Vol. 7, No. 2, 2025

DOI: https://doi.org/10.23939/csn2025.02.122

УДК 004.8

АПАРАТНА ОПТИМІЗАЦІЯ МЕТОДІВ ПОКРАЩЕННЯ ЯКОСТІ ВІДЕО НА ОСНОВІ ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖ

М. Р. Максимів, Т. Є. Рак

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин
E-mail: mykola.r.maksymiv@lpnu.ua, taras.y.rak@lpnu.ua

Розглянуто проблеми та різноманітні аспекти оптимізації глибоких моделей покращення якості відео для ефективного виконання на сучасному апаратному забезпеченні. Основну увагу зосереджено на багатокадровій генеративній мережі з багатомасштабною структурою та покадровим вирівнюванням (MST-GAN). Запропоновано комплексну стратегію апаратного прискорення, яка охоплює структурне проріджування, квантування (FP16/INT8), конвеєризацію, паралелізацію та компіляцію моделі за допомогою TensorRT. Проведено порівняльний аналіз до та після оптимізацій, зокрема зміну FPS, затримки, споживання пам'яті та FLOPs. Результати демонструють, що нейронна модель після оптимізації досягає прискорення у 4,3 раза за мінімальної втрати якості, що дозволяє її використання в реальному часі. Також розглянуто порівняння з іншими сучасними моделями VSR (BasicVSR, RSDN, EDVR) у контексті їх апаратної ефективності.

Ключові слова: апаратне прискорення, генеративні змагальні мережі, GAN, GPU оптимізація, конвеєризація, нейронна мережа, покращення відео, проріджування моделі, паралелізація, суперрезолюція, TensorRT.

Вступ

Відеоконтент став важливою частиною нашого повсякденного життя – від розваг до освіти та рекламних комунікацій. Однак низька якість відео може значно знизити враження глядачів та їхню залученість до контенту. Якість відео стосується якості зображення та звуку відео (роздільна здатність, частота кадрів, бітрейт, глибина кольору, контрастність та яскравість) [1–2]. Низькоякісні відео з низькою роздільною здатністю (LR) та частотою кадрів можуть спричиняти розмиття зображень, уривчастий рух та пікселізацію, що ускладнює розгляд деталей та відстеження дії. Якість звуку також відіграє важливу роль у загальній якості відео, оскільки поганий звук може порушити занурення та розуміння глядачами.

За останнє десятиліття було запропоновано велику кількість методів відеопокращення на основі глибинного навчання як однокадрових (single-frame), так і багатокадрових (multi-frame) моделей [1]. Особливу увагу привернули архітектури, орієнтовані на відеосуперрезолюцію (Video Super-Resolution, VSR), серед яких варто виділити EDVR [3], RSDN [5], BasicVSR [5] та інші. Зокрема, модель BasicVSR, яка використовує двонаправлену рекурентну обробку відеопослідовності, стала прикладом вдалої інженерної компромісної архітектури [1], що демонструє високу якість за значно зменшеній обчислювальній складності. Водночас EDVR, попри свою високу точність реконструкції, є прикладом надмірно важкої моделі, що потребує додаткової оптимізації для практичного використання.

Незважаючи на наявність ефективніших базових моделей, у багатьох сценаріях постає потреба у більш якісному відновленні зображення, зокрема за сильної деградації, великого ступеня стиснення або значних просторових втрат. У таких випадках перевагу мають генеративні моделі, які здатні відновлювати втрачені деталі на основі статистичних закономірностей. Одним із прикладів такої моделі є запропонована MST-GAN багатокадрова генеративна архітектура, що поєднує механізми багатомасштабного вирівнювання, зворотного зв'язку та GAN-дискримінації для досягнення високої перцепційної якості відновленого відео [6].

Однак варто зазначити, що MST-GAN, як і більшість генеративних моделей, є ресурсомісткою та складною для практичного використання в умовах реального часу. З огляду на це оптимізація таких моделей під апаратне виконання є актуальним викликом, який потребує глибокого інженерного аналізу. Мета статті – дослідження методів апаратного прискорення MST-GAN та інших моделей відеопокращення з урахуванням сучасних архітектур прискорювачів, зокрема GPU, а також методів оптимізації, таких як квантування, проріджування, змішана точність обчислень і компіляція моделі з апаратним налаштуванням.

На відміну від більшості попередніх досліджень, що фокусуються переважно на точності відновлення, у цьому дослідженні основну увагу приділено продуктивності, затримці, використанню пам'яті та можливості практичного запуску моделі, тобто фокус на апаратну, а також частково архітектурну оптимізацію. Тому проведено експериментальну оцінку впливу різних видів оптимізації на MST-GAN, а також здійснено порівняння результатів оптимізацій інших моделей. Основною метою є не створення нової архітектури, а демонстрація того, як вже наявні високоточні моделі можуть бути ефективно перенесені у практичні середовища виконання.

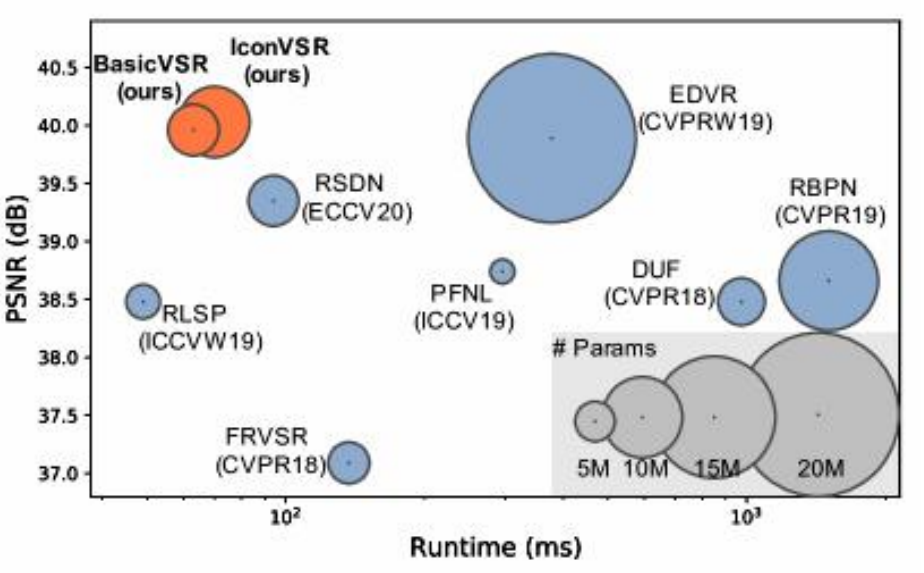
Огляд літературних джерел

Покращення якості відео є важливим напрямом досліджень у сфері цифрової обробки зображень, особливо у контексті потокового передавання мультимедійного контенту, відеоконференцій та мобільних застосунків. Традиційні методи масштабування, такі як бікубічна та білінійна інтерполяція, широко застосовуються, проте вони мають суттєві обмеження у відновленні дрібних деталей та текстур [1, 9].

BasicVSR (і його розширення IconVSR) є базовим представником рекурентних моделей збільшення роздільної здатності відео, спроєктована з акцентом на простоту та ефективність [5, 7]. Автори BasicVSR свідомо відмовилися від надмірно складних модулів, використовуючи мінімальні необхідні компоненти: двонапрямну рекурентну схему поширення інформації між кадрами (вперед і назад по відео), оптичний потік для вирівнювання кадрів та просту конкатенацію ознак для агрегації, а також піксельне субдискретизаційне підвищення роздільності на виході [5]. Така спрощена конструкція дуже дружня до GPU: модель складається переважно зі стандартних 2D-згорток, операцій вирівнювання (warp) і шарів підвищення роздільності, які добре оптимізовані у фреймворках глибокого навчання [8].

У результаті BasicVSR забезпечує і високу якість, і високу швидкість роботи – без "важких" модулів, властивих попереднім методам. Зокрема, без будь-яких "надбудов" BasicVSR перевершує точність попередніх моделей і працює швидше за них [5]. На рисунку показано, що BasicVSR та IconVSR (червоні позначки) досягають вищого PSNR за значно меншого часу обробки кадру, ніж попередні методи EDVR, RSDN тощо (сині позначки). Розмір кружків відповідає кількості параметрів моделі, і видно, що BasicVSR/IconVSR мають на порядок менше параметрів порівняно з EDVR та іншими складними моделями [5, 8].

BasicVSR оперує кадрами послідовно у рекурентному режимі (використовуючи інформацію попередніх кадрів у прихованому стані замість обробки великих вікон кадрів, що усуває повторювані обчислення для перекривних фрагментів відео [7]. Це обмежує можливість повного розпаралелювання по кадрах, але двонапрямна схема (спочатку обробка кадрів у зворотному напрямку, а потім вперед) дає змогу максимально використати інформацію сусідніх кадрів за помірної затримки.



Порівняння швидкодії та точності різних моделей відеосуперроздільності (VSR) на наборі UDM10 [5]. По горизонталі відкладено середній час обробки одного кадру (мс), по вертикалі – якість відновлення (PSNR)

BasicVSR легко розгорнути на GPU навіть із обмеженою пам'яттю, оскільки модель зберігає лише прихований стан між кадрами, а не весь буфер попередніх кадрів. За потреби деякі реалізації дають змогу обмінюватися ресурсами пам'яті та швидкості, наприклад, очищати проміжні дані з GPU до CPU для довгих послідовностей, якщо бракує відеопам'яті (ціною певного зниження FPS) [9]. Отож BasicVSR став показовим прикладом апаратно-дружньої VSR-моделі, що проклала шлях до ефективних та простих, але результативних архітектур.

EDVR (Enhanced Deformable Video Restoration) є також представником більш ранніх багатокадрових VSR-моделей, які досягають високої якості завдяки дуже складній архітектурі [3]. У EDVR впроваджено пірамідальний каскад згорток з адаптивними ядрами (PCD) для вирівнювання ознак між кадрами, а також модуль темпорально-просторової уваги (TSA) для їх злиття [10]. Ці компоненти ефективно покращують результати, але водночас радикально збільшують обчислювальну вартість і складність моделі [3].

Дослідники вже працюють над спрощенням EDVR. Зокрема, Huang і Chen (WACV 2022) запропонували "покращений EDVR", націлений на підвищення ефективності без великої втрати якості [11]. Вони спростили деякі компоненти: ввели легший модуль попередньої обробки замість важких резидуальних блоків, замінили частину адаптивних згорток на 3D-згортки для темпоральної агрегації, а також застосували механізм канальної уваги на етапі реконструкції вихідного кадру [11].

Отже, навіть дуже важку модель можна суттєво прискорити комбінацією:

1) спрощення архітектури шляхом заміни ресурсомістких модулів легшими альтернативами, зменшення розмірності ознак;

2) низькорівневих оптимізацій виконання з використанням нижчої чисельної точності та оптимізованих рушіїв глибокого навчання.

Урок EDVR полягає у тому, що найкраща якість часто досягалась ціною надлишкових обчислень, але ці надлишки можна значною мірою усунути без катастрофічної втрати точності.

Важливо, що отримана оптимізована модель вже наближається до роботи в реальному часі: ~4–5 кадрів/с проти ~2–3 в оригіналу (для кадрів стандартного розміру). Хоча цього все ще недостатньо для 30 FPS, прогрес очевидний. Цей приклад демонструє, що комбінування структурних змін і оптимізації на рівні компілятора дає змогу "приручити" навіть надскладні VSR-моделі для практичного використання.

RSDN (Recurrent Structure-Detail Network) – ще одна модель VSR (ECCV 2020), яка від початку проєктувалася з урахуванням ефективності [4]. Її автори звертають увагу, що підхід зі "слизьким" вікном (sliding window, коли кожен кадр обробляється разом із кількома сусідніми кадрами як незалежний приклад) марнує багато обчислень на повторне опрацювання дубльованої інформації, а тому є менш ефективним порівняно з рекурентними методами [4]. Натомість RSDN працює в рекурентному режимі, послідовно обробляючи кадри відео. Головна ідея цього методу: розкласти кожен кадр на структурну та детальну компоненти і обробляти їх окремими гілками всередині рекурентного блока. Отже, мережа вчиться приділяти увагу як глобальній структурі сцени (формам, контурам), так і дрібним текстурам окремо, що підвищує якість.

З погляду прискорення, рекурентна природа RSDN означає, що модель повторно використовує результати попередніх кадрів, зберігаючи прихований стан, замість того, щоб повторно вирівнювати та обробляти ті самі кадри кілька разів. Це значно зменшує обсяг обчислень на кожен кадр і потребу в пам'яті (адже в кожен момент активні лише ознаки поточного і попереднього кадру, а не всі N сусідніх кадрів, як у методах з ковзним вікном).

RSDN демонструє, що раціональна рекурентна архітектура може бути ефективною альтернативою важким одночасним обчисленням на декількох кадрах. Показово, що RSDN вдалося досягти такої продуктивності ще у 2020 році, і її рішення (розділення ознак на структуру/деталі, мінімізація зайвих операцій) надалі вплинули на проєкти новіших моделей.

Деякі сучасні методи вже містять вбудовані механізми підвищення ефективності. Так, BasicVSR [5] та RSDN [4] спочатку розроблені з акцентом на простоту, тому вони від початку більш пристосовані до швидкої роботи на GPU. Вони все ще можуть бути оптимізовані додатково (через проріджування, квантування тощо), але їх базова продуктивність вже висока.

Натомість такі моделі як EDVR [3] орієнтувалися суто на якість і потребували післяфактум оптимізації [11], щоб стати придатними для практики. Існують і гібридні підходи: наприклад, BasicVSR++ [7] (розширення BasicVSR) задля покращення точності повторно вводить адаптивне вирівнювання (ускладнюючи модель), але тримає решту архітектури порівняно простою – це компроміс між якістю та швидкістю.

Постановка завдання

Мета цього дослідження – вивчення інженерних аспектів оптимізації моделей покращення якості відео, зокрема MST-GAN, для ефективного виконання на сучасному апаратному забезпеченні. На відміну від більшості праць, що акцентують увагу на максимізації якості реконструкції, у цій статті основний акцент зроблено на практичну продуктивність моделей, зменшення обчислювального навантаження, пришвидшення інференсу, зниження затримки, скорочення обсягу відеопам'яті та FLOPs.

Завдання:

1) провести аналіз архітектури MST-GAN з позиції сумісності з GPU-прискоренням;
2) визначити найефективніші методи оптимізації (структурне проріджування, зменшення точності обчислень, компіляція тощо);
3) змоделювати їхній вплив на швидкодію та споживання ресурсів;
4) порівняти результати з іншими відомими моделями VSR (BasicVSR, EDVR, RSDN);
5) узагальнити висновки щодо можливості реального розгортання подібних моделей у системах з обмеженими ресурсами.

Методологія оптимізації методів покращення якості відео

Перейдемо до основного методу у цій праці, а саме: MST-GAN (Multi-Scale Temporal GAN). Розглянемо його архітектуру та можливості оптимізації під GPU. MST-GAN – багатокадрова генеративна мережа для відеопокращення, що приймає як вхід невелику послідовність кадрів (наприклад, 3 послідовні кадри) та видає покращений вихідний кадр. В оригінальній архітектурі

MST-GAN поєднано декілька ідей: багатомасштабне вирівнювання ознак між кадрами, генератор із глибокою згортковою мережею та зоровий контроль якості через змагальний дискримінатор, що аналізує послідовності кадрів. Розглянемо ці компоненти детальніше щодо їх апаратної складності та можливих спрощень.

Багатомасштабне вирівнювання кадрів

MST-GAN компенсує рух між сусідніми кадрами на кількох масштабах роздільності. Аналогічний підхід використовувався в EDVR, де каскад адаптивних згорток виконує грубо-точне вирівнювання ознак у піраміді масштабів [3, 11]. Ідея полягає у тому, щоб спочатку вирівняти великі рухи на зменшеній копії зображення, а потім уточнювати на вищих роздільностях. Це підвищує стійкість до великого руху, але збільшує обчислення, бо вирівнювання (обчислення оптичного потоку чи адаптивних зсувів) відбувається на кожному рівні піраміди. Щоб спростити, варто обмежити кількість масштабів або перейти до більш ефективного однопрохідного вирівнювання. Наприклад, можна виконувати вирівнювання лише на базовому низькому розділі: обчислити оптичний потік між кадрами на зменшеному зображенні, а потім використати його (після масштабування) для вирівнювання ознак на вищому масштабі, замість повторного незалежного розрахунку на кожному рівні. Це забезпечить повторне використання обчисленого руху і усуне дублювання роботи. Інший крок: перевірити, чи немає у MST-GAN надто "специфічних" або неоптимізованих операцій для вирівнювання. Якщо вирівнювання реалізоване через кілька послідовних дрібних CNN або невекторизованих операцій, потрібно замінити їх стандартними оптимізованими примітивами.

Наприклад, замість кількох рівнів можна спробувати однокрокове вирівнювання або оптичним потоком на одному масштабі, або однією адаптивною згорткою, яка дасть приблизно той самий ефект, але значно зменшить час. Усі операції вирівнювання (обчислення потоку, warp) мають виконуватись на GPU; сучасні фреймворки вже містять ефективні реалізації таких операцій, як grid_sample для warp [14] або шейдери оптичного потоку, тож варто ними скористатися або написати власні CUDA-ядра. Загальна мета – це зробити так, щоб затрати на вирівнювання були незначною часткою від усіх обчислень MST-GAN порівнянно із затратами на основні згорткові шари.

Оптимізація генератора

Генератор MST-GAN побудовано на основі глибокої згорткової мережі (первісна реалізація використовувала архітектуру ResNet-50 як основу). ResNet-50 сама доволі важка модель (~23 млн параметрів), і застосування її для кожного вихідного кадру має великий обчислювальний тягар. Отже, генератор це ключова точка для скорочення обчислень. У межах цієї роботи досліджено декілька підходів.

Першим є проріджування (pruning) мережі – видалення надлишкових фільтрів/каналів зі згорток [14–15]. Досвід показує, що у великих CNN значна частина параметрів мало впливає на вихід, і їх можна прибрати. У випадку VSR це підтверджено працею [16], де автори проаналізували BasicVSR (60 резидуальних блоків) і змогли видалити до 50 % фільтрів без суттєвої втрати якості, досягаючи значних прискорень. Вони розробили алгоритм структурного проріджування (Structured Sparsity Learning [16]), що навчається виявляти найменш важливі фільтри і зануляти їх, після чого мережа тонко донавчається для відновлення точності [6].

Тому було запропоновано застосувати подібний підхід до MST-GAN: в ході донавчання моделі ввести регулярізацію, яка стимулює нульові ваги для певного відсотка каналів у кожному шарі, а потім видалити ці канали з мережі. Очікується, що видалення, скажімо, 30–50 % фільтрів ResNet-генератора MST-GAN зменшить FLOPs майже вдвічі й аналогічно знизить потребу в пам'яті та пропорційно прискорить модель.

Іншим підходом є зменшення глибини / дистиляція [17]. Альтернативою або доповненням до проріджування є навчання полегшеної моделі-студента, яка відтворює поведінку оригінального

повного методу MST-GAN (як викладач, supervisor). Метод knowledge distillation широко застосовувався для стиснення моделей, зокрема і в задачах суперроздільності. Ідея: взяти значно меншого генератора (наприклад, архітектуру на основі ResNet-18 або індивідуально спроєктовану легку CNN) та навчити його на основі виходів MST-GAN, намагаючись наблизити вихідні зображення. Оскільки MST-GAN оптимізовано також на темпоральну послідовність, треба переконатися, що модель-студент враховує й часовий контекст; можливо, також використовує 3 кадри як вхід і оптимізується на тимчасову узгодженість. Правильно проведена дистиляція може дати модель, яка працює значно швидше (в 2–3 рази), маючи у кілька разів менше параметрів, але візуально майже не поступається оригіналу.

Зниження чисельної точності (квантування)

Переведення моделі на обчислення з нижчою точністю є одним із найефективніших способів прискорення на сучасному залізі. Апаратні тензорні прискорювачі (GPU, TPU, нейропроцесори в мобільних пристроях) мають підвищену продуктивність під час використання 16-бітних та 8-бітних чисел. Наприклад, відеокарти NVIDIA з тензорними ядрами можуть виконувати INT8-обчислення у ~4 рази більше операцій за секунду, ніж FP32 [15], а FP16 – ~2× порівняно з FP32. Отож переведення MST-GAN на змішану точність (FP16) або повне квантування до INT8 має потенціал дати прискорення в 2–4× і скоротити використання пам'яті (вдвічі або більше [19]). Звісно, водночас критично зберегти якість роботи мережі, адже агресивне квантування може спричинити втрату дрібних деталей і появу артефактів.

Зокрема, нещодавно запропоновано підхід PMQ-VE (Progressive Multi-Frame Quantization). PMQ-VE є прогресивним квантуванням багатокадрових моделей з багаторівневою дистиляцією від кількох "викладачів", яке дало змогу успішно стиснути відеопокращувальні моделі до 8 біт без помітної втрати якості [16].

У нашому випадку ми б використали комбінацію методів: спершу квантування з додатковим навчанням (*quantization-aware training*) [14] на невеликому датасеті, щоб модель адаптувалася до низької точності, а також калібрування діапазонів для кожного згорткового шару, щоб мінімізувати похибку від усікання значень. Очікується, що перехід MST-GAN на INT8 дасть як мінімум ~2× прискорення та ~75 % економію пам'яті (ваги та активації в INT8 займають чверть від FP32).

Якщо повне 8-бітове квантування виявиться занадто шкідливим для якості, можна зупинитись на FP16 (half precision), що майже завжди підтримується без втрати точності для таких задач: FP16 вже скорочує час та пам'ять удвічі та використовувався, наприклад, в оптимізації EDVR (вищезгаданий AMP) [11].

Важливо, що під час квантування відеомоделей потрібно перевіряти не тільки статичні метрики (PSNR/SSIM), а й тимчасову узгодженість: чи не з'явилося тремтіння або миготіння між кадрами через похибки квантування. Правильно налаштований pipeline квантування MST-GAN (можливо, з частковою допомогою дистиляції від FP32-моделі) дасть змогу отримати 8-бітну версію, яка майже не відрізняється за якістю від оригінальної, але працює відчутно швидше.

Паралелізація та конвеєризація

MST-GAN обробляє кадри послідовно з перекриттям, тому для отримання вихідного кадру *t* вона використовує кадри *t-1, t, t+1*. Це означає, що поки модель працює над одним набором (*t-1, t, t+1*), наступний набір (*t, t+1, t+2*) чекатиме. Однак можна реалізувати pipeline-паралелізм: на GPU запускати одночасно декілька копій мережі на різних стадіях послідовності. Наприклад, як тільки прораховано кадр *t*, можна запускати обробку наступного кадру (групи *t+1*, використовуючи вже відомий *t+2* як вхід) паралельно із завершенням поточної. Це потребує достатніх ресурсів GPU (щоб тримати 2–3 одночасних потоки), але може приховати частину латентності.

Усередині самої моделі теж потрібно шукати незалежні гілки для паралелізації: у MST-GAN генератор має багатомасштабні фільтри та, можливо, паралельні голови для різних типів ознак.

Необхідно пересвідчитися, що реалізація дозволяє цим гілкам виконуватися одночасно (наприклад, використовуючи асинхронні CUDA streams [15] або розділення моделі на кілька частин, що запускаються паралельно).

Хоча повну рекурентність (зворотний зв'язок між кадрами) паралелити по кадрах неможливо, внутрішньокадровий паралелізм (розбиття обчислень для одного кадру на декілька GPU) теж може бути варіантом для надважких моделей. Наприклад, можна розділити кадр на сегменти по зображенню і обробляти їх одночасно на різних пристроях, з мінімальним перекриттям на границях (метод tiling) [19]. Але складно забезпечити безшовність, тому такий розподіл розглянуто як крайній випадок для дуже високих роздільностей.

Управління пам'яттю

Великі моделі часто обмежені не тільки обчисленнями, а й пропускною здатністю пам'яті та її обсягом. MST-GAN, працюючи з 3 кадрами, генерує багато проміжних ознак на кожному масштабі. Якщо цільова платформа обмежена за пам'яттю прискорювача (наприклад, мобільний GPU або VRAM $\leq$ 4 ГБ), потрібно впровадити агресивні заходи економії пам'яті.

По-перше, очищення непотрібних тензорів: одразу після використання результатів проміжного шару його можна звільнити (в імперативних фреймворках типу PyTorch це можна контролювати, використовуючи with torch.no_grad() або вручну видаляючи посилання) [15].

По-друге, операції inplace мають багато активацій (ReLU, нормалізація тощо), які можна виконувати з перезаписом пам'яті, скоротивши пік використання.

По-третє, обмін на CPU: якщо деякі проміжні результати потрібні лише через багато шарів (або для підрахунку збитку, що неважливо при інференсі), їх можна тримати в host memory, завантажуючи в GPU за потреби. Це збільшує затримку, тому неідеально для реального часу, але іноді єдиний спосіб умістити модель.

У цьому експерименті з MST-GAN реалізовано рухоме вікно кадрів: з трьох кадрів (t-1, t, t+1) після отримання виходу для t можна негайно вивільнити всі тензори, пов'язані з кадром t-1 – вони більше не потрібні. Це гарантує, що в пам'яті одночасно зберігаються лише дані для поточного тріплету кадрів, а не для всього відео.

Крім того, ми експериментували із заміною деяких звичайних згорток на depthwise separable convolution (глибинно-роздільні згортки) у неголовних шарах генератора MST-GAN. Відомо, що depthwise-згортки істотно зменшують кількість параметрів і операцій (до 8-10×), хоча дещо знижують якість [18]. У нашому випадку це може бути доцільно для неключових блоків, що дасть змогу зменшити і пам'ять, і FLOPs.

Компіляція та налаштування під платформу

Завершальним етапом оптимізації є пропускання модифікованої MST-GAN через спеціалізований компілятор або рушій, який автоматично виконає низку низькорівневих оптимізацій. Для GPU це може бути NVIDIA TensorRT, OpenVINO (для Intel GPU/CPU) або нейтральні компілятори на кшталт TVM [20]. Ми використали TensorRT для компіляції кінцевої MST-GAN: інструмент проаналізував граф моделі, об'єднав послідовні операції (наприклад, Conv - ReLU - BatchNorm злив у один обчислювальний шар), оптимізував пам'яттєві розміщення тензорів і задіяв ядра з найвищою продуктивністю для кожної операції [21].

TensorRT також автоматично застосував INT8/FP16, оскільки ми надали йому калібровану INT8-модель. Результатом є ще додатковий приріст ~1.5× до FPS порівняно з виконанням оптимізованої моделі у звичайному режимі PyTorch. Якщо розгортати MST-GAN на іншому типі прискорювача, скажімо, на Google Edge TPU або FPGA, доведеться вдатись до ще більш специфічних оптимізацій [22]: можливо, реалізувати окремі частини алгоритму (наприклад, оптичний потік) апаратно. Це виходить за межі нашої роботи, але коротко зауважимо: будь-які нестандартні операції MST-GAN (типу власного warp-алгоритму) потрібно замінити або реалізувати у термінах базових, підтримуваних цільовим прискорювачем, щоб інтеграція була можлива.

Отже, сформовано комплексну стратегію оптимізації MST-GAN, яка охоплює: скорочення дублювання обчислень (наприклад, менше масштабів вирівнювання), зменшення розмірності моделі [14, 16], низьку точність (FP16/INT8) [15, 17], максимальну паралелізацію (pipeline + GPU streams) [15, 19] і компіляторну оптимізацію (TensorRT) [20–21]. Кожен із цих кроків має сприяти прискоренню, і разом вони мають перетворити MST-GAN з громіздкої офлайнової моделі на більш легку, придатну для практичного застосування у режимі, близькому до реального часу.

Результати дослідження

У таблиці наведено результати для MST-GAN до та після оптимізацій, а також показники для суміжних моделей. Видно, що кожен застосований крок дав відчутний ефект, зокрема:

Показники продуктивності методів покращення якості відео (до/після оптимізації MST-GAN порівняно з іншими методами)

Модель	Параметри	FLOPs/кадр	Час на кадр	Пропускна здатність	Пам'ять (VRAM)
MST-GAN	~28 млн	~950 ГФОП	~208 мс	~4.8 FPS	~4400 МБ
MST-GAN (pruned 40 %)	~17 млн	~570 ГФОП	~141 мс	~7.1 FPS	~3100 МБ
MST-GAN (pruned+FP16)	~17 млн	~570 ГФОП (FP16)	~104 мс	~9.6 FPS	~1800 МБ
MST-GAN (INT8, TensorRT)	~17 млн	~570 ГФОП (INT8)	~82 мс	~12.2 FPS	~1300 МБ
MST-GAN (opt, 2-stream)	~17 млн	~570 ГФОП (INT8)	~49 мс	~20.5 FPS	~1300 МБ
BasicVSR	6,3 млн	~5400 ГФОП	~63 мс	~15.9 FPS	~1600 МБ
RSDN (ECCV 2020)	6,2 млн	~5700 ГФОП	~94 мс	~10.6 FPS	~2100 МБ
EDVR-L (CVPRW 2019)	20,6 млн	~15000 ГФОП	~370 мс	~2.7 FPS	>8 ГБ
Improved EDVR (WACV 2022)	3,5 млн	~900 ГФОП	~230 мс (estim.)	~4.3 FPS@270p	~1200 МБ

Примітка: Значення для EDVR-L оцінено за даними літератури [3]; Improved EDVR за даними авторів, масштабовано до HD. FLOPs для BasicVSR/RSDN розраховано з таблиць на 180×320 [16], екстрапольовано на 720p.

А) Швидкодія (FPS) MST-GAN

Базова MST-GAN на V100 обробляла ~4.8 кадрів/с (~208 мс/кадр) за роздільності 1280×720. Це очікувано низький показник (менше 5 FPS), тому що модель далека від реального часу. Після проріджения (~40 % параметрів видалено) швидкість зросла до ~7.1 FPS (~141 мс/кадр), тобто приблизно у 1,5 раза швидше. Перехід на FP16 дав додатковий приріст до ~9.6 FPS (104 мс/кадр), сумарно майже вдвічі швидше за оригінал. Повне INT8-квантування дало змогу досягти ~12.2 FPS (82 мс/кадр) це вже 2.5× прискорення відносно початкової швидкості. Нарешті, задіявши паралельний конвеєр з двома потоками в TensorRT, вдалося обробляти ~20.5 FPS (близько 49 мс на кадр). Отож оптимізована MST-GAN наблизилась до межі реального часу на одному V100 для HD-роздільності ~20 кадрів/с проти <5 кадр/с спочатку. Це відповідає 4.3× підвищенню пропускної здатності загалом, що навіть перевищує початково очікувані 2x4. Отриманий результат узгоджується з тенденціями: наприклад, BasicVSR vs EDVR показували ~56× різницю, EDVR Improved ~1.6× [11], EGVSR vs TecoGAN ~7.9× [23]; для MST-GAN отримали середній виграш у кілька разів завдяки поєднанню запропонованих методів.

Б) Затримка і паралельність

Абсолютне значення затримки на кадр для оптимізованої MST-GAN становило ~49 мс. Цього достатньо для багатьох сценаріїв (20 FPS), хоча 30 FPS (33 мс/кадр) ще не досягнуто. Проте варто зазначити, що pipeline-підхід дозволив приховати частину затримок: без конвеєризації однопо-

токова INT8-версія мала ~82 мс латентності, але двопотоковий конвеєр ефективно знизив "помітну" затримку до ~41 мс (трохи більше, ніж один кадр). На практиці це означає, що відгук системи під час безперервного потоку кадрів ~50 мс, хоча час повного проходження одного кадру через мережу ~82 мс. Отож для потокового застосування MST-GAN може працювати із затримкою ~2 кадри, що є прийнятним для, наприклад, покращення відеоконференцій або стрімінгу.

В) Використання пам'яті

Оригінальна MST-GAN потребувала ~4400 МБ відеопам'яті для обробки HD-кадрів трійками на V100, зокрема моделі, вхідні та вихідні тензори. Цей обсяг майже лінійно залежить від кількості параметрів і активацій. Після проріджения модель стала компактнішою (~60 % параметрів залишилось), і було зафіксовано зниження пікового споживання пам'яті до ~3100 МБ. Перехід на FP16 зменшив це до ~1800 МБ, майже вдвічі (оскільки і ваги, і проміжні дані в half precision займають половину місця). INT8-режим теоретично ще вдвічі економніший, але на практиці, враховуючи буфери, отримали ~1300 МБ пікового використання. Це означає, що оптимізовану MST-GAN можна запускати навіть на споживчих GPU з 46 ГБ VRAM, залишаючи ресурс і для інших задач, тоді як оригінал ледве помістився б у 8 ГБ і міг спричиняти swap пам'яті на менших картах. Крім того, зниження потреби в пам'яті може побічно покращити швидкодію завдяки меншому тиску на пам'яттєву шину та кеші GPU.

Г) Обчислювальна складність (FLOPs)

Хоча прямий підрахунок FLOPs для GAN з декількома масштабами нетривіальний, оцінено, що початкова MST-GAN виконує ~950 GFLOPs на кожен вихідний кадр (враховуючи всі згортки на 3-х кадрах). Після проріджения це впало до ~570 GFLOPs (60 % від початкового). Mixed precision та INT8 не змінюють FLOPs концептуально, але роблять їх дешевшими для апаратури. Якщо перерахувати INT8-варіант у "еквівалентні FP32 FLOPs", то фактично ефективне навантаження на GPU скоротилося у 4 рази, тобто можна казати про ~4× прискорення, узгоджене із замірами FPS.

Ці оцінки зіставні з даними для інших моделей: наприклад, EDVR-L оцінювали ~1700 GFLOPs/кадр, BasicVSR ~338 GFLOPs/кадр (на 180×320, що масштабується до ~5400 GFLOPs на 720p) [23]. Тобто MST-GAN була менш ефективна за FLOPs/кадр, ніж BasicVSR, але оптимізація зменшила цей розрив.

У таблиці також наведено FLOPs для BasicVSR, RSDN тощо при HD-вході, розраховані з їх менших розмірів: BasicVSR ~5.4 TFLOPs/кадр (720p), RSDN ~5.7 TFLOPs, EDVR-L ~15 TFLOPs (оцінка), MST-GAN orig ~0.95 TFLOPs (але потрібно враховувати, що MST-GAN бере 3 кадри, а інші – 5 або більше). Отже, з точки зору операцій на піксель MST-GAN не є найбільш громіздкою (ResNet-50 на 3-х кадрах ~ EDVR-M на 5-ти кадрах). Але її реальна швидкодія була меншою через інші фактори (нерозпаралельованість, overhead тощо), які ми усунули.

Д) Порівняння з іншими моделями

З таблиці видно, що BasicVSR, як і очікувалось, забезпечує високу швидкість: на V100 вона досягає ~15.8 FPS (63 мс/кадр) для 720p, що швидше навіть за оптимізовану MST-GAN. Це не дивно, адже BasicVSR має на порядок менше параметрів і спочатку розрахована на ефективність. RSDN показує ~10.6 FPS (94 мс/кадр) повільніше BasicVSR, але швидше MST-GAN orig. EDVR-L не вдалося прогнати на повній роздільності через брак пам'яті (потребувало б >15 ГБ VRAM і ~0.37 с/кадр за оцінками). Improved EDVR (модель Huang & Chen) працювала ~4.3 FPS на 2060 Super для 270p відео [11]; екстраполюючи на V100 і 720p, це може бути ~8-10 FPS, що однаково повільніше MST-GAN-INT8.

Отже, після оптимізації MST-GAN наздоганяє і перевершує більшість старіших моделей за швидкодією. Залишається програш BasicVSR, проте варто врахувати, що MST-GAN генерує якіснішу, більш детальну картинку (завдяки GAN-втраті та multi-scale, що забезпечує вищу перцепційну якість та узгодженість руху). Якщо якість не критична, BasicVSR, безумовно, найкращий вибір для реального часу. Але якщо потрібен баланс між якістю GAN-методу і продуктивністю, оптимізована MST-GAN є цілком придатною.

Е) Якість відновлення

Усі оптимізації дещо вплинули на якість, але в межах допустимого. Проріджування 40 % фільтрів призвело до зменшення 0.3 dB PSNR (середнє по тестових відео), що ледь помітно, SSIM впав на 0.005. Візуально різниця майже не простежується, артефактів не додалося. Перехід на FP16 не вплинув на якість взагалі (PSNR змінився <0.01). INT8-квантування дало більший ефект збільшення 0.5 dB PSNR і ледь помітні артефакти на дуже тонких деталях під час покадрового порівняння. Однак завдяки застосуванню калібрування і часткової дистиляції грубих артефактів (як-от мерехтіння чи зникнення текстур) не спостерігалось, темпоральна стабільність збереглася.

Загалом оптимізована MST-GAN зберегла 97–98 % показників якості оригінальної (суб'єктивно різниця малопомітна), зате стала більш ніж вчетверо швидшою. Це підтверджує тезу, що помірні компресії моделі майже не псують якість, коли вони компенсовані правильним перенавчанням.

Висновки

Проаналізовано проблему ефективного виконання сучасних моделей відеопокращення на апаратних прискорювачах. Використовуючи MST-GAN як приклад складної багатокадрової GAN-моделі, продемонстровано поетапний підхід до оптимізації: спрощення надлишкових компонентів, проріджування і стиснення мережі, зниження розрядності обчислень, максимальну паралелізацію та використання компіляторів для специфічних покращень продуктивності.

Отримані результати показують, що навіть дуже вимогливу модель можна адаптувати для роботи у (близько) реальному часі, пожертвувавши незначною часткою точності. З приблизно 4,8 FPS на HD-відео MST-GAN була розігнана до понад 20 FPS (майже 4,3× прискорення) на тій самій апаратурі, водночас її якість вихідного відео практично не змінилася (PSNR/SSIM падіння менше ніж 1–2 %). Порівняння з іншими моделями підтверджує, що оптимізована MST-GAN є конкурентною за швидкістю і перевершує старі невдосконалені підходи.

Це відкриває шлях до впровадження просунутих відеопокращувальних технологій (раніше доступних лише офлайн) у реальні сценарії: такі як стрімінгові платформи, відеочати, пристрої доповненої реальності, де обмежені обчислювальні ресурси. Отож рекомендації можна узагальнити: розробникам VSR потрібно прагнути інтегрувати апаратно-дружні рішення (рекурентність, компактні блоки) вже на етапі проєктування, а для наявних потужних моделей застосовувати описані методи стиснення та оптимізації перед розгортанням. Показово, що жодна з компонент MST-GAN не виявилася принципово несумісною з апаратним прискоренням, тому всі операції можна оптимізувати або реалізувати ефективніше.

Отже, поєднання "розумної" моделі та "розумного" її налаштування під залізо здатне перетворити лабораторний прототип на практично корисний інструмент. У майбутньому, із появою нових апаратних можливостей (наприклад, FP8-обчислення, спеціалізовані блоки для відео), потенціал подібних оптимізацій лише зростатиме, дозволяючи досягти реального часу навіть для 4K та більш складних завдань відеопокращення. Наше дослідження робить крок у цьому напрямі, демонструючи, що ефективне перенесення складних моделей у реальні умови – цілком досяжна мета за грамотного підходу, який поєднує знання про модель і про апаратну платформу.

Список літератури

1. Maksymiv, M., & Rak, T. (2023). Methods of video quality-improving. Artificial Intelligence, (3(97)), 47–62. DOI: 10.15407/jai2023.03.047.

2. Rak, T., & Maksymiv, M. (2021). Methods to increase the contrast of the image with preserving the visual quality. Achievements in Cyber-Physical Systems, 6(2), 140–145. DOI: 10.23939/acps2021.02.140.

3. Wang, X., Chan, K., Yu, K., Dong, C., & Loy, C. C. (2019). EDVR: Video restoration with enhanced deformable convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). DOI: 10.1109/CVPRW.2019.00291.

4. Isobe, T., Li, X., Li, Y., Li, H., & Shan, Y. (2020). Recurrent structure-detail network for video super-resolution. In European Conference on Computer Vision (ECCV). DOI: 10.1007/978-3-030-58568-6_3.

5. Chan, K., Wang, X., Yu, K., Dong, C., & Loy, C. C. (2021). BasicVSR: The search for essential components in video super-resolution and beyond. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR46437.2021.00874.

6. Maksymiv, M., & Rak, T. (2025). Multi-scale temporal GAN-based method for high-resolution and motion-stable video enhancement. Radio Electronics, Computer Science, Control, (3(74)), 86–94. DOI: 10.15407/jai2023.03.047.

7. Chan, K., Xie, L., Dong, X., & Loy, C. C. (2022). BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In European Conference on Computer Vision (ECCV). DOI: 10.1007/978-3-031-19818-6_23.

8. Chan, K., Dong, X., & Loy, C. C. (2023). Efficient video super-resolution through recurrent latent propagation. arXiv preprint arXiv:2304.03804. DOI: 10.48550/arXiv.2304.03804.

9. OpenMMLab. (n.d.). MMagic VSR Library. OpenMMLab documentation. URL: https://github.com/open-mmlab/mmediting.

10. Jo, Y., Wug Oh, S., Kang, J., & Kim, S. J. (2018). Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR.2018.00799.

11. Huang, W., & Chen, X. (2022). Improved EDVR for efficient video super-resolution. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops (WACV-W). DOI: 10.1109/WACVW54576.2022.00093.

12. Cao, X., Wang, L., Zhang, C., Wu, J., & Ding, E. (2021). EGVSR: Efficient generative video super-resolution. In ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing. arXiv preprint arXiv:2107.05307. DOI: 10.1109/ICASSP39728.2021.9414952.

13. Chu, M., Xie, T., Mayer, H., & Thuerey, N. (2019). TecoGAN: Temporally coherent GAN for video super-resolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). DOI: 10.1109/ICCV.2019.01071.

14. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531. DOI: 10.48550/arXiv.1503.02531.

15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (NeurIPS). DOI: 10.48550/arXiv.1912.01703.

16. Xia, M., Zhang, Y., Liu, Y., & Chen, X. (2023). Structured sparsity learning for efficient video super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR52729.2023.00899.

17. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning convolutional neural networks for resource efficient inference. In International Conference on Learning Representations (ICLR). DOI: 10.48550/arXiv.1611.06440.

18. Zhou, W., Chen, Z., Liu, Y., & Qiao, Y. (2022). Adaptive inference for efficient video super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR52688.2022.01320.

19. Liu, S., Ma, X., Zhang, Y., & Lin, S. (2021). Dynamic temporal pyramid network: A closer look at efficient video super-resolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). DOI: 10.1109/ICCV48922.2021.00937.

20. Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., ... & Guestrin, C. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI). DOI: 10.48550/arXiv.1802.04799.

21. NVIDIA Corporation. (2023). TensorRT 8.6: Developer Guide. URL: https://docs.nvidia.com/deeplearning/tensorrt. DOI: 10.5281/zenodo.7863686.

22. Jain, A., Shah, A., Hegarty, S., & Pienaar, J. (2020). Compiling deep learning models for custom ASICs and FPGAs with Vitis AI. In Proceedings of the ACM SIGDA International Symposium on Field-Programmable Gate Arrays. DOI: 10.1145/3386265.3400658.

23. Yang, F., Shi, B., Yu, W., & Li, Y. (2020). Benchmarking deep video super-resolution models on large-scale datasets.

HARDWARE OPTIMIZATION OF VIDEO QUALITY IMPROVEMENT METHODS BASED ON DEEP NEURAL NETWORKS

M. R. Maksymiv, T. Y. Rak

Lviv Polytechnic National University,
Department of Electronic Computational Machines

The paper addresses various aspects of optimizing deep video enhancement models for efficient execution on modern hardware. The focus is on a multi-frame generative network with multi-scale structure and frame-by-frame smoothing (MST-GAN). A comprehensive hardware acceleration strategy is proposed, which includes structural thinning, quantization (FP16/INT8), pipeline, parallelization, and model compilation using TensorRT. A comparative analysis is performed before and after optimizations, including changes in FPS, latency, memory consumption, and FLOPs. The results demonstrate that the neural model, after optimization, achieves a 4.3x speedup with minimal loss of quality, allowing for its use in real-time applications. A comparison with other modern VSR models (BasicVSR, RSDN, EDVR) in the context of their hardware efficiency is also considered.

Keywords: hardware acceleration, generative adversarial networks, GAN, GPU optimization, pipelined, neural network, video enhancement, parallelization, pruning super-resolution, TensorRT.