СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____

Кафедра ___інформаційних технологій та програмування_____

Пояснювальна записка

до кваліфікаційної магістерської роботи

_____магістр_____

(освітньо-кваліфікаційний рівень)

на тему: ___Інформаційна технологія обробки растрових зображень_____

_____методами глибокого навчання_____

Виконав: студент 2 курсу, групи ІСТ-24зм

126 «Інформаційні системи та технології___

(шифр і назва спеціальності)

_____Крохмаль А. В._____

(прізвище та ініціали)

Керівник __Захожай О.І._____

(прізвище та ініціали)

Рецензент __Меняйленко О.С._____

(прізвище та ініціали)

Київ – 2025 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____
Кафедра ___інформаційних технологій та програмування_____
Освітньо-кваліфікаційний рівень __магістр_____
Спеціальність 126 «Інформаційні системи та технології»_____
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., доц. Захожай О.І.
(підпис)
« ___ » _____2025 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу студенту
_____Крохмалю Андрію Віталійовичу_____
(прізвище, ім'я, по батькові)

1.Тема роботи: _Інформаційна технологія обробки растрових зображень методами глибокого навчання,_____
керівник роботи__д.т.н., проф. Захожай Олег Ігорович ,_____
(вчене звання, науковий ступінь, прізвище, ім'я, по батькові)
затверджені наказом університету від «28» 11 2025 року № 241/17.03
2. Строк подання студентом роботи: 20.12.2025
3. Вихідні дані до роботи: _Матеріали науково-дослідної практики, науково-методична література, дані інтернет-мережі_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
Вступ.
Розділ 1. Аналітичний огляд (методи програмної обробки растрових зображень, класифікація веб-сайтів та веб-застосунків, архітектура веб-застосунків, засоби розробки та розгортання веб-додатків).
Розділ 2. Розробка інформаційної технології обробки зображень (методи та алгоритми обробки зображень, методи глибокого навчання).
Розділ 3. Програмна реалізація в форматі веб-застосунку (проєктування архітектури застосунку, вибір програмних засобів, особливості програмної реалізації, приклад роботи застосунку, Розгортання веб-застосунку).
Висновки. Перелік джерел посилань. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 10.11.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Одержання завдання на виконання роботи	10.11.2025	виконано
2.	Укладання і погодження з керівником плану і етапів виконання роботи	14.11.2025	виконано
3.	Узагальнення даних літературних джерел	19.11.2025	виконано
4.	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху виконання завдання	26.11.2025	виконано
5.	Аналіз технічних засобів та існуючих систем	02.12.2025	виконано
6.	Реалізація практичної частини завдання	09.12.2025	виконано
7.	Укладання, оформлення та погодження пояснювальної записки з керівником	19.12.2025	виконано
8.	Надання пояснювальної записки на кафедру	20.12.2025	виконано
9.	Підготовка доповіді та презентації	23.12.2025	виконано

Студент _____ Крохмаль А.В.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Захожай О.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Робота містить: 48 сторінок основного тексту, 33 рисунки, 34 використаних джерела.

Метою випускної кваліфікаційної роботи є удосконалення процесу редагування растрових зображень через впровадження інтелектуальних методів обробки, зокрема методів глибокого навчання, підвищення ефективності людино-машинної взаємодії через використання єдиного веб-інтерфейсу, а також покращення його продуктивності та безпеки.

Розглянуто сучасні методи обробки зображень та архітектурні підходи до створення веб-застосунків, обґрунтовано вибір алгоритмів і технологій (U^2-Net, Vue.js, OpenCV.js) для реалізації SPA-додатку. Розроблено інтерактивний веб-застосунок із трьома ключовими функціями: видалення фону, стилізація зображень та автоматичне видалення людей з фото. Застосунок розгорнуто на Firebase Hosting для забезпечення стабільності та швидкодії. Визначено подальші напрями вдосконалення – розширення функціоналу та оптимізація продуктивності.

Розроблений веб-додаток може бути корисним як у навчальному, так і у прикладному контексті – для дизайнерів, освітян, користувачів соцмереж, блогерів та інших.

Ключові слова: РАСТРОВЕ ЗОБРАЖЕННЯ, ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ, DEEP LEARNING, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ПОПЕРЕДНЯ ОБРОБКА, ПОСТОБРОБКА, U-2-NET, ВЕБ-ЗАСТОСУНОК, SINGLE PAGE APPLICATION, ВЕБ-ПРОГРАМУВАННЯ, VUE.JS, ФРОНТЕНД.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку засобів візуалізації та мультимедійного контенту виникає потреба у створенні інструментів, здатних ефективно обробляти зображення із використанням сучасних методів штучного інтелекту. Незважаючи на стрімкий розвиток генеративного штучного інтелекту та великих мовних моделей (LLM), одним із поширених напрямів досліджень залишається застосування глибокого навчання до задач комп'ютерного зору, зокрема, для редагування та обробки растрових зображень в умовах обмежених обчислювальних ресурсів.

Видалення фону зображення з додаванням альфа-каналу є важливим інструментом у багатьох прикладних завданнях комп'ютерної графіки, веб-дизайну та електронної комерції. Результат такої обробки дозволяє отримати зображення з прозорим фоном, що може бути легко інтегроване у різні цифрові середовища без потреби в додатковій ручній обробці. Зокрема, така функціональність є корисною для створення товарних карток у маркетплейсах, генерації іконок або аватарів, розробки банерів та інших візуальних матеріалів, де потрібна гнучкість у компонуванні графічного контенту.

Створення художнього портрета за допомогою стилізації вхідного зображення надає можливість перетворити звичайну фотографію на зображення з унікальним художнім оформленням. Цей режим є особливо корисним у сфері цифрового мистецтва, маркетингу, персоналізації контенту та розробки сувенірної продукції. Результати роботи цього режиму можуть застосовуватись для створення профільних зображень, персоналізованих подарунків, або як елементи дизайну у соціальних мережах і креативних проєктах, що сприяє підвищенню естетичної цінності візуального контенту.

Автоматичне видалення людей із фотографій з адаптивним заповненням фону є надзвичайно актуальним інструментом у редагуванні зображень, коли виникає потреба усунути небажані об'єкти чи особи з кадру. Така

функціональність корисна для обробки туристичних фотографій, знімків нерухомості, архітектури або пейзажів, де необхідно залишити лише фон без сторонніх елементів. Адаптивне заповнення фону дозволяє уникнути візуальних артефактів, забезпечуючи реалістичність та цілісність зображення після редагування, що значно підвищує якість кінцевого результату без необхідності у складному ручному втручанні.

Також у сучасному світі спостерігається стрімке зростання популярності веб-сторінок, веб-сервісів та мережі Інтернет. Сьогодні практично кожен користується веб-ресурсами або веб-додатками для задоволення повсякденних потреб, і кількість таких користувачів щороку зростає. Цей розвиток відбувся надзвичайно динамічно: якщо проаналізувати еволюцію інтернет-технологій за останні 10–15 років, можна помітити значне розширення їхніх можливостей і функціональності [1, 2].

Окреме місце в архітектурі сучасних веб-додатків займають веб-сервіси. Це технології, що забезпечують обмін даними між програмами незалежно від операційної системи та мови програмування. Веб-сервіс має програмний інтерфейс (API), який приймає запити з мережі у стандартизованому форматі, виконує певні обчислення або операції та повертає результат у вигляді відповіді. Як формат обміну даними зазвичай використовується XML або його різновиди. Передача інформації здійснюється за протоколами TCP/IP, найчастіше через HTTP або HTTPS. Завдяки цьому забезпечується простота інтеграції веб-сервісів у різнорідні інформаційні системи, що сприяє гнучкому та масштабованому побудуванню архітектур програмного забезпечення.

Однією з основних архітектурних парадигм, що реалізує концепцію веб-сервісів, є сервіс-орієнтована архітектура (SOA – Service-Oriented Architecture). У межах цієї моделі веб-сервіси виступають як самодостатні функціональні одиниці, які можуть бути повторно використані, комбіновані та викликані з інших систем або додатків для виконання бізнес-логіки.

У сучасному інформаційному середовищі веб-сервіси відіграють ключову роль у таких сферах:

Інтеграція систем: Веб-сервіси дозволяють поєднувати функціональність різних програмних продуктів та платформ у єдину взаємодіючу систему. Це особливо важливо для підприємств, які використовують різні інформаційні системи (наприклад, CRM, ERP, бухгалтерію тощо).

Розподілені обчислення та хмарні технології: У середовищі хмарних обчислень веб-сервіси дозволяють клієнтам отримувати доступ до обчислювальних потужностей, сховищ, баз даних та інших ресурсів у вигляді сервісів за запитом (on-demand).

Мобільні додатки: Практично всі сучасні мобільні застосунки використовують веб-сервіси для обміну даними з віддаленими серверами. Це дозволяє підтримувати синхронізацію, обробку запитів користувача, збереження інформації у хмарному середовищі тощо.

Інтернет речей (IoT): Пристрої, що входять до складу IoT-інфраструктури, обмінюються даними через веб-сервіси, що забезпечує взаємодію між сенсорами, контролерами та аналітичними платформами.

Електронна комерція та фінансові технології: Веб-сервіси використовуються для обробки платіжних транзакцій, взаємодії з банківськими системами, оновлення даних про товари, логістику тощо [3].

Крім того, розвиток стандартів безпеки (таких як OAuth, HTTPS, API-ключі) дозволив забезпечити надійну та безпечну взаємодію з веб-сервісами навіть у чутливих галузях – охороні здоров’я, державному управлінні, правовій сфері.

Таким чином, веб-сервіси є основою для побудови гнучких, масштабованих, розподілених систем, які задовольняють потреби сучасного суспільства в ефективній цифровій взаємодії. Розробка веб-сервісів є однією з найактуальніших задач у сфері інформаційних технологій, що зумовлено глобальним поширенням цифрових рішень у різних галузях діяльності. Універсальність веб-сервісів полягає в їх здатності забезпечувати взаємодію між додатками, реалізованими на різних мовах програмування та платформах,

що дозволяє ефективно організовувати обмін даними в умовах гетерогенного інформаційного середовища. Отже, розробка веб-сервісів залишається актуальним напрямом, що визначає технологічний прогрес сучасного цифрового суспільства.

Основною задачею кваліфікаційної роботи є розробка інформаційної технології обробки зображень засобами глибокого навчання, що зробить можливим створення інтерактивного веб-застосунку, що працює у форматі SPA, має монолітну архітектуру та виконує всі обчислення на клієнтській стороні. Передбачається, що розроблений застосунок забезпечить користувачам можливість здійснювати наступні операції:

- *автоматичне видалення фону зображення* з одночасним додаванням альфа-каналу, що забезпечує прозорість на місці видаленого фону;
- *створення художнього портрета* на основі вхідного зображення шляхом стилізації з використанням моделей глибокого навчання;
- *автоматичне видалення людей з фотографій* із можливістю адаптивного заповнення фону у місцях видалення.

Інтерфейс застосунку має бути інтерактивним, інтуїтивно зрозумілим та адаптивним до різних пристроїв Веб-застосунок має бути реалізовано виключно у вигляді клієнтської частини (frontend), без використання серверної обробки. Це передбачає використання технології WebAssembly для локального виконання моделей глибокого навчання у браузері користувача. Такий підхід дозволяє працювати із застосунком навіть в умовах обмеженого або відсутнього інтернет-з’єднання, забезпечуючи офлайн-доступ до його основного функціоналу. Отже, обрана архітектура та технологічний стек сприяють створенню ефективного, безпечного та зручного у використанні веб-застосунку для обробки зображень.

Об'єктом дослідження є методи, моделі та інформаційна технологія обробки растрових зображень, зокрема такі, що можуть бути програмно реалізовані на клієнтській стороні веб-застосунку.

Предметом дослідження є інформаційна технологія інтелектуальної обробки зображень з використанням глибоких штучних нейронних мереж (ШНМ), методи та засоби проектування фронтенд-орієнтованих односторінкових web-додатків.

Метою дослідження є удосконалення процесу редагування растрових зображень через впровадження інтелектуальних методів обробки, що використовують технології штучного інтелекту, зокрема методи глибокого навчання ШНМ. Використання запропонованої розробленої інформаційної технології має підвищити ефективність людино-машинної взаємодії через використання єдиного веб-інтерфейсу та покращити продуктивність, безпеку та конфіденційність в процесі обробки.

Таким чином, в рамках даної кваліфікаційної роботи магістра, ***основними задачами***, що були поставлені задля досягнення мети роботи, є:

- аналіз існуючих підходів до програмної обробки растрових зображень;
- аналіз сучасних типів та архітектур веб застосунків, технологій та інструментів їх проєктування, розгортання та обслуговування;
- розробка методів обробки зображень з використанням глибоких нейронних мереж для виконання поставлених задач;
- вибір та адаптація моделей глибокого навчання, придатних для виконання зазначених функцій на стороні клієнта;
- проєктування архітектури SPA-застосунку для забезпечення інтерактивної взаємодії користувача з інтерфейсом;
- реалізація веб-додатку з використанням сучасних фреймворків та мови програмування JavaScript; реалізація алгоритмів обробки зображень безпосередньо на стороні браузера користувача;

- проведення тестування якості обробки зображень та якісна оцінка ефективності реалізованого рішення.

Методи дослідження. Для вирішення поставлених задач застосовані:

- *загальнонаукові методи:* аналітико-синтетичний, графічний, теоретичного пошуку; концептуально-порівняльного аналізу, визначення теоретичних і прикладних аспектів дослідження, визначення структури і змісту підготовки;
- *емпіричні методи:* дослідження предметної області, дослідження об'єкту, що вивчається у штучно створених для нього умовах, порівняння об'єкту, що досліджується з аналогом;
- *конкретнонаукові:* архітектурне та алгоритмічне моделювання, динамічне завантаження компонентів (lazy loading), реактивні властивості та двонаправлений зв'язкок, асинхронне програмування, методи нормалізації та зворотньої нормалізації даних, глибоке навчання нейронних мереж, методи цифрової обробки зображень і комп'ютерного зору (гаусівська фільтрація, маскування, зважене додавання, інпейнтинг).

Наукова новизна дослідження полягає у тому, що було вперше реалізовано інформаційну технологію обробки зображень засобами Deep Learning через веб-застосункок, який реалізує виконання всіх етапів обробки зображень на стороні клієнта та має функціонал видалення фону, створення художніх портретів та видалення людей на зображенні.

Теоретична значущість дослідження полягає в розробці інформаційної технології обробки растрових зображень на основі методів глибокого навчання, її реалізація у форматі веб-застосунку, включаючи принципи організації архітектури веб-сервісів, обробки графічної інформації, та використання глибоких ШНМ у браузерному середовищі. Також увага приділялася реалізації архітектурної моделі Single Page Application, принципів побудови користувацького інтерфейсу, а також механізмам інтеграції

алгоритмів комп’ютерного зору та моделей глибокого навчання через WebAssembly. У ході дослідження було розглянуто етапи розробки інформаційної технології, особливості обробки зображень на стороні клієнта, а також вплив використаних технологічних рішень на продуктивність і безпеку системи.

Практична значущість дослідження полягає у створенні сучасного веб-застосунку, який реалізує розроблену інформаційну технологію та надає можливість виконання складних завдань обробки зображень – зокрема, автоматичного видалення фону, стилізації зображень у вигляді художнього портрету та видалення людей із кадру. Застосунок буде здатний виконувати зазначені задачі без потреби у серверній обробці, що забезпечить зручність використання, конфіденційність даних та високу доступність технології для кінцевого користувача. Розроблений веб-додаток може бути корисним як у навчальному, так і у прикладному контексті – для дизайнерів, освітян, користувачів соцмереж, блогерів та інших.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Методи програмної обробки растрових зображень

У сучасному цифровому середовищі прикладні інструменти та програмні засоби для роботи з графікою, зокрема растровою, розвиваються досить стрімко, що відкриває нові можливості для творчої діяльності у сфері обробки фотозображень та графічного дизайну. Існує широке коло методів та технологій обробки фотографічних зображень, які охоплюють як базове корегування, так і складні художні трансформації. Ці методи реалізуються у межах сучасних графічних програмних середовищ, що дозволяє не лише покращувати якість зображень, а й створювати нові візуальні форми, поєднуючи фотографію з елементами графіки, тексту та дизайну [4].

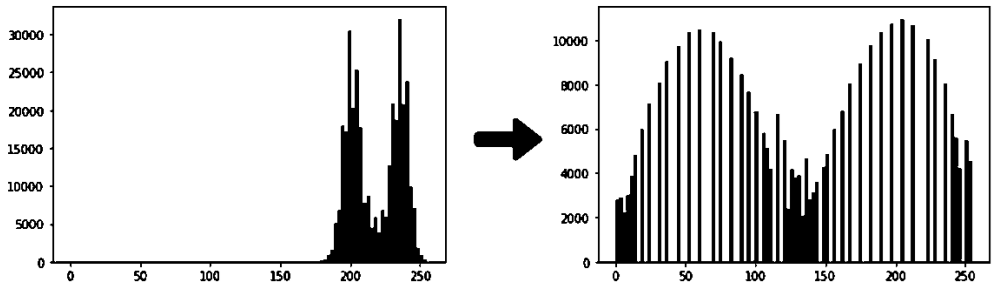
Умовно методи обробки растрової графіки, що використовують у найбільш поширених графічних редакторах, таких як Gimp та Adobe Photoshop, можна розділити на чотири групи. Першу групу методів становлять корекційні операції, спрямовані на покращення художньо-естетичних якостей зображення: баланс білого, корекція тону, насиченості, яскравості, геометричне вирівнювання та видалення дефектів. Друга група охоплює глибші зміни, що дозволяють реалізувати авторський задум: ретушування, реставрація, розфарбовування чорно-білих фото, тонування та кольорокорекція. Третя група включає стилізацію фотографій з використанням цифрових ефектів, подвійної експозиції, імітації живопису тощо. Четверта група передбачає створення композицій, де фотографії поєднуються з текстовими та графічними елементами, що є характерним для рекламної та презентаційної продукції [4].

Корегування експозиції та контрастності є найпростішими і найпоширенішими перетвореннями інтенсивності в растрових редакторах. Формально ці операції часто реалізуються як афінні перетворення інтенсивності пікселів:

$$I' = \alpha I + \beta,$$

де множник α змінює контраст, а зсув β – яскравість; складніші методи використовують локальні адаптації або авто-балансування (наприклад, автоматичне масштабування діапазону інтенсивностей за статистикою зображення). Практична реалізація в редакторах поєднує прості оператори з інструментами візуальної корекції та локального тонального корегування [5].

Метод вирівнювання гістограми (histogram equalization) та його адаптивні варіанти (наприклад, CLAHE – Contrast Limited Adaptive Histogram Equalization) призначені для перерозподілу інтенсивностей з метою покращення локального контрасту без значного підсилення шуму (рисунок 1.1.1). Глобальна еквалізація перетворює кумулятивну функцію розподілу інтенсивностей в більш «рівномірну», тоді як CLAHE розбиває зображення на блоки та застосовує обмеження підсилення для запобігання локальному засвіченню артефактів. У застосунках машинного зору та редакторах це є стандартним засобом покращення видимості деталей в умовах нерівномірної освітленості [6, 7].



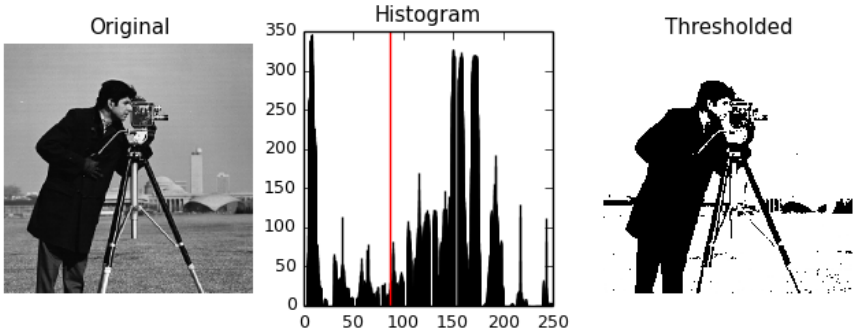
а) б)

Рисунок 1.1.1 – Результат виконання операції вирівнювання гістограми: а) початкова гістограма, б) вирівняна гістограма

Порогова обробка (thresholding) – простий, математично детермінований інструмент сегментації за інтенсивністю. Простий поріг розділяє пікселі за критерієм:

$$I(x, y) \gtrless T$$

Адаптивні пороги та метод Отцу автоматично підбирають поріг за статистичною мірою міжкласової дисперсії (рисунок 1.1.2). Порогова обробка часто використовується як попередній етап для бінаризації документів, виділення контурів або як маска для подальшої обробки. Основними недоліками є чутливість до шуму і неоднорідності освітлення, що пояснює необхідність комбінувати порогування з фільтрацією або локальною компенсацією [8-10].



а) б) в)

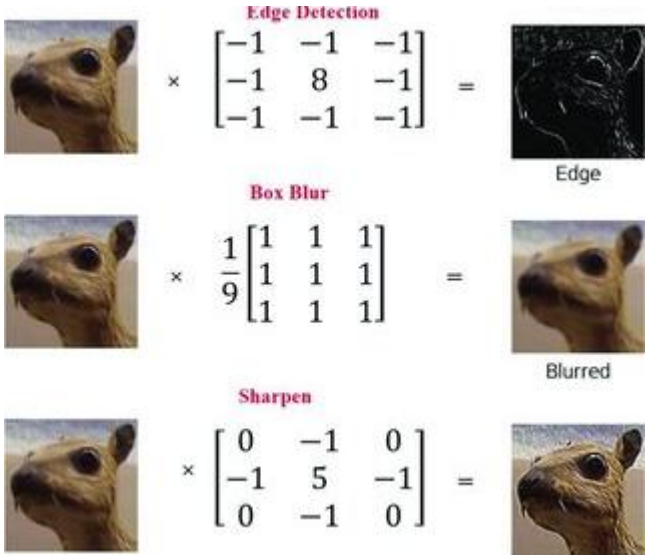
Рисунок 1.1.2 – Визначення порогового значення за методом Отцу: а) вхідне зображення (відтінки сірого), б) гістограма та порогове значення, в) бінарне зображення

Згорткові фільтри (convolutional filters) становлять математичну основу багатьох традиційних трансформацій: згладжування (усереднюючий, гаусівський фільтр), підсилення різниці (sharpening), виділення країв (Sobel, Prewitt, Laplacian) та більш складні лінійні та нелінійні ядра. Формально операція згортки визначається як:

$$G(x,y) = \sum_{i,j} K(i,j)\, I(x-i, y-j)$$

Вона дає змогу представити фільтри як лінійні оператори над сигналом. На практиці її реалізацію оптимізують через розкладання ядра (сепарабельність). Сучасні редактори надають набір таких фільтрів для шумоподавлення, корекції різкості та художніх ефектів (рисунок 1.1.3), а

також комбінують їх з масками та адаптивними вагами для локальної обробки [4, 11].



а) б) в)

Рисунок 1.1.3 – Застосування згорткового фільтру для виділення країв, послаблення та посилення високочастотних складових: а) вхідне зображення, б) ядро згортки, в) вихідне зображення

Суперроздільність (single-image super-resolution, SISR) належить до класу «інтелектуальних» методів і використовує статистичні властивості природних зображень та навчання на великих датасетах для відновлення високочастотних деталей при збільшенні роздільності (рисунок 1.1.4). Історично перші нейромережеві методи (наприклад, SRCNN) показали перевагу перед традиційними методами інтерполяції, більш пізні підходи, зокрема SRGAN і ESRGAN, застосували генеративні змагальні мережі (GAN) та архітектури з «щільними» блоками для відновлення фотореалістичних текстур, покращивши суб’єктивну якість зображень, але інколи породжуючи «галюцинації» деталей. Сучасні інструменти, включно з реалізаціями в графічних редакторах та спеціалізованому програмному забезпеченні, комбінують моделі, натреновані на реалістичних даних, і механізми корекції артефактів [12-14].

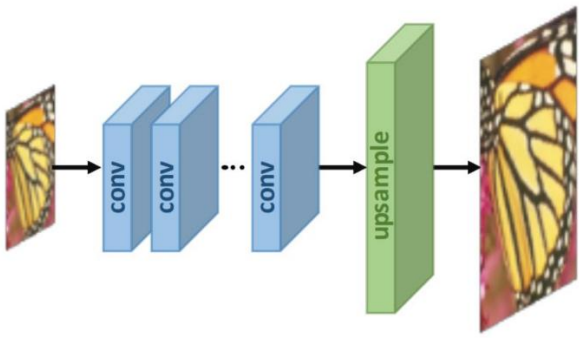


Рисунок 1.1.4 – Структурна схема методу super-resolution з використанням глибокої згорткової ШНМ

Нейронний перенос стилю (Neural Style Transfer, NST) – метод, що формалізує художній «стиль» та «зміст» зображення через представлення в прихованих шарах попередньо навчених глибоких згорткових мереж (наприклад, VGG). Метод Леона Гетіса і співавторів розділяє ці представлення та оптимізує результуюче зображення за сумарною втратою, що поєднує втрату змісту та матрицю Гремма для стилю, що дозволяє перенести текстури та мазки живопису на фотографію (рисунок 1.1.5). Хоча оригінальний підхід є обчислювально затратний, в подальшому було розроблено більш ефективні методи перетворення стилю через параметризовані перетворювальні мережі для інтерактивних застосунків та редакторів [15].

а) б) в)

Рисунок 1.1.5 – Нейронний перенос стилю: а) вхідне зображення, б) зразок стилю, в) вихідне зображення

Видалення об’єктів і контекстне заповнення – технологія, що поєднує алгоритмічні методи пошуку відповідності і генеративні нейронні мережі. Класичний метод PatchMatch [16] виконує пошук відповідності між патчами зображення для заповнення «дірок» адаптованим вмістом сусідніх ділянок і лежить в основі багатьох контекстно-залежних інструментів ретуші у графічних редакторах. Результат такої операції наведено на рисунок 1.1.6. Сучасні підходи глибинного навчання трактують задачу як генеративну: мережа навчається синтезувати заповнення, враховуючи глобальну семантику сцени й контекст, що дає кращі результати у випадку складних або великих відсутніх областей [17]. У практичних застосуваннях часто застосовують гібридні рішення: швидкий PatchMatch/Content-Aware Fill для інтерактивних правок і генеративні глибокі ШНМ для складних випадків [18-20].

а) б) в)

Рисунок 1.1.6 – Генеративне видалення об’єкта: а) вхідне зображення, б) бінарне зображення-маска, в) вихідне зображення

Отже, традиційні методи (лінійні перетворення, гістограми, згорткові фільтри, порогова обробка) забезпечують детерміністичний, інтерпретований та обчислювально ефективний набір засобів базової корекції та підготовки зображень; інтелектуальні методи (суперроздільність, NST, inpainting) розширюють можливості за рахунок використання технологій штучного інтелекту, зокрема глибоких ШНМ, і здатні відновлювати або синтезувати

складні структури, однак вимагають оцінки «галюцинацій» контенту та узгодження етичних аспектів. Комбінування обох груп методів у робочих процесах сучасних редакторів дає найбільш гнучкі та якісні результати.

1.2 Класифікація веб-сайтів та веб-застосунків

Різноманітність веб-застосунків визначається широким спектром їх функціональних можливостей, структурною організацією та технологічною реалізацією. Незалежно від конкретного призначення, будь-який веб-застосунок має у своїй структурі головну (стартову) сторінку, яка виконує функцію центрального вузла навігації і забезпечує доступ до основних тематичних розділів системи. Ці розділи, у свою чергу, можуть містити власні вхідні сторінки, пов’язані як з головною сторінкою, так і з іншими інформаційними підсистемами.

Залежно від застосованих технологій, веб-застосунки класифікуються на

- *статичні* (усі сторінки реалізовані як незмінні HTML-документи),
- *динамічні* (контент генерується на стороні сервера або клієнта),
- *засновані на мультимедійних технологіях*,
- *змішаного типу*.

За приналежністю до суб’єктів цифрової взаємодії, розрізняють

- *персональні* веб-застосунки (створені індивідуальними користувачами для представлення власної діяльності),
- *корпоративні* (комерційні),
- *некомерційні* (державні, освітні, соціальні тощо).

Корпоративні веб-застосунки поділяються за функціональним призначенням на кілька категорій:

- *застосунки-візитки*, які виконують представницьку функцію, демонструючи базову інформацію про організацію,
- *промо-застосунки*, орієнтовані на просування товарів і послуг, реалізують інтерактивні рекламні механізми та інформують про комерційні акції,

- *застосунки електронної комерції*, що підтримують повний цикл торгівлі: від демонстрації асортименту до обробки замовлень і здійснення оплат.

Некомерційні веб-застосунки мають на меті інформування населення, підвищення прозорості організацій, забезпечення громадського доступу до відкритих ресурсів та зворотного зв’язку. Вони часто інтегрують сервіси для комунікації, подання документів, доступу до баз даних тощо.

Функціональна класифікація веб-застосунків охоплює:

- *інформаційні системи*, до яких належать новинні портали, цифрові бібліотеки, енциклопедії та каталоги,
- *комунікаційні застосунки*, що забезпечують обмін повідомленнями (форуми, соціальні мережі, чати, блоги),
- *торговельні платформи*, які реалізують механізми купівлі-продажу, електронних платежів, біржової торгівлі,
- *онлайн-сервіси*, призначені для надання функціональних можливостей у віддаленому режимі, зокрема сервіси електронної пошти, пошукові системи, віртуальні офіси, системи автоматизованої розробки контенту тощо.

З огляду на обсяг вмісту, веб-застосунки поділяються на:

- *малі*, що охоплюють вузькотематичний контент з обмеженою кількістю сторінок (наприклад, персональні сторінки або мікросайти),
- *тематичні*, які зосереджені на поглибленому висвітленні окремої предметної області,
- *багатофункціональні портали*, що інтегрують різноманітні сервіси і бази знань для обслуговування широкої аудиторії.

Отже, сучасні веб-застосунки не лише є джерелом цифрової інформації, але й виступають інтерактивними середовищами для взаємодії, торгівлі,

комунікації та надання послуг, що зумовлює їх фундаментальну роль у соціально-економічному житті та інформатизації суспільства.

1.3 Архітектура веб-застосунків

Архітектура веб-застосунків – це структурна основа, яка відповідає за їх проєктування і дизайн. Вона охоплює набір принципів і найкращих практик, що визначають, як різні частини застосунку працюють разом, щоб забезпечити виконання його функцій. Основна мета вибору архітектури полягає в створенні єдиної та масштабованої системи, яка відповідає функціональним потребам застосунку, зважаючи також на такі фактори, як-от продуктивність, безпека та підтримка [21].

Архітектура сучасних веб-застосунків ґрунтується на багаторівневих моделях, що забезпечують гнучкість, масштабованість, розподіленість та безпеку систем. Найпоширенішою є трирівнева архітектура (three-tier architecture), яка включає клієнтський рівень (frontend), серверний рівень (backend) і рівень зберігання даних (database layer) (рисунок 1.3.1). Така структура дозволяє розмежувати функціональні обов'язки різних компонентів, спрощує обслуговування та розвиток застосунку [22].

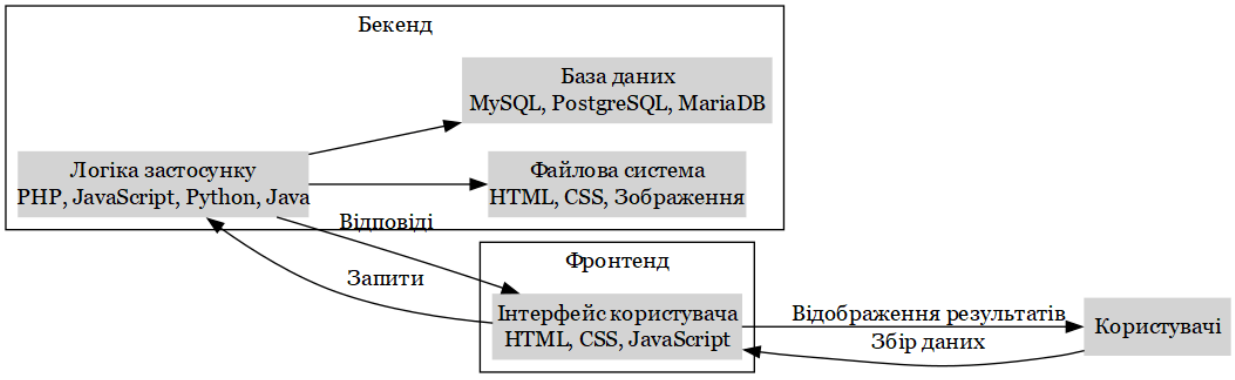


Рисунок 1.3.1 – Загальна архітектура web-застосунку

Клієнтський рівень (frontend) відповідає за відображення інтерфейсу користувача і взаємодію з ним. У сучасних веб-застосунках фронтенд реалізується за допомогою HTML, CSS та JavaScript, з використанням

популярних фреймворків, таких як React, Angular або Vue.js. Важливою особливістю є застосування асинхронного обміну даними через API (зокрема RESTful або GraphQL), що дозволяє динамічно оновлювати інтерфейс без перезавантаження сторінки (технологія AJAX).

Серверний рівень (backend) виконує обробку запитів, реалізацію бізнес-логіки та взаємодію з базами даних і зовнішніми сервісами. Цей рівень зазвичай реалізується мовами програмування, такими як Python (Django, Flask), JavaScript (Node.js), Java (Spring), PHP (Laravel), Ruby (Rails) тощо. Серверний рівень також включає реалізацію механізмів автентифікації, авторизації, обробки помилок, логування та контролю доступу.

Рівень зберігання даних охоплює бази даних, у яких зберігається структурована або неструктурована інформація. Залежно від задач, використовуються реляційні бази даних (MySQL, PostgreSQL, MS SQL Server) або нереляційні (NoSQL) системи, такі як MongoDB, Redis, Cassandra. Важливою частиною архітектури також є використання кешуючих механізмів (наприклад, Memcached, Redis) для оптимізації доступу до часто використовуваних даних.

У складніших веб-застосунках застосовується *мікросервісна архітектура*, що передбачає поділ функціоналу на незалежні сервіси, кожен з яких реалізує окрему бізнес-функцію та взаємодіє з іншими через API. Такий підхід сприяє горизонтальному масштабуванню, підвищенню надійності, спрощенню тестування та гнучкості розробки.

Мікросервісна та монолітна архітектури є двома основними підходами до розробки архітектури веб-сервісів. *Монолітна архітектура* передбачає створення програмного продукту як єдиної цілісної системи, в якій усі компоненти взаємопов’язані та функціонують у спільному середовищі (рисунок 1.3.2). Вона відзначається високою продуктивністю завдяки локальним викликам між модулями та простішою розробкою, оскільки весь код міститься в єдиному сховищі. Проте монолітні системи можуть бути

складними у масштабуванні, а внесення змін або оновлення потребує повторного розгортання всього застосунку.

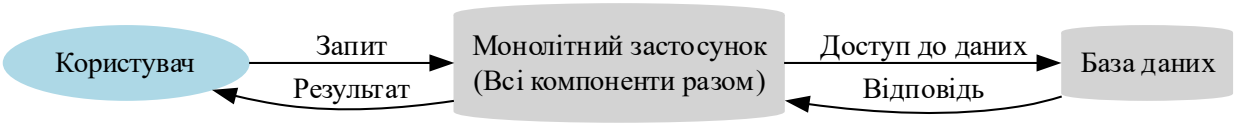


Рисунок 1.3.2 – Монолітна архітектура

На противагу цьому, мікросервісна архітектура ґрунтується на розподіленій структурі, де кожен сервіс виконує окрему функцію та взаємодіє з іншими через API (рисунок 1.3.3). Такий підхід забезпечує високу гнучкість у масштабуванні, даючи можливість незалежного розгортання й оновлення окремих модулів. Окрім того, мікросервісна архітектура дозволяє застосовувати різні технології для окремих сервісів, що підвищує адаптивність та ефективність розробки. Водночас керування численними мікросервісами потребує додаткових інструментів оркестрації, таких як Kubernetes, а мережеві запити можуть впливати на продуктивність системи [22, 23].

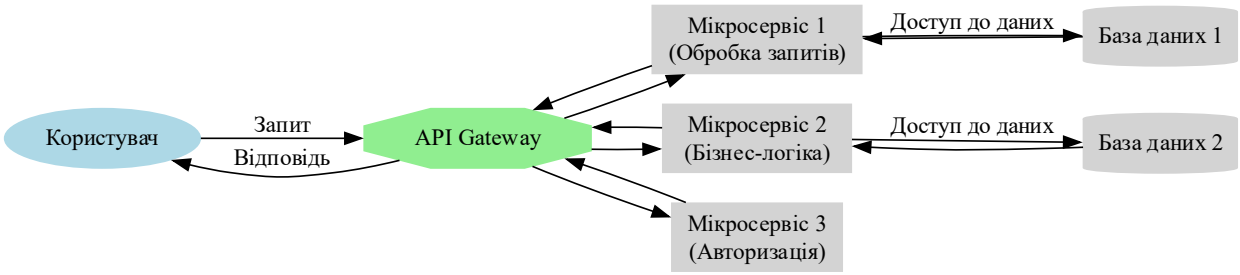


Рисунок 1.3.3 – Мікросервісна архітектура

Щодо основних підходів до організації веб-інтерфейсу, слід виділити такі архітектури як односторінкові (SPA), багатосторінкові (MPA) та прогресивні веб-застосунки (PWA).

Односторінкові застосунки (SPA) функціонують на основі динамічного завантаження контенту без необхідності перезавантаження сторінки. Основним механізмом SPA є використання API для отримання даних та відображення оновленого контенту на клієнтській стороні (рисунок 1.3.4). Цей

підхід дозволяє покращити швидкість реакції застосунку, знизити навантаження на сервер та забезпечити плавний користувацький досвід. Водночас недоліками SPA є ускладнення SEO-оптимізації, оскільки початковий контент може не бути повністю доступним для пошукових систем, а також підвищені вимоги до ресурсів браузера, що може вплинути на продуктивність слабких пристроїв.

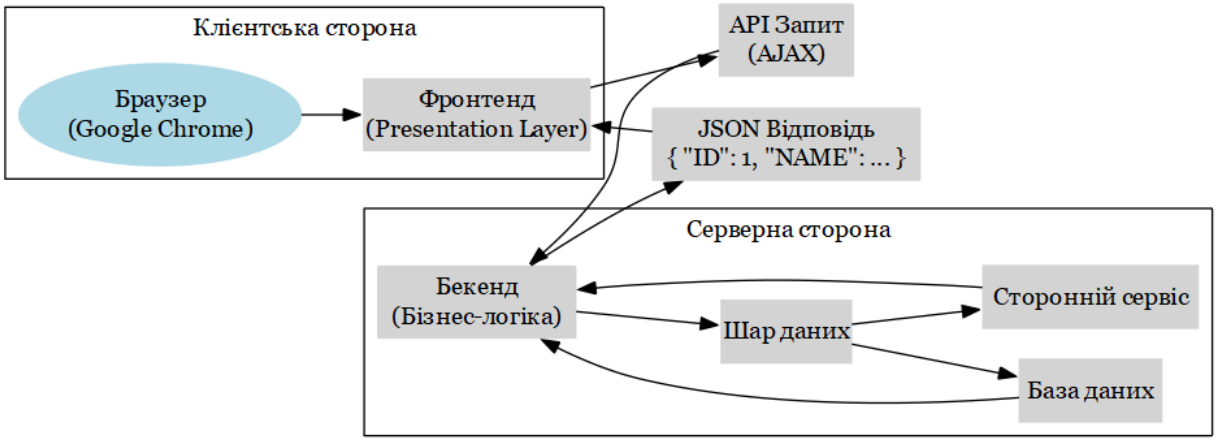


Рисунок 1.3.4 – Архітектура SPA-додатку

Багатосторінкові застосунки (MPA) мають класичну архітектуру, у якій кожен запит користувача призводить до завантаження нової HTML-сторінки з серверної частини. Цей підхід є традиційним для веб-сайтів з великою кількістю контенту, як-от новинні портали або електронні магазини. Перевагами MPA є краща SEO-оптимізація, оскільки пошукові системи безперешкодно індексують статичний контент, а також стабільність роботи незалежно від складності бізнес-логіки. Основним недоліком є повільніша взаємодія з користувачем через необхідність завантаження повних сторінок, що може призводити до затримок у відтворенні контенту.

Прогресивні веб-застосунки (PWA) поєднують переваги SPA та MPA, надаючи можливість працювати як із динамічними інтерфейсами, так і з попередньо завантаженим контентом. Використовуючи технології кешування та офлайн-доступу, PWA можуть функціонувати навіть без активного інтернет-з’єднання, що робить їх зручним рішенням для мобільних

користувачів. Додатковими перевагами PWA є можливість інтеграції з пристроєм, зокрема підтримка push-сповіщень та роботи у фоновому режимі. Разом із тим, повна підтримка PWA може бути обмеженою на деяких платформах, особливо в екосистемі iOS, що створює певні бар'єри для їх впровадження [23].

Таким чином, вибір архітектури веб-застосунку залежить від характеристик проєкту, зокрема від вимог до продуктивності, оптимізації для пошукових систем та інтерактивності.

1.4 Засоби розробки та розгортання веб-додатків

У процесі розробки та розгортання веб-додатків використовується широкий спектр інструментів і технологій, які охоплюють усі етапи життєвого циклу програмного забезпечення — від написання коду до його запуску в експлуатаційне середовище. Ці засоби поділяються на декілька категорій відповідно до функціонального призначення: засоби програмування, системи керування версіями, фреймворки, інструменти тестування, платформи для розгортання та хмарні сервіси. Основу розробки становлять *мови програмування* та *фреймворки*, які забезпечують зручне та ефективне створення функціональних компонентів веб-додатку. Для клієнтської частини (frontend) найпоширенішими є HTML, CSS, JavaScript, а також сучасні бібліотеки та фреймворки, зокрема React, Angular і Vue.js. Серверна частина (backend) зазвичай реалізується на основі мов Python, JavaScript (Node.js), PHP, Ruby або Java, а зручність та стандартизацію забезпечують відповідні фреймворки — наприклад, Django, Express, Laravel, Ruby on Rails, Spring Boot тощо.

Розглянемо детальніше найбільш популярні фреймворки, що застосовуються для розробки frontend. Особливої популярності в даній екосистемі набули фреймворки та бібліотеки JavaScript, серед яких слід виокремити React, Vue.js та Angular, які реалізують концепцію компонентного підходу до побудови інтерфейсів користувача.

React — це JavaScript-бібліотека, розроблена компанією Meta (раніше Facebook), яка спеціалізується на створенні багаторазових інтерфейсних компонентів. Основною особливістю React є віртуальний DOM (Document Object Model), який оптимізує оновлення сторінки й дозволяє значно підвищити продуктивність при роботі з динамічними даними. React надає можливість використовувати JSX — синтаксичне розширення JavaScript, що дозволяє описувати елементи інтерфейсу в декларативній формі. Бібліотека забезпечує ефективну інтеграцію з іншими інструментами й є основою для побудови масштабованих SPA (Single Page Application).

Vue.js — це прогресивний JavaScript-фреймворк, що орієнтований на поступове впровадження у проєкти, від малих віджетів до повноцінних SPA. Vue має простий синтаксис і зручну систему двостороннього зв'язку даних (two-way data binding), що робить його особливо привабливим для розробників із різним рівнем досвіду. Важливою перевагою Vue є його гнучка архітектура, що дозволяє легко масштабувати застосунок та інтегрувати сторонні бібліотеки. Крім того, Vue активно підтримується спільнотою та має розвинену екосистему, зокрема Vue Router, Vuex для керування станом і Vue CLI для генерації проєктів.

Angular — це повнофункціональний фреймворк, розроблений компанією Google, який ґрунтується на TypeScript. Angular реалізує модель MVVM (Model–View–ViewModel) та включає інструменти для двостороннього зв'язку даних, шаблонного зв'язування, ін'єкції залежностей, маршрутизації, тестування та компіляції коду. Angular вирізняється високим рівнем структурованості, що робить його зручним для реалізації великих корпоративних додатків. Однак через складність конфігурації та значний обсяг коду він вимагає глибших знань і досвіду в порівнянні з React або Vue [24, 25].

Загалом, вибір фреймворку залежить від багатьох факторів, таких як розмір та складність проєкту, вимоги до продуктивності, наявність готових компонентів, а також рівень кваліфікації команди розробників.

З метою забезпечення *контролю версій* та колективної роботи над проєктами широко використовуються системи на кшталт Git у поєднанні з платформами спільної розробки, такими як GitHub, GitLab або Bitbucket. Для автоматизації процесів збирання, тестування, інтеграції та розгортання застосовуються *CI/CD-системи* (Continuous Integration/Continuous Deployment), наприклад, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI тощо.

Тестування веб-додатків виконується з використанням як *модульних*, так і *інтеграційних* інструментів. До популярних засобів автоматизованого тестування належать Selenium, Cypress, Jest, Mocha, а також середовища для забезпечення тестування API, такі як Postman або SoapUI.

Розгортання веб-додатків відбувається за допомогою *виділених серверів*, платформ як Heroku, Vercel, Netlify, а також у *хмарних середовищах* (AWS, Microsoft Azure, Google Cloud Platform). У складніших випадках застосовуються *контейнеризація* (Docker) та *системи оркестрації контейнерів* (Kubernetes), що дозволяють ефективно керувати масштабованими розподіленими системами. Крім того, для керування конфігураціями та автоматизації розгортання використовуються засоби на кшталт Ansible, Terraform, Helm [21].

Таким чином, сучасна розробка веб-додатків передбачає інтеграцію різноманітних засобів, які забезпечують високу якість програмного забезпечення, безперервність доставки оновлень, стабільну роботу систем у промисловому середовищі та зручність масштабування.

Висновки до розділу 1

Таким чином, у даному розділі було розглянуто поширені методи цифрової обробки растрових зображень, що найчастіше використовуються у задачах обробки фотографічних зображень, та реалізовані в багатьох редакторах растрової графіки та інших інструментальних засобах обробки візуальних даних. Серед них досліджено як традиційні (коригування яскравості та контрастності, вирівнювання гістограми, порогова обробка,

застування згорткових фільтрів), так і методи інтелектуальної обробки, зокрема сегментація сцени, художня стилізація, генеративне видалення об’єктів, редагування та створення графіки за допомогою генеративного штучного інтелекту. Для реалізації інформаційної системи обробки зображень засобами глибокого навчання обрано метод автоматичного видалення фону, художньої обробки портретів та генеративне видалення людей на зображенні.

Зроблено аналіз існуючих видів веб-додатків, розглянуто їх основні класифікації за типом контенту, функціональним призначенням, обсягом вмісту тощо. Розглянуто архітектурні підходи до побудови веб-застосунків (трирівнева, мікросервісна та монолітна модель). Проаналізовано сучасні засоби розробки та розгортання веб-додатків (фреймворки, засоби тестування, розгортання та обслуговування). Для побудови застосунку для обробки зображень було обрано фреймворк Vue.

РОЗДІЛ 2. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОБРОБКИ ЗОБРАЖЕНЬ

2.1 Методи та алгоритми обробки зображень

Метод обробки зображень у даному застосунку реалізовано з використанням комбінації нейронних мереж та алгоритмів комп’ютерного зору, причому для кожного режиму застосовується специфічна послідовність операцій.

Режим видалення фону. На початковому етапі зображення проходить процес нормалізації, що включає коригування значень кожного каналу кольору (BGR) відповідно до статистичних параметрів – середнього значення та стандартного відхилення. Це забезпечує узгодженість вхідних даних, покращує їхню обробку нейронною мережею та мінімізує вплив можливих аномалій.

Після нормалізації зображення масштабується до розміру, необхідного для коректної роботи нейромережі. Далі воно подається на вхід глибокої ШНМ, яка здійснює сегментацію. Основною метою цього етапу є генерація маски, яка вказує ймовірність належності кожного окремого пікселя до переднього плану – тобто до значимих об'єктів зображення. Нейронна мережа в процесі навчання формує високоточні критерії розпізнавання об’єктів, що дозволяє досягти ефективного відокремлення фону.

Після отримання маски вона проходить процес постобробки: нормалізується та масштабується до вихідного розміру зображення. Це гарантує збереження коректних пропорцій та точності позиціонування об'єктів. Завершальний етап передбачає заміну альфа-каналу вихідного зображення на отриману маску, що дозволяє безпосередньо інтегрувати сегментовані області, виключаючи фон. Такий підхід забезпечує високу точність видалення фону, особливо при обробці деталізованих зображень.

Режим видалення людей на зображенні. Як і у випадку видалення фону, попередня обробка входу передбачає нормалізацію параметрів зображення та

його масштабування до розміру, оптимального для аналізу нейромережею. В результаті роботи моделі формується двійкова маска, що визначає ймовірність присутності людських об'єктів у кожному пікселі зображення.

Після генерації маски вона проходить етап постобробки, що включає нормалізацію та адаптацію до вихідного розміру зображення, забезпечуючи точну локалізацію видалених об'єктів. Для заміщення вилучених областей застосовується алгоритм інпейнтингу (inpainting), який використовує контекстну інформацію навколишнього середовища зображення для реконструкції відсутніх ділянок.

Завершальний етап включає конвертацію отриманого зображення у формат з альфа-каналом, що дозволяє збереження прозорості та подальше коректне відображення у багатошарових графічних редакторах або системах обробки зображень.

Режим створення художніх портретів. Основний принцип полягає у застосуванні ШНМ для генерації монохромного представлення зображення, що створює ефект графічного стилю, аналогічного традиційним художнім технікам.

Спочатку монохромне зображення проходить етап конвертації у трьохканальний формат, за яким слідує зміна розміру відповідно до розмірності вхідного шару ШНМ та нормалізація. У той же час вихідне зображення піддається розмиттю методом Гауса, який використовується для усунення дрібних деталей та створення плавного переходу тонів. Цей метод базується на згорткових операціях, що моделюють розподіл яскравості у відповідності до нормального розподілу, що дозволяє ефективно зменшити локальні контрастні перепади та досягти природної розмитості.

Далі реалізується процес зваженого змішування (blending) між розмитим зображенням та отриманою портретною маскою. Завдяки застосуванню вагових коефіцієнтів відбувається адаптивне поєднання контурів портрета із фонованими ділянками, що надає композиції м'якість та глибину. Такий підхід

дозволяє створити ефект художньої обробки, наближеної до стилю живописних або графічних портретів.

На завершальному етапі отриманий результат конвертується у формат з альфа-каналом, що забезпечує можливість гнучкого використання зображення в багатошарових графічних редакторах.

Блок-схеми методів перетворення зображень для кожного режиму роботи застосунку наведено на рисунках 2.1.1 – 2.1.3.

У процесі обробки зображень [26] у даному випадку нормалізація даних здійснюються відповідно до формули:

$$x'_{h,w,j} = \frac{\frac{x_{h,w,j}}{M} - \mu_j}{\sigma_j}$$

де $x'_{h,w,j}$ — значення пікселя у позиції (h, w) та каналі j, M — максимальне значення серед усіх пікселів (для приведення до діапазону [0,1]), μ_j та σ_j — середнє та стандартне відхилення для відповідного каналу (визначені як μ = [0.485, 0.456, 0.406]), σ = [0.229, 0.224, 0.225]).

Для приведення результату виведення до формату растрового зображення виконується зворотна нормалізація [26] за формулою:

$$y'_{h,w} = \frac{y_{h,w} - y_{\min}}{y_{\max} - y_{\min}} \cdot 255$$

де $y'_{h,w}$ — значення пікселя у матриці, $y_{\min}$ та $y_{\max}$ — мінімальне та максимальне значення у матриці відповідно.

Розмиття в виконується за Гаусом [27]: коефіцієнти для побудови ядра згортки обчислюються за формулою нормального розподілу:

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

де σ – стандартне відхилення нормального розподілу, яке визначає ступінь розмиття, x і y – зміщення від центру ядра по горизонталі та вертикалі.

У випадку створення портретів використовується зважене змішування зображень. Формально для змішування двох зображень A та B однакових розмірів результат змішування C обчислюється за формулою:

$$C = \alpha \cdot A + \beta \cdot B + \gamma$$

де α та β — вагові коефіцієнти (зазвичай такі, що $\alpha + \beta = 1$), а γ — додатковий зсув (bias), який часто дорівнює нулю.

Застосований метод інпейнтингу, описаний в [28], заснований на методі швидкого маршу [29]. Основна ідея полягає у поступовому поширенні кольорової інформації з меж пошкоджених областей у їхню внутрішню частину, забезпечуючи плавне продовження ізофот (ліній рівного значення яскравості). Цей алгоритм є простим у реалізації та забезпечує швидке відновлення зображень при збереженні їхньої структурної цілісності. Він застосовується для видалення тексту, логотипів, реконструкції пошкоджених зображень та створення художніх ефектів.

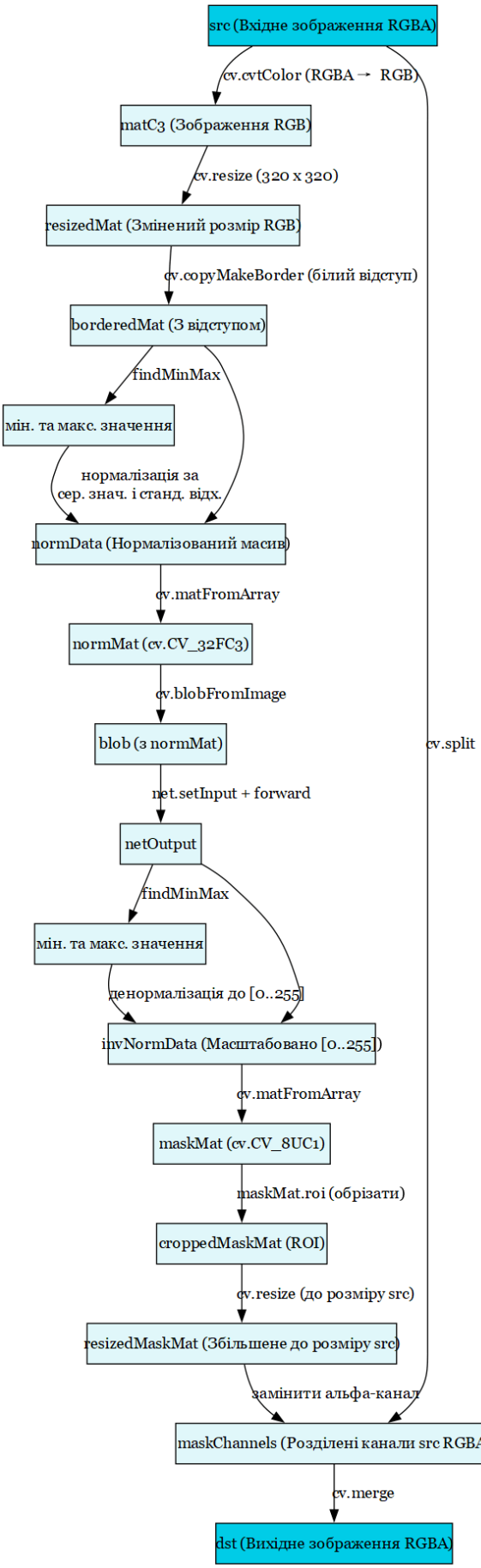


Рисунок 2.1.1 – Блок-схема конвеєру обробки зображення для видалення фону

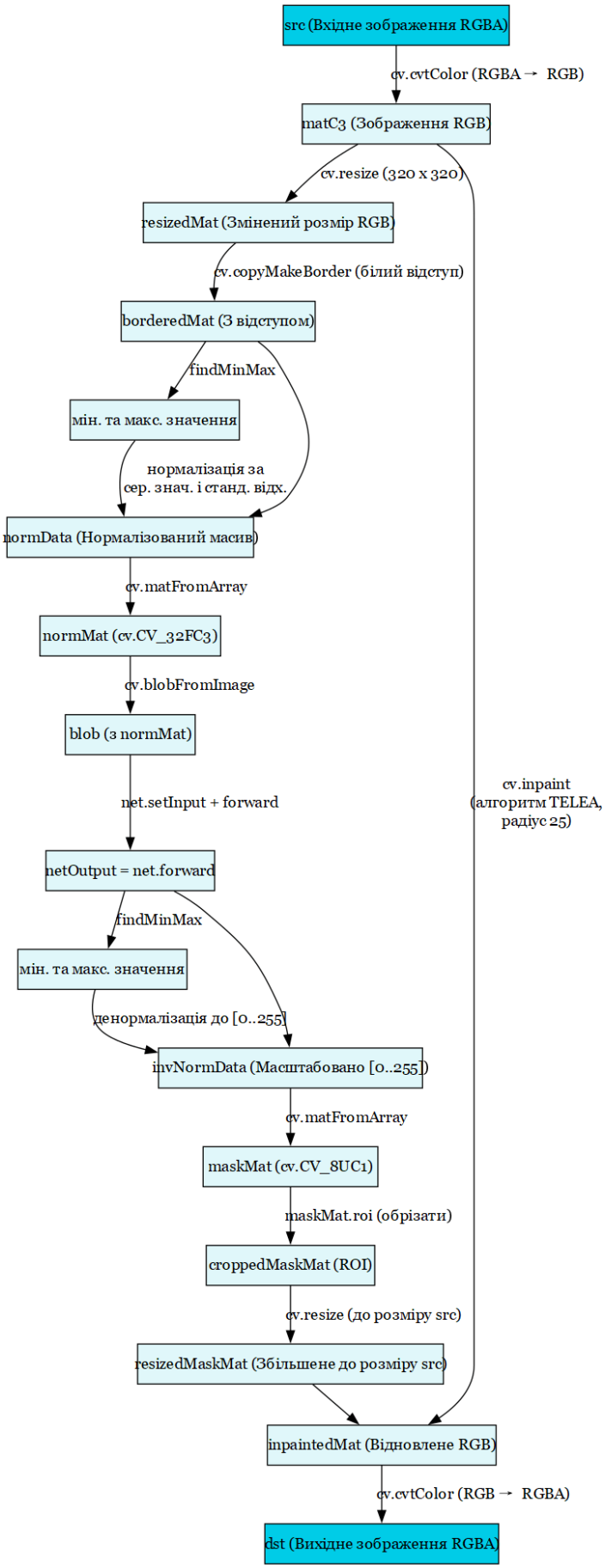


Рисунок 2.1.2 – Блок-схема конвеєру обробки зображення для видалення зображень людей

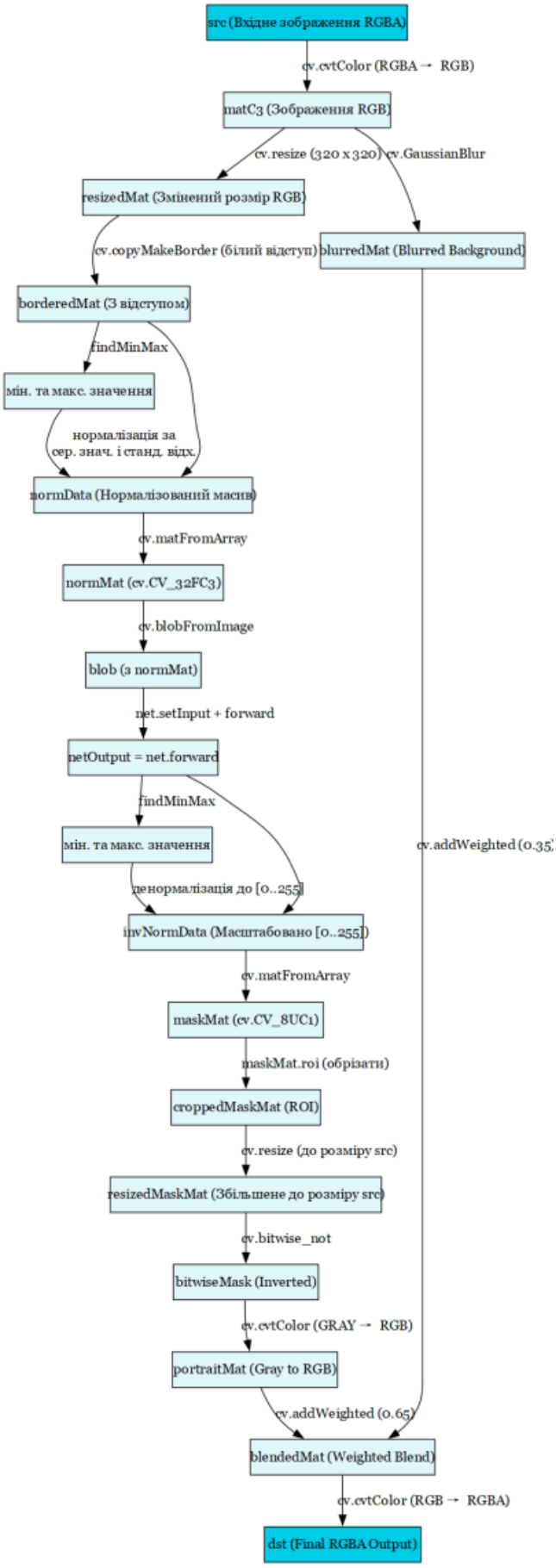


Рисунок 2.1.3 – Блок-схема конвеєру обробки зображення для створення портретів

2.2 Методи глибокого навчання

Як основу для моделей ШНМ для всіх перетворень було обрано U^2-Net [30] через її ефективну архітектуру, яка дозволяє мережі досягати високої роздільної здатності, не суттєво збільшуючи обчислювальні витрати та обсяги пам’яті. Модель базується на дворівневій вкладеній U-структурі, що складається з 11 стадій: 6 кодувальників та 5 декодувальників (рисунок 2.2.1). На нижньому рівні використовується новаторський модуль ReSidual U-block (RSU), який здатний виділяти внутрішні мультимасштабні ознаки, не знижуючи роздільної здатності карти ознак. На верхньому рівні мається структура, схожа на U-Net, де кожний етап є блоком RSU. Така модель ШНМ дозволяє досягти високої якості перетворення, зменшуючи витрати ресурсів та спрощуючи обчислювальний процес. Повна структурна схема ШНМ U^2-Net зображена в додатку А. Застосування RSU- модуля, в свою чергу, дозволяє зберігати важливі характеристики об'єктів на різних масштабах [31].

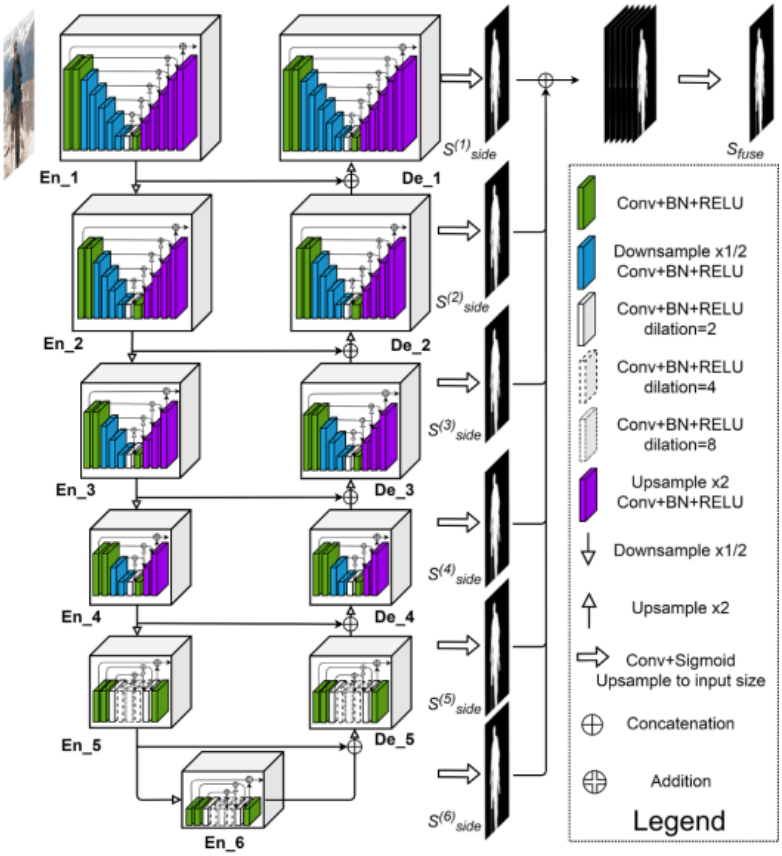


Рисунок 2.2.1 – Архітектура ШНМ U2-Net

Тренування ШНМ для перетворення зображення здійснюється шляхом мінімізації функції втрат методом градієнтного спуску. Використано оптимізатор Adam з початковою швидкістю навчання 0,001. Тренування тривало до збіжності функції втрат, що зазвичай займало близько 600 000 ітерацій з розміром пакету 12.

Під час тренування для задач сегментації використовується бінарна крос-ентропійна функція втрат (Binary Cross-Entropy Loss), яка визначає різницю між передбаченою маскою сегментації та еталонною (реальною) маскою [32]. Функція бінарної крос-ентропії для одного пікселя задається наступним рівнянням:

$$L_{bce}(p,g) = -\left[g \cdot \log(p) + (1-g) \cdot \log(1-p)\right]$$

Де $p \in [0,1]$ — передбачене значення ймовірності для пікселя (результат сигмоїди); $g \in 0,1$ — еталонне (істинне) значення пікселя (0 — фон, 1 — об'єкт).

Оскільки U²-Net має кілька вихідних рівнів (6 проміжних карт ймовірності та одну фінальну), загальна функція втрат обчислюється як сума втрат по всіх виходах, що допомагає прискорити та стабілізувати процес навчання.

Загальна функція втрат:

$$L_{total} = \sum_{i=1}^{6} L_{bce}(S_i, G) + L_{bce}(S_{fuse}, G)$$

Де S_i — вихід на і-му рівні сегментації; S_{fuse} — остаточний вихід; G — еталонна маска сегментації.

Для задачі видалення фону в процесі тренування U²-Net використовувався датасет DUTS-TR (рисунок 2.2.2), який є найбільшим та найбільш часто використовуваним набором даних для задачі виявлення об'єктів переднього плану. Цей датасет містить 10 553 зображення, які були доповнені за допомогою горизонтального відображення, що збільшило обсяг тренувального набору до 21 106 зображень.

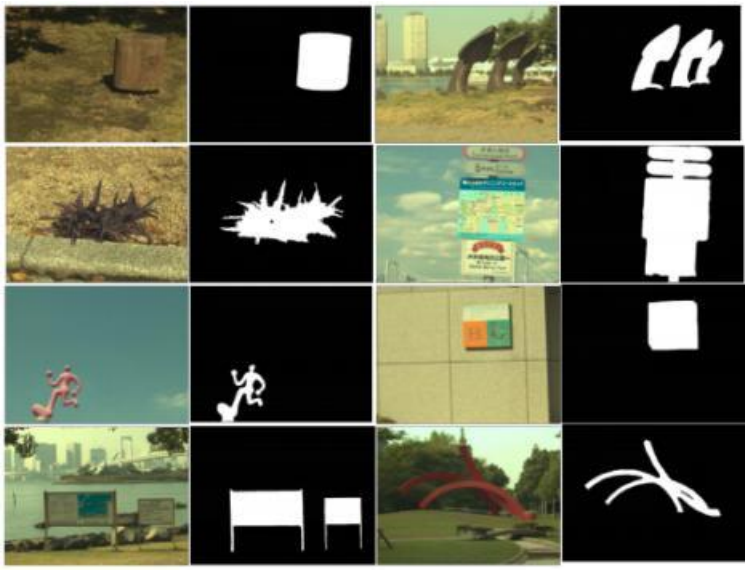


Рисунок 2.2.2 – Приклад зображень датасету DUTS-TR

Для вирішення задачі сегментації людей, мережа U²-Net була адаптована шляхом тренування на іншому наборі даних - модель була навчена на Supervisely Person Dataset (рисунок 2.2.3), який містить зображення людей з відповідними масками сегментації. Цей датасет забезпечує різноманітність поз та оточень, що сприяє підвищенню здатності моделі до узагальнення даних.

Рисунок 2.2.3 – Приклад зображень датасету Supervisely Person Dataset

Для задачі генерації художніх портретів на основі фотографій, U²-Net була навчена на APDrawingGAN dataset, який складається із 140 пар фотографій обличь та відповідних художніх малюнків, створених професійним художником (рисунок 2.2.4). Цей підхід дозволив моделі навчитися відображати риси обличчя у стилі художнього малюнка, забезпечуючи високу якість та художню виразність згенерованих зображень.

Рисунок 2.2.4 – Приклад зображень датасету APDrawingGAN

Висновки до розділу 2

У цьому розділі кваліфікаційної роботи магістра було розроблено методи обробки зображень для обраних завдань, кожен з яких є комбінацією алгоритмів комп’ютерного зору та глибокої нейронної мережі U²-Net. Остання була обрана через ефективну згорткову архітектуру, що включає глибокі згорткові структури з вузькими місцями, поєднані залишковими зв’язками. Така архітектура дозволяє їй обробляти багатомасштабні патерни, що робить її ефективною в широкому колі задач перетворення великорозмірних зображень. Завдяки універсальності моделі в залежності від завдань змінюються лише параметри моделі без необхідності зміни архітектури.

Було реалізовано метод автоматичного видалення фону, що використовує нейромережу для виконання сегментації, залишаючи лише область, що відповідає об’єкту на передньому плані. Метод видалення людей на зображенні використовує нейромережу для отримання сегментів, що відповідають зображенням людей, після чого виконується генеративне заповнення цих ділянок. Метод художньої стилізації портретів використовує нейромережу, навчену для цієї задачі спільно зі зваженим змішуванням із оригінальним зображенням.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ В ФОРМАТІ ВЕБ-ЗАСТОСУНКУ

3.1 Проєктування архітектури застосунку

Архітектура даного програмного продукту побудована на основі компонентного підходу, притаманного сучасним односторінковим веб-додаткам. Основу структури складають окремі компоненти, які відповідають за різні аспекти функціональності та інтерфейсу користувача. Компоненти розділені на логічні групи: базові елементи інтерфейсу (header, footer), функціональні модулі (process, dnnWorker, credentials), а також спеціалізовані UI-компоненти (canvas, overlay, tabstack), що забезпечують взаємодію з користувачем та відображення даних.

Кожен з компонентів виконує чітко визначену роль у системі. Та компонент header відповідає за навігацію та вибір мови, footer — за відображення інформації про авторські права та посилання. Функціональний компонент process реалізує основну бізнес-логіку обробки зображень, взаємодіючи з допоміжними модулями, такими як dnnWorker, що відповідає за виконання обчислювальних задач на стороні клієнта.

Важливою частиною архітектури є компоненти, що відповідають за відображення різних сторінок (HomeView, BGRemoveView, PortraitCreateView, PeopleRemoveView, AboutView). Вони організовані відповідно до маршрутизації додатку та забезпечують розмежування функціональних зон. Для забезпечення гнучкості та повторного використання коду використовуються вкладені компоненти, такі як tabstack для організації вкладок, canvas для роботи з графічними даними та overlay для відображення повідомлень.

Головний компонент App виконує роль кореневого контейнера для всього додатку. Він ініціалізує основну структуру інтерфейсу користувача, організовуючи розміщення «шапки» (header), «підвалу» (footer), а також динамічний вміст, який змінюється відповідно до обраного маршруту. У

структурі App використовується роутер, який дозволяє відображати відповідний компонент сторінки залежно від поточного маршруту, що забезпечує гнучку навігацію у межах односторінкового додатку.

Загалом, архітектура даного програмного забезпечення орієнтована на модульність, масштабованість та підтримку повторного використання коду. Всі компоненти взаємодіють між собою через чітко визначені інтерфейси, що спрощує супровід та розширення функціональності додатку. Такий підхід забезпечує високу якість коду, зручність тестування та адаптацію до змін вимог у майбутньому. Розглядаючи кожен компонент як окремий функціональний блок, можна створити класову UML-діаграму [33], що візуалізує обрану архітектуру веб-додатку (рисунок 3.1.1).

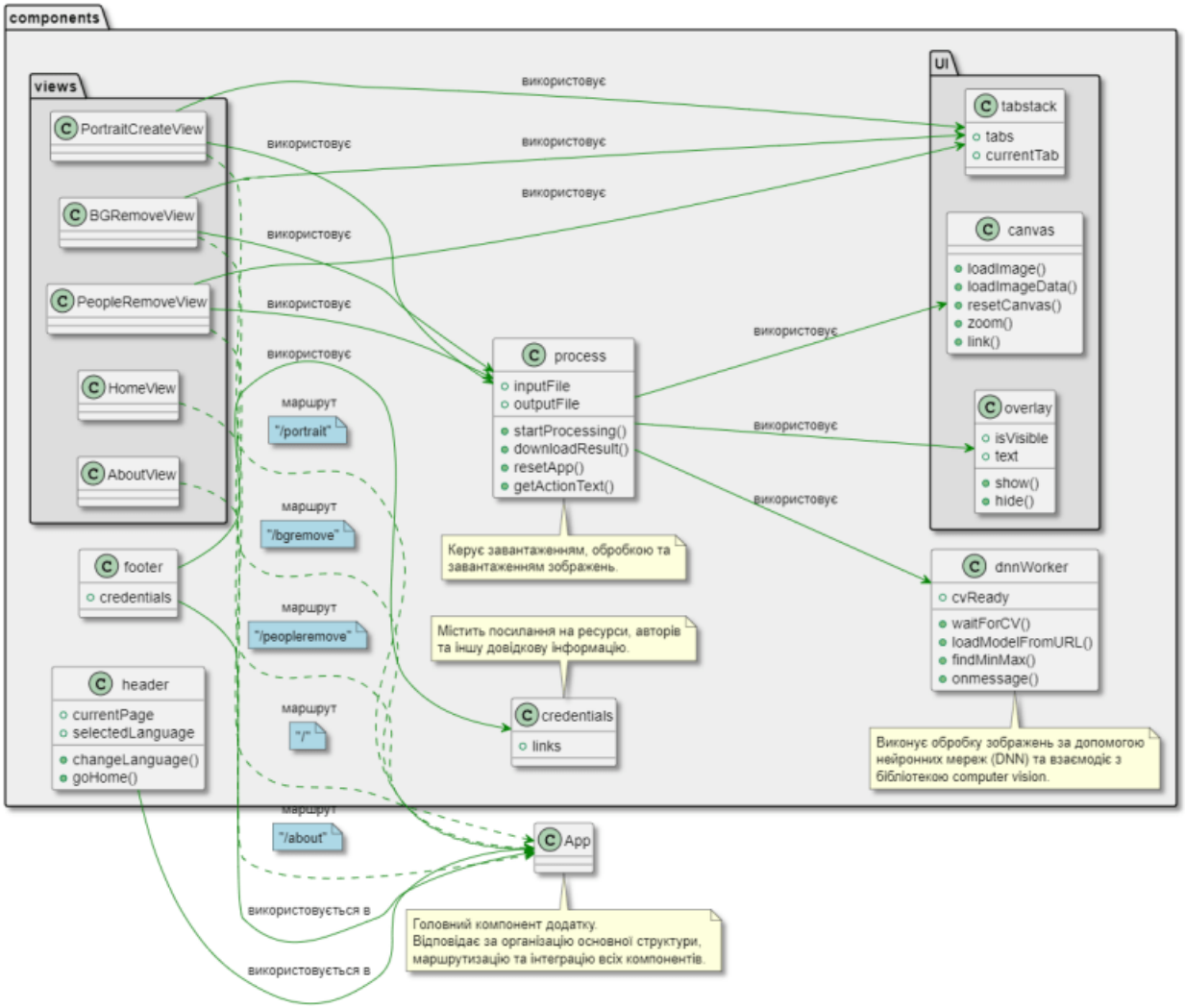


Рисунок 3.1.1 – UML-діаграма класів web-застосунку

3.2 Вибір програмних засобів

Перед тим як приступити до програмної реалізації веб-застосунку для обробки растрових зображень методами глибокого навчання було здійснено вибір відповідного програмного забезпечення, що дозволило б забезпечити гнучкість, продуктивність, масштабованість та простоту розгортання. Основною вимогою до платформи є можливість реалізації односторінкового застосунку (SPA) із повністю клієнтською архітектурою (лише frontend), здатного виконувати обчислення локально на стороні браузера.

У якості основного фреймворку для побудови користувацького інтерфейсу було обрано Vue.js — прогресивний JavaScript-фреймворк, орієнтований на розробку інтерактивних веб-інтерфейсів. Його перевагою є модульність, легка інтеграція в HTML-сторінки без потреби у складній конфігурації або системах збірки (Webpack, Vite тощо) [34]. У даному програмному застосунку Vue.js використовується у вигляді ES-модуля (версія без системи збірки), що дозволяє включати компоненти безпосередньо у браузер середовища виконання, знижуючи поріг входу, спрощуючи розгортання та забезпечуючи кращу контрольованість коду.

Для організації маршрутизації між логічними сторінками застосунку використовується офіційна бібліотека Vue Router. Вона забезпечує динамічне керування маршрутами у межах SPA-застосунку, дозволяє створювати вкладені маршрути, керувати історією переходів, а також реалізовувати навігацію без перезавантаження сторінки. Це особливо важливо при реалізації декількох функціональних модулів (наприклад, видалення фону, стилізації портретів, налаштувань тощо) в межах одного застосунку.

Для забезпечення багатомовності інтерфейсу застосовується бібліотека Vue I18n, яка дозволяє легко інтегрувати підтримку декількох мов. Це розширює доступність застосунку для користувачів з різною мовною компетенцією та відповідає сучасним вимогам до інтернаціоналізації програмного забезпечення. Завдяки динамічному завантаженню ресурсів

перекладу, i18n не впливає на продуктивність застосунку та зберігає його модульність.

Для реалізації обробки зображень безпосередньо у браузері обрано бібліотеку OpenCV.js, що є портованою до WebAssembly версією популярної бібліотеки OpenCV. Використання WebAssembly забезпечує високу продуктивність, наближену до нативного виконання, що дозволяє виконувати складні операції над зображеннями (фільтрація, сегментація, морфологічні перетворення та навіть виведення у глибоких ШНМ) без необхідності серверної обробки. Крім того, OpenCV.js має добре задокументований API, підтримку роботи з HTML5-елементами (<canvas>, <img>, <video> тощо) та активну спільноту розробників.

Було обрано модуль OpenCV DNN (Deep Neural Networks) як основний засіб для виконання операцій глибокого навчання. Цей модуль входить до складу OpenCV та підтримується у WebAssembly-версії бібліотеки, що дозволяє використовувати його в браузері без залучення серверних ресурсів. Однією з ключових переваг модуля OpenCV DNN є його універсальність — він підтримує імпорт та виконання моделей, збережених у різних форматах, зокрема TensorFlow, Caffe, ONNX, Torch. Це забезпечує можливість використання попередньо навчених моделей без необхідності їх конвертації у нестандартні формати або адаптації під специфічні фреймворки. Такий підхід знижує час на інтеграцію моделей та дозволяє зосередитися на реалізації прикладної логіки застосунку.

Модуль має спрощений API, який дозволяє швидко виконувати forward-проходження через модель, здійснювати попередню обробку даних (нормалізацію, масштабування) та постобробку результатів. Завдяки компіляції до WebAssembly, OpenCV DNN забезпечує високу продуктивність у браузері. Іншою суттєвою перевагою є інтеграція з іншими модулями OpenCV, що дозволяє легко комбінувати результати нейромережевого аналізу з класичними алгоритмами комп’ютерного зору. Це особливо актуально у

випадках, коли потрібно поєднувати сегментацію зображення, фільтрацію, геометричні трансформації або морфологічні операції.

Отже, обрані програмні засоби забезпечують баланс між функціональністю, продуктивністю та простотою інтеграції, що дозволяє реалізувати клієнтський SPA-застосунок з повноцінною локальною обробкою зображень без необхідності у серверній інфраструктурі.

3.3 Особливості програмної реалізації

HTML-документ index.js (рисунок 3.3.1) реалізує односторінковий веб-застосунок (SPA) із використанням сучасного стеку технологій на основі фреймворку Vue.js третьої версії. У секції <head> визначено метатеги для SEO-оптимізації, а також підключено favicon. Особливістю є використання import map, що дозволяє явно вказати шляхи до зовнішніх бібліотек (Vue, Vue Router, Vue I18n) через CDN, забезпечуючи модульність і гнучкість підключення залежностей.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Background Removal Tool">
    <meta name="author" content="Andrii Krokhmal">
    <meta name="keywords" content="background removal, portrait creation, people removal, image processing, AI, machine learning, deep learning, U-2-Net">

    <title></title>
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
</head>
```

```
    import { createApp } from 'vue';
    import { createI18n } from 'vue-i18n';
    import { createRouter, createWebHistory, RouterView, RouterLink } from 'vue-router';

    import CustomHeader from '/components/header.js';
    import CustomFooter from '/components/footer.js';
```

```
const router = createRouter({
    history: createWebHistory(),
    routes,
});

const app = createApp({
    components: { RouterView, RouterLink, CustomHeader, CustomFooter },
});
app.use(i18n);
app.use(router);
app.mount('#app');
```

```
<div id="app">
    <link rel="stylesheet" href="/style/main.css">
    <custom-header></custom-header>
    <router-view></router-view>
    <div style="flex: 1"></div>
    <custom-footer></custom-footer>
</div>
```

Рисунок 3.3.1 – Ключові фрагменти програмного коду index.html

Основна логіка ініціалізації застосунку реалізована у скрипті з типом module. Тут імпортуються необхідні функції та компоненти з Vue, Vue Router, Vue I18n, а також користувацькі компоненти заголовка та підвалу. Для підтримки багатомовності створюється об’єкт messages, що містить локалізовані ресурси трьома мовами (англійською, українською, німецькою) для різних сторінок застосунку. Інтернаціоналізація реалізована через створення екземпляра i18n з fallback на англійську мову.

Маршрутизація організована за допомогою Vue Router у режимі історії браузера (createWebHistory). Кожен маршрут асоціюється з асинхронно імпортованим компонентом відповідної сторінки, що сприяє оптимізації завантаження (lazy loading). Це дозволяє завантажувати лише необхідний код для поточної сторінки, зменшуючи початковий час завантаження застосунку.

Головний екземпляр застосунку Vue створюється з метою реєстрації основних компонентів (RouterView, RouterLink, CustomHeader, CustomFooter) та підключення плагінів i18n і router. Монтований застосунок прив’язується до DOM-елемента з ідентифікатором app. У розмітці цього елемента підключається основний CSS-файл, а також розміщуються компоненти заголовка, підвалу та динамічний контейнер для відображення поточного маршруту (<router-view>).

Компонент шапки сайту реалізовано як окремий Vue-компонент із використанням опції export default. Його шаблон містить структуровану розмітку, що включає логотип, динамічний заголовок і підзаголовок сторінки, а також панель навігації з перемикачем мов. Для стилізації підключається окремий CSS-файл безпосередньо у шаблоні, що забезпечує ізольованість стилів компонента.

Важливою особливістю є динамічне формування заголовка та опису сторінки через обчислювану властивість currentPage, яка використовує механізм локалізації $t для отримання відповідних текстів залежно від активного маршруту та вибраної мови (рисунок 3.3.2). Це дозволяє забезпечити багатомовність інтерфейсу без дублювання коду.

Рисунок 3.3.2 – Фрагмент програмного коду header.js

Крім того, при зміні маршруту або мови автоматично оновлюється заголовок сторінки у вкладці браузера, що підвищує зручність користування. Механізм перемикання мов реалізовано через випадаючий список, значення якого зв’язане з реактивною змінною selectedLanguage. Зміна мови викликає відповідний метод, який оновлює локаль у глобальному екземплярі i18n. Для кожної мови передбачено відображення прапорця та назви, що підвищує інтуїтивність вибору для користувача. Також у компоненті реалізовано навігаційне посилання на головну сторінку, яке відображається лише якщо користувач знаходиться не на головній.

Компонент сторінки видалення фону BGRemoveView реалізовано як окремий Vue-компонент із використанням сучасного підходу до організації інтерфейсу на основі вкладених компонентів та багатомовності. Основу шаблону складає компонент TabStack, який відповідає за відображення вкладок і їхнього вмісту. Для кожної вкладки визначено окремий слот: основна вкладка містить компонент Process із режимом salient_object, що безпосередньо реалізує функціонал видалення фону; вкладка "Допомога" містить структурований перелік кроків для користувача; вкладка "Про програму" – інформацію про інструмент (рисунок 3.3.3). Така організація забезпечує логічне розділення функціоналу та зручність навігації.

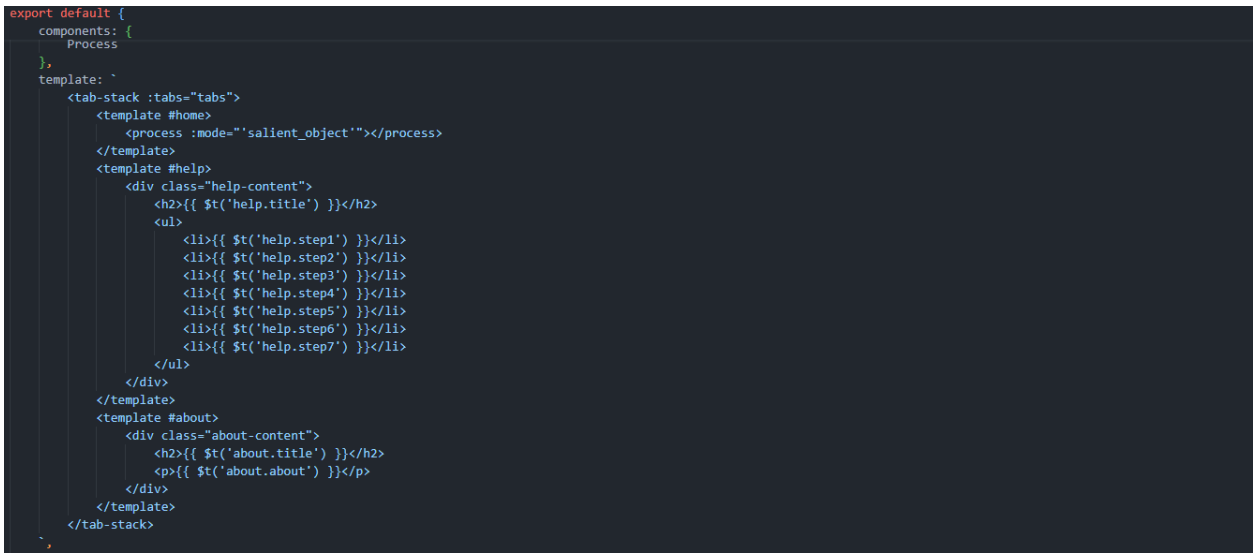


Рисунок 3.3.3 – Фрагмент програмного коду BGRemoveView.js

Важливою деталлю є використання обчислюваної властивості tabs, яка формує масив вкладок із локалізованими назвами, отриманими через $t. Це дозволяє динамічно змінювати мову інтерфейсу без перезавантаження сторінки. Локалізовані ресурси для вкладок, інструкцій та опису інструменту визначені безпосередньо у компоненті через опцію i18n, що забезпечує незалежність і гнучкість локалізації.

Компонент обробки зображень Process реалізовано як універсальний Vue-компонент із підтримкою багатомовності, асинхронної обробки та інтерактивного інтерфейсу (рисунок 3.3.4). Його шаблон містить кілька ключових елементів: область для завантаження зображення шляхом перетягування або вибору файлу, дві області для відображення вхідного та вихідного зображення на кастомних канвасах, а також набір кнопок для запуску обробки, завантаження результату та скидання стану.

Взаємодія з користувачем організована через реактивні змінні inputFile та outputFile, які визначають стан інтерфейсу: до завантаження зображення відображається лише зона завантаження, після – канваси та кнопки керування. Завантаження зображення можливе як через drag-and-drop, так і через стандартний діалог вибору файлу. Відповідні методи обробляють події,

створюють URL для попереднього перегляду та ініціалізують завантаження зображення у CustomCanvas.

Ключовою особливістю є асинхронна обробка зображення за допомогою Web Worker (dnnWorker.js). Після запуску обробки компонент відправляє дані зображення та режим роботи воркеру, а також відображає оверлей із прогресом, текст якого локалізовано. Відповіді воркера обробляються через події: при отриманні результату зображення відображається на вихідному канвасі, при помилках — користувач отримує відповідне повідомлення. Такий підхід дозволяє не блокувати основний потік інтерфейсу під час обчислень.

```
methods: {
  triggerFileInput() {
    this.$refs.fileInput.click();
  },
  async handleDrop(event) {
    event.preventDefault();
    const file = event.dataTransfer.files[0];
    if (file) {
      await this.loadFile(file);
    }
  },
  async handleFileSelect(event) {
    const file = event.target.files[0];
    if (file) {
      await this.loadFile(file);
    }
  },
  async loadFile(file) {
    let imgUrl = URL.createObjectURL(file);
    this.inputFile = (await this.$refs.canvasInput.loadImage(imgUrl)).src;
  },
  async startProcessing() {
    let canvas = this.$refs.canvasOutput;
    let overlay = this.$refs.overlay;
    overlay.text = this.$t("home.overlay.prepare");
    overlay.show();
    let imgData = this.$refs.canvasInput.imagedata;
    const worker = new Worker("/components/dnnWorker.js");
    worker.postMessage({image: imgData, mode: this.mode});
```

```
    warnUnsavedChanges(event) {
      if (this.inputFile) {
        event.preventDefault();
        event.returnValue = "";
      }
    },
  },
  mounted() {
    this.$refs.canvasInput.link(this.$refs.canvasOutput);
  },
  created() {
    window.addEventListener("beforeunload", this.warnUnsavedChanges);
  },
  beforeDestroy() {
    window.removeEventListener("beforeunload", this.warnUnsavedChanges);
  },
```

Рисунок 3.3.4 – Ключові фрагменти програмного коду process.js

Компонент підтримує різні режими роботи (видалення фону, створення портрету, видалення людей), що визначається через prop mode і впливає на текст кнопки дії та логіку обробки. Для захисту від втрати незбережених змін реалізовано обробник події beforeunload, який попереджає користувача при

спробі закрити сторінку з незбереженим результатом. Також передбачено повне скидання стану для повторної обробки нових зображень.

Веб-воркер dnnWorker.js (рисунок 3.3.5) реалізує асинхронну обробку зображень із використанням бібліотеки OpenCV.js, що дозволяє виконувати складні обчислення поза основним потоком інтерфейсу. Завантаження OpenCV.js здійснюється через importScripts, після чого ініціалізація бібліотеки контролюється через обробник cv['onRuntimeInitialized'], що гарантує готовність функціоналу перед початком обробки. Для цього використовується асинхронна функція waitForCV, яка очікує завершення ініціалізації OpenCV.

```
self.importScripts('https://docs.opencv.org/4.10.0/opencv.js'); // Import OpenCV.js

let cvReady = false;
cv['onRuntimeInitialized'] = () => {
    console.log('OpenCV.js is ready.');
    cvReady = true;
};

const waitForCV = () => {
    return new Promise((resolve) => {
        const check = () => {
            if (cvReady) {
                resolve();
            } else {
                setTimeout(check, 10); // Check every 10ms
            }
        };
        check();
    });
};
```

```
self.onmessage = async (e) => {
    self.postMessage({ progress: "prepare" });

    await waitForCV(); // Wait until OpenCV is initialized

    self.postMessage({ progress: "loadModel" });
```

```
    const resultImageData = new ImageData(
        new Uint8ClampedArray(dst.data),
        dst.cols,
        dst.rows
    );

    dst.delete();

    // Send result back to main thread
    self.postMessage({ progress: "done" });
    self.postMessage({ imageData: resultImageData });
```

Рисунок 3.3.5 – Ключові фрагменти програмного коду dnnWorker.js

Комунікація з основною програмою організована через обробник подій onmessage, який приймає дані для обробки (зображення, режим роботи) та надсилає повідомлення про стан виконання (прогрес, помилки, результат) за допомогою postMessage. Воркер поетапно інформує основний потік про підготовку, завантаження моделі, попередню обробку, виведення та постобробку, що дозволяє відображати прогрес користувачу. Після завершення обробки результат повертається у вигляді об’єкта ImageData. Така

архітектура забезпечує ефективний розподіл навантаження, ізоляцію обчислень та стабільну інтеграцію з основним застосунком на Vue.js.

Компонент холста (CustomCanvas) реалізовано як окремий Vue-компонент для інтерактивного відображення та обробки зображень у межах веб-застосунку. Його структура побудована на реактивних даних, що зберігають стан зображення, масштабування, позиціювання, а також підтримують зв’язок із іншим канвасом для синхронного відображення для порівняння вхідного та вихідного результату.

Завантаження зображення реалізовано через асинхронні методи loadImage та loadImageData, які дозволяють працювати як із зовнішніми файлами, так і з об’єктами ImageData. Після завантаження зображення автоматично виконується масштабування під розмір канвасу та збереження піксельних даних для подальшої обробки. Окремий метод resetCanvas дозволяє повністю очистити стан компонента, а resetZoom — повернути масштаб до початкового або заданого значення.

Взаємодія з користувачем реалізована через обробку подій миші та сенсорних подій: підтримується масштабування колесом миші, панорамування (переміщення зображення) шляхом drag-and-drop, а також відповідні жести на сенсорних пристроях. Для цього визначено низку методів (handleWheel, handleMouseDown, handleMouseMove, handleTouchStart тощо), які змінюють стан компонента та викликають оновлення канвасу (рисунок 3.3.6).

Особливістю компонента є підтримка адаптивності: при зміні розміру вікна або контейнера автоматично викликається метод handleResize, що підлаштовує розміри канвасу та масштабує зображення. Для цього використовується IntersectionObserver та обробка події resize.

Додатково, для зручності користувача, під канвасом розміщено панель керування з кнопками масштабування та скидання, а також відображенням поточного масштабу з локалізованим підписом.

```
    handleWheel(event) {
        event.preventDefault();
        this.zoom(event.deltaY < 0, event.offsetX, event.offsetY)
    },
    handleMouseDown(event) {
        this.isPanning = true;
        this.startX = event.clientX - this.originX;
        this.startY = event.clientY - this.originY;
    },
    handleMouseMove(event) {
        if (!this.isPanning) return;
        this.originX = event.clientX - this.startX;
        this.originY = event.clientY - this.startY;
        this.redrawCanvas();
    },
    handleMouseUp() {
        this.isPanning = false;
    },
```

```
handleResize() {
    const container = this.$parent.$el.querySelector('.canvas-container');
    const canvas = this.$refs.canvasRef;
    if (container.clientWidth > 0) {
        if (canvas.width != container.clientWidth) {
            canvas.width = container.clientWidth;
            canvas.height = container.clientWidth;
            if (this.img.src.length > 0) {
                this.resetZoom();
            } else {
                this.redrawCanvas();
            }
        }
    }

},
redrawCanvas(sendBack = true) {
    const canvas = this.$refs.canvasRef;
    const ctx = canvas.getContext("2d");
    this.clearCanvas();
    ctx.setTransform(this.scale, 0, 0, this.scale, this.originX, this.originY);
    ctx.drawImage(this.img, 0, 0);
    if (this.linkedCanvas && sendBack) {
        this.linkedCanvas.scale = this.scale;
        this.linkedCanvas.originX = this.originX;
        this.linkedCanvas.originY = this.originY;
        this.linkedCanvas.redrawCanvas(false);
    }
},
```

Рисунок 3.3.6 – Ключові фрагменти програмного коду canvas.js

Повний лістинг програмного коду розробленого веб-застосунку наведено в додатку Б.

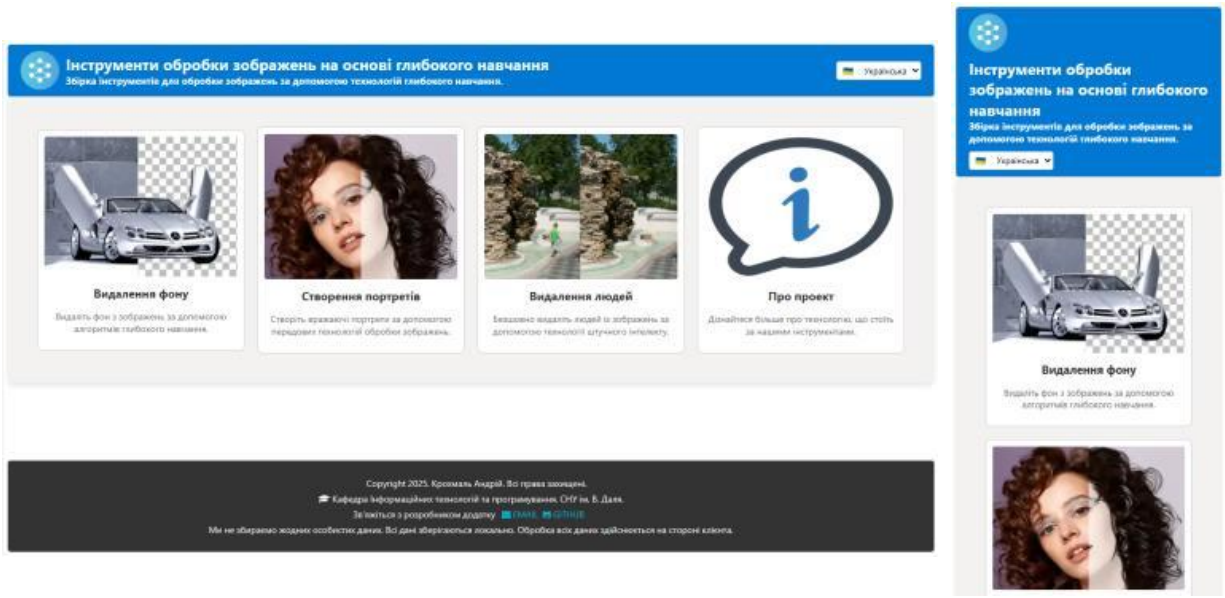
3.4 Приклад роботи застосунку

При переході на сторінку веб-застоунку, потрапляємо на домашню сторінку (рисунок 3.4.1). Вона містить чотири основні розділи, пов’язані з обробкою зображень на основі глибокого навчання:

- *Видалення фону* – дозволяє користувачам швидко видаляти фон із зображень.
- *Створення портретів* – призначене для генерації портретів.
- *Видалення людей* – дозволяє автоматично прибирати людей із фотографій.

- *Про проект* – надає інформацію про сам застосунок та його розробку.

У верхній частині сторінки є заголовок, що пояснює головну мету веб-застосунку, а в нижній частині – підвал (футер) із авторськими правами, контактами та інформацією про конфіденційність даних.



а) б)

Рисунок 3.4.1 – Головна сторінка веб-застосунку: а) вигляд на ПК, б) вигляд на мобільному пристрої

Головна сторінка веб-застосунку спроєктована з урахуванням принципів UI/UX, забезпечуючи зручну та інтуїтивну взаємодію користувачів із системою. Візуальна структура є мінімалістичною, що сприяє швидкому засвоєнню інформації та ефективному навігаційному процесу. Основні функціональні розділи представлені у формі чітко виражених інтерактивних елементів, що полегшує доступ до ключових можливостей застосунку, таких як видалення фону, створення портретів та видалення людей із зображень.

Колірна палітра та типографіка підібрані таким чином, щоб забезпечити комфортне візуальне сприйняття та уникнути перевантаження елементами. Використання адаптивного дизайну дозволяє забезпечити оптимальну продуктивність веб-застосунку на різних пристроях, включаючи смартфони та

планшети, що підвищує рівень доступності для користувачів. Крім того, передбачено логічну ієрархію інформації, що сприяє легкому орієнтуванню навіть для нових користувачів.

Інтерактивні елементи, такі як кнопки та анімовані переходи між розділами, використовуються з метою покращення користувацького досвіду. Відгук на дії користувачів є миттєвим, а завантаження сторінки відбувається без затримок, що сприяє загальному відчуттю плавності та ефективності взаємодії. Футер містить корисну інформацію, зокрема деталі про конфіденційність даних та авторські права, що демонструє увагу до юридичних аспектів користування.

При натисканні на кнопку, що позначає один з варіантів обробки, роутер спрямовує користувача на відповідну сторінку. Вигляд сторінки для видалення фону зображень наведено на рисунку 3.4.2.

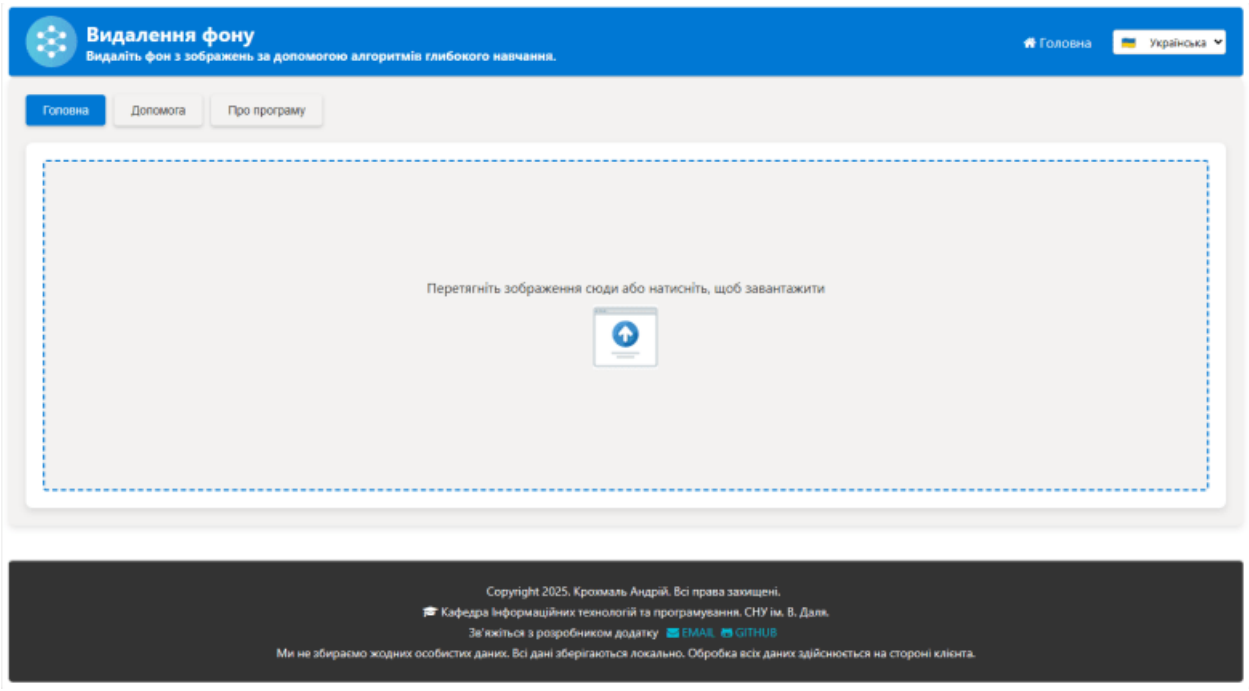


Рисунок 3.4.2 – сторінка видалення фону зображень (вигляд на ПК)

Основну частину сторінки займає контейнер з трьома вкладками: *Головна*, *Допомога* та *Про програму*. Перша вкладка містить drag-and-drop-панель для перетягування користувацького зображення. Альтернативним варіантом завантаження вхідного зображення є натискання на панель та вибій

за допомогою файлового діалогу. Щойно зображення було обрано та завантажено, панель замінюється на два холсти з кнопками керування, кнопками *Видалити фон* та *Завантажити нове зображення* (рисунок 3.4.3).

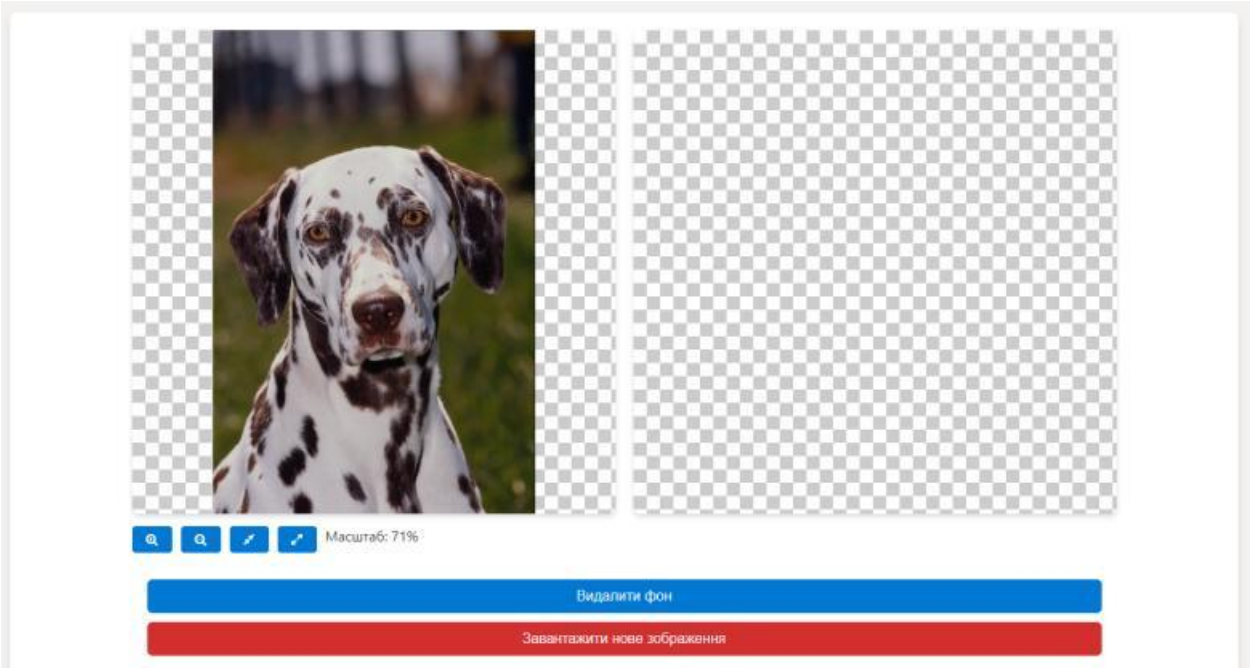


Рисунок 3.4.3 – сторінка видалення фону після завантаження зображення (вигляд на ПК)

Змінити масштаб зображення можна за допомогою кнопок керування (зліва направо: *збільшення масштабу, зменшення масштабу, вписати зображення в холст, показати оригінальний розмір*) або за допомогою колеса миші. Натисканням на кнопку *Завантажити нове зображення* можна повернутися до попереднього представлення (рисунок 3.4.2).

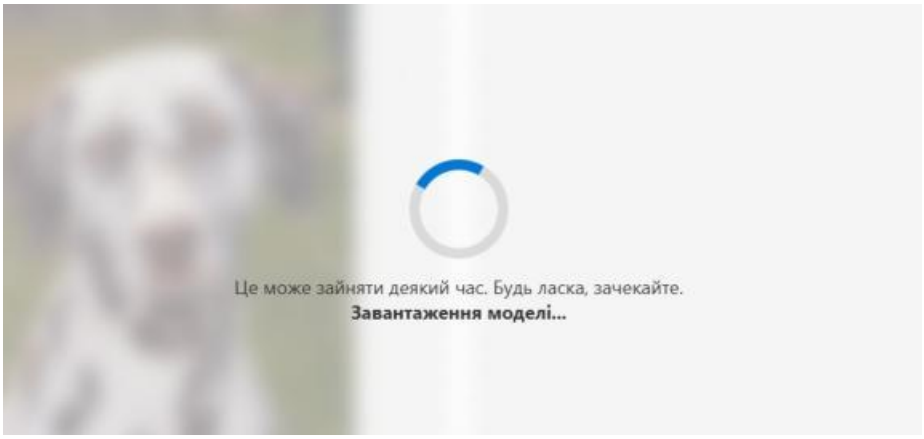


Рисунок 3.4.4 – Виконання обробки зображення

Так як обробка виконується на стороні клієнта, це може зайняти певний час залежно від обчислювальної потужності апаратної частини пристрою користувача (рисунок 3.4.4). Після завершення обробки оверлей зникає, отримане зображення відображається справа (рисунок 3.4.5). Користувачу надається можливість дослідити отриманий результат, порівняти вихідне зображення із вхідним. При наявності вихідного зображення на правому холсті, воно масштабується та панорамується синхронно з вхідним зображенням (на лівому холсті) та навпаки. Для завантаження зображення слід натиснути кнопку Завантажити результат, щоб скасувати та спробувати обробити нове – натиснути кнопку *Завантажити нове зображення.*

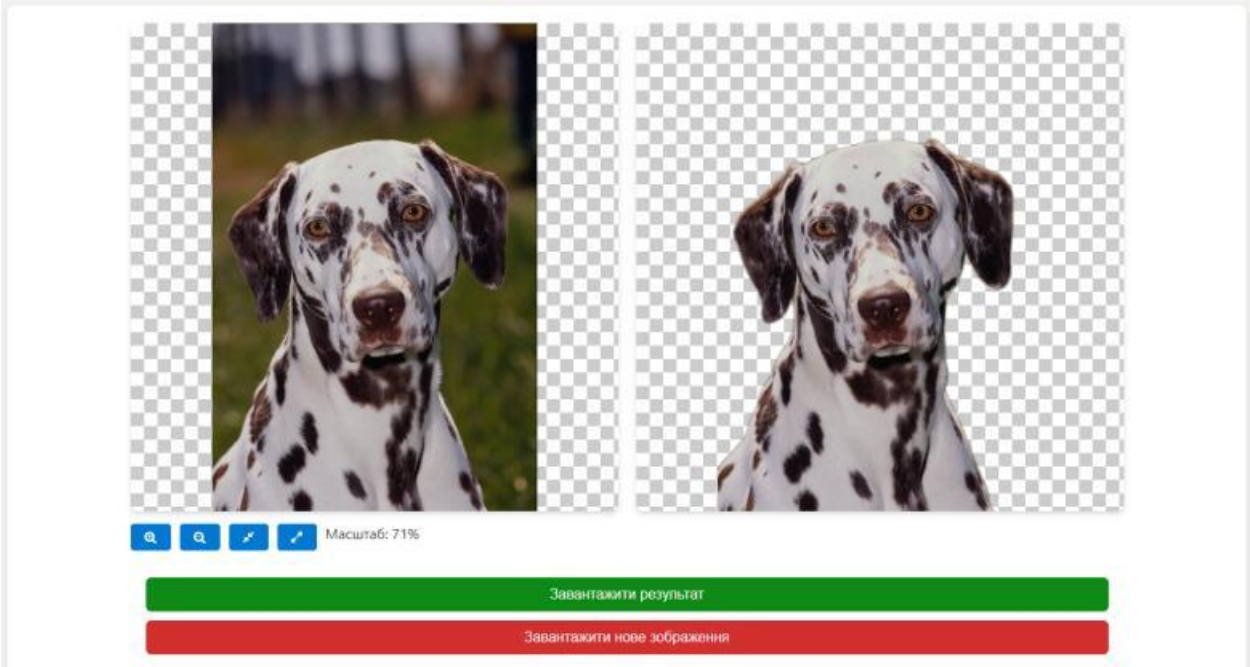


Рисунок 3.4.5 – Перегляд результатів обробки

Для того щоб отримати довідкову інформацію про те як користуватися обраним інструментом обробки або про можливості його застосування, користувач може звернутися до вкладок *Допомога* та *Про програму* на цій самій сторінці (рисунок 3.4.6). Для повернення користувача на головну сторінку застосунку слід натиснути кнопку *На головну* в шапці сторінки або натиснути на логотип зліва від заголовку. Сторінки *Створення портретів* та *Видалення людей* виглядають аналогічно до сторінки *Видалення фону*.

Відрізняються лише напис на зеленій кнопці та вміст вкладок *Допомога* та *Про програму*.

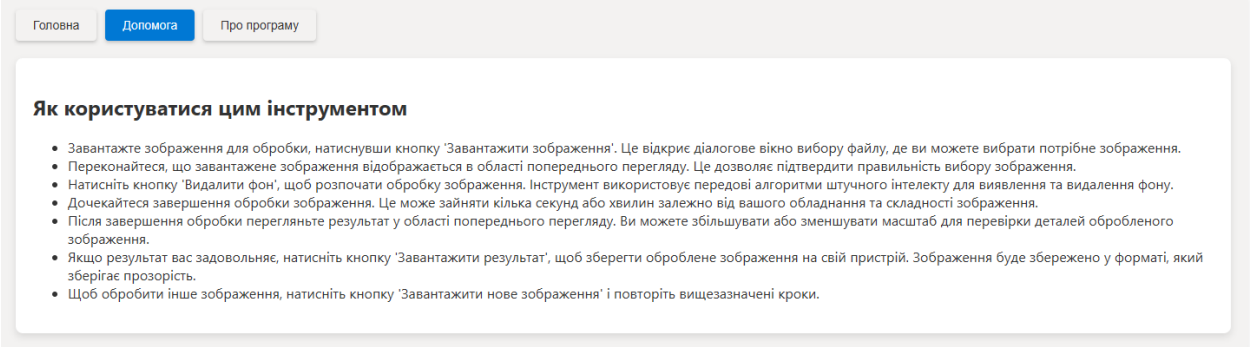


Рисунок 3.4.6 – Вкладки Допомога (зверху) та Про програму (знизу)

Сторінка *Про проект* (рисунок 3.4.7) містить відомості про технологію глибокого навчання та архітектуру ШНМ U^2-Net, що була використана у даному застосунку. На сторінці також містяться посилання, за якими можна дізнатися більше технічних деталей про використані технології.

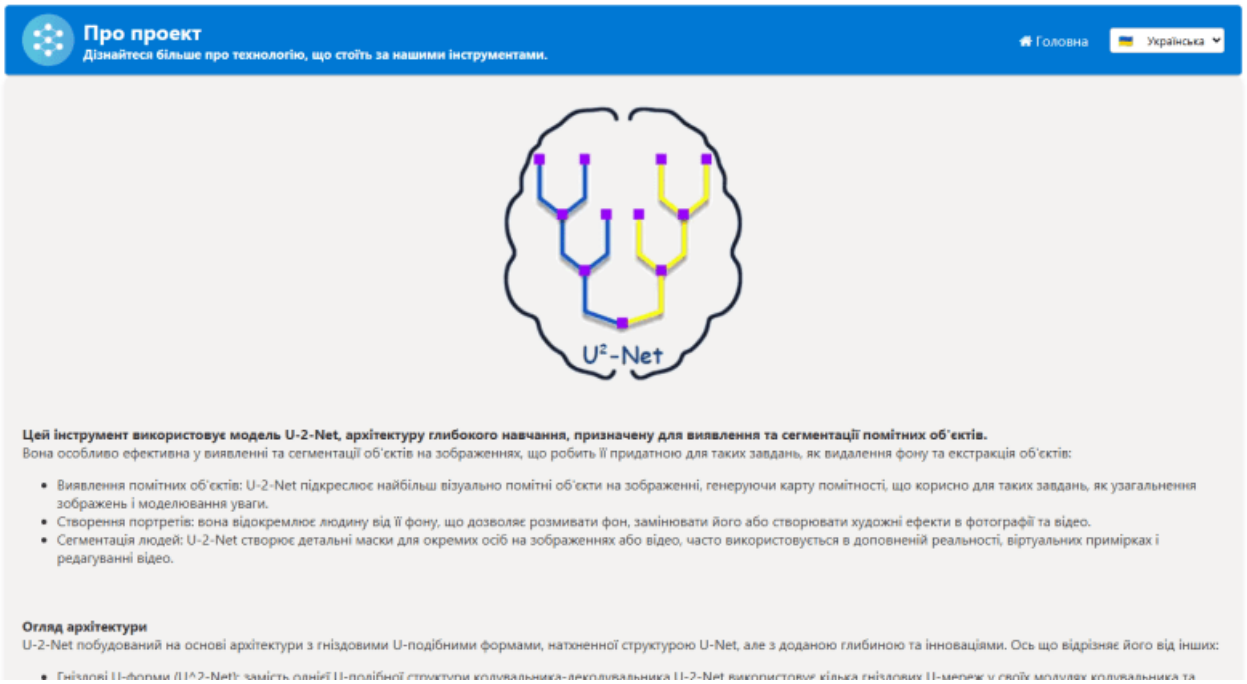
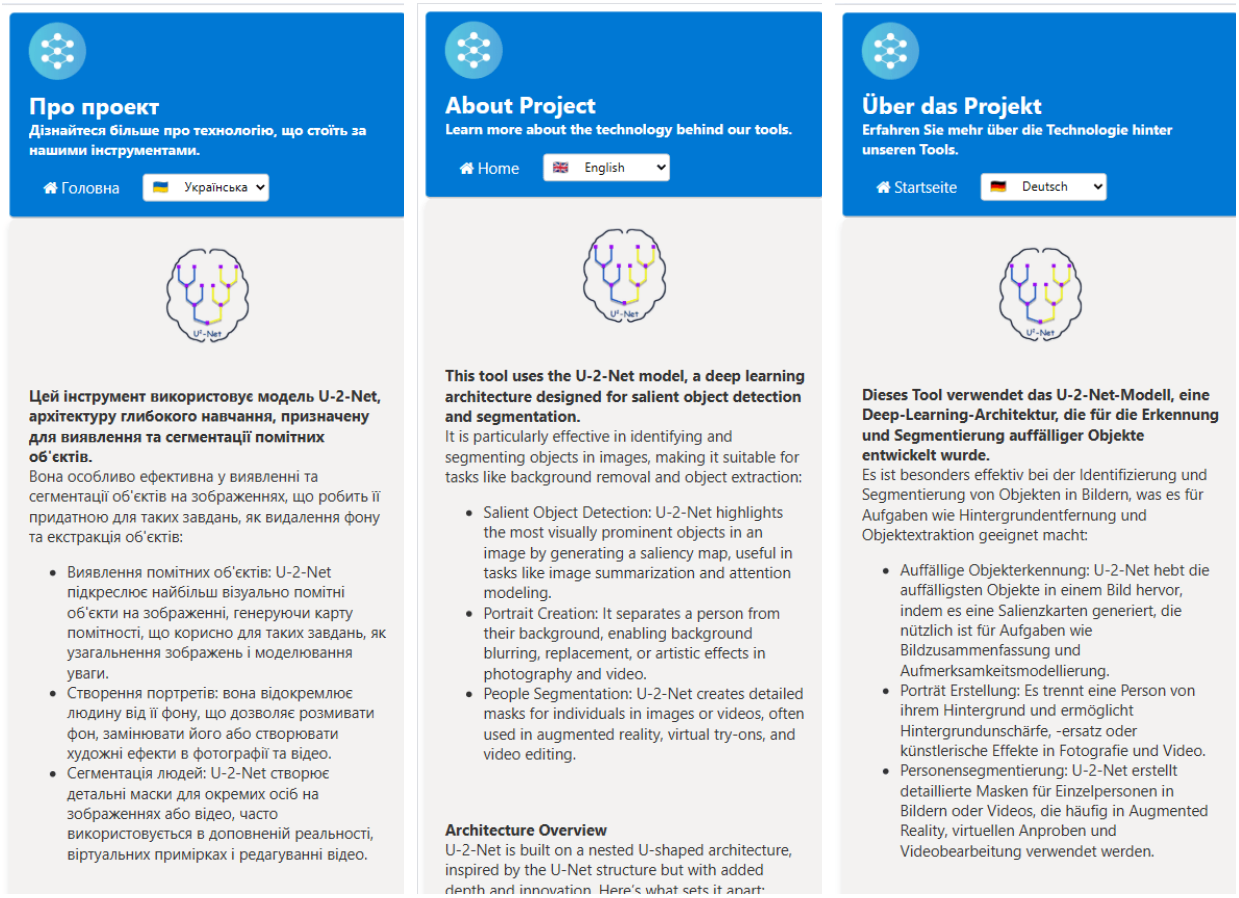


Рисунок 3.4.7 – Сторінка Про проект (вигляд на ПК)

Завдяки використанню технології i18n веб-застосунок доступний на трьох мовах – українській, англійській та німецькій. В подальшому можливе додавання перекладів на інші мови шляхом доповнення мовних варіантів строкових даних у полі i18n кожного компоненту застосунку. Зміна мови застосунку здійснюється шляхом вибору із випадаючого списку в шапці сторінки. Вигляд інтерфейсу сторінки *Про проект* на різних мовах наведено на рисунку 3.4.8.



а) б) в)

Рисунок 3.4.8 – Сторінка Про проект на різних мовах: а)українська, б) англійська, в) німецька.

Приклади вхідних та вихідних даних наведені в додатку В.

3.5 Розгортання веб-застосунку

Для забезпечення доступності розробленого веб-застосунку кінцевим користувачам було здійснено його розгортання на хмарній платформі Firebase з використанням її сервісу Firebase Hosting. Даний сервіс забезпечує швидке, безпечне та масштабоване розміщення статичних ресурсів односторінкових веб-застосунків, що повністю відповідає архітектурі створеного рішення.

Firebase Hosting було обрано з огляду на низку переваг. Насамперед, платформа надає сертифікати SSL за замовчуванням, що гарантує безпечне з'єднання через HTTPS. Іншою важливою перевагою є автоматичне кешування ресурсів на CDN (Content Delivery Network), що підвищує швидкодію та зменшує час завантаження інтерфейсу користувача. Платформа також підтримує функціональність для налаштування SPA-застосунків, дозволяючи переадресацію всіх запитів на головний HTML-файл, що є необхідним для коректної роботи vue-router.

Окрім цього, Firebase Hosting відзначається простотою розгортання, що дозволяє безпосередньо публікувати зміни або з локального середовища розробника, або шляхом автоматичної інтеграції з системами контролю версій (наприклад, GitHub). Завдяки цьому розробник отримує змогу швидко оновлювати застосунок без необхідності адміністрування серверної інфраструктури.

Розгортання застосунку на платформі Firebase може бути здійснено двома основними способами:

- *За допомогою Firebase CLI (інструмент командного рядка):*
  Ініціалізація проєкту відбувається за допомогою команди firebase init. Завершальним етапом є публікація застосунку: firebase deploy
- *Firebase Hosting підтримує CI/CD-розгортання за допомогою інтеграції з репозиторієм на GitHub:*
  У графічному інтерфейсі Firebase Console необхідно підключити GitHub-акаунт, обрати відповідний репозиторій, налаштувати гілку для розгортання та підтвердити конфігурацію. Після цього

Firebase автоматично виконує побудову та розгортання застосунку при кожному коміті у вказану гілку, що значно спрощує оновлення проєкту в реальному часі.

Обидва методи є придатними до використання залежно від потреб розробника. У межах даного дослідження було використано інструмент Firebase CLI, оскільки він забезпечує гнучкість, локальне тестування та контрольований процес публікації.

Приклад виконання команди firebase deploy в командному рядку наведено на рисунку 3.5.1.

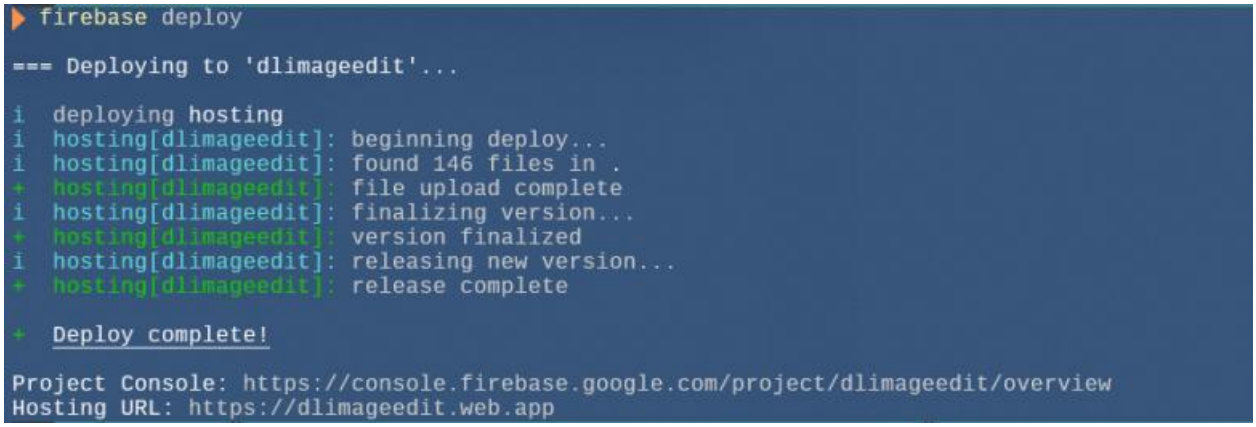


Рисунок 3.5.1 – Приклад виконання команди публікації веб-застосунку

Висновки до розділу 3

Таким чином, було спроектовано та розроблено односторінковий фронтенд-орієнтований веб-додаток, що реалізує інформаційну технологію обробки зображень методами глибокого навчання. Для забезпечення гнучкості та ефективності додатку в ньому було реалізовано такі методи та технології, як реактивність інтерфейсу, динамічна маршрутизація, багатомовність, інтерактивний HTML5-canvas, адаптивність до розміру вікна клієнта, виконання за допомогою WebAssembly, динамічне завантаження параметрів моделей ШНМ, асинхронне виконання коду за допомогою Web Worker.

Застосунок було протестовано на різних пристроях з різною роздільною здатністю дисплею (в тому числі мобільних). Він ефективно виконує поставлені задачі та повністю відповідає висунутим вимогам.

Для здійснення доступу кінцевого користувача до застосунку останній було розгорнути на хмарній хостинговій платформі Firebase, що надає сертифікати SSL, використовує CDN та підтримує CI/CD-розгортання.

ВИСНОВКИ

В ході проведеного дослідження було розроблено інформаційну технологію обробки растрових зображень засобами глибокого навчання. ЇЇ реалізовано як ефективний, безпечний веб-застосунок, що використовує інтелектуальні методи обробки зображень. Таким чином, основна мета кваліфікаційної роботи – досягнута. Всі поставлені задачі були вирішені. В результаті проведеного дослідження було отримано наступні основні результати:

1. Розглянуто найбільш поширені методи обробки зображень, що використовуються в популярних редакторах растрової графіки. Детально розглянуто принципи роботи та особливості застосування деяких з них. Окремо виділено інтелектуальні методи обробки зображень – суперроздільність, перенесення стилю, генеративне заповнення, визначені їх основні характеристики.
2. Було розглянуто основні типи веб-додатків, їх будову та характеристики. Детально проаналізовано архітектурні підходи до створення сучасних веб-застосунків, зокрема клієнт-серверну модель, багаторівневу архітектуру та модель SPA. Розглянуто основні засоби розробки веб-додатків, включаючи мови програмування та хмарні платформи для розгортання. Особливу увагу приділено аналізу сучасних фреймворків для фронтенд-розробки – React, Vue.js та Angular. Охарактеризовано їхні функціональні можливості, переваги, недоліки, а також відповідність різним сценаріям використання.
3. На основі проведеного аналізу визначено основні вимоги до розробки інтерактивного веб-застосунку для обробки растрових зображень із застосуванням методів глибокого навчання. Запропонований підхід передбачає реалізацію клієнтсько-орієнтованої архітектури у форматі SPA, що забезпечує ефективність обробки даних без необхідності взаємодії із серверною частиною.

4. Було проаналізовано та обґрунтовано вибір алгоритмів обробки зображень, що використовуються для реалізації функціоналу веб-застосунку. Для реалізації даних задач було використано моделі глибокого навчання на основі архітектури U^2-Net. Також було обґрунтовано вибір програмних засобів для створення та розгортання веб-застосунку. Зокрема, було обрано фреймворк Vue.js для побудови SPA-архітектури, vue-router для маршрутизації сторінок, vue-i18n для реалізації мультимовної підтримки інтерфейсу, а також OpenCV.js (WebAssembly-збірка бібліотеки OpenCV) для високопродуктивної обробки зображень у клієнтському середовищі.
5. Розроблено застосунок, що підтримує три ключові функції: автоматичне видалення фону зображення з генерацією альфа-каналу, стилізацію зображення для формування художніх цифрових портретів, автоматичне видалення людей з фотографій зі збереженням цілісності композиції. Розроблений веб-додаток може бути корисним для дизайнерів, користувачів соцмереж, блогерів тощо. В якості платформи для розгортання веб-застосунку було обрано Firebase Hosting, що забезпечує безперебійне розгортання SPA-застосунку з підтримкою HTTPS, CDN і автоматичного кешування.
6. Щодо можливих напрямків вдосконалення та розвитку представленої інформаційної технології, слід зазначити: створення прогресивного веб-застосунку (PWA), що дозволить зберігати веб-сторінку з усіма залежностями локально для подальшого використання без доступу до мережі; реалізація нового функціоналу через використання методів суперрозрізнення, перенесення стилю, аутпеінтингу тощо; використання технології апаратного прискорення обчислень WebGPU; оптимізація та квантування вагів ШНМ для меншого обсягу даних, що передаються.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бернерс-Лі Т., Фічетті М. Заснування Павутини : З чого починалася і до чого прийде Всесвітня мережа; [пер. з англ. А. Іщенко]. К.: Вид. дім «Києво-Могилянська академія», 2007. С. 8.
2. Інформаційні технології в бізнесі. Частина 1: Навч. посіб. / [Шевчук І.Б., Старух А.І., Васьків О.М. та ін.]; за заг. ред. І.Б. Шевчук. Львів: Видавництво ННВК «АТБ», 2020. 455 с.
3. Антоненко В. М. Сучасні Internet-технології : [навч. посіб. для студ. вищ. навч. закл.] / Антоненко В. М., Терейковський І. А., Терейковська Л. О. – Ірпінь : Нац. акад. ДПС України, 2007. – Ч. 1; Держ. податк. адмін. України, Нац. акад. держ. податк. служби України, Київ. фін.-екон. коледж НАДПС України Основи Web-дизайну, 2007. – 204 с. – Бібліогр.: с. 201.
4. Хиневич Р. В. Аналіз та систематизація видів обробки фотозображень / Р. В. Хиневич // Актуальні проблеми сучасного дизайну : збірник матеріалів Міжнародної науково-практичної конференції (20 квітня 2018 р., м. Київ) : у 2-х т. – Київ : КНУТД, 2018. – Т. 2. – С. 89-91.
5. Bradski G. The OpenCV Library // Dr. Dobb’s Journal of Software Tools – 2000 – №120. - p 122-125.
6. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – New York: Pearson, 2018. – 976 c.
7. Pizer S. M., Amburn E. P., Austin J. D. Adaptive histogram equalization and its variations // Computer Vision, Graphics, and Image Processing. – 1987. – Vol. 39, № 3. – P. 355–368. – DOI: 10.1016/S0734-189X(87)80186-X.
8. Otsu N. A Threshold Selection Method from Gray-Level Histograms // IEEE Transactions on Systems, Man, and Cybernetics. – 1979. – Vol. 9, №1. – P. 62–66. – DOI: 10.1109/TSMC.1979.4310076.
9. Захожай О. І., Крохмаль А. В. Екстенсіональний підхід до розпізнавання растрових зображень на основі їх символьних перетворень – Наукові вісті

Далівського університету. №25. 2023р. DOI: 10.33216/2222-3428- 2023-25-2.

10. Krokhmal V., Krokhmal A., Tarasov V., Rudniev Ye. Application of the Combined Method of Raster Image Recognition in the Computer Vision System of unmanned Vehicles in the Mining Industry // Geo-Technical Mechanics. 2024. № 171. DOI: https://doi.org/10.15407/geotm2024.171.098.
11. Sobel I. An Isotropic 3×3 Gradient Operator // Pattern Classification and Scene Analysis. – Stanford AI Project, 1968. – Technical Report.
12. Dong C., Loy C. C., He K., Tang X. Image Super-Resolution Using Deep Convolutional Networks // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2016. – Vol. 38, № 2. – P. 295–307. – DOI: 10.1109/TPAMI.2015.2439281.
13. Ledig C., Theis L., Huszár F. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN) // Proc. IEEE CVPR. – 2017. – P. 4681–4690. – DOI: 10.1109/CVPR.2017.19.
14. Wang X., Yu K., Dong C., Loy C. C. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks // Proc. ECCV Workshops. – 2018. – Режим доступу: https://arxiv.org/abs/1809.00219 (дата звернення: 20.10.2025).
15. Gatys L. A., Ecker A. S., Bethge M. A Neural Algorithm of Artistic Style // Nature Communications. – 2016. – Vol. 6. – P. 326 – DOI: 10.1167/16.12.326.
16. Barnes C., Shechtman E., Finkelstein A., Goldman D. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing // ACM Transactions on Graphics (SIGGRAPH). – 2009. – Vol. 28, № 3 – DOI: 10.1145/1531326.1531330.
17. Liu G., Reda F. A., Shih K. J., Wang T. C., Tao A., Catanzaro B. Image Inpainting for Irregular Holes Using Partial Convolutions // Proc. ECCV. – 2018. – P. 85–100. – DOI: 10.1007/978-3-030-01252-6_6.
18. C. H. Lin, C. -C. Chang, Y. -S. Chen, D. -C. Juan, W. Wei, H. -T. Chen. COCO-GAN: Generation by Parts via Conditional Coordinating // IEEE/CVF

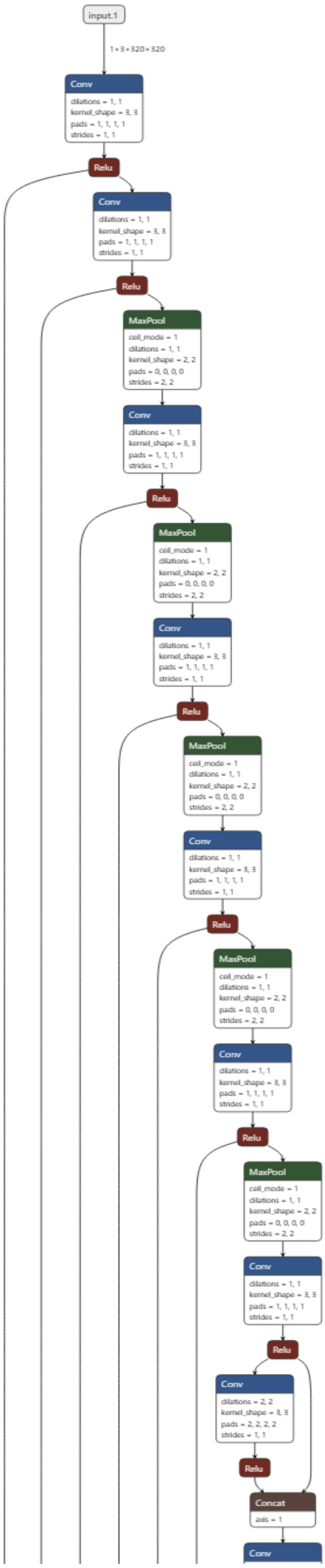
International Conference on Computer Vision (ICCV), Seoul, Korea (South). – 2019 – P. 4511-4520. – DOI: 10.1109/ICCV.2019.00461.

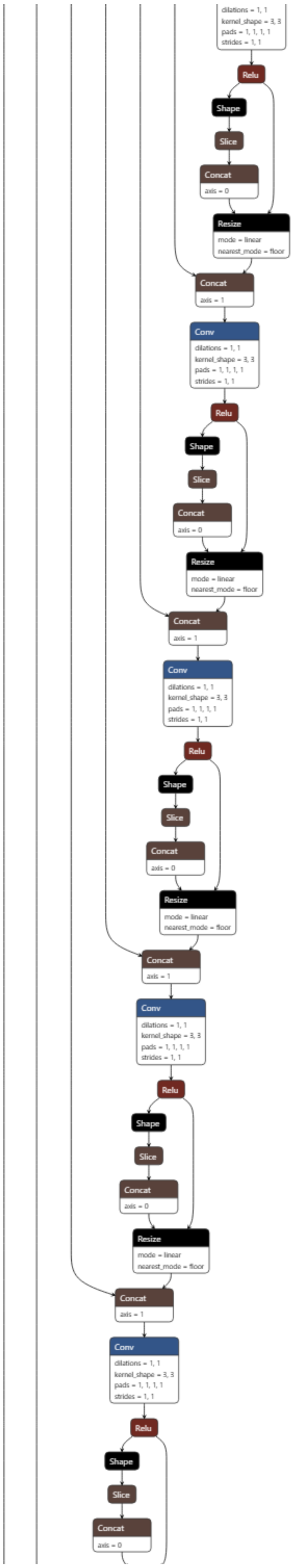
19. Iizuka S., Simo-Serra E., Ishikawa H. Globally and Locally Consistent Image Completion // ACM Transactions on Graphics (SIGGRAPH). – 2017. – Т. 36, № 4. – Стаття № 107. – DOI: 10.1145/3072959.3073659.
20. Adobe Systems Inc. Content-Aware Fill Overview. [Електронний ресурс] // Adobe Help Center. – Режим доступу: https://helpx.adobe.com/after-effects/using/content-aware-fill.html
21. Архітектура вебзастосунків 2024: Ультимативний гайд для розробників [Електронний ресурс] // Robot Dreams. – Режим доступу: https://robotdreams.cc/uk/blog/567-arhitektura-vebzastosunkiv – Дата звернення: 15.05.2025.
22. Архітектура веб-застосунків на прикладі Golang [Електронний ресурс] // DevZone. – Режим доступу: https://devzone.org.ua/post/arkhitektura-veb-zastosunkiv-na-prykladi-golang – Дата звернення: 15.05.2025.
23. Скляренко, О., Савченко, Я., Литвиненко, Л., & Сушинський, О. (2024). АРХІТЕКТУРНІ ПІДХОДИ ДО РОЗРОБКИ МАСШТАБОВАНИХ ВЕБ-ЗАСТОСУНКІВ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 4(24), 341–350. https://doi.org/10.28925/2663-4023.2024.24.341350.
24. Angular vs React vs Vue: comparing the performance [Електронний ресурс] // LogRocket Blog. – Режим доступу: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/ – Дата звернення: 15.05.2025.
25. Angular vs React vs Vue: Which framework to choose? [Електронний ресурс] // BrowserStack Guide. – Режим доступу: https://www.browserstack.com/guide/angular-vs-react-vs-vue – Дата звернення: 15.05.2025.
26. Normalization in Machine Learning [Електронний ресурс] // DataCamp – Режим доступу: https://www.datacamp.com/tutorial/normalization-in-machine-learning – (Дата звернення: 16.05.2025).

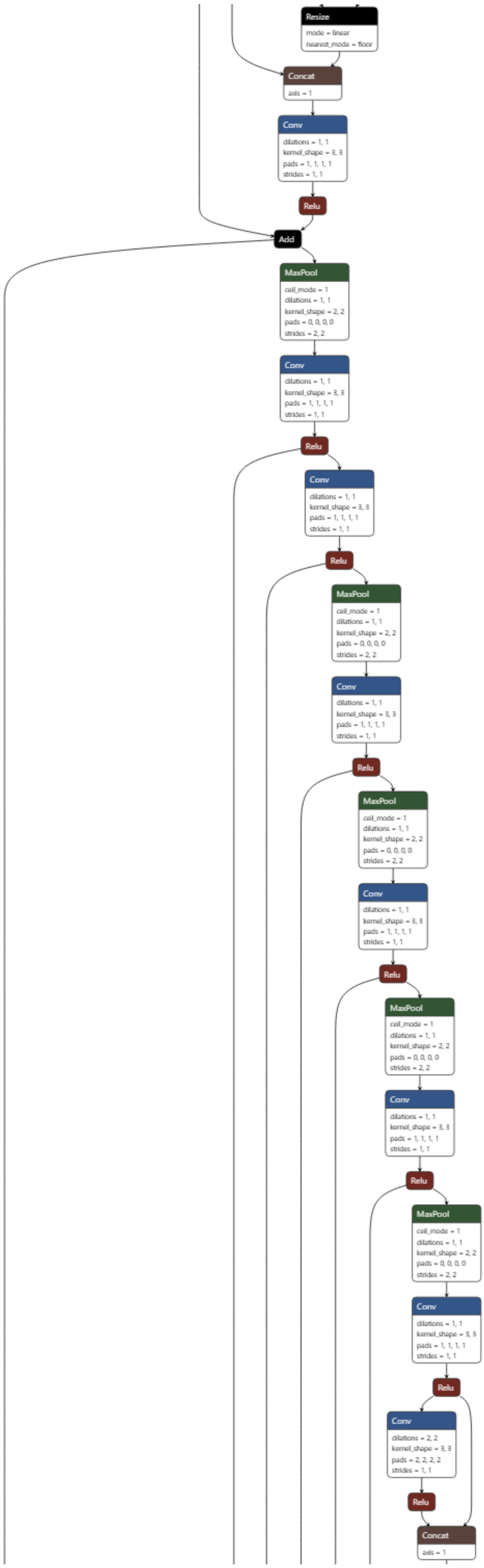
27. Gonzalez, R., Woods, R. Digital Image Processing. – Boston: Addison-Wesley Publishing Company, 1992. – p. 166.
28. Telea A. An image inpainting technique based on the fast marching method // Journal of Graphics Tools. – 2004. – Vol. 9, No. 1. – P. 23–34.
29. Sethian J.A. A Fast Marching Level Set Method for Monotonically Advancing Fronts // Proceedings of the National Academy of Sciences. – 1996. – Vol. 93, No. 4. – P. 1591–1595.
30. U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection, Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, Martin Jagersand, DOI: 10.48550/arXiv.2005.09007
31. Крохмаль А.В., Захожай О.І. Глибока нейронна мережа для перетворення зображення на малюнок трафарету для аерографії – Технологія-2024: матеріали міжн. наук.-практ. конф. 24 травня. 2024 р., м. Київ. / укладач Є. І. Зубцов – Київ : Східноукр. нац. ун-т ім. В. Даля, 2023. – 341 с.
32. Movahedi V., Elder J. H. Design and perceptual validation of performance measures for salient object segmentation // 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops. – IEEE, 2010. – P. 49–56.
33. Unified Modeling Language [Електронний ресурс] // Wikipedia – Режим доступу: https://uk.wikipedia.org/wiki/Unified_Modeling_Language – (Дата звернення: 16.05.2025).
34. Vue.js [Електронний ресурс] // Офіційний сайт – Режим доступу: https://vuejs.org/ – (Дата звернення: 16.05.2025).

ДОДАТКИ

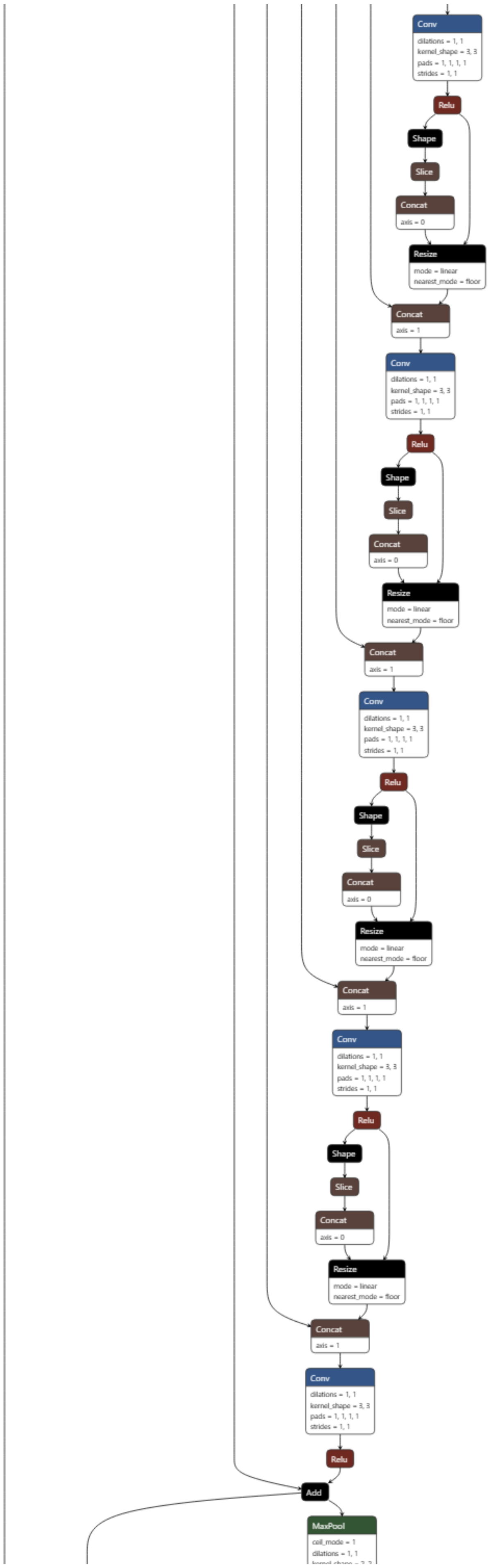
Додаток А. Архітектура ШНМ U²-Net

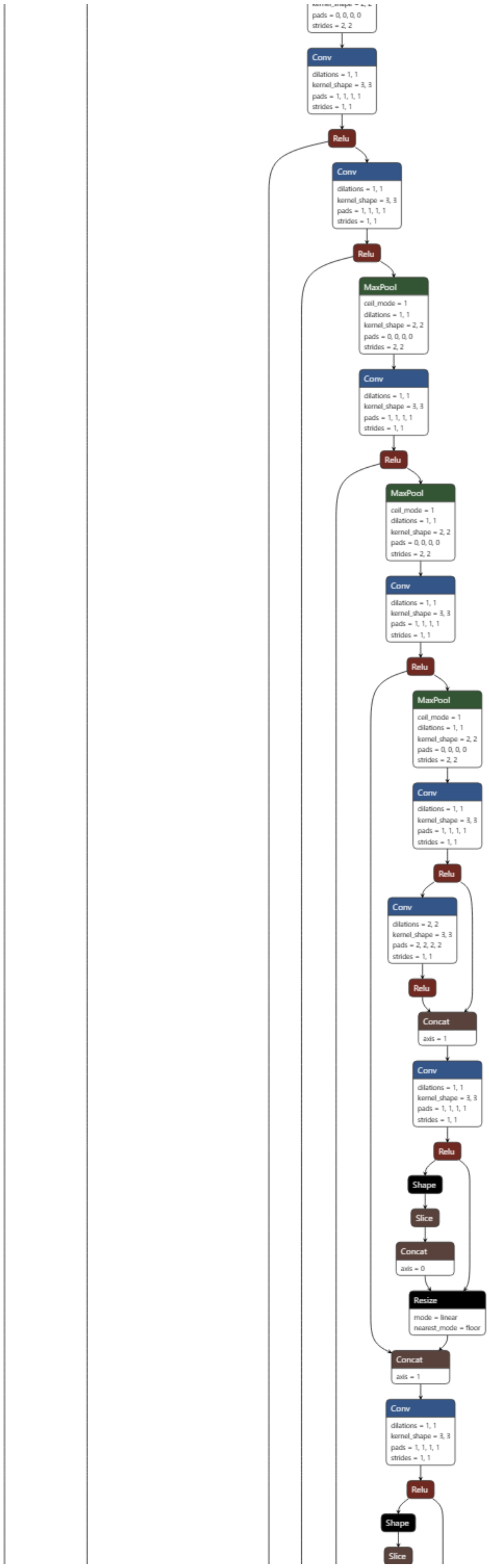


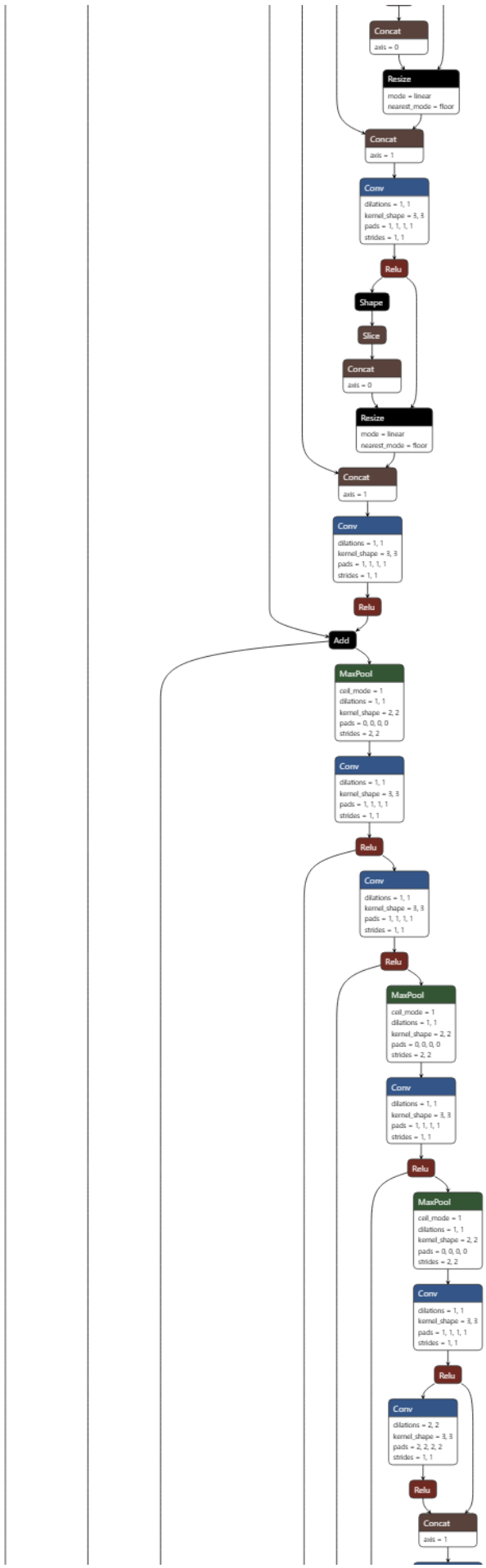

dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0

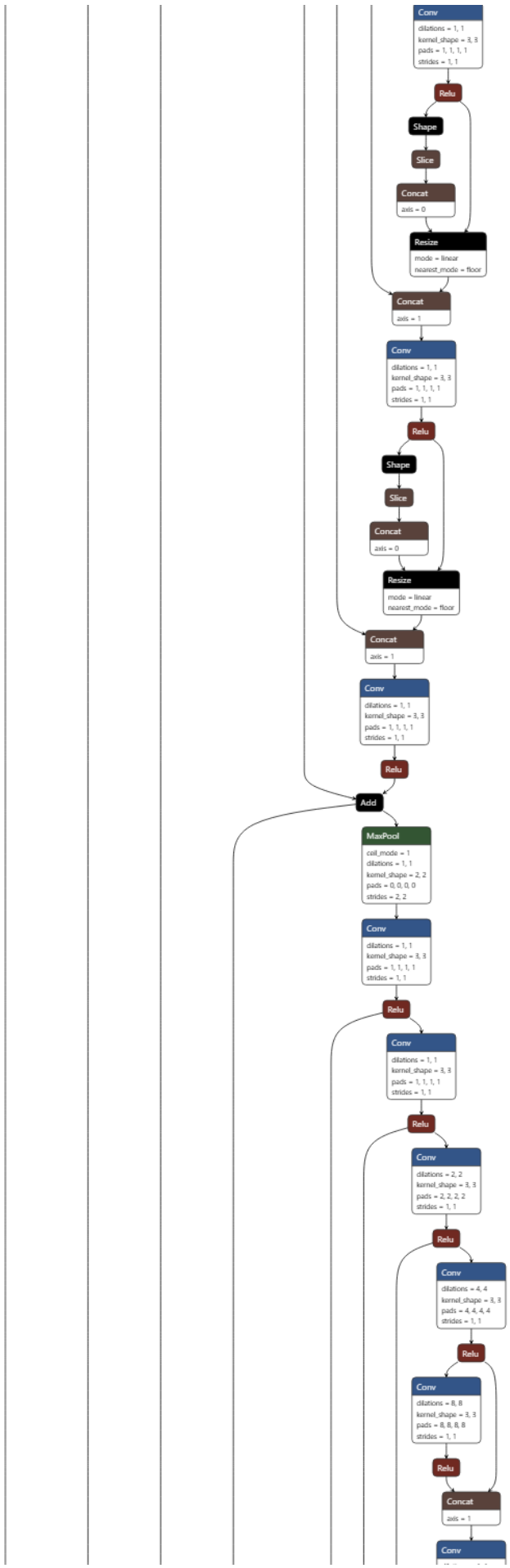


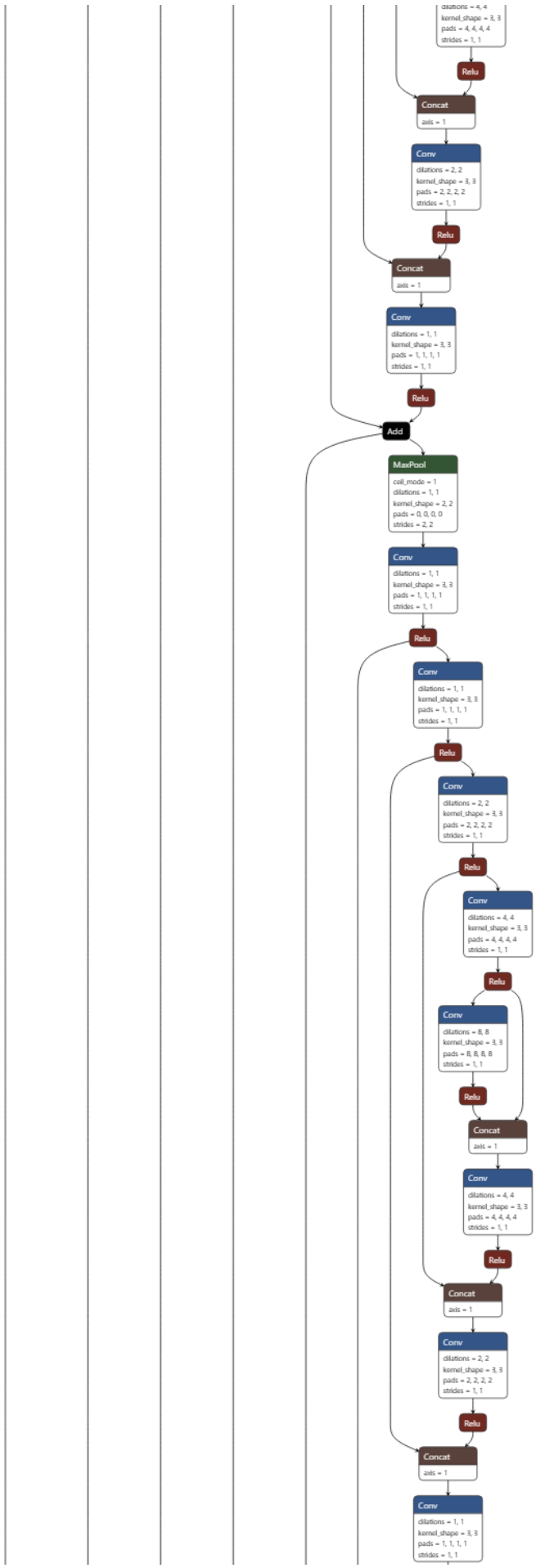
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Concat
axis = 1

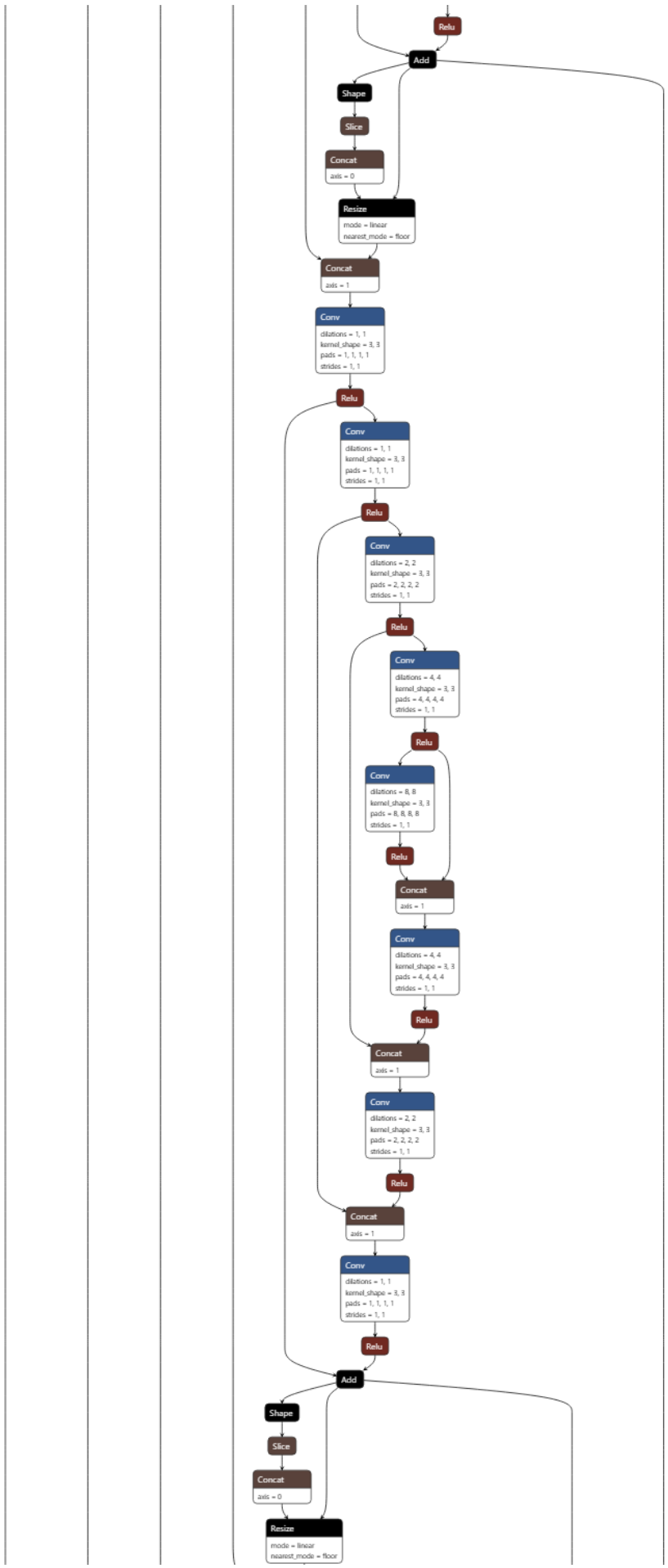

Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
MaxPool
ceil_mode = 1
dilations = 1, 1

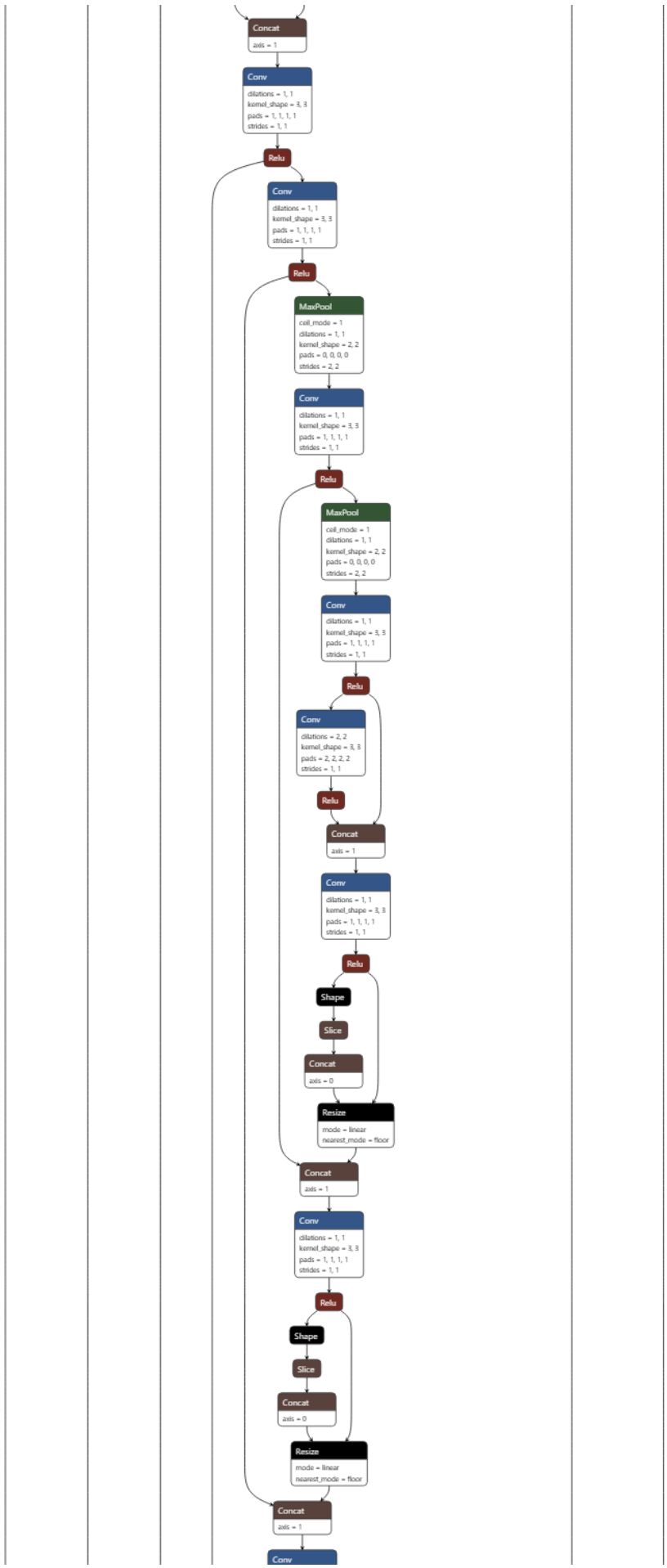

Conv
Relu
MaxPool
Concat
Shape
Slice
Resize

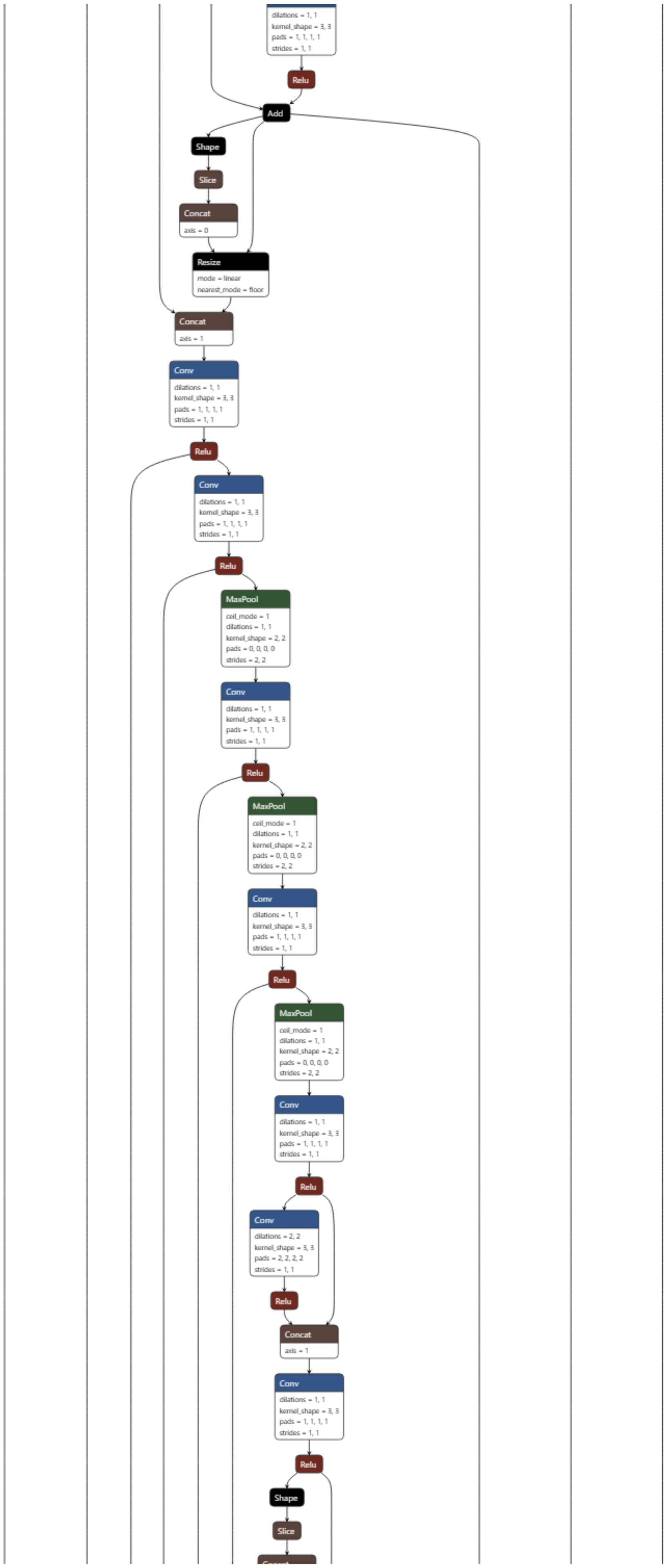

Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1

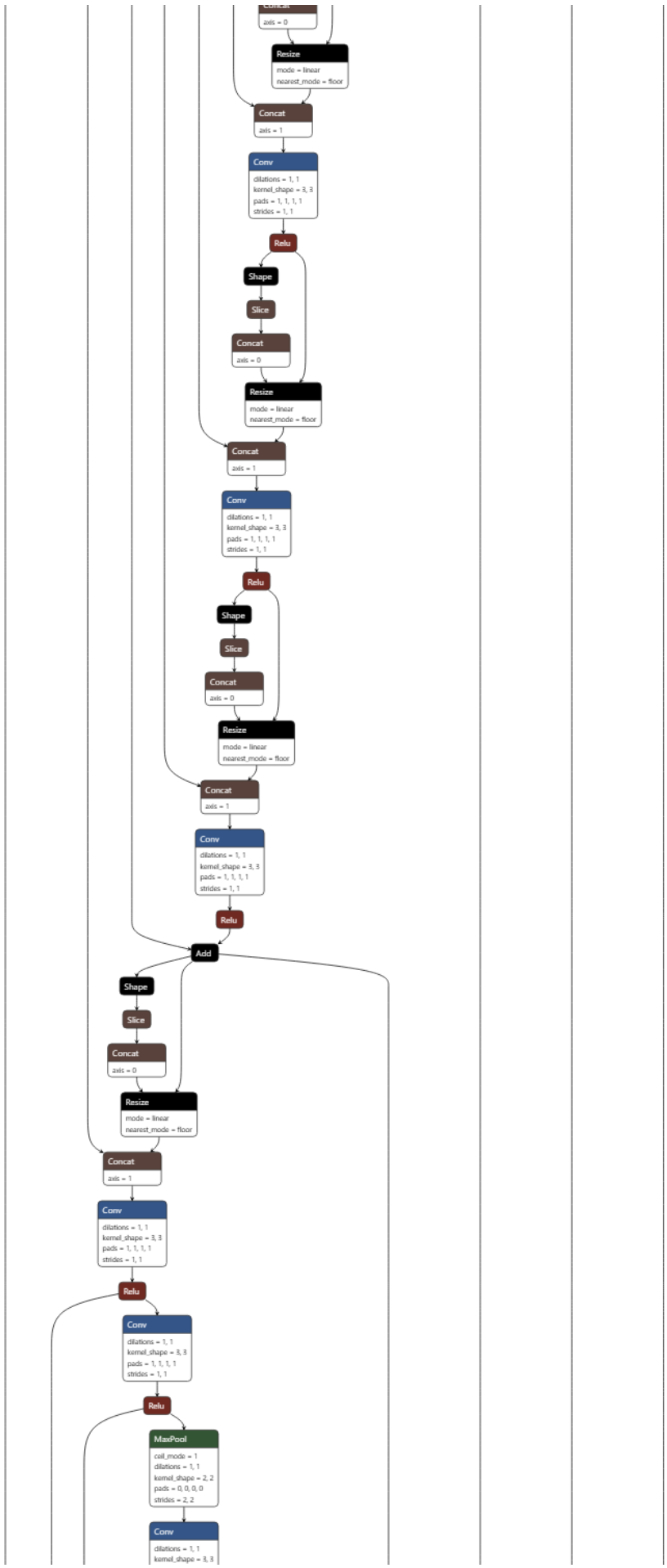

Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Conv
dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Conv
dilations = 8, 8
kernel_shape = 3, 3
pads = 8, 8, 8, 8
strides = 1, 1
Relu
Concat
axis = 1
Conv

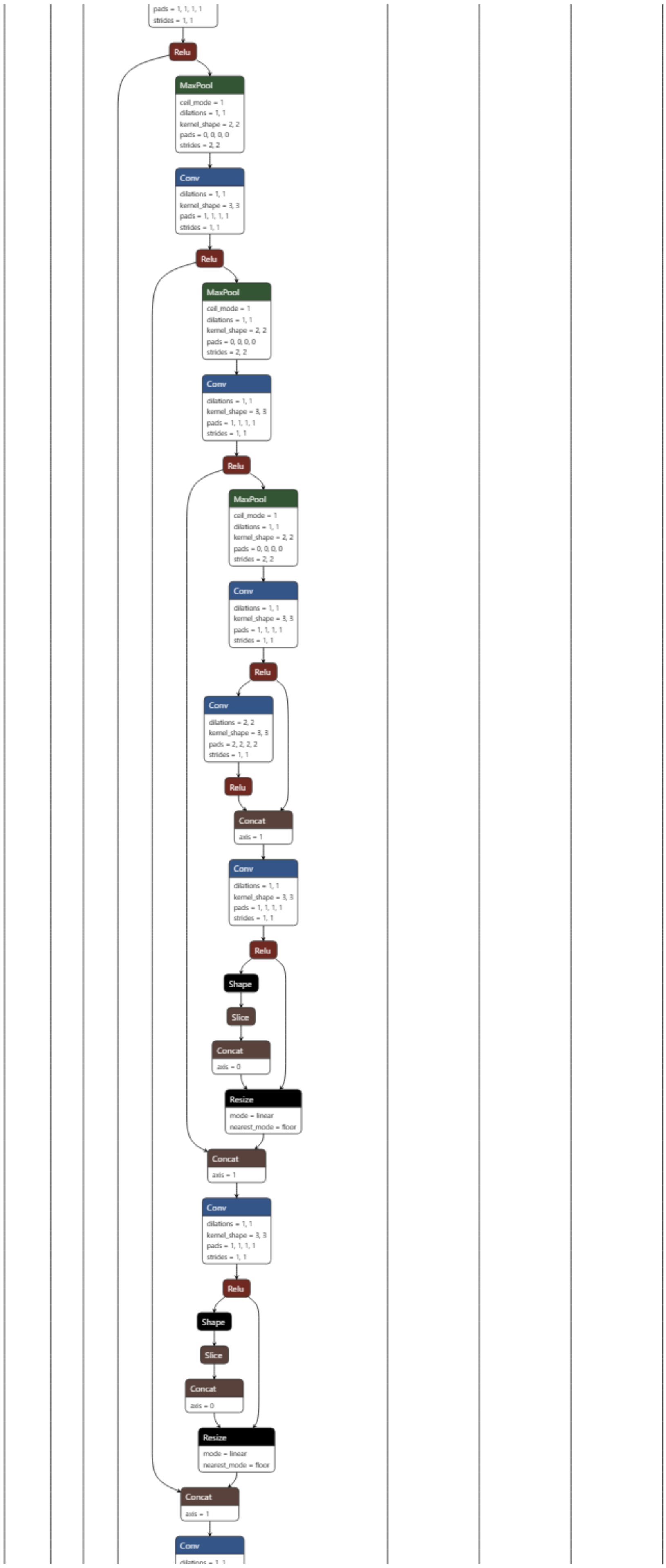

dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Conv
dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Conv
dilations = 8, 8
kernel_shape = 3, 3
pads = 8, 8, 8, 8
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1

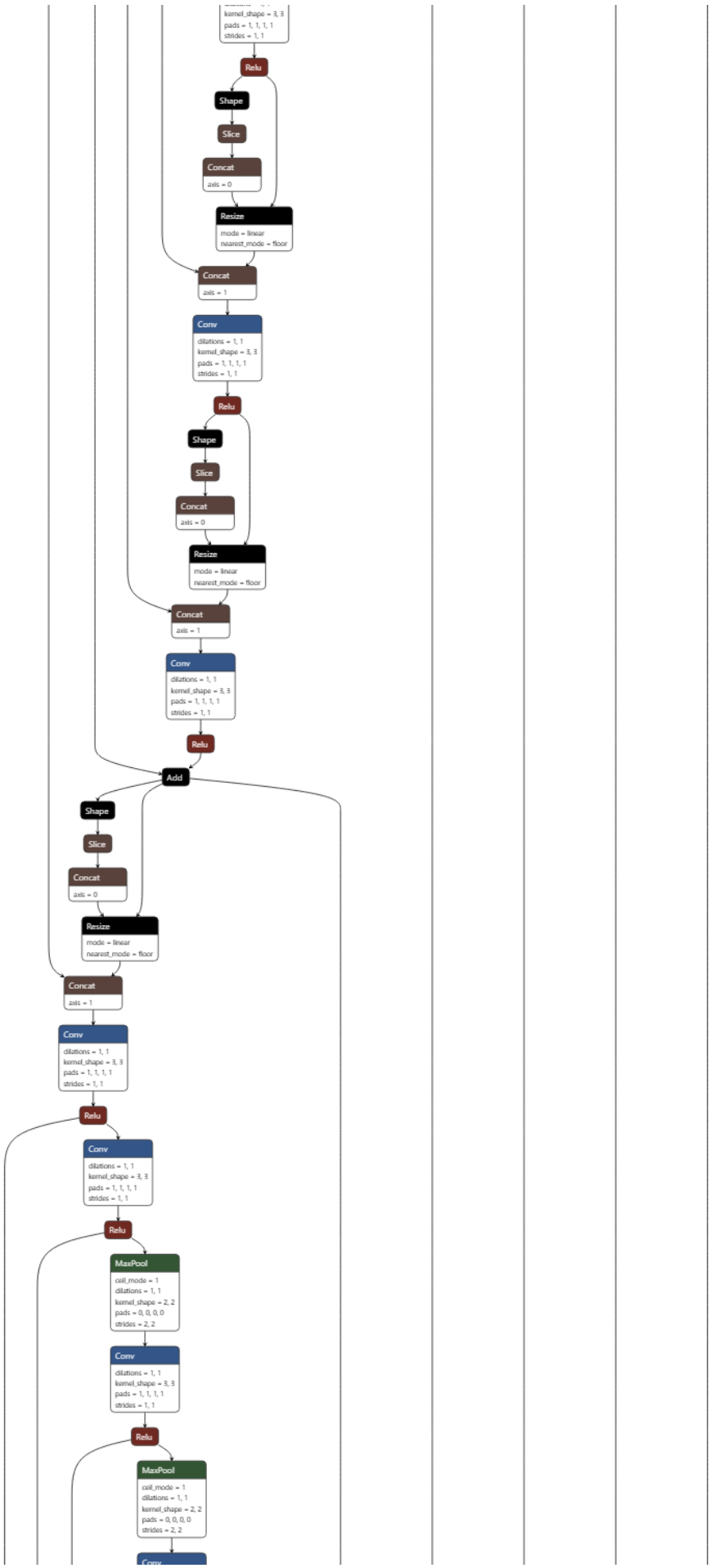

Relu
Add
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Conv
dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Conv
dilations = 8, 8
kernel_shape = 3, 3
pads = 8, 8, 8, 8
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 4, 4
kernel_shape = 3, 3
pads = 4, 4, 4, 4
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor

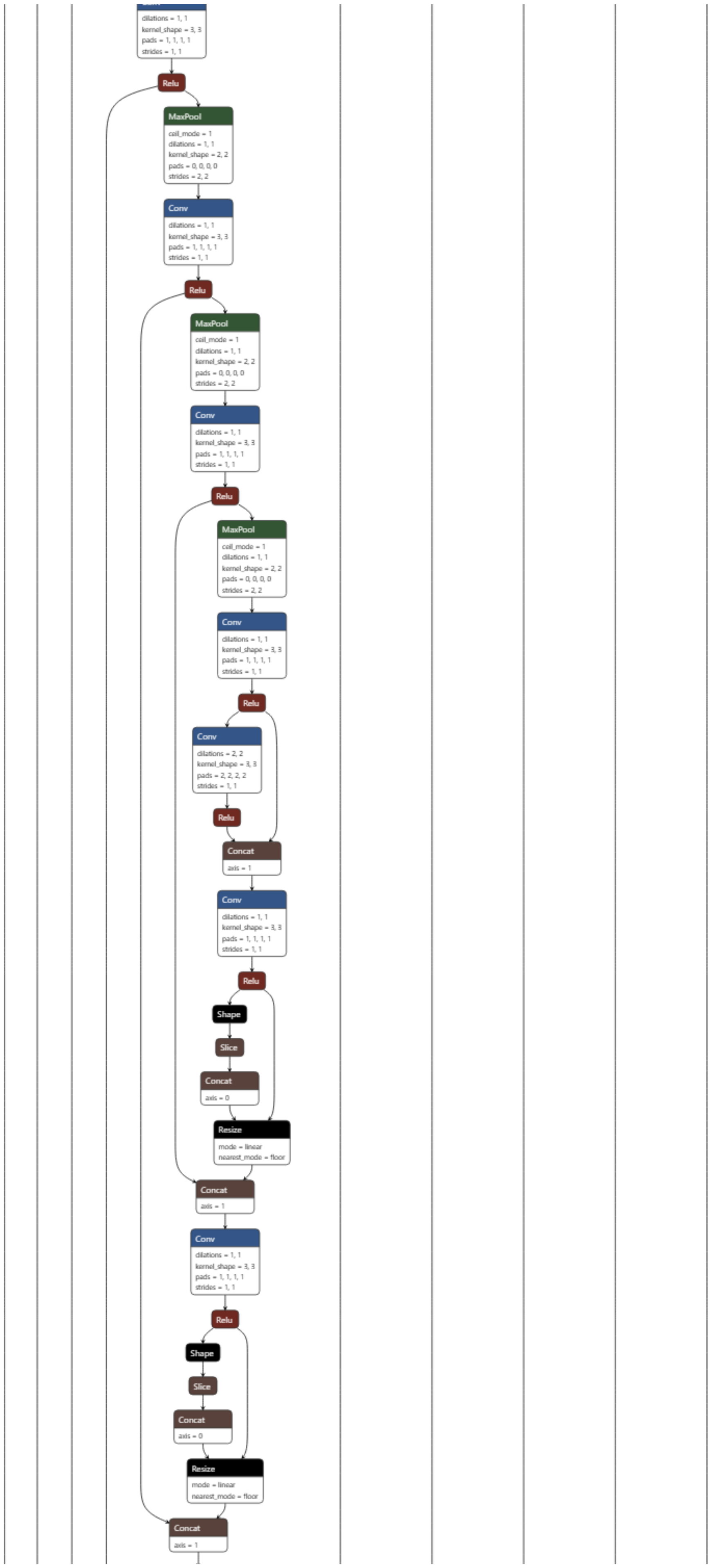

Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv

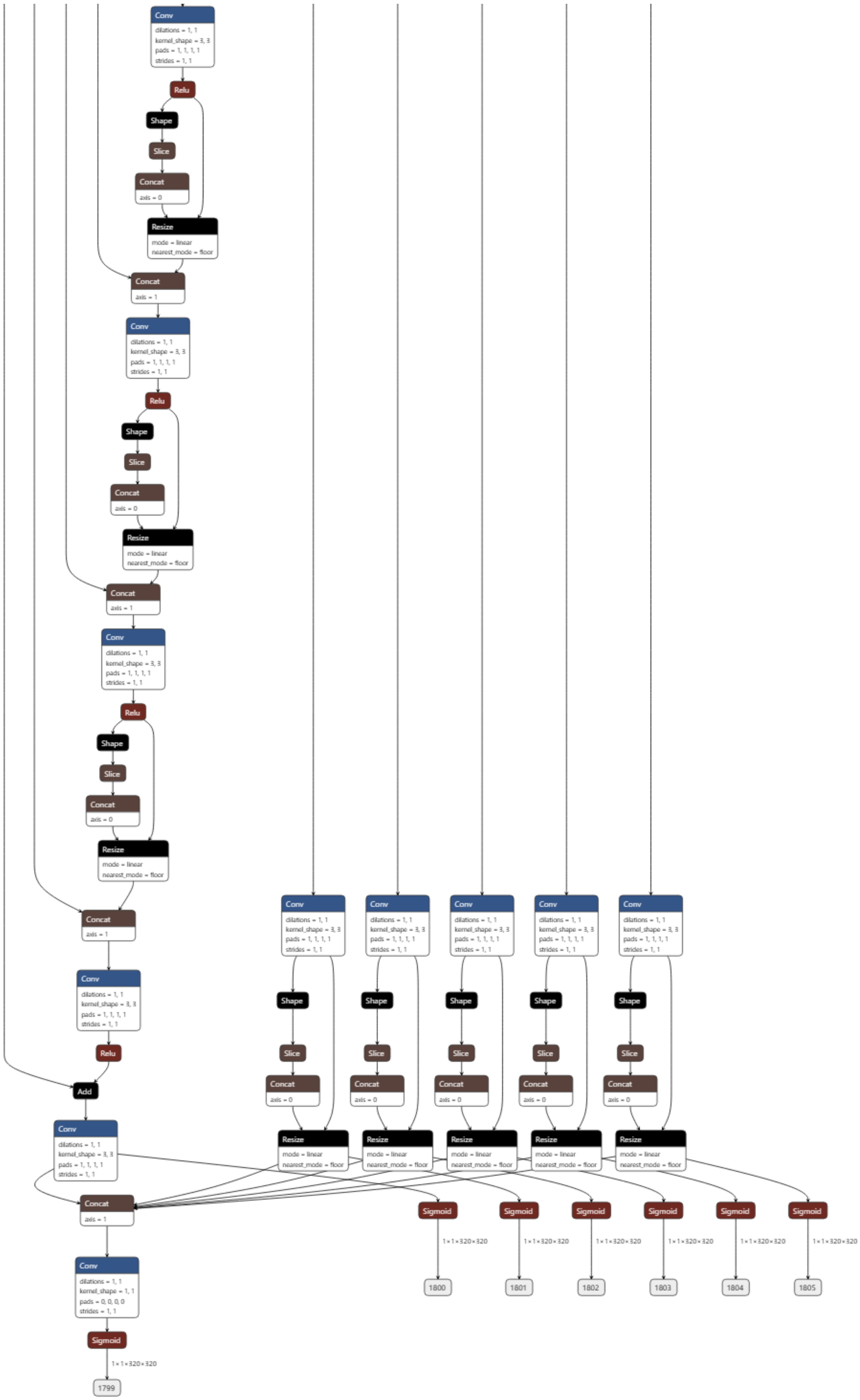

dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice


axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3


pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv


kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv


dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
MaxPool
ceil_mode = 1
dilations = 1, 1
kernel_shape = 2, 2
pads = 0, 0, 0, 0
strides = 2, 2
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Conv
dilations = 2, 2
kernel_shape = 3, 3
pads = 2, 2, 2, 2
strides = 1, 1
Relu
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1


Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Relu
Add
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Concat
axis = 1
Conv
dilations = 1, 1
kernel_shape = 1, 1
pads = 0, 0, 0, 0
strides = 1, 1
Sigmoid
1×1×320×320
1799
Conv
dilations = 1, 1
kernel_shape = 3, 3
pads = 1, 1, 1, 1
strides = 1, 1
Shape
Slice
Concat
axis = 0
Resize
mode = linear
nearest_mode = floor
Sigmoid
1×1×320×320
1800
1801
1802
1803
1804
1805

Додаток Б. Програмний код розробленого web-застосунку

Структура файлової системи директорії:

```
│   index.html
│
├──components
│   │   credentials.js
│   │   dnnWorker.js
│   │   footer.js
│   │   header.js
│   │   process.js
│   │
│   ├──UI
│   │       canvas.js
│   │       overlay.js
│   │       tabstack.js
│   │
│   └──views
│           AboutView.js
│           BGRemoveView.js
│           HomeView.js
│           PeopleRemoveView.js
│           PortraitCreateView.js
│
├──static
│       about.webp
│       bg-remove.webp
│       favicon.ico
│       logo.webp
│       people-remove.webp
│       portrait.webp
│       u2net-architecture.webp
│       u2net-logo.webp
│       u2net.onnx
│       u2net_human_seg.onnx
│       u2net_portrait.onnx
│       upload.webp
│
└──style
        canvas.css
        footer.css
        header.css
        home.css
        main.css
        overlay.css
        process.css
        tabstack.css
```

Файл *'.\index.html'*:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Background Removal Tool">
    <meta name="author" content="Andrii Krokhmal">
    <meta name="keywords" content="background removal, portrait
creation, people removal, image processing, AI, machine
learning, deep learning, U-2-Net">

    <title></title>
    <link rel="icon" href="/static/favicon.ico" type="image/x-
icon">
</head>

<body>
    <script type="importmap">
            { "imports": {
                "vue":
"https://cdnjs.cloudflare.com/ajax/libs/vue/3.2.41/vue.esm-
browser.prod.js",
                "@vue/devtools-api":
"https://unpkg.com/@vue/devtools-api@6.5.0/lib/esm/index.js",
                "vue-router":
"https://cdnjs.cloudflare.com/ajax/libs/vue-router/4.1.5/vue-
router.esm-browser.min.js",
                "vue-i18n":          "https://unpkg.com/vue-
i18n@9.2.0/dist/vue-i18n.esm-browser.prod.js"
            }
        }
    </script>

    <script type="module">
        import { createApp } from 'vue';
        import { createI18n } from 'vue-i18n';
        import { createRouter, createWebHistory, RouterView,
RouterLink } from 'vue-router';

        import CustomHeader from '/components/header.js';
        import CustomFooter from '/components/footer.js';

        const messages = {
            en: {
                home_page: {
                    title: 'Deep Learning Image Processing
Tools',
                    description: 'A collection of tools for
image processing using deep learning techniques.',
                },
                background_remove_page: {
                    title: 'Background Removal',
```

```
                    description: 'Remove the background from
images using deep learning algorithms.',
                },
                portrait_create_page: {
                    title: 'Portrait Creation',
                    description: 'Create stunning portraits
using advanced image processing techniques.',
                },
                people_remove_page: {
                    title: 'People Removal',
                    description: 'Remove people from images
seamlessly using AI technology.',
                },
                about_page: {
                    title: 'About Project',
                    description: 'Learn more about the
technology behind our tools.',
                },
            },
            uk: {
                home_page: {
                    title: 'Інструменти обробки зображень на
основі глибокого навчання',
                    description: 'Збірка інструментів для
обробки зображень за допомогою технологій глибокого навчання.',
                },
                background_remove_page: {
                    title: 'Видалення фону',
                    description: 'Видаліть фон з зображень за
допомогою алгоритмів глибокого навчання.',
                },
                portrait_create_page: {
                    title: 'Створення портретів',
                    description: 'Створіть вражаючі портрети за
допомогою передових технологій обробки зображень.',
                },
                people_remove_page: {
                    title: 'Видалення людей',
                    description: 'Безшовно видаліть людей із
зображень за допомогою технології штучного інтелекту.',
                },
                about_page: {
                    title: 'Про проект',
                    description: 'Дізнайтеся більше про
технологію, що стоїть за нашими інструментами.',
                },
            },
            de: {
                home_page: {
                    title: 'Deep Learning
Bildverarbeitungstools',
```

```
                    description: 'Eine Sammlung von Tools zur
Bildverarbeitung mit Deep-Learning-Techniken.',
                },
                background_remove_page: {
                    title: 'Hintergrundentfernung',
                    description: 'Entfernen Sie den Hintergrund
von Bildern mit Deep-Learning-Algorithmen.',
                },
                portrait_create_page: {
                    title: 'Porträt Erstellung',
                    description: 'Erstellen Sie atemberaubende
Porträts mit fortschrittlichen Bildverarbeitungstechniken.',
                },
                people_remove_page: {
                    title: 'Menschen entfernen',
                    description: 'Entfernen Sie Menschen nahtlos
aus Bildern mit KI-Technologie.',
                },
                about_page: {
                    title: 'Über das Projekt',
                    description: 'Erfahren Sie mehr über die
Technologie hinter unseren Tools.',
                },
            },
        };

        const i18n = createI18n({
            fallbackLocale: 'en',
            messages: messages,
        });

        const routes = [
            {
                name: 'home',
                path: '/',
                alias: '/home',

                component: () =>
import('/components/views/HomeView.js')
            },
            {
                name: 'background_remove',
                path: '/background_remove',
                component: () =>
import('/components/views/BgRemoveView.js')
            },
            {
                name: 'portrait_create',
                path: '/portrait_create',
                component: () =>
import('/components/views/PortraitCreateView.js')
            },
```

```
            {
                name: 'people_remove',
                path: '/people_remove',
                component: () =>
import('/components/views/PeopleRemoveView.js')
            },
            {
                name: 'about',
                path: '/about',
                component: () =>
import('/components/views/AboutView.js')
            },
        ];

        const router = createRouter({
            history: createWebHistory(),
            routes,
        });

        const app = createApp({
            components: { RouterView, RouterLink, CustomHeader,
CustomFooter },
        });
        app.use(i18n);
        app.use(router);
        app.mount('#app');
    </script>

    <div id="app">
        <link rel="stylesheet" href="/style/main.css">
        <custom-header></custom-header>
        <router-view></router-view>
        <div style="flex: 1"></div>
        <custom-footer></custom-footer>
    </div>

</body>

</html>
```

Файл *'.\components\credentials.js'*:

```
const credentials = {
    links: {
        email: "mailto:krohmal.a@snu.edu.ua",
        github: "https://github.com/Voinic"
    }
};

export default credentials;
```

Файл *'.\components\dnnWorker.js'*:

```
// dnnWorker.js

self.importScripts('https://docs.opencv.org/4.10.0/opencv.js');
// Import OpenCV.js

let cvReady = false;
cv['onRuntimeInitialized'] = () => {
    console.log('OpenCV.js is ready.');
    cvReady = true;
};

const waitForCV = () => {
    return new Promise((resolve) => {
        const check = () => {
            if (cvReady) {
                resolve();
            } else {
                setTimeout(check, 10); // Check every 10ms
            }
        };
        check();
    });
};

const loadModelFromURL = async (path, name) => {
    try {
        const response = await fetch(path);
        const buffer = await response.arrayBuffer();
        const data = new Uint8Array(buffer);
        cv.FS_createDataFile('/', name, data, true, false,
false);
        return name;
    } catch (error) {
        console.error('Error loading model:', error);
        return null;
    }
};

const findMinMax = (data, size) => {
    let minVal = Number.MAX_VALUE;
    let maxVal = Number.MIN_VALUE;
    for (let i = 0; i < size; ++i) {
        let curVal = data[i];
        if (curVal > maxVal) {
            maxVal = curVal;
        }
        if (curVal < minVal) {
            minVal = curVal;
        }
    }
    return { minVal, maxVal };
};
```

```
self.onmessage = async (e) => {
    self.postMessage({ progress: "prepare" });

    await waitForCV(); // Wait until OpenCV is initialized

    self.postMessage({ progress: "loadModel" });

    let model = null;
    let maxDim = null;
    if (e.data.mode == "salient_object") {
        model = "u2net.onnx";
        maxDim = 320;
    } else if (e.data.mode == "portrait") {
        model = "u2net_portrait.onnx";
        maxDim = 512;
    } else if (e.data.mode == "human") {
        model = "u2net_human_seg.onnx";
        maxDim = 320;
    }
    const modelPath = await loadModelFromURL("/static/" + model,
model)

    try {
        const src = cv.matFromImageData(e.data.image);

        if (!modelPath) {
            self.postMessage({ error: "Error while loading
model." });
            return;
        }

        const net = cv.readNet(modelPath);

        self.postMessage({ progress: "preprocess" });

        const mean = [0.485, 0.456, 0.406];
        const std = [0.229, 0.224, 0.225];
        let numCh = 3;

        let scale = Math.min(maxDim / src.cols, maxDim /
src.rows);
        let newW = Math.round(src.cols * scale);
        let newH = Math.round(src.rows * scale);

        let matC3 = new cv.Mat(src.rows, src.cols, cv.CV_8UC3);
        cv.cvtColor(src, matC3, cv.COLOR_RGBA2RGB);

        let resizedMat = new cv.Mat(newH, newW, cv.CV_8UC3);
        cv.resize(matC3, resizedMat, new cv.Size(newW, newH), 0,
0, cv.INTER_LINEAR);
```

```
        let borderedMat = new cv.Mat(maxDim, maxDim,
cv.CV_8UC3);
        cv.copyMakeBorder(resizedMat, borderedMat, 0, maxDim -
newH, 0, maxDim - newW,
            cv.BORDER_CONSTANT, new cv.Scalar(255, 255, 255));
        resizedMat.delete();

        let minVal, maxVal;
        ({ minVal, maxVal } = findMinMax(borderedMat.data,
maxDim * maxDim * numCh));

        let normData = [];
        for (let h = 0; h < maxDim; ++h) {
            for (let w = 0; w < maxDim; ++w) {
                for (let j = 0; j < numCh; ++j) {
                    let normalizedValue =
(borderedMat.ucharAt(h, w * numCh + j) / maxVal - mean[j]) /
std[j];
                    normData.push(normalizedValue);
                }
            }
        }
        let normMat = new cv.matFromArray(maxDim, maxDim,
cv.CV_32FC3, normData);

        let blob = cv.blobFromImage(normMat, 1, new
cv.Size(maxDim, maxDim), new cv.Scalar(0, 0, 0), true);
        normMat.delete();

        self.postMessage({ progress: "inference" });

        net.setInput(blob, "input.1");
        let netOutput = net.forward("1799");
        blob.delete();

        self.postMessage({ progress: "postprocessing" });

        ({ minVal, maxVal } = findMinMax(netOutput.data32F,
maxDim * maxDim));
        console.log("Min value: " + minVal);
        console.log("Max value: " + maxVal);

        let invNormData = [];
        for (let h = 0; h < maxDim; ++h) {
            for (let w = 0; w < maxDim; ++w) {
                let normalizedValue = (netOutput.data32F[h *
maxDim + w] - minVal) / (maxVal - minVal);
                invNormData.push(normalizedValue * 255);
            }
        }
        let maskMat = new cv.matFromArray(maxDim, maxDim,
cv.CV_8UC1, invNormData);
```

```
        let cropRect = new cv.Rect(0, 0, newW, newH);
        let croppedMaskMat = maskMat.roi(cropRect);
        let resizedMaskMat = new cv.Mat(src.rows, src.cols,
cv.CV_8UC1);
        cv.resize(croppedMaskMat, resizedMaskMat, new
cv.Size(src.cols, src.rows), 0, 0, cv.INTER_LINEAR);
        maskMat.delete();
        croppedMaskMat.delete()

        let dst = new cv.Mat(src.rows, src.cols, cv.CV_8UC4);

        if (e.data.mode == "salient_object") {
            let maskChannels = new cv.MatVector();
            cv.split(src, maskChannels);
            maskChannels.set(3, resizedMaskMat);
            cv.merge(maskChannels, dst);
            maskChannels.delete();
        }
        else if (e.data.mode == "portrait") {
            cv.bitwise_not(resizedMaskMat, resizedMaskMat);
            let portraitMat = new cv.Mat();
            cv.cvtColor(resizedMaskMat, portraitMat,
cv.COLOR_GRAY2RGB);
            let blurredMat = new cv.Mat();
            cv.GaussianBlur(matC3, blurredMat, new cv.Size(15,
15), 2);
            let blendedMat = new cv.Mat();
            cv.addWeighted(blurredMat, 0.35, portraitMat, 0.65,
0, blendedMat);
            blurredMat.delete();
            portraitMat.delete();
            cv.cvtColor(blendedMat, dst, cv.COLOR_RGB2RGBA);
            blendedMat.delete();
        }
        else if (e.data.mode == "human") {
            let inpaintedMat = new cv.Mat();
            cv.inpaint(matC3, resizedMaskMat, inpaintedMat, 25,
cv.INPAINT_TELEA);
            cv.cvtColor(inpaintedMat, dst, cv.COLOR_RGB2RGBA);
        }

        src.delete();
        matC3.delete();
        resizedMaskMat.delete();

        // Convert result back to ImageData
        const resultImageData = new ImageData(
            new Uint8ClampedArray(dst.data),
            dst.cols,
            dst.rows
        );
```

```
        dst.delete();

        // Send result back to main thread
        self.postMessage({ progress: "done" });
        self.postMessage({ imageData: resultImageData });
    } catch (error) {
        self.postMessage({ error: error.message });
    }
};
```

Файл *'.\components\footer.js'*:

```
import credentials from '/components/credentials.js';

export default {
    template: `
        <link rel="stylesheet" type="text/css"
href="/style/footer.css"/>
        <footer>
            <p>&copy; Copyright 2025. {{ $t('footer.copyright')
}}</p>
            <p>&#xF19D; {{ $t('footer.text') }}</p>
            <p>{{ $t('footer.contact') }}  
            <a v-if="credentials.links.facebook"
:href="credentials.links.facebook">&#xF230; FACEBOOK &
nbsp;</a>
            <a v-if="credentials.links.instagram"
:href="credentials.links.instagram">&#xF16D; INSTAGRAM 
; </a>
            <a v-if="credentials.links.telegram"
:href="credentials.links.telegram">&#xF1D8; TELEGRAM &
nbsp;</a>
            <a v-if="credentials.links.linkedin"
:href="credentials.links.linkedin">&#xF0E1; LINKEDIN &
nbsp;</a>
            <a v-if="credentials.links.twitter"
:href="credentials.links.twitter">&#xF099; TWITTER &nb
sp;</a>
            <a v-if="credentials.links.whatsapp"
:href="credentials.links.whatsapp">&#xF232; WHATSAPP &
nbsp;</a>
            <a v-if="credentials.links.email"
:href="credentials.links.email">&#xF0E0; EMAIL  <
/a>
            <a v-if="credentials.links.github"
:href="credentials.links.github">&#xF113; GITHUB  
;</a>
            </p>
            <p>{{ $t('footer.privacy') }}</p>
        </footer>
    `,
    name: 'custom-footer',
```

```
    data() {
        return {
            credentials: credentials,
        };
    },
    i18n: {
        messages: {
            en: {
                footer: {
                    copyright: "Andrii Krohmal. All rights
reserved.",
                    text: "Department of Information Technology
and Programming. VDEUNU.",
                    contact: "Contact app developer",
                    privacy: "We do not collect any personal
data. All data is stored locally. Processing of all data is done
on the client side.",
                },
            },
            uk: {
                footer: {
                    copyright: "Крохмаль Андрій. Всі права
захищені.",
                    text: "Кафедра Інформаційних технологій та
програмування. СНУ ім. В. Даля.",
                    contact: "Зв'яжіться з розробником додатку",
                    privacy: "Ми не збираємо жодних особистих
даних. Всі дані зберігаються локально. Обробка всіх даних
здійснюється на стороні клієнта.",
                },
            },
            de: {
                footer: {
                    copyright: "Andrii Krohmal. Alle Rechte
vorbehalten.",
                    text: "Abteilung für Informationstechnologie
und Programmierung. VDEUNU.",
                    contact: "Kontaktieren Sie den App-
Entwickler",
                    privacy: "Wir sammeln keine persönlichen
Daten. Alle Daten werden lokal gespeichert. Die Verarbeitung
aller Daten erfolgt auf der Client-Seite.",
                },
            },
        }
    }
};
```

Файл *'.\components\header.js'*:

```
export default {
    template: `
```

```
        <div class="header">
            <link rel="stylesheet" type="text/css"
href="/style/header.css"/>
            <img src="/static/logo.webp" alt="Logo" class="logo"
@click="goHome"/>
            <div class="title-wrapper">
            <h1 class="title">{{ currentPage.title }}</h1>
            <h3 class="subtitle">{{ currentPage.description
}}</h3>
            </div>
            <div class="selector">
                <router-link to="/" v-if="this.$route.name !=
'home'" class="home-link">&#xF015; {{ $t('header.home')
}}</router-link>
                <select id="language-select" v-
model="selectedLanguage" @change="changeLanguage">
                    <option v-for="lang in languages"
:key="lang.code" :value="lang.code">
                        {{ lang.flag }} {{ lang.name }}
                    </option>
                </select>
            </div>
        </div>
    `,
    name: 'custom-header',
    data() {
        return {
            selectedLanguage: 'uk',
            languages: [
                { code: 'uk', name: 'Українська', flag: 'UA' },
                { code: 'en', name: 'English', flag: 'GB' },
                { code: 'de', name: 'Deutsch', flag: 'DE' }
            ],
        };
    },
    methods: {
        changeLanguage() {
            this.$root.$i18n.locale = this.selectedLanguage;
        },
        goHome() {
            this.$router.push('/');
        }
    },
    mounted() {
        this.$root.$i18n.locale = this.selectedLanguage;
        document.title = this.currentPage.title;
    },
    watch: {
        '$i18n.locale': function (newLang) {
            console.log('Language changed to:', newLang);
            this.selectedLanguage = newLang;
            document.title = this.currentPage.title;
```

```
        },
    },
    computed: {
        currentPage() {
            return {
                title: this.$route.name ?
this.$t(`${this.$route.name}_page.title`) :
this.$t('home_page.title'),
                description: this.$route.name ?
this.$t(`${this.$route.name}_page.description`) :
this.$t('home_page.description')
            };
        }
    },
    i18n: {
        messages: {
            en: {
                header: {
                    home: 'Home',
                },
            },
            uk: {
                header: {
                    home: 'Головна',
                },
            },
            de: {
                header: {
                    home: 'Startseite',
                },
            },
        }
    },
};
```

Файл *'.\components\process.js'*:

```
import CustomCanvas from "/components/UI/canvas.js"
import Overlay from "/components/UI/overlay.js"

export default {
  template: `
    <div>
  <link rel="stylesheet" type="text/css"
href="/style/process.css"/>

  <!-- Drop area -->
  <div
  class="drop-area"
  @dragover.prevent
  @dragenter.prevent
  @drop="handleDrop"
  @click="triggerFileInput"
```

```
  v-if="!inputFile"
  >
  <div class="drop-area-content">
    <Text variant="medium">{{ $t('home.dropAreaText')
}}</Text><br/>
    <img
    style="height: 75px; padding-top: 10px;"
    src="/static/upload.webp"
    alt="Upload Icon"
    class="upload-icon"
    />
  </div>
  <input
    type="file"
    ref="fileInput"
    @change="handleFileSelect"
    accept="image/*"
    hidden
  />
  </div>

  <!-- Canvas display -->
  <div v-show="inputFile">
  <div class="canvas-wrapper">
    <CustomCanvas ref="canvasInput"/>
    <CustomCanvas ref="canvasOutput"/>
  </div>

  <!-- Buttons -->
  <div class="buttons-wrapper">
    <button
    v-if="outputFile"
    class="download-btn"
    @click="downloadResult"
    >
    {{ $t('home.downloadResult') }}
    </button>
    <button
    v-else
    class="start-btn"
    @click="startProcessing"
    >
    {{ this.getActionText() }}
    </button>
    <button
    v-if="inputFile"
    class="cancel-btn"
    @click="resetApp"
    >
    {{ $t('home.uploadNewImage') }}
    </button>
  </div>
```

```
  </div>

  <Overlay ref="overlay" />
</div>
  `,
  name: "Process",
  components: {
    CustomCanvas,
    Overlay
  },
  props: {
    mode: {
      type: String,
      default: "salient_object"
    }
  },
  data() {
    return {
      inputFile: null,
      outputFile: null
    }
  },
  methods: {
    triggerFileInput() {
      this.$refs.fileInput.click();
    },
    async handleDrop(event) {
      event.preventDefault();
      const file = event.dataTransfer.files[0];
      if (file) {
        await this.loadFile(file);
      }
    },
    async handleFileSelect(event) {
      const file = event.target.files[0];
      if (file) {
        await this.loadFile(file);
      }
    },
    async loadFile(file) {
      let imgUrl = URL.createObjectURL(file);
      this.inputFile = (await
this.$refs.canvasInput.loadImage(imgUrl)).src;
    },
    async startProcessing() {
      let canvas = this.$refs.canvasOutput;
      let overlay = this.$refs.overlay;
      overlay.text = this.$t("home.overlay.prepare");
      overlay.show();
      let imgData = this.$refs.canvasInput.imagedata;
      const worker = new Worker("/components/dnnWorker.js");
      worker.postMessage({image: imgData, mode: this.mode});
```

```
      worker.onmessage = async (e) => {
        if (e.data) {
          if (e.data.imageData) {
            this.outputFile = (await
canvas.loadImageData(e.data.imageData)).src;
            overlay.hide();
          } else if (e.data.progress) {
            console.log(e.data.progress);
            overlay.text =
this.$t("home.overlay."+e.data.progress);
          } else if (e.data.error) {
            console.error("Error from worker:", e.data.error);
            overlay.hide();
            alert(this.$t('home.processingError'));
          }
        }
      };
      worker.onerror = (error) => {
        console.error("Worker error:", error);
        overlay.hide();
        alert(this.$t('home.processingError'));
      };
    },
    downloadResult() {
      const link = document.createElement("a");
      link.href = this.outputFile;
      link.download = "output.png";
      link.click();
    },
    resetApp() {
      this.$refs.canvasInput.resetCanvas();
      this.$refs.canvasOutput.resetCanvas();
      this.inputFile = null;
      this.outputFile = null;
    },
    getActionText() {
      let action = "removeBackground";
      if (this.mode === "portrait") {
        action = "createPortrait";
      }
      else if (this.mode === "human") {
        action = "removePeople";
      }
      return this.$t(`home.action.${action}`);
    },
    warnUnsavedChanges(event) {
      if (this.inputFile) {
        event.preventDefault();
        event.returnValue = "";
      }
    },
  },
```

```
  mounted() {
    this.$refs.canvasInput.link(this.$refs.canvasOutput);
  },
  created() {
    window.addEventListener("beforeunload",
this.warnUnsavedChanges);
  },
  beforeDestroy() {
    window.removeEventListener("beforeunload",
this.warnUnsavedChanges);
  },
  i18n: {
    messages: {
      en: {
        home: {
          dropAreaText: "Drag and drop an image here or click to
upload",
          action: {
            removeBackground: "Remove Background",
            createPortrait: "Create Portrait",
            removePeople: "Remove People"
          },
          overlay: {
            prepare: 'Preparing...',
            loadModel: 'Loading model...',
            preprocessing: 'Preprocessing image...',
            inference: 'Running inference...',
            postprocessing: 'Postprocessing result...',
            done: 'Done!',
            error: 'Error!',
          },
          downloadResult: "Download Result",
          uploadNewImage: "Upload New Image",
          processingError: "Error processing the image. Please
try again."
        },
      },
      uk: {
        home: {
          dropAreaText: "Перетягніть зображення сюди або
натисніть, щоб завантажити",
          action: {
            removeBackground: "Видалити фон",
            createPortrait: "Створити портрет",
            removePeople: "Видалити людей"
          },
          overlay: {
            prepare: 'Підготовка...',
            loadModel: 'Завантаження моделі...',
            preprocessing: 'Попередня обробка зображення...',
            inference: 'Виконання виведення...',
            postprocessing: 'Постобробка результату...',
```

```
            done: 'Готово!',
            error: 'Помилка!',
          },
          downloadResult: "Завантажити результат",
          uploadNewImage: "Завантажити нове зображення",
          processingError: "Помилка обробки зображення. Будь
ласка, спробуйте ще раз."
        },
      },
      de: {
        home: {
          dropAreaText: "Ziehen Sie ein Bild hierher oder
klicken Sie, um es hochzuladen",
          action: {
            removeBackground: "Hintergrund entfernen",
            createPortrait: "Porträt erstellen",
            removePeople: "Menschen entfernen"
          },
          overlay: {
            prepare: 'Vorbereitung...',
            loadModel: 'Modell wird geladen...',
            preprocessing: 'Bild wird vorverarbeitet...',
            inference: 'Inference wird durchgeführt...',
            postprocessing: 'Ergebnis wird nachbearbeitet...',
            done: 'Fertig!',
            error: 'Fehler!',
          },
          downloadResult: "Ergebnis herunterladen",
          uploadNewImage: "Neues Bild hochladen",
          processingError: "Fehler bei der Verarbeitung des
Bildes. Bitte versuchen Sie es erneut."
        },
      },
    }
  }
};
```

Файл *'.\components\UI\canvas.js'*:

```
export default {
    name: 'CustomCanvas',
    data() {
        return {
            linkedCanvas: null,
            isLinked: false,
            observer: null,
            img: new Image(),
            imagedata: null,
            scale: 1,
            originX: 0,
            originY: 0,
            isPanning: false,
            startX: 0,
```

```
            startY: 0,
        };
    },
    methods: {
        async loadImage(url) {
            return new Promise((resolve, reject) => {
                this.img.crossOrigin = 'anonymous';
                this.img.onload = () => {
                    const canvas = this.$refs.canvasRef;
                    const ctx = canvas.getContext("2d");
                    ctx.drawImage(this.img, 0, 0);
                    this.resetZoom();

                    const canvas2 =
document.createElement("canvas");
                    const ctx2 = canvas2.getContext("2d");
                    canvas2.width = this.img.width;
                    canvas2.height = this.img.height;
                    ctx2.drawImage(this.img, 0, 0);
                    this.imagedata = ctx2.getImageData(0, 0,
this.img.width, this.img.height);

                    resolve(this.img);
                };
                this.img.onerror = (error) => {
                    reject(error);
                };
                this.img.src = url;
            });
        },
        async loadImageData(imagedata) {
            return new Promise((resolve, reject) => {
                this.imagedata = imagedata;

                const canvas2 =
document.createElement("canvas");
                const ctx2 = canvas2.getContext("2d");
                canvas2.width = this.imagedata.width;
                canvas2.height = this.imagedata.height;
                ctx2.putImageData(this.imagedata, 0, 0);

                this.img.onload = () => {
                    const canvas = this.$refs.canvasRef;
                    const ctx = canvas.getContext("2d");
                    ctx.drawImage(this.img, 0, 0);
                    this.resetZoom();
                    resolve(this.img);
                };
                this.img.onerror = (error) => {
                    reject(error);
                };
                this.img.src = canvas2.toDataURL();
```

```
            });
        },
        resetCanvas() {
            this.img = new Image();
            this.imagedata = null;
            this.redrawCanvas();
        },
        resetZoom(scale = null) {
            let canvas = this.$refs.canvasRef;
            if (scale) {
                this.scale = scale;
            } else {
                const scaleX = canvas.width / this.img.width;
                const scaleY = canvas.height / this.img.height;
                this.scale = Math.min(scaleX, scaleY);
            }
            this.originX = (canvas.width / 2) - (this.img.width
/ 2) * this.scale;
            this.originY = (canvas.height / 2) -
(this.img.height / 2) * this.scale;
            this.redrawCanvas();
        },
        clearCanvas() {
            let canvas = this.$refs.canvasRef;
            let ctx = canvas.getContext("2d");
            ctx.save();
            ctx.setTransform(1, 0, 0, 1, 0, 0);
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            const squareSize = 15;
            for (let i = 0; i < canvas.width; i += squareSize) {
                for (let j = 0; j < canvas.height; j +=
squareSize) {
                    ctx.fillStyle = (i / squareSize + j /
squareSize) % 2 === 0 ? '#ffffff' : '#cccccc';
                    ctx.fillRect(i, j, squareSize, squareSize);
                }
            }
            ctx.restore();
        },
        zoom(direction, centerX = null, centerY = null) {
            const canvas = this.$refs.canvasRef;
            const ctx = canvas.getContext("2d");

            const zoomFactor = 0.1;
            const zoom = direction > 0 ? 1 + zoomFactor : 1 -
zoomFactor;

            ctx.translate(this.originX, this.originY);
            ctx.scale(zoom, zoom);
            ctx.translate(-this.originX, -this.originY);

            if (!centerX || !centerY) {
```

```
                centerX = canvas.width / 2;
                centerY = canvas.height / 2;
            }

            this.scale *= zoom;
            this.originX = centerX - (centerX - this.originX) *
zoom;
            this.originY = centerY - (centerY - this.originY) *
zoom;

            this.redrawCanvas();
        },
        handleWheel(event) {
            event.preventDefault();
            this.zoom(event.deltaY < 0, event.offsetX,
event.offsetY)
        },
        handleMouseDown(event) {
            this.isPanning = true;
            this.startX = event.clientX - this.originX;
            this.startY = event.clientY - this.originY;
        },
        handleMouseMove(event) {
            if (!this.isPanning) return;
            this.originX = event.clientX - this.startX;
            this.originY = event.clientY - this.startY;
            this.redrawCanvas();
        },
        handleMouseUp() {
            this.isPanning = false;
        },
        handleTouchStart(event) {
            if (event.touches.length === 1) {
                this.isPanning = true;
                this.startX = event.touches[0].clientX -
this.originX;
                this.startY = event.touches[0].clientY -
this.originY;
            }
        },
        handleTouchMove(event) {
            event.preventDefault();
            if (event.touches.length === 1 && this.isPanning) {
                this.originX = event.touches[0].clientX -
this.startX;
                this.originY = event.touches[0].clientY -
this.startY;
                this.redrawCanvas();
            }
        },
        handleTouchEnd() {
            this.isPanning = false;
```

```
        },
        handleResize() {
            const container =
this.$parent.$el.querySelector('.canvas-container');
            const canvas = this.$refs.canvasRef;
            if (container.clientWidth > 0) {
                if (canvas.width != container.clientWidth) {
                    canvas.width = container.clientWidth;
                    canvas.height = container.clientWidth;
                    if (this.img.src.length > 0) {
                        this.resetZoom();
                    } else {
                        this.redrawCanvas();
                    }
                }
            }

        },
        redrawCanvas(sendBack = true) {
            const canvas = this.$refs.canvasRef;
            const ctx = canvas.getContext("2d");
            this.clearCanvas();
            ctx.setTransform(this.scale, 0, 0, this.scale,
this.originX, this.originY);
            ctx.drawImage(this.img, 0, 0);
            if (this.linkedCanvas && sendBack) {
                this.linkedCanvas.scale = this.scale;
                this.linkedCanvas.originX = this.originX;
                this.linkedCanvas.originY = this.originY;
                this.linkedCanvas.redrawCanvas(false);
            }
        },
        link(linkedCanvas) {
            this.linkedCanvas = linkedCanvas;
            this.linkedCanvas.linkedCanvas = this;
            this.linkedCanvas.isLinked = true;
        }
    },
    mounted() {
        const canvas = this.$refs.canvasRef;
        window.addEventListener('resize', this.handleResize);
        canvas.addEventListener('wheel', this.handleWheel);
        canvas.addEventListener('mousedown',
this.handleMouseDown);
        canvas.addEventListener('mousemove',
this.handleMouseMove);
        canvas.addEventListener('mouseup', this.handleMouseUp);
        canvas.addEventListener('mouseleave',
this.handleMouseUp);
        canvas.addEventListener('touchstart',
this.handleTouchStart);
```

```
        canvas.addEventListener('touchmove',
this.handleTouchMove);
        canvas.addEventListener('touchend',
this.handleTouchEnd);
        this.clearCanvas();
        this.observer = new
IntersectionObserver(this.handleResize);
        this.observer.observe(canvas, { attributes: true,
childList: true });
    },
    beforeDestroy() {
        const canvas = this.$refs.canvasRef;
        window.removeEventListener('resize', this.handleResize);
        canvas.removeEventListener('wheel', this.handleWheel);
        canvas.removeEventListener('mousedown',
this.handleMouseDown);
        canvas.removeEventListener('mousemove',
this.handleMouseMove);
        canvas.removeEventListener('mouseup',
this.handleMouseUp);
        canvas.removeEventListener('mouseleave',
this.handleMouseUp);
        canvas.removeEventListener('touchstart',
this.handleTouchStart);
        canvas.removeEventListener('touchmove',
this.handleTouchMove);
        canvas.removeEventListener('touchend',
this.handleTouchEnd);
        if (this.observer) {
            this.observer.disconnect();
        }
    },
    template: `
        <link rel="stylesheet" type="text/css"
href="/style/canvas.css"/>
        <div class="canvas-container">
            <canvas ref="canvasRef" class="image-
canvas"></canvas>
            <div v-if="!this.isLinked" class="canvas-control">
                <button class="canvas-control-button"
@click="zoom(1)">&#xf00e;</button>
                <button class="canvas-control-button"
@click="zoom(-1)">&#xf010;</button>
                <button class="canvas-control-button"
@click="resetZoom()">&#xf066;</button>
                <button class="canvas-control-button"
@click="resetZoom(1)">&#xf065;</button>
                <Text>{{ $t('canvas.scale') }}: {{ (this.scale *
100).toFixed(0) }}%</Text>
            </div>
        </div>
    `,
```

```
    i18n: {
        messages: {
            en: {
                canvas: {
                    scale: 'Scale',
                }
            },
            uk: {
                canvas: {
                    scale: 'Масштаб',
                }
            },
            de: {
                canvas: {
                    scale: 'Maßstab',
                }
            },
        }
    },
};
```

Файл *'.\components\UI\overlay.js'*:

```
export default {
    name: 'Overlay',
    data() {
        return {
            isVisible: false,
            text: "",
        };
    },
    methods: {
        show() {
            this.isVisible = true;
            document.body.classList.add('no-interaction');
        },
        hide() {
            this.isVisible = false;
            document.body.classList.remove('no-interaction');
        },
    },
    template: `
        <div v-if="isVisible" class="overlay">
            <link rel="stylesheet" type="text/css"
href="/style/overlay.css"/>
            <div class="loading-wrapper">
                <div class="loading-animation"></div>
                <div class="text">{{ $t('wait') }}<br><strong>{{
text }}</strong></div>
            </div>
        </div>
    `,
    i18n: {
```

```
        messages: {
            en: {
                wait: "It may take some time. Please wait.",
            },
            uk: {
                wait: "Це може зайняти деякий час. Будь ласка,
зачекайте.",
            },
            de: {
                wait: "Es kann eine Weile dauern. Bitte
warten.",
            },
        }
    }
};
```

Файл *'.\components\UI\tabstack.js'*:

```
export default {
    template: `
        <link rel="stylesheet" type="text/css"
href="/style/tabstack.css"/>
        <div class="tabstack-container">
            <div class="tabstack-buttons">
                <button
                    v-for="(tab, index) in tabs"
                    :key="index"
                    @click="currentTab = index"
                    :class="{'active-tab': currentTab ===
index}"
                >
                    {{ tab.title }}
                </button>
            </div>
                <div class="tabstack-content">
                <slot :name="tabs[currentTab].id"></slot>
                </div>
            </div>
        </div>
    `,
    name: 'TabStack',
    props: {
        tabs: {
            type: Array,
            required: true,
        },
    },
    data() {
        return {
            currentTab: 0,
        };
    },
}
```

Файл *'.\components\views\AboutView.js'*:

```
export default {
    name: 'AboutView',
    template: `
           <div style="padding: 20px; text-align: left; font-
family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #F3F2F1; border-radius: 8px; box-shadow: 0 4px
8px rgba(0, 0, 0, 0.1);">
           <a href='https://github.com/xuebinqin/U-2-Net'><img
src="/static/u2net-logo.webp" alt="U-2-Net Logo" style="width:
25%; max-width: 300px; height: auto; margin-top: 10px; display:
block; margin-left: auto; margin-right: auto;"/></a>
    <br><br>
           <strong>{{ $t('intro') }}</strong>
           <br>
           {{ $t('description') }}<br>
<ul>
    <li>{{ $t('salientObjectDetection') }}</li>
    <li>{{ $t('portraitCreation') }}</li>
    <li>{{ $t('peopleSegmentation') }}</li>
</ul>
    <br><br>
    <strong>{{ $t('architectureOverview') }}</strong>
    <br>
    {{ $t('architectureDescription') }}
    <ul>
        <li>{{ $t('nestedUShapes') }}</li>
        <li>{{ $t('deepSupervision') }}</li>
        <li>{{ $t('receptiveFieldExpansion') }}</li>
    </ul>
    <img src="/static/u2net-architecture.webp" alt="U-2-Net
Architecture" style="width: 100%; max-width: 650px; height:
auto; margin-top: 10px; display: block; margin-left: auto;
margin-right: auto;"/>
    <br><br>
    <strong>{{ $t('keyFeatures') }}</strong>
    <ul>
        <li>{{ $t('twoLevelArchitecture') }}</li>
        <li>{{ $t('residualBlocks') }}</li>
        <li>{{ $t('attentionMechanisms') }}</li>
    </ul>
    <strong>{{ $t('backgroundSubtraction') }}</strong>
    <ul>
        <li>{{ $t('saliencyMapGeneration') }}</li>
        <li>{{ $t('postProcessing') }}</li>
        <li>{{ $t('backgroundManipulation') }}</li>
    </ul>
        </div>
    `,
    i18n: {
        messages: {
```

```
            en: {
                intro: "This tool uses the U-2-Net model, a deep
learning architecture designed for salient object detection and
segmentation.",
                description: "It is particularly effective in
identifying and segmenting objects in images, making it suitable
for tasks like background removal and object extraction:",
                salientObjectDetection: "Salient Object
Detection: U-2-Net highlights the most visually prominent
objects in an image by generating a saliency map, useful in
tasks like image summarization and attention modeling.",
                portraitCreation: "Portrait Creation: It
separates a person from their background, enabling background
blurring, replacement, or artistic effects in photography and
video.",
                peopleSegmentation: "People Segmentation: U-2-
Net creates detailed masks for individuals in images or videos,
often used in augmented reality, virtual try-ons, and video
editing.",
                architectureOverview: "Architecture Overview",
                architectureDescription: "U-2-Net is built on a
nested U-shaped architecture, inspired by the U-Net structure
but with added depth and innovation. Here’s what sets it
apart:",
                nestedUShapes: "Nested U-Shapes (U^2-Net):
Instead of a single U-shaped encoder-decoder structure, U-2-Net
uses multiple nested U-nets within its encoder and decoder
modules. This improves the model's ability to capture both
global context and fine-grained details.",
                deepSupervision: "Deep Supervision: Each layer
of the nested U-shape is supervised independently during
training, ensuring that the model learns robust features at
different scales.",
                receptiveFieldExpansion: "Receptive Field
Expansion: By combining deep and shallow layers within the
nested architecture, U-2-Net effectively enlarges the receptive
field to better capture context from the entire image.",
                keyFeatures: "Key Features",
                twoLevelArchitecture: "Two-level Hierarchical
Architecture: The first level captures coarse features, while
the second focuses on finer details, enhancing precision in
segmentation tasks.",
                residualBlocks: "Residual Blocks: U-2-Net
incorporates residual blocks to improve gradient flow during
training, reducing vanishing gradient problems and enabling
deeper networks.",
                attentionMechanisms: "Attention Mechanisms:
Certain implementations integrate attention modules to enhance
feature discrimination, ensuring that the model focuses on
salient regions.",
                backgroundSubtraction: "Application for
Background Subtraction",
```

```
                saliencyMapGeneration: "Saliency Map Generation:
The model outputs a mask highlighting the foreground objects.",
                postProcessing: "Post-processing: Techniques
like thresholding or morphological operations refine the
segmentation result.",
                backgroundManipulation: "Background
Manipulation: The mask is used to extract or modify the
background as needed."
            },
            uk: {
                intro: "Цей інструмент використовує модель U-2-
Net, архітектуру глибокого навчання, призначену для виявлення та
сегментації помітних об'єктів.",
                description: "Вона особливо ефективна у
виявленні та сегментації об'єктів на зображеннях, що робить її
придатною для таких завдань, як видалення фону та екстракція
об'єктів:",
                salientObjectDetection: "Виявлення помітних
об'єктів: U-2-Net підкреслює найбільш візуально помітні об'єкти
на зображенні, генеруючи карту помітності, що корисно для таких
завдань, як узагальнення зображень і моделювання уваги.",
                portraitCreation: "Створення портретів: вона
відокремлює людину від її фону, що дозволяє розмивати фон,
замінювати його або створювати художні ефекти в фотографії та
відео.",
                peopleSegmentation: "Сегментація людей: U-2-Net
створює детальні маски для окремих осіб на зображеннях або
відео, часто використовується в доповненій реальності,
віртуальних примірках і редагуванні відео.",
                architectureOverview: "Огляд архітектури",
                architectureDescription: "U-2-Net побудований на
основі архітектури з гніздовими U-подібними формами, натхненної
структурою U-Net, але з доданою глибиною та інноваціями. Ось що
відрізняє його від інших:",
                nestedUShapes: "Гніздові U-форми (U^2-Net):
замість однієї U-подібної структури кодувальника-декодувальника
U-2-Net використовує кілька гніздових U-мереж у своїх модулях
кодувальника та декодувальника. Це покращує здатність моделі
захоплювати як глобальний контекст, так і деталі тонкого
масштабу.",
                deepSupervision: "Глибоке навчання: кожен шар
гніздової U-форми контролюється незалежно під час навчання, що
забезпечує навчання моделі надійним ознакам на різних
масштабах.",
                receptiveFieldExpansion: "Розширення
рецептивного поля: поєднуючи глибокі та поверхневі шари в межах
гніздової архітектури, U-2-Net ефективно збільшує рецептивне
поле, щоб краще захопити контекст з усього зображення.",
                keyFeatures: "Ключові особливості",
                twoLevelArchitecture: "Дворівнева ієрархічна
архітектура: перший рівень захоплює грубі риси, тоді як другий
```

```
зосереджується на тонших деталях, підвищуючи точність
сегментації.",
                residualBlocks: "Залишкові блоки: U-2-Net
включає залишкові блоки для покращення потоку градієнтів під час
навчання, зменшуючи проблеми з зникненням градієнта та
дозволяючи створювати глибші мережі.",
                attentionMechanisms: "Механізми уваги: певні
реалізації інтегрують модулі уваги для підвищення дискримінації
ознак, забезпечуючи зосередження моделі на помітних областях.",
                backgroundSubtraction: "Застосування для
видалення фону",
                saliencyMapGeneration: "Генерація карти
помітності: модель виводить маску, що підкреслює об'єкти
переднього плану.",
                postProcessing: "Постобробка: такі методи, як
порогове значення або морфологічні операції, уточнюють результат
сегментації.",
                backgroundManipulation: "Маніпуляція фоном:
маска використовується для вилучення або зміни фону за потреби."
            },
            de: {
                intro: "Dieses Tool verwendet das U-2-Net-
Modell, eine Deep-Learning-Architektur, die für die Erkennung
und Segmentierung auffälliger Objekte entwickelt wurde.",
                description: "Es ist besonders effektiv bei der
Identifizierung und Segmentierung von Objekten in Bildern, was
es für Aufgaben wie Hintergrundentfernung und Objektextraktion
geeignet macht:",
                salientObjectDetection: "Auffällige
Objekterkennung: U-2-Net hebt die auffälligsten Objekte in einem
Bild hervor, indem es eine Salienzkarten generiert, die nützlich
ist für Aufgaben wie Bildzusammenfassung und
Aufmerksamkeitsmodellierung.",
                portraitCreation: "Porträt Erstellung: Es trennt
eine Person von ihrem Hintergrund und ermöglicht
Hintergrundunschärfe, -ersatz oder künstlerische Effekte in
Fotografie und Video.",
                peopleSegmentation: "Personensegmentierung: U-2-
Net erstellt detaillierte Masken für Einzelpersonen in Bildern
oder Videos, die häufig in Augmented Reality, virtuellen
Anproben und Videobearbeitung verwendet werden.",
                architectureOverview: "Architekturübersicht",
                architectureDescription: "U-2-Net basiert auf
einer geschachtelten U-förmigen Architektur, die von der U-Net-
Struktur inspiriert ist, aber mit zusätzlicher Tiefe und
Innovation. Hier sind die Merkmale, die es auszeichnen:",
                nestedUShapes: "Verschachtelte U-Formen (U^2-
Net): Anstelle einer einzelnen U-förmigen Encoder-Decoder-
Struktur verwendet U-2-Net mehrere geschachtelte U-Netze
innerhalb seiner Encoder- und Decoder-Module. Dies verbessert
die Fähigkeit des Modells, sowohl den globalen Kontext als auch
feinkörnige Details zu erfassen.",
```

```
                deepSupervision: "Tiefe Überwachung: Jede
Schicht der geschachtelten U-Form wird während des Trainings
unabhängig überwacht, um sicherzustellen, dass das Modell
robuste Merkmale auf verschiedenen Skalen erlernt.",
                receptiveFieldExpansion: "Erweiterung des
rezeptiven Feldes: Durch die Kombination von tiefen und flachen
Schichten innerhalb der geschachtelten Architektur erweitert U-
2-Net effektiv das rezeptive Feld, um den Kontext des gesamten
Bildes besser zu erfassen.",
                keyFeatures: "Hauptmerkmale",
                twoLevelArchitecture: "Zwei-Ebenen-Hierarchische
Architektur: Die erste Ebene erfasst grobe Merkmale, während
sich die zweite auf feinere Details konzentriert und die
Präzision bei Segmentierungsaufgaben erhöht.",
                residualBlocks: "Residualblöcke: U-2-Net
integriert Residualblöcke, um den Gradientenfluss während des
Trainings zu verbessern, Probleme mit dem verschwindenden
Gradienten zu reduzieren und tiefere Netzwerke zu ermöglichen.",
                attentionMechanisms:
"Aufmerksamkeitsmechanismen: Bestimmte Implementierungen
integrieren Aufmerksamkeitsmodule, um die
Merkmalsdiskriminierung zu verbessern und sicherzustellen, dass
sich das Modell auf auffällige Regionen konzentriert.",
                backgroundSubtraction: "Anwendung zur
Hintergrundsubtraktion",
                saliencyMapGeneration: "Generierung von
Salienzkarten: Das Modell gibt eine Maske aus, die die Objekte
im Vordergrund hervorhebt.",
                postProcessing: "Nachbearbeitung: Techniken wie
Schwellenwertbildung oder morphologische Operationen verfeinern
das Segmentierungsergebnis.",
                backgroundManipulation:
"Hintergrundmanipulation: Die Maske wird verwendet, um den
Hintergrund nach Bedarf zu extrahieren oder zu ändern."
            },
        },
    },
}
```

Файл *'.\components\views\BGRemoveView.js'*:

```
import TabStack from '/components/UI/tabstack.js';
import Process from '/components/process.js';

export default {
    name: 'BGRemoveView',
    components: {
        TabStack,
        Process
    },
    template: `
        <tab-stack :tabs="tabs">
```

```
            <template #home>
                <process :mode="'salient_object'"></process>
            </template>
            <template #help>
                <div class="help-content">
                    <h2>{{ $t('help.title') }}</h2>
                    <ul>
                        <li>{{ $t('help.step1') }}</li>
                        <li>{{ $t('help.step2') }}</li>
                        <li>{{ $t('help.step3') }}</li>
                        <li>{{ $t('help.step4') }}</li>
                        <li>{{ $t('help.step5') }}</li>
                        <li>{{ $t('help.step6') }}</li>
                        <li>{{ $t('help.step7') }}</li>
                    </ul>
                </div>
            </template>
            <template #about>
                <div class="about-content">
                    <h2>{{ $t('about.title') }}</h2>
                    <p>{{ $t('about.about') }}</p>
                </div>
            </template>
        </tab-stack>
    `,
    computed: {
        tabs() {
            return [
                { id: 'home', title: this.$t('tabs.home') },
                { id: 'help', title: this.$t('tabs.help') },
                { id: 'about', title: this.$t('tabs.about') },
            ];
        }
    },
    i18n: {
        messages: {
            en: {
                tabs: {
                    home: "Home",
                    help: "Help",
                    about: "About",
                },
                help: {
                    title: "How to Use This Tool",
                    step1: "Upload an image to process by
clicking the 'Upload Image' button. This will open a file picker
dialog where you can select the image you want to process.",
                    step2: "Ensure the uploaded image is
displayed in the preview area. This allows you to confirm that
the correct image has been selected.",
```

```
                    step3: "Click the 'Remove Background' button
to start processing the image. The tool will use advanced AI
algorithms to detect and remove the background.",
                    step4: "Wait for the image processing to
complete. This may take a few seconds or a few minutes depending
on your hardware and the complexity of the image.",
                    step5: "Once processing is complete, review
the result in the preview area. You can zoom in or out to
inspect the details of the processed image.",
                    step6: "If satisfied, click the 'Download
Result' button to save the processed image to your device. The
image will be saved in a format that preserves transparency.",
                    step7: "To process another image, click the
'Upload New Image' button and repeat the steps above."
                },
                about: {
                    title: "About This Tool",
                    about: "This is a background removal tool
that uses advanced AI algorithms to remove backgrounds from
images. It is designed to be user-friendly and efficient,
allowing you to quickly process images without any technical
knowledge. The tool supports various image formats and provides
high-quality results suitable for professional use.",
                }
            },
            uk: {
                tabs: {
                    home: "Головна",
                    help: "Допомога",
                    about: "Про програму",
                },
                help: {
                    title: "Як користуватися цим інструментом",
                    step1: "Завантажте зображення для обробки,
натиснувши кнопку 'Завантажити зображення'. Це відкриє діалогове
вікно вибору файлу, де ви можете вибрати потрібне зображення.",
                    step2: "Переконайтеся, що завантажене
зображення відображається в області попереднього перегляду. Це
дозволяє підтвердити правильність вибору зображення.",
                    step3: "Натисніть кнопку 'Видалити фон', щоб
розпочати обробку зображення. Інструмент використовує передові
алгоритми штучного інтелекту для виявлення та видалення фону.",
                    step4: "Дочекайтеся завершення обробки
зображення. Це може зайняти кілька секунд або хвилин залежно від
вашого обладнання та складності зображення.",
                    step5: "Після завершення обробки перегляньте
результат у області попереднього перегляду. Ви можете
збільшувати або зменшувати масштаб для перевірки деталей
обробленого зображення.",
                    step6: "Якщо результат вас задовольняє,
натисніть кнопку 'Завантажити результат', щоб зберегти оброблене
```

```
зображення на свій пристрій. Зображення буде збережено у
форматі, який зберігає прозорість.",
                    step7: "Щоб обробити інше зображення,
натисніть кнопку 'Завантажити нове зображення' і повторіть
вищезазначені кроки."
                },
                about: {
                    title: "Про цей інструмент",
                    about: "Це інструмент для видалення фону,
який використовує передові алгоритми штучного інтелекту для
видалення фону із зображень. Він розроблений для зручності
користувачів і ефективності, дозволяючи швидко обробляти
зображення без технічних знань. Інструмент підтримує різні
формати зображень і забезпечує високу якість результатів,
придатних для професійного використання.",
                }
            },
            de: {
                tabs: {
                    home: "Startseite",
                    help: "Hilfe",
                    about: "Über das Tool",
                },
                help: {
                    title: "Wie man dieses Tool benutzt",
                    step1: "Laden Sie ein Bild zur Verarbeitung
hoch, indem Sie auf die Schaltfläche 'Bild hochladen' klicken.
Dadurch wird ein Dateiauswahldialog geöffnet, in dem Sie das
gewünschte Bild auswählen können.",
                    step2: "Stellen Sie sicher, dass das
hochgeladene Bild im Vorschau-Bereich angezeigt wird. So können
Sie bestätigen, dass das richtige Bild ausgewählt wurde.",
                    step3: "Klicken Sie auf die Schaltfläche
'Hintergrund entfernen', um die Bildverarbeitung zu starten. Das
Tool verwendet fortschrittliche KI-Algorithmen, um den
Hintergrund zu erkennen und zu entfernen.",
                    step4: "Warten Sie, bis die Bildverarbeitung
abgeschlossen ist. Dies kann je nach Hardware und Komplexität
des Bildes einige Sekunden oder Minuten dauern.",
                    step5: "Überprüfen Sie nach Abschluss der
Verarbeitung das Ergebnis im Vorschau-Bereich. Sie können
hinein- oder herauszoomen, um die Details des verarbeiteten
Bildes zu überprüfen.",
                    step6: "Wenn Sie zufrieden sind, klicken Sie
auf die Schaltfläche 'Ergebnis herunterladen', um das
verarbeitete Bild auf Ihrem Gerät zu speichern. Das Bild wird in
einem Format gespeichert, das die Transparenz beibehält.",
                    step7: "Um ein weiteres Bild zu verarbeiten,
klicken Sie auf die Schaltfläche 'Neues Bild hochladen' und
wiederholen Sie die oben genannten Schritte."
                },
                about: {
```

```
                    title: "Über dieses Tool",
                    about: "Dies ist ein Tool zur
Hintergrundentfernung, das fortschrittliche KI-Algorithmen
verwendet, um Hintergründe aus Bildern zu entfernen. Es wurde
benutzerfreundlich und effizient gestaltet, sodass Sie Bilder
schnell und ohne technisches Wissen verarbeiten können. Das Tool
unterstützt verschiedene Bildformate und liefert hochwertige
Ergebnisse, die für den professionellen Einsatz geeignet sind.",
                }
            }
        }
    }
}
```

Файл *'.\components\views\HomeView.js'*:

```
export default {
    name: 'HomeView',
    template: `
    <link rel="stylesheet" type="text/css"
href="/style/home.css"/>
        <nav class="nav-container">
            <ul class="nav-list">
                <li v-for="item in navItems" :key="item.name"
class="nav-item">
                    <router-link :to="item.link" class="nav-
link">
                        <img :src="item.image" class="nav-
image">
                        <h3 class="nav-title">{{ $t(item.name)
}}</h3>
                        <p class="nav-description">{{
$t(item.description) }}</p>
                    </router-link>
                </li>
            </ul>
        </nav>
    `,
    data() {
        return {
            navItems: [
                {
                    name: 'background_remove_page.title',
                    link: '/background_remove',
                    image: '/static/bg-remove.webp',
                    description:
'background_remove_page.description',
                },
                {
                    name: 'portrait_create_page.title',
                    link: '/portrait_create',
                    image: '/static/portrait.webp',
```

```
                    description:
'portrait_create_page.description',
                },
                {
                    name: 'people_remove_page.title',
                    link: '/people_remove',
                    image: '/static/people-remove.webp',
                    description:
'people_remove_page.description',
                },
                {
                    name: 'about_page.title',
                    link: '/about',
                    image: '/static/about.webp',
                    description: 'about_page.description'
                },
            ],
        };
    },
    i18n: {
        messages: {
            en: {
            },
        },
    },
}
```

Файл *'.\components\views\PeopleRemoveView.js'*:

```
import TabStack from '/components/UI/tabstack.js';
import Process from '/components/process.js';

export default {
    name: 'PeopleRemoveView',
    components: {
        TabStack,
        Process
    },
    template: `
        <tab-stack :tabs="tabs">
            <template #home>
                <process :mode="'human'"></process>
            </template>
            <template #help>
                <div class="help-content">
                    <h2>{{ $t('help.title') }}</h2>
                    <ul>
                        <li>{{ $t('help.step1') }}</li>
                        <li>{{ $t('help.step2') }}</li>
                        <li>{{ $t('help.step3') }}</li>
                        <li>{{ $t('help.step4') }}</li>
                        <li>{{ $t('help.step5') }}</li>
                        <li>{{ $t('help.step6') }}</li>
```

```
                    </ul>
                </div>
            </template>
            <template #about>
                <div class="about-content">
                    <h2>{{ $t('about.title') }}</h2>
                    <p>{{ $t('about.about') }}</p>
                </div>
            </template>
        </tab-stack>
    `,
    computed: {
        tabs() {
            return [
                { id: 'home', title: this.$t('tabs.home') },
                { id: 'help', title: this.$t('tabs.help') },
                { id: 'about', title: this.$t('tabs.about') },
            ];
        }
    },
    i18n: {
        messages: {
            en: {
                tabs: {
                    home: "Home",
                    help: "Help",
                    about: "About",
                },
                help: {
                    title: "How to Use This Tool",
                    step1: "Upload an image by clicking the
'Upload Image' button. This will open a file picker dialog where
you can select the image you want to process.",
                    step2: "Ensure the uploaded image is
displayed in the preview area. This allows you to confirm that
the correct image has been selected.",
                    step3: "Click the 'Remove People' button to
start processing the image. The tool will use advanced AI
algorithms to remove people from your image.",
                    step4: "Wait for the image processing to
complete. This may take a few seconds or a few minutes depending
on your hardware and the complexity of the image.",
                    step5: "Once processing is complete, review
the result in the preview area. You can zoom in or out to
inspect the details of the processed image.",
                    step6: "If satisfied, click the 'Download
Result' button to save the processed image to your device.",
                },
                about: {
                    title: "About This Tool",
                    about: "This tool allows you to remove
people from your images using advanced AI technology. It is
```

```
designed to be user-friendly and efficient, enabling you to
process images quickly and achieve professional-quality
results."
                }
            },
            uk: {
                tabs: {
                    home: "Головна",
                    help: "Допомога",
                    about: "Про програму",
                },
                help: {
                    title: "Як користуватися цим інструментом",
                    step1: "Завантажте зображення, натиснувши
кнопку 'Завантажити зображення'. Це відкриє діалогове вікно
вибору файлу, де ви можете вибрати потрібне зображення.",
                    step2: "Переконайтеся, що завантажене
зображення відображається в області попереднього перегляду. Це
дозволяє підтвердити правильність вибору зображення.",
                    step3: "Натисніть кнопку 'Видалити людей',
щоб почати обробку зображення. Інструмент використовує передові
алгоритми штучного інтелекту для видалення людей із вашого
зображення.",
                    step4: "Дочекайтеся завершення обробки
зображення. Це може зайняти кілька секунд або хвилин залежно від
вашого обладнання та складності зображення.",
                    step5: "Після завершення обробки перегляньте
результат у області попереднього перегляду. Ви можете
збільшувати або зменшувати масштаб для перевірки деталей
обробленого зображення.",
                    step6: "Якщо результат вас влаштовує,
натисніть кнопку 'Завантажити результат', щоб зберегти оброблене
зображення на свій пристрій.",
                },
                about: {
                    title: "Про цей інструмент",
                    about: "Цей інструмент дозволяє видаляти
людей із ваших зображень за допомогою передових технологій
штучного інтелекту. Він розроблений для зручності та
ефективності, що дозволяє швидко обробляти зображення та
досягати професійної якості результатів."
                }
            },
            de: {
                tabs: {
                    home: "Startseite",
                    help: "Hilfe",
                    about: "Über das Tool",
                },
                help: {
                    title: "Wie man dieses Tool benutzt",
```

```
                    step1: "Laden Sie ein Bild hoch, indem Sie
auf die Schaltfläche 'Bild hochladen' klicken. Dadurch wird ein
Dateiauswahldialog geöffnet, in dem Sie das gewünschte Bild
auswählen können.",
                    step2: "Stellen Sie sicher, dass das
hochgeladene Bild im Vorschau-Bereich angezeigt wird. So können
Sie bestätigen, dass das richtige Bild ausgewählt wurde.",
                    step3: "Klicken Sie auf die Schaltfläche
'Personen entfernen', um die Bildverarbeitung zu starten. Das
Tool verwendet fortschrittliche KI-Algorithmen, um Personen aus
Ihrem Bild zu entfernen.",
                    step4: "Warten Sie, bis die Bildverarbeitung
abgeschlossen ist. Dies kann je nach Hardware und Komplexität
des Bildes einige Sekunden oder Minuten dauern.",
                    step5: "Überprüfen Sie nach Abschluss der
Verarbeitung das Ergebnis im Vorschau-Bereich. Sie können
hinein- oder herauszoomen, um die Details des bearbeiteten
Bildes zu überprüfen.",
                    step6: "Wenn Sie zufrieden sind, klicken Sie
auf die Schaltfläche 'Ergebnis herunterladen', um das
bearbeitete Bild auf Ihrem Gerät zu speichern.",
                },
                about: {
                    title: "Über dieses Tool",
                    about: "Dieses Tool ermöglicht es Ihnen,
Personen aus Ihren Bildern mithilfe fortschrittlicher KI-
Technologie zu entfernen. Es ist benutzerfreundlich und
effizient gestaltet, sodass Sie Bilder schnell verarbeiten und
Ergebnisse in professioneller Qualität erzielen können."
                }
            }
        }
    }
}
```

Файл *'.\components\views\PortraitCreateView.js'*:

```
import TabStack from '/components/UI/tabstack.js';
import Process from '/components/process.js';

export default {
    name: 'PortraitCreateView',
    components: {
        TabStack,
        Process
    },
    template: `
        <tab-stack :tabs="tabs">
            <template #home>
                <process :mode="'portrait'"></process>
            </template>
            <template #help>
                <div class="help-content">
```

```
                    <h2>{{ $t('help.title') }}</h2>
                    <ul>
                        <li>{{ $t('help.step1') }}</li>
                        <li>{{ $t('help.step2') }}</li>
                        <li>{{ $t('help.step3') }}</li>
                        <li>{{ $t('help.step4') }}</li>
                        <li>{{ $t('help.step5') }}</li>
                        <li>{{ $t('help.step6') }}</li>
                    </ul>
                </div>
            </template>
            <template #about>
                <div class="about-content">
                    <h2>{{ $t('about.title') }}</h2>
                    <p>{{ $t('about.about') }}</p>
                </div>
            </template>
        </tab-stack>
    `,
    computed: {
        tabs() {
            return [
                { id: 'home', title: this.$t('tabs.home') },
                { id: 'help', title: this.$t('tabs.help') },
                { id: 'about', title: this.$t('tabs.about') },
            ];
        }
    },
    i18n: {
        messages: {
            en: {
                tabs: {
                    home: "Home",
                    help: "Help",
                    about: "About",
                },
                help: {
                    title: "How to Use This Tool",
                    step1: "Upload an image to process by
clicking the 'Upload Image' button. This will open a file picker
dialog where you can select the image you want to process.",
                    step2: "Ensure the uploaded image is
displayed in the preview area. This allows you to confirm that
the correct image has been selected.",
                    step3: "Click the 'Create Portrait' button
to start processing the image. The tool will use advanced AI
algorithms to create an artistic portrait from your image.",
                    step4: "Wait for the image processing to
complete. This may take a few seconds or a few minutes depending
on your hardware and the complexity of the image.",
```

```
                    step5: "Once processing is complete, review
the result in the preview area. You can zoom in or out to
inspect the details of the processed image.",
                    step6: "If satisfied, click the 'Download
Result' button to save the processed image to your device.",
                    step7: "To process another image, click the
'Upload New Image' button and repeat the steps above."
                },
                about: {
                    title: "About This Tool",
                    about: "This is a portrait creation tool
that allows you to design and personalize portraits from your
images. It provides a variety of tools and features to help you
create professional-quality portraits with ease. Whether for
personal or professional use, this tool is designed to be
intuitive and efficient."
                }
            },
            uk: {
                tabs: {
                    home: "Головна",
                    help: "Допомога",
                    about: "Про програму",
                },
                help: {
                    title: "Як користуватися цим інструментом",
                    step1: "Завантажте зображення для обробки,
натиснувши кнопку 'Завантажити зображення'. Це відкриє діалогове
вікно вибору файлу, де ви можете вибрати потрібне зображення.",
                    step2: "Переконайтеся, що завантажене
зображення відображається в області попереднього перегляду. Це
дозволяє підтвердити правильність вибору зображення.",
                    step3: "Натисніть кнопку 'Створити портрет',
щоб розпочати обробку зображення. Інструмент використовує
передові алгоритми штучного інтелекту для створення художнього
портрета з вашого зображення.",
                    step4: "Дочекайтеся завершення обробки
зображення. Це може зайняти кілька секунд або хвилин залежно від
вашого обладнання та складності зображення.",
                    step5: "Після завершення обробки перегляньте
результат в області попереднього перегляду. Ви можете
збільшувати або зменшувати масштаб, щоб детально розглянути
оброблене зображення.",
                    step6: "Якщо вас влаштовує результат,
натисніть кнопку 'Завантажити результат', щоб зберегти оброблене
зображення на свій пристрій.",
                    step7: "Щоб обробити інше зображення,
натисніть кнопку 'Завантажити нове зображення' та повторіть
наведені вище кроки."
                },
                about: {
                    title: "Про цей інструмент",
```

```
                    about: "Це інструмент для створення
портретів, який дозволяє розробляти та персоналізувати портрети
з ваших зображень. Він надає різноманітні інструменти та
функції, які допоможуть вам створювати портрети професійної
якості з легкістю. Незалежно від того, чи це для особистого чи
професійного використання, цей інструмент розроблений, щоб бути
інтуїтивно зрозумілим та ефективним."
                }
            },
            de: {
                tabs: {
                    home: "Startseite",
                    help: "Hilfe",
                    about: "Über das Tool",
                },
                help: {
                    title: "Wie man dieses Tool benutzt",
                    step1: "Laden Sie ein Bild zur Verarbeitung
hoch, indem Sie auf die Schaltfläche 'Bild hochladen' klicken.
Dadurch wird ein Dateiauswahldialog geöffnet, in dem Sie das
gewünschte Bild auswählen können.",
                    step2: "Stellen Sie sicher, dass das
hochgeladene Bild im Vorschau-Bereich angezeigt wird. Dadurch
können Sie bestätigen, dass das richtige Bild ausgewählt
wurde.",
                    step3: "Klicken Sie auf die Schaltfläche
'Porträt erstellen', um die Bildverarbeitung zu starten. Das
Tool verwendet fortschrittliche KI-Algorithmen, um ein
künstlerisches Porträt aus Ihrem Bild zu erstellen.",
                    step4: "Warten Sie, bis die Bildverarbeitung
abgeschlossen ist. Dies kann je nach Hardware und Komplexität
des Bildes einige Sekunden oder Minuten dauern.",
                    step5: "Sobald die Verarbeitung
abgeschlossen ist, überprüfen Sie das Ergebnis im Vorschau-
Bereich. Sie können hinein- oder herauszoomen, um die Details
des verarbeiteten Bildes zu betrachten.",
                    step6: "Wenn Sie zufrieden sind, klicken Sie
auf die Schaltfläche 'Ergebnis herunterladen', um das
verarbeitete Bild auf Ihrem Gerät zu speichern.",
                    step7: "Um ein weiteres Bild zu verarbeiten,
klicken Sie auf die Schaltfläche 'Neues Bild hochladen' und
wiederholen Sie die oben genannten Schritte."
                },
                about: {
                    title: "Über dieses Tool",
                    about: "Dies ist ein Porträt-
Erstellungstool, mit dem Sie Porträts aus Ihren Bildern
entwerfen und personalisieren können. Es bietet eine Vielzahl
von Werkzeugen und Funktionen, die Ihnen helfen, Porträts in
professioneller Qualität mit Leichtigkeit zu erstellen. Ob für
den persönlichen oder professionellen Gebrauch, dieses Tool ist
darauf ausgelegt, intuitiv und effizient zu sein."
```

```
                }
            }
        }
    }
}
```

Файл *'.\style\canvas.css'*:

```
.canvas-container {
    align-items: center;
    justify-content: center;
    height: 100%;
    min-width: 30%;
}

canvas.image-canvas {
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1), 0px 1px 3px
rgba(0, 0, 0, 0.06);
    border-radius: 8px;
    background-color: #f3f3f3;
}

.canvas-control {
    display: flex;
    justify-content: left;
    align-items: left;
    width: 100%;
    gap: 10px;
    padding: 5px 0px;
    margin: 5px 0px;
}

button.canvas-control-button {
    font-family: 'FontAwesome';
    background-color: #0078d4;
    color: white;
    border: none;
    border-radius: 4px;
    padding: 8px 16px;
    font-size: 14px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
button.canvas-control-button:hover {
    background-color: #005a9e;
}
button.canvas-control-button:active {
    background-color: #004578;
}
```

Файл *'.\style\footer.css'*:

```
/* Footer Styles */
```

```
footer {
    display: block;
    width: 100%;
    background-color: #333;
    color: #fff;
    padding: 20px 0;
    border-top: 2px solid #ccc;
    border-radius: 5px;
    text-align: center;
    font-family: 'FontAwesome', 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif;
    font-size: 14px;
}

footer a {
    color: #00bcd4;
    text-decoration: none;
}

footer a:hover {
    text-decoration: underline;
}

footer .footer-links {
    margin: 10px 0;
}

footer .footer-links a {
    margin: 0 10px;
}

footer p {
    margin: 5px 0;
}
```

Файл *'.\style\header.css'*:

```
select {
    height: fit-content;
    align-self: center;
    padding: 5px;
    border-radius: 4px;
    border: 1px solid #ccc;
    background-color: #fff;
    cursor: pointer;
    font-family: 'Noto Color Emoji', 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif;
}

.header {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
```

```
    gap: 10px;
    align-items: center;
    padding: 10px 20px;
    background-color: #0078D4;
    color: white;
    border-bottom: 2px solid #ccc;
    border-radius: 5px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
}

.logo {
    height: 60px;
    width: auto;
}

.title-wrapper {
    display: flex;
    flex-direction: column;
    flex-grow: 1;
}

.title {
    font-size: 24px;
    font-weight: bold;
    margin: 0;
    text-align: left;
}

.subtitle {
    font-size: 15px;
    margin: 0;
    text-align: left;
}

.selector {
    display: flex;
    flex-grow: 0;
    gap: 10px;
}

.home-link {
    color: white;
    text-decoration: none;
    padding: 10px 15px;
    border-radius: 5px;
    align-items: right;
    display: flex;
    transition: background-color 0.3s ease;
    font-family: 'FontAwesome', 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif;
}
```

```
.home-link:hover {
    background-color: rgba(255, 255, 255, 0.2);
}

@font-face {
    font-family: 'Noto Color Emoji';
    src: url(https://raw.githack.com/googlefonts/noto-
emoji/main/fonts/NotoColorEmoji.ttf);
}
```

Файл *'.\style\home.css'*:

```
.nav-container {
    padding: 20px;
    background-color: #F3F2F1;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.nav-list {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
    display: flex;
    flex-wrap: wrap;
    list-style: none;
    padding: 0;
    align-items: center;
    justify-content: center;
}
.nav-item {
    background-color: #FFFFFF;
    margin: 10px;
    border: 1px solid #ccc;
    border-radius: 8px;
    padding: 10px;
    width: 300px;
    text-align: center;
}
.nav-link {
    display: block;
    text-decoration: none;
    color: #323130;
}
.nav-title {
    font-size: 18px;
    margin: 10px 0 5px;
}
.nav-description {
    font-size: 14px;
    color: #666;
}
.nav-image {
    width: 100%;
```

```
    height: auto;
    border-radius: 4px;
}
```

Файл *'.\style\main.css'*:

```
#app {
    min-height: calc(100vh - 20px);
    display: flex;
    flex-direction: column;
    flex-wrap: nowrap;
    color: #333;
}

button {
    background-color: #0078d4;
    color: white;
    border: none;
    border-radius: 4px;
    padding: 8px 16px;
    font-size: 14px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
button:hover {
    background-color: #005a9e;
}
button:active {
    background-color: #004578;
}

@font-face {
    font-family: "FontAwesome";
    font-weight: normal;
    font-style : normal;
           src : url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-webfont.eot?v=4.3.0");
           src : url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-webfont.eot?#iefix&v=4.3.0")
format("embedded-opentype"),
                url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-webfont.woff2?v=4.3.0")
format("woff2"),
                url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-webfont.woff?v=4.3.0")
format("woff"),
                url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-webfont.ttf?v=4.3.0")
format("truetype"),
                url("https://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/fonts/fontawesome-
webfont.svg?v=4.3.0#fontawesomeregular") format("svg");
}
```

Файл *'.\style\overlay.css'*:

```
.overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(255, 255, 255, 0.6); /* Semi-
transparent white */
    backdrop-filter: blur(10px); /* Frosted glass effect */
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 1000;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Subtle shadow
for depth */
}

.loading-wrapper {
    text-align: center;
}

.loading-animation {
    width: 75px;
    height: 75px;
    border: 10px solid rgba(200, 200, 200, 0.6); /* Lighter
border */
    border-top: 10px solid rgba(0, 120, 215, 1); /* Fluent
Design accent color */
    border-radius: 50%;
    animation: spin 1s linear infinite;
    margin: 0 auto;
}

.text {
    margin-top: 15px;
    font-size: 18px;
    color: #333;
}

@keyframes spin {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}
```

Файл *'.\style\process.css'*:

```
.drop-area {
    padding-top: 10%;
    padding-bottom: 10%;
    border: 2px dashed #0078d4; /* Fluent UI primary color */
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: pointer;
    text-align: center;
    background-color: #f3f2f1; /* Fluent UI neutral light */
    border-radius: 4px;
    transition: background-color 0.2s ease, border-color 0.2s
ease;
}
.drop-area:hover {
    background-color: #e1dfdd; /* Fluent UI neutral lighter */
    border-color:rgb(0, 90, 158); /* Fluent UI primary darker */
}

.canvas-wrapper {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: center;
    align-items: top;
    width: 100%;
    min-height: 50%;
}

.buttons-wrapper {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 10px;
    margin-top: 20px;
}

/* Fluent-style button styling */
.download-btn {
    background-color: #0E8A16; /* Green */
    color: white;
    border: none;
    width: 80%;
    padding: 10px 20px;
    border-radius: 6px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
```

```
.download-btn:hover {
    background-color: #0C7212; /* Darker green */
}

.start-btn {
    background-color: #0078D4; /* Blue */
    color: white;
    border: none;
    width: 80%;
    padding: 10px 20px;
    border-radius: 6px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.start-btn:hover {
    background-color: #006CBE; /* Darker blue */
}

.cancel-btn {
    background-color: #D32F2F; /* Red */
    color: white;
    border: none;
    width: 80%;
    padding: 10px 20px;
    border-radius: 6px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.cancel-btn:hover {
    background-color: #B71C1C; /* Darker red */
}
```

Файл *'.\style\tabstack.css'*:

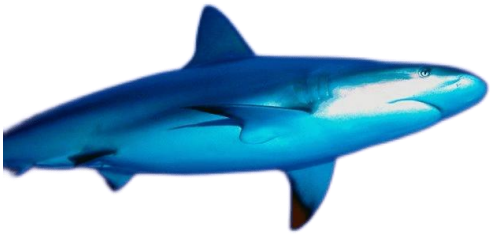
```
.tabstack-container {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
    padding: 20px;
    background-color: #F3F2F1;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.tabstack-buttons {
    display: flex;
    gap: 10px;
    margin-bottom: 20px;
}
.tabstack-buttons button {
    border: none;
```

```
    border-radius: 4px;
    padding: 10px 20px;
    cursor: pointer;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    background-color: #F3F2F1;
    color: #323130;
}
.tabstack-buttons button.active-tab {
    background-color: #0078D4;
    color: #FFFFFF;
}
.tabstack-content {
    padding: 20px;
    background-color: #FFFFFF;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.tabstack-text {
    color: #323130;
    font-size: 16px;
}
```

Додаток В. Вхідні та вихідні дані

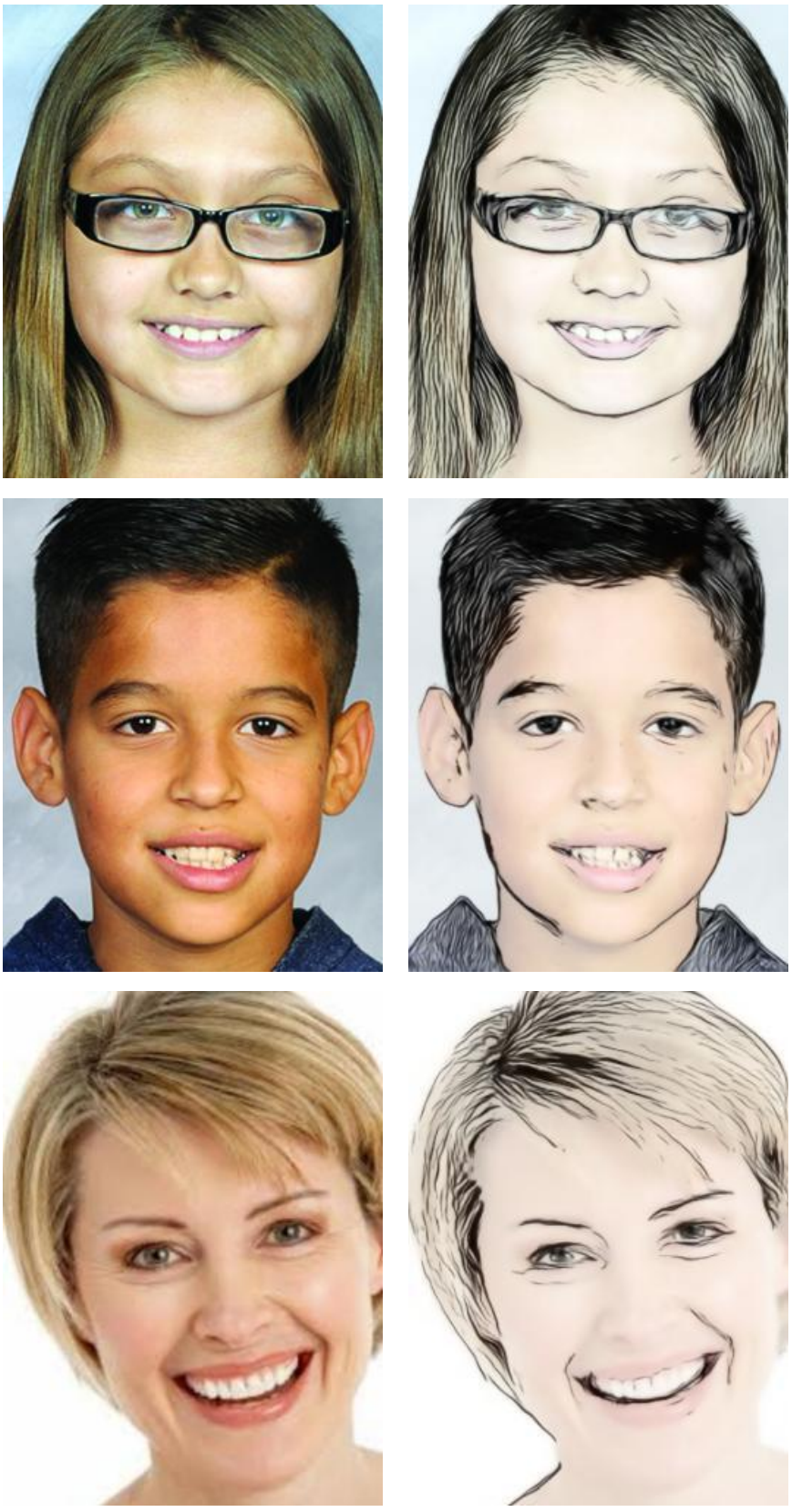
Результати видалення фону:

Вхідне зображення *Вихідне (перетворене) зображення*



Результати створення художніх портретів:

Вхідне зображення *Вихідне (перетворене) зображення*



Результати видалення людей на зображеннях:

Вхідне зображення *Вихідне (перетворене) зображення*