# OBJEKTUM-DETEKTÁLÁS

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

# Osztályozás+lokalizálás

- Képfeldolgozási alkalmazásról lesz szó, mint a konvolúciós neuronhálóknál
- Ott feltettük, hogy minden képhez egyetlen címke tartozik
  - Azaz a képen egyetlen objektum látható, és az az egész képet betölti
  - Ezért a feladatot osztályozási feladatként tudtuk megfogalmazni
- Ha egyetlen objektumunk van, de nem tölti be a képet, akkor egy osztályozási és egy lokalizálási feladatot kell egyszerre megoldanunk
  - Meg kell találni az objektum címkéjét is pontos pozícióját is

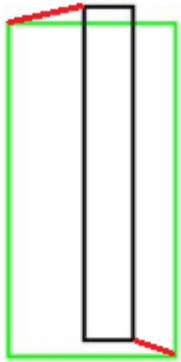
Osztályozás


Osztályozás + lokalizálás

# Osztályozás+lokalizálás

- Az objektum pozícióját általában egy befoglaló téglalappal adjuk meg
- De bizonyos esetekben pixel-pontos meghatározásra is szükség lehet
  - Ezt szegmentálási feladatnak hívják, meghatározandó az objektum maszkja

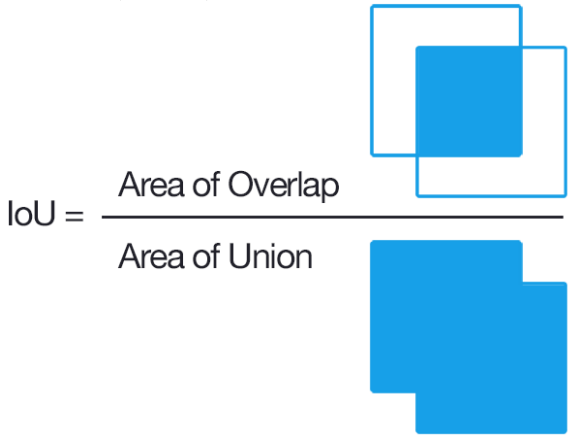
- Az osztályozás+lokalizálás egyidejűleg megoldható egy multi-taszk hálóval
- Az egyik feladat az objektum felismerése (osztályozási feladat)
- A másik feladat a befoglaló téglalap pontos helyének megtalálása
  - Ez megadható pl. a bal felső és a jobb alsó sarok 2-2 koordinátájával
  - A hiba mérhető a valódi és a tippelt téglalap koordinátáinak négyzetes eltérésével (a képen a piros szakaszok összhossza)
  - A hiba egy valós szám, tehát a feladat regresszió

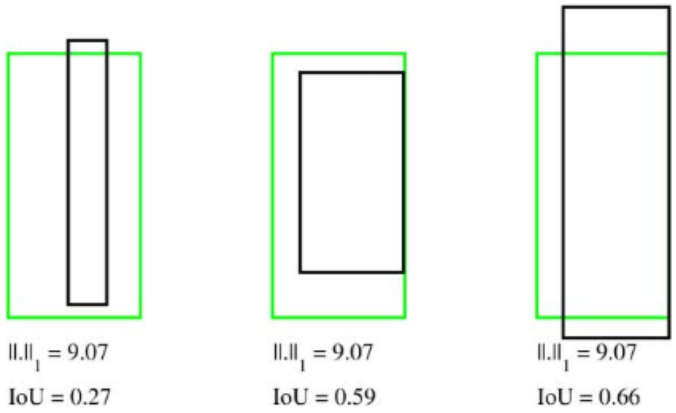


- A multi-taszk feladat össz-hibája az osztályozási és a regressziós feladat hibájának súlyozott összegeként definiálható:
  - Hiba = osztályozási_hiba + $\alpha \cdot$ regressziós_hiba

# Osztályozás+lokalizálás kiértékelése

- Az osztálycímke diszkrét érték, vagy eltaláltuk, vagy nem
- A befoglaló téglalap valóditól való eltérése viszont folytonos érték
- A pontossága mérésére az intersection over union (IoU) értéket szokás használni
- Általában akkor tekintjük helyesnek a detektálást, ha a címke helyes, és IoU≥0,5
- Megjegyzés: a koordináták négyzetes hibája és az IoU nem pontosan ugyanazt méri, pl:

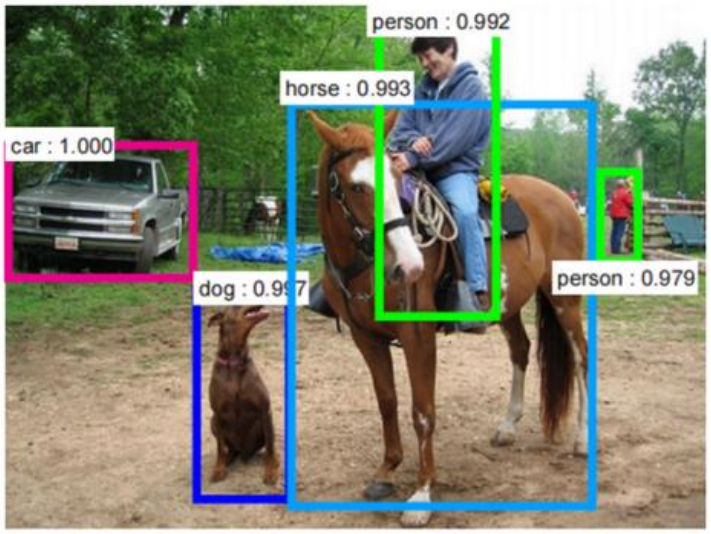



- Az IoU precízebb, pl. nem függ az objektum méretétől
- Viszont a tanításhoz nem jó, mert ha az átfedés 0, akkor mindig 0 lesz, és a derivált is 0

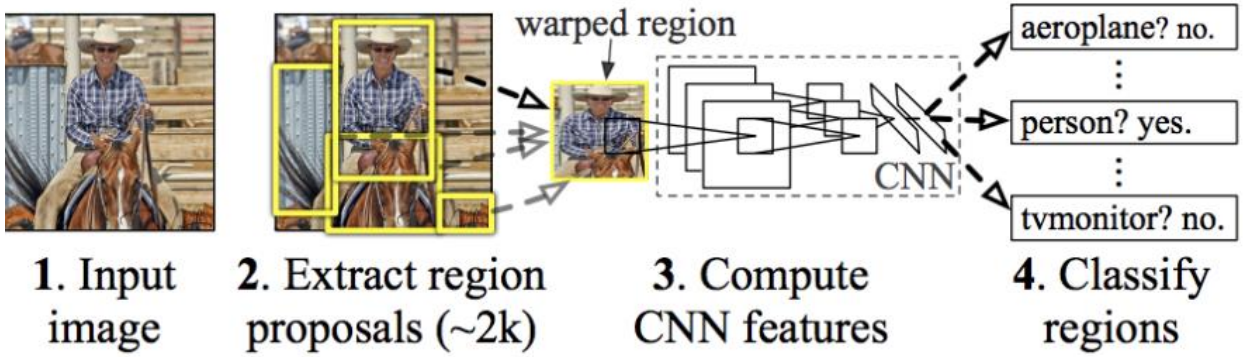
# Objektum-detektálás

- Nagyon sok gyakorlati alkalmazásban nem csak egyetlen objektum lehet a képen, hanem több is (pl. önvezető autók)
- Ilyenkor a kimenet nem egyetlen címke, hanem:
  - Meg kell találni a képen levő összes objektum pozícióját
  - És azonosítani kell őket
- Párhuzamosan kell elvégezni az objektumok osztályozását és detektálását
- Probléma: Az összes lehetséges pozíciót és ablakméretet végig kellene próbálni → túl sok lehetőség
- Ráadásul sok alkalmazásban ennek valós időben kellene működnie
- És lehetőleg szerényebb képességű célhardveren, nem szuper képességű szerveren (pl. önvezető autók)



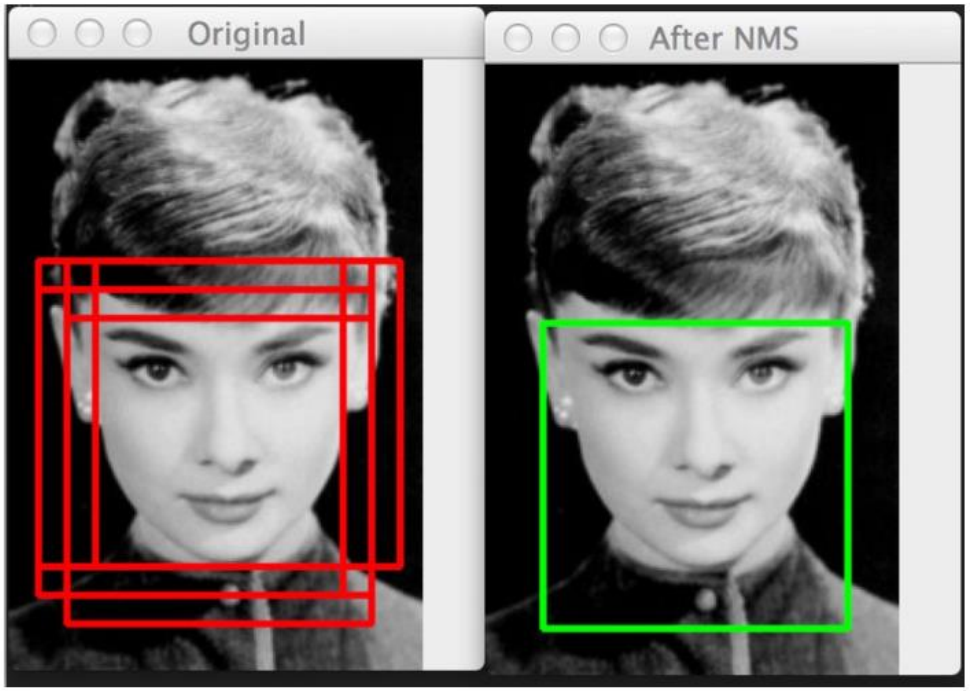
# R-CNN

- A régiójavaslaton (region proposal) alapuló módszerek esetén a lehetséges detekciós ablakok (régiók) számát valamilyen algoritmussal leszűkítjük.
- A korai módszereknél a kiértékelésre javasolt régiókat egy külső, neuronhálótól független algoritmus szállítja
- Az R-CNN (2014) esetén ez egy hierarchikus képszegmentáló-klaszterező algoritmus, amely képeként kb. 2000 régiójavaslatot állít elő
- A javasolt régiókat egységes méretűre/arányúra konvertáljuk
- Ezeket egy CNN segítségével feldolgozzuk (jellemzőkinyerés)
- Az osztályozást egy lineáris SVM-mel végezzük
- A régiók utólagos pontosítása is lehetséges (ld. lokalizációs feladat)



# Non-Maximum Suppression

- A detektált régiók között nagyon nagy átfedés lehetséges. Utólagosan meg lehet szűrni a régiókat a non-maximum suppression módszerrel. Ennek lényege, hogy a bizonyos küszöbértéknél magasabb arányban átfedő régiók közül csak a legnagyobb osztályozási valószínűséget adót tartjuk meg.




https://www.pyimagesearch.com/wp-content/uploads/2014/10/hog_object_detection_nms.jpg

# Fast R-CNN

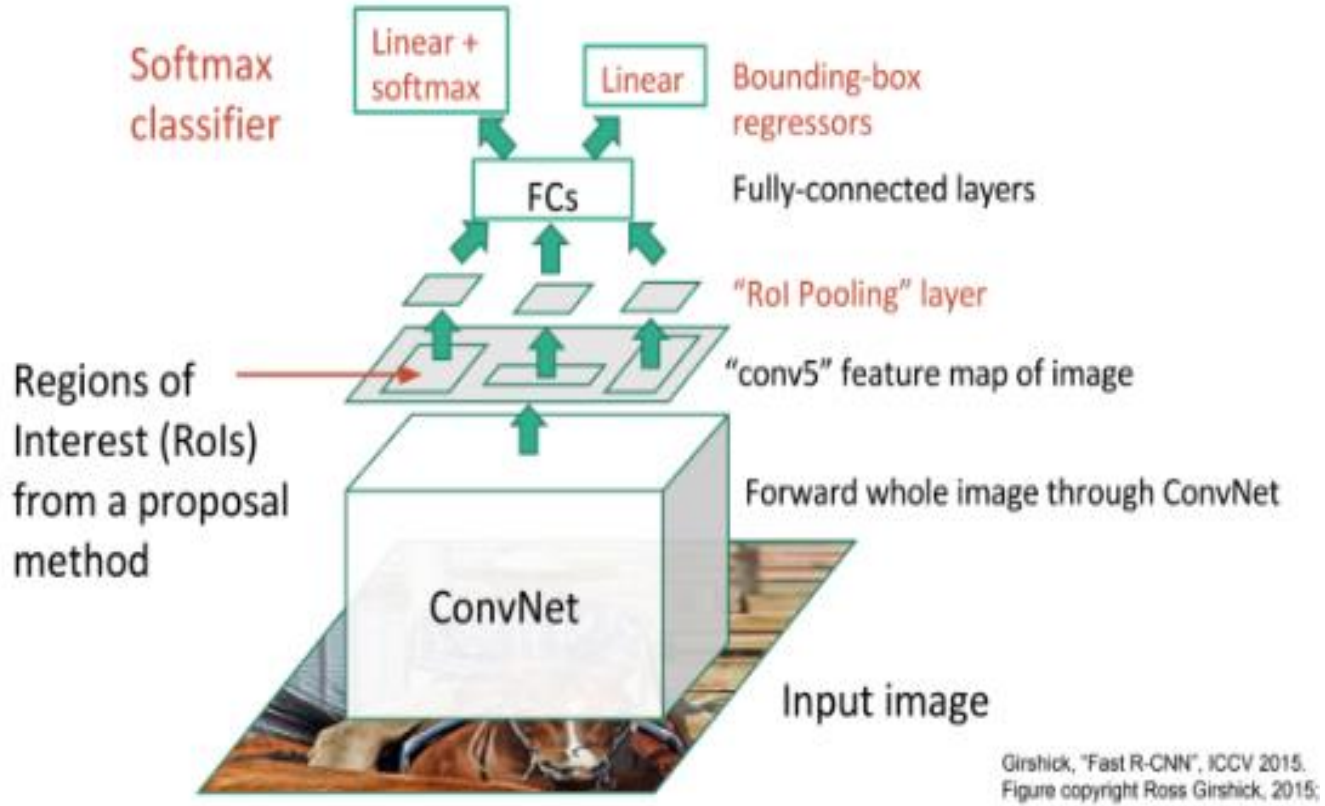
- Az R-CNN volt az egyik legelső algoritmus, amelyik neuronhálót használt objektum-detektálásra, és nyilván sok gyenge pontja van:
- 3 különböző tanulóalgoritmusból áll össze, ezek külön-külön vannak tanítva: CNN (jellemzőkinyerés), SVM (osztályozás), ANN ( regresszió)
- Egy negyedik algoritmust használ a régiók megtalálására
- Lassú, mert egyetlen képen 2000-szer kell lefuttatni a CNN jellemzőkinyerést (minden javasolt régión külön-külön)
- A Fast R-CNN modellben két módosítást javasoltak. Az egyik a három tanuló egyesítése egyetlen tanulómodellé. Ehhez csak az SVM-et kellett lecserélni egy fully connected osztályozó rétegre, valamint a három részmodellt egyben tanítani a korábban látott multi-taszk módon
- A másik módosítás célja az volt, hogy ne kelljen a CNN részt minden régióra külön-külön kiszámolni

# Region of interest pooling

- Emlékezzünk, hogy a CNN eleve lokalizáltan dolgozza fel a kép különböző részeit.
- Ezért nem a kivágott régiókon futtatjuk le a teljes CNN-t (külön-külön)
- Hanem a CNN-t ráeresztjük a *teljes* képre *egyszer*
- Majd a kapott konvolúciós jellemzőkből vágjuk ki az egyes régiókat leíró részeket
- Mivel a régiók nem egyforma méretűek, valahogy egységes méretűre kell őket hozni az osztályozó és a lokalizáló számára
- Ezt végzi el a „region of interest pooling”
  - A túl nagy régiókat egységnyi részekre bontjuk, és a részekre belül pooling-gal egyesítünk
  - A túl kicsi régiókat felnagyítjuk

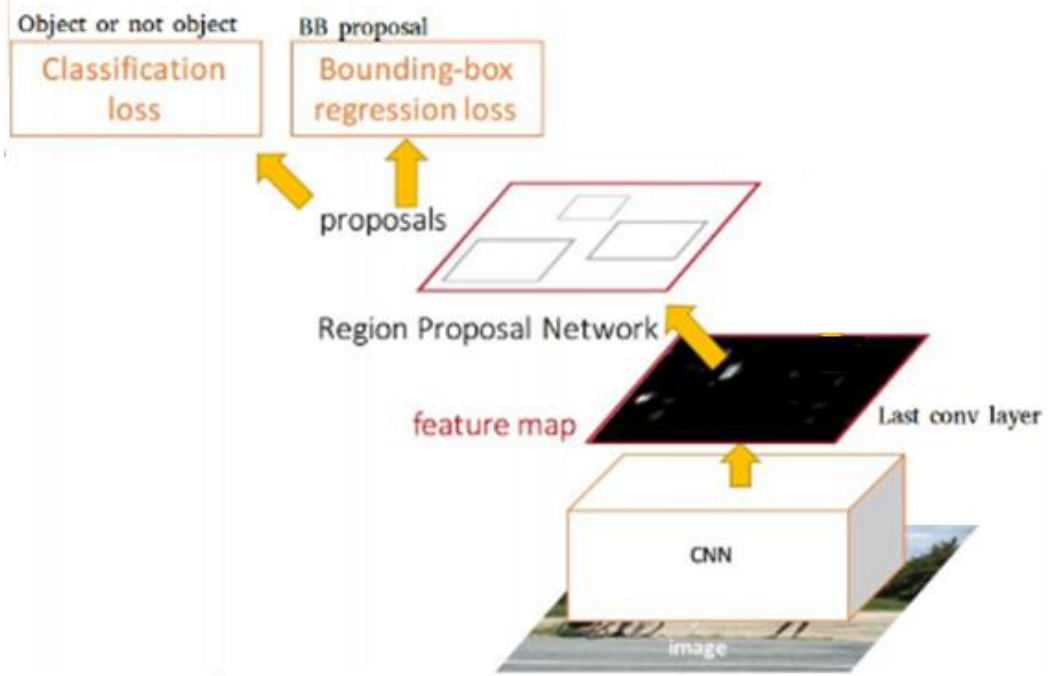
# Fast R-CNN

- Összefoglaló ábra:



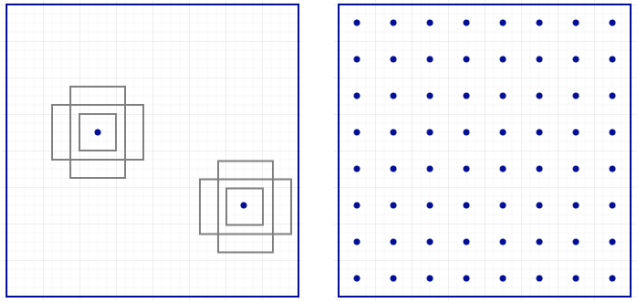
# Faster R-CNN

- A fast R-CNN annyival gyorsabb az R-CNN-nél, hogy a futási időt itt már a régiójavasló algoritmus dominálja
- A faster R-CNN (2016) megpróbálja a régiójavaslatokat is egy neuronháló segítségével előállítani (Region Proposal Network)
- Ehhez a konvolúciós rétegek kimenetét használja inputként, nem az eredeti képet
- Az RPN két dolgot becsül:
  - A régió objektum-e
  - Ha igen, mik a pontos koordinátái



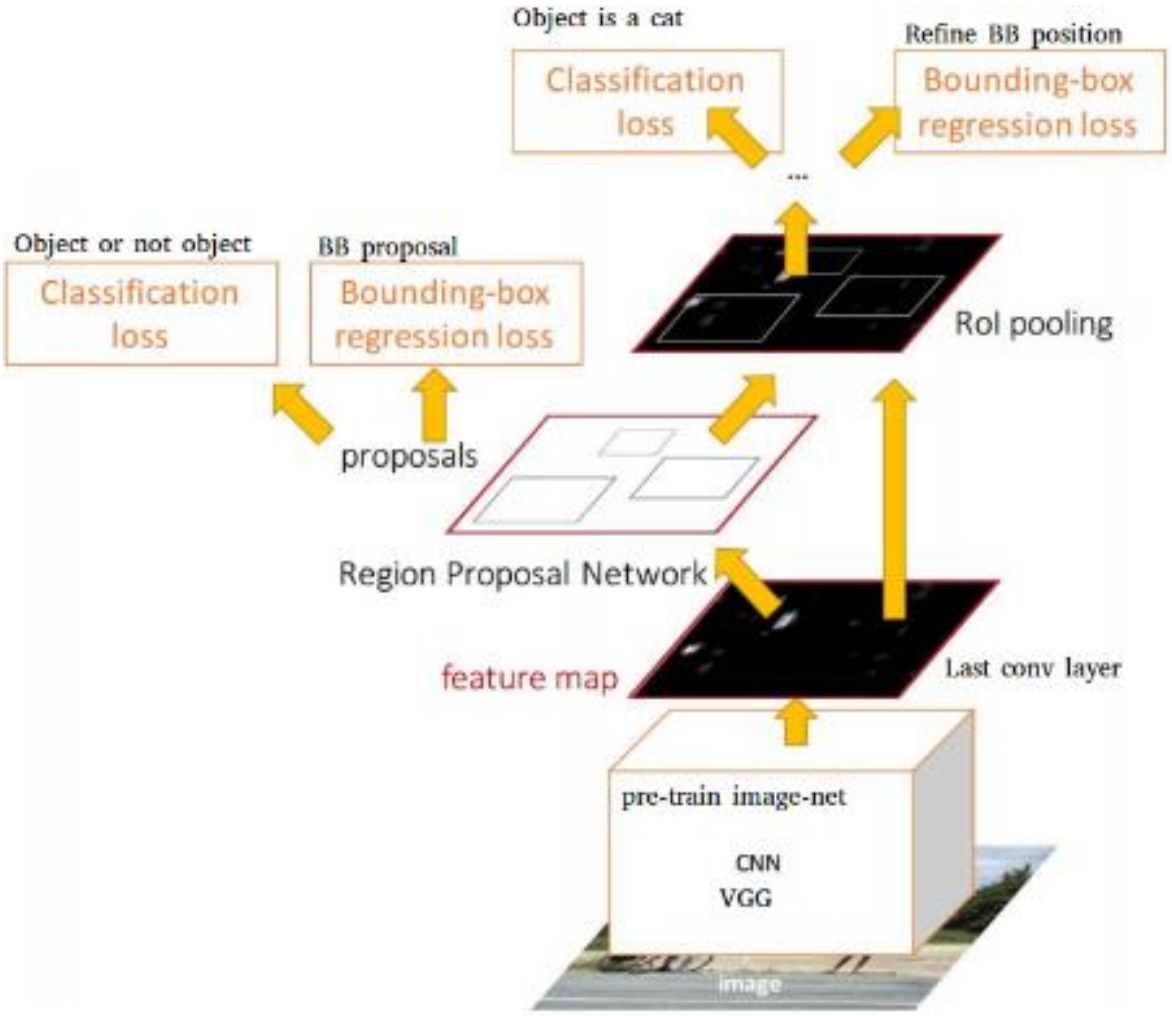
# Az RPN működése

- Az RPN nem vizsgál meg minden lehetséges régiót, hanem leszűkíti a keresést bizonyos „anchor box”-okra
  - Az eredeti cikkben csak 9 lehetséges ablak-alakot vizsgál
  - És ezeket 16-16 pixelenként helyezi el
  - Ezzel együtt is tízezres nagyságrendű lehetséges régió marad



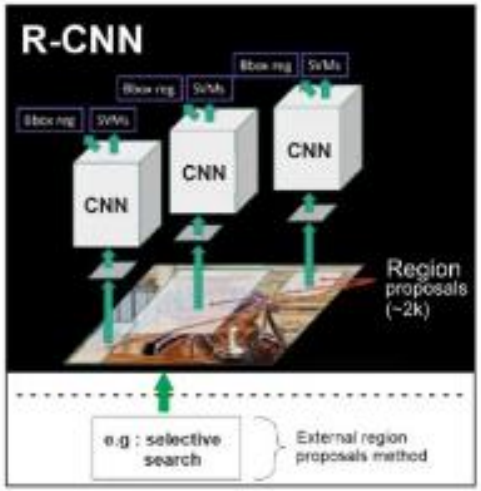
- Az RPN minden egyes anchor box-ra megmondja, hogy
  - Van/nincs benne objektum (2-osztályos osztályozás)
  - Ha van benne objektum, pontosítja a doboz koordinátáit (regresszió)
- Az objektum-tartalmazási valószínűség szerint ki lehet válogatni a legjobb régiókat, illetve non-maximum suppression-nel is szűrhetők
- A megmaradt régiók mennek tovább osztályozásra+regresszióra (a háló főágán)

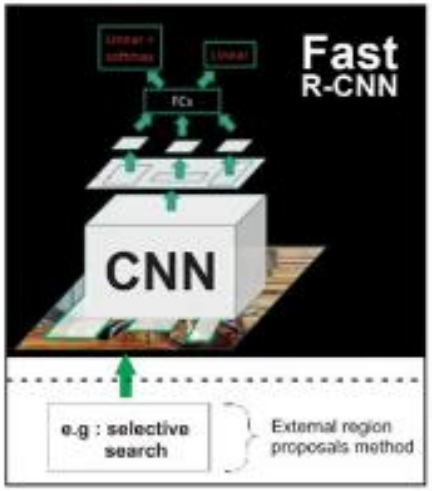
# Faster R-CNN - összegzés

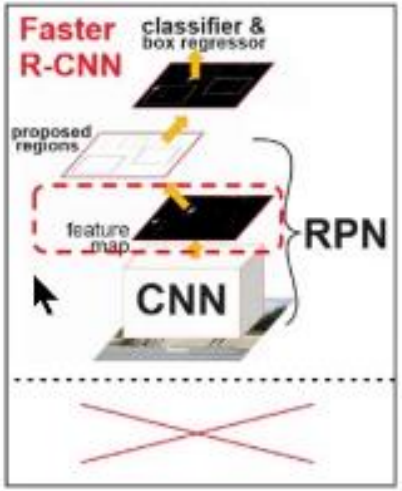


# Összehasonlítás

- Az R-CNN különböző változatainak sebessége és pontossága



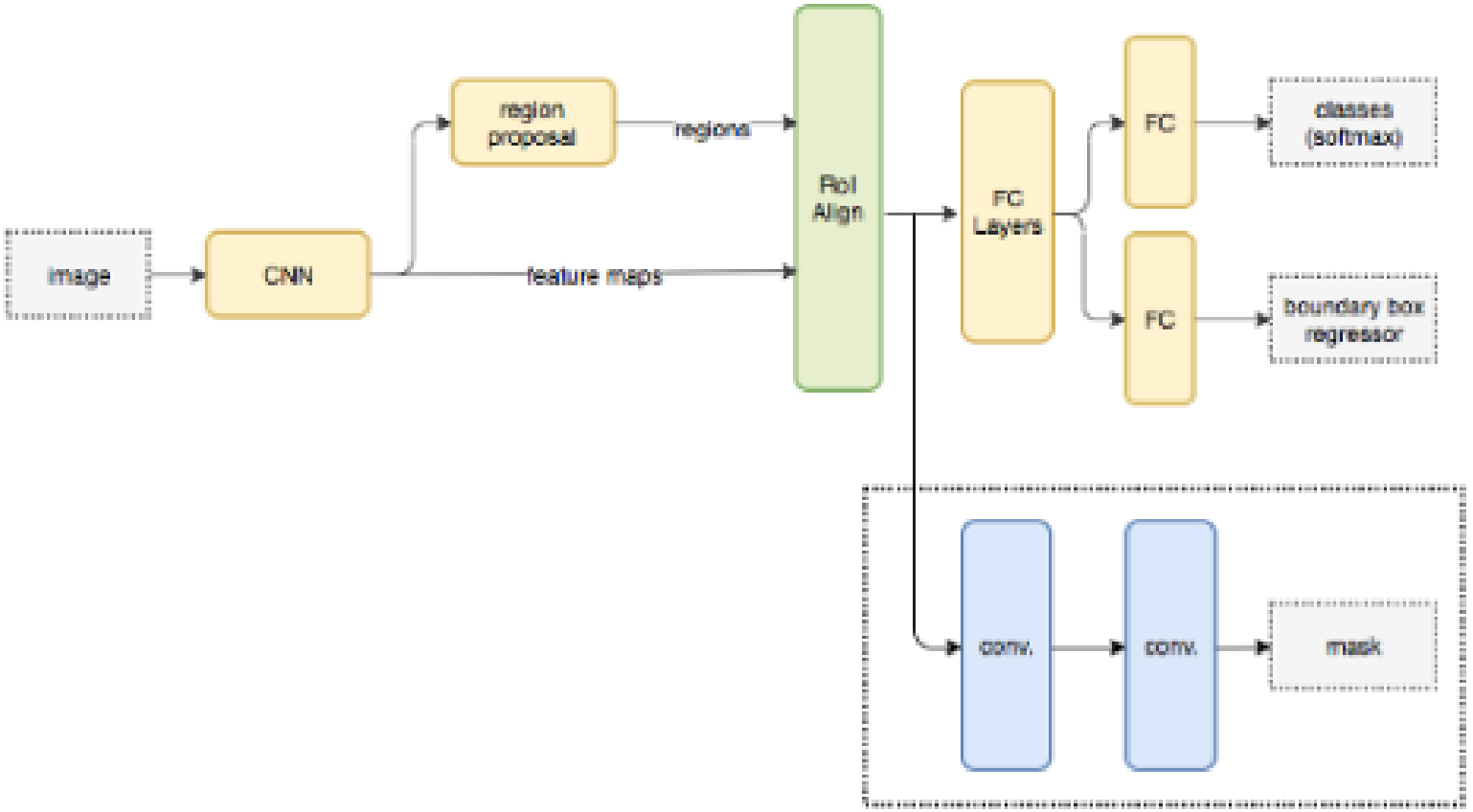




| | R-CNN | Fast R-CNN | Faster R-CNN |
|---|---|---|---|
| Test time per image | 50 seconds | 2 seconds | 0.2 seconds |
| Speed-up | 1x | 25x | 250x |
| mAP (VOC 2007) | 66.0% | 66.9% | 66.9% |

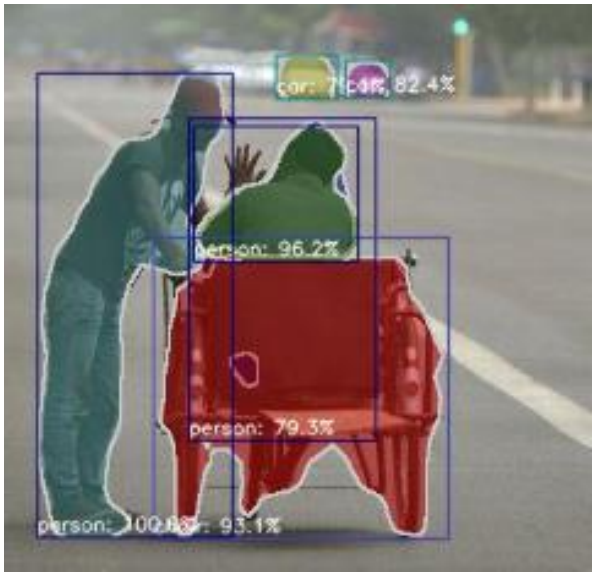
* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy

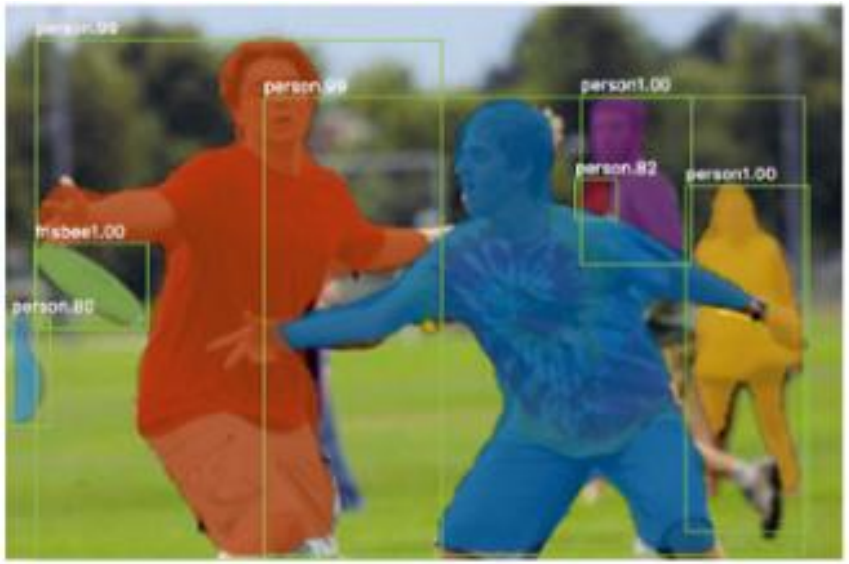
# Mask R-CNN

- A faster R-CNN modellt kiegészíthetjük egy újabb ággal, ami egyúttal a maszkokat is megjósolja



# Mask R-CNN példa



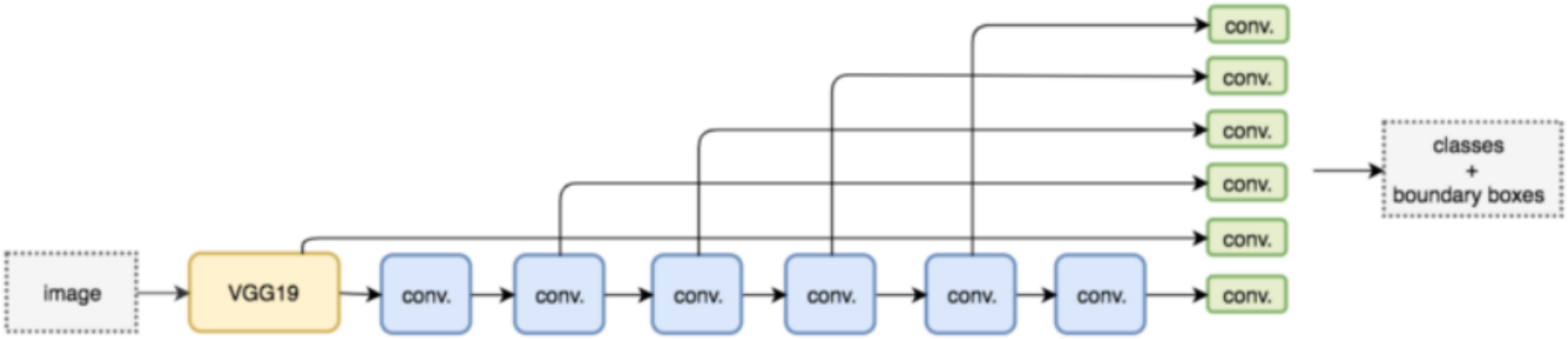

- Alkalmazás: pl. testtartás felismerése (pose recognition)

# Single-shot detektorok

- A faster R-CNN esetén az RPN ág és a főág eléggé hasonló feladatot végeznek. Nem lehetne ezeket összevonni?
- A single-shot detektorok kihagyják a régiójavaslatok készítését, az osztályozási+regressziós feladatot közvetlenül, egy lépésben („single-shot”) próbálják megoldani.
- Ezen módszerek közül a két legismertebb a YOLO és az SSD.
- Az R-CNN „anchor”-jaihoz hasonló régiókból indulnak ki.
- Egy konvolúciós háló által adott konvolúciós jellemzőkön dolgoznak.
- További konvolúciós rétegekkel próbálják az osztályozási+ regressziós feladatot egyszerre megoldani

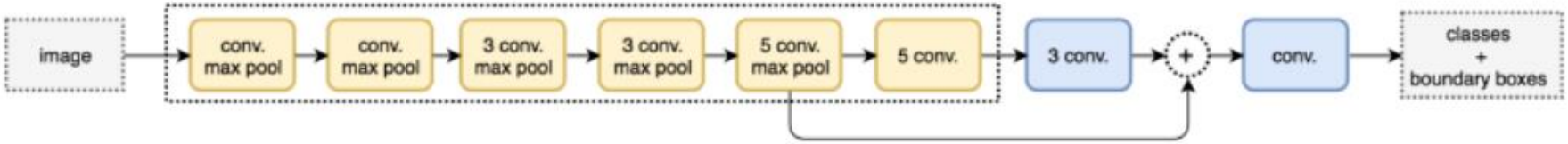
# Single-shot detector (SSD)

- A kevés vizsgált ablakméret miatt hajlamosak a túl nagy vagy túl kicsi objektumokat nem észrevenni.
- Az SSD ezért a (különböző felbontású!) konvolúciós rétegek mindegyikének kimenetén elvégzi a detekciót

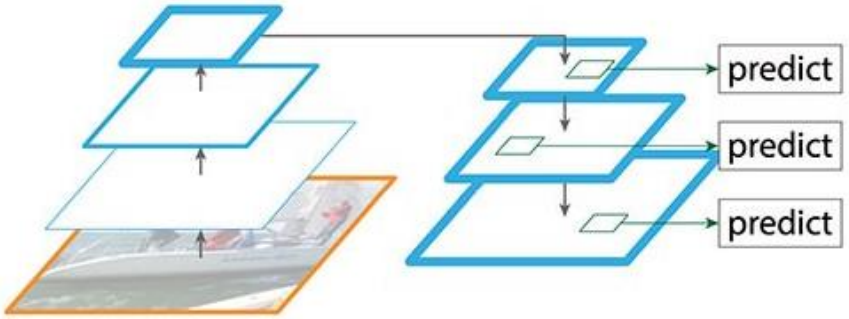


# YOLO („you look only once”)

- A YOLO módszer pedig a konvolúciós feature map-ek össze-konkatenálásával állít elő inputot a detekció számára
- Így csak egyszer kell elvégezni a detekciót, de az eredmény pontatlanabb lesz

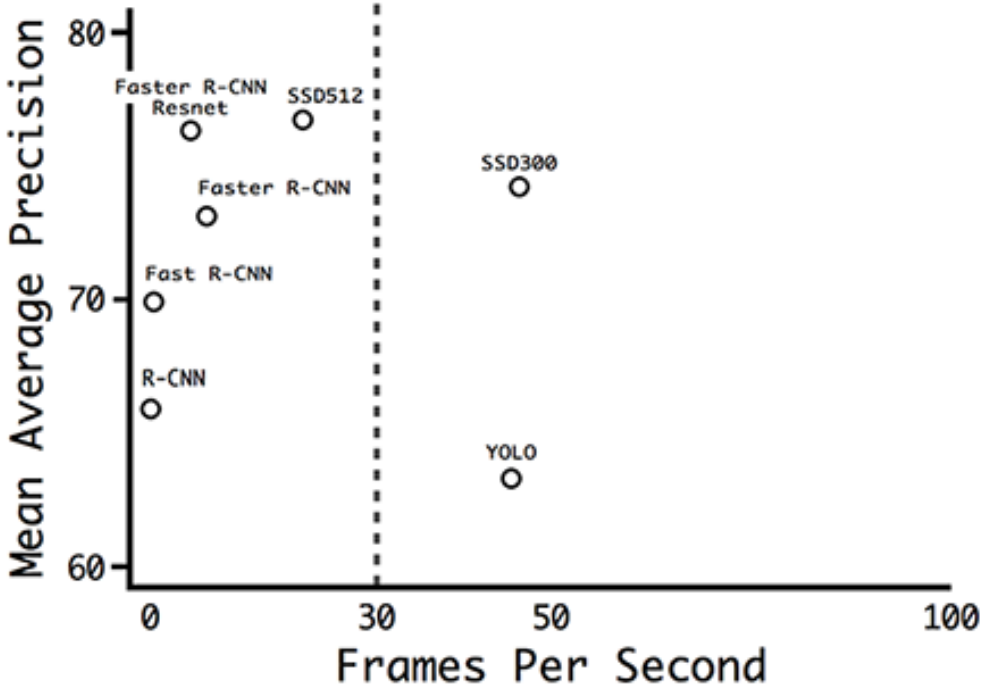



- Megjegyzés: az újabb, „feature-pyramid” alapú módszerek is igyekeznek a konvolúciós rétegek mindegyikét használni, a skála-függetlenség növelése céljából



# Sebesség vs. pontosság

- A YOLO gyors, de pontatlan,
- az R-CNN változatai lassabbak, de pontosabbak
- az SSD mindkettőben jó




(SSD512 és SSD300 esetén a szám a képek felbontását jelöli)

# KÖSZÖNÖM A FIGYELMET!

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.