Lekce 7

Write down how is AP_{50} computed. [5]

1. Pro každou ze tříd narankujeme bounding boxy podle confidence levelu
2. V rámci jedné třídy postupujeme v pořadí z bodu (1) a kreslíme si precision/recall curve. Box bereme jako match pokud má $IoU > 0.5$
3. Upravíme křivku tak, aby byla monotónní
4. AP pro jednu třídu je průměrná precision v recallu 0, 0.1, 0.2, ..., 1.0, a AP pro celý dataset je průměr AP jednotlivých tříd

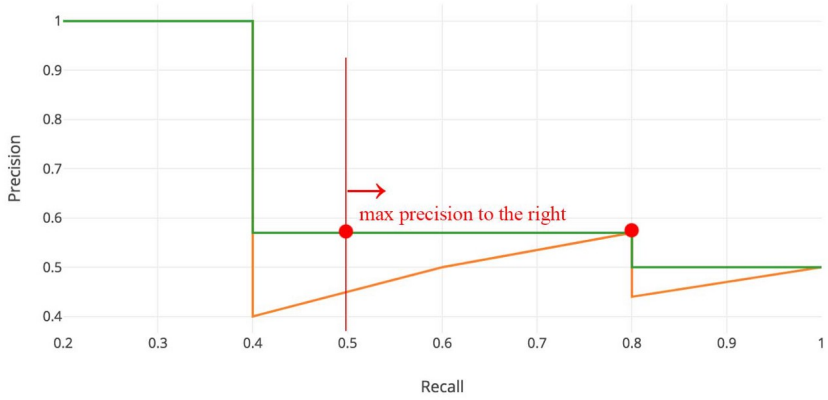

Figure 1: img

Considering a Fast-RCNN architecture, draw overall network architecture, explain what a RoI-pooling layer is, show how the network parametrizes bounding boxes and write down the loss. Finally, describe non-maximum suppression and how is the Fast-RCNN prediction performed. [10]

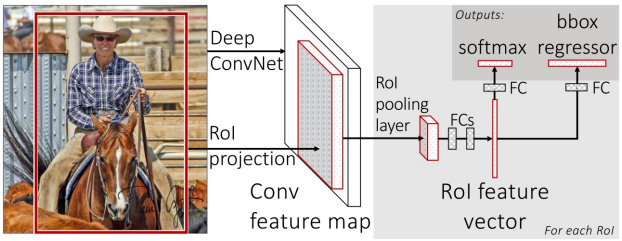

Figure 2: img

1. Začátek jako VGG, získáme 14x14 reprezentaci obrázku
2. Místo max poolingu na vyrobíme 7x7 reprezentaci pomocí RoI poolingu (viz níže)

3. Získám tím 7x7 reprezentaci každého RoI, kterou proženu FC a softmaxem (jako ve VGG) abych získal její classu (kterých je $K+1$, abychom mohli říct "nic"), a bbox regresorem, abych získal její pozici (viz níže)

RoI Pooling

RoI rozdělím na 7x7 binů, každý z nich zaokrouhlím na původní 14x14 reprezentaci (viz obrázek níže) a jen z nich udělám max pooling (čímž jeden ze 7x7 binů).

Parametrizace bounding box

Pozice je dána relativně k RoI. Konkrétně

$$t_x = (x - x_r)/w_r, \quad t_y = (y - y_r)/h_r$$
$$t_w = \log(w/w_r), \quad t_h = \log(h/h_r)$$

Logaritmy jsou ve w a h proto, že zmenšují range generovaných čísel (což ej pro síť vždycky fajn).

Loss

Pro bbox se používá tzv. Huber loss, která se stará o gradient clipping.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Celková loss ještě počítá s loss za klasifikaci, s tím, že loss za bbox se počítá jen pro "opravdové" třídy (tedy nesnažíme se bbox dávat kolem třídy "nic"),

$$L(\hat{c}, \hat{t}, c, t) = L_{\text{cls}}(\hat{c}, c) + \lambda \cdot [c \geq 1] \cdot \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}\left(\hat{t}_i - t_i\right)$$

Inference

Non-maximum supression se stará o to, aby se nám jeden objekt nezahlásil v několika různých RoI. Ignoruje RoI, které mají IoU nad nějakou hranicí s jiným RoI ze stejné třídy, který je lepší. Lepší jsou takové RoI, které mají vyšší pnost správné třídy.

> Considering a Faster-RCNN architecture, describe the region proposal network (its architecture, what are anchors, what does the loss look like). [5]

RPN

1. Posouváme 3x3 window po získáne conv reprezentaci
2. Pro každou pozici vygenerujeme *anchory*, většinou jich bývá devět (tři různé velikosti, tři různé poměry stran)

3. Pro každý anchor predikujeme, jestli je v něm nějaký objekt (tj. jen binární cross-entropy loss nad sigmoidem), a pokud ano, tak kde leží jeho bbox.

Trénink

1. Máme "opravdové" gold objekty i s umístěním
2. Vygenerované anchory s největším překryvem s nějakým "opravdovým" objektem, a anchory s IoU $> 70\%$, bereme jako pozitivní příklady (třída: objekt)
3. Anchory s IoU $< 30\%$ bereme jako negativní příklady (třída: nic)
4. Zbytek achorů ignorujeme

Hlavy tedy trénujeme tak, aby správně předpovídaly třídu z (2) a (3), popř. ještě bbox těch "opravdových" objektů.

Inference

1. RPN vyhodí nějaké anchory spolu s tím, jestli na nich něco je a popř. kde
2. Za pomocí non-maximum supression vyhodíme anchory ukazující na stejný objekt
3. Zbytek non-background anchorů použijeme jako RoI ve zbytku Fast-RCNN sítě

> Considering Mask-RCNN architecture, describe the additions to a Faster-RCNN architecture (the RoI-Align layer, the new mask-producing head). [5]

RoI Pooling je nahrazen RoI Align. Každý ze 7x7 binů si rozdělíme na 4 podbiny, a jejich hodnoty získáme bilineární interpolací hodnot z původní reprezentace 14x14. Tyto čtyři podbiny se zkombinují do finální hodnoty.

Pro vytvoření masky nejprve upscalujeme 7x7 reprezentaci zpět na 14x14, nebo 28x28. Poslední maskující konvoluce má tolik kanálů, kolik máme tříd, a masku tvoříme (a trénujeme) pro každou třídu zvlášť.

> Write down the focal loss with class weighting, including the commonly used hyperparameter values. [5]

Úprava loss tak, aby se nám pozitivní příklady objektů neutopily v těch negativních.

$$\mathcal{L}_{\text{focal-loss}} = -\left(1 - p_{\text{model}}\left(y \mid x\right)\right)^{\gamma} \cdot \log p_{\text{model}}\left(y \mid x\right)$$

Pro $\gamma = 0$ je tato loss prostě cross-entropy, pro vyšší γ vlastně snižujeme váhu loss u příkladů, u kterých si jsme hodně jistí výsledkem. Nejčastěji se používá $\gamma = 2$.

Navíc se ještě může každá třída různě navážit pevnou konstatnou,

$$-\alpha_y \cdot \left(1 - p_{\text{model}}\left(y \mid x\right)\right)^{\gamma} \cdot \log p_{\text{model}}\left(y \mid x\right)$$

Pro vzácné třídy bývá nejčastěji používána hodnota $\alpha = 0.25$.

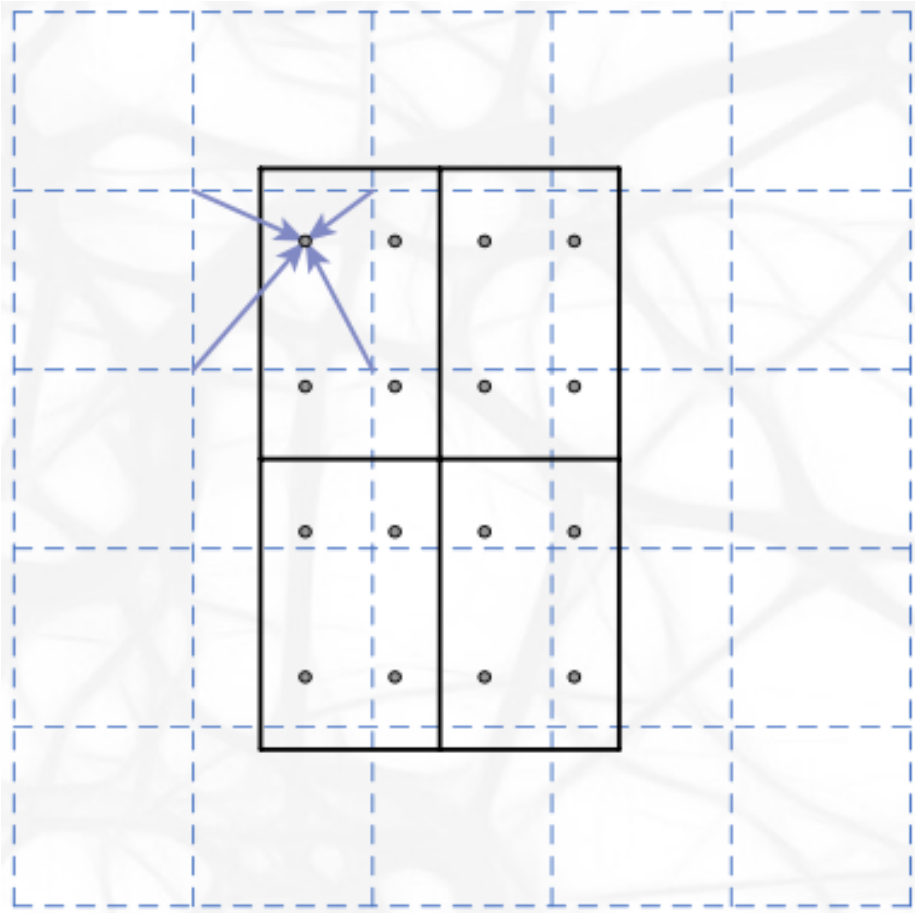

Figure 3: image-20210629131428781

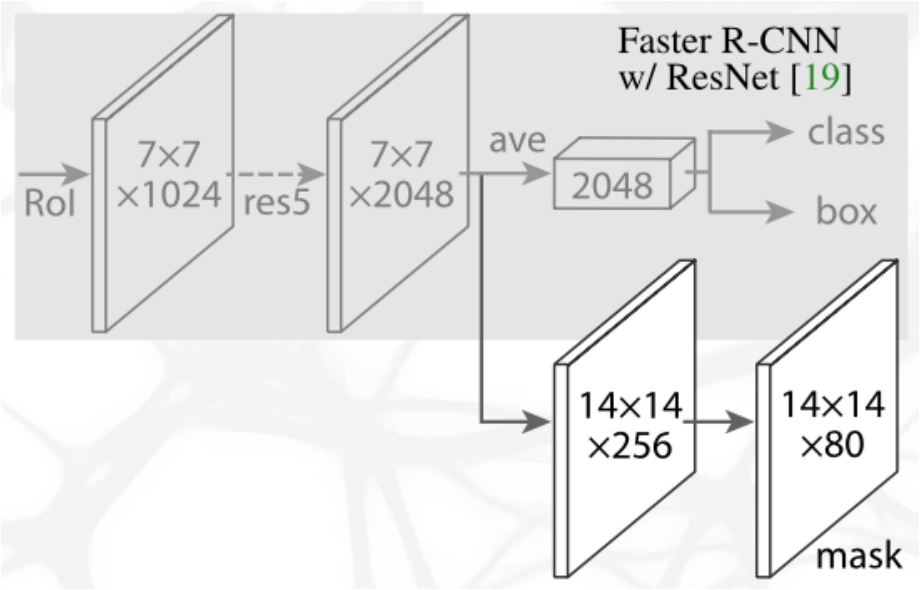

Figure 4: image-20210629133903556

> Draw the overall architecture of a RetinaNet architecture (the FPN architecture including the block combining feature maps of different resolutions; the classification and bounding box generation heads, including their output size). [5]

Oproti ResNetu mají navíc ještě C6 a C7, tj celkově dělají 7 max poolingů.

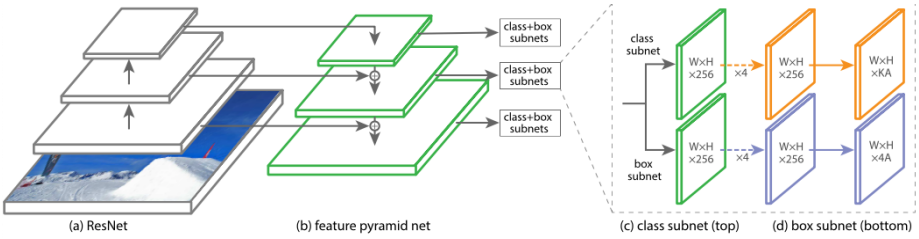

Figure 5: image-20210629151525773

Klasifikační hlava má na výstupu $K \cdot A$ kanálů. Kolem každého "pixelu" výstupu máme A anchorů (většinou 9) a pro každý z nich potřebujeme říct pnost každé z K tříd. Klasifikace je tedy plně obstarána těmito konvolucemi, žádný pooling už nenásleduje.

Bounding boxová hlava má $4 \cdot A$ kanálů, pro každý anchor určuje hodnotu čtyřech parametrů.

V rámci ResNetu jsou mezi C vrstvami prostě max poolingy, v FPN probíhá jednoduchý 2x upscaling (doslova stávající hodnota zkopíruje na ta nová místa) a featury zleva projdou 1x1 konvolucí, aby měly správný počet kanálů.

> Draw the BiFPN block architecture, including the positions of all convolutions, BatchNorms and ReLUs. [5]

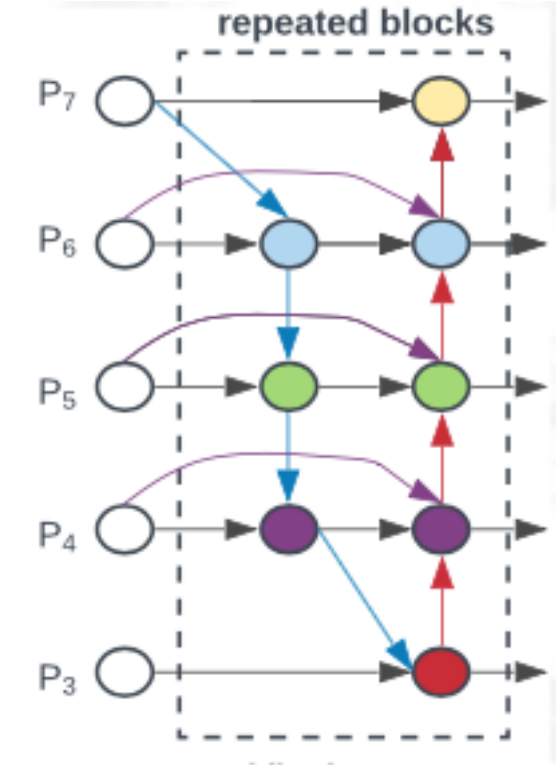

Figure 6: image-20210629154103859

Součást EfficientDet. Všechny bloky jsou 3x3 separabilní kovoluce s BN a ReLU. Důležité jsou reziduální hrany, a dva chybějící bloky v prostředním sloupci, do kterých vedla jen jedna hrana a tak bylo možné je odstranit bez změny výsledku.

Downscaling je přes max pooling, upsampling jednoduše zkopírováním hodnot. Součet hodnot je navážený.