УДК 004.932.1; 004.932.2; 004.932.4.

*П.О. Приставка, д.т.н., А.К. Шевченко*
*(Національний Авіаційний Університет, Україна)*

**Дослідження швидкодії операції згортки що було оптимізовано за допомог ою SIMD NEON ARM64 в умовах обробки відео з файлу.**

*Запропоновано новий підхід щодо пришвидшення виконання операції згортки, що підвищує її швидкодію за рахунок «розмотування циклу» найбільш вимогливої частини алгоритму, яка використовує SIMD A64 (ARM64) операції.*

**Постановка задачі.**

Обробка цифрових зображень (ЦЗ) грає важливу роль в розробці програмного забезпечення (ПЗ), такого як: обробка відеопотоку (стабілізація, фільтрація, зменшення шуму, стиснення тощо), обробка одного зображення у задачах машинного навчання [1,2]. Нами було досліджено швидкодію оптимізованої операції згортки (ООЗ) за допомогою SIMD NEON ARM64 [3]. Результатом цього дослідження можна вважати суттєве збільшення швидкодії за рахунок використання ядерами згортки простих умов: 1) елементи ядер мають мати «int8_t» тип, а результат не має перевищувати діапазон значень «int16_t»; 2) елементи ядер мають мати «uint8_t» тип, а результат не має перевищувати діапазон значень «uint16_t». А результат дослідження полягає у підвищені швидкодії ООЗ у 4,5 разів (у середньому). Це призвело до більш детального аналізу, як коду алгоритму, так і продовженню експериментів, у контексті аналізу підвищення швидкодії ООЗ, в умовах обробки відео з файлу/потоку.

**Опис дослідження.**

Як і було згадано, нами було проведено аналіз коду ООЗ. Було виявлено факт того що швидкодія може бути збільшена на 10-25% за допомогою «розмотування циклів». Експериментально (на практиці) було доведено, що принцип «розмотування циклів» підвищив швидкодія на 8-19% у порівнянні з [3]. Таке підвищення було проаналізовано, та було прийнято рішення надалі провести дослідження швидкодії ООЗ в умовах обробки відео/потоку.

Отже сам експеримент був створений за допомогою комбінації бінарного/вихідного файлу та алгоритму проведення та відслідковування FPS, написаного на Bash. Суть самого алгоритму заміру швидкодії ООЗ відносно cv::filter2d(…), заводиться до того, що було проведено заміри FPS (для кожного з них) впродовж 1000 секунд (для кожного розміру ядра [2x2 … 16x16]). Усе було зведено у графіки залежності (розміру ядра згортки) VS (розмір зображення) VS (FPS ООЗ/FPS cv::filter2d(…)) – (рис. 1).

Як видно з поданого результату, показник швидкодії має дещо спільне з результатами [3]. Але сам показник швидкодії у 1,8 раз менший ніж просто заміри обробки зображення. Отже нами був проаналізований код програми і було виявлено, що частину ресурсу на себе забирає акт зчитування зображення з відеофайлу (за допомоги valgrind). Нами було зроблено припущення, що частина «кода», що робить збір/формування зображення з відеофайлу, має бути написана за допомогою бібліотеки FFmpeg в ручну, без використання OpenCV. Але було

виявлено факт, що OpenCV вже використовує оптимальний підхід, щодо збіру зображення за допомогою бібліотеки FFmpeg. Більш того, ця частина бібліотеки (OpenCV) виконує це у багато поточному режимі, що позитивно впливає на продуктивність і не вносить суттєвих похибок у акт заміру FPS (навіть сприяє).

Деякі особливості бінарного/вихідного файлу слід також оголосити. Нами було прийнято рішення, яке стосується методики забору фреймів з відеофайлу, яке полягає у тому, що ми проводили збір, наступного для обробки, фрейму та ООЗ у паралельному режимі. Може здатися що це не є важливо, але таким чином ми намагалися зменшити вплив часової похибки на показник FPS при зборі даних для обох підходів.

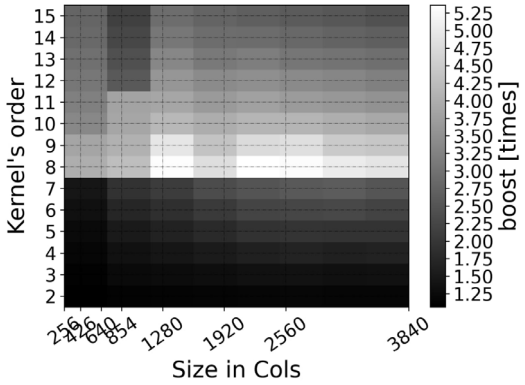


Рис. 1. Графік демонструє показник Швидкодії ООЗ від cv::filter2d(…). Ox ((Size in Cols) - розміру ядра згортки); Oy ((Kernel's order) - розмір ширини зображення); швидкодія (FPS ООЗ/FPS cv::filter2d(…)) виражена чорно-білим градієнтом (білий – краще).

Також у розрізі цього додаткового дослідження було помічено досить цікаву особливість роботи бібліотеки OpenCV (при зборі фреймів). Нами було реалізовано програму, що збирає фрейми у паралельному режимі, але номери фреймів йшли не один за одним а у форматі:

$$n(^{fr\_num}/_{N}) + i \qquad (1)$$

де $N$ – кількість потоків, $n$ – номер потоку *[0..N-1]*, *fr_num* – кількість фреймів відео файлу, $i$ – номер наступного фрейму (для кожного «потоку зчитування» номер свій («стан гонки» за можливість зчитати з відео-файлу фрейм)). Отже ключовою була функція, що забезпечувала таку можливість - videoCapture.set(cv::CAP_PROP_POS_FRAMES, i). Намагання були досить змістовними, кожен з потоків має свою чергу пар (фрейм та його номер). «Обробляючий потік» (у єдиному числі) збирав з найбільш «насиченої» черги (кожен «потік зчитування» мав свою чергу) частину фреймів для обробки, щоб збалансувати навантаження на акт зчитування. Але результат був дещо

неочікуваний. Виявилося, що ця стратегія призводить до зменшення показників FPS саме через функцію videoCapture.set. Причиною є те, що бібліотека FFmpeg мала кожного разу зміщати вказівник файлу на відстань (1), що відповідало точному розташуванню фрейму. Саме це й призводило до зменшення показників FPS.

Але слід відмітити що стратегія, що надана віще є вірною. Якщо змінити підхід, та почати збирати кожен наступний фрейм, то виявляється що швидкість програми набагато збільшується, а саме та її частини, що відповідає за формування черги пар. Але ця частина є темою подальших досліджень.

**Висновки.**

Було проаналізовано швидкодію ООЗ при обробці відео потоку. Було виявлено факт суттєвого зниження швидкодії через частину коду що зчитує зображення з відеофайлу та передає на обробку. Також було отримано кореляцію отриманих результатів з результатом дослідження [3]. Було окремо проаналізована швидкодія ООЗ та cv::filter2d(…) без впливу частини програми що зчитує зображення з відеофайлу – результат співпадає з показниками наданими у [3].

## Список літератури


1. Zhu, Ligeng, et al. "Sparsely aggregated convolutional networks." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
2. Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." Proceedings of the IEEE international conference on computer vision. 2017.
3. Shevchenko, Andrii, Pylyp Prystavka, and Vitalii Tymchyshyn. "Research on Possible Convolution Operation Speed Enhancement via AArch64 SIMD." International Conference on Computer Science, Engineering and Education Applications. Springer, Cham, 2022.