ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методу виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

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

_____ Юрій ВОЙЦЕХОВСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти група ПДМ-64
Юрій ВОЙЦЕХОВСЬКИЙ

Керівник: Андрій БОНДАРЧУК
д.т.н., професор

Рецензент:
науковий ступінь, вчене звання
_____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Войцеховському Юрій Юрійовичу

1. Тема кваліфікаційної роботи: Розробка методу виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання

керівник кваліфікаційної роботи Андрій БОНДАРЧУК д.т.н., професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, метод виявлення та класифікації дефектів, метрики сегментації.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження предметної області та засобів сегментації зображень.
2. Аналіз технологій машинного навчання та можливості застосування для виявлення та класифікації дефектів на металевих поверхнях.

3. Розробка моделі нейронної мережі та програмного забезпечення.
4. Тестування та аналіз отриманих результатів.

5. Перелік графічного матеріалу: *презентація*
   1. Мета, об’єкта та предмет дослідження.
   2. Обґрунтування вибору моделі.
   3. Архітектура нейронної мережі.
   4. Метрики оцінки ефективності.
   5. Результати тестування.

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд предметної області	19.10-05.11.23	
2	Аналіз наявних методів сегментації зображень	06.11-12.11.23	
3	Розгляд архітектур нейронних мереж для сегментації зображень	13.11-19.11.23	
4	Проектування моделі нейронної мережі	20.11-26.11.23	
5	Розробка програмного забезпечення	27.11-03.12.23	
6	Тестування та порівняльний аналіз результатів	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Юрій ВОЙЦЕХОВСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 72 стор., 3 табл., 41 рис., 39 джерел.

Мета роботи – покращення процесу виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

Об'єкт дослідження – процес виявлення та класифікації дефектів на металевих поверхнях.

Предмет дослідження – засоби виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

Методи дослідження – методи теорії сегментації зображень, методи теорії машинного навчання, методи проектування та розробки програмного забезпечення, технології об'єктно-орієнтованого програмування.

Короткий зміст роботи: У роботі здійснено огляд використовуваних методів класової сегментації зображень, аналіз наявних технічних засобів для виявлення та класифікації дефектів на металевих поверхнях, проведено розгляд архітектур нейронних мереж, що можуть застосовуватися для виконання поставленого завдання. Проаналізовано можливості використання модифікацій архітектури, обґрунтовано вибір використаних методів, визначено математичні критерії ефективності, проведено проектування методу виявлення та класифікації дефектів на металевих поверхнях. Розроблено нейронну мережу, що надає можливість детектування дефектів за чотирма класами та виділення виявлених об'єктів відповідними контурами. Проведено навчання нейронної мережі на базі набору тренувальних даних, створеного з залученням кваліфікованого фахівця. Проведено тестування розробленої системи, здійснено аналіз одержаних результатів. Проведено порівняльну характеристику з результатами роботи наявних засобів, підтверджено наявність покращених показників.

КЛЮЧОВІ СЛОВА: ВИЯВЛЕННЯ ДЕФЕКТІВ, МАШИННЕ НАВЧАННЯ, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, КЛАСИФІКАЦІЯ, НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

Text part of the master's qualification work:

72 pages, 41 pictures, 3 tables, 39 sources.

The purpose of the work is improving the process of detecting and classifying defects on metal surfaces using machine learning technologies.

Object of research – the process of detecting and classifying defects on metal surfaces.

Subject of research – means of detecting and classifying defects on metal surfaces using machine learning technologies.

Summary of the work: The work includes a review of the methods used for image class segmentation, an analysis of available technical means for detection and classification of defects on metal surfaces, a consideration of neural network architectures applicable to the defined task. The possibilities of using architectural modifications are analyzed, the choice of methods is substantiated, mathematical criteria for efficiency are determined, and the design of a method for detecting and classifying defects on metal surfaces is carried out. A neural network has been developed that enables the detection of defects in four classes and the highlighting of detected objects with corresponding contours. The neural network was trained on a dataset created with the involvement of a qualified expert. Testing of the developed system was conducted, and an analysis of the obtained results was performed. A comparative characterization was carried out with the results of existing tools, confirming the presence of improved indicators.

KEYWORDS: DEFECT DETECTION, MACHINE LEARNING, IMAGE SEGMENTATION, CLASSIFICATION, NEURAL NETWORKS.

ЗМІСТ

ВСТУП

Виявлення дефектів представляє собою вагомий аспект будь-якого процесу виготовлення або використання виробів. Даний аспект сприяє забезпеченню якості продукції та надійності її експлуатації, знижуючи небезпеку виникнення негативних наслідків як для споживачів, так і для постачальників.

Наразі різноманітна металева продукція має значне поширення у численних галузях застосування. За цією причиною виявлення її поверхневих пошкоджень, що можуть значним негативним чином впливати на загальну якість виробу, є актуальним питанням у виробничих процесах. Дані пошкодження несуть ризики для подальшої промислової обробки чи експлуатації виробів, в гіршому випаду призводячи до непередбачуваних подій, що несуть небезпеку для здоров'я оточуючих.

Виявлення дефектів на металевих поверхнях надає змогу дослідити потенційні причини, що спричиняють їх появу, та запровадити до промислових процесів належні міри, що позитивним чином впливає на практичну та фінансову результативність виробництва, скорочуючи збиткові відсотки.

Отже, задача розробки методу виявлення та класифікації дефектів на металевих поверхнях є сучасною та актуальною.

Об'єкт дослідження – процес виявлення та класифікації дефектів на металевих поверхнях.

Предмет дослідження – засоби виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

Мета роботи – покращення процесу виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

Методи дослідження – методи теорії сегментації зображень, методи тес машинного навчання, методи проектування та розробки програм забезпечення, технології об'єктно-орієнтованого програмування.

Практична значущість результатів полягає в застосуванні розробленого методу для детектування дефектів на металевих поверхнях під час виробництва сталевого прокату у промислових умовах.

Для досягнення мети виконувалися наступні задачі:

1. Розгляд наявних методів виявлення пошкоджень на металевих поверхнях у виробничих умовах.
2. Дослідження архітектур нейронних мереж для вирішення завдань сегментації зображень.
3. Проектування методу детектування пошкоджень на зображеннях сталевого прокату за визначеними класами.
4. Розробка нейронної мережі засобами програмної реалізації.
5. Навчання нейронної мережі на базі набору тренувальних даних, створеного з залученням кваліфікованого фахівця
6. Проведення тестування розробленої системи, аналіз одержаних результатів, складання порівняльної характеристики.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЇ ДЕФЕКТІВ НА МЕТАЛЕВИХ ПОВЕРХНЯХ

1.1 Аналіз видів дефектів металевого прокату

Проведемо розгляд різних типів можливих дефектів на металевих поверхнях, базуючись на даних з професійних джерел [1, 2, 3, 4].

Розкатана тріщина (рис. 1.1) – це дефект, що може утворитись на металевій поверхні під час обробки виробу розкатом. Цей дефект являє собою наслідок поєднання низької температури та механічної напруги, що можуть негативно вплинути на міцність металу та призвести до утворення тріщини. Може виникнути внаслідок неоднорідності початкового матеріалу, що може з'явитися, наприклад, у процесі виробництва, обробки або зберігання.

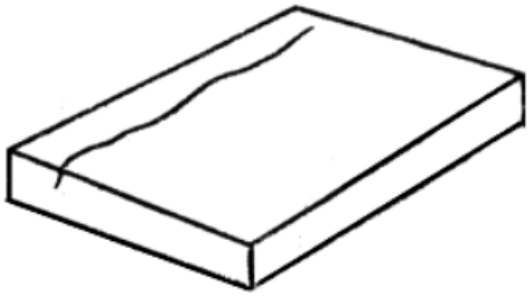


Рис. 1.1 Розкатана тріщина

Зливкова рванина (рис. 1.2) – це дефект, що має можливість утворитись під час лиття металевої продукції, коли залитий матеріал піддається нерівномірному затвердінню. Як результат у ділянці зливу з'являється тріщина чи розлом. Причина такого явища полягає у тому, що під час лиття металу залиття піддається

поступовому охолодженню, та у тому, що залитий матеріал змінюється у загальному об'ємі, що може призвести до виникнення напруг у ділянці зливу.

Рис. 1.2 Зливкова рванина

Розкатана бульбашка (рис. 1.3) – це дефект, що може з'явитися на металевій поверхні під час обробки виробу розкатом. Вигляд даного дефекту представляє собою схожу на бульбашку пухку ділянку, вигнуту в середину. Виникають через затримки газів, що перебувають у металі, у процесі розкату.

Рис. 1.3 Розкатана бульбашка

Бульбашка-здуття (рис. 1.4) – це дефект металевої поверхні, що являє собою округлу порожнину або випуклість. Даний дефект може володіти різними розмірними характеристиками та може утворюватися як під час виробництва, так протягом використання виробу.

Причиною виникнення бульбашок-здуттів є промисловий процес, коли у структурі металу протягом його затвердіння зостається надмірна кількість розчиненого газу або інших домішок. У випадку, якщо даний газ або домішки виходять на металічну поверхню, вони можуть утворити бульбашку-здуття.

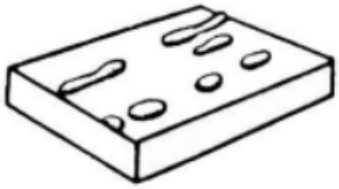


Рис. 1.4 Бульбашка-здуття

Зливкова пліва (рис. 1.5) – це дефект металевих поверхонь, що утворюється під час лиття матеріалу у зливок. Даний дефект являє собою тонкий шар металу, що з'являється на зовнішній поверхні зливка і може відділятися від основної структури.

Зливкова пліва може здійснити негативний плив на якість продукту, особливо у випаду, якщо її розташування знаходиться у критичних ділянках виробу. Причиною появи зливкової пліви є явище, коли під час процесу лиття поверхня розплавленого металу вступає у контакт з повітрям чи іншими речовинами, що мають змогу вступити з металом у хімічні реакції. Зазвичай зливкова пліва виникає на зовнішній поверхні низькотемпературних сплавів на прикладі алюмінію у зв'язку з тим, що дані сплави порівняно швидко вступають у реакцію з повітряним киснем та іншими газуватими речовинами.

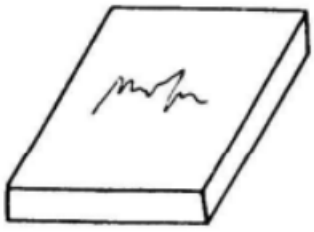


Рис. 1.5 Зливкова пліва

Розкатане забруднення (рис. 1.6) – це дефект, що виникає на металевих поверхнях під час промислових процесі. Даний дефект являє собою вузьку шаровидну включену композицію, що може відрізнятися за своєю формою та розмірами. Розкатане забруднення виникає за нестачі здібності розчинення різних забруднень у матеріалі протягом виробництва.

Рис. 1.6 Розкатане забруднення

Лускатість (рис. 1.7) представляє собою один з поверхневих дефектів металевого матеріалу, коли на зовнішньому шарі металу виникають згорблення, що за формою нагадують хвилі. Даний дефект може утворитися протягом обробки поверхні матеріалу, зокрема протягом фрезерування, гартування або у випадку обробки різцем. Ключової причиною появи лускатості є неправильне виконання обробки, наприклад, перебільшене навантаження на інструментарій.

Рис. 1.7 Лускатість

Тріщина напруження (рис. 1.8) – це дефект на металевій поверхні, що виникає через високі навантаження і у результаті загрожує руйнуванням виробу.

Здебільшого ці тріщини з'являються безпосередньо на зовнішньому шарі виробу чи в глибині металу, через мінливість у навантаженні на матеріал. Це може відбутися, наприклад, при згині чи видовженні виробу у тому випадку, якщо метал має порівняно невисокий показник міцності.

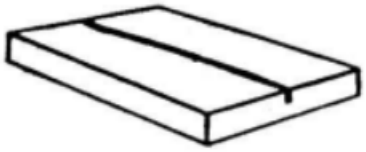


Рис. 1.8 Тріщина напруження

Деформаційна рванина (рис. 1.9) – це дефект, що може утворитися під час лиття продукції, що включає у своїй структурі високотемпературні сплави. Деформаційна рванина зазвичай утворюється через нерівномірне зміцнення матеріалу протягом охолодження, що може спричинити розрив структури у ділянках, на які припадає високе навантаження.

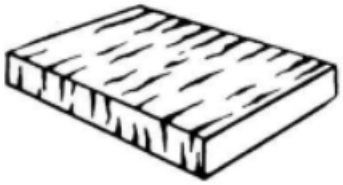


Рис. 1.9 Деформаційна рванина

Прокатана плівка (рис. 1.10) – це дефект, що представляє собою тонкий металевий шар, який утворюється на металевій поверхні протягом прокатки. Даний металевий шар може відрізнятись за формою та розмірами та приймати вигляд сірої або білої плівки, що відокремлюється від продукту протягом деформації. Поява прокатаної плівки може бути спричинена некоректною

взаємодією між матеріалом та поверхнею валків протягом процесу виробничої прокатки.

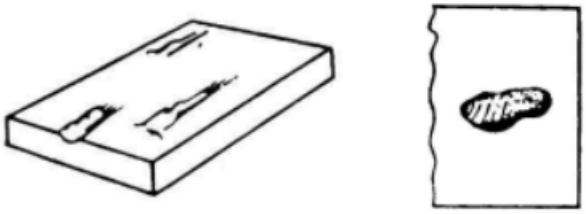


Рис. 1.10 Прокатана плівка

Риска (рис. 1.11) – це дефект на металевій поверхні, що представляє собою візуально помітну лінію чи щілину, що може мати пряму або довільну форму. Поява рисок може бути зумовлена різними чинниками, що загалом пов'язані з експлуатацією валків, наприклад, недостачею мастила, наявністю нерівностей або перегріванням.

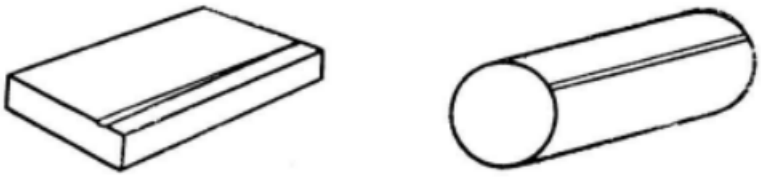


Рис. 1.11 Риска

Вкатана окалина (рис. 1.12) – це дефект металевих поверхонь, що здебільшого виникає після гарячої прокатки. Даний дефект з'являється, коли протягом прокатки між валками потрапляють деякі забруднення та залишаються між ними та поверхнею продукту. У подальшому дані забруднення здавлюються та прямим чином приєднуються до металевої структури, створюючи плівку на

зовнішньому шарі матеріалу. Дана плівка може мати різний загальний рівень нерівностей та негативним чином впливати на якість продукту.

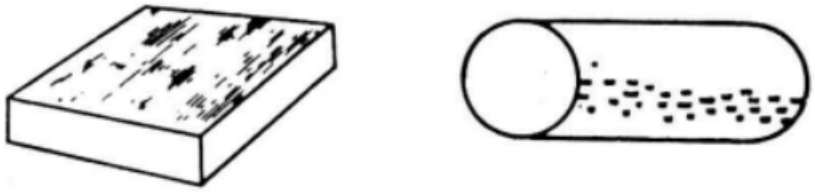


Рис. 1.12 Вкатана окалина

Рябизна (рис. 1.13) – це дефект зовнішнього шару матеріалу, що представляє собою появу на металевій поверхні ямок чи отворів, що можуть відрізнятися своїми розмірами та глибиною. Рябизна з'являється протягом промислових процесів, наприклад, прокатки чи гартування, коли на зовнішньому шарі матеріалу виникають мікротріщини. Дані мікротріщини можуть представляти собою результат недостатньо якісної обробки металевої поверхні, сумнівної якості початкових матеріалів чи недотримання вимог виробничого процесу.

Рис. 1.13 Рябизна

Раковина-вдав (рис. 1.14) – це вид дефектів на металевих поверхнях, який утворюється під час затвердіння розплавленого матеріалу. Виникненя раковини-вдаву зумовлене випадками, коли метал, що твердіє, втрачає деякий об'єм

протягом охолодження, але через недостачу рідини у литому металі дана втрата не може бути компенсована, результатом чого стає поява порожнини у металі.

Рис. 1.14 Раковина-вдав

Відбитки (рис. 1.15) – це дефекти поверхонь металевих виробів, які утворюються протягом процесу прокатки. Відбитки являють собою заглиблені ямки, що розташовуються вздовж металевої поверїні, здебільшого розміщуючись паралельними рядками. Поява відбитків відбувається через неоднорідне надходження металевої сировини до прокатного станка, результатом чого стає зміна напружень у продукті. У тому випадку, коли дані напруження переходять межу міцності продукту, трапляється деформація його зовнішнього шару, і з'являються відбитки.

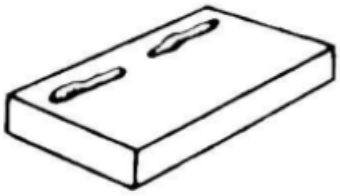


Рис. 1.15 Відбитки

1.2 Оцінка методів сегментації зображень

Сегментація зображень представляє собою процес розбивання пікселів на зображенні у деякі групи, які відповідають об'єктам чи регіонам, що розташовано на зображенні [5]. Даний процес є одним із ключових етапів у численних завданнях, що пов'язані з обробкою зображень, і процес виявлення дефектів на металевих поверхнях полягає у переліку таких завдань.

Метод K-середніх (K-Means) належить до переліку одних з найбільш поширених засобів кластерного аналізу. Даний алгоритм застосовується для систематизації первинного набору даних у деяку кількість кластерів [6, 7].

Загальну структуру алгоритму K-Means можна описати наступним чином:

1. Визначається кількість кластерів K, що задається користувачем.

2. Визначаються початкові центроїди кластерів. Центроїдою називають фіктивну точку, що являє собою центр кожного з наявних кластерів.

3. Кожна з точок даних призначається до найближчої центроїди.

4. Для кожної центроїди проводиться обчислення нового середнього значення з використанням всіх точок даних, що було призначено до неї.

5. Відбувається повторення третього та четвертого кроків до тих пір, поки не відбудеться стабілізація усіх центроїді або досягнення максимально можливої кількості ітерацій.

Головним недоліком методу K-Means є його велика чутливість до шуму та викидів у дані через можливий вплив на центроїди та спричинення похибок, що можуть призвести до неякісної кластеризації. Окрім того, вибір збалансованої кількості кластерів може стати складним завданням для користувача, адже результатом хибного визначення може стати неправильна кластеризація або значні втрати доцільних даних.

Метод порогової обробки (Thresholding) – це один з найпоширеніших та найбільш простих за структурою методів сегментації зображень, ключовий принцип якого полягає в розбитті пікселів на дві групи, в залежності від деякого порогового значення. Суть методу полягає в тому, що пікселі, чий показник

переважає над пороговим значенням, включаються до однієї групи, у той час як до іншої групи відповідним чином включаються пікселі з меншим показником [8].

Загальна структура методу порогової обробки виглядає наступним чином:

1. Завантаження зображення та переведення його кольорової палітри у сірий колір.

2. Вибір користувачем порового значення, що буде використовуватися.

3. Перевірка всіх пікселів зображення відповідно до порогового значення. У випадку, коли показник пікселя перевищує порогове значення, він включається до однієї групи, у протилежному випадку – до другої.

4. Опціональне використання допоміжних операцій для вдосконалення кінцевих результатів.

5. Збереження підсумку сегментації у якості чорно-білого зображення, на якому першу групу позначено білими пікселями, а другу групу – чорними.

Головними недоліками методу порогової обробки є його слабкість у роботі з зображеннями, для яких притаманний значний показник контрастності, та з дефектами, для яких є характерною відсутність виразно зазначених меж. Також визначення збалансованого порогового значення може потребувати значного фаху з боку користувача та загалом представляти собою складне завдання.

Метод активних контурів (Active Contours Model) застосовує власне контур, що видовжується по вхідному зображенню та трансформується відповідно з його властивостями [9, 10].

При використанні даного методу спершу навкіл ділянки зображення, де розташовується об’єкт, що потрібно виділити, формується контур, який представляє собою замкнуту криву. Після цього дана крива видовжується по зображенню, трансформуючись та зсуваючись відносно градієнту яскравості. Процес буде повторюватися доти, доки даний контур та реальний контур об’єкта, що необхідно виділити, не зійдуться.

Недоліками методу активних контурів можна назвати значну залежність від початкових показників, слабкість у роботі зі складними формами об’єктів,

потенційно високі часові витрати, складне налаштування параметрів, вразливість до даних зі значним показником контрастності.

Метод сегментації зображень з застосуванням згорткових нейронних мереж (Convolutional Neural Networks) представляє собою один з найбільш ефективних засобів сегментації зображень. Згорткові нейронні мережі являють собою нейромережеві архітектури, що обчислюють значення функцій на базі певних ділянок первинного зображення, які іменують ядрами [11, 12].

Ключовий принцип роботи полягає у застосуванні комплекту загорткових шарів, що окреслюють прості ознаки зображення, наприклад, текстури або межі. Дані ознаки приймають подальші шари нейронної мережі, що комбінують їх для виокремлення більш складних ознак, що застосовуються для класифікації та сегментації.

Згорткові нейронні мережі надають більш точні результати порівняно з іншими методами сегментації зображень, маючи змогу обробляти значні набори даних, швидко вдосконалювати свої алгоритми та обирати найбільш доцільні ознаки зображень.

Однак у той же час згорткові нейронні мережі здебільшого потребують значної апаратної потужності, що ускладнює їх застосування на приладах несучасних моделей. Окрім того, згорткові нейронні мережі вимагають значних об’ємів тренувальних даних, що є проблемою для застосування у деяких галузях. Тим не менш, на даний час згорткові нейронні мережі представляють собою доволі потужний засіб для завдань, що стосуються виявлення об’єктів, забезпечуючи високу точність результатів та швидкодію обробки.

1.3 Огляд наявних робіт

У праці «Detection of surface defects on steel strips based on singular value decomposition of digital image» представлено метод виявлення та орієнтовного визначення місцеположення дефектів металевої поверхні, в основі якого полягає розкладу на одиночні значення (Singular Value Decomposition) [13]. Матриця сірих

відтінків зображення проектується на одиночні вектори, отримані за допомогою розкладу на одиночні значення. Дефект відображається у якості раптової зміни в даних проекціях. Таким чином відбувається виявлення та орієнтовне визначення місцеположення дефектів.

Однак ефективність даного методу зменшується, якщо на зображенні присутні одинарні дефекти занадто великих розмірів або значна кількість невеликих дефектів, розкиданих по всьому зображенню. Також на ефективність розглянутого методу значним чином впливає освітлення: для прикладу, системи може розглянути направлене в одну ділянку світло як дефект на металевій поверхні. Передбачені системою розміри та форма не завжди відповідають дійсності, число виявлених дефектів не завжди відповідає дійсності.

Авторами статті «Segmentation of rust defects on painted steel surfaces by intelligent image analysis» описано метод, принцип якого полягає у підвищенні точності результатів виявлення корозійних пошкоджень з використанням традиційних засобів [14]. На результати обробки зображень негативним чином впливають незбалансоване освітлення і схожості між кольорами корозійних утворень та захисного покриття. Ці фактори можуть призвести до порушень у роботі сегментаційних алгоритмів. Висунутий авторами метод для виявлення корозійних дефектів на пофарбованих металевих поверхнях включає використання вслід за попередньою обробкою зображень процедури альфа-матування, що здійснює сегментацією з моделлю змішування за Гауссом.

Даний метод утворює помітні спотворення зображення біля контурів тіней, що певною мірою розв'язується за рахунок вибору деяких порогових значень. Однак даний підхід дієво працює лише у випадку невеликих ділянок у зв'язку з тим, що різні області зображення можуть потребувати різних порогових значень для ефективного виявлення дефектів. Також алгоритм сегментації корозійних пошкоджень на червоному фоні несе значну ймовірність похибки для зображень, де розташовані лише пошкоджені або лише цілісні ділянки.

У праці «Localization and segmentation of metal cracks using deep learning» представлена пропозиція залучення технологій комп'ютерного зору для детекції

дефектів на металевих поверхнях [15]. В основі даного методу полягає застосування архітектури глибинних згорткових нейронних мереж. Модель оперує кількома засобами попередньої та кінцевої обробки зображення для оптимізації методу, ефективно виокремлюючи ділянки навіть за випадків, коли первинне зображення включає у себе значну кількість спотворень. Однією з додаткових переваг розглянутої системи є гнучкість її структури: її можна з порівняною легкістю адаптувати для розв'язання подібних завдань з виявлення об'єктів на зображеннях, впровадивши в алгоритм незначні зміни.

Тим не менш, наведений метод призначения для виявлення тільки одного класу дефектів – тріщин, що є недоліком у зв'язку з частою потребою у виявленні та класифікації дефектів більш ніж одного виду. Через це наявна потреба створення методу на основі технологій машинного навчання, що матиме здатність сегментувати зображення, де наявні кілька різних видів пошкоджень.

Автори праці «The Amalgamation of the Object Detection and Semantic Segmentation for Steel Surface Defect Detection» пропонують ієрархічний засіб розпізнавання та класифікації пошкоджень на металевих поверхнях [16]. Розглянутий метод застосовує ієрархічну структуру для розділу об'єктів на два класи на першому рівні обробки, та засоби семантичної сегментації на другому рівні обробки.

Водночас, недоліком наведеного методу можна назвати те, що обидва рівні обробки використовують дві окремі нейронні мережі. Дана архітектура є складною: натомість однієї мережі необхідно здійснювати тренування двох, що негативним чином впливає на адаптивні можливості моделі та ускладнює вимоги до тренувальних даних. Також, автори наводять, що системі важко розпізнавати окремі пошкодження через те, що відмінність між ними та загальним фоном зображень є невиразно.

1.4 Аналіз існуючих методів та алгоритмів для виявлення та класифікації дефектів на металевих поверхнях

Здійснивши аналіз наукових джерел на тему роботи, з методів виявлення дефектів на металевих поверхнях можна виділити такі напрями, як метод K-середніх, метод порогової обробки, метод активних контурів та метод сегментації зображень з застосуванням згорткових нейронних мереж.

Метод K-середніх, метод порогової обробки та метод активних контурів несуть певні недоліки, значною мірою пов’язані з необхідністю налаштування складних параметрів та порівняно низька ефективність у роботі з об’єктами, що мають складні або нечіткі межі.

Тому доцільним визнано застосування згорткових нейронних мереж, що мають потужний базовий функціонал в області виявлення об’єктів на зображеннях та здатність ефективно тренуватися за рахунок значних об’ємів даних, завдяки своїй архітектурі широко використовуючись у різноманітних завданнях, що потребують роботи з зображеннями.

1.5 Аналіз сучасних засобів програмної інженерії для створення методу виявлення та класифікації дефектів на металевих поверхнях

Для реалізації методу виявлення та класифікації дефектів за допомогою технологій машинного навчання будо обрано мову програмування Python. Було застосовано спеціальну бібліотеку «segmentation_models», що базується на бібліотеці Keras. Keras зі свого боку базується на бібліотеці TensorFlow. TensorFlow було розроблено групою спеціалістів Google Brain, а авторство Keras в свою чергу полягає за французьким програмним інженером Франсуа Шолле з Google. TensorFlow використовується компанією Google у значній кількості послуг та програмних продуктів, що пов’язані з комплексними завданнями машинного навчання. Компанією Google було здійснено значний вплив на вдосконалення TensorFlow загалом та Keras зокрема, тому дані засоби мають

статус одних з найбільш поширених та пропонованими опціями для розробок у сфері машинного навчання.

Мова програмування Python володіє статусом однієї з найпоширеніших мов програмування у галузі створення нейронних мереж. Її особливими рисами є мінімалізм та простота структури, що надають можливість створювати комплексні алгоритми у порівняно недовгі терміни [17, 18]. Нейронні мережі являють собою зазвичай невеликі програми, проте зчаста виникають потреби у внесенні в них певних змін, здійснюючи вибір архітектури, передобробки даних да інших показників, тому використання мови програмування Python є опцією, що вдовольняє дані вимоги. Окрім того, на сьогодні Python включає великий перелік бібліотек, що є спеціалізованими інструментами для створення нейронних мереж.

TensorFlow представляє собою наскрізну платформу машинного навчання з відкритим вихідним кодом, що було створено корпорацією Google. TensorFlow здобув популярність завдяки доступній навчальній підтримці, гнучким можливостям масштабування, наявності кількох рівнів абстракції та можливості до підтримки різних платформ [19, 20].

TensorFlow утворює бібліотеку символьної математики, що застосовується у розробці нейронних мереж та ефективним чином личить для програмування потоків даних в об’ємному переліку задач. TensorFlow надає декілька рівнів складності для конструювання та тренування моделей.

Keras представляє собою якісний високорівневий інтерфейс для розробки нейронних мереж, створений на мові програмування Python. Завданням даної бібліотеки з відкритим вихідним кодом є здійснення швидких експериментів з нейронними мережами, що фіксується на комфортності використання, модульних властивостях та здатності до розширення [21].

1.6 Вибір архітектури нейронної мережі

Визначення конкретної архітектури нейронної мережі представляє собою один із ключових етапів проектування. Серед факторів, що підкреслюють обґрунтування важливості цього вибору, полягають: запланований розмір та складність даних, потреби в апаратній потужності, рівень інтерпретованості архітектури, доступність навчальних даних.

FCN (Fully Convolutional Network) – це архітектура нейронної мережі (рис. 1.16), спеціалізована для реалізації завдань семантичної сегментації графічних зображень. FCN була однією з найперших структур нейронних мереж, що надали змогу до реалізації завдань семантичної сегментації графічних зображень з порівняно значною точністю. Архітектура FCN застосовує лише згорткові та транспоновані прошарки, що забезпечує можливість обробки зображень, що відрізняються за розмірами, та визначення маски, що відображає клас кожного з окремих пікселів зображення [22, 23].

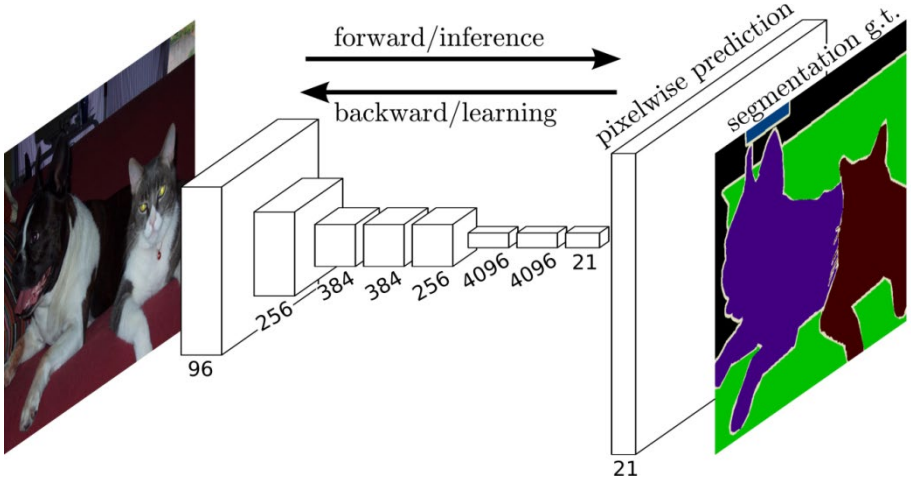


Рис. 1.16 Архітектура FCN

Структуру FCN можна поділити на дві ключові частини, представлені енкодером та декодером. Енкодер утворюється з декількох шарів згортки, які здійснюють поступову згортку вхідного зображення, зменшуючи його розміри та в ході даного процесу зберігаючи його контекстні дані. Декодер утворюється з

транспонованих шарів згортки, які здійснюють розширення розмірів зображення і повертають його роздільну здатність. Здебільшого для кожного з класів, які необхідно розпізнати, FCN застосовує кілька каналів маски. Вслід за одержанням маски можна здійснити встановлення класу для кожного із пікселів, що розташовані на зображенні. Основною перевагою і в одночас недоліком архітектури FCN є порівняно незначна кількість параметрів. Нейромережі з даною структурою можуть мати більш високі показники швидкодії для певних завдань та використовувати менш значні об'єми пам'яті на пристроях у зв'язку з тим, що вони включають меншу кількість параметрів та не застосовують проміжні з'єднувальні шари. Однак за даними ж причинами вони можуть надавати результати з нижчою точністю для зображень, що мають значні показники контрастності та роздільної здатності. Також існує проблема з поновленням різних форм об'єктів у зв'язку з тим, що архітектура не застосовує для зберігання контекстних даних з'єднувальні шари.

Архітектура DeepLab, створена компанією Google, має у своему складі кілька вагомих новаторських аспектів на прикладі атрибутної пулінгової мережі та модулю атрибутних згорток.

Структура DeepLab застосовує методи з мережами глибокого навчання (рис. 1.17), зосібна шари згортки та пудлінгу для вилучення ознак зображення різних рівнів. Тим не менш, архітектура додатково застосовує модуль атрибут них згорток, що надає можливість фіксуватися на ділянках зображення з вагомою значимістю.

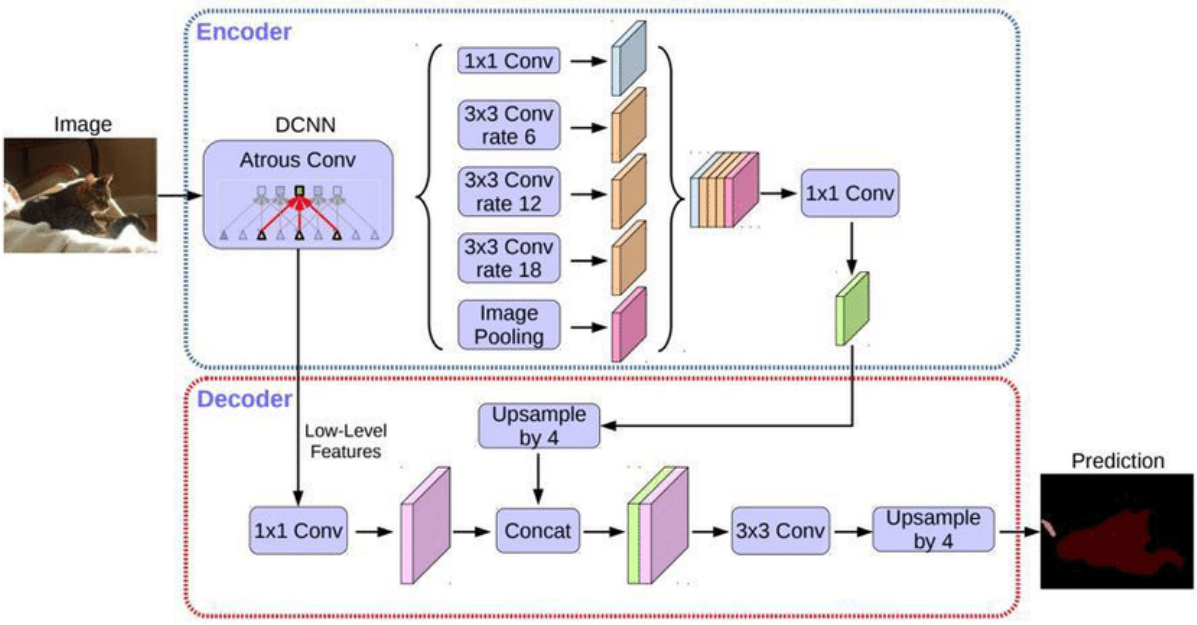


Рис. 1.17 Архітектура DeepLab V3+

Однією з ключових ознак структури DeepLab є застосування множинних шарів атрибутної пулінгової мережі, що надає можливість вивчати вхідне зображення на різних масштабах. Також, у складі архітектури полягає декодер, який застосовує виокремлення зображень та інтерполяцію, що дозволяє досягнути ефективних результатів сегментації [24, 25].

Модель DeepLab володіє декількома перевагами, серед яких значний показник точності сегментації, здібність до обробки зображень, що є високооб'ємними, та порівняно висока швидкість роботи, яка забезпечує можливість до використання архітектурі в режимі реального часу.

Однак дана структура несе певні недоліки, у числі яких можна виділити доволі комплексну структуру та значні потреби у апаратній потужності, що обмежує можливості використання нейронних мереж на її основі для пристоїв з відносно простою побудовою. Окрім того, на ефективність обробки певним чином може впливати наявність спотворень на межах сегментації.

Архітектура LinkNet (рис. 1.18) є однією з архітектур нейронних мереж, створених для сегментації зображень, чия спеціалізація сфокусована на обробці графічних даних високої якості з порівняно невисокими об'ємами наборів даних.

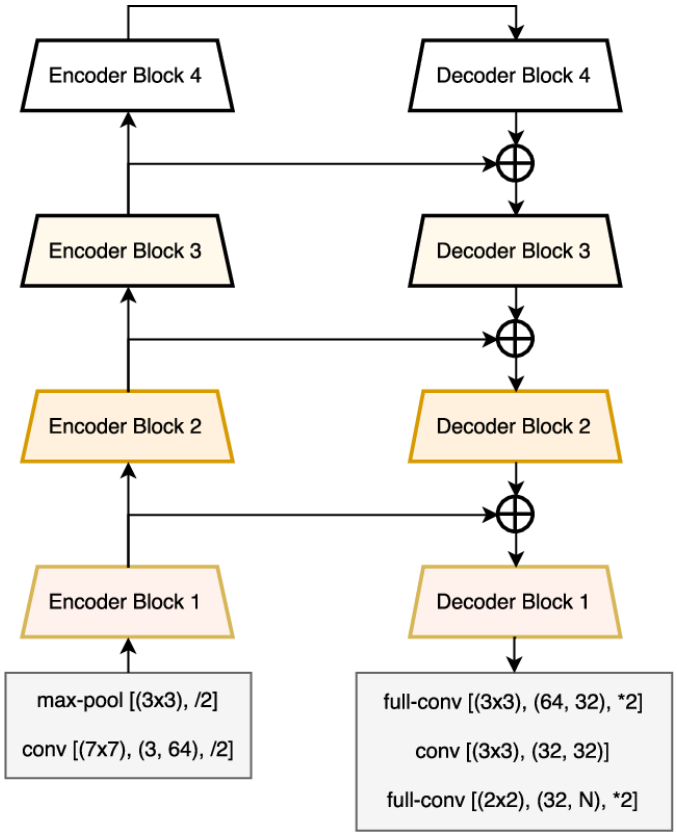


Рис. 1.18 Архітектура LinkNet

Структура LinkNet складається з енкодера, декодера та проміжної зв'язки, що об'єднує відповідні шари енкодера та декодера. Ключовий концепт архітектури полягає у скороченні переліку параметрів, що мають потребу у тренуванні, та організації більш дієвої обробки невеликих наборів даних [26].

У складі енкодера полягають шари згортки та пулінгу, що піддають вхідне зображення поступовому зменшенню та розбивають його на невеликі частини-сегменти. Після цього кожен із шарів у складі декодера розгортається та складається з шаром енкодера, який йому відповідає, для того аби повернути розміри відображення та відбудувати сегментацію зображення.

В структурі зв'язки, що поєднує енкодер та декодер, застосовується не класичне з'єднання (concatenation), а зв’язок виду "shortcut", який пересилає дані без необхідності у підвищенні кількості показників, що надає можливість пришвидшити тренування нейронної мережі.

Серед основних переваг описаної структури полягають високі показники швидкодії та загальної ефективності у зв’язку з тим, що нейронні мережі на її

основі мають можливість обробляти зображення значних розмірів з високою роздільною здатністю. Також рівень зв'язки забезпечує можливість зберігання вагомих подробиць під час перебудови зображення, завдяки чому система надає порівняно значну точність сегментації.

В числі недоліків полягає невисока потужність LinkNet відносно аналогових архітектур нейронних мереж. Також, точність її роботи може падати під час обробки зображень, що містять комплексну побудову та текстуру, де розпізнання кордонів об'єктів стає важким завданням.

На даний час одні з найбільш якісних результатів стосовно виконання задач сегментації зображень надає архітектура U-Net (рис. 1.19). Дана архітектура представлена згортковою побудовою, що є оптимізованою для семантичної сегментації графічних даних. Головним концептом U-Net є розподіл мережі на два шляхи, через які пролягає прохід інформації: звужуючий і розширюючий. Метою звужуючого шляху є поступове зменшення розмірів графічного зображення та вилучення абстрактних ознак. Призначенням розширюючого шляху є відбудова зображення до первинних розмірів та точна побудова сегментаційної маски.

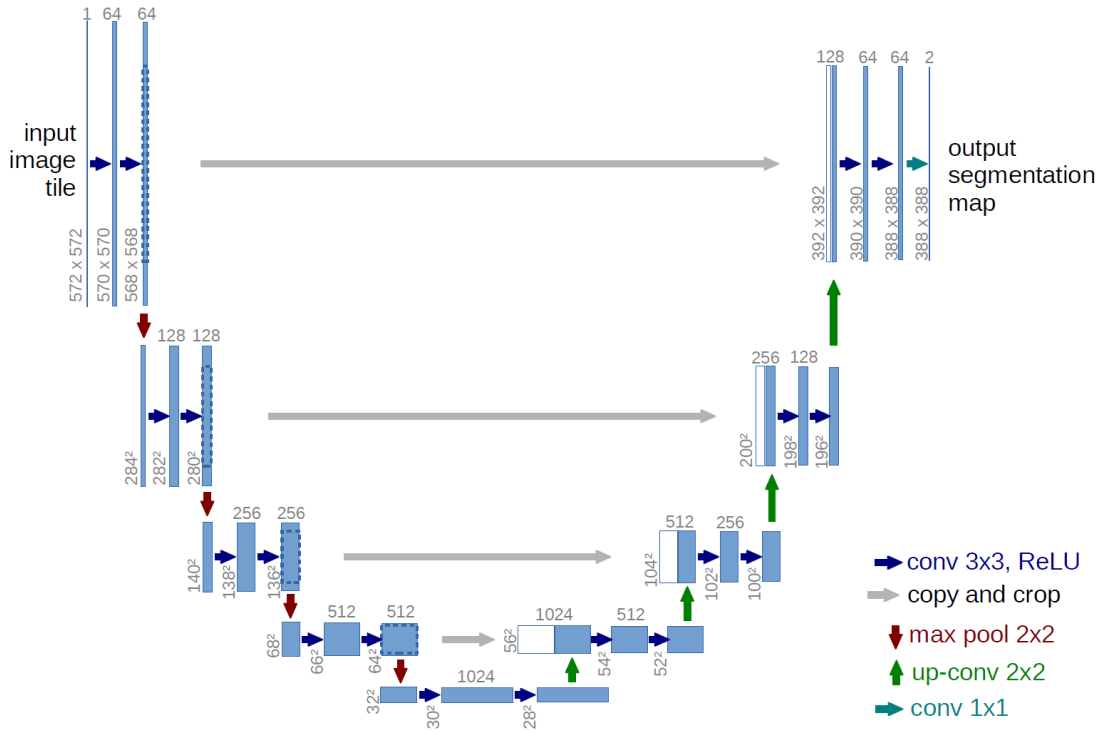


Рис. 1.19 Архітектура U-Net

В основі звужуючого шляху полягають декілька шарів згортки, що використовують у якості функції активації ReLU, і шари підвибірки з максимальним значенням. Даний процес зменшує розміри вхідного зображення і дозволяє виокремлювати найважливіші риси [27].

Шари розширюючого шляху включають у своїй побудові оператори зворотної підвибірки, що розширюють розміри карт ознак, вслід за чим відбувається залучення згорткових шарів для того, аби скоротити кількість каналів ознак. Потім з застосуванням конкатенації система намагається зберегти подробиці первинного зображення та підтвердити сегментаційні маски.

Останній шар системи представляє собою шар згортки з фільтрами розміру 1x1, що скорочує перелік каналів до кількості класів, які визначаються системою. Для прикладу, у випадку, коли модель визначає три класи, останній шар згортки поверне три канали, кожний із яких включатиме сегментаційну маску для того виду дефекту, що визначається ним.

В цілому, система має спроможність опрацьовувати зображення з порівняно високою точністю. Крім того, U-Net застосовує у роботі зв'язки "skip-connections", що надають змогу передавати дані між різними шарами архітектурі, що надає можливість застосовувати дані з різних рівнів роздільної здатності та дає змогу знизити ризики втрати контексту через дрібні деталі.

1.7 Обґрунтування вибору модифікацій U-Net

На даний час існують різні модифікації архітектури U-Net, що мають свої особливі ознаки, переваги та недоліки, які можуть мати певне значення для вирішення того чи іншого завдання. Проведено розгляд різних модифікацій з ціллю визначення найбільш оптимальної опції для вирішення поставленого завдання.

Побудова модифікації U-Net++ включає енкодер і декодер (рис. 1.20, а), що поєднані між собою завдяки групі вкладених блоків густої згорткової мережі. Ключовий концепт даної модифікації зводиться до скорочення семантичної

відстані між картами ознак енкодера та декодера перед злиттям [28]. Для прикладу, семантичну відстань між ($X^{0,0}$, $X^{0,4}$) було скорочено завдяку залученню блоку густої згорткової мережі з трьома шарами згортки. На схемах чорним забарвленням відображено первинну архітектуру U-Net, блакитним та зеленим забарвленням продемонстровано блоки густої згорткової мережі на шляхах зв'язку, а червоним забарвленням позначено глибоке керування. Відповідно, блакитні, зелені та червоні складові відмежовують структуру модифікації від оригінальної архітектури. Більш ретельний аналіз першого шляху зв'язку описано на рис. 1.20, b. У випадку тренування з глибоким керуванням модифікацію можна обрізати під час виводу (рис. 1.20, c).

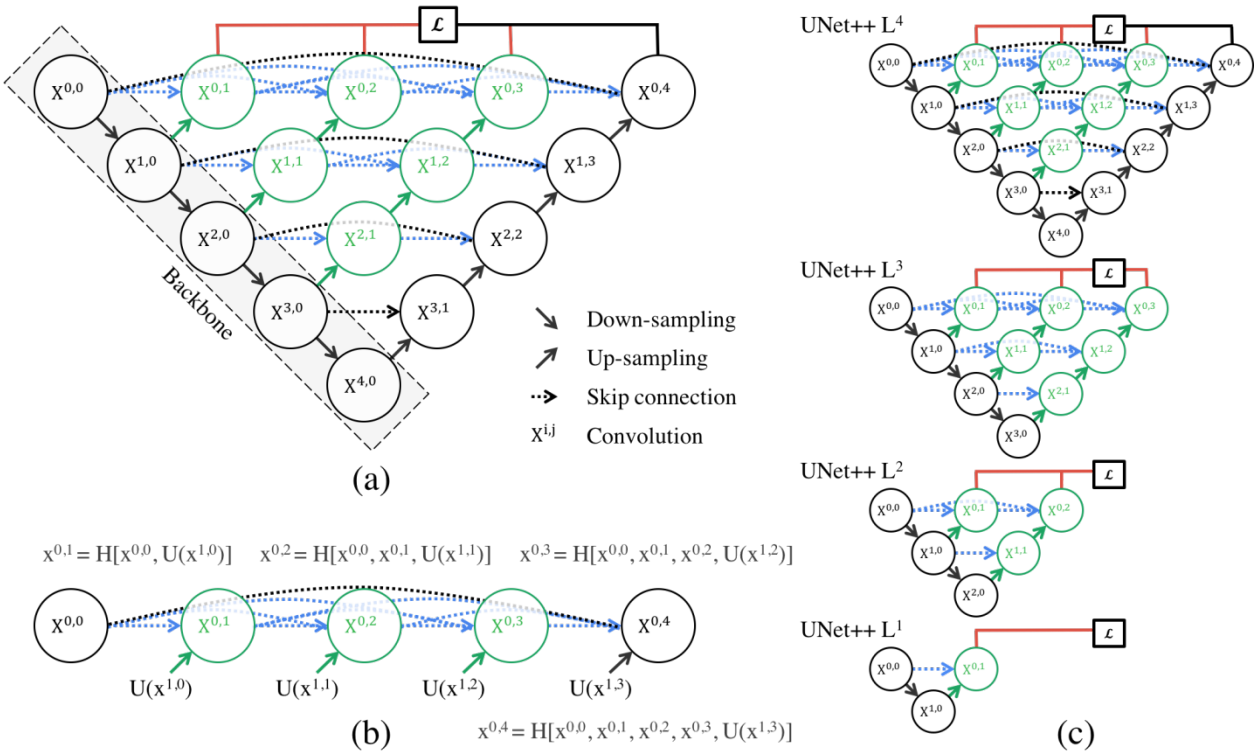


Рис. 1.20 Архітектура U-Net++: a – енкодер та декодер, b – покроковий розгляд першого шляху, c – обрізана частина UNet++

Аналіз структури DoubleU-Net наведено на рисунку 1.21. Як продемонстровано на схемі, модифікація починається з VGG-19 як підмережа енкодера, вслід за якою йде підмережа декодера. В першій мережі, позначеній як

NETWORK 1, головну відмінність модифікації від оригінальної архітектури складає застосування VGG-19, виділеного жовтим забарвленням, шару атрибутної пулінгової мережі, виділеного блакитним забарвленням, та блоку декодера, виділеного світло-зеленим забарвленням. Блок squeeze-and-excite застосовується в енкодері першої мережі та у блоках декодера першої мережі та другої мережі, позначеної як NETWORK 2. Між вихідним та вхідним рівнями першої мережі застосовується елементне множення. Відмінність між модифікацією та оригінальною архітектурою у побудові другої мережі виражається тільки в залученні шару атрибутної пулінгової мережі та блоку squeeze-and-excite, у той час як інші складові не зазнають змін [29].

В першій мережі первинне зображення надходить до модифікованої U-Net, що створює передбачену маску, яка позначається як Output1. Після цього дана маска перемножується з первинним зображенням, граючи роль вхідних даних до другої модифікованої U-Net, яка у свою чергу створює другу маску, що позначається як Output2. Як результат, над обома масками відбувається конкатенація з метою продемонструвати ефективні відмінності між проміжним передбаченням та остаточним.

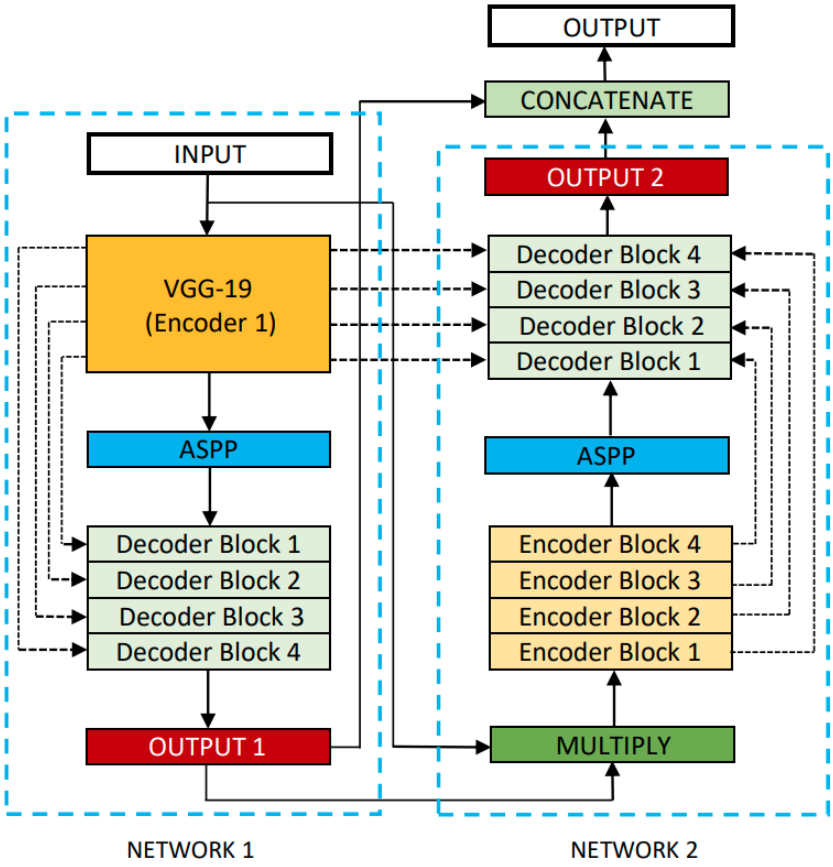


Рис. 1.21 Архітектура DoubleU-Net

Основною особливістю модифікації MultiResUNet (рис. 1.22) є застосування замість послідовності з двох шару згортки блоку MultiRes. Для кожного з таких блоків визначається показник W, мета якого полягає у контролюванні кількості фільтрів шарів згортки, що складають побудову даного блоку. Для збереження порівнянного співвідношення між кількістю показників у оригінальній архітектурі та даній модифікації, розрахування W відбувається шляхом помноження кількості фільтрів U, що включає в себе певний шар, на коефіцієнт масштабування α. Таким чином отримано ефективний метод як контролювання кількості показників, так і збереження їх порівнянними з оригінальною архітектурою. Значення α становить 1,67 у зв'язку з тим, що це надає можливість зберегти кількість параметрів у системі декілька меншою, ніж в оригінальній архітектурі. Схожим з нею чином, вслід за кожним проведеним об'єднанням чи зворотною згорткою показник W подвоюється. Разом із застосуванням блоків MultiRes, відбувається заміщення класичних коротких з'єднань на шляхи Residual. Через це відбувається використання певних операцій згортки до карт ознак, які

передаються від рівня енкодеру до рівня декодеру. Разом з цим скорочується кількість згорткових блоків, які застосовуються на Residual шляхах [30].

Усі шари згортки за винятком останнього, які застосовуються у даній модифікації, використовують у якості функції активації ReLU. Схожим до оригінальної архітектури чином, останній шар застосовує сигмоїдну логістичну функцію.

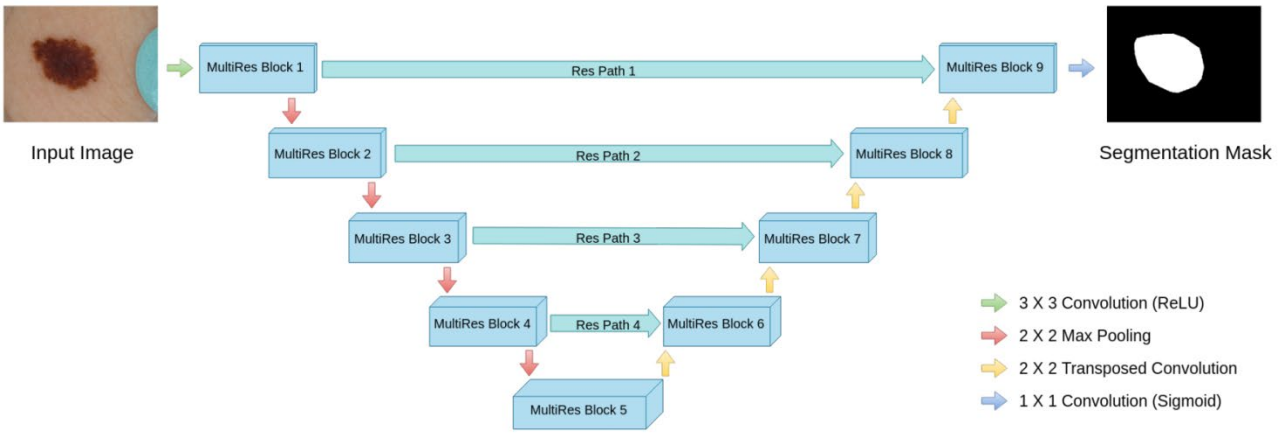


Рис. 1.22 Архітектура MultiResUNet

Кожна з проаналізованих модифікацій U-Net має перспективи для ефективного застосування у різних контекстах. Тим не менш, на даний момент часу немає наявних досліджень, згідно з якими дана архітектура була б використана для виявлення та класифікації дефектів на металевих поверхнях. У зв’язку з цим, є доцільним здійснити наступні дослідження, застосовуючи оригінальну архітектуру з ціллю встановити точність її роботи щодо поставленої мети, так як базова модель має можливість надати потрібну достовірність. Тим не менш, за потреби вдосконалення результатів, є доцільним розгляд наведених модифікацій для подальшого підвищення ефективності роботи системи. Даний підхід надає змогу максимальним чином оцінити та використати можливості оригінальної архітектури та у перспективі вдосконалити її результати щодо точного виявлення та класифікації дефектів на металевих поверхнях.

1.8 Обґрунтування використання основних блоків

Для розробки нейронної мережі було вибрано оригінальну архітектуру U-Net. Було проаналізовано її базову побудову, однак для вдосконалення функціонування нейронної мережі можливо застосування основних блоків. Основний блок (англ. backbone) – це основна частина структури нейронної мережі, що приймає певні вхідні дані та здійснює певні операції, аби виокремити з них корисну інформацію. Після цього відбувається передача вилучених ознак до інших частин системи для подальшого опрацювання та вирішення завдання. Основний блок представляє собою вагий аспект нейронної мережі, що сприяє забезпеченню її точного функціонування.

Побудова глибокої згорткової нейронної мережі, найменованої VGG (Visual Geometry Group) була запропонована для виконання завдань з класифікації великомасштабних зображень (рис. 1.23). Розміри згорткових фільтрів було фіксовано на рівні 3x3 для кожного з шарів, аби надати можливість поглибленої реалізації з одночасним підвищенням не лінійності функцій для аналізу комплексних графічних даних. Вслід за загортковими шарами вихідні дані було об’єднано перед поєднанням із трьома цілком поєднаними шарами 4096-D, результатом чого є значний перелік параметрів для навчання [31].

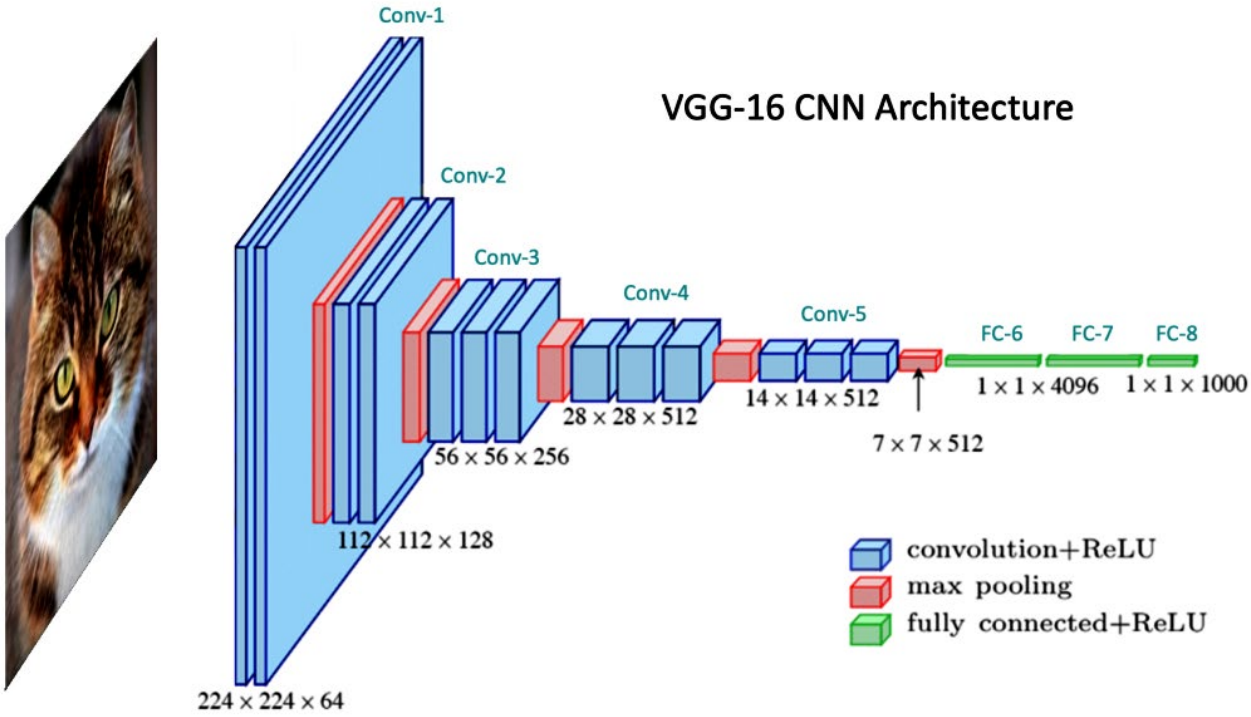


Рис. 1.23 Архітектура VGG-16

Авторами праці «Densely Connected Convolutional Networks» було запропоновано проектування глибокої архітектури шляхом поєднання кожного зв'язку шару згортки з усіма подальшими шарами, утворюючи щільно з'єднану мережу: DenseNet (рис. 1.24). Особливістю даної побудови є те, що карти ознак від попередніх поєднаних шарів використовуються у подальших шарах, що скорочує кількість навчальних параметрів. Зокрема, завдяки залишковим зв'язкам проміж шарами, що переводять вихід попереднього шару до наступного та надають доступо до градієнтів з кожного шару протягом зворотнього поширення помилки, відбувається розв'язання проблеми затухаючих градієнтів. Дана система несе набагато меншу кількість параметрів у зрівнянні з ResNet однакового рівня глибини, однак не втрачає значної ємності для тренування при поглибленні моделі [32].

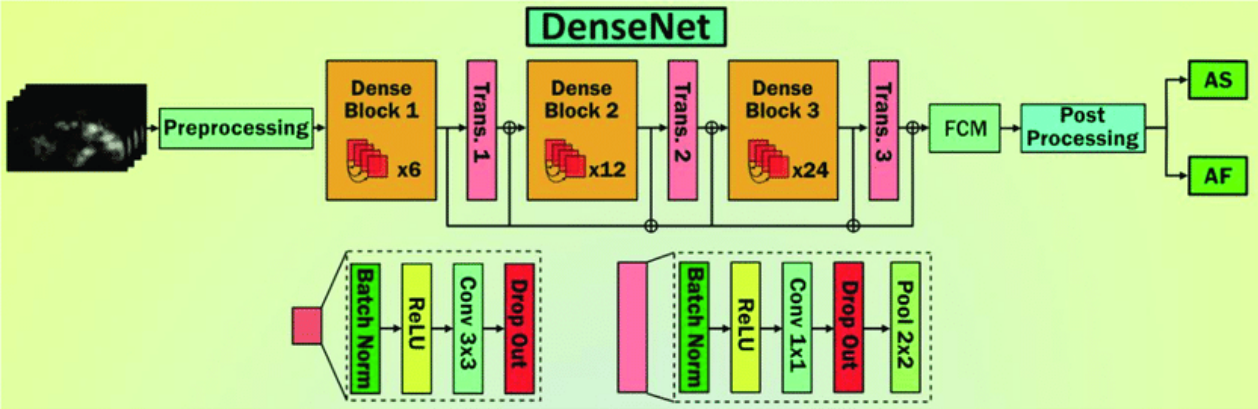


Рис. 1.24 Архітектура DenseNet

Здебільшого у ролі основного блоку для архітектури U-Net використовується побудова ResNet (Рис. 1.25). Головне новаторство даної архітектури полягає у її особливому підході до розв'язання проблеми зникнення градієнта. Аби подолати дану проблему, структура використовує метод залишкового навчання. Даний метод заключається у тому, що системи навчається передбачати залишки, або різницю між входом та виходом, а не сам вихід, що здобувається завдяки застосуванню з'єднань швидкого доступу (skip connections), які надають змогу обходити вхідні дані одним чи декількома рівнями та прямим чином під'єднуватися до вихідних даних. Описані з'єднання надають градієнтам можливість простіше проходити через систему, розв'язуючи проблему зникнення градієнтів та даючи змогу тренувати значно більш глибокі мережі з більшою ефективністю [33].

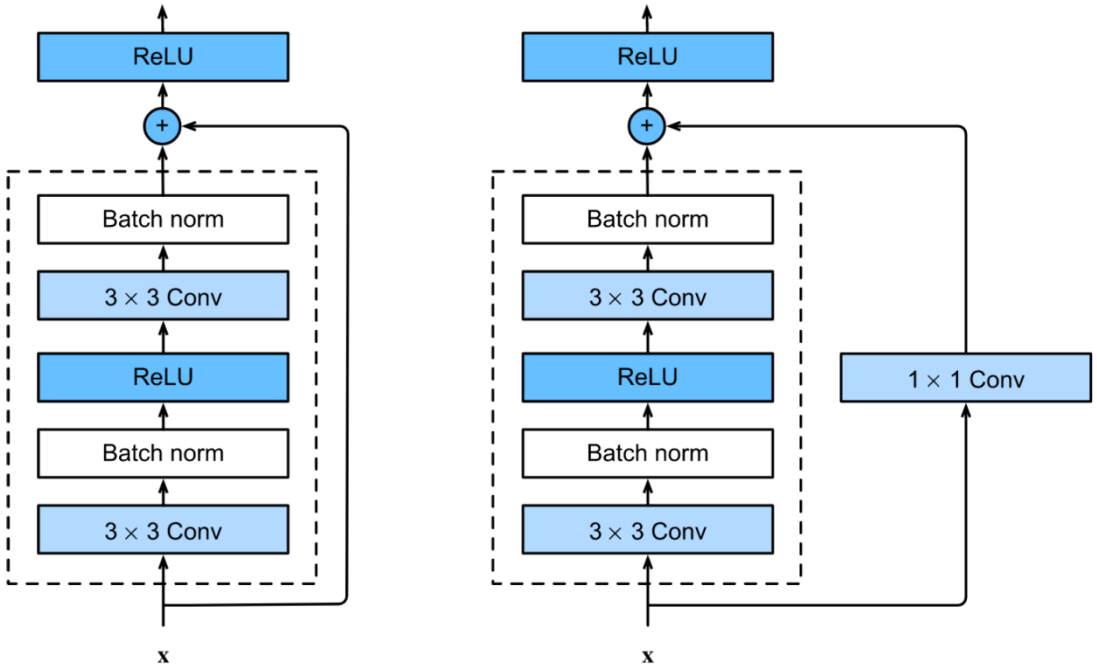


Рис. 1.25. Архітектура ResNet

У даній роботі застосовано саме даний основний блок через його більшу ефективність за вже розглянуті варіанти.

У даному випадку, структура звужуючого шляху моделі U-Net матиме побудову, рівнозначну до структури ResNet, у той час як розширюючий шлях представлятиме її обернену копію. Математична потужність системи може мінятися залежно від комплексності основного блоку нейронної мережі.

2 РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЇ ДЕФЕКТІВ НА МЕТАЛЕВИХ ПОВЕРХНЯХ

2.1 Вибір метрик сегментації

Задача сегментації потребує застосування особливих метрик, що мають враховувати як точність утворених масок дефектів, так і вірогідність розподілу за класами.

Традиційна метрика accuracy демонструє частину вірно класифікованих пікселів стосовно загального числа пік селів у наборі даних. Для прикладу, у випаду, якщо система достовірно класифікує 80 пікселів зі 100, то показник accuracy матиме значення 80%. Основним недоліком даної метрики є низька достовірність у роботі з незбалансованими класами об’єктів.

$$accuracy = \frac{true_{positive}+true_{negative}}{true_{positive}+true_{negative}+ false_{positive}+false_{negative}}, \quad (2.1)$$

де $true_{positive}$ – істинно-позитивний результат;

$true_{negative}$ – істинно-негативний результат;

$false_{positive}$ – помилково-позитивний результат;

$false_{negative}$ – помилково-негативний результат [34].

Оцінки, що мають більш високу достовірність, можна розрахувати за допомогою метрик, що є заснованими на оцінці помилок першого та другого роду. Наприклад, метрика під назвою precision показує частину вірно класифікованих пікселів за відношенням до усіх пікселів, що було призначено системою до дефектного класу. Для прикладу, у випадку, якщо система класифікує 100 пікселів як дефектні, та 70 із них справді є дефектними, а інші 30 було визначено помилково, то precision матиме значення 70%.

$$precision = \frac{true_{positive}}{true_{positive}+false_{positive}}, \quad (2.2)$$

де $true_{positive}$ – істинно-позитивний результат;

$false_{positive}$ – помилково-позитивний результат [34].

Recall – найменування метрики, що розраховує, яка частина дефектних пікселів була виявлена системою відносно до всього числа пікселів, які справді є дефектними. Оцінка recall полягає у тому, з якою ефективністю система розшукує дефектні пікселі.

$$recall = \frac{true_{positive}}{true_{positive} + false_{negative}}, \tag{2.3}$$

де $true_{positive}$ – істинно-позитивний результат;

$false_{negative}$ – помилково-негативний результат [34].

Для кінцевої оцінки ефективності класифікації здебільшого застосовується коефіцієнт Дайса. Дана метрика надає можливість проаналізувати збалансованість між показниками precision і recall. Коефіцієнт Дайса набуває максимального значення, коли дані показники мають однаково високе значення. Це демонструватиме, що система з однаковою ефективністю розшукує і класифікує дефектні пікселі. Розрахунок цієї метрики відбувається за формулою:

$$DC = \frac{2(precision * recall)}{(precision + recall)}, \tag{2.4}$$

де $precision$ та $recall$ – показники відповідних метрик [35].

У випадку застосування коефіцієнту Дайса для аналізу ефективності сегментації формула для обчислення приймає наступний вигляд:

$$DC = \frac{2 * true_{positive}}{2 * true_{positive} + 2 * false_{positive} + 2 * false_{negative}} \tag{2.5}$$

де $true_{positive}$ – істинно-позитивний результат;

$true_{negative}$ – істинно-негативний результат;

$false_{positive}$ – помилково-позитивний результат;

$false_{negative}$ – помилково-негативний результат [35].

Показник коефіцієнту Дайса може мати значення від 0 де 1, де 0 свідчить про абсолютну відсутність спільних елементів, а 1 свідчить про абсолютну збіжність між наведеними вибірками.

Функція втрат представляє собою вагомий елемент тренування системи: протягом навчання моделі, дана функція розраховує втрату системи – математичний показник того, наскільки хибною є відповідь, що була визначена

системою. На базі втрати здійснюється зворотнє поширення помилки, і утворений градієнт застосовується для внесення змін у параметри системи. Окрім того, функція втрат застосовується для того, аби встановити точність роботи системи на навчальних та валідаційних даних як протягом тренування, так і після нього. Таким чином, визначення функції втрат є одним з вагомих кроків для результативного навчання нейронної мережі.

При навчанні з вчителем, функції втрат набувають вигляду порівняння результату, що був визначений системою, із очікуваною відповіддю для встановлення розбіжності між ними, та відповідним чином показнику похибки системи. У випаду, коли система надає відповіді, що є схожими до очікуваних, прийнято вважати, що система спроможна до вирішення завдання.

У якості функції втрат для даної моделі було обрано бінарну перехресну ентропію.

При розрахуванні перехресної ентропії кожна відбувається порівняння кожної із спрогнозованих класових ймовірностей з дійсним показником класу 0 або 1, і обчислюється втрата, що штрафує спрогнозовану ймовірність, базуючись на тому, наскільки значною мірою вона далеко від істинного показника. У своїй суті наведена функція втрат є логарифмічною, що результує у значний штраф для високих розбіжностей, що наближаються за значенням до 1, і невеликий штраф для незначних розбіжностей, що наближаються за значенням до 0. Вона застосовується для корекції ваг системи протягом тренування. Мета зводиться до того, аби оптимізувати функцію втрати, тобто знизити втрати максимально можливим чином, так як мінімальні втрати призводять до більш ефективної системи. Бездоганна система має перехресну ентропію, чий показник дорівнює 0. У дійсності на нікотрому навчальному наборі даних для дійсно існуючого завдання нереально досягти даного показника, тому більш точне означення – бездоганна система має перехресну ентропію, чий показник близький до 0. Обчислення бінарної перехресної ентропії здебільшого відбувається як розрахунок середньої перехресної ентропії для всіх прикладів даних.

Наведені дані стосовно перехресної ентропії осягають завдання класифікації зображень. Однак завдання семантичної сегментації зображень представляє собою завдання попіксельної класифікації. Інакше кажучи, аби спрогнозувати для зображення одну класову мітку, відбувається прогнозування даної класової мітки для кожного із пікселів наведеного зображення.

За цією причиною для завдання семантичної сегментації зображень розрахунок функції втрат бінарної перехресної ентропії набуває даного вигляду:

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}\frac{1}{M}\sum_{i=1}^{M}\left[t_{ji}\log\left(p_{ji}\right)+\left(1-t_{ji}\right)\log\left(1-p_{ji}\right)\right]\right], \quad (2.6)$$

де N – кількість зображень вибірки;

M – кількість пікселів зображення;

t_{ji} – істинне значення;

p_{ji} – передбачена ймовірність для *i*-го пікселя *j*-го зображення [36].

2.2 Опис навчального набору даних

Для тренування системи було застосовано набір зображень, що було створено з використанням високочастотних фотокамер та промарковано за участі фахівця. Металева поверхня на кожному із використаних зображень може зовсім не нести дефектів, нести дефекти одного конкретного класу чи дефекти, що належать до кількох різних класів. Загалом набір зображень включає можливість присутності чотирьох різних класів дефектів: відколів, одиночних вертикальних тріщин, кількох вертикальних тріщин та великих плям. Маска для кожного із видів пошкоджень закодована в один рядок, навіть у випадку, коли на поверхні наявно кілька дефектних областей, що не є спорідненими. Набір даних включає 12568 зображень для тренування і 5506 зображень для валідації. Роздільна здатність кожного з зображень дорівнює 128x800 пікселів.

На рисунку 2.1 для прикладу наведено зображення з навчальної вибірки даних, що містить дефекти, які належать до третього та четвертого класу та відповідно позначені фіолетовим та червоним забарвленням.

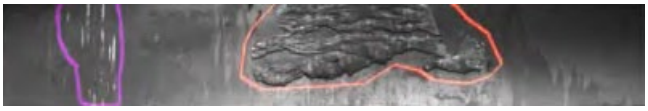


Рис. 2.1 Приклад зображення з тренувального набору даних

Рисунок 2.2 містить гістограми, що наочно демонструють класовий розподіл дефектів у тренувальній вибірці. Розглянувши дані гістограми, можна зазначити, що класи дефектів не є збалансованими. Більша частина зображень, що включають дефекти, містять пошкодження третього класу (77.3%), і лише 3.7% включають дефекти другого класу. Спричиненість даної відсутності збалансованості класів полягає у тому, що протягом виробничих процесів одні типи пошкоджень утворюються порівняно часто, а інші протилежним чином, порівняно рідко. За цією причиною не наявна змога здійснити більшу кількість фотографій поверхонь, що містять, для прикладу, дефекти другого класу.

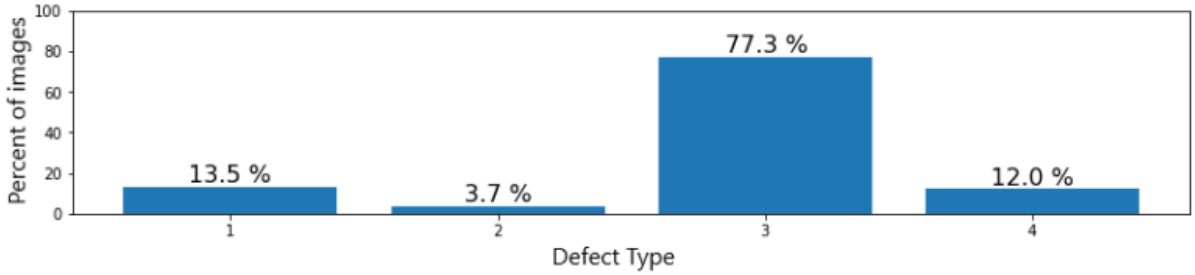


Рис. 2.2 Розподіл класів дефектів

2.3 Математична модель нейронної мережі

Штучний (формальний) нейрон, також відомий як негроподібний елемент, представляє собою простий обчислювальний пристрій, який включає у своїй побудові декілька входів та один вихід та являє собою базовий елемент нейронних мереж. Побудову формального нейрона розглянуто на рисунку 2.3.

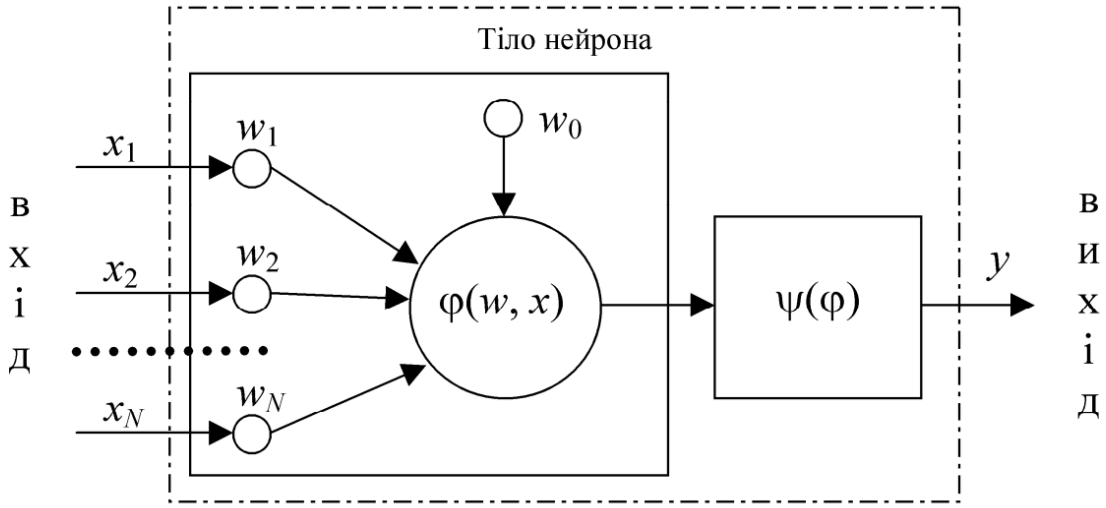


Рис. 2.3 Структура штучного нейрона

Вхід нейроподібного елемента приймає вхідний вектор, що представляє собою набір вхідних сигналів $x = \{x_j\}, j = 1, 2, ..., N$, де N – число входів.

Кожен із вхідних сигналів піддається зважуванню відносно зіставленої йому ваги зв'язку w_j, що здійснює моделювання перетворення сигналу у міжнейронному контакті.

Завдання дискримінантної, або вагової функції нейроподібного елемента φ полягає у поєднанні зважених вхідних сигналів для одержання постсинаптичного потенціалу та передачі його показника до функції активації ψ, що в свою чергу розраховує скалярне значення, яке передається на виході нейроподібного елемента.

Таким чином, загальна математична модель роботи нейроподібного елемента виражається наступним виглядом:

$$y = \psi(\varphi(w, x)), \tag{2.7}$$

де x – це вектор вхідних аргументів;

y – значення на виході нейрона;

ψ – функція активації;

φ – дискримінантна функція;

$w = \{w_j\}$ – вектор, який включає значення вагових коефіцієнтів w_j і значення зсуву w_0 [37].

Набір вагових коефіцієнтів нейроподібного елемента w здійснює моделювання його пам'яті. За цією причиною нейроподібні елементи можна сприймати у якості запам’ятовуючих пристроїв. Також нейроподібні елементи можна сприймати як прості процесори, які виконують розрахунок показника функції активації на основі показника дискримінаційної функції вхідних даних та ваг.

Здебільшого для роботи нейронних мереж застосовуються описані далі дискримінаційні функції.

Дискримінаційна функція зваженої суми виражається наступним чином:

$$\varphi(w,x) = \sum_{j=1}^{N} w_j x_j + w_0, \tag{2.8}$$

де x_j – вхідний сигнал;

w_j – значення вагових коефіцієнтів;

w_0 – значення зсуву;

N – число входів [37].

Дискримінаційна функція зваженого добутку виражається наступним чином:

$$\varphi(w,x) = \prod_{j=1}^{N} w_j x_j = w_0 \prod_{j=1}^{N} x_j, \tag{2.9}$$

де x_j – вхідний сигнал;

w_j – значення вагових коефіцієнтів;

w_0 – значення зсуву;

N – число входів [37].

Дискримінаційна функція евклідової відстані виражається наступним чином:

$$\varphi(w,x) = \sum_{j=1}^{N} (w_j - x_j)^2, \tag{2.10}$$

де x_j – вхідний сигнал;

w_j – значення вагових коефіцієнтів;

N – число входів [37].

Випрямлена лінійна одиниця (англ. rectified linear unit, скорочено ReLU) представляє собою функцію активації, що набуває високої популярності

використання у сучасних нейронних мережах. Застосування даної функції активації обґрунтовано тим, що обробка даних та навчання моделі з її використанням відбуваються у кілька разів швидше порівняно з рештою функцій активації, разом з тим утворюючи не лінійність. Даний фактор разом з підвищенням швидкості тренування моделі знижує ризики перенавчання у зв'язку з тим, що системи навчається більш комплексним зв'язкам за однаковий проміжок часу відносно інших функцій активації, що зі свого боку надає можливість навчати більш комплексні системи на тих самих даних. У математичному представленні випрямлена лінійна одиниця зображається наступною формулою:

$$\psi(\varphi) = \max(0, \varphi), \tag{2.11}$$

де φ – дискримінантна функція [37].

Базуючись на формулі наведеної функції, вона позбувається негативних значень, призначаючи їм нульове значення, взамін здійснюючи тотожне перетворення для інших значень. З точки зор потужності швидкодія випрямленої лінійної одиниці вдосконалюється і тим, що за своює суттю вона представляє собою просте порівняння вхідного значення з нулем. Функціонування алгоритму зворотнього поширення помилки також полегшується, так як значення похідної випрямленої лінійної одиниці представляє 0 для від'ємних вхідних даних та 1 для усіх інших.

У завданнях багатокласової класифікації на кінцевому шарі моделі здебільшого застосовується функція активації Softmax (нормована експоненційна функція), у зв'язку з тим що дана функція враховує усі значення із минулого шару системи та надає змогу обчислити відповідні ймовірності для кожного із класів. Дану функцію було використано для формування результатів у даній роботі. Формула нормованої експоненційної функції виглядає наступним чином:

$$\psi(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}, \tag{2.12}$$

де z_j – значення вагової суми вхідного елемента;

k – число класів, для яких встановлюються ймовірності;

e – число Ейлера [37].

Згорткові операції здійснюються у згорткових шарах та власне представляють собою ключовий аспект згорткових нейронних мереж. Згортка являє собою математичну операцію, що завдяки дії згорткового ядра k (англ. feature detector чи kernel) на первинне зображення у вигляді тензора x здобуває у якості результату карту ознак m (англ. feature map). В своїй структурі кожен з детекторів ознак складається із певного встановленого числа фільтрій, кожен з яких зі свого боку представляє собою тензор параметрів з розмірністю, що зазвичай надає змогу покривати тільки незначну область зображення за шириною та висотою, однак у той же час обов'язковим чином повинна відповідати розмірності за глибиною. Дані фільтри власне представляють собою параметричну частину нейронної мережі, для якої здійснюється навчання. Крім того, фільтри можуть бути застосовані не тільки для детектування рис та переходів, а і в якості інструментів для підвищення показників різкості або розмиття.

Також, безпосереднє число фільтрів може відігравати роль одного з гіперпараметрів моделі, так як підвищення числа фільтрів призводить до підвищення числа потенційно детектованих рис і закономірностей. Однак у той же час надмірна кількість фільтрів може результувати ризиками повторень та виявлення рис, що не належать до визначальних, тобто ускладненням і потенційним перенавчанням нейронної мережі. Отже, число фільтрів відповідає числу карт ознак, що встановлює рівень глибини вихідних даних, а разом з цим і рівень глибини вхідних даних для подальшого шару моделі, відповідним чином встановлюючи число необхідних параметрів і на подальшому кроці. Протягом навчання фільтри здобувають можливісти встановлювати риси зображення, що утворюють собою карти ознак. Кожен із фільтрів відповідає за утворення однієї карти ознак. Карта ознак утворюється за причиною того, що фільтр пересувається по площі зображення та встановлює риси для сусідних рецептивних полів. Крок пересування s (англ. stride) може різнитися та представляє собою один із параметрів нейронної мережі. Отже, хоча абсолютне місцеположення кожної із

рис зображення не зберігається на карті ознак, це не має певного значення за причиною того, що відносне місцеположення рис стосовно одна одної залишається сталим, і як підсумок, зберігаються риси зображення.

В математичному представленні операція згортки може бути відображена наступною формулою:

$$\mathrm{M}(i,j) = (K * X)(i,j) = \sum_m \sum_n K(m,n) X(i-m, j-n), \tag{2.13}$$

де $M(i,j)$ – елемент карти ознак з координатами i та j;

X – вхідне зображення;

K – детектор ознак;

m, n – розмірності детектора ознак [38].

Шари субдискретизації чи об'єднувальні шари (англ. pooling layers) представляють собою один із ключових аспектів побудови згорткових нейронних мереж. Ключовою метою шарів субдискретизації є зменшення розмірності даних разом зі збереженням найбільш релевантних ознак шляхом утворення залежності між декількома елементами (нейронами) з минулого шару з єдиним елементом наявного шару. За даною причиною при розробці моделі об'єднувальні шари здебільшого застосовуються із деякою періодичністю між шарами згортки. Існує два підтипи шарів субдискретизації: усереднювальні (англ. average) і максимізаційні (англ. maximal). Відмінність між ними полягає у тому, що усереднювальні шари призначають кожному із нейронів власне середнє значення серед тих, що включає відповідна йому група елементів із минулого шару, у той час як максимізаційні призначають максимальне значення.

Математично операція усередненого об'єднання формулюється наступним чином:

$$p(i,j) = \frac{\sum_{m,n} x(i-m, j-n)}{m*n}, \tag{2.14}$$

де де $p(i,j)$ – значення елементу поточного рівня з координатами i та j;

x – вхідні дані з попередніх рівнів;

m, n – розмірність рецептивного поля [38].

У той час як операція максимизаційного об'єднання формулюється наступним чином:

$$p(i,j) = max_{i,j}\big(x(i-m, j-n)\big), \tag{2.15}$$

де $p(i,j)$ – значення елементу поточного рівня з координатами i та j;

x – вхідні дані з попередніх рівнів;

m, n – розмірність рецептивного поля [38].

Значна кількість алгоритмів машинного навчання здійснюють спроби пошуку закономірностей в даних через порівняння характеристик точок даних. Але в ході даного процесу утворюється проблема, коли властивості значним чином відрізнаються за масштабами. Для прикладу, розглянемо набір даних про будинки. Двома можливими характеристиками можна вважати число кімнату у будинку та загальну тривалість його існування у роках. Нейронна мережа може зробити спробу спрогнозувати, який із будинків може підійти вам найкращим чином. Однак у процесі порівняння моделлю точок даних, об’єкт, що має великий масштаб, абсолютним чином домінує над іншим. У випадку, коли дані приймають настільки стислий вигляд, це утворює проблему. Нейронна мережа повинна усвідомлювати існування кардинальної різниці між будинком, що включає 5 кімнат, та будинком, що включає 500 кімнат. Однак у даний момент часу за причиною того, що дані будинки можуть мати відмінність у віці, що складає 90 років один від одного, різниця у числі кімнат здійснює менш вагомий вплив на глобальну різницю. У тому, аби встановити для всіх точок даних збіжність за масштабом та зробити так, щоб кожна із властивостей несла однакову важливість, полягає завдання нормалізації.

Батч-нормалізація – це засіб прискорення машинного навчання. Даний засіб дозволяє розв’язати проблему, що негативним чином впливає на тренування нейронних мереж: з тим, як сигнал поширюється системою, навіть у випадку його нормалізації на вході, при проходженні через внутрішні рівні, даний сигнал несе ризики значного спотворення як за математичним очікуванням, так і за дисперсією, що може спричинити кардинальні незбіжності між градієнтами на різних шарах. Це може стати причиною застосування більш сильних

регуляризаторів, що у свою чергу може негативним чином вплинути на швидкість навчання.

Розв’язання наведеної проблеми за використанням батч-нормалізації зводиться до того, аби нормалізувати дані на вході таким чином, аби математичне очікування мало значення 0, а дисперсія – 1. Проведення нормалізації відбувається перед входом до кожного шару. Тобто, протягом навчання відбувається нормалізація batch_size прикладів, а протягом тестування проводиться нормалізація статистики, яка одержується на базі усієї тренувальної множини, так як завчасний розгляд тестових даних не є можливим. А саме, для деякого батча (пакету) розрахунок математичного очікування відбувається наступним чином:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{2.16}$$

де B = $x_1, \dots, x_m$ – пакет даних, що піддається обробці [39].

У той час як обчислення дисперсії формулюватиметься наступним чином:

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B) \tag{2.17}$$

де B = $x_1, \dots, x_m$ – пакет даних, що піддається обробці;

μ_B – обчислене математичне очікування [39].

Завдяки даним статистичним показникам ми здійснюємо перетворення функції активації таким чином, аби протягом усього пакету її математичне очікування дорівнювало 0, а дисперсія – 1:

$$x_l = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \tag{2.18}$$

де μ_B – обчислене математичне очікування;

σ_B^2 – обчислена дисперсія;

ε > 0 – параметр, який захищає операцію від поділу на 0 [39].

Зрештою, для одержання кінцевої функції активації *y*, необхідно впевнитися, що протягом процесу нормалізації не було втрачено здатності до узагальнення, можна застосувати довільне масштабування та зсув нормалізованих значень, одержуючи кінцеву функцію активації:

$$y_i = \gamma x_l + \beta, \tag{2.19}$$

де γ, β –параметри батч-нормалізації, що можуть бути навчені системою [39].

2.4 Схематична модель нейронної мережі

Як було постановлено за результатами аналізу архітектур згорткових нейронних мереж, у даній роботі було застосовано архітектуру для семантичної сегментації зображень під найменуванням U-Net. Більшою частиною структура U-Net поділяється на енкодер та декодер. Завдання частини, що представляє собою енкодер, заключається у тому, аби вилучити загальний контекст зображення, з'ясовуючи, що на ньому розташовано, шляхом покрокового зменшення розмірів даного зображення, при цьому збільшуючи його глибину. Даний процес здійснюється завдяки згортковим шарам та шарам максимізаційного об'єднання. Завдання частини, що представляє собою декодер, заключається у тому, аби перекласти дану інформацію у вихідне розташування пікселів через підвищення дискретизації зображення завдяки транспонованим шарам згортки. Дані шари виконують обернений до функціонування енкодеру процес, збільшуючи розміри зображення й у той же час зменшуючи його глибину. Також з метою вдосконалення процесу дискретизації зображення, на кожному етапі енкодера застосовуються з'єднання швидкого доступу, що об'єднують карти ознак енкодера та вихідні дані транспонованих шарів згортки. Дані карти ознак включають первинну просторову інформацію, що було втрачено протягом багатоетапної згортки, надаючи декодеру змогу утворити більш достовірні результати сегментації. З метою підвищення ефективності системи, замість стандартного кодувальника архітектури U-Net для побудови енкодера було застосовано основні блоки архітектури ResNet.

Загальну схему моделі зображено на рисунку 2.4, де частину-енкодер, що складається із блоків ResNet, позначено жовтим забарвленням, та частину-енкодер позначено сіро-зеленим забарвленням.

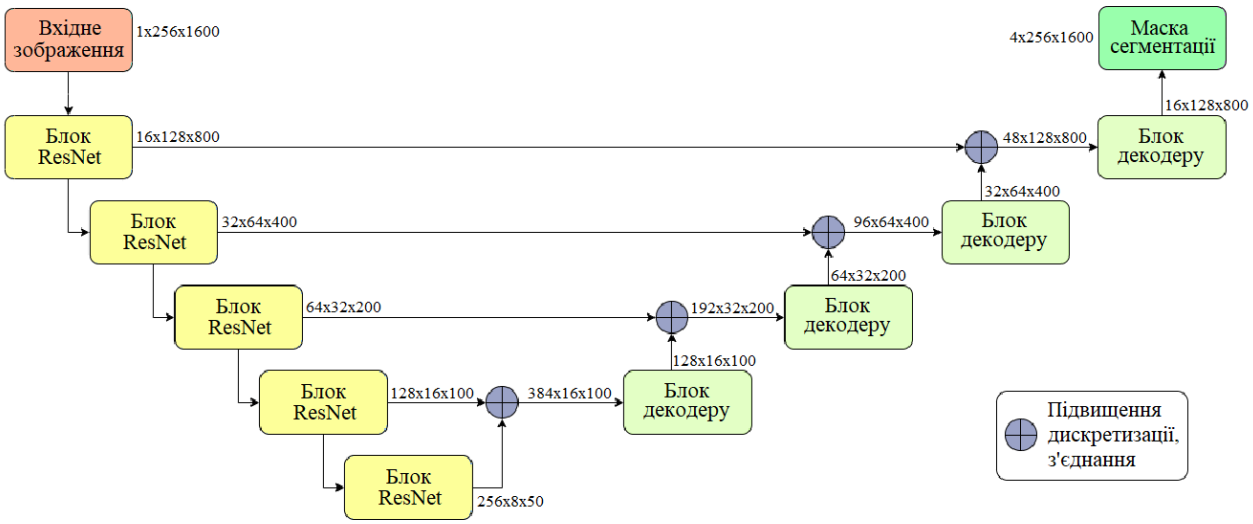


Рис. 2.4 Загальна схема моделі

Стосовно детальної структури енкодеру, подальші експерименти з основними блоками архітектури ResNet продемонстрували, що найбільш ефективним варіантом з можливих є застосування основного блоку ResNet18. Дана побудова представляє собою відносно невелику та компактну структуру ResNet з 18 шарами, включаючи залишкові блоки з двома-трьома згортковими шарами у складі кожного рівня, що відносно швидко піддається навчанню. Схему одиничного блоку ResNet18 наведено на рисунку 2.5.

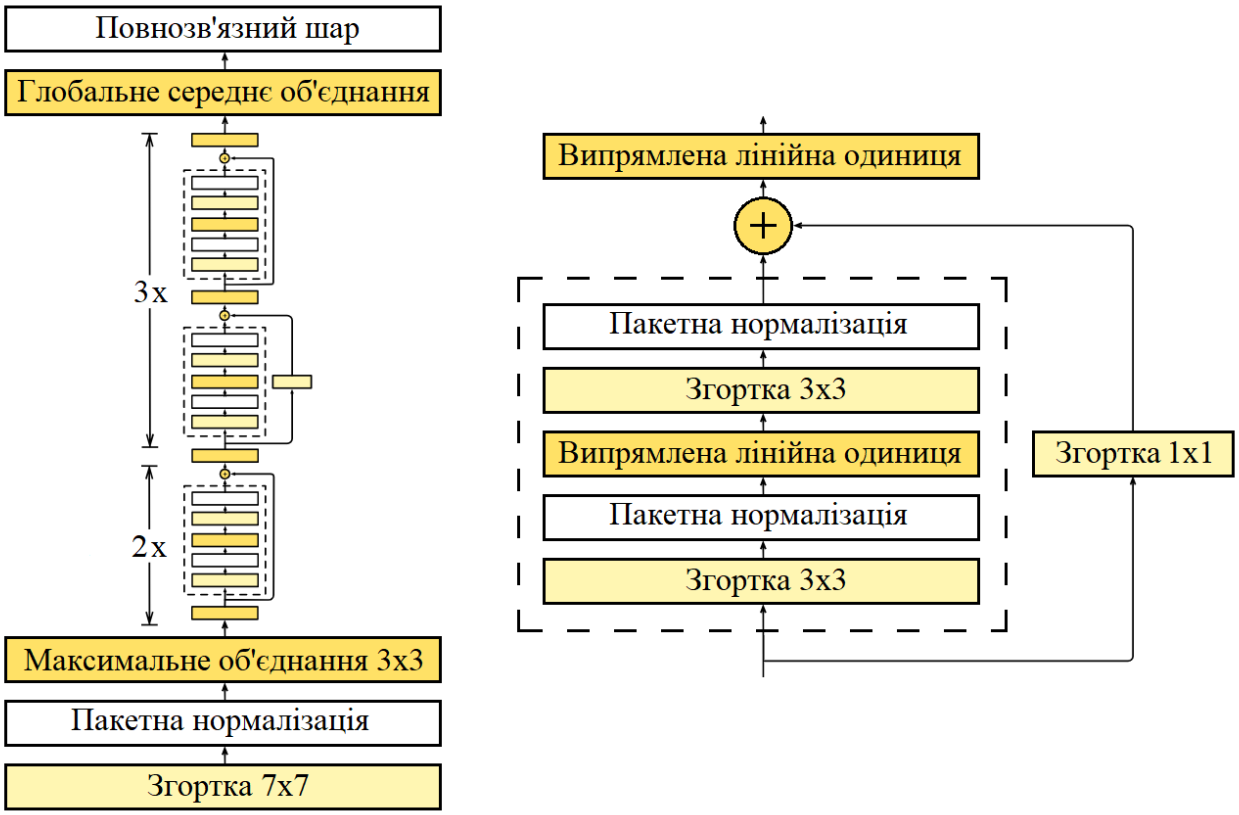


Рис. 2.5. Побудова блоку ResNet18

2.5 Опис розроблених класів

Клас SteelDataset (рис. 2.6) використовується для створення PyTorch Dataset для моделі сегментації зображень. Він містить методи для отримання зображень та відповідних масок, а також можливість використовувати трансформації під час тренування та валідації. Клас SteelDataset містить методи __init__(self, df, data_folder, mean, std, phase), __getitem__(self, idx), __len__(self), get_transforms(phase, mean, std), provider(data_folder, df_path ,phase, mean=None, std=None, batch_size=8, num_workers=4,).

SteelDataset
__init__(self, df, data_folder, mean, std, phase)
__getitem__(self, idx)
__len__(self), get_transforms(phase, mean, std)
provider(data_folder, df_path, phase, mean=None, std=None, batch_size=8, num_workers=4)

Рис. 2.6 Побудова класу SteelDataset

Метод __init__ цього класу виглядає як конструктор (ініціалізатор) і призначений для створення об'єктів цього класу. Загалом, цей конструктор призначений для ініціалізації об'єкта даного класу з визначенням необхідних параметрів та підготовкою даних для подачі моделі з використанням трансформацій. Розглянемо кожний рядок коду:

- self.df = df: Зберігання DataFrame df, який містить інформацію про дані.
- self.root = data_folder: Зберігання шляху до кореневого каталогу даних.
- self.mean = mean: Зберігання середніх значень для нормалізації даних.
- self.std = std: Зберігання стандартних відхилень для нормалізації даних.
- self.phase = phase: Зберігання фази, яка вказує, чи це тренування чи валідація.
- self.transforms = get_transforms(phase, mean, std): Виклик функції get_transforms для отримання об'єкта трансформації зображень.
- self.fnames = self.df.index.tolist(): Створення списку імен файлів, які взяті з індексу DataFrame df.

Метод __getitem__ використовується для отримання зображення та відповідної маски за індексом для використання у тренуванні моделі. Розглянемо його:

- image_id, mask = make_mask(idx, self.df): Виклик функції make_mask, яка генерує маску для зображення за індексом idx. Отримані image_id та mask зберігаються.
- image_path = os.path.join(self.root, "train_images", image_id): Формування повного шляху до зображення на основі кореневого шляху, папки "train_images" та ідентифікатора зображення.
- img = cv2.imread(image_path): Зчитування зображення з файлу за допомогою OpenCV.
- augmented = self.transforms(image=img, mask=mask): Застосування трансформацій до зображення та маски.
- img = augmented['image']: Отримання трансформованого зображення.
- mask = augmented['mask']: Отримання трансформованої маски.

- mask = mask[0].permute(2, 0, 1): Перестановка осей у масці для відповідності порядку осей, що використовується в бібліотеці PyTorch.
- return img, mask: Повернення трансформованого зображення та маски. Це значення буде використане, коли ви викликаєте __getitem__ на об'єкті цього класу.

Метод __len__ цього класу визначає, скільки елементів (зображень) доступно для витягування з цього набору даних. У даному випадку, цей метод повертає довжину списку імен файлів (self.fnames), який був ініціалізований при створенні об'єкта класу.

Метод get_transforms генерує та повертає об'єкт трансформації за вказаними параметрами phase, mean і std. Розглянемо його функціонал:

1. Ініціалізація порожнього списку трансформацій list_transforms.

- Якщо phase рівний "train" (тренування), то додається горизонтальне відображення з ймовірністю 0.5.

2. Додавання трансформацій для обох фаз.

- Нормалізація (Normalize): Використовується середнє значення mean та стандартне відхилення std для нормалізації значень пікселів.
- Перетворення в тензор (ToTensor): Перетворює зображення та маску в тензори PyTorch.

3. Створення композиції трансформацій (list_trfms).

- Compose використовується для об'єднання всіх трансформацій у єдиний об'єкт.

4. Повернення об'єкта трансформації (list_trfms).

- Згенерований об'єкт трансформації повертається для використання під час підготовки даних для моделі.

Метод provider є функцією для створення та повернення об'єкта DataLoader, який використовується для завантаження даних для тренування чи валідації моделі. Розглянемо його функціонал:

1. Читання та підготовка даних:

- Зчитує CSV-файл (df_path) з інформацією про зображення та маски.

- Розділяє колонку 'ImageId_ClassId' на 'ImageId' та 'ClassId'.
- Перетворює широкий формат DataFrame в ширший формат, де кожен рядок представляє зображення, а стовпці відповідають класам дефектів, а також додає стовпець 'defects', який вказує кількість дефектів на кожному зображенні.

2. Розділення даних на тренувальні та валідаційні:

- Використовує функцію train_test_split для розділення даних на тренувальні та валідаційні з наведеними параметрами.
- Вибирає train_df або val_df в залежності від фази ("train" чи "val").

3. Створення об'єкта SteelDataset та DataLoader:

- Створює об'єкт SteelDataset на основі вибраного DataFrame, кореневого каталогу даних, середніх значень та стандартних відхилень, а також фази.
- Створює об'єкт DataLoader для ефективного завантаження та пакетної обробки даних. Використовуються параметри, такі як розмір пакета, кількість робочих потоків та переміщення даних до пам'яті (pin_memory).

4. Повернення об'єкта DataLoader:

- Повертає створений об'єкт DataLoader для використання у тренувальному чи валідаційному циклі моделі.

Клас Meter (рис. 2.7) є інструментом для відстеження та обчислення метрик під час тренування моделі для завдання сегментації зображень. Він дозволяє збирати статистику та відстежувати ефективність моделі на протязі тренування. Клас Meter використовує методи predict(X, threshold) і metric(probability, truth, threshold=0.5, reduction='none') та включає методи __init__(self, phase, epoch), update(self, targets, outputs) і get_metrics(self).

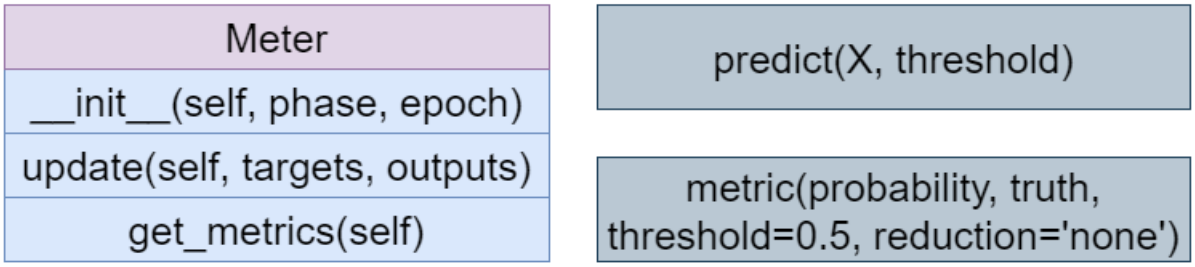


Рис. 2.7 Побудова класу Meter

Метод predict використовується для виконання передбачень на основі виводу softmax моделі. Розглянемо її структуру:

- X_p = np.copy(X): Використовує np.copy() для створення глибокої копії вхідного масиву X.
- preds = (X_p > threshold).astype('uint8'): Генерує масив preds, який містить 0 або 1 в залежності від того, чи перевищує значення в X_p порогове значення threshold. .astype('uint8') використовується для перетворення отриманого булевого масиву на масив цілих чисел з беззнаковим 8-бітовим типом.
- return preds: Повертає результати передбачень.

Функція metric використовується для обчислення метрик, зокрема коефіцієнта Дайса. Проаналізуємо її:

- batch_size = len(truth): Обчислюється розмір партії (batch size) на основі розміру truth.
- with torch.no_grad(): Використовує контекст torch.no_grad(), щоб вимкнути обчислення градієнтів під час оцінки метрик.
- probability = probability.view(batch_size, -1): probability і truth перетворюються в масиви з двома вимірами, де кожен рядок представляє одне зображення, а стовпці відповідають пікселям.
- p = (probability > threshold).float() та t = (truth > 0.5).float(): Створюються бінарні масиви p та t на основі порогових значень для ймовірностей та справжніх значень.
- t_sum = t.sum(-1) та p_sum = p.sum(-1): Обчислюється сума значень для кожного зображення (по останньому вимірі).
- neg_index = torch.nonzero(t_sum == 0) та pos_index = torch.nonzero(t_sum >= 1): Знаходяться індекси негативних та позитивних зображень, відповідно.
- dice_neg = (p_sum == 0).float() та dice_pos = 2 * (p*t).sum(-1)/((p+t).sum(-1)): Обчислюється коефіцієнт Дайса для негативних і позитивних зображень відповідно.

- dice_neg = dice_neg[neg_index] та dice_pos = dice_pos[pos_index]: Вибираються значення коефіцієнта Дайса для негативних та позитивних зображень.
- dice = torch.cat([dice_pos, dice_neg]): Створюється один тензор, що об'єднує коефіцієнти Дайса для позитивних і негативних зображень.
- num_neg = len(neg_index) та num_pos = len(pos_index): Обчислюється кількість негативних та позитивних зображень.
- return dice, dice_neg, dice_pos, num_neg, num_pos: Повертається п'ять значень: загальний коефіцієнт Дайса (dice), коефіцієнт Дайса для негативних зображень (dice_neg), коефіцієнт Дайса для позитивних зображень (dice_pos), кількість негативних зображень (num_neg), і кількість позитивних зображень (num_pos).

Метод __init__ призначений для ініціалізації початкових значень атрибутів для об'єкта класу, що використовуються в інших методах класу під час виконання різних завдань.. Проаналізуємо його:

- self.base_threshold = 0.5: Встановлює атрибут base_threshold об'єкта класу рівним 0.5.
- self.base_dice_scores = [], self.dice_neg_scores = [], і self.dice_pos_scores = []: Встановлюють атрибути base_dice_scores, dice_neg_scores і dice_pos_scores як пусті списки. Ці атрибути призначені для зберігання значень коефіцієнтів Дайса під час виконання обчислень в процесі тренування моделі.

Метод update використовується для зберігання та оновлення показників ефективності моделі під час тренування. Розглянемо його:

- Застосування softmax до виводів моделі: Виводи моделі (outputs) піддаються операції softmax за допомогою функції torch.softmax. dim=1 вказує, що softmax повинен бути застосований по другому виміру, який в даному випадку є виміром класів.
- Обчислення коефіцієнтів Дайса за допомогою функції metric: Виводи моделі після softmax і справжні значення (targets) передаються до функції

metric, яка обчислює коефіцієнти Дайса для позитивних та негативних класів.

- Розширення списків зі значеннями коефіцієнтів Дайса: Значення коефіцієнтів Дайса розширюються і додаються до списків self.base_dice_scores, self.dice_pos_scores і self.dice_neg_scores.
- Використання функції predict для отримання передбачень: Виводи після softmax передаються до функції predict, яка використовує поріг (self.base_threshold) для отримання бінарних передбачень.

Метод get_metrics призначений для обчислення середніх значень метрик, отриманих під час тренування моделі. Розглянемо його:

- Обчислення середнього значення коефіцієнта Дайса: Використовуючи бібліотеку NumPy, обчислюється середнє значення коефіцієнта Дайса для всіх зібраних значень у списку self.base_dice_scores.
- Обчислення середнього значення коефіцієнта Дайса для негативних класів: Аналогічно до попереднього кроку, обчислюється середнє значення коефіцієнта Дайса для негативних класів і зберігається в dice_neg.
- Обчислення середнього значення коефіцієнта Дайса для позитивних класів: Аналогічно до попередніх кроків, обчислюється середнє значення коефіцієнта Дайса для позитивних класів і зберігається в dice_pos.
- Створення списку з отриманих значень: Створюється список dices, який містить середні значення коефіцієнта Дайса для всього, негативних та позитивних класів.
- Повернення створеного списку: Отриманий список dices повертається як результат методу.

Клас Trainer (рис. 2.8) координує процес навчання та відповідає за ведення метрик, зберігання та оновлення моделі, а також виведення інформації про процес тренування. Клас Trainer включає методи __init__(self, model), forward(self, images, targets), iterate(self, epoch, phase) та start(self).

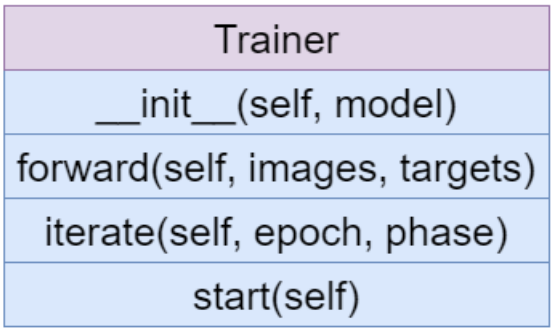


Рис. 2.8 Побудова класу Trainer

Метод __init__ використовується для ініціалізації об'єкта класу з початковими значеннями параметрів та компонентів, які будуть використовуватися під час тренування моделі. Розглянемо його структуру:

- Налаштування параметрів тренування: Задані параметри, такі як кількість робочих потоків (num_workers), розмір пакету (batch_size), кількість накопичення кроків (accumulation_steps), швидкість навчання (lr), кількість епох (num_epochs), та інші.
- Визначення фаз тренування: Визначені фази тренування, які включають "train" і "val" (валідація).
- Визначення пристрою (GPU): Визначений пристрій, на якому буде виконуватися тренування (GPU).
- Ініціалізація моделі, критерію, оптимізатора та планувальника: Ініціалізовані модель, критерій (BCEWithLogitsLoss), оптимізатор (Adam) та планувальник (ReduceLROnPlateau).
- Переміщення моделі на пристрій (GPU): Модель переміщена на визначений пристрій.
- Ініціалізація обробників даних для тренування та валідації: Ініціалізовані обробники даних для кожної фази тренування.
- Ініціалізація списків для втрат та коефіцієнтів Дайса: Створені порожні списки для втрат та коефіцієнтів Дайса для кожної фази тренування.

Метод forward приймає зображення та маски, переміщує їх на пристрій (GPU) та обчислює виводи моделі і втрати. Проаналізуємо його побудову:

- Переміщення зображень та масок на пристрій (GPU): Зображення та маски, передані як аргументи, переміщаються на пристрій, на якому виконується тренування (self.device).
- Отримання виводів моделі: Зображення (images) подаються на вхід моделі (self.net), і отримуються виводи.
- Обчислення втрат за допомогою критерію: З виводами моделі та масками (targets), обчислюється значення втрат за допомогою критерію (self.criterion), який у даному випадку є BCEWithLogitsLoss.
- Повернення втрат та виводів моделі: Повертається значення втрат та виводи моделі. Це дозволяє використовувати цей метод при здійсненні зворотнього поширення та оптимізації в методі навчання (backward та optimizer.step()).

Метод iterate викликається для кожної епохи ітерації. Він обробляє кожен батч, використовуючи метод forward, і оновлює метрики та втрати за допомогою об'єкта Meter. Розглянемо його кроки:

- Створення об'єкту класу Meter для вимірювання метрик: Створюється об'єкт Meter для вимірювання метрик під час поточної епохи та фази.
- Отримання початкового часу та виведення інформації про поточний етап: Виводиться інформація про поточну епоху та фазу тренування.
- Встановлення режиму тренування або валідації для моделі: Режим моделі встановлюється в режим тренування, якщо фаза — "train", і в режим валідації у протилежному випадку.
- Отримання даталоадера для поточної фази: Отримується даталоадер для відповідної фази тренування чи валідації.
- Ініціалізація змінних для втрат та кількості пакетів: Ініціалізуються змінні для відстеження втрат та загальної кількості пакетів у даталоадері.
- Обнулення градієнтів перед кожним батчем: Градієнти обнуляються перед обробкою кожного батчу.
- Ітерація по даталоадеру: Починається ітерація по даталоадеру для обробки кожного батчу.

- Отримання втрат та виводів за допомогою методу forward: Викликається метод forward, який обчислює втрати та отримує виводи моделі.
- Нормалізація втрат за кількість накопичень: Втрати нормалізуються на кількість накопичень для врахування накопичення градієнтів.
- Зворотнє поширення та оновлення ваг моделі для фази тренування: Якщо фаза — "train", то здійснюється зворотнє поширення та оновлення ваг моделі кожних self.accumulation_steps батчів.
- Оновлення втрат та метрик з використанням об'єкта Meter: Втрати та виводи оновлюються в об'єкті Meter для вимірювання метрик.
- Обчислення середніх втрат за епоху: Середні втрати за епоху обчислюються на підставі накопичених втрат та кількості батчів.
- Виведення інформації та логування метрик: Інформація про епоху виводиться, і метрики логуються з використанням функції epoch_log.
- Збереження втрат та коефіцієнтів Дайса: Втрати та коефіцієнти Дайса зберігаються для подальшого аналізу.
- Очищення пам'яті на GPU: Використовується torch.cuda.empty_cache() для очищення пам'яті на GPU після кожної епохи.
- Повернення значення втрат для поточної епохи: Повертається значення втрат для поточної епохи для подальшого використання.

Метод start ініціює тренування моделі протягом заданої кількості епох. Він викликає метод iterate для тренування та валідації, зберігає стан моделі, якщо знайдено новий оптимум, і оновлює швидкість навчання. Проаналізуємо його роботу:

- Ітерація по заданій кількості епох: Здійснюється ітерація по епохам для тренування та валідації.
- Виклик методу iterate для тренування та збереження стану моделі: Викликається метод iterate для фази тренування ("train"). Зберігається стан моделі, оптимізатора та інших параметрів.
- Виклик методу iterate для валідації та оновлення швидкості навчання: Здійснюється виклик методу iterate для фази валідації ("val"). Параметр

val_loss використовується для оновлення швидкості навчання згідно з розкладом, заданим планувальником (self.scheduler).

- Збереження стану моделі, якщо знайдено новий оптимум: Якщо втрати на валідації менше за поточний найкращий показник (self.best_loss), то зберігається стан моделі, оптимізатора та інших параметрів.

3 ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1 Оцінка результатів роботи системи

На рисунку 3.1 зображено ілюстрації оброблених зображень з накладеними сегментаційними масками. Виявлені дефекти першого класу позначаються жовтим забарвленням, дефекти другого класу – блакитним, дефекти третього класу – фіолетовим та дефекти четвертого класу – червоним.

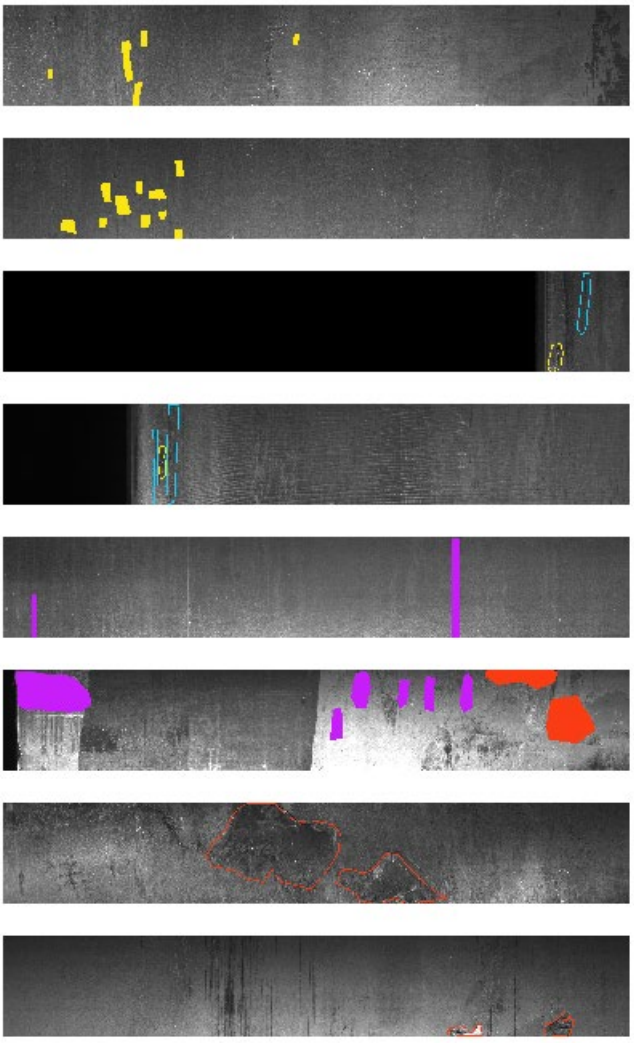


Рис. 3.1 Приклади сегментації виявлених дефектів за класами

Рисунки 3.2 та 3.3 містять демонстрацію дефекта, виділеного контуром фіолетового забарвлення, що відповідає третьому класу, надаючи можливість порівняти межі, промарковані фахівцем, та межі, спрогнозовані моделлю. Можна

зазначити, що системою було порівняно чітко позначено усі дефекти, що були присутні на металевій поверхні. Величина та структура контурів, проведених мережею, збігаються з межами, окресленими на навчальному зображенні. На поверхні існують білі точки, що могли б бути хибно виділени системою як шукані пошкодження, однак неіснуючих дефектів позначено не було.

Рис. 3.2 Контури дефектів третього класу з навчального набору даних

Рис. 3.3 Контури дефектів третього класу, виділені нейронною мережею

Рисунки 3.4 та 3.5 містять демонстрацію металевої поверхні, що одночасно містить два види пошкоджень. Дефекти, що виділено контурами з червоним забарвленням, належать до четвертого класу, у той час як дефекти, що виділено контурами з фіолетовим забарвленням, належать до третього класу. Можна зазначити, що за наявності дефектів двох різних типів система функціонує трішки менш ефективно. Наявні області, що модель хибно виділила у якості дефектних, у той час як одну із дійсно дефектних областей система окреслила частково. Однак усі області з наявними пошкодженнями було вірно виявлено та класифіковано, що є задовільним результатом.

Рис. 3.4 Контури дефектів третього класу та четвертого класів з навчального набору даних

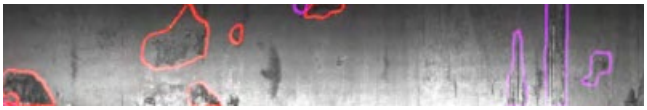


Рис. 3.5 Контури дефектів третього та четвертого класів, виділені нейронною мережею

Випробування результатів діяльності нейронної мережі проводилось протягом 50 епох. Таблиця 3.1 демонструє зміни втрат за обчисленнями бінарної перехресної ентропії та коефіцієнту Дайса. Для зручності кожен рядок відображає дані на момент кожної п’ятої епохи.

Таблиця 3.1

Епоха	Втрати (тренування), %	Втрати (валідація), %	Коефіцієнт Дайса (тренування), %	Коефіцієнт Дайса (валідація), %
				1
				69.1

Рисунок 3.6 відображає наглядні графіки за вказаними числовими даними. Графіками синього кольору позначено результати під час тренування нейронної мережі, а графіками червоного кольору – результати під час валідації.

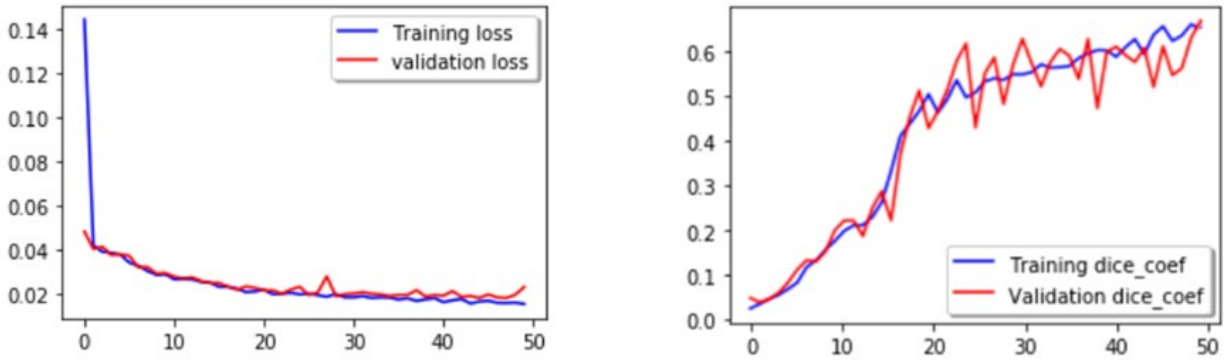


Рис. 3.6 Графіки змін функції втрат та коефіцієнту Дайса від епохи до епохи

3.2 Аналіз ефективності використання різних основних блоків

Протягом тренування системи було здійснено випробування п'яти різних варіантів основних блоків побудови ResNet. Узагальнені результати функціонування системи з застосуванням кожного із розглянутих основних блоків містяться у таблиці 2. На перший погляд, метрика accuracy демонструє відмінні результати, однак під час розрахування цієї метрики враховується клас фону, що покриває значну площу зображення. Це в свою чергу результує кардинальною незбалансованістю класів, що робить показник accuracy невірогідним для поставлених умов, на відміну від розрахунків за коефіцієнтом Дайса.

Найбільш точні результати демонструє система з застосуванням основного блоку Resnet18, що демонструє коефіцієнт Дайса значенням 69%. У той же час, дана система несе найменше число параметрів, у зв'язку з чим її тренування потребує значним чином менше часу у порівнянні з основними блоками більш комплексної побудови. Найменш точні результати було одержано від системи з застосуванням основного блоку ResNet152, не зважаючи на те, що дана модель включає найвище число параметрів серед можливих варіантів.

Таблиця 3.2

Основний блок	Кількість параметрів	Показник Accuracy, %	Коефіцієнт Дайса, %
ResNet18	11M	99.2	69.1
ResNet34	21M	99.2	67.2
ResNet50	23M	99.2	67.2
ResNet101	42M	99.2	67.4
ResNet152	58M	99.2	66.6

Рисунок 3.7 відображає наочні порівняльні гістограми за вказаними числовими даними.

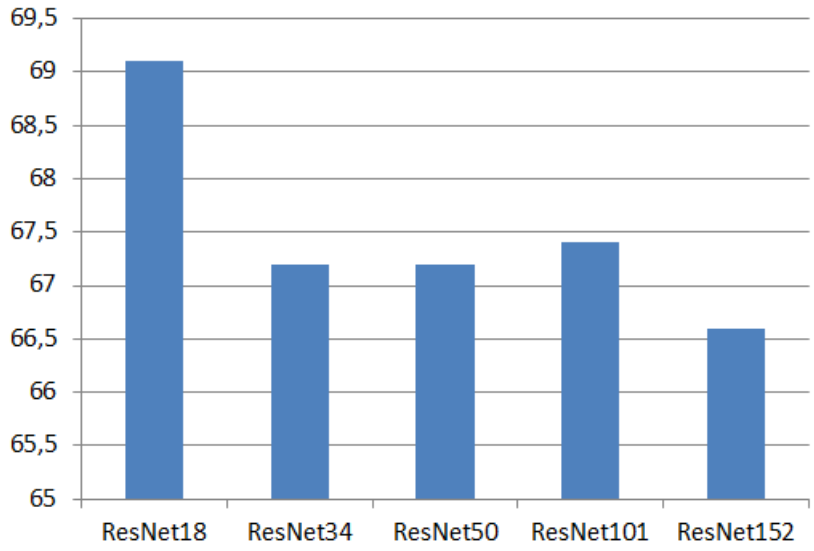


Рис. 3.7. Порівняльні гістограми ефективності різних основних блоків

З таблиці 3.2 та рисунку 3.7 зробити висновки, що підвищення кількості використовуваних параметрів нейронної мережі не у всіх випадках результує вдосконаленням її ефективності. Також, наведені дані демонструють, що всі проаналізовані системи несуть значення коефіцієнту Дайса на рівні 69%, що свідчить про задовільну достовірність одержаних сегментаційних масок.

3.3 Числове порівняння результатів з наявними аналогами

Компанія Cognex представляє собою одного з провідних виробників систем зору, програмного забезпечення, датчиків та промислових зчитувачів штрих-кодів, що застосовуються підприємствами для оптимізації виробничих процесів.

Системи зору Cognex надають підприємствам можливість вдосконалити кінцеву якість виробів, знизити ризики виникнення промислових похибок та зменшити загальні промислові витрати. Пропоновані компанією Cognex засоби машинного зору включають серед свого функціоналу виявлення дефектів на металевих поверхнях.

OMRON Corporation – це велика корпорація, що є постачальником електроніки та займає статус одного зі світових лідерів у постачанні засобів автоматизації. Omron представляє собою одного з найбільших виробників апаратних та програмних комплексів для автоматизації промислових процесів, включаючи вичерпний асортимент засобів технічного контролю, побудованих на основі технологій комп'ютерного зору та машинного навчання, зокрема засоби детектування пошкоджень на металевих поверхнях.

Порівняльну характеристику відсоткової ефективності виялення дефектів на металевих поверхнях між засобами виробництва представлених організацій та даного проекту наведено у таблиці 3.3.

Таблиця 3.3

Засіб	Достовірність за Коефіцієнтом Дайса, %	Помилка сегментації, %
OMRON	64	36
Cognex	67	33
Наведений проект	69	31

Рисунок 3.8 відображає наочні порівняльні гістограми відносно представлених числових даних.

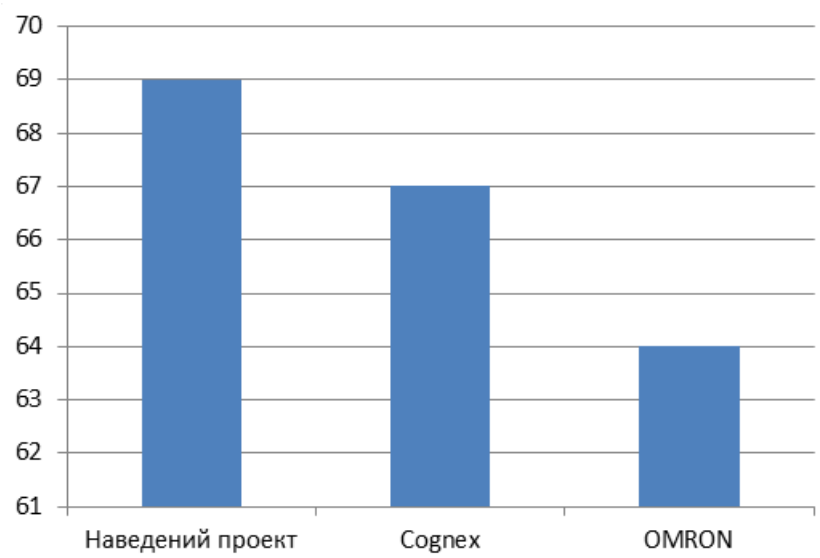


Рис. 3.8. Порівняльні гістограми достовірності роботи засобів для виявлення дефектів на металевих поверхнях

3.4 Потенційні способи вдосконалення моделі

Застосування методів машинного навчання несе певні переваги порівняно з традиційними методами сегментації зображень, але існують і деякі обмеження. Для прикладу, тренування нейронної мережі потребує значного числа анотованих зображень, що містять пошкоджені металеві поверхні. Недостатні об’єми навчальних даних або їх слабка репрезентативність можуть результувати зниженням ефективності сегментації, що являє собою загальний недолік для систем машинного навчання. Обмеження архітектури U-Net може виражатись у зниженні ефективносі сегментації у випадку, коли дефекти на металевій поверхні є більш комплексними, ніж ті, на яких проводилось тренування моделі.

Одним з потенційних шляхів проведення подальших досліджень є поліпшення структури моделі для одержання більш високих показників розглянутих метрик ефективності. Для прикладу, можна застосувати розглянуті модифікації класичної архітектури U-Net, або розглянути можливості застосування інших архітектур чи їх гібридних варіацій. Також доцільною задачею є доповнення набору навчальних зображень, підвищення їх роздільної здатності, збільшення числа класів пошкоджень і якості їх маркування.

ВИСНОВКИ

1. Проведено розгляд наявних методів сегментації зображень, що можуть бути використані для виявлення дефектів на металевих поверхнях. Найбільш оптимальною опцією для основи встановлено застосування методу сегментації зображень на базі згорткової нейронної мережі. Дані мережі мають змогу обробляти значні об'єми даних, дієво тренуватись за допомогою засобів навчання та здійснювати вибір найбільш доцільних рис зображення для сегментації.

2. Здійснено огляд архітектур нейронних мереж та їх особливостей, розглянуто можливості в області покращення процесу виявлення та систематизації дефектів на металевих поверхнях. На базі даного огляду було здійснено вибір структури U-Net.

3. Проведено моделювання та розробку програмного забезпечення, що надає можливість обробки зображень з детектуванням на них дефектів чотирьох класів, а саме: відколів, одиночних вертикальних тріщин, сукупностей вертикальних тріщин, великих плям. Виявлені дефекти позначаються контурами відповідного класу забарвлення та довільної форми. Для розробки було використано мову програмування Python.

4. Здійснено аналіз роботи системи з залученням різних основних блоків. Встановлено, що основні блоки ResNet34, ResNet50, ResNet101 і ResNet152, що несуть більшу кількість параметрів, ніж основний блок ResNet18, не надають більш ефективних результатів, тому доцільним рішенням є використання основного блоку ResNet18, що має найшвидший час тренування завдяки невеликій кількості параметрів.

5. Здійснено тестування розробленого методу на реальних даних, зібрано статистику щодо ефективності роботи, встановлено показник ефективності виявлення дефектів у 69% за розрахунками Коефіцієнту Дайса. Підтверджено відсоткову різницю ефективності у 2% порівняно з засобами Cognex та у 5% порівняно з засобами OMRON.

ПЕРЕЛІК ПОСИЛАНЬ

1. Глобалізація і безпека розвитку: Навчальний посібник. / В. А. Єрмоленко // Черкаси: Вертикаль, 2012. – 336 с.
2. Неруйнівні методи контролю: Навчальний посібник. / Л. М. Сусліков, І. П. Студеняк // Ужгород: Видавництво УжНУ, 2016. – 34 с.
3. Основи металознавства: Навчальний посібник. / Т.Б. Боброва // Ресурсний центр ГУРТ, 2019. – 29 с.
4. Хімічна корозія та захист металів: Навчальний посібник. / П. І. Стоєв, С. В. Литовченко, І. О. Гірка, В. Т. Грицина // ХНУ імені В. Н. Каразіна, 2019. – 216 с.
5. Pattern Recognition (Image Segmentation) / Lei Yan, Hongying Zhao, Yi Lin, Yanbiao Sun // Math Physics Foundation of Advanced Remote Sensing Digital Image Processing. – 2023. – С. 373–402.
6. k-Means Clustering / Chuck Easttom // Machine Learning for Neuroscience. – 2023. – С. 237–254.
7. K-Means Algorithm | AI and Machine Learning [Електронний ресурс] / Awanthika Madhushani. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/373718347_K-Means_Algorithm_AI_and_Machine_Learning
8. Image Thresholding in Image Processing [Електронний ресурс] / Nikolaj Buhl. – 2023. – Режим доступу до ресурсу: https://encord.com/blog/image-thresholding-image-processing/
9. An overview of intelligent image segmentation using active contour models / Yiyang Chen, Pengqiang Ge, Guina Wang, Guirong Weng // Intelligence & Robotics. – 2023. – С. 23–55.
10. Active Contours – A Method for Image Segmentation in Computer Vision [Електронний ресурс] / Dulari Bhatt. – 2023. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/09/active-contours-a-method-for-image-segmentation-in-computer-vision/

11. Convolutional Neural Network / Xiu Zhang, Xin Zhang, Wei Wang // Intelligent Information Processing with Matlab. – 2023. – С. 39–71.
12. Convolutional Neural Networks / James L. Crowley // Human-Centered Artificial Intelligence. – 2023. – С. 67–80.
13. Detection of surface defects on steel strips based on singular value decomposition of digital image / Qianlai Sun, Jianghui Cai, Zhiyi Sun // Mathematical problems in engineering. – 2016. – С. 1–12.
14. Segmentation of rust defects on painted steel surfaces by intelligent image analysis [Електронний ресурс] / Roman Vorobel // Automation in construction. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/348154368_Segmentation_of_rust_defects_on_painted_steel_surfaces_by_intelligent_image_analysis
15. Localization and segmentation of metal cracks using deep learning [Електронний ресурс] / Yasir Aslam // Journal of ambient intelligence and humanized computing. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/339573214_Localization_and_segmentation_of_metal_cracks_using_deep_learning
16. The Amalgamation of the Object Detection and Semantic Segmentation for Steel Surface Defect Detection / Mansi Sharma, Jongtae Lim, Hansung Lee // Applied Sciences. – 2022. – Том. 12, № 12. – С. 6004.
17. Python Programming: A Step-by-Step Guide to Learning the Language [Електронний ресурс] / Chhinder Kaur, Tejinder Pal Singh Brar, Poonam Rana // Manakin Press. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/375029984_Python_Programming_A_Step-by-Step_Guide_to_Learning_the_Language
18. Teaching the Python programming language in higher education institutions / N. A. Otahanov // Informatics and Education. – 2023. – С. 78–86.
19. Deep Learning with TensorFlow / Fabio Nelli // Python Data Analytics. – 2023. – С. 289–321.

20. Image Classification by Tensorflow [Електронний ресурс] / Kashif Shaikh, Dnyaneshwar Pawar, Gaurav Patil, Rahul Patil, Hemant Wani // International Journal for Research in Applied Science & Engineering Technology. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/371962752_Image_Classification_by_Tensorflow
21. KERAS DEEP LEARNING PACKAGE IN PYTHON: A REVIEW [Електронний ресурс] / Dalia shihab Ahmed, Rasha shaker ibrahim Al-badri, Firas Ali Hashim, Nadia Mahmood Hussien // Al-Mustansiriya University. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/374023047_KERAS_DEEP_LEARNING_PACKAGE_IN_PYTHON_A_REVIEW
22. On the Universal Approximation Property of Deep Fully Convolutional Neural Networks [Електронний ресурс] / Ting Lin, Zuowei Shen, Qianxiao Li // International Conference on Learning Representations. – 2022. – Режим доступу до ресурсу: https://www.researchgate.net/publication/365783595_On_the_Universal_Approximation_Property_of_Deep_Fully_Convolutional_Neural_Networks
23. Fully Convolutional Neural Network Structure and Its Loss Function for Image Classification [Електронний ресурс] / Qiuyu Zhu, Xuewen Zu // Institute of Electrical and Electronics Engineers. – 2022. – Режим доступу до ресурсу: https://www.researchgate.net/publication/359637037_Fully_Convolutional_Neural_Network_Structure_and_Its_Loss_Function_for_Image_Classification
24. Research on Semantic Segmentation and Object Grasping Strategy Generation Based on Deeplab Algorithm / Shaobo Li, Qiang Ba, Yang Jing, Liya Yu // The International Conference on Image, Vision and Intelligent Systems. – 2022. – С. 467–476.
25. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation [Електронний ресурс] / Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, Li Fei-Fei // Institute of Electrical and Electronics Engineers. – 2019. – Режим доступу до ресурсу:

https://www.researchgate.net/publication/330304697_Auto-DeepLab_Hierarchical_Neural_Architecture_Search_for_Semantic_Image_Segmentation

26. LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation [Електронний ресурс] / Abhishek Chaurasia, Eugenio Culurciello // Electrical and Electronics Engineers. – 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1707.03718
27. U-Net: Convolutional Networks for Biomedical Image Segmentation [Електронний ресурс] / Olaf Ronneberger, Philipp Fischer, Thomas Brox // Computer Science Department and BIOSS Centre for Biological Signalling Studies. – 2015. – Режим доступу до ресурсу: https://arxiv.org/abs/1505.04597
28. UNet++: A Nested U-Net Architecture for Medical Image Segmentation / Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang // Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. – 2018. – С. 3–11.
29. DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation [Електронний ресурс] / Debesh Jha, Michael A. Riegler, Dag Johansen, Pål Halvorsen, Håvard D. Johansen // IEEE 33rd International Symposium on Computer-Based Medical Systems. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/344051841_DoubleU-Net_A_Deep_Convolutional_Neural_Network_for_Medical_Image_Segmentation
30. MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation [Електронний ресурс] / Nabil Ibtehaz, Mohammad Sohel Rahman // Samsung R&D Institute. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/335634164_MultiResUNet_Rethinking_the_U-Net_architecture_for_multimodal_biomedical_image_segmentation
31. Convolutional Neural Networks (VGG-16) [Електронний ресурс] / Usama Arshad // Ghulam Ishaq Khan Institute of Engineering Sciences and Technology. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/367157946_Convolutional_Neural_Networks_VGG-16

32. Densely Connected Convolutional Networks [Електронний ресурс] / Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger // Institute of Electrical and Electronics Engineers. – 2018. – Режим доступу до ресурсу: https://www.researchgate.net/publication/319770123_Densely_Connected_Convolutional_Networks
33. Deep Residual Learning for Image Recognition [Електронний ресурс] / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // Institute of Electrical and Electronics Engineers. – 2015. – Режим доступу до ресурсу: https://arxiv.org/abs/1512.03385
34. Metrics to Evaluate your Machine Learning Algorithm [Електронний ресурс] / Aditya Mishra // Towards Data Science. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234
35. Metrics to Evaluate your Semantic Segmentation Model [Електронний ресурс] / Ekin Tiu // Towards Data Science. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2
36. Cross-Entropy Loss Function [Електронний ресурс] / Kiprono Elijah Koech // Towards Data Science. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
37. НЕЙРОННІ МЕРЕЖІ: ТЕОРІЯ ТА ПРАКТИКА / С. О. Субботін // Житомир: Вид. О. О. Євенок. – 2020. – С. 9–13.
38. Convolutional Neural Network (CNN) Simplified [Електронний ресурс] / Renu Khandelwal // DataDrivenInvestor. – 2018. – Режим доступу до ресурсу: https://medium.datadriveninvestor.com/convolutional-neural-network-cnn-simplified-ecafd4ee52c5
39. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / Sergey Ioffe, Christian Szegedy // PMLR, vol. 37. – 2015. – С. 448–456

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«РОЗРОБКА МЕТОДУ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЇ ДЕФЕКТІВ НА МЕТАЛЕВИХ ПОВЕРХНЯХ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ»

Виконав: студент групи ПДМ-64 Войцеховський Юрій Юрійович

Керівник: д.т.н., проф., професор кафедри ІПЗ Бондарчук Андрій Петрович

Київ - 2024

МЕТА, ОБ’ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращення процесу виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

Об’єкт дослідження: процес виявлення та класифікації дефектів на металевих поверхнях.

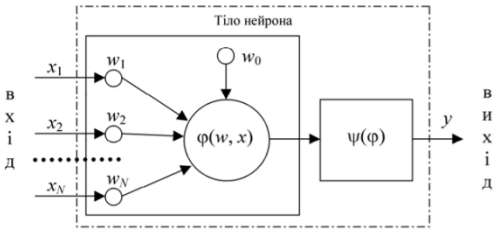
Предмет дослідження: засоби виявлення та класифікації дефектів на металевих поверхнях за допомогою технологій машинного навчання.

АНАЛІЗ МЕТОДІВ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

Метод	Недоліки	Переваги
Метод K-середніх	- Велика чутливість до шуму та викидів у дані - Складність вибору збалансованої кількості кластерів	+ Простота у реалізації і інтерпретації + Висока швидкість використання
Порогова обробка	- Слабкість у роботі з нечітко вираженими межами об'єктів - Складність визначення порогового показника	+ Ефективність обробки зображень, чиї окремі однорідні області відрізняються середньою яскравістю
Активні контури	- Слабкість у роботі зі складними об'єктами - Значна залежність від початкових показників, складність їх налаштування	+ Автономний та адаптивний пошук мінімального стану + Можливість застосування для динамічних об'єктів
Згорткові нейронні мережі	- Значна потреба у тренувальних даних - Слабка ефективність у роботі з невеликими об'ємами даних	+ Здатність до ієрархічного вивчення ознак + Можливість розпізнавання об'єктів незалежно від їх розташування

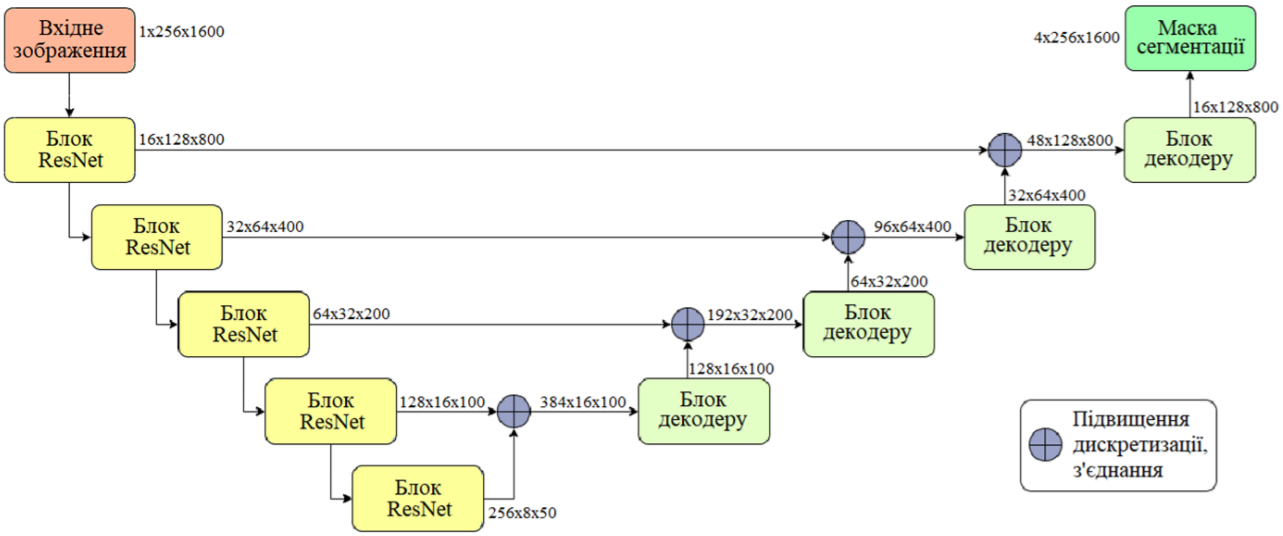
НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

Функціонування штучного нейрону	$y = \psi\ (\varphi(w, x))$
Дискримінаційні функції	
Зважена сума	$\varphi(w,x) = \sum_{j=1}^{N} w_j x_j + w_0$
Зважений добуток	$\varphi(w,x) = \prod_{j=1}^{N} w_j x_j = w_0 \prod_{j=1}^{N} x_j$
Евклідова відстань	$\varphi(w,x) = \sum_{j=1}^{N} (w_j - x_j)^2$
Функції активації	
Випрямлена лінійна одиниця	$\psi(\varphi) = \max(0, \varphi)$
Нормована експоненційна	$\psi(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$

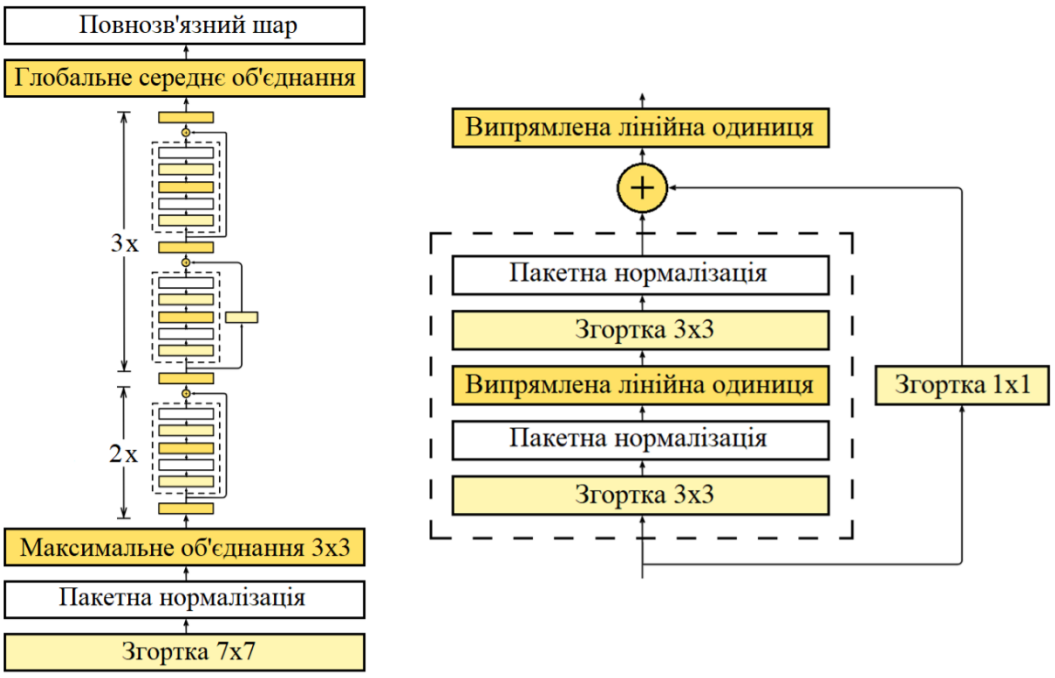


Структура штучного нейрона

АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ



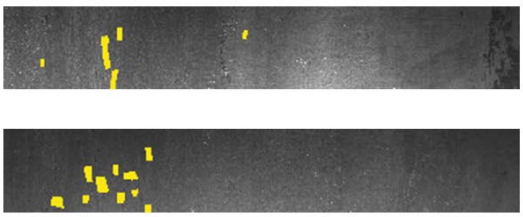
ПОБУДОВА ОСНОВНОГО БЛОКУ



МАТЕМАТИЧНА ОЦІНКА ЕФЕКТИВНОСТІ НЕЙРОННОЇ МЕРЕЖІ

Метрика	Формула	Недоліки	Переваги
Точність	$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$	- Низька достовірність при роботі з незбалансованими класами	+ Порівняно висока достовірність при роботі зі збалансованими класами
Влучність	$precision = \frac{TP}{TP + FP}$	- Чутливість до помилково-позитивних результатів	+ Наочна демонстрація, яка частина позитивно визначених прикладів є дійсними
Повнота	$recall = \frac{TP}{TP + FN}$	- Чутливість до помилково-негативних результатів	+ Наочна демонстрація усіх позитивно визначених прикладів
Коефіцієнт Дайса	$DC = \frac{2(precision \times recall)}{(precision + recall)}$	- Менша достовірність для завдань бінарної класифікації	+ Комбінація влучності і повноти для отримання збалансованих результатів

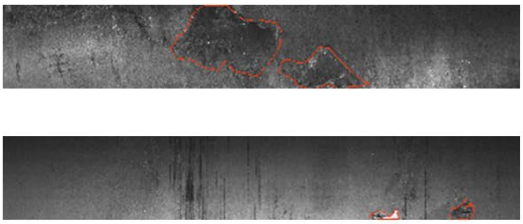
ВИЯВЛЕНІ ДЕФЕКТИ



Дефекти першого класу

Дефекти другого класу

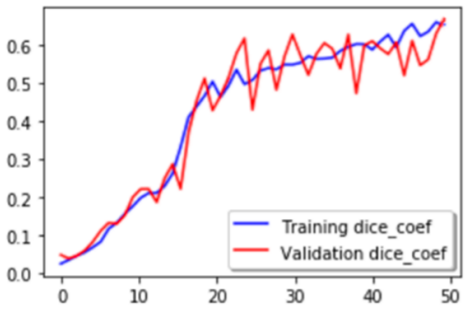
Дефекти третього класу



Дефекти четвертого класу

РЕЗУЛЬТАТИ ТЕСТУВАННЯ СИСТЕМИ

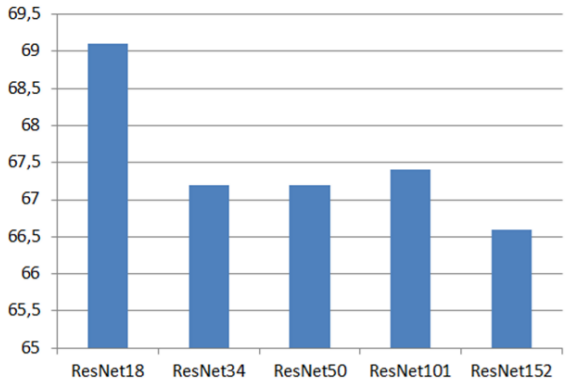
Епоха	Достовірність за коефіцієнтом Дайса (тренування), %	Достовірність за коефіцієнтом Дайса (валідація), %
1	2.75	5.16
5	7.24	8.75
10	18.5	20.99
15	27.56	30.14
20	52.87	44.91
25	53.28	45.10
30	57.48	65.82
35	59.42	61.87
40	61.6	63.97
45	68.75	64.11
50	68.58	69.1



Графік змін коефіцієнту Дайса від епохи до епохи

АНАЛІЗ ЕФЕКТИВНОСТІ ОСНОВНИХ БЛОКІВ

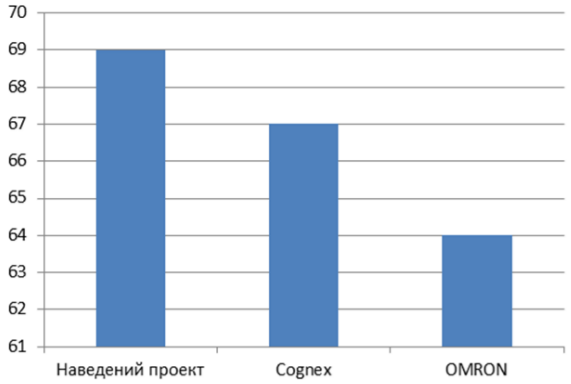
Основний блок	Кількість параметрів	Достовірність за коефіцієнтом Дайса, %
ResNet152	58M	66.6
ResNet101	42M	67.4
ResNet50	23M	67.2
ResNet34	21M	67.2
ResNet18	11M	69.1



Порівняльні гістограми ефективності різних основних блоків

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА

Засоби	Достовірність за коефіцієнтом Дайса, %	Помилка сегментації, %
OMRON	64	36
Cognex	67	33
Наведений проект	69	31



Порівняльні гістограми достовірності роботи засобів для виявлення дефектів на металевих поверхнях

ВИСНОВКИ

1. Проведено огляд існуючих методів сегментації зображень, їх видів та особливостей. Розглянуто можливості таких методів в області покращення процесу виявлення та класифікації дефектів на металевих поверхнях.
2. Побудова моделі для виявлення та класифікації дефектів на металевих поверхнях була проведена на основі архітектури U-Net, оскільки ця побудова у якості основи пропонує дієвий функціонал для локалізації об'єктів на графічних зображеннях та надає ефективні результати при роботі з великими об'ємами даних. Застосовано основні блоки на базі побудови ResNet. Остаточну модель було побудовано за допомогою мови програмування Python.
3. Здійснено тестування розробленого методу на реальних даних, зібрано статистику щодо ефективності роботи, встановлено показник достовірності виявлення дефектів у **69%** за розрахунками Коефіцієнту Дайса. Підтверджено позитивну відсоткову різницю достовірності у **2%** порівняно з засобами Cognex та у **5%** порівняно з засобами OMRON.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Войцеховський Ю.Ю., Бондарчук А.П. Аналіз архітектур нейронних мереж для виконання завдань сегментації зображень. // II Міжнародна наукова конференція «Період трансформаційних процесів в світовій науці: задачі та виклики». – Кривий Ріг, 2024.

Тези доповідей:

1. Войцеховський Ю.Ю., Бондарчук А.П. Аналіз застосування технологій машинного навчання у металургійній промисловості. // Науково-практична конференція «Проблеми комп'ютерної інженерії». – Київ: ДУІКТ, 2023.
2. Войцеховський Ю.Ю. , Бондарчук А.П. Проблеми та перспективи впровадження технологій машинного навчання на металургійних пітприємствах. // VI Міжнародна наукова конференція «Науковий простір: актуальні питання, досягнення та інновації». – Київ, 2023.

ДЯКУЮ ЗА УВАГУ!