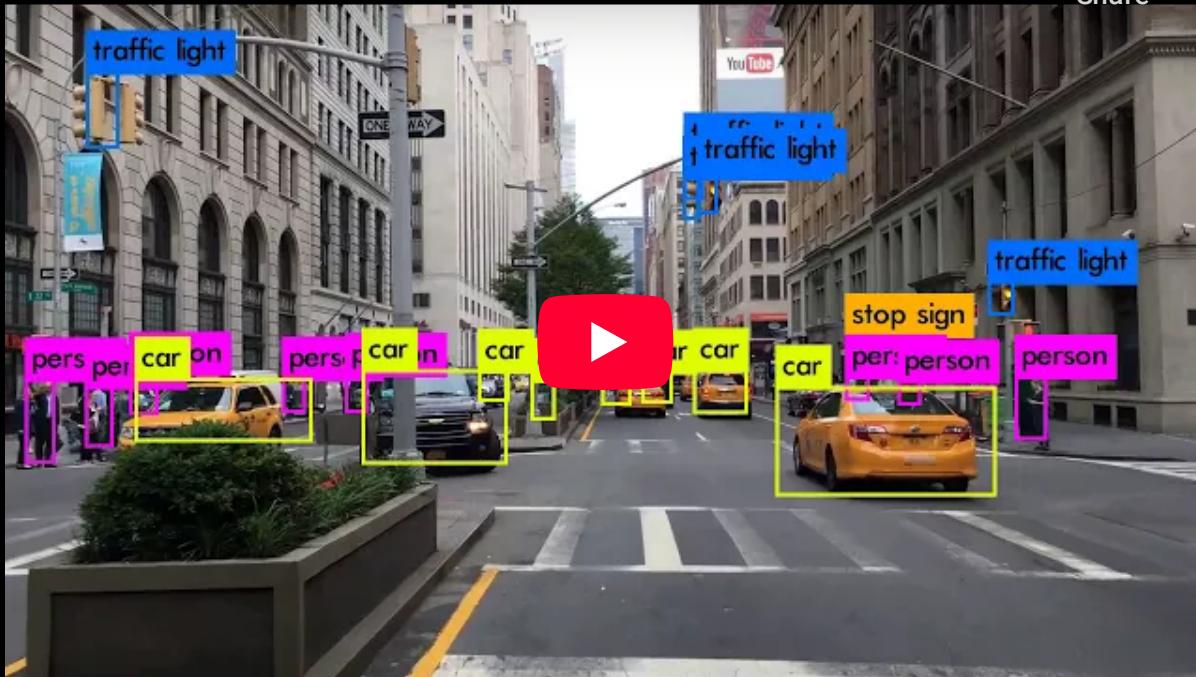
- попереднє навчання перших 20 згорткових шарів на датасеті ImageNet для задачі класифікації;
- використання вхідних зображень розміром  $448 \times 448$  для детекції замість  $224 \times 224$ ;
- застосування Leaky ReLU як функцій активації для всіх шарів;
- використання dropout після першого згорткового шару;
- нормалізація параметрів обмежувальних рамок до діапазону  $[0, 1]$ ;
- квадратична функція втрат не лише для координат прямокутників, але й для довіри (confidence) та класових оцінок;
- зменшення ваги великих обмежувальних рамок шляхом використання квадратного кореня з їх розміру у функції втрат;
- зменшення впливу порожніх комірок (які не містять об'єктів) шляхом меншого вагового коефіцієнта у функції втрат для довіри;
- аугментація даних за допомогою масштабування, трансляції та перетворень у просторі HSV.



YOLO in New York



Share



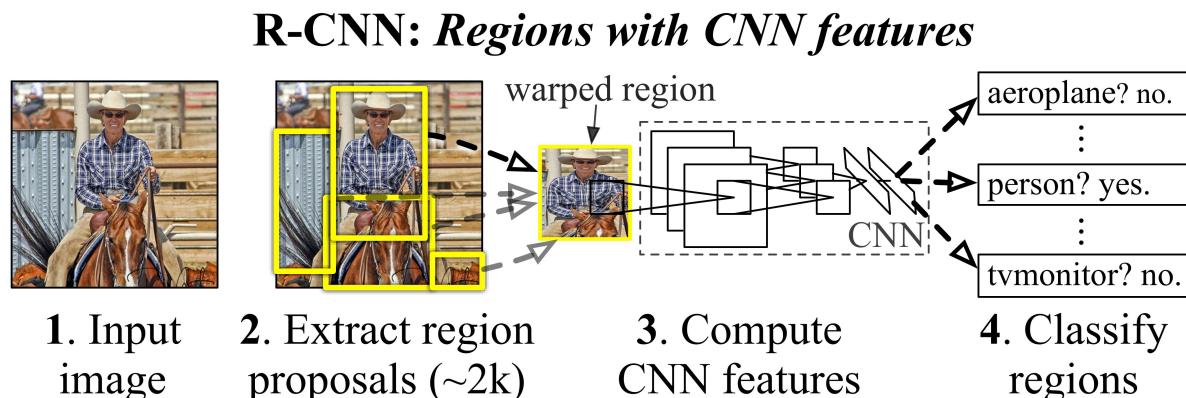
YOLO (Redmon, 2017).

# Region-based CNNs (R-CNNs)

Альтернатива великій кількості заздалегідь визначених боксів – використання **пропозицій регіонів**, які спочатку виділяються на зображенні.

Головні архітектури цього підходу – R-CNNs:

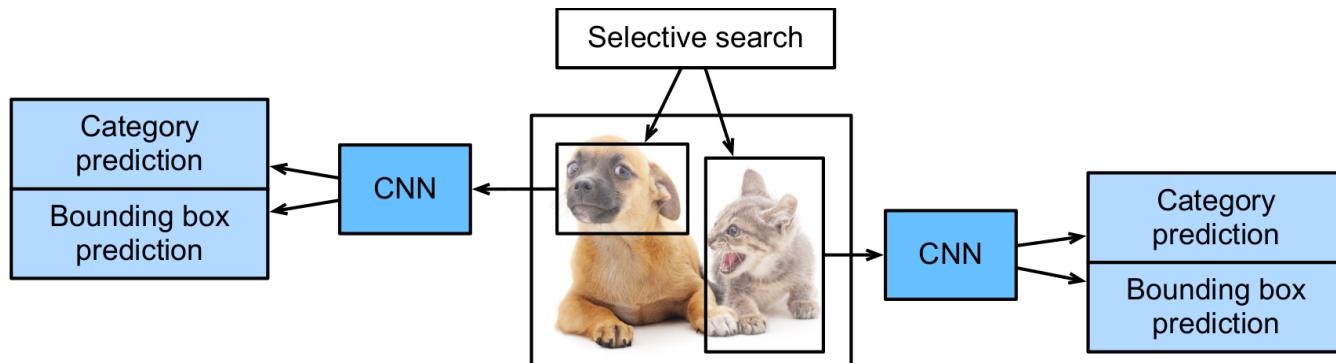
- (Slow) R-CNN (Girshick et al, 2014)
- Fast R-CNN ([Girshick et al, 2015](#))
- Faster R-CNN ([Ren et al, 2015](#))
- Mask R-CNN ([He et al, 2017](#))

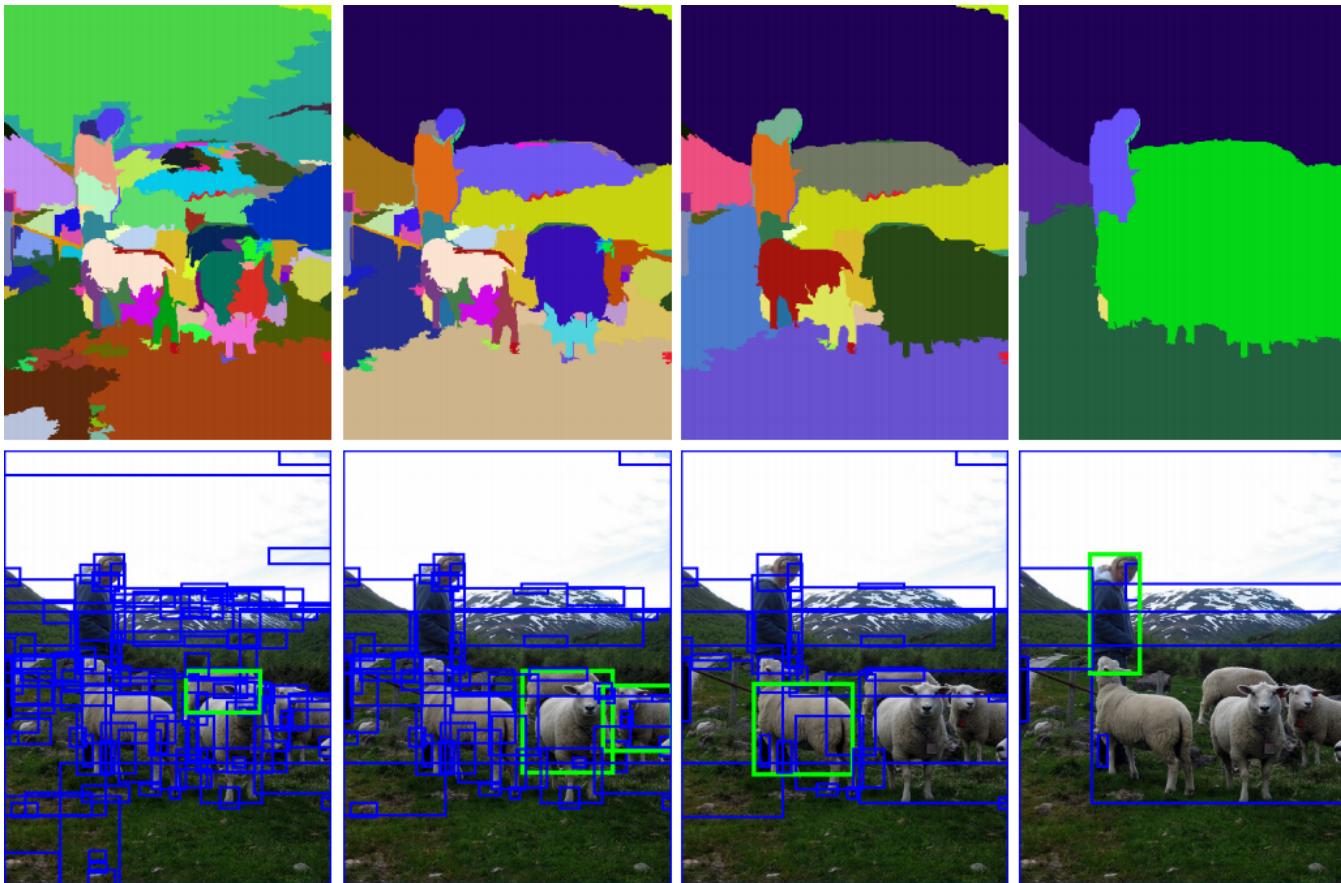


# R-CNN

Архітектура R-CNN складається з 4 частин:

1. Селективний пошук застосовується до вхідного зображення для генерації якісних регіонів-кандидатів.
2. Попередньо навчена згорткова нейронна мережа (опорна мережа, backbone) вставляється перед вихідним шаром. Вона масштабує кожен запропонований регіон до потрібного розміру та виконує прямий прохід, щоб отримати вектор ознак для кожного регіону.
3. Отримані ознаки подаються на SVM-класифікатор для визначення класу об'єкта.
4. Ті ж ознаки використовуються в лінійній регресії для прогнозування координат обмежувальної рамки (bounding box).





Селективний пошук (Uijlings et al, 2013) групує сусідні пікселі зі схожою текстурою, кольором або яскравістю, аналізуючи вікна різних розмірів на зображенні.

## Fast R-CNN

- R-CNN повільна, оскільки ознаки витягаються окремо для кожного регіону-кандидата.
- Fast R-CNN використовує все зображення як вход для CNN і витягує спільну карту ознак, замість обробки кожного регіону окремо.
- Fast R-CNN вводить ROI pooling, який дозволяє отримувати вектори ознак фіксованого розміру для регіонів-кандидатів різного розміру.

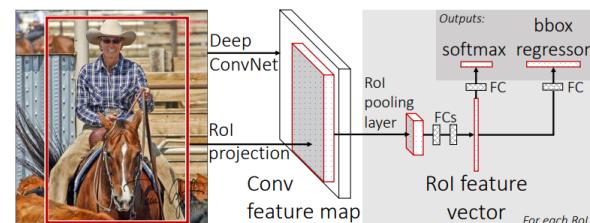
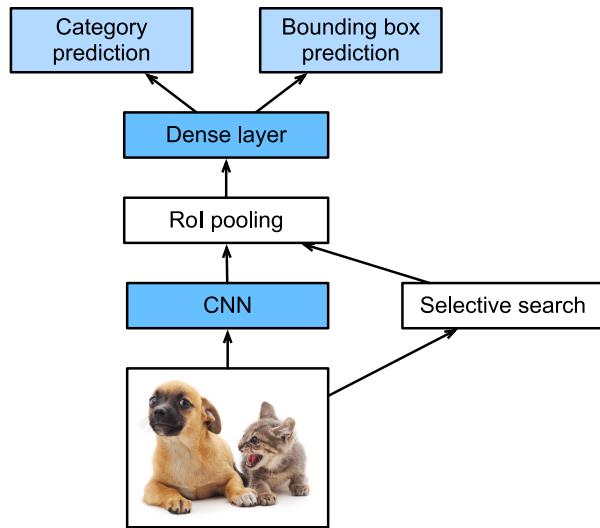
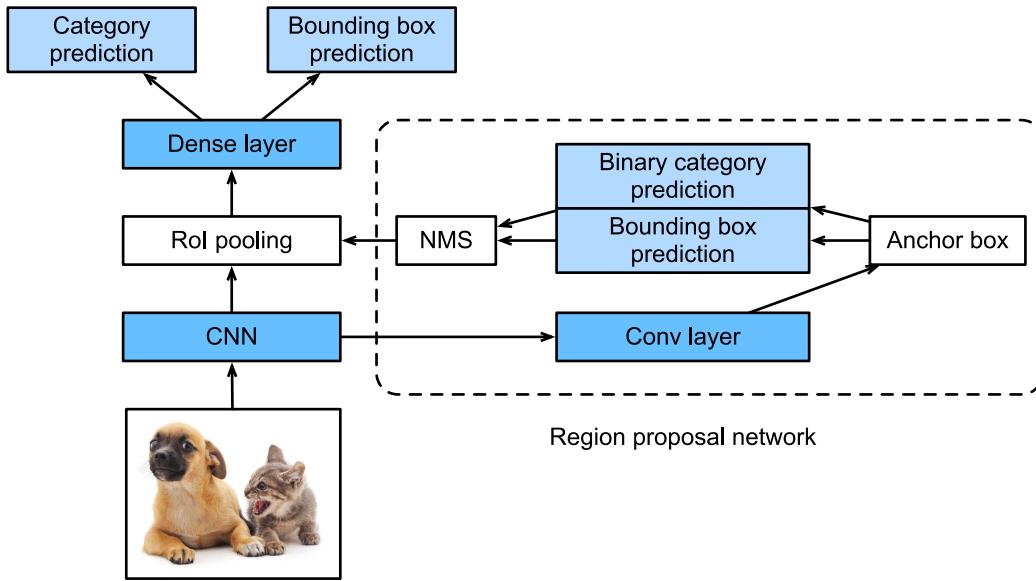


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (ROIs) are input into a fully convolutional network. Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per ROI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.



## Faster R-CNN

- Продуктивність як R-CNN, так і Fast R-CNN залежить від якості регіонів-кандидатів, отриманих за допомогою селективного пошуку.
- Faster R-CNN замінює селективний пошук на мережу пропозицій регіонів (Region Proposal Network, RPN).
- Ця мережа зменшує кількість запропонованих регіонів, зберігаючи високу точність виявлення об'єктів.



YoloV2, Yolo 9000, SSD Mobilenet, Faster RCNN NasNet...



Share



YOLO (v2) vs YOLO 9000 vs SSD vs Faster RCNN

# Основні висновки

- Одноетапні детектори (YOLO, SSD, RetinaNet тощо) – швидкі для інференсу, але зазвичай не найточніші для виявлення об'єктів.
- Двоетапні детектори (Fast R-CNN, Faster R-CNN, R-FCN, Light head R-CNN, тощо) – зазвичай повільніші, але часто є точнішими.
- Вибір мережі залежать від багатьох інженерних рішень.



- Як змінюється точність та розпізнавання об'єктів залежно від відстані до камери?
- Як модель справляється з виявленням об'єктів по одному та у скученні?
- Як трансформації зображення впливають на результати детекції?

# Сегментація



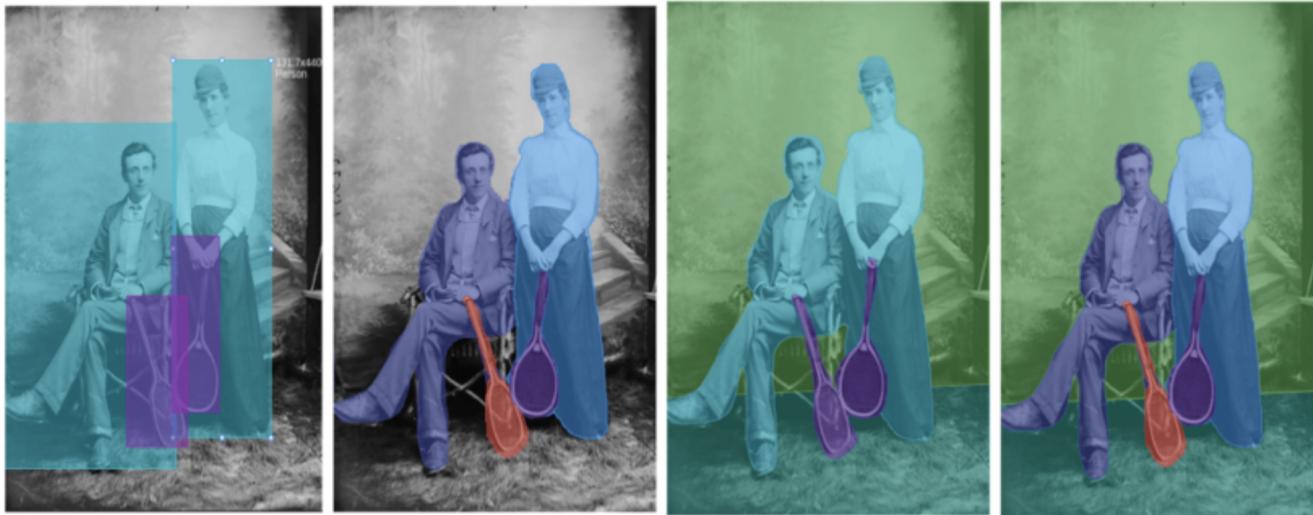
Сегментація – це розпізнавання об'єктів на зображенні на рівні пікселів:

- Семантична сегментація. Кожному пікселю на зображенні присвоєно мітку відповідно до їх класу (наприклад, автомобіль, пішохід, дорога).



Сегментація – це розпізнавання об'єктів на зображенні на рівні пікселів:

- Семантична сегментація. Кожному пікселю на зображенні присвоєно мітку відповідно до їх класу (наприклад, автомобіль, пішохід, дорога).
- Екземплярна сегментація. Крім класифікації пікселів за категоріями, розрізняються окремі об'єкти (екземпляри) одного класу, що підлягають підрахунку (наприклад, "автомобіль 1", "автомобіль 2", "пішохід 1").



Detection  
(Bounding Boxes)

Instance Segmentation

Semantic Segmentation

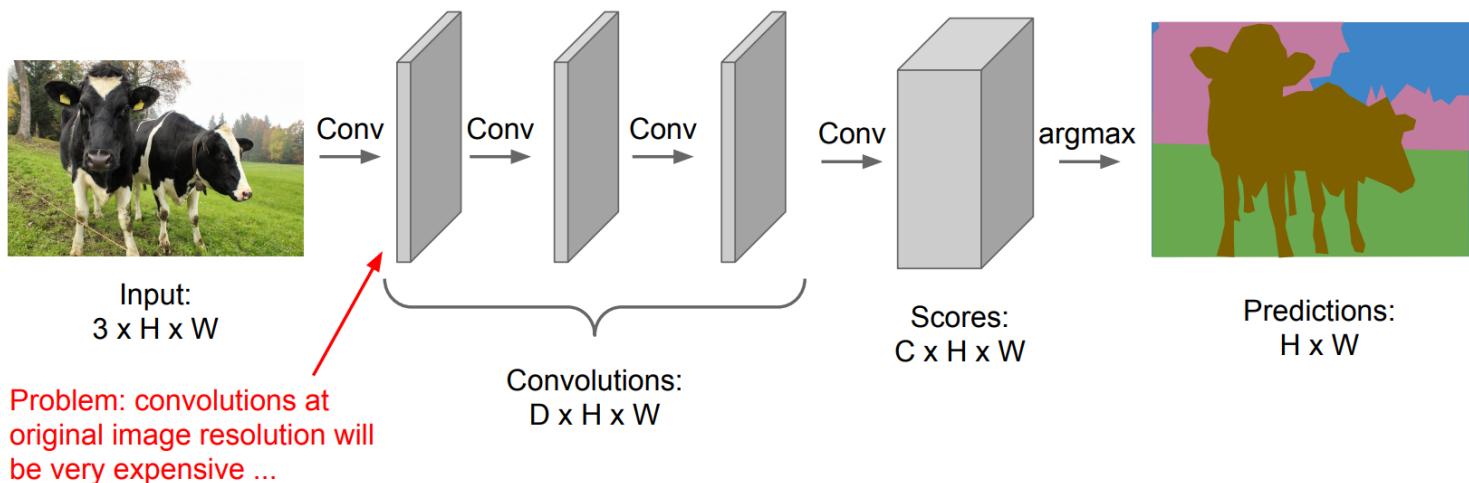
Panoptic Segmentation

Сегментація – це розпізнавання об'єктів на зображенні на рівні пікселів:

- Семантична сегментація. Кожному пікселю на зображенні присвоєно мітку відповідно до їх класу (наприклад, автомобіль, пішохід, дорога).
- Екземплярна сегментація. Крім класифікації пікселів за категоріями, розрізняються окремі об'єкти (екземпляри) одного класу, що підлягають підрахунку (наприклад, "автомобіль 1", "автомобіль 2", "пішохід 1").
- Паноптична сегментація. Поєднання семантичної та екземплярної сегментації (визначення класу для фону та об'єктів + розділення об'єктів).

Глибоке навчання розглядає семантичну сегментацію як класифікацію пікселів. Згорткові нейронні мережі можуть використовуватися для цього, але потребують певних адаптацій.

Design a network as a bunch of convolutional layers  
to make predictions for pixels all at once!

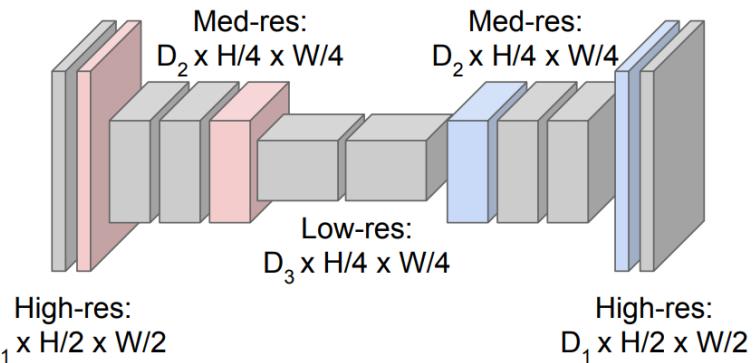


**Downsampling:**  
Pooling, strided convolution



Input:  
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with  
**downsampling** and **upsampling** inside the network!



**Upsampling:**  
???



Predictions:  
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015  
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

# Транспонована згортка

0	1
2	3

Вхід

Транспонована  
згортка

0	1
2	3

Фільтр

=

0	0	
0	0	

+

	0	1
	2	3

+

0	2	
4	6	

+

	0	3
	6	9

=

0	0	1
0	4	6
4	12	9

Вихід

$$O = (I - 1)S + F - 2P$$

# Транспонована згортка

0	1
2	3

Вхід

Транспонована  
згортка  
 $S = 2, P = 1$

0	1
2	3

Фільтр

=

0	0		
0	0		

+

		0	1
		2	3

+

0	2		
4	6		

+

	0	3	
6	9		

=

0	0	0	1
0	0	2	3
0	2	0	3
4	6	6	9

Вихід

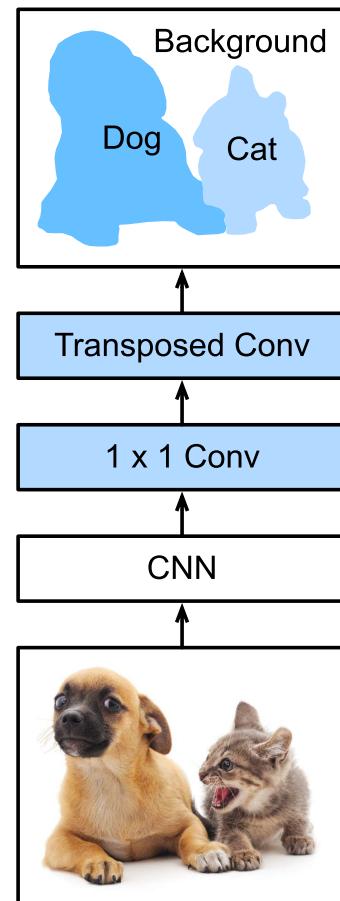
$$O = (I - 1)S + F - 2P$$

## Повністю згорткові мережі (FCNs)

Повністю згорткова мережа (FCN) – це згорткова мережа, у якій повнозв'язні шари замінено на згорткові та транспоновані згорткові шари.

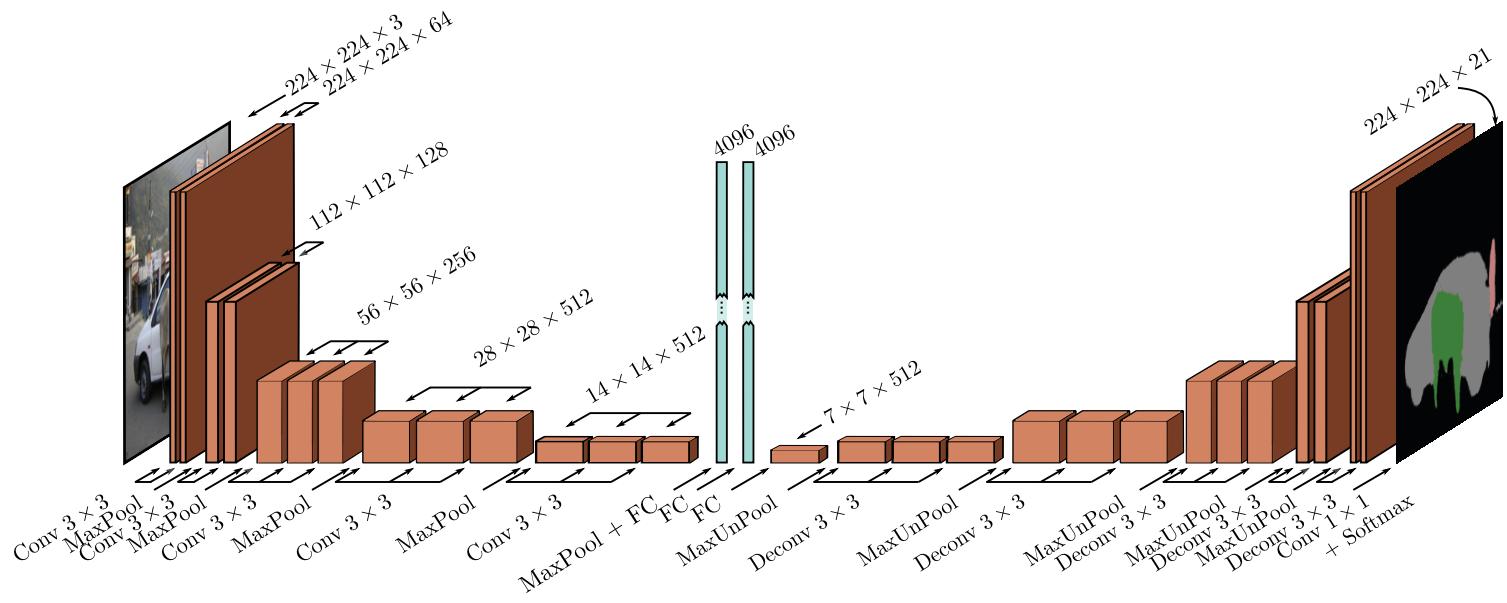
Для семантичної сегментації найпростіша архітектура FCN включає:

- використання (попередньо навченої) згорткової мережі для зменшення роздільності та вилучення ознак зображення;
- заміну повнозв'язних шарів на  $1 \times 1$  згортку для перетворення кількості каналів у кількість категорій;
- збільшення розміру карти ознак до розміру вхідного зображення за допомогою одного (або кількох) транспонованих згорткових шарів.



На відміну від повнозв'язних мереж, розміри вихідних даних повністю згорткової мережі (FCN) не є фіксованими. Вони залежать від розмірів вхідного зображення, яке може мати довільні розміри.

FCN адаптується до будь-якого розміру вхідного зображення, забезпечуючи гнучкість для семантичної сегментації.



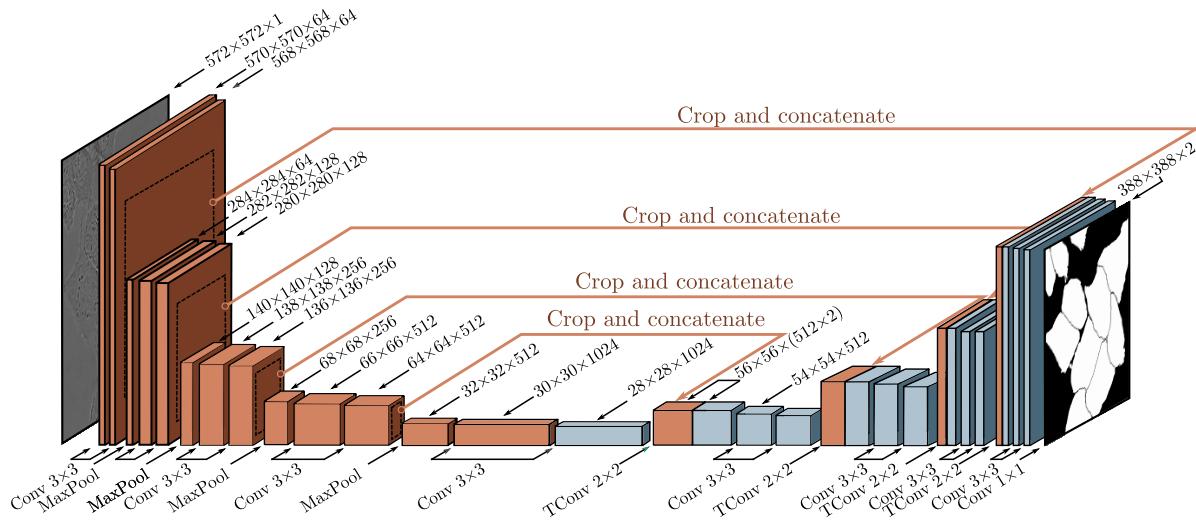
Попередня архітектура кодер-декодер є простим і ефективним способом виконання семантичної сегментації.

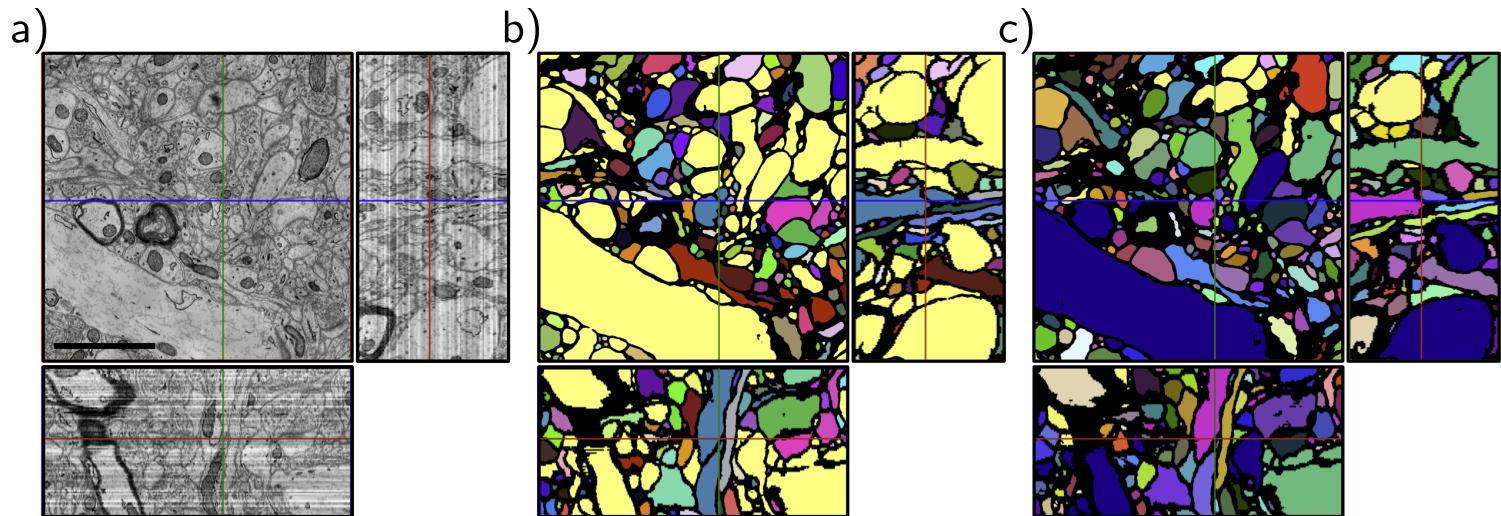
Однак представлення з низькою роздільною здатністю в середині мережі може стати вузьким місцем для продуктивності сегментації, оскільки воно має зберігати достатньо інформації для відновлення карти сегментації з високою роздільною здатністю.

# UNet

Архітектура **UNet** реалізує підхід енкодер-декодер із пропускними з'єднаннями (зазвичай конкатенаціями), які напряму з'єднують відповідні шари енкодера та декодера з однаковою просторовою роздільністю. Це дає змогу декодеру використовувати як:

- високороздільні ознаки з енкодера
- так і низькороздільні ознаки з глибших шарів мережі.





Результати 3D-сегментації з використанням архітектури UNet.

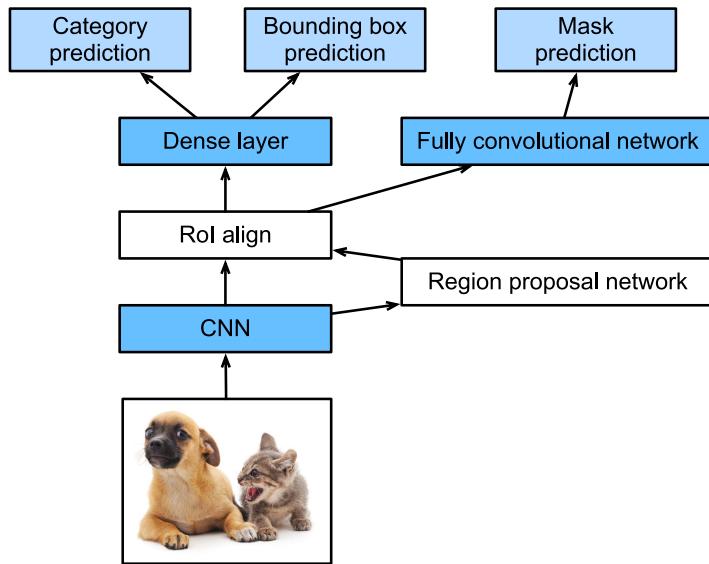
(a) Зрізи 3D-об'єму кори головного мозку миші, (b) UNet використовується для класифікації вокселів як таких, що належать або не належать до нейритів. Зв'язані області позначені різними кольорами, (c) Ансамбль із 5 моделей UNet.

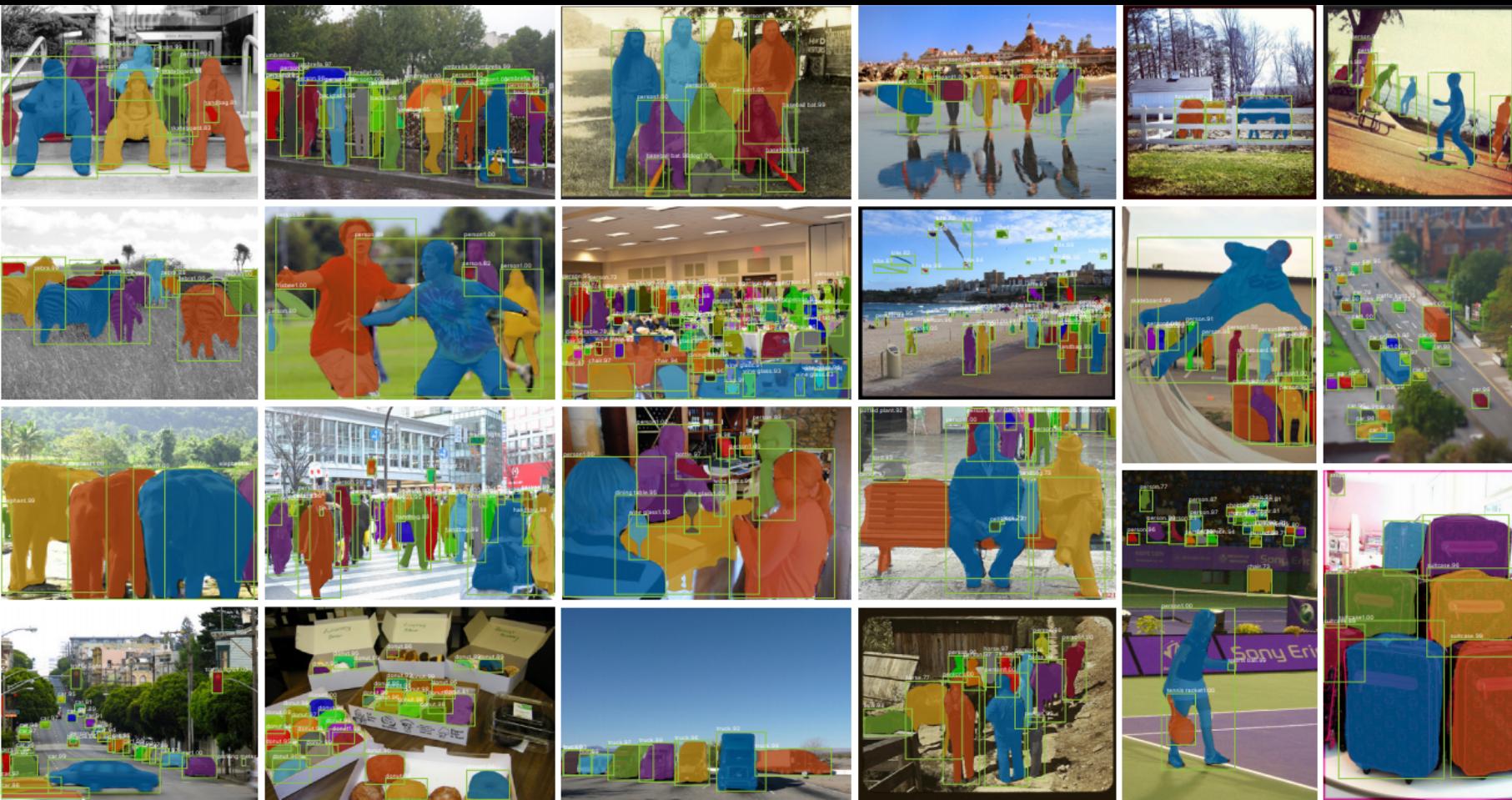


# Mask R-CNN

Задача сегментації є природним розширенням задачі детекції об'єктів. Наприклад, **Mask R-CNN** – це розширення **Faster R-CNN** для семантичної сегментації:

- Замість шару RoI Pooling використовується RoI Align, що забезпечує точніше відображення ознак.
- Додано підмережу на основі FCN для побудови бінарної маски сегментації для кожного виявленого об'єкта.
- Такий підхід дозволяє не лише знаходити об'єкти, а й точно вирізати їх на зображенні (сегментація екземплярів).







## Mask RCNN - COCO - instance segmentation



Share

Mask R-CNN  
- Object Detection  
- Segmentation



Watch on YouTube

Значну увагу слід звернути на те, що задачі детекції та сегментації значною мірою базуються на повторному використанні великих моделей, попередньо навчених для класифікації.

Як самі моделі, так і вихідні коди відповідних алгоритмів або дані, на яких вони навчались, є загальновживаними і придатними до повторного використання ресурсами.

