ZADÁNÍ PROJEKTU

POSTPROCESSING DAT Z REGION PROPOSAL DETEKČNÍ NEURONOVÉ SÍTĚ

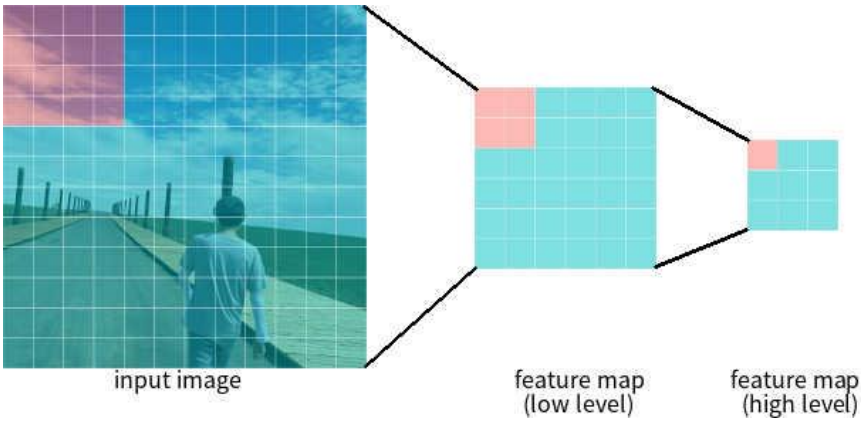
Cílem projektu je vytvořit optimalizovaný algoritmus pro zpracování výstupu z neuronové sítě pro detekci objektů v obraze. V rámci tohoto projektu se zaměříme pouze na výstupy z neuronové sítě RetinaNet a tak, s výjimkou následujících informací, není nutné tuto síť podrobně znát. Pro naše účely předpokládáme, že neuronová síť detekuje jednu třídu objektu. Výstupem takovéto sítě jsou vektory hodnot. Jeden z těchto vektorů obsahuje čtveřice hodnot (x, y, w, h) typu *float*, které jsou vázány na tzv. *Anchors/prior boxes* (kotvy). Každá čtveřice má svou přesně danou pozici na snímku. Druhý vektor určuje pravděpodobnost výskytu každé čtveřice v obraze.

V rámci projektu je vaším úkolem získat C++ vektor/pole, kde jednotlivé prvky jsou tvořeny strukturou s reálnou pozicí ohraničujícího obdélníku (*bounding box*) ve snímku, tedy x, y, w a h.

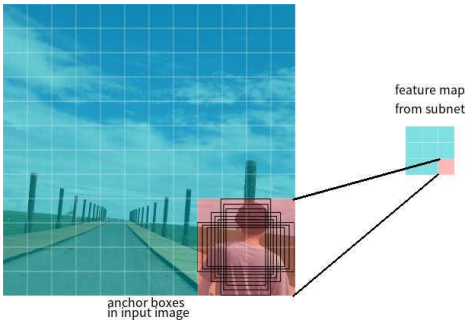
Princip fungování RetinaNet neuronové sítě

Anchors/Prior boxes

Vstupní snímek je rozdělen do 1-n mřížek, přičemž každá úroveň je zodpovědná za detekci určité velikosti objektu.



Když se podíváte na největší obrázek, tak granularita mřížky neumožňuje zachytit celou postavu, tu je možné zachytí až na dalších úrovních. Každé buňce jsou přiřazeny *anchors*, které mají různé poměry (viz další obrázek).



Pokud víme, že ke každé kotvě je přiřazena jedna čtveřice (x,y,w,h) a máme vstupní obrázek, který je rozdělen do mřížky 12x12,6x6 a 3x3 s tím, že každá buňka obsahuje dvě kotvy, tak celková velikost pole pro určení souřadnic je: (144+36+9)×2×4.

NMS (Non Maximum Suppression)

Detekční sítě často produkují více detekcí kolem jednoho objektu. Aby finálním výstupem byla pouze jedna oblast, která "obklopuje" objekt, používá se algoritmus NMS, který rovněž způsobuje režii.

https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

Šablona

Pro projekt jsme vám připravili podobnou šablonu, jakou používáme v rámci cvičení. Šablonu můžete libovolně upravovat s tím, že algoritmy mají být zachované. V poskytnutém C++ řešeni se používají *float* vektory, které můžete bez problému nahradit za pole. Hlavním cílem projektu je **optimalizovat dobu postprocessingu**, která může představovat u méně výkonných zařízení významné zdržení při použití neuronové sítě. Projekt vykresluje obrázek a náhodně rozmístěné obdélníky, které simulují detekce.

Hodnotit se bude zejména rychlost zpracování a přístup, jaký k řešení projektu využijete. Odevzdanému projektu musíte plně rozumět.

Pro x86-64:

- Povolené jsou instrukce SSE, AVX, AVX2, které mají podporu na Merlinovi. Nepoužívejte FMA instrukce.

Pro ARM:

- NEON instrukce

Odevzdání

Odevzdávat budete zdrojové kódy, které by měly být přeložitelné a spustitelné buď pomocí VS Code Tasks nebo pomocí Makefile. V případě, že použijete Makefile, tak aplikace musí být spustitelná pomocí make run. Šablonu si můžete libovolně upravovat, nicméně algoritmus musí být zachován.

Dokumentace

K projektu budete odevzdávat stručnou dokumentaci, kde uvedete, jak jste projekt řešili a jakým způsobem jste dosáhli prezentovaného zrychlení.

Struktura odevzdání

```
xlogin00.zip
/src/*
/doc.pdf
/README.md
```

Zdroje

https://builtin.com/machine-learning/non-maximum-suppression

https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

https://blog.zenggyu.com/posts/en/2018-12-05-retinanet-explained-and-demystified/index.html

https://www.oreilly.com/library/view/practical-machine-learning/9781098102357/ch04.html