**Detekcia áut v obraze s použitím NN**

V dnešnej dobe stále narastá počet a rôzne odlišné typy oblastí, v ktorých môžeme použiť systémy na detekciu/rozpoznávanie objektov. V oblastiach automatizácie, kde stroj prichádza do kontaktu s človekom sme však konzervatívnejší. Dbáme na bezpečnosť a tým spomaľujeme nasadenie v bežných aplikáciách. Na druhej strane to núti ľudí stále viac a viac inovovať a ošetrovať používané systémy, prípadne vymýšľať nové tak, aby splňovali na ne kladené kritéria. Typickým príkladom je nasadenie systému rozpoznávania vozidiel a chodcov pre aplikácia autonómnych vozidiel, kde môžu výrazne znížiť počet nehôd a smrteľných úrazov.

V minulosti sa na aplikácie zautomatizovania dopravy používalo veľké množstvo iných senzorov (lasery GPS, radar, inteligentné mapy). Tie však samé o sebe mali svoje osobité obmedzenia. Metódy počítačového videnia majú potenciál nahradiť tieto senzory a zároveň zvýšiť bezpečnosť cestujúcich. Metódy detekcia vozidiel sa stretávajú ale so svojimi vlastnými problémami:

-Rozmery objektu - So snímaním kamerou strácame informáciu o hĺbke. Vzniká projekčná chyba, kde objekty vzdialené majú na snímke opticky menšiu veľkosť a objekty blízko kamery zase veľkú. Na dvoch snímkach teda môže byť ten istý predmet rôzne veľký.

-Lokalizácia v obraze – Detekované objekty budeme mať len veľmi vzácne vycentrované na stred obrazu. Preto metódy detekcie vozidiel zvádzajú na použitie kĺzavého okna po celom obraze a porovnávania s modelom vozidla.

-Veľké množstvo typov objektov a ich variácia vo vzhľade – Vozidiel máme veľké množstvo najrôznejších značiek. Môže sa jednať napríklad o osobné autá, nákladné autá, kabriolety alebo pracovné stroje typu báger. Rovnako môžu byť rôznej farby. Tiež treba myslieť na to, že vozidlo môže byť snímané pod rôznym uhlom.

- Modelovanie – Ako zároveň spojiť tieto problémy do jedného modelu. Obrázok môže byť rôzne veľký, umiestnený kdekoľvek v obrázku, rôzne otočený a s rôznou farbou.

Bežný prístup:

V priebehu rokov sa vyskytlo veľké množstvo metód pre detekciu áut v obraze. Tieto metódy však nevykazovali dostatočné realtime vlastnosti. Porovnávanie s databázou a kĺzavé okná sú príliš výpočetne náročné. Jeden z dvoch metód, ktoré sú stále používané, ktoré zvolili iný prístup sú napríklad:

**Viola-Jones framework (2001)** [10.18]- Publikované v článku ‘Robust Real-time Object Detection’  Paulom Michaelom Jonesom. Pracuje na princípe generovania stoviek až tisícok jednoduchých binárnych Háárových klasifikátorov. 

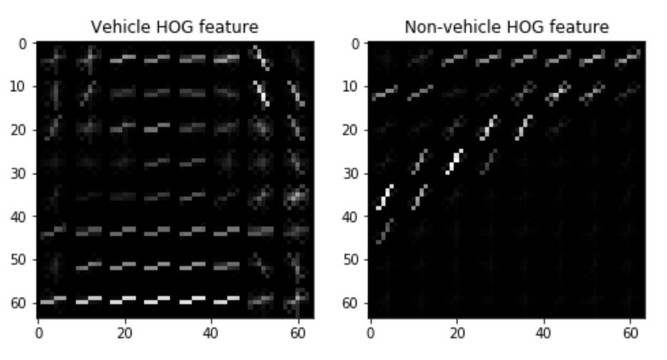
Obr.x) Ukážka Háárových obrazcov, použitých na dekciu tváre [10.18]

Využívalo sa to pre detekciu tváre pomocou web-kamery. Samotný princíp je jednoduchý a relatívne rýchly, zvláda aj realtime processing. Klasifikátory využívajú na posudzovanie multiscale kĺzavé okno v kaskáde. Rýchlo sa to implementovalo do ***OpenCV.*** Základný princíp pri tom pozostával z nasledujúcich krokov:

1. Zostrojenie datasetu s tisíckami obrázov tváre pod rovnakým uhlom a natočením.
2. Natvrdo zakódovať pomocou Háárových vlastností vlastnosti tváre. Napríklad, ako je v obrázku vidieť, oči sú tmavšie ako miesto medzi nimi odpovedajúce nosu alebo napríklad tmavá oblasť očí a svetlá obasť pod očami.
3. Natrénujeme SVM klasifikátor na daný dataset.
4. Využijeme natrénovaný klasifikátor na nové obrázky z web-kamery.

Veľkou nevýhodou bol fakt, že tvár dokázali rozpoznať iba z rovnakého úhlu, v akom boli naučené. Ak bola tvár naklonená alebo pootočené, algoritmus zlyhával.

**HOG (Histogram of oriented gradients**) **(2005)** [10.17] s využitím tzv. a SVM(Support Vector Machine) na klasifikáciu je druhým využívaným princípom. Tiež využíva multiscale kĺzavé okno. Dosahuje väčšiu presnosť ako Viola-Jones, ale je výrazne pomalší. Pôvodne vyvinutý na detekciu chodcov. Vlastnosti sú však rovnako natvrdo zakódované.



Obr.x) Ukážka funkcie HOG [10.17]

Základný princíp:

1. Pre každý pixel zistíme smer k najtmavšiemu pixelovému susedovi v jeho bízkom okolí.
2. Nakreslíme v tomto mieste šípku smerujúcu k tmavému susedovi. Vytvoríme tak gradient intenzity v obraze. Gradienty ukazujú tok od svetlých oblastí ku tmavým oblastiam.
3. Celý obraz rozdelíme na subatice o veľkosti zvyčajne 16x16 pixelov. Vytvoríme jednu centrálnu šípku podľa vhodného kritéria, napríklad funkciou max().

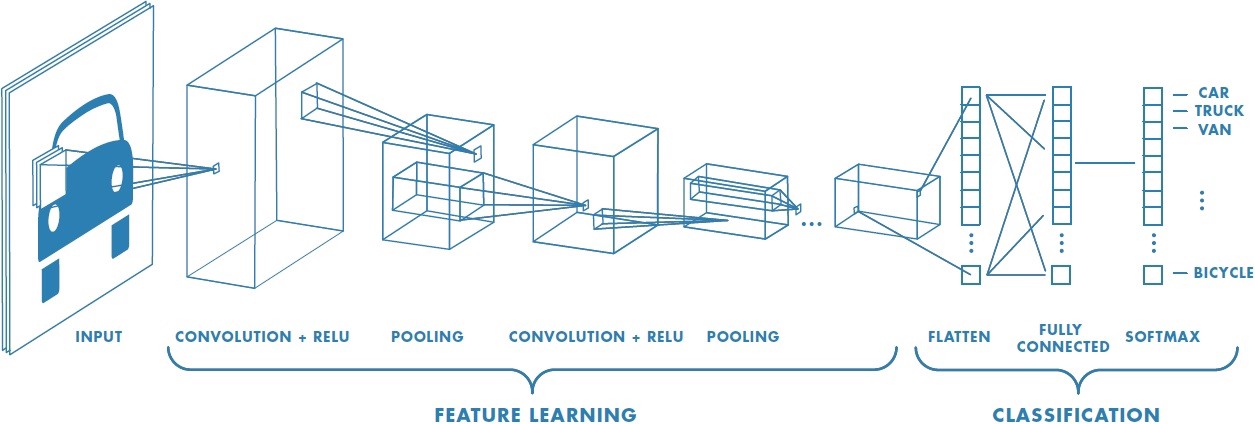
Nevýhoda:

Vlastnosti sú zakódované natvrdo. Pri zmene obrazu alebo šum zlyhávajú. Neposkytujú teda užívateľovi dostatočnú dynamiku.

Prístup s využitím neurónových sietí:

Nie je žiadnym prekvapením, že neurónové siete nahradili v efektivite bežne používané postupy ako aj v detekcii objektov tak aj pri ich klasifikácii.

**CNN (2012) -** Vý sledký výkonnostného testu naKriszhevskeho CNN ImageNete ukázaly možnosť využitia konvolučných neurónových sietí. Používa sa kĺzavé okno pohybujúe sa po celom obraze a prebieha klasifikácia. V každom kroku vypočítame, Vznikne nám tak obrovský počet klasifikovaných objektov, väčšina však má veľmi male confidence skóre – pravdepodobnosť, že san a obrázku nachádza object porovnávaného klasifikátoru. Tent spôsob funguje, ale je pomalý. Je to spôsobené práve veľkým počtom porovnáaní.



**[10.15] obr dsdsfsdf**

Princíp:

1. Zoberieme obrázok, rozsekáme ho na sériu submatíc obrazu.
2. Na každú submaticu obrazu spustíme CNN klasifikátor.
3. Pre každú class klasifikátora vypočítame confidence skóre, reprezentujúce pravdepodobnosť, že sa v obrázku nachádza objekt daného klasifikátoru.
4. Uchováme si iba značky s dostatočne vysokým confidence skóre.

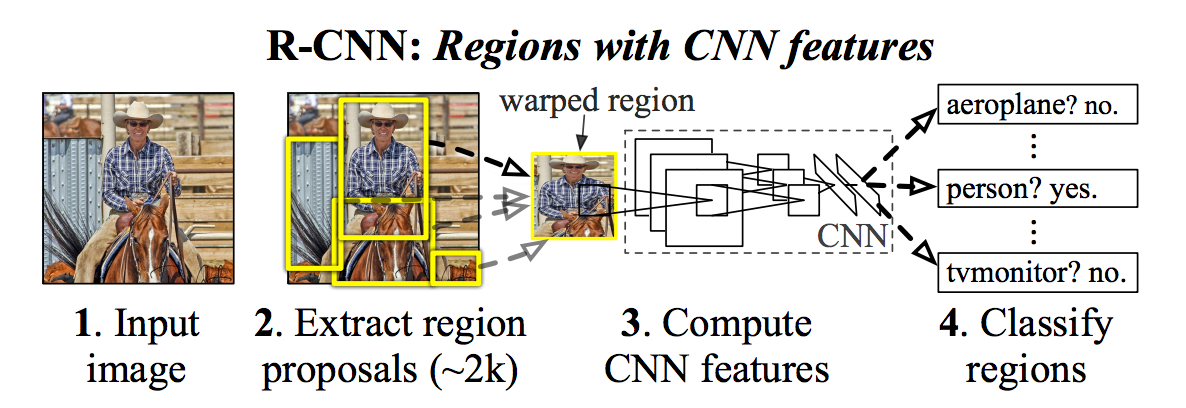
Nevýhody: Je pomalý. Prebieha tu veľký počet výpočtov a len ťažko použiteľný pre real-time aplikácie.

**OverFeat(2013)[10.9] –** Jeden z prvých pokrokov od začiatku používania CNN.Využíva viacero rôzne veľkých okien kĺzajúcich po obraze. V publikovanej verzii využívali 6 rôzne veľkých okien. Nevýhodou bolo, že výpočet prebiehal iba pre jedno okno súčasne ale na druhej strane bola zväčšená robustnosť kódu.

****

**[10.9] Zjednodušená ukážka funkcie kĺzajúcého okna**

**R-CNN (2014)** [10.0] **-** Inteligentnejšia metóda ako bežné CNN. Využíva procesu nazývaného Selektívne hľadanie. Selektívne hladanie vytvára hraničné oblasti tzv. *Bounding boxes*. Na každý obrázok je pozerané cez okná rôzných velkostí a  hľadá pixelovo podobné oblasti označujúce regiónové umiestnenie ojektu. Následne sa extrahujú vlastnosti v obraze pomocou CNN siete. Na klsifikáciu sa použije SVM.



Obr.x) Ukážka princípu R-CNN [10.0]

Postup:

1. Vytvoríme vo vloženom obrázku návrhy na Boundin boxy.
2. Jednotlivé oblasti vybraté pomocou Bounding boxov sú analyzované CNN sieťou a sú získané vlastnosti v obraze.
3. SVM nám ponúkne najlepšiu zhodu s jedným z klasifikátorov.
4. Na bounding boxy použijeme lineárny regresný model na získanie presnejších koordinátov už klasifikovaného objektu.

Nevýhody:

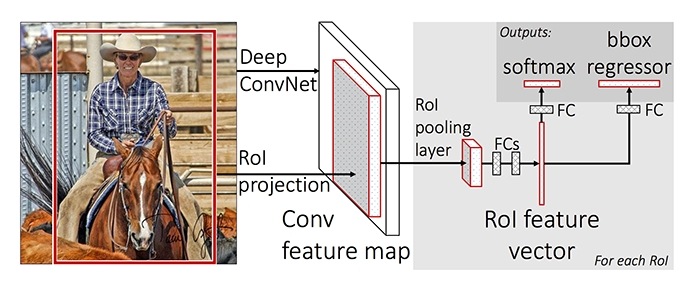
-pomalé pri tréningu aj pri testovaní. Každý regiónový návh je posunutý do CNN samosatne. Učenie neurónovej siete na VOC 2007 datasete trvalo 84 hodín.

- Vlastnosti získané z CNN nie sú updatované na základe dát z SVM a regresie.

**Vylepšenia implementácie R-CNN** [10.16]**:**

*Fast R-CNN (2015):*

Ponúka viaceré vylepšenia oproti bežnému R-CNN v rýchlosti trénovania a testovania. Zároveň je zvýšená presnosť detekcie. Pre dataset z PASCAL VOC (2012) bola dosiahnutá 9x väčšia rýchlosť trénovania a 140-213x (podľa implementie) pre testovanie ako pre bežné R-CNN. R-CNN je pomalšie. Využíva vlstnosti z R-CNNako je napríklad návrhy boundin boxov a selektívne hľade. Rozdiel je v tom, že R-CNN extrahuje jednotlivé regióny samstatne a tie posiela do neuronovej sieti a na klasifikovanie pomocou SVM klasifikátorov.ale posiela celý obrázok s extrahanými vlastnosťami. Fast R-CNN použije CNN na celý obrázok a následne nájde a zlúči regiony záujmu- ***Region of Interest (RoI) Pooling*** na obraz stnosí z CNN*.* Následne sa až posiela na klasifikáciu a regresiu. Tento prístup bol rýchlejší a umožňoval jednoduchšie učenie siete.



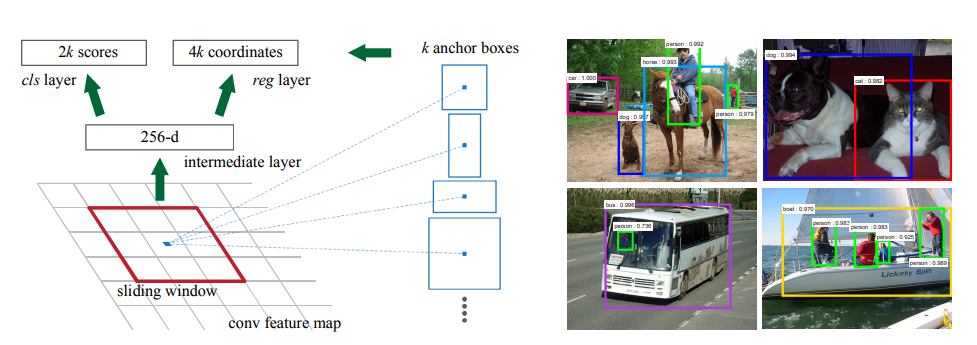
Obr.x) Ukážka funkcie Fast R-CNN[10.16]

Nevýhoda:

Jeho funkcia stále závisela na selektívnom vyhľadávaní alebo inej metóde regiónovo založeného algoritmu. Stalo sa to najslabším a najpomalším miestom tejto metódy.

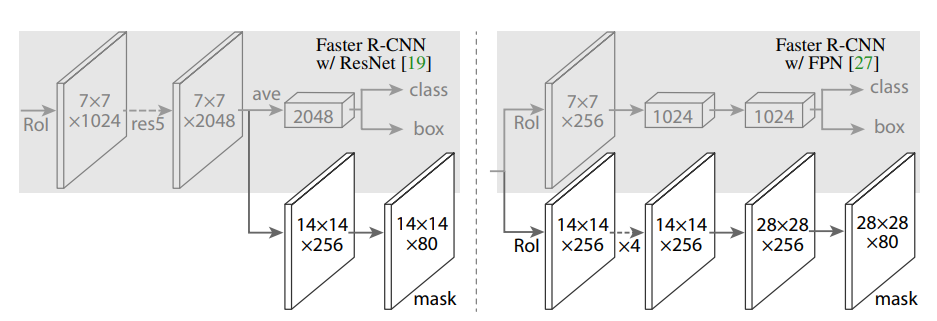
**Faster R-CNN (2015): [10.6] [10.7]**

Výrazné zrýchlenie rýchlosti. Pre trénovanie na VOC 2007 datasete rýchlosť učenia menej ako 1 hodina. Dosahuje až 250x zrýchlenie oproti R-CNN. Za zrýchlenie vďačia hlavne pridaním vrstiev pre odhad regiónov Region Proposal Network (RPN). Odstránili tak predchádzajúci slabý článok Fast R-CNN a to bolo selektívne vyhľadávanie. Výstupom RPN sú objekty s vlastnými objektovými hodnoteniami charakterizujúci, ako moc sú si istý prítomnosťou objektov v obraze. Tieto objekty sú následne poslané do ďalších vrstiev CNN na nájdenie regiónov záujmu a klasifikáciu. Dosiahnem tak jednoducho kompletne trénovatelný model neurónovej siete.



Obr.x) Ukážka funkcie návrhu regiónov pomocou RPN. Testovanie prebehlo na VOC 2007 datasete.[10.7]

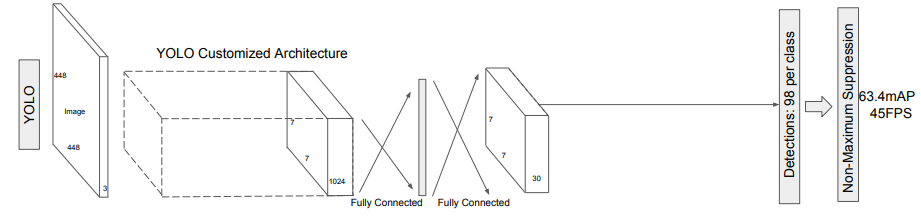
**Mask R-CNN(2017) [10.8]**

Využíva podobnú dvoj-časťovú architektúru ako Faster R-CNN. Časť *Region Proposal Network (RPN)* je identická ako v metóde Faster R-CNN. V druhej časti, paralélne spredikciou skpiny Mask R-CNN vyhodnocuje aj binárnu masku pre každý Region of Interest (RoI). Maska je naučená pomocou plne prepojenej neurónovej siete. Pomáha to v pixelovm učení a hľadaní pixelovo-objektovej zhode príslušiacej jednému objektu na báze podobnosti. Získali sme tak veľmi dobrú objektovú segmentáciu. Nevýhodou je mierne zmenšenie v rýchlosti ale je zanedbateľné pri zohladnení prínosu, ktorý prináša.

Obrázok. x) Ukážka implementácie Mask R-CNN na architektúre Faster R-CNN s využitím ResNetu a FPN[10.8]

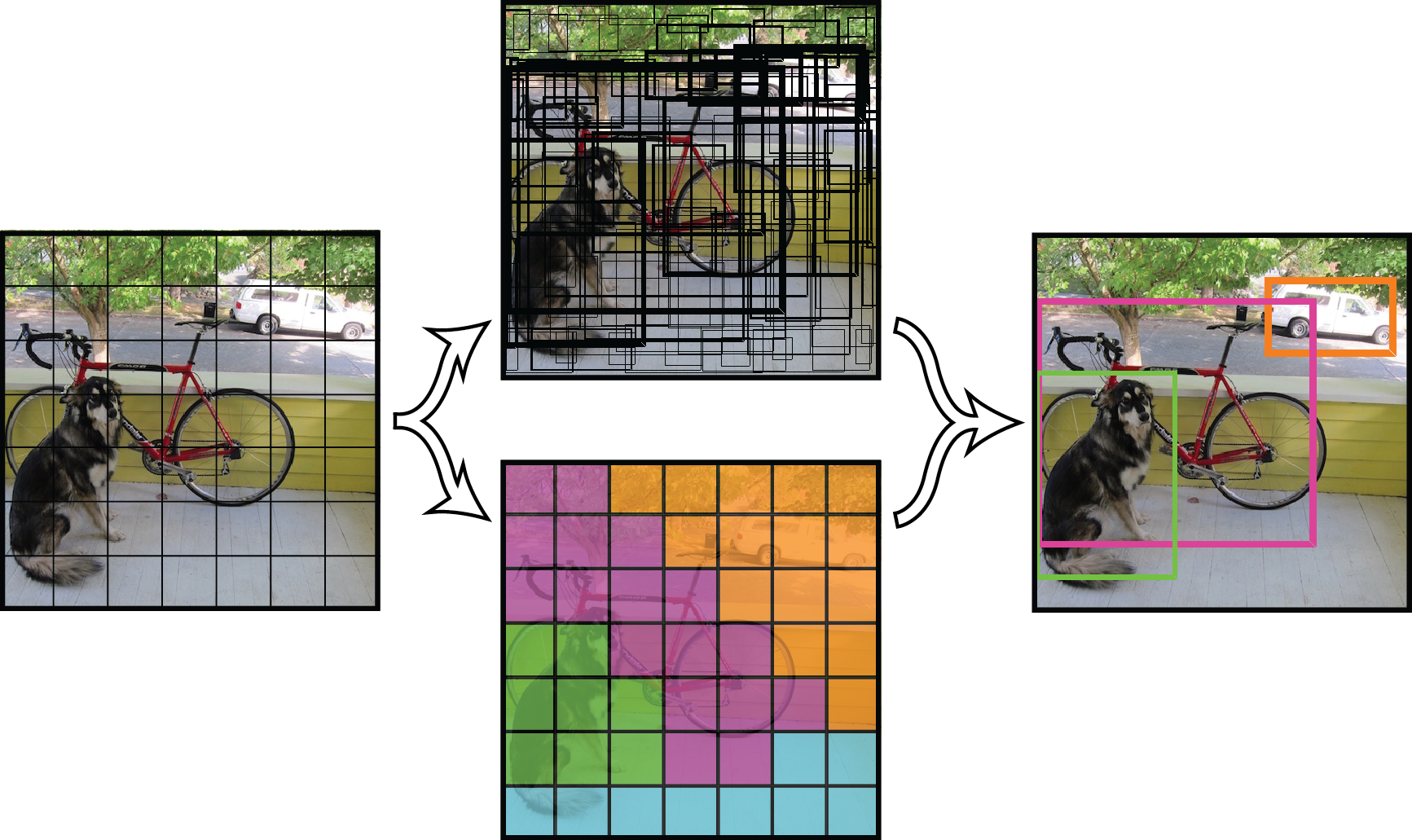
**YOLO - You Only Look Once (2016) [10.10]**

Všetky siete spomínané doteraz využívajú lokalizáciu a klasifikáciu objektov pre detekciu. Použijú model na časti obrazu s rôznou veľkosťou a pohybujúce sa po obraze. YOLO na rozdiel od nich využíva iba jednu jedinú sieť na celý obraz.



Obr.xx) Architektúra YOLO [10.13]

Táto sieť rozdelí obraz na regióny a predikuje Bounding Boxy a pravdepodobnosti pre každý Bounding Box. V obraze sa nachádza obrovské množstvo detekcií ale väčšina má predvídanú pravdepodobnosť, že sa v nich skutočne objekt nachádza veľmi nízku. Ak naprahujeme všetky nájdené aBounding Boxy a ponecháme v obraze iba tie, ktoré majú väčšiu pravdepodobnosť ako je predom stanovený prah, získame iba pár detekcií výrazne vyššie odpovedajúce skutočnému výskytu objektov.



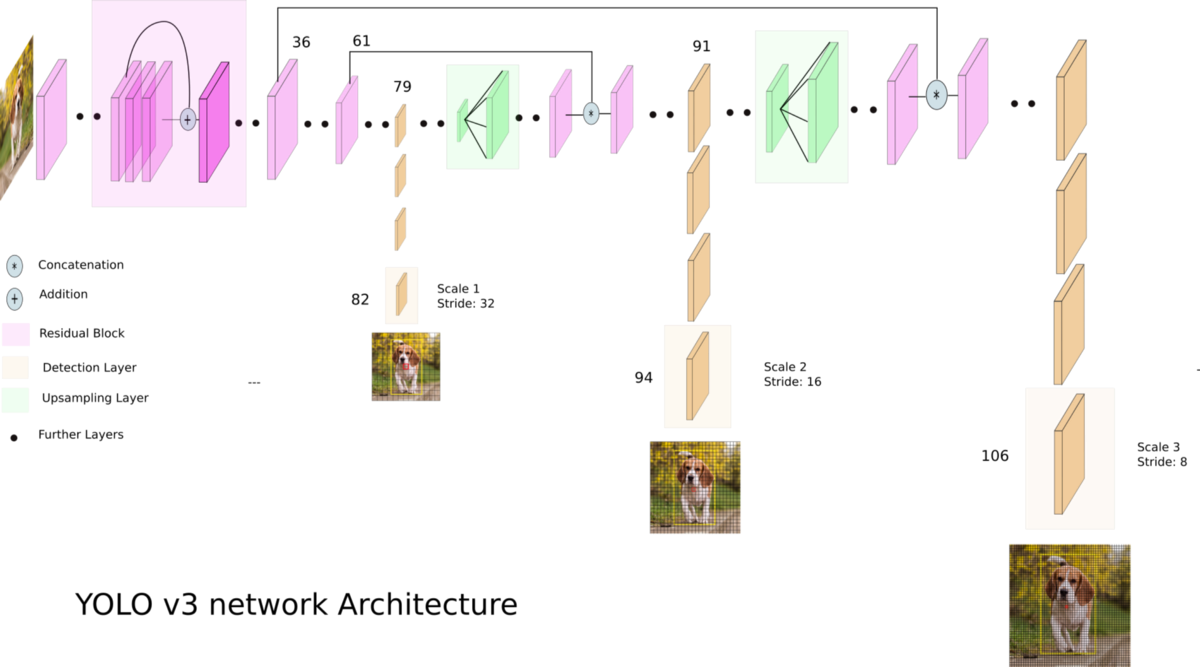
Obr.x) Ukážka priebehu hľadania Bounding Boxov pre YOLO[10.10]

YOLO použije iba jediný prichod cez CNN pre celý obraz. Nepoužíva tisíce priechodov pre všetky oblasti obrazu. Je tým dosiahnuta výrazne rýchlejší chod a po prvý krát kompletne real-time aplikácia. Táto metóda pri tom dosahuje vysoké presnosti. Od vtedy však táto metóda prešla sériou vylepšení.

***YOLO v2 (9000): Better, Faster, Stronger 2017*** [10.11]

Využíva zopár zlepšení a trikov ako zlepšiť efektivitu a rýchlosť pôvodného YOLOa. Používa architektúru darknetu s 19 vrstvovou sieťou s ďaľšími 11 vrstvami pre objektovú detekciu. Na Titan X doshuje rýchlosť až 45 fps. Jeho veľkou nevýhodou bolo, že strácal kvalitu detekcie malých objektov z dôvodu podvzorkovania vstupného obrazu. Z tohoto dôvodu sa začali používať zlúčenia vlastnostných máp z predchádzajúcich vrstiev. Podobne ako OverFeat a SSD používa plne konvolučný model ale stale sa trénuje na celom obrázku. Podobne ako pri Faster R-CNN upravujeme hranice na bounding boxoch namiesto priamej počiatočnej predikcie hraníc.

***YOLO v3*** ***: An Incremental Improvement 2018 [10.12]***

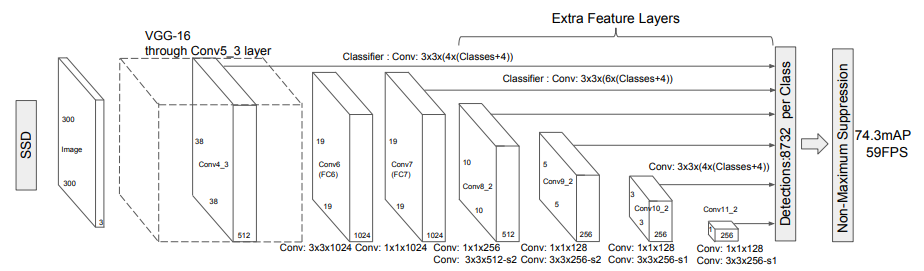
YOLO v2 patrilo v svojom čase k najrýchlejším a najpresnejším detektorom objektov v obraze. Netrvalo však dlho a prišlo sa s metódami ako RetinaNet a SSD(popísané nižšie), ktoré ho výkonnostne porazili. Rýchlostne však stále patril k najrýchlejším. YOLO v3 používa až 53 pôvodných vrstiev Darknetu pre extrahovanie vlastností obrazu a ďaľších 53 vrstiev pre detekciu. Prišlo z zhoršeniu rýchlosti za cenu zvýšenia presnosti. Dosahuje iba 30 fps. Veľkou výhodou je aj zlepšenie detekcie malých objektov vďaka doprednému pričítaniu predchádzajúcich nepodvzorkovaných máp vlastností. 

Obr.x) Architektúra YOLO v3 [10.12]

Pre YOLO v3, detekcia je vykonaná aplikáciou 1x1 detekčných kernelov na mapu vlastností o troch odlišných veľkostiach a v iných častiach siete.

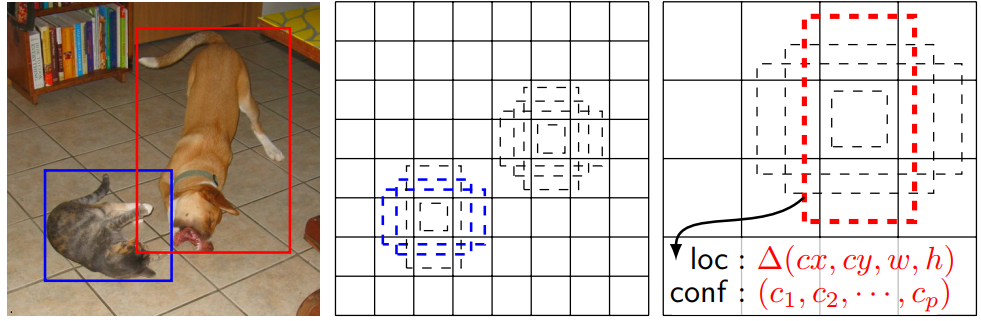
**SSD - Single Shot MultiBox Detector [10.12]**

Predchádzajúce metódy detekcie objektov zdielali jednu rovnakú vlastnosť. Sú rozdelené na dve časti. Prvá časť je určená k odhadom regiónového ohraničenia objektov a druhá časť s kvalitnými klasifikátormi na klasifikáciu objektov v ohraničení pomocou Bounding Boxes.



Obr.xx) Architektúra SSD [10.13]

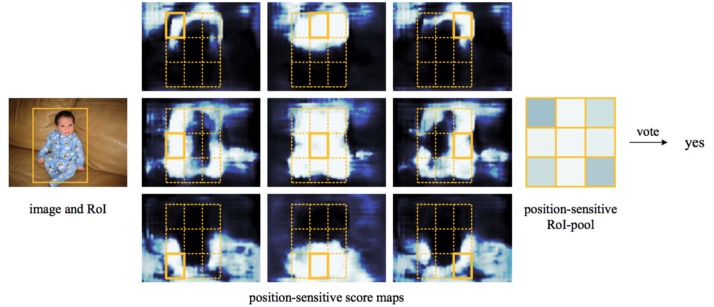
Tieto metódy sú veľmi presné, ale platia za to vysokými výpočetnými nákladmi – malé fps. Z tohto dôvodu nie sú vhodným kandidátom na použitie v embedded zariadeniach. SSD je navrhnuté tak, aby skombinovali tieto dve úlohy do jednej siete. Namiesto siete vytvárajúcej návrhy na Bounding Boxy môžeme mať predefinované boxy, v ktorých hľadáme objekty. Používajúc vlastností obrazu získaných z konvolučných máp z vrstiev, nachádzajúcich sa v druhej polovici siete, použijeme malé konvolučné filtre na predikciu máp vlastností z obrazu na predikciu pravdepodobnosti výskytu objektu v obraze a samotného bounding boxu. Pre nájdenie Bounding Boxov a triedy objektu používame viaceré aktivačné mapy s rôznymi veľkosťami pre predikciu.



Obr. x) Ukážka nájdenia vlastností pomocou máp vlastností v obraze. [10.13]

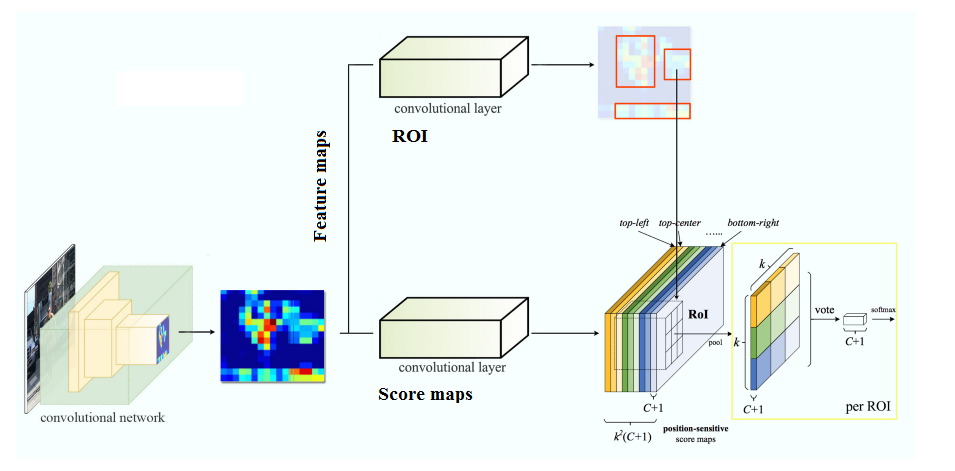
**R-FCN - Region-based Fully Convolutional Networks [10.14]**

R-FCN prináša zlepšenie rýchlosti vďaka zníženiu potrebnej práce potrebnej pre každú oblasť záujmu. Mapa vlastností je posielaná na paralélne vetvy. V jednej prebieha klasický výpočet oblastí záujmov a v druhej sa nachádzajú tzv. Score maps. V týchto skóre mapách sa nachádza zhoda časti obrazu s objektom. Na nasledujúcom obrázku je ukážka princípu skóre máp. Oblasť Bounding boxu je rozčiastkovaná v tomtorípade na 9 subblokov s vlastnými mapami vlastností a v každom sa porovnáva zhoda s časťou objektu, ktorá by sa a nachádzať na rovnakej pozícii, ak by tam objekt bol. V obrázku sa snažíme nájsť v objekte dieťa. Lokalizácia prebehla úspešne.

****

**Obr.X) Ukážka prcípu** Skóre máp **[10.14]**

Samostatné mapy vlastností pre ždý subblok nedáva výrazne precíznu odpoveď. Ak ich ale skombinujeme viacej, v tomto prípade 9, a vieme čo sa má nachádzať v ktorej časti objetu, dokážeme výrazne spresniť detekciu. Architektúra siete je na nasledujúcom obrázku. Využíva podobnú architektúru ako Faster R-CCN ale iba s konvolučnými sieťami.

****

**Obr.x) Architektúra R-FCN [10.14]**

**Výber vhodného datasetu: [10.0]**

Dataset môže zohrávať významnú rolu v procese učenia neurónových sietí. Pri učení sa požaduje dostatočná variácia zisťovaných objektov ale zároveň dostatočná specificita oblasti použitia. Napríklad pri detekcii značiek v Českej Republike bude vhodné učiť NN na značky získané v CR. Odlišnosti v značke môžu zhoršovať ich detekciu. Na druhej strane napríklad pri príchode zahraničného vozidla musí byť značka detekovaná tiež.

Pri väčšom množstve dát pri učení neurónovej sieti stúpa efektívnosť neurónovej sieti. Nezabúdame pri tom na možnosť preučenia. V praxi sa však ťažko dostávame k väčšiemu objemu obrázkov použiteľných pre učenie úloh v oblasti object classification. Firmy nerady dávajú svoje dáta zadarmo a získavanie obrázkov napríklad pomocou Google môže byť prácne na anotáciu a triedenie. V nasledujúcej tabuľke je zoznam najznámejších bežne používaných voľne dostupných datasetov:

|  |  |  |  |
| --- | --- | --- | --- |
| Meno | Počet obrázkov | Počet tried | Posledný update |
| COCO [10.1] | 200K | 80 | 2017 |
| Pascal Voc [10.2] | 12K | 20 | 2012 |
| Oxford-IIIT Pet [10.3] | 7,5K | 37 | 2012 |
| KITTI Vision [10.4] | 15K | Only cars 3 | 2012 |
| Stanford Car dataset [10.5] | 16K | 196 of cars | 2012 |

Voľnodostupné porovnania výkonnosti z iných meraní:

Použité neurónové siete:

**SSD: Single Shot MultiBox Object Detector využívajúci PyTorch [5.0]**

citácie:

[10.0] https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/

[10.1] http://cocodataset.org/#download

[10.2] http://host.robots.ox.ac.uk/pascal/VOC/

[10.3] http://www.robots.ox.ac.uk/~vgg/data/pets/

[10.4] http://www.cvlibs.net/datasets/kitti/eval\_object.php?obj\_benchmark=2d

[10.5] <http://ai.stanford.edu/~jkrause/cars/car_dataset.html>

[10.6] https://web.cs.hacettepe.edu.tr/~aykut/classes/spring2016/bil722/slides/w05-FasterR-CNN.pdf

[10.7] <https://arxiv.org/pdf/1506.01497.pdf>

[10.8] <https://arxiv.org/pdf/1703.06870.pdf>

[10.9] <http://slideplayer.com/slide/10395667/>

[10.10] <https://arxiv.org/pdf/1506.02640.pdf>

[10.11] <https://pjreddie.com/media/files/papers/YOLO9000.pdf>

[10.12] https://arxiv.org/pdf/1512.02325.pdf

[10.12 ]<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

[10.13] <https://arxiv.org/pdf/1512.02325.pdf>

[10.14] <https://medium.com/@jonathan_hui/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99>

[10.15] <https://medium.com/@rohanthomas.me/convolutional-networks-for-everyone-1d0699de1a9d>

[10.16] <https://arxiv.org/pdf/1504.08083.pdf>

[10.17]<https://towardsdatascience.com/detecting-vehicles-using-machine-learning-and-computer-vision-e319ee149e10>

[10.18] https://www.superdatascience.com/opencv-face-detection/

[5.0] https://github.com/amdegroot/ssd.pytorch