
# IPA - PROJEKT

## POSTPROCESSING DAT Z REGION PROPOSAL DETEKČNÍ NEURONOVÉ SÍTĚ

Marek Kozumplík - xkozum08

12. května 2024

# 1 Úvod

Optimalizoval jsem verzi aplikace pro x86 na Windows. Na serveru Merlin se mi projekt nepodařilo přeložit. Testování a výsledky počtu instrukcí jsou z vlastního počítače. Používám intrinsic funkce v C++.

# 2 Read

Tato část aplikace zabírá největší počet instrukcí. Přepsal jsem funkci readFloatsFromFile, která nyní rovnou naplní předalokované 2D pole `__m256` hodnot. Není nutné používat splitFloats. Toto umožní lepší vektorizaci výpočtů u decode. Nejlepší způsob optimalizace by byl jiný formát vstupních float hodnot, aby se nemusel každý `float` překládat ze `stringu`. Vstup se parsuje po jednotlivých znacích místo řádků. Float hodnoty ve vstupním souboru mají velký počet desetinných míst, které `std::stof` stejně zaokrouhlí. Díky čtení po znaku můžu po několika číslicích čtení přeskočit až do další float hodnoty. Toto zaokrouhlování snížilo počet instrukcí skoro 3x (záleží podle počtu desetinných čísel, které čteme), výsledné hodnoty se téměř neliší a obraz se zaokrouhlí na jednotlivé pixely. U čtení score jsem si všiml, že se používá pouze každé druhé score, proto se čtou jen tyto hodnoty a ostatní se přeskakují.

```
// vec[0][[x1,...,x8],[x9,...,x16],...,[...x12600]]
// vec[1][[y1,...,y8],[y9,...,y16],...,[...y12600]]
// vec[2][[w1,...,w8],[w9,...,w16],...,[...w12600]]
// vec[3][[h1,...,h8],[h9,...,h16],...,[...h12600]]
// vec[4][[score1,...,score8],[score9,...,score16],...,[...score12600]]
// it prepares for decode() function that uses avx intrinsic instructions
```

# 3 Decode

U decode díky upravenému výstupu z read provádíme stejné výpočty jako v šabloně, ale pomocí intrinsic AVX funkcí pro 8 float hodnot najednou. Jelikož nešla použít funkce `__m256 _mm256_exp_ps`, použil jsem exponenciální funkci pomocí Taylorova rozvoje:Odkaz. Výstup vypadá následovně.

```
// vec[0][[decoded_xmin1,...]...]
// vec[1][[decoded_ymin1,...]...]
// vec[2][[decoded_xmax1,...]...]
// vec[3][[decoded_ymax1,...]...]
```

# 4 Confidence threshold a resolution scale

U filtrování boxů pomocí confidence threshold používám intrinsic funkci `_mm256_cmp_ps`. Pro scale rozlišení používam funkci `_mm256_mul_ps`. `det_boxes` jsou místo vectoru předalokované pole `__m256` hodnot.

# 5 NMS

Zkoušel jsem funkci optimalizovat pomocí AVX funkcí, ale vždy byl výsledný počet instrukcí vyšší než sekvenční přístup. U vyššího počtu `det_boxes` by se AVX vyplatilo, ale tato funkce provádí velmi malou část instrukcí oproti zbytku programu. Dosáhl jsem menšího počtu instrukcí po úpravě while loopu na for a předalokovaným polem místo `vector<vector<float>>`.

# 6 Porovnání počtů instrukcí

Výsledný počet instrukcí před a po optimalizaci a s přepínačem -O3

před:

```
READ: 231911244
DECODE: 14641164
FILTER BY SCORE + SCALE RESOLUTION: 637668
NMS: 48312
Box 201.762115 154.436249 269.211731 221.164856 0.999921
Box 459.522980 311.795715 498.443726 352.232941 0.999570
Box 517.825378 144.791428 552.312317 178.225937 0.999422
Box 381.514008 268.326721 398.672150 285.971924 0.999350
Box 470.997192 6.486499 504.983704 40.751251 0.999324
Box 271.562012 194.635025 340.292084 271.930237 0.999250
Box 601.669373 332.585052 622.928223 350.409271 0.999081
Box 318.770508 62.721382 335.149536 79.139435 0.999045
```

po:

```
READ: 122548761
DECODE: 2235240
FILTER BY SCORE + SCALE RESOLUTION: 95184
NMS: 25884
Box 201.762146 154.436264 269.211700 221.164841 0.999921
Box 459.522980 311.795715 498.443726 352.232941 0.999569
Box 517.825378 144.791428 552.312317 178.225937 0.999422
Box 381.514008 268.326721 398.672150 285.971924 0.999349
Box 470.997192 6.486499 504.983704 40.751251 0.999323
Box 271.561981 194.635025 340.292053 271.930206 0.999250
Box 601.669373 332.585052 622.928223 350.409271 0.999081
Box 318.770508 62.721382 335.149536 79.139435 0.999044
```

s -O3:

```
READ: 92663064
DECODE: 301356
FILTER BY SCORE + SCALE RESOLUTION: 59724
NMS: 5436
Box 201.762146 154.436264 269.211700 221.164841 0.999921
Box 459.522980 311.795715 498.443726 352.232941 0.999569
Box 517.825378 144.791428 552.312317 178.225937 0.999422
Box 381.514008 268.326721 398.672150 285.971924 0.999349
Box 470.997192 6.486499 504.983704 40.751251 0.999323
Box 271.561981 194.635025 340.292053 271.930206 0.999250
Box 601.669373 332.585052 622.928223 350.409271 0.999081
Box 318.770508 62.721382 335.149536 79.139435 0.999044
```