math_2023_021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики

Кафедра математичного моделювання

Використання генеративних змагальних super-resolution мереж для покращення якості зображення інфрачервоних камер в реальному часі

Кваліфікаційна робота

Рівень вищої освіти - другий (магістерський)

Виконав:
студент 6 курсу, групи **607**
Перевозний Андрій Юрійович

Керівник:
професор Черевко І.М.

До захисту допущено:

Протокол засідання кафедри № 9 від 5 грудня 2023 р.

зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2023

Анотація

У даній роботі аналізуються важливі аспекти, пов'язані із вдосконаленням роздільної здатності зображень та відеоінформації. Основна увага дипломної роботи спрямована на використання генеративних змагальних super-resolution мереж для оптимізації візуальної якості інфрачервоних зображень у реальному часі. Обґрунтовується актуальність проблеми низької роздільної здатності та обмеженої частоти кадрів ІЧ-камер. Розроблено застосунок для роботи з ІЧ-камерами на базі ОС Android.

Ключові слова: інфрачервоні зображення, інфрачервоні камери, Android, NDK, C++, Vulkan, OpenCV, Super-Resolution, генеративні змагальні нейронні мережі, SRGAN, RTSRGAN, RealSRGAN, RealESRGAN, згорткові нейронні мережі, CNN, NCNN

The present work analyzes important aspects related to improving the resolution of images and video information. The main focus of the thesis is on the utilization of generative adversarial super-resolution networks to optimize the visual quality of infrared images in real-time. The relevance of the low resolution problem and the limited frame rate of IR-cameras is justified. An application has been developed to work with IR-cameras based on the Android operating system.

Keywords: infrared images, infrared cameras, Android, NDK, C++, Vulkan, OpenCV, Super-Resolution, generative adversarial neural networks, SRGAN, RTSRGAN, RealSRGAN, RealESRGAN, convolutional neural networks, CNN, NCNN.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ А.Ю. Перевозний

Зміст

Вступ

Актуальність роботи

Враховуючи стрімке зростання застосування інфрачервоних камер у різних областях медицини, науки, житлової та безпекової сфер, виникає необхідність вдосконалення їх візуальної продуктивності. На сучасному етапі розвитку, де бюджетні та компактні рішення все ще мають технологічні обмеження в роздільній здатності, високий рівень теплових шумів та низьку частоту кадрів, задача покращення візуальної якості отриманих зображень та відео залишається одним із перспективних напрямків наукових досліджень та розробки ефективних алгоритмів.

Сучасний розвиток super-resolution нейромереж надає зручний та ефективний спосіб зменшити ці технічні обмеження та покращити зручність користування подібними пристроями.

У кваліфікаційній роботі досліджується можливість використання генеративних змагальних нейронних мереж для підвищення якості відображення та збільшення плавності відеоінформації. Особлива увага приділяється розробці алгоритмів реального часу, спрямованих на оптимальне використання сучасних мобільних пристроїв на базі Android

Мета

Дана дипломна робота націлена на розробку та впровадження алгоритмів super-resolution в реальному часі для ІЧ-камер у поєднанні з мобільними пристроями на базі операційної системи Android. Використання тензорних ядер мобільних процесорів дозволить досягти оптимальної швидкодії обчислень, забезпечуючи при цьому стабільність та ефективність в реальному часі [6].

Завдання роботи

Результатом роботи є прикладний додаток, що дозволить покращити ефективність ІЧ-камер, забезпечуючи більш широке застосування в різних сферах, де важлива підвищена деталізація зображення та покращена якість візуального спостереження в реальному часі.

Під час роботи над кваліфікаційною роботою було розв’язано такі задачі:

- вивчення технологій роботи з USB OTG пристроями на базі ОС Android
- розробка застосунку для ОС Android для покращення відео в реальному часі за допомогою мови програмування C++ та Vulkan API.
- Тестування застосунку на пристроях з топовими процесорами Qualcomm Snapdragon 8 Gen 1 та 8 Gen 2.

Об’єкт дослідження

В даний час основними недоліками багатьох ІЧ-сенсорів та ІЧ-камер є низька роздільна здатність (близько 206×156 крапок або навіть нижча), невелика частота кадрів та високий рівень теплових шумів. Всі ці недоліки безпосередньо пов’язані з фізичними обмеженнями технологій виробництва компактних ІЧ-сенсорів без додаткового охолодження, а камери з більшою роздільною здатністю дуже дорого коштують.

Стрімке підвищення потужності мобільних процесорів відкриває перспективи використання super-resolution нейромереж у реальному часі для програмного поліпшення візуальної якості зображень та відео, підвищення плавності (збільшення частоту кадрів) відображення відеоінформації за рахунок розрахунку проміжних відеокадрів.

Практичне застосування

ОС Android на даний момент часу займає близько 70% світового ринку мобільних пристроїв та близько 40-50% ринків у розвинених країнах. Таким чином, використання компактних інфрачервоних камер разом з мобільними пристроями на базі ОС Android за допомогою нейронних мереж реального часу з використання тензорних ядер мобільних процесорів дозволить зменшити розрив між візуальною якістю зображення бюджетних та дорогих ІЧ-камер, покращити деталізацію, зменшить шуми ІЧ-сенсорів та розширить можливості їх застосування у різних сферах людської діяльності, де обмеження роздільної здатності не дозволяють в даний час використовувати бюджетні рішення.

Для камер з більшою роздільною здатністю (від 320×240 мікроболометрів або вищою) ця технологія дозволить отримувати якість відображення подібну до звичайних оптичних HD+ камер в реальному часі.

Окремою проблемою для українських користувачів також є той факт, що деякі країни обмежують експорт високоякісних ІЧ-камер та сенсорів, як товарів подвійного призначення.

Зазначена технологія не тільки вирішить проблему обмежень бюджетних ІЧ-камер, але й розширить їхнє застосування в сферах, де вимоги до візуальної якості та динаміки зображень є високими та постійно зростають. Це допоможе зробити інфрачервоні камери більш доступними та ефективними для різноманітних завдань в реальному часі, таких як біологічні, медичні та фізичні дослідження, роботи у сфері енергозбереження та ліквідації наслідків надзвичайних ситуацій та техногенних аварій.

1. Основні відомості про обробку зображень

1.1 Огляд розвитку super-resolution методів

Супер-розширення зображень - це технології, які використовуються для підвищення роздільної здатності зображень. Це може бути корисно для таких завдань [3, 4].

- Корекція оптичного розмиття: супер-розширення можна використовувати для усунення оптичного розмиття з зображень, наприклад, розмиття, спричинене рухом камери або об'єкта.
- Відновлення зображень: супер-розширення можна використовувати для відновлення зображень, які були пошкоджені або спотворені, наприклад, зображення, які були зашумлені або спотворені.
- Збільшення роздільної здатності відео: супер-розширення можна використовувати для збільшення роздільної здатності відео, наприклад, для створення 4K відео з 1080p відео.

Перші методи супер-розширення використовували прості лінійні алгоритми, такі як інтерполяція. Однак ці методи часто приводили до зображень з низькою якістю, які мали артефакти, такі як шум і різкі краї.

У 1990-х роках були розроблені більш складні методи супер-розширення, які використовували нелінійну обробку зображень. Ці методи були більш ефективними в запобіганні артефактам, але вони все ще не були здатні генерувати зображення високої якості.

У 2010-х роках завдяки розвитку штучного інтелекту були розроблені супер-розширювальні методи, які використовували нейронні мережі. Ці методи були здатні генерувати зображення високої якості, які були важко відрізнити від справжніх.

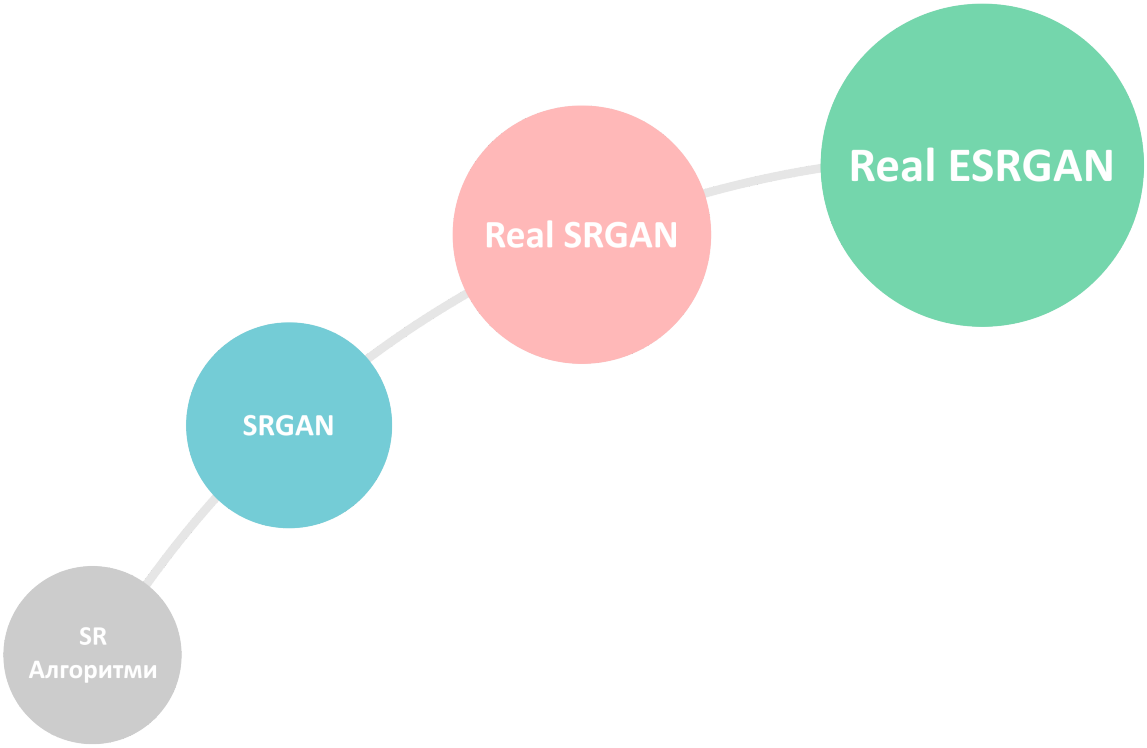

Рис 1. Схема розвитку super-resolution методів

Деякі з найважливіших super-resolution методів, які були розроблені в останні роки, включають [1,2]:

- SRGAN (Super-Resolution Generative Adversarial Network) — це метод супер-розширення, який використовує генеративно-змагальну модель (GAN). GAN складаються з двох нейромереж: генератора та дискримінатора. Генератор намагається створити нові зображення, які неможливо відрізнити від справжніх. Дискримінатор намагається відрізнити справжні зображення від згенерованих [7].
- Real-SRGAN — це модифікація SRGAN, яка була розроблена для підвищення якості зображень, отриманих з реальних камер. Real-SRGAN використовує більш реалістичну модель розподілу зображень, ніж SRGAN.
- Real-ESRGAN є покращеною версією Real-SRGAN, яка була розроблена для підвищення якості зображень, отриманих з реальних

камер і використовує більш складну модель нейронної мережі, ніж Real-SRGAN. Це дозволяє Real-ESRGAN створювати більш реалістичні та детальні зображення. Ця нейромережа використовує більш складний процес навчання, та була протестована на більшій кількості даних, ніж Real-SRGAN.. Це дозволяє Real-ESRGAN краще адаптуватися і демонструвати більш стабільні результати на різних типах реальних фотографій.

1.2 Використання нейромереж для обробки інфрачервоних зображень

Інфрачервоні зображення мають ряд переваг перед видимими зображеннями, наприклад, вони можуть бути отримані в умовах слабкого освітлення або через туман. Однак інфрачервоні зображення часто мають низьку роздільну здатність і обмежену частоту кадрів [8].

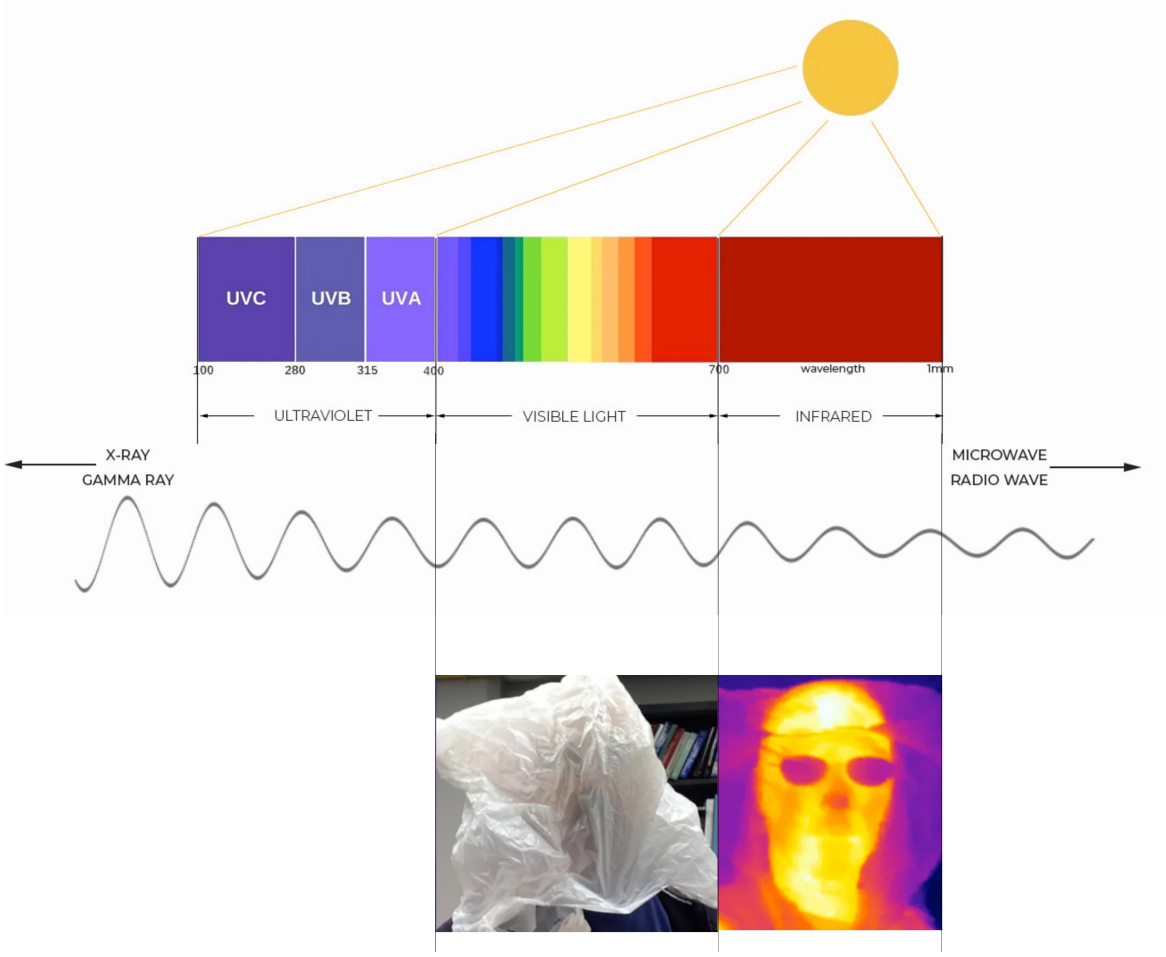

Рис 2. Порівняння видимого та ІЧ зображень

Нейромережі можна використовувати для підвищення роздільної здатності та частоти кадрів інфрачервоних зображень. Нейромережі можуть навчитися виявляти та відтворювати деталі, які не можуть бути легко розпізнані на зображеннях з низькою роздільною здатністю.

Рис 3. Приклад покращення роздільної здатності ІЧ-зображення за допомогою нейромережі Real-ESRGAN

Одним із підходів до використання нейромереж для обробки інфрачервоних зображень є використання генеративних змагальних супер-розширювальних SRGAN мереж.

Нейромережі є перспективним підходом до підвищення роздільної здатності та частоти кадрів інфрачервоних зображень. Вони можуть

використовуватися в різних додатках, таких як охорона, медична діагностика, автономне керування транспортними засобами та ін.

2. Структура та методи генеративних змагальних нейромереж

2.1 Загальна інформація про мережі Real-ESRGAN

Real-ESRGAN (Real-Enhanced Super-Resolution Generative Adversarial Networks – реально-покращені генеративно-змагальні мережі для супер-роздільної здатності) є нейронними мережами, призначеними для поліпшення якості зображень за допомогою техніки супер-роздільної здатності. Вона є розвитком алгоритму ESRGAN[1] і являє собою глибоку генеративно-змагальну мережу (GAN), засновану на архітектурі Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN).

Основна мета Real-ESRGAN полягає в тому, щоб збільшити роздільну здатність низькоякісних зображень, додавши їм деталі та різкість, щоб вони мали реалістичніший і більш деталізований вигляд. Для цього мережа навчається на великому наборі високоякісних зображень, щоб навчитися витягувати загальні структури та особливості зображень.

Процес роботи Real-ESRGAN складається з двох основних компонентів: генератора (generator) та дискримінатора (discriminator). Генератор приймає на вхід низькоякісне зображення і намагається генерувати відповідне йому високоякісне зображення. Дискримінатор, з іншого боку, намагається розрізняти між згенерованими зображеннями та реальними високоякісними зображеннями.

Під час навчання генератор і дискримінатор взаємодіють один з одним і покращують свої навички. Генератор прагне генерувати зображення, які дискримінатор не зможе відрізнити від реальних, а дискримінатор вчиться розрізняти реальні зображення від згенерованих. Цей процес змагання між

генератором і дискримінатором допомагає генератору навчитися генерувати більш якісні зображення.

Рис 4. Порівняння якості відновлення деталей різних типів мереж [2]

Real-ESRGAN має кілька удосконалень порівняно з ESRGAN мережами, включно з використанням попередньо навченої моделі для поліпшення процесу навчання, а також різними техніками, такі як архітектурні зміни та використання допоміжних завдань, щоб поліпшити якість згенерованих зображень.

Деякі з покращень: більш швидка обробка зображень, врахування контекстів з інших кадрів (для відео), застосування залишкових (residual) блоків. Резидуальні (залишкові) блоки допомагають моделі зосередитися на відновленні лише відмінностей між низькороздільним (low-resolution) та високороздільним (high-resolution) зображеннями, що називається "резидуал". Вони використовують переходи (skip connections), щоб додати цей резидуал до вихідного зображення. Це дозволяє моделі фокусуватися на відновленні доданих деталей, а не на повній реконструкції зображення з низькою роздільною здатністю. Застосування резидуальних блоків в Real-ESRGAN допомагає покращити якість високороздільних зображень,

зберігаючи та передаючи важливі ознаки та деталі. Вони також сприяють зменшенню навчального часу та обчислювальних витрат, оскільки модель може сконцентруватися на відновленні тільки резидуалів замість повного зображення.

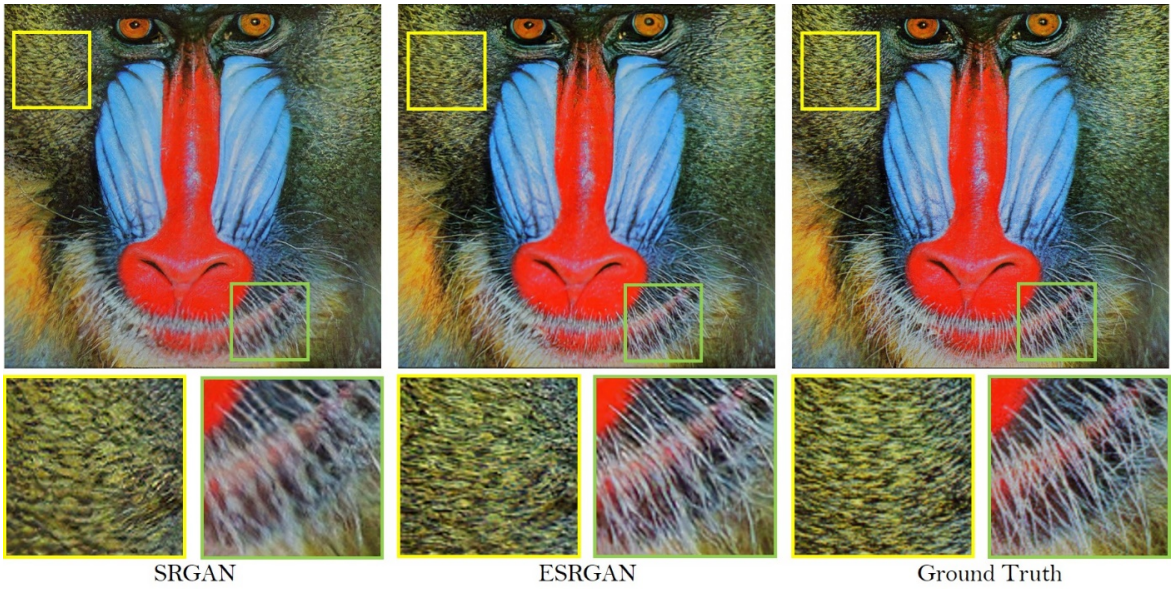

Рис 5. Порівняння якості відтворення деталей (зліва-направо) SRGAN, ESRGAN та оригінального зображення з високою роздільною здатністю [1]

2.2 Особливості та переваги Real-ESRGAN мереж

Real-ESRGAN має кілька ключових особливостей і переваг, які роблять його ефективним у поліпшенні якості зображень:

Глибока архітектура: Real-ESRGAN використовує глибоку нейронну мережу, що складається з безлічі шарів. Це дає змогу моделі вивчити складні залежності в даних і генерувати більш реалістичні та деталізовані зображення.

Генеративно-змагальна мережа (GAN): Використання GAN-архітектури дозволяє Real-ESRGAN створювати зображення, які важко

відрізнити від реальних. Генератор і дискримінатор взаємодіють у процесі навчання, підвищуючи якість згенерованих зображень.

Використання попередньо навчених моделей: Real-ESRGAN може використовувати попередньо навчені моделі для поліпшення процесу навчання. Це допомагає мережі швидше сходитися і генерувати більш якісні зображення.

Аугментація даних і допоміжні завдання: У процесі навчання Real-ESRGAN використовує різні техніки, такі як аугментація даних (наприклад, випадкові повороти й обрізки) і допоміжні завдання (наприклад, передбачення оригінального зображення за його низькоякісною версією). Це допомагає поліпшити здатність моделі до вивчення особливостей зображень і генерації більш точних результатів.

Управління відтворюваністю: Real-ESRGAN забезпечує контроль над процесом відтворюваності результатів, що дає змогу повторно генерувати зображення з однаковими параметрами й отримувати узгоджені результати.

Загалом, Real-ESRGAN є потужним інструментом для поліпшення якості зображень. Він здатний додати деталі, текстури та різкість до низькоякісних зображень, роблячи їх реалістичнішими та придатнішими для різноманітних завдань, такі як візуалізація, обробка зображень та машинний зір.

2.3 Опис роботи Real-ESRGAN мережі [1, 2, 5]

Генератор:

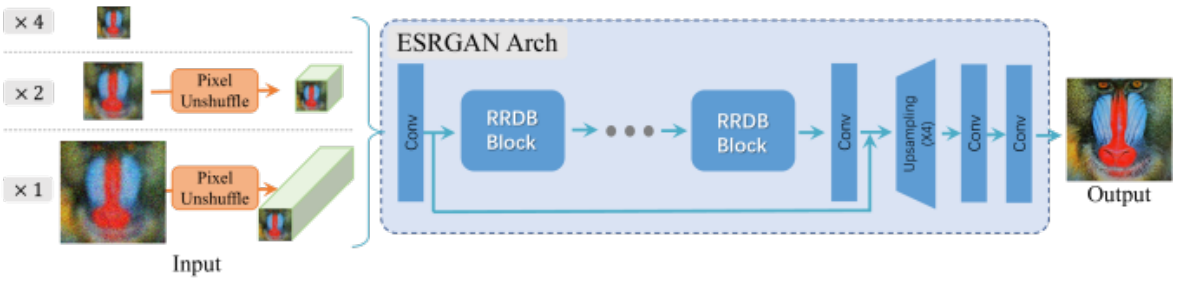

Рис 6. Схема генератора Real-ESRGAN мережі [2]

Мережа Real-ESRGAN використовує ту саму генераторну мережу, що й ESRGAN [1]. Для коефіцієнтів масштабування ×2 та ×1 вона спочатку застосовує операцію піксельної перетасовки, щоб зменшити просторовий розмір і перемістити інформацію в канальний вимір.

Вхід: Низькороздільне зображення (LR).

Первинний апсемплінг: Застосовується метод апсемплінгу, щоб збільшити розмір і відтворити загальні деталі низькороздільного зображення.

Глибокий прохід: Проходження через глибокий стек резидуальних блоків, які допомагають покращити якість зображення та вилучити важливі ознаки. Кожен резидуальний блок включає шари згортки та активаційні функції.

Апсемплінг: Застосовується метод апсемплінгу для збільшення роздільної здатності зображення.

Вихідний шар: Використовується згортка для перетворення вихідного сигналу на високороздільне зображення (SR).

RRDB (Residual Reconstruction Dense Block) блоки [1] - це тип блоків, які складаються з кількох згорткових шарів з функцією активації ReLU, які з'єднані за допомогою операції прямого додавання. Кожен згортковий шар у блоці RRDB має різну кількість ядер, що дозволяє мережі навчатися відтворювати різні рівні деталей зображення. Операція прямого додавання дозволяє мережі вчитися зберігати інформацію з низькоякісного зображення, а також генерувати нові деталі.

Reshape: Операція Reshape використовується для зміни форми тензора без зміни його значень. Вона може бути використана для перетворення тензора в інший розмір або форму, що відповідає потребам моделі.

ReLU (Rectified Linear Unit): ReLU є функцією активації, яка застосовується після конволюційного шару або повнозв'язаного шару в генераторі. Вона залишає значення незмінними, якщо вони позитивні, а негативні значення замінює нулем. Це допомагає усунути від'ємні значення та активувати тільки позитивні значення, сприяючи нелінійності та активації в мережі.

BN_norm (Batch Normalization): Batch Normalization є методом нормалізації, який використовується для нормалізації активаційних значень у генераторі. Вона допомагає стабілізувати і прискорити навчання шляхом центрування та нормалізації активаційних значень за допомогою середнього значення та стандартного відхилення пакету даних.

Tanh (гіперболічний тангенс): Tanh є функцією активації, яка використовується в вихідному шарі генератора для обмеження значень від -1 до 1. Вона перетворює значення в діапазон гіперболічного тангенсу, що дозволяє генерувати високоякісні зображення з покращеною якістю деталей.

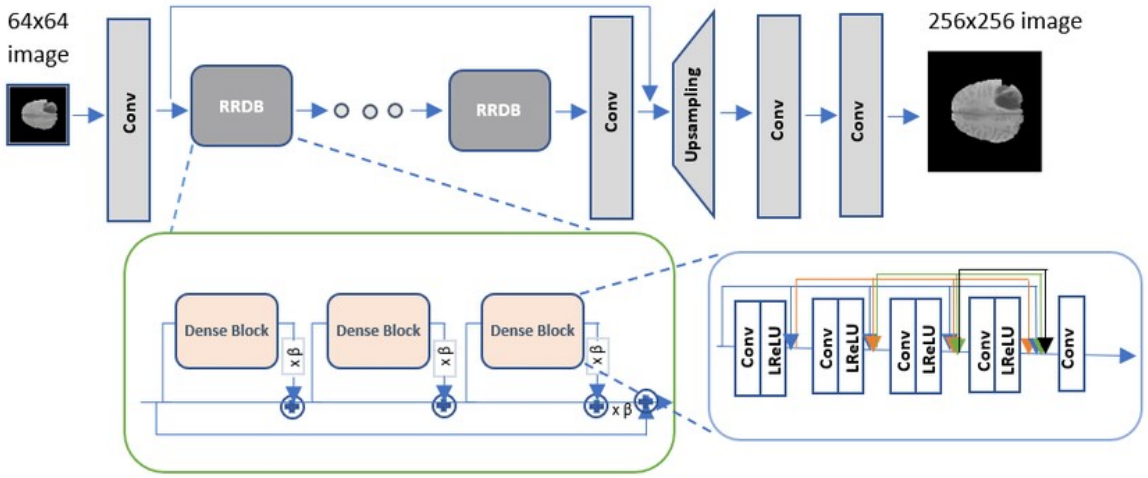

Рис. 7. Архітектура генератора Real-ESRGAN [5]

Дискримінатор:

Вхід: Високороздільне зображення (SR) або відтворене зображення.

Конволюційні шари: Зображення проходить через набір конволюційних шарів, які виконують вилучення ознак та зменшення розмірності.

Попередня обробка: Отримані ознаки піддаються попередній обробці, такі як підсумування або згортання.

Вихідний шар: Використовується один або кілька шарів звичайних нейронів для визначення ймовірності того, що зображення є реалістичним або синтезованим.

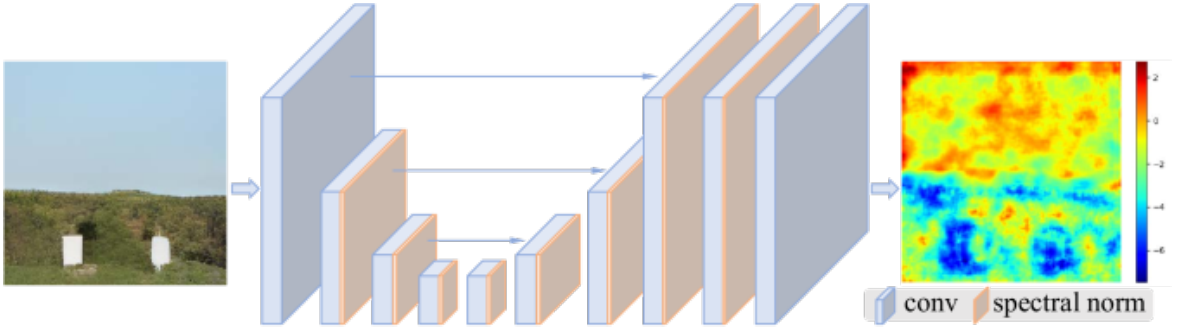

Рис 8. Архітектура U-Net дискримінатора [2]

Генератор та дискримінатор працюють у взаємодії під час навчання GAN. Генератор намагається створити високоякісне високороздільне зображення, яке дискримінатор не може відрізнити від реальних зображень. Дискримінатор в свою чергу намагається правильно класифікувати зображення, розрізняючи між синтезованими та реальними зображеннями.

Оскільки Real-ESRGAN націлений на вирішення набагато ширшого спектра деградації, ніж ESRGAN, оригінальний дизайн дискримінатора в ESRGAN більше не підходить. Зокрема, дискримінатор у Real-ESRGAN потребує більшої здатності розрізняти складні результати навчання. Замість того, щоб розрізняти глобальні стилі, він також повинен генерувати точний зворотній зв'язок по градієнту для локальних текстур. Тому Real-ESRGAN використовує покращений дискримінатор U-Net з пропускними з'єднаннями (Рис. 8). U-Net виводить значення реальності для кожного пікселя і може надавати генератору детальний відгук по кожному пікселю. [2,3]

3. Розробка Android додатку

3.1 Інструменти для розробки

Для розробки Android-додатку використовується мова програмування C++, Android NDK, Vulkan API та IDE Android Studio. Для додавання потрібного функціоналу використовується набір бібліотек [9-14]:

- libusb (робота з USB)
- libseek-thermal (робота з пристроями компанії Seek Thermal через USB-OTG),
- OpenCV (обробка та аналіз зображень, калібрування камер, комп'ютерний зір)
- Real-ESRGAN-NCNN-Vulkan (реалізація Real-ESRGAN на базі API Vulkan)
- GLM (OpenGL Mathematics, векторна та матрична математика) та ImGui (immediate-mode інтерфейс)
- Dear ImGui – бібліотека для створення швидких інтерфейсів

Для отримання максимальної швидкості розрахунків був обраний низькорівневий, відкритий стандартний графічний API Vulkan та реалізація мережі Real-ESRGAN для підвищення роздільної здатності зображень.

Цей набір технологій дозволяє досягти найкращої швидкодії та ефективності операцій, необхідних для генерування високоякісних зображень. Результатом даної роботи є створення додатку для Android, який може покращувати якість зображень інфрачервоних камер у реальному часі за допомогою генеративних змагальних super-resolution мереж.

3.2 Обладнання

Додаток орієнтований на підтримку найсучасніших SOC Qualcomm Snapdragon 8 Gen 1, Snapdragon 8 Gen 2 або рішення від компанії Samsung - SOC Exynos 2300, який оснащений найбільшою кількістю тензорних ядер на момент написання даної роботи.

Для тестування програмного забезпечення використовується набір мобільних пристроїв компанії Motorola на базі вищевказаних мобільних процесорів компанії Qualcomm, а саме телефони моделей Edge+ 2022 та Edge+ 2023.

За допомогою бібліотек libusb та libseek-thermal реалізована підтримка взаємодії з інфрачервоними камерами американської компанії Seek Thermal – відомого виробника рішень для отримання зображень в інфрачервоному спектрі.

Для отримання ІЧ-зображень використовується компактна теплова камера Seek Thermal Compact з наступними характеристиками [8]:

- Розмір термочутливої матриці (мікроболометра): 206×156 пікселів
- Загальна кількість пікселів: 32000
- Діапазон температур: від -40 до +330 ˚C
- Відстань між елементами матриці: 12мкм
- Кут огляду: 36°
- Теплова чутливість: 100 mK
- Довгохвильовий інфрачервоний діапазон: 7.2 – 13мкм
- Тип термочутливої матриці: Оксид-ванадієвий мікроболометр
- Частота кадрів: до 9 Гц
- Дальність виявлення: до 300 метрів

Рис 9. Вигляд камери Seek Thermal Compact [8]

Бібліотека OpenCV використовується для передобробки, постобробки, аналізу та роботи з зображеннями.

Бібліотека Real-ESRGAN-NCNN-Vulkan виступає ключовим елементом для генеративного підвищення роздільної здатності, покращення деталізації та реалістичності зображень, зменшення шуму, що надходять від інфрачервоних камер.

Dear ImGui – бібліотека для створення швидких інтерфейсів у негайному режимі (immediate-mode GUI), що дозволяє створювати інтерфейси, які не мають ввідного лагу, так як відтворення та ввід даних відбувається під час обробки кожного прорахованого кадру.

Усі ці технології та інструменти об'єднуються в компактному Android-додатку, розробка якого реалізована з урахуванням високих технічних стандартів та потужності сучасних мобільних пристроїв.

3.3. Етапи розробки Android додатку

Етапи розробки Android-додатку для покращення якості зображення інфрачервоних камер в реальному часі можна поділити на наступні пункти:

1. **Аналіз вимог:**

- Розроблення чітких технічних вимог та функціональних вимог до додатку.
- Визначення характеристик та можливостей цільових пристроїв, особливостей інфрачервоних камер.

2. **Проектування:**

- Створення структури додатку та вибір архітектури.
- Проектування інтерфейсу користувача та взаємодії з користувачем.
- Вибір технологій, бібліотек, та інструментів розробки.

3. **Реалізація функціоналу:**

- Розробка логіки обробки та підвищення роздільності інфрачервоних зображень.
- Інтеграція OpenCV, NCNN, Real-ESRGAN-NCNN-Vulkan для застосування генеративного підвищення роздільності.
- Взаємодія з бібліотеками libusb, libseek-thermal, OpenCV для роботи з інфрачервоними камерами.

4. **Розробка інтерфейсу:**

- Створення інтерфейсу користувача з використанням бібліотеки ImGui (Immediate Mode GUI)

- Реалізація функціоналу для налаштування параметрів та відображення зображень.

5. **Тестування:**
   - Виконання модульних, інтеграційних та системних тестів.
   - Тестування на різних моделях пристроїв на платформі Qualcomm Snapdragon 8 Gen 1 або Snapdragon 8 Gen 2.
6. **Оптимізація та вдосконалення:**
   - Покращення продуктивності та оптимізація використання ресурсів пристроїв.
   - Вдосконалення алгоритмів обробки та генеративного підвищення роздільності.
7. **Випробування в реальних умовах:**
   - Проведення випробувань на реальних інфрачервоних зображеннях у різних умовах освітлення та сценаріях використання.

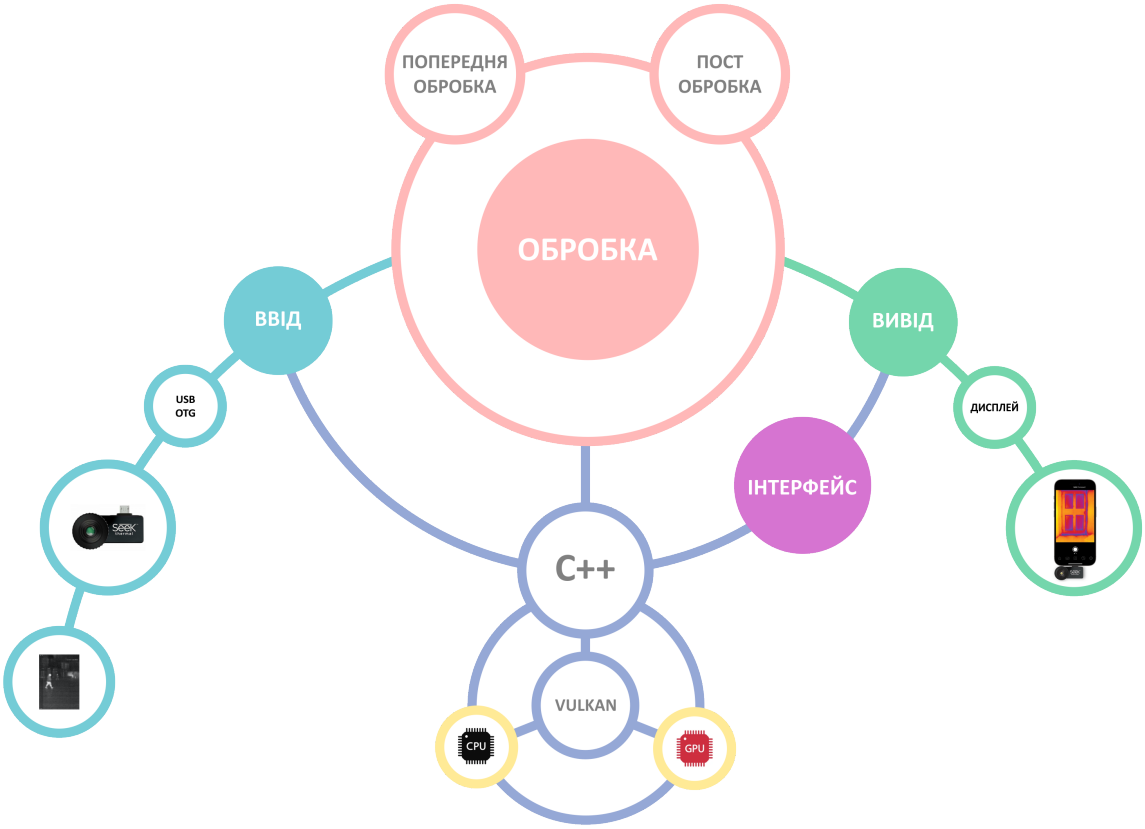

Рис 10. Загальна схема роботи додатку

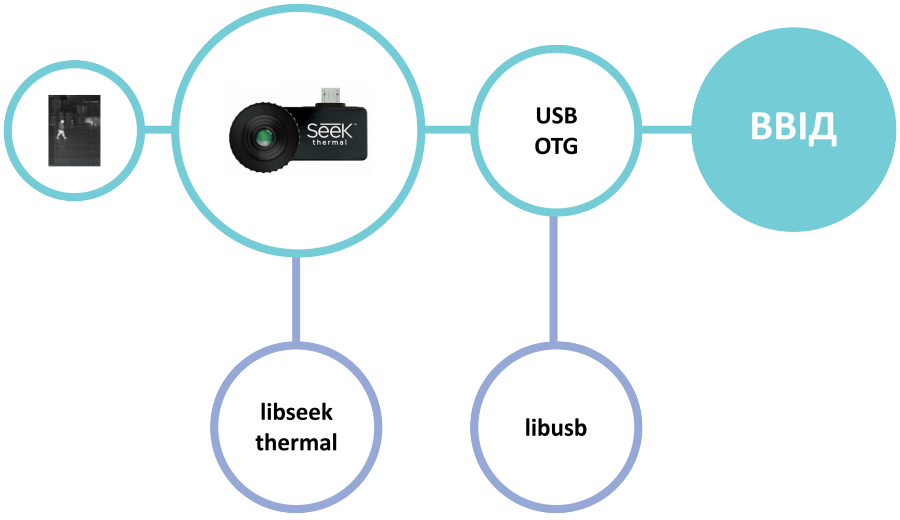

Рис 11. Схема вводу зображення

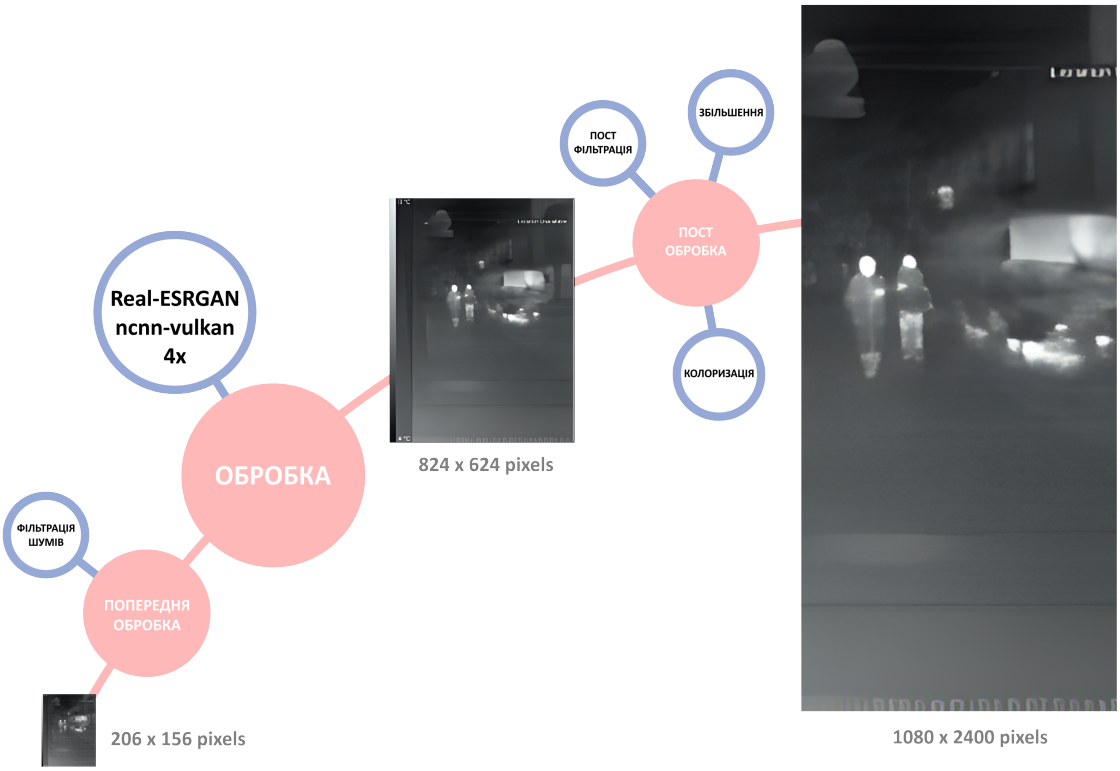

Рис 12. Схема обробки введеного зображення

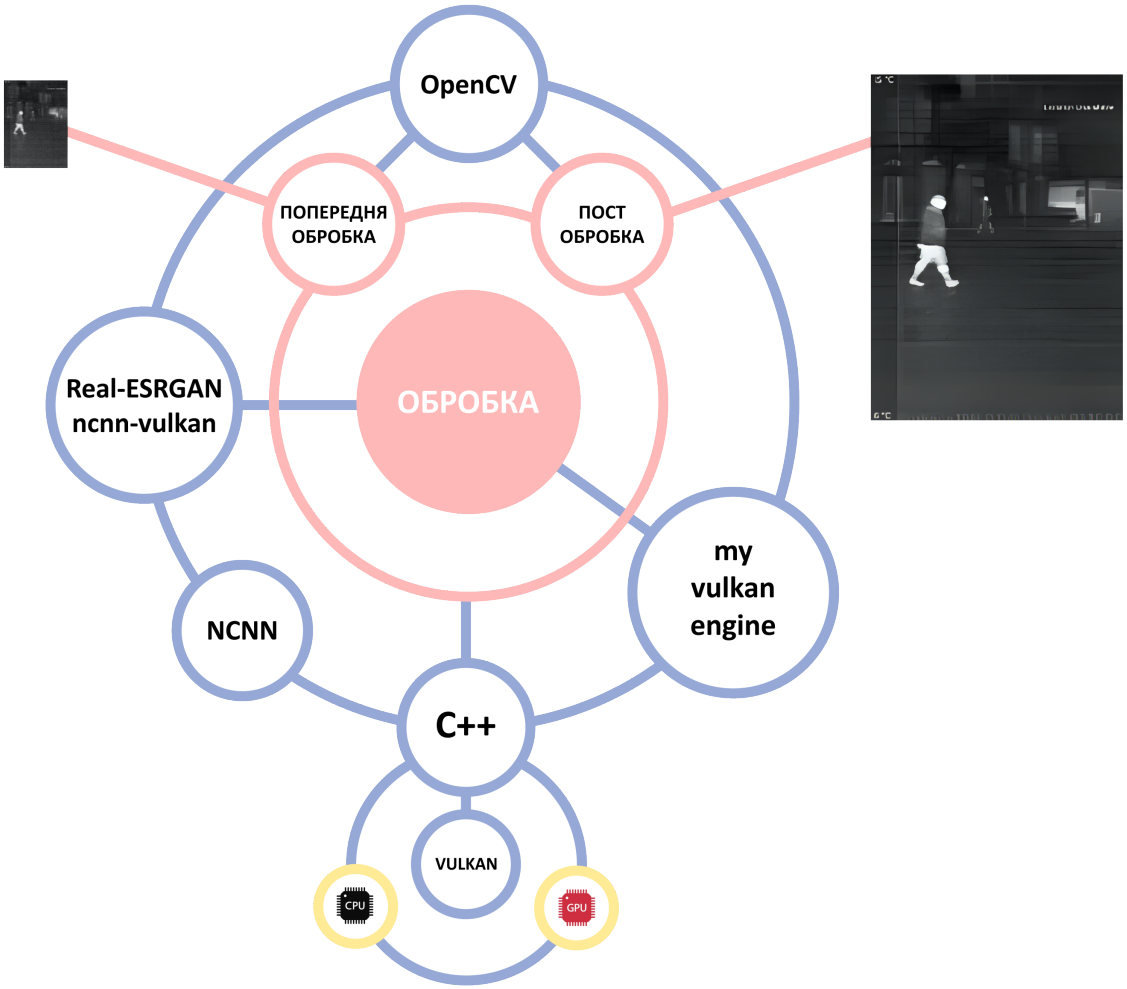

Рис 13. Схема процесу обробки зображення та використаних технологій

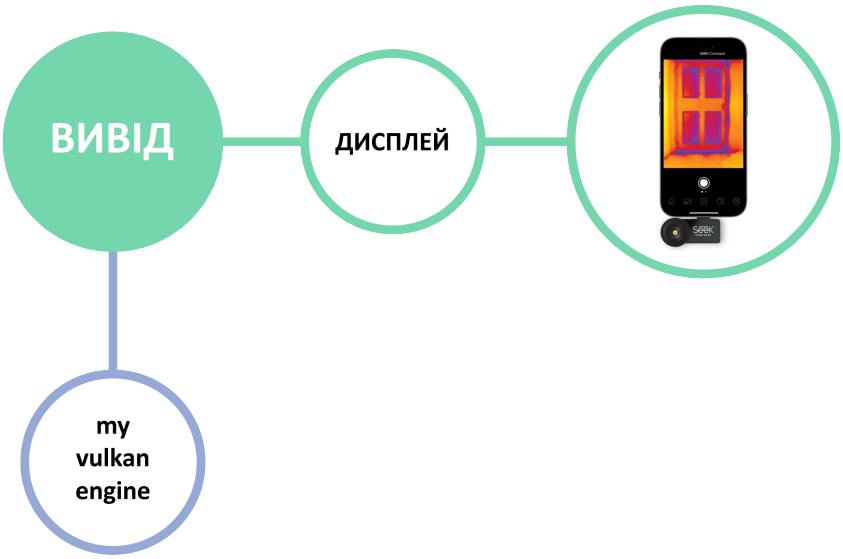

Рис 14. Схема виводу зображення

3.4 Приклади застосування розробленого додатку

Розроблений Android-додаток для покращення візуальної якості зображень інфрачервоних камер в реальному часі може знайти застосування в різних областях.

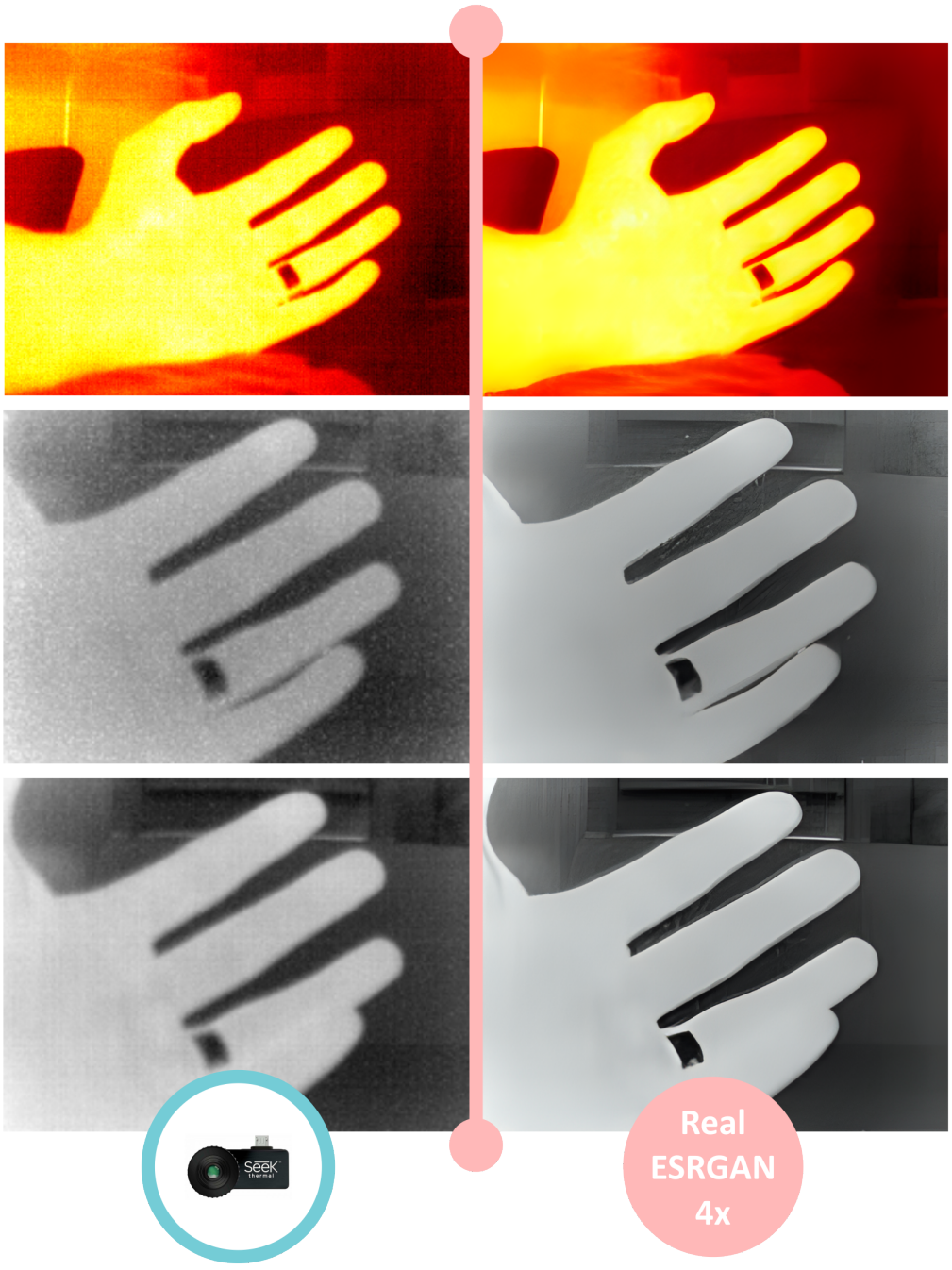

Рис 15. Покращення роздільної здатності та зменшення шуму: лівий стовбець – зображення з камери, правий - результат роботи Real-ESRGAN

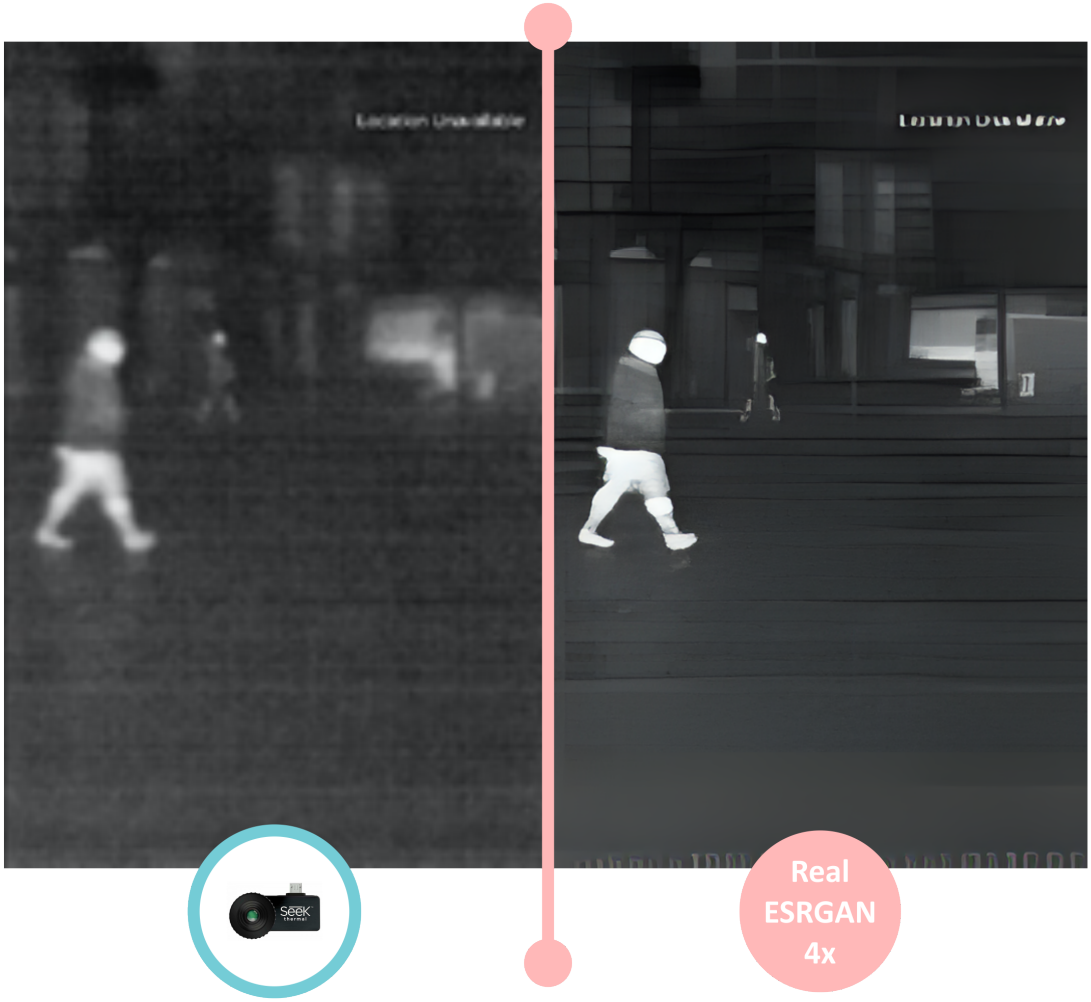

Рис 16. Приклад: Суттєве покращення чіткості зображення (лівий стовбець – зображення з камери (низька роздільна здатність), правий - результат роботи Real-ESRGAN (покращена роздільна здатність)

Рис 17. Приклад: Суттєве покращення чіткості зображення (лівий стовбець – зображення з камери (низька роздільна здатність), правий - результат роботи Real-ESRGAN (покращена роздільна здатність).

3.5. Перспективні напрямки застосування

Ось деякі приклади можливого застосування цієї технології:

- **Медицина:**

  Застосування додатку в медичних дослідженнях для поліпшення деталізації зображень, що дозволяє лікарям отримувати більш точну інформацію.

- **Безпека та нагляд:**

  Використання додатку для вдосконалення зображень, отриманих від інфрачервоних камер в системах безпеки та нагляду для забезпечення кращої чіткості зображень об'єктів чи сцен.

- **Будівництво та технічне обслуговування:**

  Використання додатку для підвищення роздільності інфрачервоних зображень у будівельній, сервісній сфері або під час технічного обслуговування, щоб ефективно виявляти аномалії або дефекти матеріалів, обладнання, живлення та ін.

- **Енергетика:**

  Застосування додатку в енергетичній галузі для поліпшення якості теплових карт інфрачервоних камер, що дозволяє більш ефективно виявляти аварійні або інші теплові аномалії та покращувати процеси діагностики.

- **Наука та дослідження:**

  Використання додатку в лабораторних умовах або наукових дослідженнях для аналізу інфрачервоних зображень з використанням генеративного підвищення роздільної здатності.

- **Розваги та туризм:**

  Впровадження додатку у галузі розваг та туризму для створення інфрачервоних зображень природних пейзажів або історичних об'єктів.

Ці приклади вказують на різноманітність застосування додатку, який може допомогти в різних галузях, де важлива висока якість інфрачервоних зображень.

Висновки

Основні результати проекту.

1. Real-ESRGAN є потужною мережею генерації високоякісних зображень з використанням глибокого навчання. Вона використовує генеративно-змагальну мережу (GAN), щоб покращити роздільну здатність та реалістичність зображень. Резидуальні блоки, що використовуються в моделі, дозволяють зберегти та передати важливу інформацію, що сприяє поліпшенню якості зображень.
2. Real-ESRGAN вирізняється здатністю використовувати інформацію з сусідніх кадрів для обробки відео в реальному часі. Це забезпечує більш точне відтворення динамічних змін на зображеннях. Крім того, модель має оптимізації, які забезпечують швидку обробку зображень та зменшення витрат обчислювальних ресурсів.
3. Real-ESRGAN є важливим кроком у вдосконаленні генерації високоякісних зображень. Вона знаходить застосування у багатьох галузях, включаючи обробку фотографій, відео та розширення роздільності зображень. Відкритий характер моделі дозволяє дослідникам та розробникам використовувати та адаптувати її для різних завдань обробки зображень.
4. Результатом роботи є прикладний додаток, що дозволяє покращити ефективність ІЧ-камер та покращує якість візуального спостереження в реальному часі.
5. Для розробки Android-додатку використовується мова програмування C++, Android NDK, Vulkan API та IDE Android Studio.

Перспективи розвитку та покращення проекту.

Загалом, Real-ESRGAN відкриває широкі можливості для покращення якості зображень та забезпечення реалістичних результатів. Ця мережа продовжує привертати увагу дослідників та відкривати нові горизонти в області генерації високоякісних зображень.

Застосування Real-ESRGAN включає такі області як покращення якості фото, графіки, кіно та відео з масиву даних згенерованого у XIX-XX століттях з моменту винаходу кіно- та фотоапаратів, що дає можливість покращити сприйняття як документальних так і художніх творів цього періоду часу, які були зроблені з використанням достатньо низькоякісної техніки та/або зазнали вплив часу та інші специфічні пошкодження.

Для сучасного (зазвичай набагато більш якісного фото та відеоконтенту) Real-ESRGAN допомагають у покращенні деталізації, кольоропередачі, зменшенні впливу артефактів компресії та ін. Real-ESRGAN мережі, які працюють в реальному часі, дозволяють зменшувати бітрейт необхідний для отримання високоякісного зображення.

Цікавим напрямком використання Real-ESRGAN є автоматизоване покрашення (збільшення якості, покращення деталізації та ін.) текстур для старих комп’ютерних ігор

Список використаної літератури

1. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., & Tang, X. (2018). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. arXiv preprint arXiv:1809.00219.

2. Wang, X., Xie, L., Dong, C., & Shan, Y. (2021). Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data. modelarXiv:2107.10833v2 [eess.IV], 17 Aug.

3. Wei, Z., Huang, Y., Chen, Y., Zheng, C., & Gao, J. (2021). A-ESRGAN: Training Real-World Blind Super-Resolution with Attention U-NET Discriminators. arXiv:2112.10046v1 [eess.IV], 19 Dec.

4. Shi, Y., Li, H., Zhang, S., Yang, Z., & Wang, X. (2022). Criteria Comparative Learning for Real-scene Image Super-Resolution. arXiv:2207.12767v1 [cs.CV], 26 Jul.

5. Rashid, S. I., Shakibapour, E., & Ebrahimi, M. (). Single MR Image Super-Resolution Using Generative Adversarial Network

6. Li, X., Wu, Y., Zhang, W., Wang, R., & Hou, F. (2019). Deep learning methods in real-time image super-resolution: a survey. Journal of Real-Time Image Processing.

7. Agarwal, A., Chhotaray, S., Roul, N. K., & Mehta, S. N. (2023). A Review Super Resolution Using Generative Adversarial Network-Applications and Challenges. Middle East Research Journal of Engineering and Technology.

8. Thermal. (2023). Thermal Seek Compact Camera, Sellsheet USA [Електронний ресурс] [PDF] (версія для веб-сайту) https://www.thermal.com/uploads/1/0/1/3/101388544/compact-sellsheet-usa_web.pdf

9. OpenCV [Електронний ресурс] Intel Corporation, , https://docs.opencv.org/4.8.0/

10. libusb [Електронний ресурс], https://libusb.org/

11. libseek-thermal [Електронний ресурс] https://github.com/seekthermal/libseek-thermal

12. Real-ESRGAN-NCNN-Vulkan [Електронний ресурс] https://github.com/xinntao/Real-ESRGAN-NCNN-Vulkan

13. GLM [Електронний ресурс] https://glm.g-truc.net/0.9.9/index.html

14. ImGui [Електронний ресурс] https://github.com/ocornut/imgui

15. Native Camera Vulkan [Електронний ресурс] https://github.com/ktzevani/native-camera-vulkan

Додатки

thermal_reader.hpp:

```
#ifndef NCV_THERMAL_READER_HPP
#define NCV_THERMAL_READER_HPP

#include <media/NdkImage.h>
#include <media/NdkImageReader.h>

#include <vector>

// Image Processing libraries
#include <opencv2/opencv.hpp> // Processing and VL
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

#include <libseek-thermal/src/seek.h> // Thermal Imaging
#include <libseek-thermal/src/SeekCam.h>
//#include <libseek-thermal/examples/args.h>

#include <processing/processing.hpp>

#include <signal.h>
#include <fcntl.h>

using namespace cv;
using namespace LibSeek;
using namespace processing;
```

```
namespace devices

{

    class thermal_reader

    {

    public:

        using image_ptr = std::unique_ptr<AImage,
decltype(&AImage_delete)>;

        using img_reader_ptr = std::unique_ptr<AImageReader,
decltype(&AImageReader_delete)>;


        thermal_reader(uint32_t a_width, uint32_t a_height, uint32_t
a_format, uint64_t a_usage, uint32_t a_max_images);


        ~thermal_reader();


        AHardwareBuffer* get_latest_buffer();

        ANativeWindow* get_window() const;


    private:


        uint32_t m_cur_index;

        ANativeWindow* m_window = nullptr;

        std::vector<AHardwareBuffer*> m_buffers;


        img_reader_ptr m_reader;

        std::vector<image_ptr> m_images;


        //Thermal cam


        float scale = 1.0;

        // 7fps seems to be about what you get from a seek thermal
compact

        // Note: fps doesn't influence how often frames are processed,
just the VideoWriter interpolation
```

```
        int fps = 7;

        // Colormap int corresponding to enum:
https://docs.opencv.org/master/d3/d50/group__imgproc__colormap.html#ga
9a805d8262bcbe273f16be9ea2055a65

        int colormap = 0;

        // Rotate default is landscape view to match camera
logo/markings

        int rotate = 270;

        // Mat containers for seek frames

        cv::Mat seekframe, outframe, frame_u16, frame_g8;


        // Setup Seek Thermal Compact camera

        bool auto_exposure_lock = false;

        bool isProcessing = false;

        bool isThermalCameraStarted = false;

        int smoothing = 1;

        std::string camtype = "seek";

        LibSeek::SeekThermal seek;

        LibSeek::SeekCam* cam;

        int initThermalCam();

        void getThermalCamFrame();


        // processing

        super_resolution my_processing;

    };

}


#endif // NCV_THERMAL_READER_HPP
```

thermal_reader.cpp:

```
#include <devices/thermal_reader.hpp>

#include <utilities/log.hpp>
```

```
using namespace ::std;

using namespace ::utilities;


namespace devices

{

    thermal_reader::thermal_reader(uint32_t a_width, uint32_t
a_height, uint32_t a_format, uint64_t a_usage, uint32_t a_max_images)

        : m_cur_index{a_max_images-1}, m_reader{nullptr,
AImageReader_delete}

    {

        if(a_max_images < 2)

            throw runtime_error("Max images must be at least 2.");


        for(uint32_t i = 0; i < a_max_images; ++i)

            m_images.push_back({nullptr, AImage_delete});


        m_buffers = vector<AHardwareBuffer*>(a_max_images, nullptr);


        auto pt = m_reader.release();

        AImageReader_newWithUsage(a_width, a_height, a_format,
a_usage, m_images.size()+2, &pt);

        m_reader.reset(pt);


        if(!m_reader)

            throw runtime_error("Failed to create image reader.");


        auto result = AImageReader_getWindow(m_reader.get(),
&m_window);


        if (result != AMEDIA_OK || m_window == nullptr)

            throw runtime_error("Failed to obtain window handle.");
```

```
        if constexpr(__ncv_logging_enabled)
            _log_android(log_level::info) << "Image reader created.";

        initThermalCam();
    }

    thermal_reader::~thermal_reader()
    {
        if constexpr(__ncv_logging_enabled)
            _log_android(log_level::info) << "Destroying image
reader...";
    }

    int thermal_reader::initThermalCam() {

        cam = &seek;

        if (!cam->open()) {
            if constexpr(__ncv_logging_enabled)
                _log_android(log_level::verbose) << "Failed to open "
<< camtype << " cam";
            return -1;
        }
        return 0;
    }

    void thermal_reader::getThermalCamFrame() {
        cam->retrieve(frame_u16);

        normalize(frame_u16, frame_u16, 0, 65535, NORM_MINMAX);

        // Convert seek CV_16UC1 to CV_8UC1
```

```
        frame_u16.convertTo(frame_g8, CV_8UC1, 1.0 / 256.0);


        if(isProcessing){
            // Processing frame using Real-ESRGAN
            my_processing.process_frame(frame_g8, outframe, scale,
colormap, rotate);
        }


    }


    AHardwareBuffer * thermal_reader::get_latest_buffer()
    {
        getThermalCamFrame();


        AImage *image = nullptr;
        auto result = AImageReader_acquireNextImage(m_reader.get(),
&image);
        if (result != AMEDIA_OK || !image) {
            return nullptr;
        }


        // Check format
        int width;
        result = AImage_getWidth(image, &width);
        int height;
        result = AImage_getHeight(image, &height);
        //int stride = image->stride;
        if (width != outframe.cols || height != outframe.rows ||
outframe.channels() != 4) {
            AImage_delete(image);
            return nullptr;
        }
```

```
        // Convert outframe to AHardwareBuffer
        cv::Mat rgba = cv::Mat(height, width, CV_8UC4);
        cv::cvtColor(outframe, rgba, cv::COLOR_BGR2RGBA);

        AHardwareBuffer* buffer = nullptr;
        result = AImage_getHardwareBuffer(image, &buffer);
        if (result != AMEDIA_OK || !buffer) {
            AImage_delete(image);
            return nullptr;
        }

        m_cur_index++;
        if (m_cur_index == m_images.size()) {
            m_cur_index = 0;
        }
        m_images[m_cur_index].reset(image);
        m_buffers[m_cur_index] = buffer;

        return m_buffers[m_cur_index];

    }

    ANativeWindow * thermal_reader::get_window() const
    {
        return m_window;
    }
}
```

processing.hpp:

```
#ifndef NCV_PROCESSING_HPP
```

```
#define NCV_PROCESSING_HPP

#include <memory>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
// ncnn
#include <cpu.h>
#include <gpu.h>
#include <platform.h>

#include <Real-ESRGAN-ncnn-vulkan/src/realesrgan.h>

using namespace cv;
using namespace ncnn;

namespace processing {
    class super_resolution {
    public:

        bool auto_exposure_lock = false;
        char parampath[256] = "assets/models/realesrgan-x4plus.param";
        char modelpath[256] = "assets/models/realesrgan-x4plus.bin";
        int scale = 4;

        cv::Mat realESRGAN(ncnn::Mat &img);

        // Normalize the image so that it uses the full color space
available for display.
        void normalize(cv::Mat &inframe, bool auto_exposure_lock);

        // Function to process a raw (corrected) seek frame
```

```
        void process_frame(cv::Mat &inframe, cv::Mat &outframe, float
scale, int colormap, int rotate);

        ncnn::Mat   cvMat_to_ncnnMat(const   cv::Mat&   cv_mat,   int
pixel_type = ncnn::Mat::PIXEL_BGR);

        cv::Mat   ncnnMat_to_cvMat(const   ncnn::Mat&   ncnn_mat,   int
pixel_type = ncnn::Mat::PIXEL_BGR);


    };
}


#endif //NCV_PROCESSING_HPP
```

processing.cpp:

```
#include <processing/processing.hpp>
#include <utilities/log.hpp>


using namespace ::utilities;


namespace processing {


        cv::Mat super_resolution::realESRGAN(ncnn::Mat &img) {
            // Init
            RealESRGAN realsrgan(0);
            realsrgan.load(parampath, modelpath);


            // Resize
            ncnn::Mat outimage;
            cv::Mat cv_outimage;
            outimage = ncnn::Mat(img.w * scale, img.h * scale,
(size_t)img.c, img.c);
            realsrgan.process(img, outimage);
```

```
            cv_outimage = ncnnMat_to_cvMat(outimage);
            return cv_outimage;
        }

        void super_resolution::normalize(cv::Mat &inframe, bool
auto_exposure_lock) {
            static double min = -1, max = -1;
            static float multiplier = -1;

            if (auto_exposure_lock) {
                if (min == -1) {
                    cv::minMaxLoc(inframe, &min, &max);
                    multiplier = 65535 / (max - min);
                }
                for (int y = 0; y < inframe.rows; y++) {
                    for (int x = 0; x < inframe.cols; x++) {
                        uint16_t val = inframe.at<uint16_t>(y, x);
                        if (val > max) {
                            val = 65535;
                        } else if (val < min) {
                            val = 0;
                        } else {
                            val = (val - min) * multiplier;
                        }
                        inframe.at<uint16_t>(y, x) = val;
                    }
                }
            } else {
                cv::normalize(inframe, inframe, 0, 65535,
NORM_MINMAX);
                min = -1;
            }
```

```
        }


        // Function to process a raw (corrected) seek frame
        void super_resolution::process_frame(cv::Mat &inframe, cv::Mat
&outframe, float scale, int colormap, int rotate) {
            cv::Mat frame_g8, upscaled_frame; // Transient Mat
containers for processing
            ncnn::Mat frame_ncnn; // Transient Mat containers for
processing


            super_resolution::normalize(inframe, auto_exposure_lock);


            // Convert seek CV_16UC1 to CV_8UC1
            inframe.convertTo(frame_g8, CV_8UC1, 1.0 / 256.0);


            // Rotate image
            if (rotate == 90) {
                transpose(frame_g8, frame_g8);
                flip(frame_g8, frame_g8, 1);
            } else if (rotate == 180) {
                flip(frame_g8, frame_g8, -1);
            } else if (rotate == 270) {
                transpose(frame_g8, frame_g8);
                flip(frame_g8, frame_g8, 0);
            }


            frame_ncnn = cvMat_to_ncnnMat(frame_g8);
            upscaled_frame = realESRGAN(frame_ncnn);


            // Resize image
            // Note this is expensive computationally, only do if
option set != 1
            if (scale != 1.0) {
```

```
                resize(upscaled_frame, upscaled_frame, Size(), scale,
scale, INTER_LINEAR);
            }


            // Apply colormap
            if (colormap != -1) {
                cv::applyColorMap(upscaled_frame, outframe, colormap);
            } else {
                cv::cvtColor(upscaled_frame, outframe,
cv::COLOR_GRAY2BGR);
            }


        }


    ncnn::Mat super_resolution::cvMat_to_ncnnMat(const cv::Mat&
cv_mat, int pixel_type = ncnn::Mat::PIXEL_BGR) {
        int w = cv_mat.cols;
        int h = cv_mat.rows;
        int c = cv_mat.channels();


        ncnn::Mat ncnn_mat;


        if (c == 1) {
            // 1 channel
            ncnn_mat = ncnn::Mat::from_pixels(cv_mat.data,
ncnn::Mat::PIXEL_GRAY, w, h);
        } else if (c == 3) {
            // 3 channels
            ncnn_mat = ncnn::Mat::from_pixels(cv_mat.data, pixel_type,
w, h);
        } else {
            // Unknown color space
            if constexpr(__ncv_logging_enabled)
```

```
                _log_android(log_level::verbose) << "Unsupported
channel number: " << c;

            return ncnn::Mat(); // empty

        }


        return ncnn_mat;

    }


    cv::Mat super_resolution::ncnnMat_to_cvMat(const ncnn::Mat&
ncnn_mat, int pixel_type = ncnn::Mat::PIXEL_BGR) {

        int w = ncnn_mat.w;

        int h = ncnn_mat.h;

        int c = ncnn_mat.c;


        cv::Mat cv_mat;


        if (c == 1) {

            // 1 channel

            cv_mat.create(h, w, CV_8UC1);

            ncnn_mat.to_pixels(cv_mat.data, pixel_type);

        } else if (c == 3) {

            // 3 channel

            cv_mat.create(h, w, CV_8UC3);

            ncnn_mat.to_pixels(cv_mat.data, pixel_type);

        } else {

            // Unknown color space

            std::cerr << "Unsupported channel number: " << c <<
std::endl;

        }


        return cv_mat;

    }

}
```