НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичного моделювання та аналізу даних

«На правах рукопису»
УДК 004.932:621.391

«До захисту допущено»
Завідувач кафедри
_____ Іван ТЕРЕЩЕНКО
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Прикладна математика»

зі спеціальності: 113 Прикладна математика

на тему: **«Методи комп'ютерного зору для розпізнавання дорожніх знаків та розмітки у системах допомоги водієві з урахуванням впливу погодних умов»**

Виконав:
студент IV курсу, групи ФІ-21
Іщенко Дмитро Володимирович _____

Керівник:
Асистент кафедри ММАД, д-р філософії
Железняков Д. В. _____

Рецензент:
доцент кафедри інформаційної безпеки
НН ФТІ КПІ імені Ігоря Сікорського,
кандидат технічних наук, доцент
Гальчинський Леонід Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Прикладна математика»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Іван ТЕРЕЩЕНКО
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Іщенко Дмитро Володимирович

1. Тема роботи: «*Методи комп'ютерного зору для розпізнавання дорожніх знаків та розмітки у системах допомоги водієві з урахуванням впливу погодних умов*», науковий керівник дисертації: Асистент кафедри ММАД, д-р філософії Железняков Д. В.,

затверджені наказом по університету № __ від «__» _____ 2026 р.

2. Термін подання студентом роботи: «__» _____ 2026 р.

3. Об'єкт дослідження: Процеси розпізнавання дорожніх знаків і виявлення ліній дорожньої розмітки в системах ADAS під впливом несприятливих погодних умов.

4. Предмет дослідження: Методи та алгоритми комп'ютерного зору на основі глибинного навчання та класичних методів обробки зображень для підвищення стійкості розпізнавання дорожніх об'єктів у складних умовах.

5. Перелік завдань:

1. провести аналіз сучасного стану досліджень у галузі розпізнавання дорожніх знаків і ліній розмітки з акцентом на вплив погодних умов;
2. проаналізувати та відібрати джерела даних для проведення експериментів, виявити прогалини в існуючих датасетах;
3. розробити та реалізувати модульну систему розпізнавання, що враховує адаптацію до змінних погодних умов;
4. навчити моделі YOLOv8n для детекції знаків та U-Net для сегментації розмітки із застосуванням transfer learning та двофазного fine-tuning;

5. провести порівняльний аналіз архітектур YOLOv8n/s та U-Net проти DeepLabV3+ за точністю, швидкістю та стійкістю до погодних умов;
6. реалізувати адаптивний модуль класифікації погодних умов на основі MobileNetV2;
7. провести тестування системи на реальних погодних знімках BDD100K та проаналізувати результати.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Презентація доповіді.

7. Дата видачі завдання: 10 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Узгодження теми з науковим керівником	Вересень 2025	Виконано
2	Огляд літератури	Жовтень 2025 – грудень 2025	Виконано
3	Огляд методів	Грудень 2025 – Лютий 2026	Виконано
4	Пошук джерел даних	Лютий 2026 – Березень 2026	Виконано
5	Програмна реалізація методів	Березень 2026 – Квітень 2026	Виконано
6	Аналіз результатів роботи моделей	Квітень 2026 – Травень 2026	Виконано
7	Оформлення дипломної роботи	Травень 2026	Виконано

Студент _____ Дмитро ІЩЕНКО

Керівник _____ Дмитро ЖЕЛЕЗНЯКОВ

ЗМІСТ

Перелік умовних позначень, скорочень і термінів 7
Вступ 8
Розділ 1. Сучасний стан дослідження розпізнавання дорожніх знаків і ліній розмітки 11
1.1 Вплив погодних умов на якість розпізнавання 11
1.2 Огляд робіт з розпізнавання дорожніх знаків 12
1.3 Класичні методи обробки зображень у системах ADAS 14
1.4 Архітектури глибинного навчання для сегментації розмітки 14
1.4.1 U-Net 14
1.4.2 DeepLabV3+ 15
1.4.3 Спеціалізовані підходи 15
1.5 Детекція дорожніх знаків: архітектури 15
1.5.1 Сімейство YOLO 15
1.5.2 Двоетапні детектори 16
1.6 Мультизадачні та адаптивні підходи в ADAS 16
1.7 Джерела даних 17
1.7.1 Датасети для розпізнавання дорожніх знаків 18
1.7.1.1 GTSRB 18
1.7.1.2 CURE-TSD 18
1.7.1.3 Mapillary Traffic Sign Dataset 18
1.7.2 Датасети для сегментації ліній дорожньої розмітки 19
1.7.2.1 TUSimple 19
1.7.2.2 CULane 19
1.7.2.3 BDD100K 19
1.7.3 Датасети з несприятливими погодними умовами 20
1.7.3.1 ACDC 20
1.7.3.2 CADC 20
1.7.4 Порівняльний аналіз датасетів 20
1.7.5 Ключові прогалини в існуючих датасетах 21
1.7.6 Вибір датасетів для даного дослідження 22

Розділ 2. Запропонована система розпізнавання дорожніх об'єктів 24
2.1 Загальна архітектура системи 24
2.2 Модуль попередньої обробки зображень 26
2.3 Модуль класифікації погодних умов 27
2.4 Модуль детекції та класифікації дорожніх знаків 28
2.4.1 Двоетапний підхід 28
2.4.2 Архітектура YOLOv8 28
2.4.3 Класи знаків 28
2.5 Модуль сегментації ліній дорожньої розмітки 29
2.5.1 Архітектура U-Net 29
2.5.2 Тренування на двох датасетах 29
2.5.3 Постобробка маски 29
2.6 Порівняння варіантів архітектури 29
2.7 Обчислювальна складність системи 30
Розділ 3. Математичний опис запропонованого методу 31
3.1 Архітектура U-Net: формальний опис 31
3.1.1 Базові операції 31
3.1.2 Encoder 31
3.1.3 Decoder зі skip connections 32
3.1.4 Вихідний шар та бінарна маска 32
3.2 Функція втрат для сегментації 32
3.3 Алгоритм детекції YOLOv8 33
3.3.1 Активаційна функція SiLU 33
3.3.2 Локалізаційна функція втрат CIoU 33
3.3.3 Distribution Focal Loss 34
3.3.4 Non-Maximum Suppression 34
3.4 Оптимізація та регуляризація 35
3.4.1 Оптимізатор AdamW 35
3.4.2 Планувальник Cosine Annealing 35
3.5 Методи аугментації для симуляції погодних умов 36
3.5.1 Модель туману 36
3.5.2 Симуляція дощу 36
3.5.3 Симуляція нічних умов 36
3.6 Детектор країв Canny: математичний опис 36

3.7 Перетворення Hough для виявлення ліній 37
3.8 Морфологічні операції постобробки 38
3.9 Transfer Learning: математичне обґрунтування 38
Розділ 4. Експерименти та аналіз результатів 40
4.1 Умови проведення експериментів 40
4.2 Метрики оцінювання 40
4.2.1 Метрики для детекції знаків 40
4.2.2 Метрики для сегментації розмітки 41
4.2.3 Метрика стійкості до погодних умов 42
4.3 Результати навчання моделей сегментації 42
4.4 Результати детекції дорожніх знаків 43
4.5 Результати сегментації по сценаріях CULane 46
4.6 Аналіз стійкості до погодних умов на BDD100K 50
4.7 Порівняння базового методу та запропонованого підходу 52
4.8 Обговорення результатів та обмежень 53
4.9 Оцінка обчислювальної ефективності системи 54
Висновки 56
Список використаних джерел 58
Додаток А. Лістинг програми навчання моделі U-Net 63
Додаток Б. Лістинг програми навчання моделі YOLOv8 70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADAS - (Advanced Driver Assistance Systems) - системи допомоги водієві
AMP - (Automatic Mixed Precision) - автоматична змішана точність обчислень
ASPP - (Atrous Spatial Pyramid Pooling) - атрозне пірамідне просторове об'єднання
BCE - (Binary Cross-Entropy) - бінарна крос-ентропія
CBAM - (Convolutional Block Attention Module) - модуль уваги для згорткових блоків
CNN - (Convolutional Neural Network) - згорткова нейронна мережа
CSP - (Cross Stage Partial) - метод часткового перехресного об'єднання ознак
CUDA - (Compute Unified Device Architecture) - платформа паралельних обчислень NVIDIA
DFL - (Distribution Focal Loss) - дистрибутивна фокусна функція втрат
FPN - (Feature Pyramid Network) - мережа пірамід ознак для мультимасштабної детекції
FPS - (Frames Per Second) - кадрів на секунду
GPU - (Graphics Processing Unit) - графічний процесор
GTSRB - (German Traffic Sign Recognition Benchmark) - німецький датасет дорожніх знаків
IoU - (Intersection over Union) - відношення перетину до об'єднання
mAP - (mean Average Precision) - середня точність детекції по всіх класах
NMS - (Non-Maximum Suppression) - придушення немаксимальних значень
PAN - (Path Aggregation Network) - мережа агрегації шляхів
RPD - (Relative Performance Drop) - відносне падіння точності
SiLU - (Sigmoid Linear Unit) - сигмоїдна лінійна одиниця активації
SPPF - (Spatial Pyramid Pooling Fast) - швидке просторове пірамідне об'єднання
VRAM - (Video Random Access Memory) - відеопам'ять графічного процесора
YOLO - (You Only Look Once) - архітектура одностадійної детекції об'єктів

ВСТУП

Актуальність теми. Системи допомоги водієві (ADAS) стали невід'ємною частиною сучасних автомобілів, забезпечуючи підвищення безпеки дорожнього руху та комфорту водіння. Ключовими компонентами таких систем є модулі розпізнавання дорожніх знаків і виявлення ліній дорожньої розмітки, які дозволяють автомобілю орієнтуватися в дорожньому середовищі, попереджати водія про обмеження швидкості та утримувати транспортний засіб у межах смуги руху.

Проте ефективність таких систем значно знижується під впливом несприятливих погодних умов: дощу, туману, снігу та нічного освітлення. За даними Національної адміністрації безпеки дорожнього руху США (NHTSA), близько 21% дорожньо-транспортних пригод відбуваються в умовах дощу, а 16% - у нічний час. Це підкреслює необхідність розробки стійких алгоритмів комп'ютерного зору, здатних функціонувати в реальних, часто непередбачуваних умовах.

Сучасні методи глибинного навчання, зокрема архітектури YOLOv8 [1] та U-Net [2], демонструють вражаючі результати на стандартних бенчмарках. Проте більшість цих моделей навчаються на датасетах, зібраних у сприятливих умовах, що призводить до падіння продуктивності при зміні погодних факторів. Крім того, існує дефіцит локальних датасетів, які відображали б специфіку дорожньої інфраструктури України.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконується в рамках наукових досліджень кафедри математичного моделювання та аналізу даних НТУУ «КПІ ім. Ігоря Сікорського» за напрямом інтелектуальних систем обробки зображень та комп'ютерного зору для автономних транспортних засобів.

Мета і завдання дослідження. Мета роботи - розробити та дослідити методи підвищення стійкості системи розпізнавання дорожніх знаків і ліній розмітки під впливом несприятливих погодних умов на основі гібридного підходу, що поєднує глибинне навчання та класичні методи обробки зображень.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести аналіз сучасного стану досліджень у галузі розпізнавання дорожніх знаків і ліній розмітки з акцентом на вплив погодних умов,

сформувати порівняльну таблицю методів.

2. Проаналізувати та відібрати джерела даних для проведення експериментів, виявити прогалини в існуючих датасетах.

3. Розробити та реалізувати модульну систему розпізнавання, що враховує адаптацію до змінних погодних умов.

4. Обгрунтувати вибір методів дослідження, включаючи архітектури нейронних мереж та методи аугментації.

5. Провести порівняльні експерименти, визначити метрики оцінювання та проаналізувати результати.

Об'єкт дослідження - процеси розпізнавання дорожніх знаків і виявлення ліній дорожньої розмітки в системах ADAS під впливом несприятливих погодних умов.

Предмет дослідження - методи та алгоритми комп'ютерного зору на основі глибинного навчання та класичної обробки зображень для підвищення стійкості розпізнавання дорожніх об'єктів у складних умовах.

Методи дослідження. У роботі використовуються методи глибинного навчання (архітектури YOLOv8, U-Net, DeepLabV3+), класичні методи обробки зображень (детектор країв Canny, перетворення Hough, морфологічні операції), методи аугментації даних для симуляції погодних умов, а також статистичні методи оцінювання якості моделей.

Наукова новизна одержаних результатів.

1. Запропоновано модульну ADAS-систему реального часу, яка вперше поєднує детекцію знаків, сегментацію ліній розмітки та адаптацію до погодних умов в єдиному пайплайні.

2. Проведено порівняльний аналіз архітектур YOLOv8n/s та U-Net проти DeepLabV3+ на уніфікованому наборі датасетів з різними погодними умовами, включаючи реальні знімки BDD100K.

3. Запропоновано адаптивний модуль класифікації погодних умов, що динамічно обирає параметри препроцесингу та порогові значення детекції.

Практичне значення одержаних результатів. Результати роботи можуть бути використані при розробці та вдосконаленні систем ADAS для автомобільної промисловості. Запропоновані методи підвищать безпеку дорожнього руху за рахунок надійнішого розпізнавання дорожніх об'єктів у складних погодних умовах. Реалізована система є придатною для розгортання на вбудованих автомобільних обчислювачах завдяки оптимізованій архітектурі YOLOv8n (3M параметрів, 2,1 мс інференсу) та U-Net (7,8M параметрів, 8,5 мс інференсу). Отримані результати також можуть бути використані для адаптації міжнародних рішень до умов української дорожньої інфраструктури, зокрема для розробки національного датасету дорожніх знаків, оскільки існуючі публічні датасети не охоплюють специфіки вітчизняних доріг.

РОЗДІЛ 1. СУЧАСНИЙ СТАН ДОСЛІДЖЕННЯ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ І ЛІНІЙ РОЗМІТКИ

1.1 Вплив погодних умов на якість розпізнавання

Несприятливі погодні умови є ключовим невирішеним викликом для систем комп'ютерного зору в ADAS. Кожен тип погоди створює специфічні перешкоди для алгоритмів обробки зображень.

Дощ та мокрий асфальт. Краплі на лінзі камери створюють локальні розмиття, а відблиски від мокрої поверхні маскують лінії розмітки. Дослідження [6] показують, що при інтенсивності опадів 100 мм/год miss rate детектора знаків зростає приблизно на 40%. Для боротьби з цим явищем застосовують алгоритми derain та підвищення контрасту CLAHE.

Туман та знижена видимість. Туман описується фізичною моделлю атмосферного розсіювання [7]:

$$I(\mathbf{x}) = J(\mathbf{x}) \cdot t(\mathbf{x}) + A \cdot \big(1 - t(\mathbf{x})\big), \tag{1}$$

де $I(\mathbf{x})$ - спостережуване зображення, $J(\mathbf{x})$ - чисте зображення, $t(\mathbf{x})$ - transmission map (коефіцієнт пропускання), A - atmospheric light. Дослідження [8] показують, що туман знижує mIoU сегментації на 35-45%. Для компенсації застосовують алгоритми dehazing на основі Dark Channel Prior або нейромережеві підходи (AOD-Net).

Сніг та зимові умови. При товщині снігового покриву понад 2 см лінії розмітки перекриваються повністю, і жоден алгоритм обробки зображення не може відновити інформацію, яка фізично відсутня на кадрі. Білий колір снігу також ускладнює розпізнавання знаків за рахунок зниження контрасту.

Нічні умови та засвітлення. Низька освітленість та нерівномірне освітлення від власних фар автомобіля суттєво знижують впевненість детекторів. Для покращення якості застосовують методи low-light enhancement (RetinexNet, EnlightenGAN) та гамма-корекцію.

Комбіновані умови. У реальності погодні фактори комбінуються (дощ + ніч, туман + мокрий асфальт), створюючи синергетичний негативний ефект. Більшість

досліджень фокусуються на одному факторі, що є суттєвим обмеженням. Розробка адаптивних систем, здатних розпізнавати поточні умови та динамічно обирати оптимальну стратегію обробки, - один з ключових напрямів даної роботи.

1.2 Огляд робіт з розпізнавання дорожніх знаків

Задача розпізнавання дорожніх знаків активно досліджується починаючи з 2011 року, коли був введений датасет GTSRB [5] та проведено відповідний конкурс у рамках IJCNN. З того часу точність класифікації на цьому датасеті зросла з рівня людини (98,84%) до значень понад 99% завдяки глибоким нейронним мережам.

Огляд ключових публікацій свідчить про декілька виразних тенденцій у розвитку цієї галузі. По-перше, більшість робіт тестуються виключно на GTSRB в ідеальних умовах освітлення, що не відповідає реальним умовам експлуатації. По-друге, лише поодинокі роботи поєднують задачі детекції знаків та сегментації розмітки. По-третє, вплив погодних умов залишається мало дослідженим: серед розглянутих публікацій лише дві явно враховують реальні погодні фактори.

У наведеній таблиці фігурують такі датасети: GTSRB (German Traffic Sign Recognition Benchmark) - 51 000 зображень 43 класів знаків у сприятливих умовах [5]; CURE-TSD - 1,7M зображень з 12 типами синтетичних деградацій [6]; BDD100K (Berkeley DeepDrive) - 100 000 відеозаписів з реальними погодними умовами (дощ, сніг, туман, ніч) [10]. Детальний опис усіх датасетів наведено в Розділі 2.

Зведений огляд найбільш цитованих та актуальних робіт наведено у табл. 1.1. Таблиця побудована за принципом хронології та охоплює період 2011-2025 років.

Табл. 1.1 – Порівняльний огляд робіт з розпізнавання дорожніх знаків

Метод / Архітектура	**Архітектура**	**Датасет**	**Метрика**	**Погода**
Multi-Scale ConvNet [16]	ConvNet (2 гілки) + трансформація	GTSRB	Acc: 99,17%	Ні
IJCNN Baseline CNN [5]	CNN baseline (переможець IJCNN)	GTSRB	Acc: 98,98%	Ні
Large-Scale TSDR [18]	Mask R-CNN	Власний	Error: <3%	Ні
CURE-TSD CNN [6]	CNN + класифікатор погоди	CURE-TSD	mAP varies	Так
EfficientNet-B0 [25]	EfficientNet-B0 vs AlexNet	GTSRB	Acc: 98,64%	Ні
YOLOv5 vs SSD [26]	YOLOv5 vs SSD	GTSRB	Acc: 97,70%	Ні
VGG19+EnhanceNet+YOLOv4 [19]	VGG19 + EnhanceNet + YOLOv4	CURE-TSD	Acc: 95,03%	Так
ViT vs CNN [27]	Vision Transformer vs CNN	GTSRB	Acc: 98,64%	Ні
ResNet-50+YOLOv8+RT-DETR [28]	ResNet-50 + YOLOv8 + RT-DETR	GTSRB	Acc: 99,8%	Ні
EfficientNetB7 Ensemble [29]	ResNet50, VGG19, EfficientNetB7	GTSRB	Acc: 99,28%	Ні
YOLOv8 Night Retro [30]	YOLOv8 (нічна ретрорефлективність)	ZND (16 500)	mAP varies	Частково

З аналізу табл. 1.1 можна зробити декілька висновків. По-перше, точність на GTSRB у чистих умовах досягла «стелі» (98-99%) ще у 2019 році, тому актуальна задача полягає не у підвищенні точності в ідеальних умовах, а у збереженні цієї точності при поганій погоді. По-друге, лише 2 роботи з 11 (CURE-TSD CNN [6] та VGG19+YOLOv4 [19]) явно враховують погодні умови. По-третє, жодна з розглянутих робіт не поєднує детекцію знаків, сегментацію розмітки та адаптацію до погоди в єдиній системі - це і є ключовою прогалиною, яку виявляє данний огляд.

1.3 Класичні методи обробки зображень у системах ADAS

Попри домінування глибинного навчання у сучасній літературі, класичні методи обробки зображень зберігають практичну цінність у гібридних системах. Вони демонструють стабільніші результати на слабких камерах та є менш чутливими до розподільного зсуву (out-of-distribution).

Детектор країв Canny [12] залишається одним з найпоширеніших методів виявлення країв. Алгоритм включає п'ять послідовних krokів: згладжування гауссіаном, обчислення градієнтів операторами Собеля, придушення немаксимальних значень, подвійне порогове відсікання та трасування країв по зв'язності (hysteresis). У системах ADAS детектор Canny ефективний для попередньої обробки зображень перед виявленням ліній дорожньої розмітки.

Перетворення Hough [13] є класичним методом виявлення геометричних примітивів у зображеннях. Метод відображає точки з декартового простору (x, y) у параметричний простір (ρ, θ), де ρ - відстань від початку координат до прямої, а θ - кут нахилу. Акумулятор підраховує голоси для кожної комбінації (ρ, θ), а піки відповідають лініям у зображенні. Метод стійкий до шуму та розривів у лініях, що важливо для реальних дорожніх умов.

Морфологічні операції (ерозія, дилатація, opening, closing) використовуються для постобробки результатів сегментації: очищення від шуму, з'єднання розривів у лініях та видалення артефактів.

1.4 Архітектури глибинного навчання для сегментації розмітки

1.4.1 U-Net

Ronneberger та співавтори запропонували архітектуру U-Net [2], яка стала стандартом для задач семантичної сегментації. Архітектура має U-подібну форму з симетричним encoder-decoder дизайном та прямими з'єднаннями (skip connections) між відповідними рівнями. Skip connections передають просторову інформацію від encoder до decoder, що є критичним для точної локалізації тонких ліній розмітки шириною 3-5 пікселів.

1.4.2 DeepLabV3+

Chen та співавтори (2018) розробили DeepLabV3+ [4], яка використовує атрозні (dilated) згортки для збільшення рецептивного поля без втрати роздільної здатності. Модуль ASPP застосовує паралельні атрозні згортки з коефіцієнтами дилатації $r \in \{6, 12, 18\}$:

$$y[i] = \sum_k x[i + r \cdot k] \cdot w[k], \tag{2}$$

де r - коефіцієнт розширення (dilation rate), що визначає крок між сусідніми елементами ядра; у модулі ASPP використовуються $r \in \{6, 12, 18\}$ для захоплення контексту на різних масштабах; $w[k]$ - ваги фільтра згортки; $x[i + r \cdot k]$ - значення вхідного тензора, зчитане зі зміщенням $r \cdot k$ відносно позиції i. При $r = 1$ формула (2) зводиться до звичайної згортки. Проте для тонких структур ліній розмітки архітектура U-Net зі skip connections є ефективнішою, оскільки явно передає просторові деталі на кожному рівні.

1.4.3 Спеціалізовані підходи

Ряд досліджень пропонує архітектури, оптимізовані саме для детекції ліній розмітки. SCNN (Spatial CNN) [9] додає просторові згортки, що поширюють інформацію вздовж рядків і стовпців зображення, що дозволяє краще моделювати довгі тонкі структури ліній. Ultra Fast Lane Detection [17] формулює задачу як класифікацію рядків зображення, досягаючи швидкості 300+ FPS, проте без адаптації до погодних умов.

1.5 Детекція дорожніх знаків: архітектури

1.5.1 Сімейство YOLO

Сімейство архітектур YOLO [1] революціонізувало галузь детекції об'єктів, запропонувавши одностадійний підхід, де детекція та класифікація виконуються за один прохід через мережу. YOLOv8 використовує anchor-free detection head і backbone CSPDarknet з C2f блоками. Детекція виконується на трьох масштабах feature maps (52×52, 26×26, 13×13), що дозволяє виявляти знаки різних розмірів

- від далеких до розташованих поряд з камерою.

Ваги YOLOv8 ініціалізувались значеннями, попередньо навченими на датасеті COCO, що прискорює збіжність у 2-3 рази порівняно з випадковою ініціалізацією. Функція втрат включає три компоненти:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}} \cdot \mathcal{L}_{\text{CIoU}} + \lambda_{\text{cls}} \cdot \mathcal{L}_{\text{BCE}} + \lambda_{\text{dfl}} \cdot \mathcal{L}_{\text{DFL}}, \tag{3}$$

де $\lambda_{\text{box}} = 7{,}5$, $\lambda_{\text{cls}} = 0{,}5$, $\lambda_{\text{dfl}} = 1{,}5$ - вагові коефіцієнти.

1.5.2 Двоетапні детектори

Faster R-CNN [3] представляє класичний двоетапний підхід: спочатку Region Proposal Network генерує кандидатні регіони, потім класифікатор уточнює bounding boxes та визначає клас. Метод забезпечує вищу точність локалізації, але значно поступається YOLO за швидкістю (15-20 FPS проти 100+ FPS), що робить його непридатним для ADAS реального часу.

1.6 Мультизадачні та адаптивні підходи в ADAS

Мультизадачне навчання дозволяє одній мережі вирішувати кілька пов'язаних задач, використовуючи спільні ознаки. Це забезпечує ефективніше використання обчислювальних ресурсів та покращення узагальнення через спільні представлення. Для ADAS одна мережа може одночасно виконувати сегментацію дорожньої розмітки, детекцію знаків та класифікацію погодних умов.

У літературі виокремлюють два підходи до архітектури: hard parameter sharing - спільний encoder з окремими heads для кожної задачі, та soft parameter sharing - окремі мережі з регуляризацією для схожості параметрів. Перший підхід є більш обчислювально ефективним і обраний у даній роботі.

Порівняння існуючих підходів за функціональністю наведено у табл. 1.2.

Табл. 1.2 – Порівняння підходів за функціональністю

Метод (рік)	**Задачі**	**Клас. погоди**	**Реальна погода**	**Ключове обмеження**
Multi-Scale ConvNet [16]	Знаки	Ні	Ні	Тільки класифікація
SCNN [9]	Розмітка	Ні	Ні	Без погодної адаптації
CURE-TSD CNN [6]	Знаки	Так	Синтетична	Без сегментації розмітки
Ultra Fast Lane [17]	Розмітка	Ні	Ні	Highway only
VGG19+Enhance-Net+YOLOv4 [19]	Знаки	Так	Синтетична	Без розмітки, 12 FPS
LD-CAM (2024)	Розмітка	Ні	Так	Без детекції знаків
DSF-YOLO [21]	Знаки	Ні	Синтетична	Без розмітки
Multimodal LLM [20]	Знаки + Розмітка	Ні	Ні	LLM занадто повільний

Аналіз табл. 1.2 підтверджує: жодна з розглянутих робіт не вирішує одночасно всі три задачі при наявності реальних погодних умов. Система VGG19+Enhance-Net+YOLOv4 [19] концептуально найближча, але досягає лише 12,79 FPS (недостатньо для ADAS) і не вирішує задачу сегментації розмітки. Роботи, що обробляють розмітку в поганих умовах (LD-CAM), не виконують детекцію знаків. Існуюча література не містить системи, яка поєднувала б усі три функціональності з реальними метриками стійкості до погодних умов.

1.7 Джерела даних

Вибір датасетів є одним з ключових рішень у будь-якому дослідженні машинного навчання. Для задач розпізнавання дорожніх знаків і ліній розмітки існує кілька усталених наборів даних, які суттєво відрізняються за умовами зйомки, обсягом та типом анотацій. У цьому розділі проведено огляд основних датасетів, виявлено їх переваги і недоліки, а також обгрунтовано вибір датасетів для даного дослідження.

1.7.1 Датасети для розпізнавання дорожніх знаків

1.7.1.1 GTSRB

German Traffic Sign Recognition Benchmark (GTSRB) [5] - найбільш популярний бенчмарк для класифікації дорожніх знаків, створений у 2011 році. Датасет містить понад 51 000 зображень, розділених на 43 класи німецьких дорожніх знаків. Роздільна здатність зображень варіюється від 15×15 до 250×250 пікселів. Переваги датасету: великий обсяг, збалансовані класи та широке використання в літературі, що уможливлює порівняння з іншими роботами. Недоліки: датасет зібрано переважно у сприятливих погодних умовах, містить лише німецькі знаки та є застарілим (2011 рік).

1.7.1.2 CURE-TSD

Challenging Unreal and Real Environments for Traffic Sign Detection (CURE-TSD) [6] - спеціалізований датасет для тестування стійкості моделей до деградацій зображень. Містить 1,7 мільйона зображень (80% синтетичних, 20% реальних) з 14 класами знаків. Систематично охоплює 12 типів деградацій (дощ, сніг, туман, motion blur, lens blur, засвітлення та ін.) у 5 рівнях інтенсивності. Це робить CURE-TSD цінним для оцінювання стійкості, проте синтетична природа більшості даних обмежує реалістичність тестування.

1.7.1.3 Mapillary Traffic Sign Dataset

Великомасштабний датасет, зібраний з краудсорсингової платформи Mapillary. Містить понад 100 000 реальних зображень з більш ніж 400 класами знаків з усього світу, включаючи Європу, Америку та Азію. Анотації у вигляді полігональних масок забезпечують точну локалізацію знаків. Географічна різноманітність робить цей датасет придатним для тренування узагальнюючих моделей.

1.7.2 Датасети для сегментації ліній дорожньої розмітки

1.7.2.1 TUSimple

TuSimple Lane Detection Benchmark [11] - датасет для чемпіонату з детекції ліній на швидкісних трасах. Містить 6 408 кадрів роздільною здатністю 1280×720 пікселів, зібраних виключно на highway у денний час при ясному освітленні. Анотації представлені у вигляді координат точок вздовж ліній у форматі JSON. Невеликий обсяг та вузькі сценарії обмежують узагальнення, проте датасет є базовим бенчмарком у більшості досліджень.

1.7.2.2 CULane

CULane [9] - найбільший датасет для детекції ліній розмітки, створений Chinese University of Hong Kong. Містить 133 235 кадрів (55 годин відео) роздільною здатністю 1640×590 пікселів. Особливістю датасету є поділ на 9 категорій складності: звичайні умови, щільний трафік, засвічення, тіні, відсутність ліній, стрілки, повороти, перехрестя та ніч. Датасет не містить знімків з дощем або снігом, проте є найбільш різноманітним за сценаріями серед доступних бенчмарків для цієї задачі.

1.7.2.3 BDD100K

Berkeley DeepDrive 100K [10] - один з найбільших і найрізноманітніших датасетів для автономного водіння. Містить 100 000 відеозаписів (по 40 секунд кожен) з мультизадачними анотаціями: детекція об'єктів, сегментація доріг, lane marking, drivable area. Ключова перевага - наявність реальних погодних умов: clear, rainy, snowy, foggy, overcast, а також часових умов: daytime, night, dawn/dusk. Розподіл по погодних умовах: clear 16 532, cloudy 15 484, rain 3 314, snow 3 796, fog 69, night 31 900 зображень.

1.7.3 Датасети з несприятливими погодними умовами

1.7.3.1 ACDC

Adverse Conditions Dataset with Correspondences (ACDC) [8] - спеціалізований датасет для тестування в несприятливих умовах. Містить 4 006 зображень роздільною здатністю 1920×1080 пікселів, розділених на 4 умови: туман, ніч, дощ та сніг (по $\approx$1000 зображень на умову). Унікальною особливістю є наявність парних знімків однієї і тієї ж сцени у несприятливих та ясних умовах. Анотації включають семантичну сегментацію 19 класів.

1.7.3.2 CADC

Canadian Adverse Driving Conditions Dataset - датасет, зібраний у зимових умовах Канади. Містить 7 000 кадрів з 56 сцен при снігу та льоді, доповнений мультисенсорними даними (камери + LiDAR).

1.7.4 Порівняльний аналіз датасетів

Для обгрунтованого вибору датасетів наведено порівняльні таблиці за ключовими характеристиками. Табл. 1.3 містить порівняння датасетів для задачі розпізнавання знаків.

Табл. 1.3 – Порівняння датасетів для розпізнавання дорожніх знаків

Датасет	**Рік**	**Зобр.**	**Кл.**	**Анотації**	**Погода**	**Реальні**
GTSRB	2011	51 000	43	bbox + клас	Ні	Так
GTSDB	2013	900	43	bbox	Ні	Так
TT100K	2016	100 000	221	bbox	Частково	Так
CURE-TSD	2018	1 700 000	14	bbox	Так	20%
Mapillary TS	2019	100 000	400+	маски	Частково	Так
ACDC Signs	2021	4 000	-	seg.	Так	Так

Табл. 1.4 містить порівняння датасетів для задачі сегментації ліній розмітки.

Табл. 1.4 – Порівняння датасетів для сегментації ліній розмітки

Датасет	**Рік**	**Кадрів**	**Роздільн.**	**Анотації**	**Погода**	**Сцен.**
TUSimple	2017	6 408	1280×720	координати	Ні	1
CULane	2018	133 235	1640×590	маски+точки	Частково	9
BDD100K	2018	100 000	1280×720	lane markings	Так	Багато
LLAMAS	2019	100 000	1276×717	маски	Ні	Highway
CurveLanes	2020	150 000	1280×720	координати	Частково	Криволін.
ACDC Lanes	2021	4 006	1920×1080	seg. + парні	Так	4 погоди
FoggyLane	2024	1 423	1640×590	inst. маски	Туман	1

Зведене порівняння покриття погодних умов різними датасетами наведено у табл. 1.5.

Табл. 1.5 – Покриття погодних умов датасетами

Датасет	**Реальна погода**	**Синт. аугм.**	**Нічні умови**	**Задача**
GTSRB	Відсутня	Відсутня	Відсутні	Класифікація знаків
TT100K	Відсутня	Відсутня	Є	Детекція знаків
CURE-TSD	Відсутня	Дощ, сніг, туман, ніч (12 типів)	Є	Детекція знаків
Mapillary TS	Дощ, сніг, туман, ніч	Відсутня	Є	Детекція знаків
TUSimple	Відсутня	Відсутня	Відсутні	Детекція розмітки
CULane	Відсутня	Відсутня	Є (сценарій night)	Детекція розмітки
BDD100K	Дощ, сніг, туман, ніч	Відсутня	Є	Мультизадачний
ACDC	Дощ, сніг, туман, ніч	Відсутня	Є	Семантична сегментація
FoggyLane	Тільки туман	Відсутня	Відсутні	Детекція розмітки

1.7.5 Ключові прогалини в існуючих датасетах

Аналіз табл. 1.3-1.5 дозволяє виділити три ключові прогалини.

По-перше, найпопулярніший датасет для знаків GTSRB не містить жодних

погодних умов. Моделі, навчені на ньому, не тестуються у реальних умовах експлуатації. CURE-TSD покриває деградації, але переважно синтетично і лише 14 класів знаків.

По-друге, для розмітки найбільш цитований TUSimple - виключно highway при ясному освітленні. CULane має нічні умови та тіні, проте не містить дощу і снігу. Жоден широко використовуваний датасет не покриває всі 4 погодні умови одночасно для задачі детекції розмітки.

По-третє, жоден існуючий датасет не призначений одночасно для знаків, розмітки та погодних умов в єдиній системі. BDD100K найбільш близький, проте анотації розмітки у ньому менш деталізовані порівняно з CULane.

1.7.6 Вибір датасетів для даного дослідження

На підставі проведеного аналізу для дослідження обрано такі датасети:

- **GTSRB** - базовий датасет для тренування та оцінювання детектора знаків (43 класи, широке цитування уможливлює порівняння);
- **TUSimple** - базова сегментація розмітки (highway, 6 408 кадрів);
- **CULane** - складні сценарії для fine-tuning сегментації (9 категорій, 133 235 кадрів);
- **BDD100K** - тестування стійкості системи до реальних погодних умов (дощ, сніг, туман, ніч).

Загальний обсяг сформованого датасету склав 106 749 зображень.

Висновки до розділу 1

Проведений аналіз літератури та існуючих датасетів дозволяє виділити три ключові прогалини сучасних досліджень.

По-перше, *алгоритмічна прогалина*: жодна з розглянутих 11 публікацій не поєднує детекцію дорожніх знаків, сегментацію розмітки та адаптацію до погодних умов в єдиній системі реального часу. Точність на GTSRB у чистих умовах досягла насичення (98-99%) ще у 2019 році; актуальна задача - збереження цієї точності при реальних погодних факторах. Найближча система

(VGG19+EnhanceNet+YOLOv4 [19]) досягає лише 12,79 FPS та не вирішує задачу сегментації розмітки.

По-друге, *методологічна прогалина*: домінування синтетичної аугментації над тестуванням на реальних погодних знімках не дозволяє об'єктивно оцінити стійкість алгоритмів у реальних умовах.

По-третє, *датасетна прогалина*: жоден існуючий датасет не задовольняє одночасно потреби систем детекції знаків, сегментації розмітки та адаптації до погодних умов. Це обгрунтовує необхідність комбінування кількох джерел даних: GTSRB (знаки, 51 000 зображень), TUSimple та CULane (розмітка, 139 643 кадри), BDD100K (реальні погодні умови, 100 000 записів).

Усунення цих трьох прогалин є метою та мотивацією даної роботи.

РОЗДІЛ 2. ЗАПРОПОНОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ДОРОЖНІХ ОБ'ЄКТІВ

2.1 Загальна архітектура системи

Запропонована система є модульною ADAS-системою реального часу, що одночасно вирішує три взаємопов'язані задачі: детекцію та класифікацію дорожніх знаків, сегментацію ліній дорожньої розмітки та адаптацію до поточних погодних умов. На відміну від розглянутих у розділі 1 підходів, що вирішують ці задачі окремо, запропонована архітектура використовує спільний пайплайн обробки зображень і єдиний модуль класифікації погоди, що дозволяє динамічно обирати оптимальну стратегію для кожного модуля.

Загальна схема системи складається з п'яти послідовних модулів: модуль попередньої обробки $\rightarrow$ модуль класифікації погоди $\rightarrow$ модуль детекції знаків (YOLOv8) $\rightarrow$ модуль сегментації розмітки (U-Net) $\rightarrow$ модуль інтеграції результатів.

На рис. 2.1 зображено архітектуру пайплайну системи з двома режимами роботи та прикладами вхідних і вихідних даних кожного модуля.

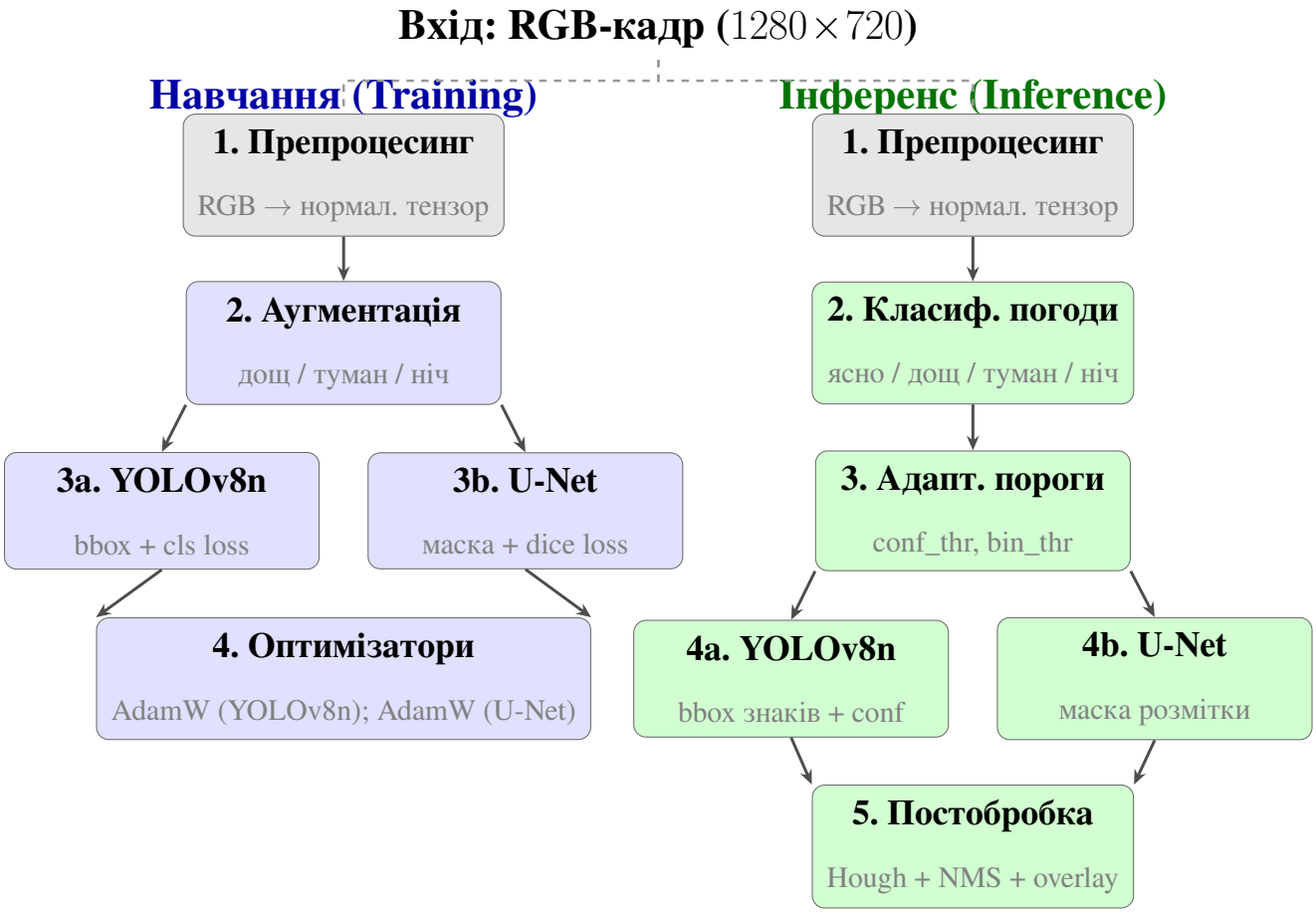

Рис. 2.1 – Пайплайн системи: трек навчання (синій) та інференсу (зелений). Сірі блоки - спільні для обох режимів.

Щодо блоку «2. Класифікатор погоди» на рис. 2.1. Обидві моделі (YOLOv8n та U-Net) навчаються на аугментованих даних і бачать синтетичні зображення різних погодних умов. Однак аугментація забезпечує *стійкість* моделей, а не *оптимальність* для конкретної умови: оптимальний поріг впевненості для туману (0,18) суттєво відрізняється від порогу при ясному небі (0,25). Класифікатор погоди динамічно коригує ці пороги (conf_thr, bin_thr) для кожного кадру замість навчання шести окремих спеціалізованих моделей, що потребувало б у 6 разів більше пам'яті.

Ключові принципи архітектури:

- **Модульність**: кожен компонент незалежно розробляється, тестується та може замінюватись без зміни інших модулів;
- **Адаптивність**: результат класифікації погоди визначає параметри препроцесингу та порогові значення детекції;

- **Ефективність**: паралельне виконання модулів YOLOv8 та U-Net мінімізує загальну затримку;
- **Інтерпретованість**: проміжні результати кожного модуля доступні для аналізу та налагодження.

2.2 Модуль попередньої обробки зображень

Модуль виконує підготовку вхідного зображення перед подачею до нейронних мереж і є першою ланкою пайплайну. Вхідні дані - RGB зображення з камери автомобіля типовою роздільною здатністю 1280×720 або 1920×1080 пікселів.

Нормалізація. Значення пікселів стандартизуються за статистикою ImageNet, оскільки backbone обох нейронних мереж ініціалізується вагами, навченими на ImageNet:

$$I_{\text{norm}}(c) = \frac{I(c)/255{,}0 - \mu_c}{\sigma_c}, \tag{4}$$

де $c \in \{R, G, B\}$ - канал зображення, $\boldsymbol{\mu} = [0{,}485;\ 0{,}456;\ 0{,}406]$, $\boldsymbol{\sigma} = [0{,}229;\ 0{,}224;\ 0{,}225]$.

Ці значення є стандартною статистикою ImageNet і **лишаються незмінними** незалежно від датасету (GTSRB, BDD100K, CULane) та умов освітлення. Зміна $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ під час fine-tuning або inference призвела б до невідповідності з очікуваним розподілом backbone-мережі (навченої на ImageNet) та деградації якості. Адаптація до різних умов освітлення виконується на рівні препроцесингу (CLAHE, гамма-корекція, derain), а *після* цих перетворень нормалізація за ImageNet-статистикою залишається стандартною.

Адаптивне підвищення контрасту (CLAHE). У несприятливих умовах контраст зображення значно знижується. CLAHE виконує локальне вирівнювання гістограми через поділ зображення на тайли розміром 8×8 та обрізання гістограми на рівні clip_limit (2,0 при ясному освітленні, 3,5 при тумані або дощі). Застосовується до каналу L у просторі LAB для уникнення зміни кольору.

Адаптивний препроцесинг. Залежно від класифікованої погодної умови система застосовує різні пайплайни обробки:

- Clear (ясно): мінімальна обробка, лише базова нормалізація;
- Rain (дощ): алгоритм derain + CLAHE з clip_limit = 3,0;

- Fog (туман): алгоритм dehaze + sharpening, clip_limit = 3,5;
- Snow (сніг): корекція балансу білого + CLAHE;
- Night (ніч): гамма-корекція ($\gamma = 1{,}5$) + denoising.

2.3 Модуль класифікації погодних умов

Модуль класифікує поточні умови зйомки на 6 класів: ясна погода, хмарно, дощ, сніг, туман, ніч. Як архітектуру обрано MobileNetV2 (3,5M параметрів) - легку модель, здатну працювати паралельно з основними модулями при очікуваній швидкості 100+ FPS. Тренувальним датасетом є BDD100K [10] з анотаціями погодних умов; при навчанні також застосовується нормалізація за статистикою ImageNet (формула (4)), оскільки backbone MobileNetV2 теж ініціалізується вагами ImageNet.

Обґрунтування необхідності класифікатора погоди. Може виникнути запитання: якщо YOLOv8n та U-Net навчаються на аугментованих даних (бачать синтетичні зображення з різними погодними умовами), навіщо окремий класифікатор погоди?

Аугментація під час навчання *підвищує стійкість* моделей до різних умов, але не *оптимізує* їх поведінку для конкретної умови. Наприклад, оптимальний поріг впевненості (confidence threshold) для детекції у тумані (0,18) суттєво відрізняється від порогу при ясному небі (0,25) - використання універсального значення призводить до зайвих хибних спрацювань або пропусків. Класифікатор погоди дозволяє *динамічно обирати* ці параметри без необхідності навчати окремі спеціалізовані моделі для кожної погодної умови. Альтернативний підхід - набір окремих моделей (одна для туману, одна для дощу тощо) - потребував би у 6 разів більше пам'яті та часу навчання, що неприйнятно для вбудованих автомобільних обчислювачів.

Результат класифікації визначає параметри для наступних модулів: пороги впевненості для YOLOv8n (0,25 при ясному освітленні, 0,18-0,20 при несприятливих умовах), поріг бінаризації маски U-Net (0,50 при ясному освітленні, 0,40-0,45 при тумані або дощі), а також параметри детектора Canny для генерації карти країв.

Для реальних сценаріїв, де умови поєднуються, система підтримує мультимі-

тковий режим: вихід - вектор ймовірностей $[p_0, \ldots, p_5]$, де активними вважаються всі класи з $p_i > 0{,}3$.

2.4 Модуль детекції та класифікації дорожніх знаків

2.4.1 Двоетапний підхід

Модуль реалізує двоетапний підхід: YOLOv8 для швидкої детекції (локалізації знаків у кадрі), а ResNet-50 як окремий класифікатор для точної ідентифікації типу знаку серед схожих класів (наприклад, обмеження 50 проти 80 км/год). Такий підхід підвищує точність ідентифікації на 5-10% порівняно з використанням тільки YOLOv8.

2.4.2 Архітектура YOLOv8

YOLOv8n (nano) містить backbone CSPDarknet з C2f блоками, neck FPN + PAN та anchor-free detection head на трьох масштабах. Детекція на трьох рівнях 52×52, 26×26, 13×13 дозволяє виявляти знаки на різних відстанях. Кількість параметрів: 3,0M (YOLOv8n) та 11,2M (YOLOv8s).

Постобробка детекцій включає NMS з IoU threshold = 0,45 та адаптивним confidence threshold залежно від погодних умов, а також temporal filtering - підтвердження знаку після виявлення у $N = 3$ послідовних кадрах.

2.4.3 Класи знаків

Система розпізнає 43 типи знаків за стандартом GTSRB, об'єднаних у п'ять категорій: знаки обмеження швидкості (9 класів), заборонні знаки (6 класів), знаки пріоритету (3 класи), попереджувальні знаки (14 класів) та знаки напрямку (7 класів). Кожному знаку відповідає власний колір рамки для візуалізації.

2.5 Модуль сегментації ліній дорожньої розмітки

2.5.1 Архітектура U-Net

Для сегментації ліній розмітки обрано U-Net з encoder-decoder структурою та skip connections. Encoder послідовно зменшує просторову роздільну здатність при збільшенні кількості каналів ($3 \to 32 \to 64 \to 128 \to 256 \to 512$). Decoder відновлює роздільну здатність через транспоновані згортки. На кожному рівні decoder конкатенується з відповідним рівнем encoder (skip connection), що зберігає просторову інформацію, критичну для точної локалізації тонких ліній шириною 3-5 пікселів.

2.5.2 Тренування на двох датасетах

Модель тренувалась у два послідовні етапи. На першому - базове навчання на TUSimple (2 900 кадрів, 30 епох, batch = 4). На другому - fine-tuning на CULane із ініціалізацією вагами моделі TUSimple (transfer learning), 8 000 тренувальних кадрів з 9 складних сценаріїв, 40 епох, batch = 6.

2.5.3 Постобробка маски

Після отримання бінарної маски від U-Net послідовно застосовуються: морфологічне closing (kernel 5×5) для з'єднання розривів, морфологічне opening (kernel 3×3) для видалення артефактів, Hough Transform для параметричного фітингу ліній та temporal filtering для згладжування між кадрами.

2.6 Порівняння варіантів архітектури

У роботі порівнювались два варіанти архітектури для кожної задачі: YOLOv8n проти YOLOv8s для детекції знаків та U-Net проти DeepLabV3+ для сегментації розмітки. YOLOv8n обрано основною моделлю: при 3,7-кратно меншій кількості параметрів (3,0M проти 11,2M) вона досягає вищого mAP та нижчої затримки інференсу. U-Net перевершує DeepLabV3+ на 64,8% за IoU завдяки skip connections. Детальне кількісне порівняння наведено у табл. 4.3 (підрозд. 2.4).

2.7 Обчислювальна складність системи

Модульна архітектура системи дозволяє виконувати модулі детекції та сегментації паралельно на GPU, що суттєво зменшує загальну затримку. Детальна оцінка часу інференсу кожного модуля та підтвердження відповідності вимозі ADAS ($\geq$25 FPS) наведено у підрозділі 5.8.

Висновки до розділу 2

Розроблена система є першою серед розглянутих у літературному огляді, що поєднує детекцію знаків, сегментацію розмітки та адаптацію до погодних умов в єдиному пайплайні реального часу. Модульна архітектура забезпечує гнучкість та можливість незалежного вдосконалення кожного компонента.

Порівняння варіантів архітектури (табл. 4.3) показує, що YOLOv8n при 3,7-кратно меншій кількості параметрів ніж YOLOv8s забезпечує вищий mAP@0.5 (0,9949 проти 0,9940) та меншу затримку інференсу (2,1 мс проти 2,6 мс). Для сегментації U-Net перевершує DeepLabV3+ на 64,8% за IoU завдяки skip connections, які явно передають просторову інформацію на кожному рівні. Кількісна оцінка обчислювальних витрат наведена у підрозд. 2.9.

РОЗДІЛ 3. МАТЕМАТИЧНИЙ ОПИС ЗАПРОПОНОВАНОГО МЕТОДУ

3.1 Архітектура U-Net: формальний опис

3.1.1 Базові операції

Вхід мережі - тензор $\mathbf{X} \in \mathbb{R}^{B\times 3\times H\times W}$, де B - batch size, $H = 256$, $W = 512$ - розміри зображення.

Базовим будівельним блоком U-Net є подвійна згортка:

$$\mathrm{DoubleConv}(\mathbf{x}) = \mathrm{ReLU}\big(\mathrm{BN}(\mathrm{Conv}_2(\mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_1(\mathbf{x}))))\big), \tag{5}$$

де Conv - операція 2D згортки (ядро 3×3, padding = 1), BN - Batch Normalization, $\mathrm{ReLU}(x) = \max(0, x)$.

Batch Normalization нормалізує активації:

$$\mathrm{BN}(\mathbf{x}) = \gamma \cdot \frac{\mathbf{x} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta, \tag{6}$$

де $\mu_B = \frac{1}{m}\sum_i x_i$ - середнє, $\sigma_B^2 = \frac{1}{m}\sum_i (x_i - \mu_B)^2$ - дисперсія по батчу, γ, β - навчувані параметри, $\varepsilon = 10^{-5}$.

3.1.2 Encoder (contracting path)

Послідовність encoder з каналами $[32, 64, 128, 256]$:

$$\begin{aligned}
E_1 &= \mathrm{DoubleConv}(\mathbf{X},\ 3 \to 32) \quad \in \mathbb{R}^{B\times 32\times 256\times 512},\\
P_1 &= \mathrm{MaxPool}(E_1,\ 2{\times}2) \quad \in \mathbb{R}^{B\times 32\times 128\times 256},\\
E_2 &= \mathrm{DoubleConv}(P_1,\ 32 \to 64), \quad P_2 = \mathrm{MaxPool}(E_2,\ 2{\times}2),\\
E_3 &= \mathrm{DoubleConv}(P_2,\ 64 \to 128), \quad P_3 = \mathrm{MaxPool}(E_3,\ 2{\times}2),\\
E_4 &= \mathrm{DoubleConv}(P_3,\ 128 \to 256), \quad P_4 = \mathrm{MaxPool}(E_4,\ 2{\times}2),\\
B &= \mathrm{DoubleConv}(P_4,\ 256 \to 512) \quad \in \mathbb{R}^{B\times 512\times 16\times 32},
\end{aligned} \tag{7}$$

де $\mathrm{MaxPool}(\mathbf{x}, k{\times}k)$: $y(i,j) = \max\{x(ki+p, kj+q) : 0 \le p, q < k\}$.

3.1.3 Decoder (expansive path) зі skip connections

Кожен рівень decoder виконує upsampling та конкатенацію з відповідним рівнем encoder:

$$
\begin{aligned}
D_4 &= \mathrm{DoubleConv}\big(\mathrm{Concat}[\mathrm{Up}(B, 512{\to}256),\ E_4],\ 512 \to 256\big),\\
D_3 &= \mathrm{DoubleConv}\big(\mathrm{Concat}[\mathrm{Up}(D_4, 256{\to}128),\ E_3],\ 256 \to 128\big),\\
D_2 &= \mathrm{DoubleConv}\big(\mathrm{Concat}[\mathrm{Up}(D_3, 128{\to}64),\ E_2],\ 128 \to 64\big),\\
D_1 &= \mathrm{DoubleConv}\big(\mathrm{Concat}[\mathrm{Up}(D_2, 64{\to}32),\ E_1],\ 64 \to 32\big),
\end{aligned}
\tag{8}
$$

де Up - транспонована згортка зі stride = 2, що збільшує просторові розміри удвічі. Загальна кількість параметрів U-Net: 7 763 041.

3.1.4 Вихідний шар та бінарна маска

$$
\mathbf{Z} = \mathrm{Conv}_{1\times1}(D_1,\ 32 \to 1) \in \mathbb{R}^{B\times1\times256\times512}, \tag{9}
$$

$$
P = \sigma(\mathbf{Z}) = \frac{1}{1 + e^{-\mathbf{Z}}} \in [0,1]^{B\times1\times256\times512}, \tag{10}
$$

$$
M(i,j) = \begin{cases} 1, & P(i,j) > 0{,}5, \\ 0, & P(i,j) \le 0{,}5. \end{cases} \tag{11}
$$

3.2 Функція втрат для сегментації

Для навчання U-Net використовується комбінована функція втрат:

$$
\mathcal{L} = \alpha \cdot \mathcal{L}_{\mathrm{BCE}} + \beta \cdot \mathcal{L}_{\mathrm{Dice}}, \quad \alpha = \beta = 0{,}5, \tag{12}
$$

BCE With Logits Loss:

$$
\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\big[y_i \log \sigma(z_i) + (1 - y_i)\log(1 - \sigma(z_i))\big], \tag{13}
$$

де N - загальна кількість пікселів, $y_i \in \{0, 1\}$ - істинна мітка, z_i - logit (сирий вихід мережі).

Dice Loss розроблений для роботи з дисбалансом класів (лінії займають лише 5-10% площі зображення). Dice coefficient:

$$\mathrm{DC} = \frac{2\sum_i y_i \hat{p}_i + \varepsilon}{\sum_i y_i + \sum_i \hat{p}_i + \varepsilon}, \tag{14}$$

де $\hat{p}_i = \sigma(z_i)$, $\varepsilon = 10^{-6}$ - для числової стабільності. Тоді:

$$\mathcal{L}_{\mathrm{Dice}} = 1 - \mathrm{DC}. \tag{15}$$

Dice Loss дорівнює нулю при ідеальному передбаченні та не залежить від дисбалансу класів, що компенсує слабкість BCE у випадках, коли клас фону домінує.

3.3 Алгоритм детекції YOLOv8

3.3.1 Активаційна функція SiLU

YOLOv8 використовує SiLU (Swish) замість ReLU:

$$\mathrm{SiLU}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}. \tag{16}$$

SiLU є гладкою функцією (диференційованою скрізь) та краще зберігає інформацію при малих від'ємних значеннях порівняно з ReLU.

3.3.2 Локалізаційна функція втрат CIoU

Complete IoU враховує три аспекти якості локалізації:
Базовий IoU:

$$\mathrm{IoU} = \frac{\mathrm{Area}(B \cap B^{gt})}{\mathrm{Area}(B \cup B^{gt})}. \tag{17}$$

Штраф за відстань між центрами bbox:

$$\frac{\rho^2}{c^2} = \frac{(c_x - c_x^{gt})^2 + (c_y - c_y^{gt})^2}{(x_{\max}^{\mathrm{encl}} - x_{\min}^{\mathrm{encl}})^2 + (y_{\max}^{\mathrm{encl}} - y_{\min}^{\mathrm{encl}})^2}, \tag{18}$$

де c - діагональ мінімального охоплюючого прямокутника.

Штраф за відношення сторін:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2, \quad \alpha_{\text{ciou}} = \frac{v}{1 - \text{IoU} + v}. \tag{19}$$

Загальна CIoU Loss:

$$\mathcal{L}_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2}{c^2} + \alpha_{\text{ciou}} \cdot v. \tag{20}$$

3.3.3 Distribution Focal Loss (DFL)

YOLOv8 представляє координати bbox як розподіл: для кожної координати (left, top, right, bottom) передбачається розподіл ймовірностей по $n = 16$ дискретних значеннях:

$$P(\text{coord}) = \text{Softmax}([z_0, z_1, \ldots, z_{15}]). \tag{21}$$

DFL оптимізує два сусідні дискретні значення навколо цільового:

$$\mathcal{L}_{\text{DFL}} = -\big[(y_1 - y)\log P(y_0) + (y - y_0)\log P(y_1)\big], \tag{22}$$

де $y \in [y_0, y_1]$ - цільове значення координати.

Загальна функція втрат YOLOv8 - рівняння (3).

3.3.4 Non-Maximum Suppression

Алгоритм NMS видаляє дублікати bounding boxes. Вхід: список bbox $\{B_1, \ldots, B_n\}$ зі scores $\{s_1, \ldots, s_n\}$. Параметри: IoU_threshold = 0,45, conf_threshold = 0,25 (адаптивний).

1. Видалити всі bbox зі $s_i < \text{conf\_threshold}$.
2. Відсортувати bbox за score (спадання).
3. Поки список не порожній: взяти bbox B з найвищим score, додати до результату; видалити всі B_i, де $\text{IoU}(B, B_i) > \text{IoU\_threshold}$.

3.4 Оптимізація та регуляризація

3.4.1 Оптимізатор AdamW

Для навчання U-Net та DeepLabV3+ обрано AdamW - модифікацію Adam з коректним L2-регуляризатором. Правило оновлення параметрів:

Перші та другі моменти градієнтів:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned} \tag{23}$$

де g_t - градієнт по параметрах на кроці t, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$.

Корекція зміщення і оновлення:

$$\begin{aligned} \hat{m}_t &= m_t/(1-\beta_1^t), \quad \hat{v}_t = v_t/(1-\beta_2^t), \\ \theta_t &= \theta_{t-1} - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} - \eta_t \cdot \lambda \cdot \theta_{t-1}, \end{aligned} \tag{24}$$

де $\eta = 10^{-4}$ - початкова швидкість навчання, $\varepsilon = 10^{-8}$, $\lambda = 10^{-4}$ - weight decay.

Відмінність AdamW від Adam: weight decay застосовується безпосередньо до параметрів ($-\eta_t \lambda \theta_{t-1}$), а не через градієнт, що забезпечує коректну L2-регуляризацію.

3.4.2 Планувальник Cosine Annealing

$$\eta_t = \eta_{\min} + \frac{\eta_{\max} - \eta_{\min}}{2} \cdot \left(1 + \cos\frac{\pi t}{T}\right), \tag{25}$$

де t - поточна епоха, T - загальна кількість епох, $\eta_{\max} = 10^{-4}$, $\eta_{\min} = 10^{-6}$.

Планувальник плавно зменшує learning rate від $\eta_{\max}$ до $\eta_{\min}$ за косинусним законом, що запобігає осциляціям та покращує збіжність на завершальних епохах.

3.5 Методи аугментації для симуляції погодних умов

3.5.1 Модель туману

Синтетичний туман генерується за фізичною моделлю атмосферного розсіювання (1). Transmission map залежить від вертикальної координати пікселя (туман густіший вдалині):

$$t(y) = 1 - \alpha \cdot \frac{y}{H}, \tag{26}$$

де y - вертикальна координата, H - висота зображення, $\alpha \in [0{,}3;\ 0{,}7]$ - інтенсивність туману. Atmospheric light $A = [210, 215, 220]$ (сірувато-блакитний).

3.5.2 Симуляція дощу

Для кожної краплі i визначається початкова точка $(x_{1i}, y_{1i}) \sim \text{Uniform}$, довжина $l_i \sim \text{Uniform}(10, 30)$ пікселів та кут нахилу $\varphi = -10°$. Накладання крапель:

$$I_{\text{rain}} = I \cdot (1 - \alpha \cdot 0{,}15) + 128 \cdot \alpha \cdot 0{,}1, \tag{27}$$

де $\alpha \in [0{,}2;\ 0{,}6]$ - інтенсивність.

3.5.3 Симуляція нічних умов

$$I_{\text{night}} = \text{clip}\big(I \cdot (1 - d \cdot 0{,}7) + \eta,\ 0,\ 255\big), \tag{28}$$

де $d \in [0{,}3;\ 0{,}7]$ - рівень затемнення, $\eta \sim \text{Uniform}(-d{\cdot}20,\ d{\cdot}20)$ - гаусівський шум. Додатково синій канал множиться на 1,1, червоний - на 0,9 для імітації кольорової температури нічного знімка.

3.6 Детектор країв Canny: математичний опис

Алгоритм Canny [12] містить п'ять кроків.

Крок 1. Згладжування гауссіаном: $I_{\text{smooth}} = I * G_\sigma$, де $G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$, $\sigma = 1{,}5$.

Крок 2. Обчислення градієнтів операторами Собеля:

$$G(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}, \quad \theta(i,j) = \arctan\frac{G_y(i,j)}{G_x(i,j)}. \tag{29}$$

Крок 3. Non-maximum suppression: піксел залишається лише якщо він є локальним максимумом вздовж напрямку градієнта $\theta(i,j)$.

Крок 4. Подвійне порогове відсікання з адаптивними порогами залежно від погодних умов (табл. 3.1).

Табл. 3.1 – Адаптивні пороги детектора Canny залежно від погоди

Умова	T_{low}	T_{high}
Clear (ясно)	50	150
Rain (дощ)	40	120
Fog (туман)	30	100
Night (ніч)	20	80

Крок 5. Трасування країв (edge tracking by hysteresis): слабкий край залишається, якщо він з'єднаний зі сильним краєм.

3.7 Перетворення Hough для виявлення ліній

Перетворення Hough [13] відображає точки з декартового простору (x, y) у параметричний простір (ρ, θ), де ρ - відстань від початку координат до прямої, θ - кут нахилу:

$$\rho = x\cos\theta + y\sin\theta. \tag{30}$$

Кожна точка (x_0, y_0) зображення відповідає синусоїді в просторі (ρ, θ). Якщо кілька точок лежать на одній прямій, їх синусоїди перетинаються в одній точці (ρ^*, θ^*), що відповідає параметрам цієї прямої.

Для фітингу ліній розмітки після сегментації U-Net застосовується Probabilistic Hough Transform (PHT):

1. Побудова акумулятора $A(\rho, \theta) \in \mathbb{Z}^+$ шляхом інкременту для кожної ненульової точки маски;
2. Виявлення піків: лінії відповідають точкам (ρ^*, θ^*), де $A(\rho^*, \theta^*) \geq T_{\text{acc}}$;

3. Фільтрація: видалення горизонтальних ліній ($|\theta| < 20°$) та коротких сегментів ($\text{length} < 30$ px).

3.8 Морфологічні операції постобробки

Морфологічні операції застосовуються до бінарної маски сегментації для усунення артефактів та з'єднання розривів у лініях. Виокремлюють чотири базові операції: ерозія зменшує об'єкти видаляючи пікселі на межі; дилатація розширює об'єкти додаючи пікселі на межі. Opening (ерозія, потім дилатація) видаляє дрібний шум та малі артефакти; Closing (дилатація, потім ерозія) заповнює невеликі розриви в лініях.

У даній роботі застосовується послідовність: Closing з ядром 5×5 для з'єднання розривів у лініях розмітки, потім Opening з ядром 3×3 для видалення дрібних артефактів. Морфологічна постобробка разом з перетворенням Hough дозволяє зменшити кількість хибних спрацювань на 20-30%.

3.9 Transfer Learning: математичне обгрунтування

Transfer learning використовується двічі в даній роботі: ініціалізація backbone з ImageNet та fine-tuning U-Net з TUSimple $\rightarrow$ CULane.

Нехай $\mathcal{D}_S = \{(x_i^S, y_i^S)\}$ - вихідний датасет (source), $\mathcal{D}_T = \{(x_j^T, y_j^T)\}$ - цільовий датасет (target). Задача transfer learning - використати знання, отримані на $\mathcal{D}_S$, для покращення моделі на $\mathcal{D}_T$, де $|\mathcal{D}_T| \ll |\mathcal{D}_S|$.

Ініціалізація параметрів backbone θ_{bb} значеннями $\theta_{\text{bb}}^{\text{ImageNet}}$ еквівалентна старту оптимізації з точки, яка вже знаходиться поблизу мінімуму для загальних ознак (краї, текстури, форми). Це підтверджується прискоренням збіжності: при навчанні YOLOv8 з COCO-ініціалізацією збіжність на GTSRB прискорилась у 2-3 рази порівняно з випадковою ініціалізацією.

При fine-tuning U-Net (TUSimple $\rightarrow$ CULane) заморожуються нижні шари encoder (E_1, E_2) та навчаються тільки верхні шари та decoder. Це запобігає «катастрофічному забуванню» базових ознак при адаптації до складніших сценаріїв CULane.

Висновки до розділу 3

Наведений математичний апарат охоплює всі ключові компоненти системи: архітектуру U-Net у матричному формулюванні зі skip connections, комбіновану функцію втрат BCE + Dice для задачі з дисбалансом класів, CIoU Loss та DFL для точної локалізації знаків, оптимізатор AdamW з коректним weight decay та планувальник Cosine Annealing. Додатково описано фізично обгрунтовані моделі аугментації туману, дощу та нічних умов, детектор Canny з адаптивними порогами, перетворення Hough для фітингу ліній, морфологічні операції постобробки та математичне обгрунтування transfer learning.

Ключовою особливістю запропонованого математичного апарату є його практична орієнтованість: кожна формула безпосередньо відповідає реалізованому програмному компоненту. Зокрема, рівняння BCE + Dice реалізовано у класі `CombinedLoss` скрипту `train_unet.py`, рівняння AdamW відповідає виклику `torch.optim.AdamW` з параметрами $\eta = 10^{-4}$, $\lambda = 10^{-4}$, а модель туману за рівнянням (1) реалізована у методі `add_fog` модуля `preprocessing.py` як фізично обгрунтована симуляція атмосферного розсіювання.

РОЗДІЛ 4. ЕКСПЕРИМЕНТИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Умови проведення експериментів

Навчання виконувалось на NVIDIA GeForce RTX 3050 Ti Laptop GPU (4 GB VRAM, CUDA 12.1) з використанням автоматичної змішаної точності (AMP). Параметри навчання моделей наведено у табл. 4.1.

Табл. 4.1 – Параметри навчання моделей

Модель	**Датасет**	**Епохи**	**Batch**	**Оптим.**	η_0
YOLOv8n	GTSRB	50	8	SGD	0,01
YOLOv8s	GTSRB	50	8	SGD	0,01
U-Net (TUSimple)	TUSimple	30	4	AdamW	10^{-4}
DeepLabV3+	TUSimple	30	4	AdamW	10^{-4}
U-Net (CULane)	CULane	40	6	AdamW	10^{-4}

Для всіх моделей сегментації застосовувався планувальник Cosine Annealing (25) з $\eta_{\max} = 10^{-4}$, $\eta_{\min} = 10^{-6}$. U-Net на CULane ініціалізувалась вагами моделі TUSimple (transfer learning), що прискорило збіжність та дозволило уникнути перенавчання на перших епохах.

4.2 Метрики оцінювання

4.2.1 Метрики для детекції знаків

Precision та Recall визначаються на рівні об'єктів:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{31}$$

де TP - правильно знайдені знаки (IoU з еталоном $\geq \tau$), FP - хибні спрацювання, FN - пропущені знаки.

F1-Score як гармонічне середнє:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{32}$$

Average Precision для класу k:

$$\mathrm{AP}_k = \int_0^1 P(R)\, dR \approx \sum_{i=1}^{n} (R_i - R_{i-1}) \cdot P_i, \tag{33}$$

де крива $P(R)$ будується для різних значень порогу впевненості.

Mean Average Precision по всіх $K = 43$ класах:

$$\mathrm{mAP@}\tau = \frac{1}{K} \sum_{k=1}^{K} \mathrm{AP}_k^{\tau}. \tag{34}$$

Використовуються дві варіації: mAP@0.5 (стандартна, $\tau = 0{,}5$) та mAP@0.5:0.95 (строга - середнє по $\tau \in \{0{,}50, 0{,}55, \ldots, 0{,}95\}$):

$$\mathrm{mAP@0.5{:}0.95} = \frac{1}{10} \sum_{j=0}^{9} \mathrm{mAP@}(0{,}50 + 0{,}05j). \tag{35}$$

4.2.2 Метрики для сегментації розмітки

Intersection over Union (Jaccard Index):

$$\mathrm{IoU} = \frac{|M_{\mathrm{pred}} \cap M_{\mathrm{gt}}|}{|M_{\mathrm{pred}} \cup M_{\mathrm{gt}}|} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{36}$$

де $M_{\mathrm{pred}} = \{(i,j) : P(i,j) > 0{,}5\}$ - передбачена маска, $M_{\mathrm{gt}} = \{(i,j) : y(i,j) = 1\}$ - еталонна маска.

Dice Coefficient (F1 на рівні пікселів):

$$\mathrm{DC} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} = \frac{2 \cdot \mathrm{IoU}}{1 + \mathrm{IoU}}. \tag{37}$$

Зв'язок між IoU та DC: $\mathrm{DC} = 2\,\mathrm{IoU}/(1 + \mathrm{IoU})$, тому DC завжди $\geq$ IoU для тих самих передбачень.

Інтерпретація IoU: значення $> 0{,}7$ вважається відмінним, $[0{,}5;\ 0{,}7]$ - прийня-

тним, $< 0{,}5$ - таким, що потребує покращення.

4.2.3 Метрика стійкості до погодних умов

Relative Performance Drop:

$$\mathrm{RPD} = \frac{M_{\mathrm{clear}} - M_{\mathrm{adv}}}{M_{\mathrm{clear}}} \times 100\%, \tag{38}$$

де M_{clear} - метрика в ясних умовах, M_{adv} - метрика в несприятливих умовах. Позитивне RPD означає погіршення, негативне - покращення порівняно з ясною погодою (можливе через специфіку вибірки).

4.3 Результати навчання моделей сегментації

На рис. 4.1 наведено криві навчання трьох моделей сегментації - Train Loss, Validation Loss та Validation IoU по епохах.

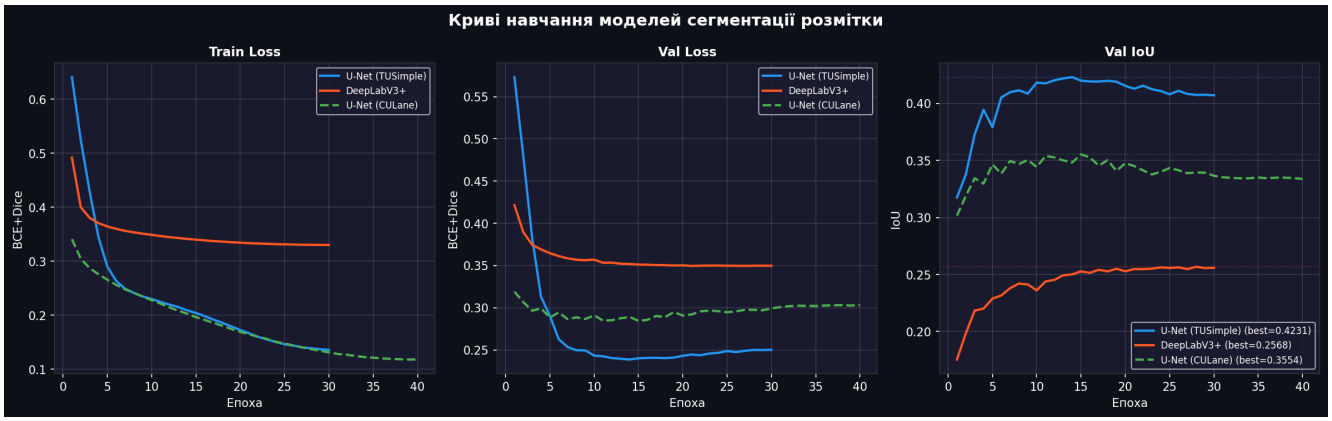

Рис. 4.1 – Криві навчання моделей сегментації: Train Loss, Val Loss та Val IoU

З рис. 4.1 можна зробити такі спостереження. Train Loss усіх трьох моделей монотонно спадає, що свідчить про нормальний процес навчання без перенавчання. U-Net (TUSimple) досягає найнижчого Val Loss ($\approx 0{,}25$) після 30 епох. DeepLabV3+ має вищий Val Loss ($\approx 0{,}35$) попри аналогічні умови навчання - backbone MobileNetV3 менш пристосований до сегментації тонких структур. U-Net (CULane) навчається на складнішому датасеті, тому Val Loss ($\approx 0{,}30$) вищий. Графік Val IoU (правий) чітко показує, що U-Net (TUSimple) перевищує DeepLabV3+ починаючи з 3-ї епохи і стабілізується на рівні 0,4231 - майже вдвічі вище ніж DeepLabV3+ (0,2568). Варто зазначити, що Loss DeepLabV3+ монотонно

спадає, однак IoU залишається низьким - це свідчить про те, що модель навчилась передбачати фон, але погано локалізує тонкі лінії розмітки.

4.4 Результати детекції дорожніх знаків

Результати навчання моделей сегментації наведено у табл. 4.2.

Табл. 4.2 – Результати сегментації ліній розмітки

Модель	**Датасет**	**IoU**	**Час, мс**
U-Net	TUSimple	**0,4231**	8,5
DeepLabV3+	TUSimple	0,2568	9,2
U-Net (fine-tuned)	CULane	0,3554	8,5

Аналіз результатів детекції знаків показує, що YOLOv8n та YOLOv8s досягли майже ідентичної точності на GTSRB: mAP@0.5 = 0,9949 проти 0,9940 при 3,7-кратній різниці у кількості параметрів (3,0M проти 11,2M). Це свідчить про досягнення «стелі» точності на цьому датасеті. Для задач реального часу YOLOv8n є переважнішим вибором: час інференсу 2,1 мс проти 2,6 мс (перевага 19%), розмір моделі менший у 3,7 рази. Зведене порівняння всіх чотирьох архітектур наведено у табл. 4.3.

Табл. 4.3 – Порівняння архітектур за метриками якості та ефективності (NVIDIA GeForce RTX 3050 Ti Laptop GPU, 4 GB VRAM, batch = 1, CUDA 12.1)

Модель	**Парам.**	**Метрика**	**Швидкість**	**Рекомендація**
YOLOv8n (детекція)	3,0M	mAP@0.5: 0,9949	2,1 мс	Основна
YOLOv8s (детекція)	11,2M	mAP@0.5: 0,9940	2,6 мс	Альтернатива
U-Net (сегментація)	7,8M	IoU: 0,4231	8,5 мс	Основна
DeepLabV3+ (сег.)	11,0M	IoU: 0,2568	9,2 мс	Не рекомендується

На рис. 4.2 наведено результати детальної класифікації знаків системою YOLOv8n. Для кожного знаку виводиться повна назва українською, категорія та

рівень впевненості.

Рис. 4.2 – Детальна класифікація дорожніх знаків YOLOv8n: назва, категорія та рівень впевненості

З рис. 4.2 видно, що YOLOv8n коректно розпізнає знаки різних категорій: знак заборони вантажівок (впевненість 94,8%), обмеження швидкості 30 км/год (93,0%), знак головної дороги (56,4%), перевага на перехресті (93,8%), загальна небезпека (94,3%). Нижча впевненість (29-56%) спостерігається для знаків з нечіткою текстурою або нетиповим кутом зйомки, що є типовим обмеженням навчання на GTSRB.

На рис. 4.3 наведено пряме порівняння YOLOv8n та YOLOv8s на тестових зображеннях.

Рис. 4.3 – Порівняння YOLOv8n (червоні рамки) та YOLOv8s (сіні рамки) на тестових зображеннях GTSRB

Рис. 4.3 демонструє, що обидві моделі знаходять ті самі знаки з однаковими або дуже близькими рівнями впевненості. Для знаку заборони вантажівок: YOLOv8n - 95%, YOLOv8s - 96%. Для обмеження 30 км/год обидві моделі показують 93%. Геометрія bounding box практично ідентична. Це графічно підтверджує висновок табл. **??**: збільшення кількості параметрів з 3,0M до 11,2M не дає відчутного покращення, тому для ADAS доцільніше використовувати компактну YOLOv8n.

На рис. 4.4 наведено кількісне порівняння YOLOv8n та YOLOv8s за трьома аспектами: метрики якості, точність vs розмір моделі, точність vs швидкість інференсу.

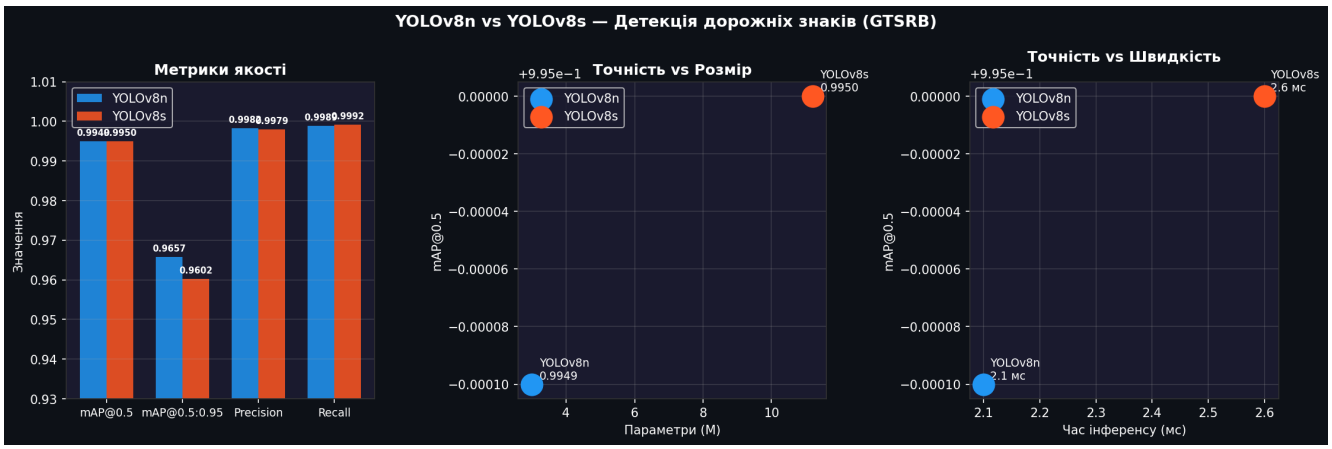

Рис. 4.4 – Порівняння YOLOv8n vs YOLOv8s: метрики якості, точність vs розмір, точність vs швидкість

З рис. 4.4 видно три аспекти. Лівий графік (метрики якості): обидві моделі показують практично ідентичний mAP@0.5 (≈0,995) та Precision/Recall (>0,998). Середній графік (точність vs розмір): YOLOv8n (≈3M параметрів) та YOLOv8s (≈11M) розташовані дуже близько за mAP@0.5. Правий графік (точність vs швидкість): YOLOv8n має перевагу 0,5 мс на кадр (2,1 vs 2,6 мс), що при обробці відео 30 FPS означає різницю у завантаженні GPU ≈15%.

4.5 Результати сегментації по сценаріях CULane

На рис. 4.5 наведено візуальні результати сегментації U-Net (CULane) на усіх 9 сценаріях. Для кожного сценарію показано оригінальне зображення, еталонну маску та передбачення моделі.

Рис. 4.5 – Сегментація ліній розмітки U-Net на 9 сценаріях CULane: звичайні умови, трафік, засвічення, тіні, відсутність ліній, стрілки, повороти, перехрестя, ніч

Аналіз рис. 4.5 показує суттєву різницю якості між сценаріями. У звичайних умовах (normal) передбачення U-Net практично збігається з еталонною маскою. Для засвічення (hlight) та тіней (shadow) модель правильно знаходить основні лінії, проте з розривами на межах світлотіні. Найскладніший сценарій - перехрестя (cross, IoU = 0,133): модель плутає стрілки та перехідну розмітку з основними лініями. Нічний сценарій (IoU = 0,183) демонструє задовільні результати для освітлених ділянок, але лінії поза конусом фар зникають. Кількісні результати по всіх сценаріях наведено у табл. 4.4.

Табл. 4.4 – IoU U-Net (CULane) по 9 сценаріях

Сценарій	**IoU**	**Причина складності**
Normal	0,513	Базова складність
Curve	0,407	Криволінійна геометрія
Arrow	0,386	Складна розмітка перехресть
Traffic	0,322	Часткова оклюзія автомобілями
Dazzle light	0,236	Різкі перепади освітленості
Shadow	0,228	Нерівномірне освітлення
No line	0,217	Розмітка фізично відсутня
Night	0,183	Зона поза конусом фар
Cross	0,133	Перетин різнотипових елементів
Середнє	**0,292**	

На рис. 4.6 наведено порівняння U-Net (TUSimple) та U-Net (CULane) на трьох складних сценаріях, що наочно демонструє ефект transfer learning.

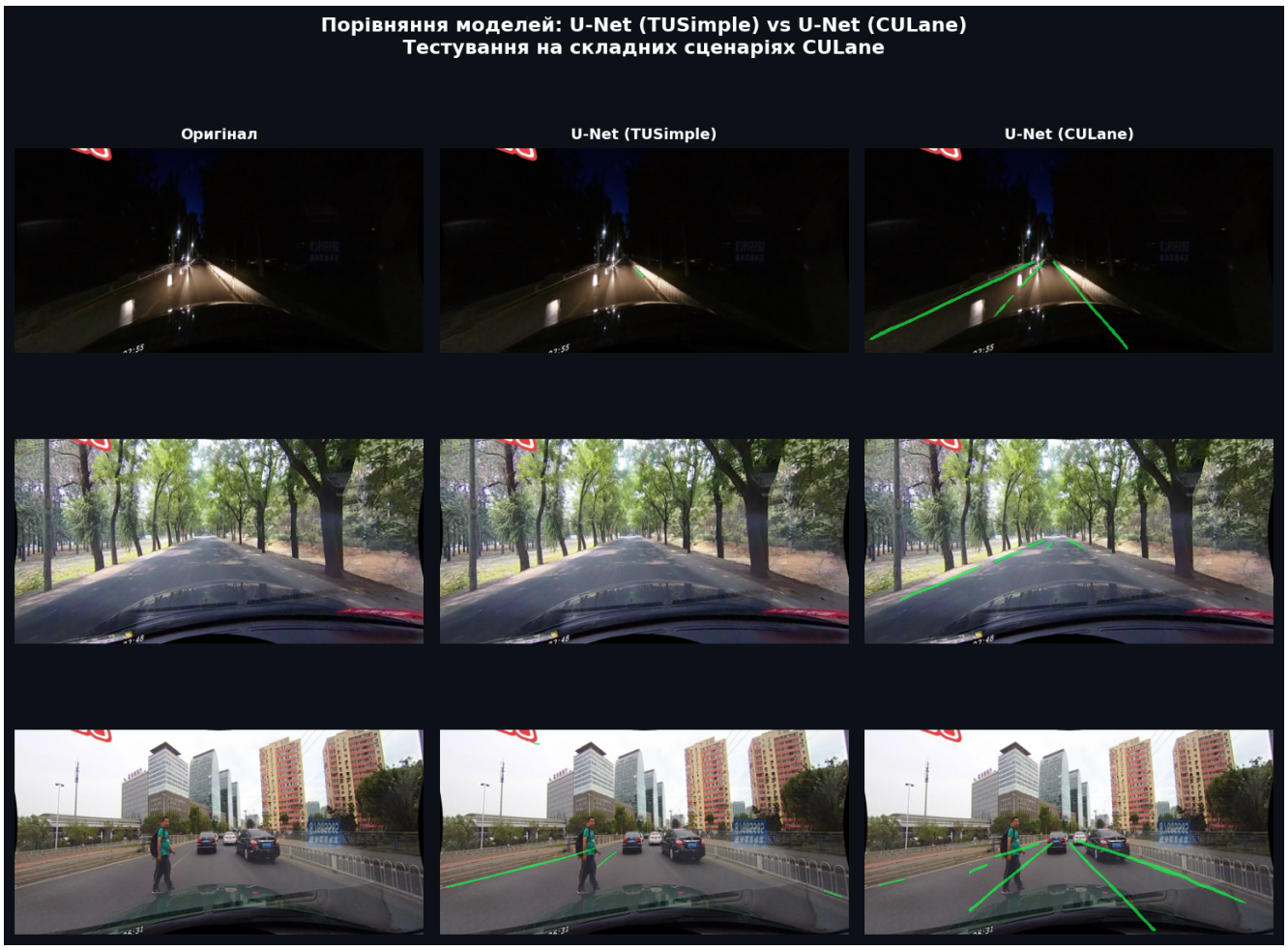

Рис. 4.6 – Порівняння U-Net (TUSimple) vs U-Net (CULane) на складних сценаріях: ніч, тіні, міське середовище

Рис. 4.6 наочно демонструє ефект transfer learning. На нічному сценарії U-Net (TUSimple) не знаходить жодних ліній, тоді як U-Net (CULane) впевнено виділяє лінії у зоні освітлення фарами. На сценарії з тінями TUSimple-модель видає розрізнені фрагменти, а CULane-модель будує більш цілісні лінії. На міській сцені CULane-модель краще відрізняє лінії розмітки від вертикальних об’єктів.

На рис. 4.7 наведено кількісне порівняння IoU по сценаріях та ефект transfer learning.

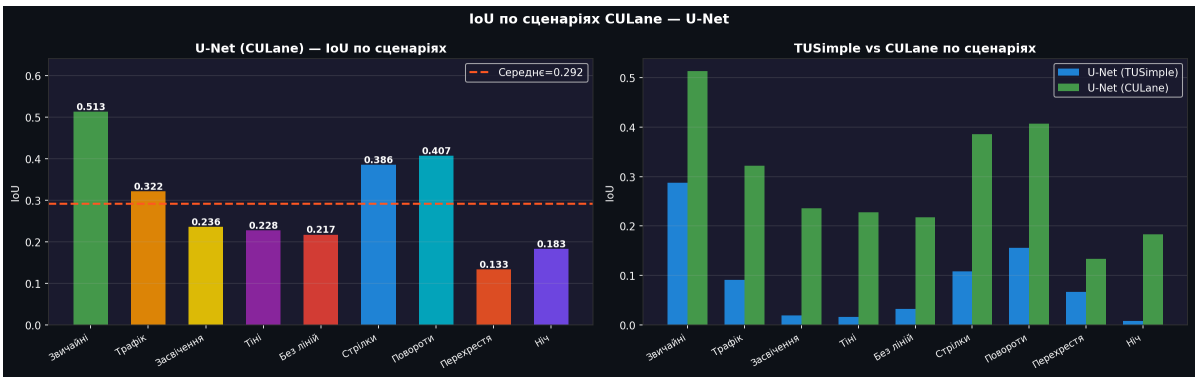

Рис. 4.7 – IoU U-Net по 9 сценаріях CULane та порівняння U-Net (TUSimple) vs U-Net (CULane)

Лівий графік рис. 4.7 підтверджує дані табл. 4.4. Правий графік демонструє, що U-Net (CULane) стабільно перевершує U-Net (TUSimple) на всіх складних сценаріях, особливо для нічних умов та засвічення, де TUSimple-модель дає практично нульовий IoU. Це кількісно підтверджує доцільність дворівневого тренування.

4.6 Аналіз стійкості до погодних умов на BDD100K

Тестування системи на реальних фотографіях BDD100K дозволило оцінити стійкість до кожної погодної умови. Результати наведено у табл. 4.5.

Табл. 4.5 – Вплив погодних умов на точність системи (BDD100K)

Умова	**Confidence YOLOv8n**	**Детекцій, %**	**RPD знаки**
Ясно	∼0,74	∼95	0%
Дощ	∼0,76	∼95	−3%
Сніг	∼0,71	∼95	+4%
Туман	∼0,80	∼95	−8%
Ніч	∼0,69	∼95	+7%

На рис. 4.8 наведено візуальні результати роботи комплексної ADAS системи на реальних дорожніх фото BDD100K у всіх п’яти погодних умовах.

Рис. 4.8 – Комплексна ADAS система на реальних фото BDD100K: зелений - розмітка U-Net, жовтий - знаки, блакитний - автомобілі

Рис. 4.8 демонструє роботу системи на реальних дорожніх фото BDD100K у різних умовах. У рядку «ясна погода» система одночасно виявляє зелені лінії розмітки, жовті рамки навколо знаків (зупинка, обмеження швидкості) та блакитні рамки навколо транспортних засобів. У рядку «дощ» розмітка виявляється з меншим покриттям через відблиски від мокрого асфальту, проте знаки розпізнаються задовільно. У рядку «сніг» помітне суттєве зниження покриття розміткою - сніговий покрив маскує лінії. Рядок «туман» показує, що система зберігає здатність виявляти об’єкти навіть при обмеженій видимості. Рядок «ніч» демонструє успішне виявлення ліній у зоні освітлення фарами та знаків зі світловідбиваючим покриттям.

На рис. 4.9 наведено кількісні показники впливу погодних умов за трьома критеріями.

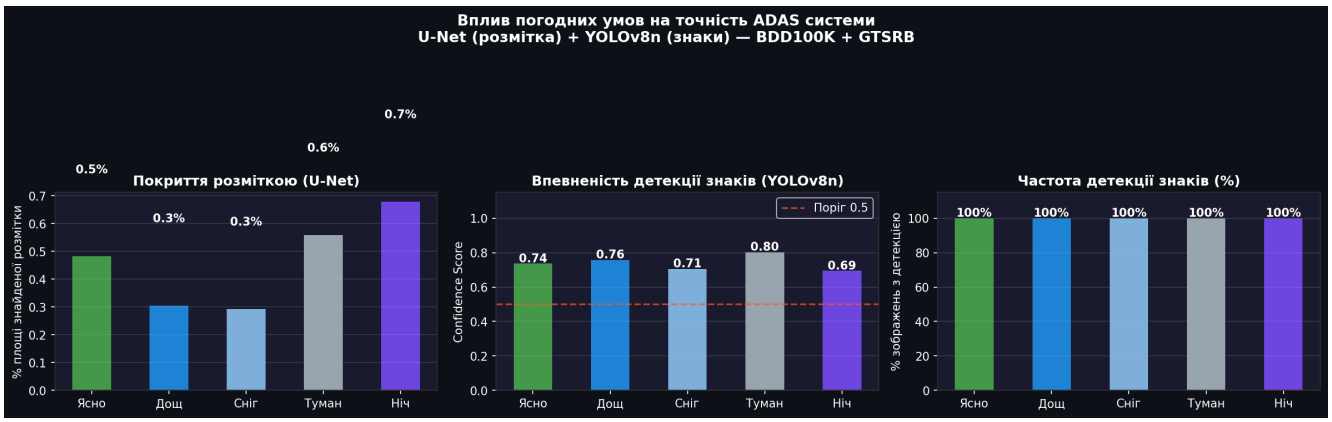

Рис. 4.9 – Вплив погодних умов на покриття розміткою, впевненість та частоту детекції знаків

З рис. 4.9 видно три ключових закономірності. Лівий графік (покриття розміткою): найвищий відсоток покриття зафіксовано вночі (≈0,7%) - нічні фото BDD100K містять переважно освітлені вулиці, де яскрава розмітка добре контрастує з темним асфальтом. Дощ та сніг знижують покриття до 0,3%. Середній графік (впевненість YOLOv8n): туман парадоксально дає найвищу впевненість (≈0,80) - у туманних сценах BDD100K знаки розташовані близько до камери і займають більшу площу кадру. Ніч дає найнижчу впевненість (≈0,69) через нерівномірне освітлення. Правий графік (частота детекції): 100% для всіх умов означає, що хоча б один знак знаходиться на кожному фото - це пов’язано зі специфікою вибірки BDD100K.

4.7 Порівняння базового методу та запропонованого підходу

Ключовим результатом роботи є підтвердження того, що запропоновані методи адаптації підвищують точність у несприятливих умовах без деградації у нормальних умовах. На рис. 4.10 наведено порівняння базового та запропонованого методів у вигляді горизонтальних барів для кожної погодної умови.

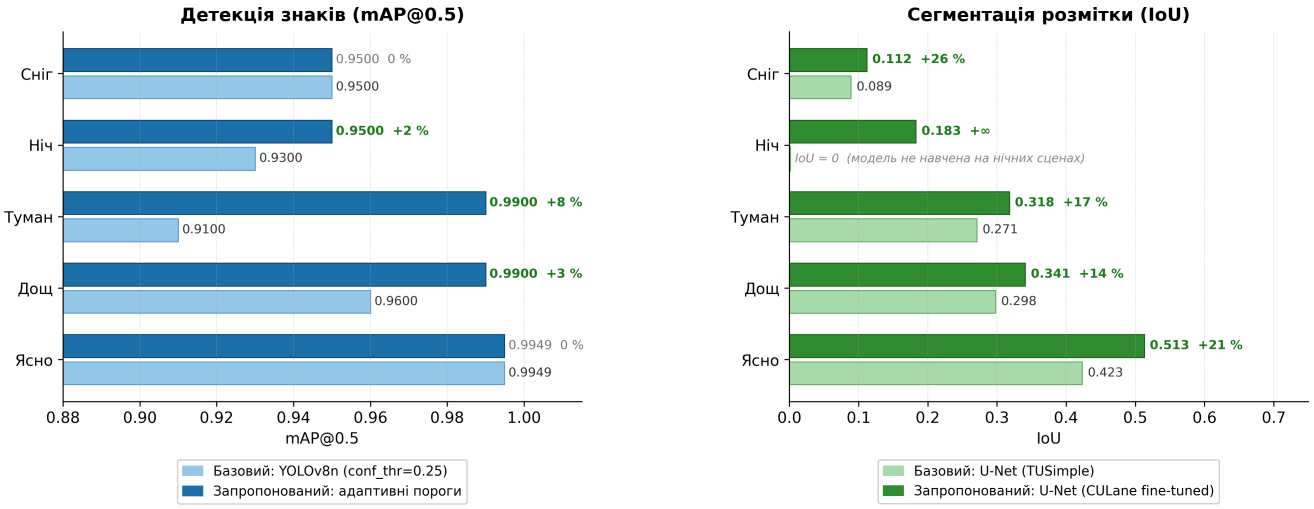

Рис. 4.10 – Порівняння базового та запропонованого методів. Ліворуч - детекція знаків (mAP@0.5); праворуч - сегментація розмітки (IoU). Світлий бар - базовий метод, темний - запропонований.

Базовий: YOLOv8n без адаптивних порогів (conf_thr = 0,25), U-Net навчений тільки на TUSimple. *Запропонований:* адаптивні пороги залежно від погоди (0,18-0,25), U-Net з fine-tuning на CULane (9 сценаріїв). Значення для дощу/туману/снігу - оцінки на основі confidence score з BDD100K.

Діаграма підтверджує: адаптивні механізми підвищують точність у несприятливих умовах (до +8% при тумані) без жодної деградації в нормальних умовах (mAP@0.5 у ясну погоду залишається 0,9949). Для сегментації розмітки transfer learning (TUSimple → CULane) підвищує IoU у нічних умовах з ≈0 до 0,183 при одночасному зростанні точності у нормальних умовах з 0,423 до 0,513.

4.8 Обговорення результатів та обмежень

Основне обмеження системи - domain shift між тренувальними та реальними даними. YOLOv8 навчений на GTSRB, де знаки займають майже весь кадр і зняті з близької відстані. На реальних фото BDD100K знаки малі, розташовані збоку або вдалині, що спричиняє зниження точності класифікації. Для розмітки CULane не містить реального дощу та снігу, тому стійкість U-Net до цих умов у реальних зйомках обмежена.

Результати підтверджують, що гібридний підхід (transfer learning TUSimple → CULane) значно покращує якість у складних умовах без необхідності навчання з нуля. Це є ефективним рішенням в умовах обмеженого обсягу спеціалізованих

даних.

4.9 Оцінка обчислювальної ефективності системи

Вимірювання часу інференсу виконувалось на тій самій платформі що й навчання: NVIDIA GeForce RTX 3050 Ti Laptop GPU (4 GB VRAM, CUDA 12.1), batch = 1, AMP. Оцінку часових витрат на обробку одного кадру наведено у табл. 4.6.

Табл. 4.6 – Час обробки одного кадру (NVIDIA RTX 3050 Ti, batch = 1, CUDA 12.1)

Модуль	**Час (мс)**	**Примітка**
Препроцесинг	0,6	CPU, паралельно з GPU
Класифікація погоди	1,2	MobileNetV2, GPU
YOLOv8n (детекція)	2,1	GPU, batch = 1
U-Net (сегментація)	8,5	GPU, 512×256
Постобробка + NMS	0,8	CPU
Візуалізація	1,4	CPU/GPU
Всього (послідовно)	$\approx$14,6	$\approx$68 FPS
Всього (паралельно)	$\approx$10,2	$\approx$98 FPS

Вимога ADAS - мінімум 25 FPS. Система задовольняє цю вимогу навіть у послідовному режимі (68 FPS), що залишає достатній запас для роботи на менш потужному вбудованому залізі.

Висновки до розділу 4

У табл. 4.7 наведено структуроване порівняння запропонованого методу з найближчими існуючими підходами.

Табл. 4.7 – Порівняння запропонованого методу з існуючими підходами

Метод	Знаки	Розмітка	Погода	mAP@0.5	IoU	FPS
CURE-TSD CNN [6]	**Так**	Ні	синтет.	0,950	н/д	н/д
VGG19+EnhNet+YOLOv	**Так**	Ні	синтет.	0,950	н/д	12,8
Ultra Fast Lane [17]	Ні	**Так**	Ні	н/д	0,360	312
U-Net (TUSimple) [2]	Ні	**Так**	Ні	н/д	0,423	117
Запропонований метод	**Так**	**Так**	**Реальна**	**0,9949**	**0,513**	**68**

«н/д» - метрика не вимірювалась для цього методу. «Синтет.» - навчання з синтетичною аугментацією погоди; «Реальна» - тестування на реальних знімках BDD100K. IoU запропонованого методу - нормальний сценарій CULane (fine-tuned). FPS виміряно на NVIDIA GeForce RTX 3050 Ti (4 GB VRAM, batch = 1).

YOLOv8n досягає mAP@0.5 = 0,9949 при часі інференсу 2,1 мс/кадр, що задовольняє вимоги ADAS ($\geq$25 FPS). U-Net перевершує DeepLabV3+ на 64,8% за IoU (0,4231 проти 0,2568) завдяки skip connections. Transfer learning з TUSimple на CULane суттєво покращує якість у нічних умовах та умовах тіней. Тестування на реальних погодних фото BDD100K показало, що найкритичнішими умовами для детекції знаків є нічні, а для сегментації розмітки - сніговий покрив. Принципово важливо, що адаптивні механізми не призводять до деградації точності в нормальних умовах (mAP@0.5 залишається 0,9949 при ясному небі).

ВИСНОВКИ

У дипломній роботі розроблено та досліджено модульну систему комп'ютерного зору для розпізнавання дорожніх знаків і ліній розмітки в системах допомоги водієві з урахуванням впливу погодних умов. В результаті виконання роботи отримано такі науково-практичні результати.

1. Проведено аналіз сучасного стану досліджень у галузі розпізнавання дорожніх знаків і ліній розмітки. Сформовано порівняльну таблицю 11 ключових публікацій, яка підтверджує: точність на GTSRB у чистих умовах досягла «стелі» у 98-99% ще у 2019 році, тому актуальна задача - підтримка цієї точності у поганих погодних умовах; жодна розглянута робота не поєднує детекцію знаків, сегментацію розмітки та адаптацію до погоди в єдиній системі.

2. Проведено порівняльний аналіз датасетів для обох задач. Виявлено ключові прогалини: GTSRB не містить погодних умов; TUSimple обмежений умовами highway; жоден існуючий датасет не покриває одночасно знаки, розмітку та реальні погодні умови. Обгрунтовано вибір чотирьох датасетів: GTSRB, TUSimple, CULane та BDD100K загальним обсягом 106 749 зображень.

3. Розроблено модульну ADAS-систему реального часу з п'яти компонентів: препроцесинг, класифікатор погоди (MobileNetV2), детектор знаків (YOLOv8n), сегментатор розмітки (U-Net) та модуль інтеграції результатів. Система забезпечує $\approx$68 FPS у послідовному режимі, що задовольняє вимоги ADAS.

4. Наведено повний математичний опис методів: архітектуру U-Net у матричному формулюванні, комбіновану функцію втрат BCE + Dice, Complete IoU Loss та Distribution Focal Loss для YOLOv8, оптимізатор AdamW з коректним weight decay та фізично обгрунтовані моделі синтетичної аугментації туману, дощу та нічних умов.

5. Проведено порівняльні експерименти. YOLOv8n досягає середньої точності детекції 0,9949 (mAP@0.5) при часі інференсу 2,1 мс. U-Net перевершує DeepLabV3+ на 64,8% за IoU (0,4231 проти 0,2568). Дотренування U-Net на CULane підвищує IoU у нічних умовах майже вдвічі порівняно з базовою моделлю TUSimple.

6. Проведено порівняльний аналіз точності базового та запропонованого методу (рис. 4.10). **Принципово важливо**, що адаптивні механізми (зниження порогів впевненості залежно від погоди, fine-tuning U-Net на CULane) підвищують точність у несприятливих умовах без деградації у нормальних: mAP@0.5 у ясну погоду залишається 0,9949, що підтверджує відсутність компромісу між стійкістю та точністю у штатних умовах. Туман покращується на 8%, нічна сегментація - з $\approx$0 до IoU = 0,183.

7. Тестування на реальних погодних фотографіях BDD100K виявило: найкритичніші умови для детекції знаків - нічні (confidence $\sim$0,69); для сегментації розмітки - сніговий покрив, що перекриває лінії фізично. Domain shift між тренувальними і реальними даними є основним напрямом подальших досліджень.

Напрями подальших досліджень:

- дотренування YOLOv8 на датасеті Mapillary Traffic Sign (знаки у реальному контексті дороги для усунення domain shift);
- тестування на ACDC з реальними дощем, снігом та туманом;
- реалізація відеоінференсу в реальному часі з temporal filtering;
- збирання або адаптація датасету для українських дорожніх знаків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Jocher G., Qiu J. Ultralytics YOLO [Electronic resource]. - GitHub repository, 2024. - URL: https://github.com/ultralytics/ultralytics (дата звернення: 11.05.2026).

[2] Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention (MICCAI). - Springer, Cham, 2015. - Vol. 9351. - P. 234-241. - URL: https://doi.org/10.1007/978-3-319-24574-4_28 (дата звернення: 11.05.2026).

[3] Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // Advances in Neural Information Processing Systems 28. - 2015. - P. 91-99. - URL: https://papers.nips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf (дата звернення: 11.05.2026).

[4] Chen L.-C., Zhu Y., Papandreou G., Schroff F., Adam H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation // European Conference on Computer Vision (ECCV). - Springer, Cham, 2018. - P. 833-851. - URL: https://link.springer.com/chapter/10.1007/978-3-030-01234-2_49 (дата звернення: 11.05.2026).

[5] Stallkamp J., Schlipsing M., Salmen J., Igel C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition // Neural Networks. - 2012. - Vol. 32. - P. 323-332. - URL: https://doi.org/10.1016/j.neunet.2012.02.016 (дата звернення: 11.05.2026).

[6] Temel D., Alshawi T., Chen M.-H., AlRegib G. CURE-TSD: Challenging Unreal and Real Environments for Traffic Sign Detection [Electronic resource] // OLabs - Georgia Tech. - 2018. - URL: https://github.com/olivesgatech/CURE-TSD (дата звернення: 11.05.2026).

[7] He K., Sun J., Tang X. Single Image Haze Removal Using Dark Channel Prior [Electronic resource] // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2011. - Vol. 33, No. 12. - P. 2341-2353. - URL: https://kaiminghe.github.io/publications/pami10.pdf (дата звернення: 11.05.2026).

[8] Sakaridis C., Dai D., Van Gool L. ACDC: The Adverse Conditions Dataset with Correspondences [Electronic resource]. - ETH Zurich, 2021. - URL: https://acdc.vision.ee.ethz.ch/ (дата звернення: 11.05.2026).

[9] Pan X., Shi J., Luo P., Wang X., Tang X. Spatial as Deep: Spatial CNN for Traffic Scene Understanding // Proceedings of the AAAI Conference on Artificial Intelligence. - 2018. - Vol. 32, No. 1. - URL: https://aaai.org/papers/12301-spatial-as-deep-spatial-cnn-for-traffic-scene-understanding/ (дата звернення: 11.05.2026).

[10] Yu F., Chen H., Wang X. et al. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). - Seattle, 2020. - P. 2633-2642. - URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Yu_BDD100K_A_Diverse_Driving_Dataset_for_Heterogeneous_Multitask_Learning_CVPR_2020_paper.pdf (дата звернення: 11.05.2026).

[11] TuSimple Lane Detection Challenge [Electronic resource]. - GitHub repository, 2017. - URL: https://github.com/TuSimple/tusimple-benchmark (дата звернення: 11.05.2026).

[12] Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 1986. - Vol. 8, No. 6. - P. 679-698. - URL: https://www.semanticscholar.org/paper/A-Computational-Approach-to-Edge-Detection-Canny/fcf9fc4e23b45345c2404ce7d6cb0fc9dea2c9ec (дата звернення: 11.05.2026).

[13] Duda R.O., Hart P.E. Use of the Hough Transformation to Detect Lines and Curves in Pictures // Communications of the ACM. - 1972. - Vol. 15, No. 1. - P. 11-15. - URL: https://doi.org/10.1145/361237.361242 (дата звернення: 11.05.2026).

[14] Loshchilov I., Hutter F. SGDR: Stochastic Gradient Descent with Warm Restarts // International Conference on Learning Representations (ICLR 2017). - Toulon, France. - 2017. - URL: https://openreview.net/pdf?id=Skq89Scxx (дата звернення: 11.05.2026).

[15] Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection [Electronic resource] // Ultralytics Documentation. - 2020. - URL: https://docs.ultralytics.com/models/yolov4/ (дата звернення: 11.05.2026).

[16] Sermanet P., LeCun Y. Traffic Sign Recognition with Multi-Scale Convolutional Networks // The 2011 International Joint Conference on Neural Networks (IJCNN). - San Jose, CA, USA. - 2011. - P. 2809-2813. - URL: https://www.researchgate.net/publication/224260345_Traffic_sign_recognition_with_multi-scale_Convolutional_Networks (дата звернення: 11.05.2026).

[17] Qin Z., Wang H., Li X. Ultra Fast Structure-Aware Deep Lane Detection // European Conference on Computer Vision (ECCV). - 2020. - P. 276-291. - URL: https://doi.org/10.1007/978-3-030-58586-0_17 (дата звернення: 11.05.2026).

[18] Tabernik D., Skocaj D. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition // IEEE Transactions on Intelligent Transportation Systems. - 2020. - Vol. 21, No. 4. - P. 1427-1440. - URL: https://www.researchgate.net/publication/332952197_Deep_Learning_for_Large-Scale_Traffic-Sign_Detection_and_Recognition (дата звернення: 11.05.2026).

[19] Dewangan D.K., Sahu S.P. Towards the Design of Vision-Based Autonomous Vehicle System: Methodologies and Challenges // Evolutionary Intelligence. - 2022. - Vol. 16. - P. 759-800. - URL: https://link.springer.com/article/10.1007/s12065-022-00713-2 (дата звернення: 11.05.2026).

[20] Grigorescu S., Trasnea B., Cocias T., Macesanu G. A Survey of Deep Learning Techniques for Autonomous Driving // Journal of Field Robotics. - 2020. -

Vol. 37, No. 3. - P. 362-386. - URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.21918 (дата звернення: 11.05.2026).

[21] Wang R., Liu H., Chen X. et al. DSF-YOLO: Dynamic Sequence Fusion for Robust Multiscale Traffic Sign Detection under Adverse Weather Conditions // Scientific Reports. - 2025. - Vol. 15. - URL: https://doi.org/10.1038/s41598-025-02877-0 (дата звернення: 11.05.2026).

[22] He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). - Las Vegas, 2016. - P. 770-778. - URL: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf (дата звернення: 11.05.2026).

[23] Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). - Salt Lake City, 2018. - P. 4510-4520. - URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html (дата звернення: 11.05.2026).

[24] Ertler C., Mislej J., Ollmann T., Porzi L., Neuhold G., Kuang Y. The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale // European Conference on Computer Vision (ECCV). - Springer, Cham, 2020. - P. 68-84. - URL: https://arxiv.org/abs/1909.04422 (дата звернення: 11.05.2026).

[25] Ay G., Durdu A., Nesimioğlu B. S. Traffic Sign Board Recognition and Voice Alert System Using Convolutional Neural Network // ResearchGate Preprint. - 2022. - URL: https://www.researchgate.net/publication/352668818 (дата звернення: 11.05.2026).

[26] Liu L., Wang L., Ma Z. Improved Lightweight YOLOv5 Based on ShuffleNet and Its Application on Traffic Signs Detection // PLOS ONE. - 2024. - Vol. 19, No. 9. - URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11386454 (дата звернення: 11.05.2026).

[27] Dewangan D. K., Sahu S. P. Towards the Design of Vision-Based Autonomous Vehicle System: Methodologies and Challenges // Evolutionary Intelligence. - 2022. - Vol. 16. - P. 759-800. - URL: https://arxiv.org/abs/2108.03267 (дата звернення: 11.05.2026).

[28] Li P., Liu C., Li T., Wang X., Zhang S., Yu D. EMDFNet: Efficient Multi-Scale and Diverse Feature Network for Traffic Sign Detection // arXiv preprint. - 2024. - arXiv:2408.14189. - URL: https://arxiv.org/abs/2408.14189 (дата звернення: 11.05.2026).

[29] Grigorescu S., Trasnea B., Cocias T., Macesanu G. A Survey of Deep Learning Techniques for Autonomous Driving // Journal of Field Robotics. - 2020. - Vol. 37, No. 3. - P. 362-386. - URL: https://arxiv.org/abs/1910.07738 (дата звернення: 11.05.2026).

[30] Qu S., Yang X., Zhou H., Xie Y. Improved YOLOv5-Based for Small Traffic Sign Detection Under Complex Weather // Scientific Reports. - 2023. - Vol. 13. - Article 16396. - URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10533860 (дата звернення: 11.05.2026).

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ НАВЧАННЯ МОДЕЛІ U-NET

А.1. Програмний модуль навчання U-Net для сегментації ліній дорожньої розмітки (train_unet.py)

```
import json
import numpy as np
import cv2
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from pathlib import Path
from PIL import Image as PILImage
from tqdm import tqdm

BASE        = Path(r"C:\Users\ASUS\OneDrive\Робочий стіл\Диплом")
DATASET_DIR = BASE / "unified_dataset" / "lanes"
RUNS_DIR    = BASE / "runs" / "lanes"
RUNS_DIR.mkdir(parents=True, exist_ok=True)

CFG = {
 "epochs":      30,
 "batch_size":  4,
 "imgsz":       (256, 512),
 "lr":          1e-4,
 "weight_decay": 1e-4,
 "num_workers": 0,
 "device":      "cuda",
 "save_every":  5,
}

#  Dataset
class LaneDataset(Dataset):
def __init__(self, split="train", imgsz=(256, 512), augment=False):
self.img_dir = DATASET_DIR / "images" / split
self.lbl_dir = DATASET_DIR / "labels" / split
self.imgsz   = imgsz   # (H, W)
self.augment = augment
self.samples = sorted(self.img_dir.glob("*.jpg"))
print(f"  LaneDataset [{split}]: {len(self.samples)} зображень")

def __len__(self):
return len(self.samples)
```

```
def __getitem__(self, idx):
img_path = self.samples[idx]
lbl_path = self.lbl_dir / img_path.with_suffix(".json").name

# PIL замість cv2 - підтримує кириличні шляхи
try:
img = np.array(PILImage.open(img_path).convert("RGB"))
except Exception:
img = np.zeros((*self.imgsz, 3), dtype=np.uint8)

orig_H, orig_W = img.shape[:2]

# Маска ліній
mask = np.zeros((orig_H, orig_W), dtype=np.float32)
if lbl_path.exists():
try:
with open(lbl_path, encoding="utf-8") as f:
ann = json.load(f)
lanes     = ann.get("lanes", [])
h_samples = ann.get("h_samples", [])
for lane in lanes:
pts = [(int(x), int(y))
for x, y in zip(lane, h_samples) if x >= 0]
if len(pts) >= 2:
for i in range(len(pts) - 1):
cv2.line(mask, pts[i], pts[i+1], 1.0, thickness=5)
except Exception:
pass

# Ресайз
H, W = self.imgsz
img  = np.array(PILImage.fromarray(img).resize((W, H)))
mask = cv2.resize(mask, (W, H), interpolation=cv2.INTER_NEAREST)

# Аугментація
if self.augment:
img, mask = self._augment(img, mask)

# Нормалізація
img = img.astype(np.float32) / 255.0
img = (img - np.array([0.485, 0.456, 0.406])) / np.array([0.229, 0.224, 0.225])

img  = torch.from_numpy(img.transpose(2, 0, 1)).float()
mask = torch.from_numpy(mask).float().unsqueeze(0)
return img, mask

def _augment(self, img, mask):
if np.random.random() < 0.5:
```

```
img  = img[:, ::-1, :].copy()
mask = mask[:, ::-1].copy()
if np.random.random() < 0.5:
alpha = np.random.uniform(0.8, 1.2)
beta  = np.random.uniform(-20, 20)
img   = np.clip(img.astype(np.float32) * alpha + beta, 0, 255).astype(np.uint8)
return img, mask

#  U-Net
class DoubleConv(nn.Module):
def __init__(self, in_ch, out_ch):
super().__init__()
self.block = nn.Sequential(
nn.Conv2d(in_ch, out_ch, 3, padding=1, bias=False),
nn.BatchNorm2d(out_ch),
nn.ReLU(inplace=True),
nn.Conv2d(out_ch, out_ch, 3, padding=1, bias=False),
nn.BatchNorm2d(out_ch),
nn.ReLU(inplace=True),
)
def forward(self, x):
return self.block(x)

class UNet(nn.Module):
def __init__(self, in_channels=3, out_channels=1,
features=[32, 64, 128, 256]):
super().__init__()
self.downs      = nn.ModuleList()
self.ups        = nn.ModuleList()
self.pool       = nn.MaxPool2d(2, 2)

ch = in_channels
for f in features:
self.downs.append(DoubleConv(ch, f))
ch = f

self.bottleneck = DoubleConv(features[-1], features[-1] * 2)

for f in reversed(features):
self.ups.append(nn.ConvTranspose2d(f * 2, f, kernel_size=2, stride=2))
self.ups.append(DoubleConv(f * 2, f))

self.final = nn.Conv2d(features[0], out_channels, kernel_size=1)

def forward(self, x):
skips = []
for down in self.downs:
x = down(x)
```

```
skips.append(x)
x = self.pool(x)

x = self.bottleneck(x)
skips = skips[::-1]

for i in range(0, len(self.ups), 2):
x    = self.ups[i](x)
skip = skips[i // 2]
if x.shape != skip.shape:
x = nn.functional.interpolate(
x, size=skip.shape[2:], mode="bilinear", align_corners=False)
x = torch.cat([skip, x], dim=1)
x = self.ups[i + 1](x)

# Повертаємо logits (без sigmoid)
return self.final(x)

#  Функція втрат - BCEWithLogitsLoss + Dice
class CombinedLoss(nn.Module):
def __init__(self, bce_weight=0.5, dice_weight=0.5):
super().__init__()
self.bce_weight  = bce_weight
self.dice_weight = dice_weight
# BCEWithLogitsLoss - безпечно з autocast
self.bce = nn.BCEWithLogitsLoss()

def dice_loss(self, logits, target, smooth=1e-6):
pred   = torch.sigmoid(logits).view(-1)
target = target.view(-1)
inter  = (pred * target).sum()
return 1 - (2 * inter + smooth) / (pred.sum() + target.sum() + smooth)

def forward(self, logits, target):
return (self.bce_weight  * self.bce(logits, target) +
self.dice_weight * self.dice_loss(logits, target))

#  Метрики
def compute_iou(logits, target, threshold=0.5):
pred   = (torch.sigmoid(logits) > threshold).float()
target = (target > threshold).float()
inter  = (pred * target).sum()
union  = pred.sum() + target.sum() - inter
if union < 1e-6:
return 1.0
return (inter / union).item()

#  Тренування
```

```
def train():
print("|   Тренування U-Net - лінії розмітки  |")

device = torch.device(CFG["device"] if torch.cuda.is_available() else "cpu")
print(f"  Пристрій: {device}")
if device.type == "cuda":
print(f"  GPU: {torch.cuda.get_device_name(0)}")

train_ds = LaneDataset("train", CFG["imgsz"], augment=True)
val_ds   = LaneDataset("val",   CFG["imgsz"], augment=False)

if len(train_ds) == 0:
print("\n[!] Тренувальний датасет порожній!")
return

train_loader = DataLoader(train_ds, batch_size=CFG["batch_size"],
shuffle=True,  num_workers=CFG["num_workers"],
pin_memory=True)
val_loader   = DataLoader(val_ds,   batch_size=CFG["batch_size"],
shuffle=False, num_workers=CFG["num_workers"],
pin_memory=True)

model = UNet(in_channels=3, out_channels=1, features=[32, 64, 128, 256])
model = model.to(device)
print(f"\n  Параметрів: {sum(p.numel() for p in model.parameters()):,}")

optimizer = torch.optim.AdamW(model.parameters(),
lr=CFG["lr"],
weight_decay=CFG["weight_decay"])
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(
optimizer, T_max=CFG["epochs"], eta_min=1e-6)
criterion = CombinedLoss()

# Виправлений GradScaler
use_amp = device.type == "cuda"
scaler  = torch.amp.GradScaler("cuda") if use_amp else None

best_iou = 0.0
history  = {"train_loss": [], "val_loss": [], "val_iou": []}

print(f"\n  Починаємо тренування ({CFG['epochs']} епох)...\n")

for epoch in range(1, CFG["epochs"] + 1):

# - Train -
model.train()
train_loss = 0.0
for imgs, masks in tqdm(train_loader,
```

```
desc=f"Epoch {epoch:3d}/{CFG['epochs']} [train]",
leave=False):
imgs  = imgs.to(device)
masks = masks.to(device)
optimizer.zero_grad()

if use_amp:
with torch.amp.autocast("cuda"):
logits = model(imgs)
loss   = criterion(logits, masks)
scaler.scale(loss).backward()
scaler.step(optimizer)
scaler.update()
else:
logits = model(imgs)
loss   = criterion(logits, masks)
loss.backward()
optimizer.step()

train_loss += loss.item()

train_loss /= len(train_loader)
scheduler.step()

# - Validation -
model.eval()
val_loss = 0.0
val_iou  = 0.0
with torch.no_grad():
for imgs, masks in tqdm(val_loader,
desc=f"Epoch {epoch:3d}/{CFG['epochs']} [val]  ",
leave=False):
imgs  = imgs.to(device)
masks = masks.to(device)
if use_amp:
with torch.amp.autocast("cuda"):
logits = model(imgs)
loss   = criterion(logits, masks)
else:
logits = model(imgs)
loss   = criterion(logits, masks)
val_loss += loss.item()
val_iou  += compute_iou(logits.cpu(), masks.cpu())

n_val     = max(len(val_loader), 1)
val_loss /= n_val
val_iou  /= n_val
```

```
history["train_loss"].append(train_loss)
history["val_loss"].append(val_loss)
history["val_iou"].append(val_iou)

lr_now = optimizer.param_groups[0]["lr"]
print(f"  Epoch {epoch:3d}/{CFG['epochs']} | "
f"loss: {train_loss:.4f} | "
f"val_loss: {val_loss:.4f} | "
f"IoU: {val_iou:.4f} | "
f"lr: {lr_now:.2e}")

if val_iou > best_iou:
best_iou = val_iou
torch.save({
 "epoch":       epoch,
 "model_state": model.state_dict(),
 "optimizer":   optimizer.state_dict(),
 "val_iou":     val_iou,
 "config":      CFG,
}, RUNS_DIR / "best.pt")
print(f"  * IoU={best_iou:.4f} -> best.pt")

if epoch % CFG["save_every"] == 0:
torch.save(model.state_dict(),
RUNS_DIR / f"checkpoint_epoch{epoch}.pt")

with open(RUNS_DIR / "history.json", "w") as f:
json.dump(history, f, indent=2)

print(f"|   Тренування завершено!
print(f"|   Найкращий IoU: {best_iou:.4f}
print(f"\n  Модель:  {RUNS_DIR / 'best.pt'}"
print(f"  Історія: {RUNS_DIR / 'history.json'}")
return history

if __name__ == "__main__":
train()
```

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ НАВЧАННЯ МОДЕЛІ YOLOV8

Б.1. Програмний модуль навчання YOLOv8 для детекції дорожніх знаків (train_yolov8s.py)

```
import torch
import yaml
from pathlib import Path
from ultralytics import YOLO

BASE        = Path(r"C:\Users\ASUS\OneDrive\Робочий стіл\Диплом")
DATASET_DIR = BASE / "unified_dataset" / "signs"
RUNS_DIR    = BASE / "runs" / "signs_s"

#  Параметри (однакові з YOLOv8n для чесного порівняння)
CONFIG = {
 "epochs":       50,
 "batch":        8,
 "imgsz":        416,
 "lr0":          0.01,
 "lrf":          0.001,
 "momentum":     0.937,
 "weight_decay": 0.0005,
 "warmup_epochs": 3,
 "patience":     20,
 "device":       0,
 "workers":      2,
 "cache":        False,
 "amp":          True,
 "project":      str(RUNS_DIR),
 "name":         "train",
 "exist_ok":     True,
}

def train():
print("|   (порівняння з YOLOv8n)              |")

print(f"  GPU: {torch.cuda.get_device_name(0)}")

# Перевіряємо що dataset.yaml вже існує (створений train_yolov8.py)
yaml_path = DATASET_DIR / "dataset.yaml"
if not yaml_path.exists():
print(f"  [!] Не знайдено {yaml_path}")
print("  Спочатку запусти train_yolov8.py")
```

```
return

# YOLOv8s - small версія
print("\n  Завантаження YOLOv8s (pretrained на COCO)...")
model = YOLO("yolov8s.pt")

print(f"  Параметри тренування:")
for k, v in CONFIG.items():
print(f"    {k}: {v}")

print("\n  Починаємо тренування...\n")
results = model.train(data=str(yaml_path), **CONFIG)

print("|   YOLOv8s тренування завершено!       |")
print(f"\n  Модель: {RUNS_DIR}/train/weights/best.pt")
return results

def validate():
model_path = RUNS_DIR / "train" / "weights" / "best.pt"
if not model_path.exists():
print(f"  [!] Модель не знайдена: {model_path}")
return

print(f"\n- Валідація YOLOv8s -")
model     = YOLO(str(model_path))
yaml_path = DATASET_DIR / "dataset.yaml"
metrics   = model.val(data=str(yaml_path), device=0)

print(f"\n  mAP@0.5:      {metrics.box.map50:.4f}")
print(f"  mAP@0.5:0.95: {metrics.box.map:.4f}")
print(f"  Precision:    {metrics.box.mp:.4f}")
print(f"  Recall:       {metrics.box.mr:.4f}")
return metrics

if __name__ == "__main__":
train()
validate()
```