ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

кваліфікаційна робота

Програмні засоби обробки зображень з використанням штучних нейронних мереж

Виконав:
студент гр. КІУКІ-21-2
Каверін М.О.

Керівник:
ас. каф. ЕОМ,
Романюк О.С.

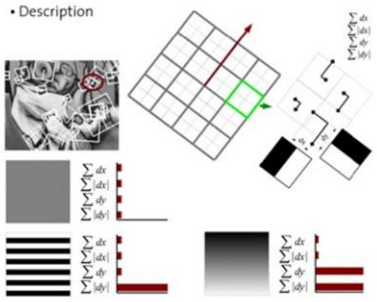
Мета роботи та завдання

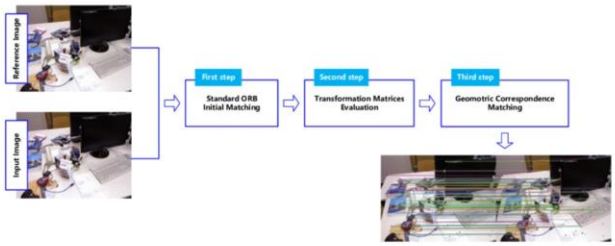
Метою роботи є розробка, реалізація та експериментальне дослідження програмних засобів обробки зображень на основі штучних нейронних мереж у середовищі Google Colab для підвищення точності автоматичного розпізнавання візуальних об’єктів.

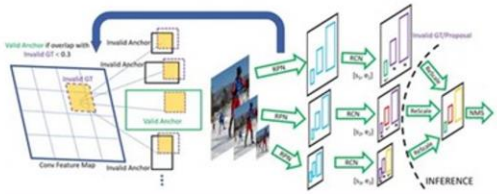
Задачі:

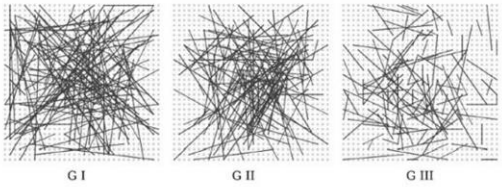
- провести аналіз сучасних підходів до обробки зображень із використанням штучних нейронних мереж;
- обґрунтувати вибір архітектури CNN для вирішення задачі класифікації зображень;
- вибрати та підготувати три репрезентативні набори зображень: MNIST, Fashion-MNIST та CIFAR-10;
- реалізувати програмні засоби попередньої обробки та візуалізації зображень у Google Colab;
- побудувати та навчити три моделі CNN відповідно до кожного з обраних датасетів;
- виконати візуалізацію результатів, включаючи графіки навчання та матриці плутанини;
- провести порівняльний аналіз ефективності моделей за точністю, швидкістю та візуальними характеристиками.

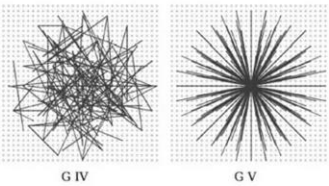
Методи обробки зображень 3

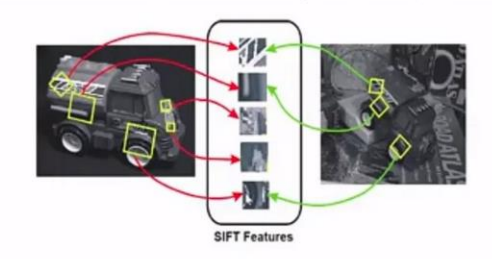












Порівняльний аналіз існуючих методів 4

Метод	Інваріантність до масштабу	Інваріантність до обертання	Тип дескриптора	Швидкодія
SIFT	Так	Так	Вектор на основі градієнтів	Низька
SURF	Так	Так	Гаусс-зважені вейвлети Хаара	Середня
BRIEF	Ні	Ні	Бінарні порівняння інтенсивності	Висока
ORB	Частково	Так	Бінарні з орієнтацією	Висока

Метод	Використання в реальному часі	Патентні обмеження	Надійність у складних умовах	Обсяг дескриптора
SIFT	Обмежене	Так	Висока	128 float
SURF	Можливе	Так	Висока	64 float
BRIEF	Придатне	Ні	Низька	256 біт
ORB	Оптимальне	Ні	Середня	256 біт

Методи попередньої обробки зображень

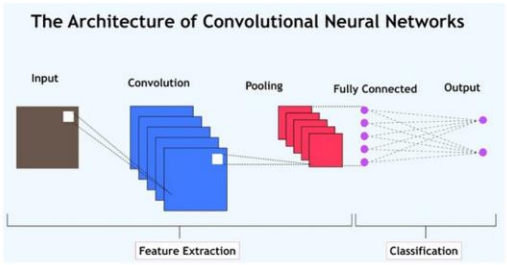
5

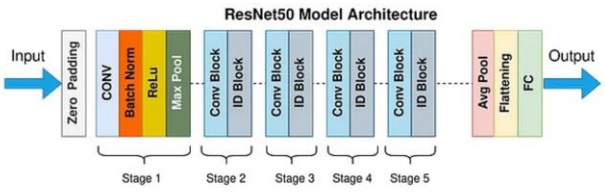
Метод	Призначення	Типові алгоритми
Фільтрація	Зменшення шумів, підсилення контурів, локальне згладжування зображення	Гауссове згладжування, медіанний фільтр, Sobel, Laplacian
Нормалізація	Уніфікація діапазону інтенсивностей пікселів для стабільного навчання нейромережі	Мін-макс нормалізація, z-нормалізація, ділення на 255
Масштабування	Приведення зображення до фіксованого розміру, сумісного з архітектурою моделі	Інтерполяція (бікв. або бікуб.), ресайз із паддінгом або crop
Метод	**Переваги**	**Обмеження**
Фільтрація	Зменшує вплив артефактів і шуму, покращує якість ознак	Може спотворити дрібні деталі при надмірному згладжуванні
Нормалізація	Забезпечує стабільність обчислень, прискорює збіжність моделі	Не враховує просторовий контекст, може втрачати локальні особливості
Масштабування	Уніфікує розміри входу, знижує обчислювальну складність	Може змінити пропорції об'єкта, втратити інформацію при обрізанні

- Фільтрація забезпечує видалення шуму та підсилення локальних контурів, що є критично важливим для стабільного витягування ознак на ранніх рівнях згорткової обробки.
- Нормалізація сприяє уніфікації діапазону інтенсивностей пікселів, що забезпечує чисельну стабільність під час градієнтного навчання мережі.
- Масштабування дозволяє адаптувати розмір зображень до фіксованих параметрів вхідного шару нейронної архітектури, зменшуючи обчислювальні витрати та дозволяючи використовувати однакову структуру моделі на різних наборах даних.

6

Архітектури ШНМ





LeNet Architecture

AlexNet Architecture

7

Порівняльний аналіз архітектур ШНМ

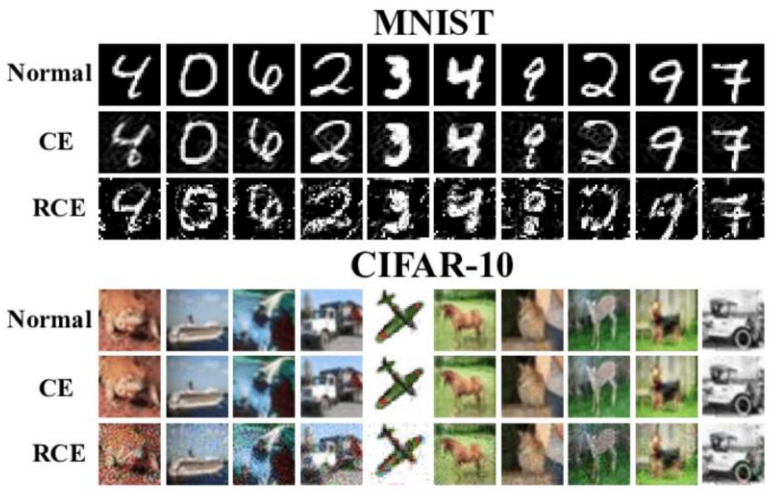
Архітектура	Призначення	Переваги	Недоліки
MLP	Загальні задачі класифікації	Простота реалізації	Не враховує структуру зображення, багато параметрів
CNN	Аналіз зображень	Локальні фільтри, просторові зв'язки, економія параметрів	Може потребувати більше ресурсів
RNN / LSTM	Обробка послідовностей (текст, мова)	Модель пам'яті, працює з часовими рядами	Не підходить для просторових зображень
Autoencoder	Зниження розмірності, денойзинг	Компресія зображень, навчання без міток	Не оптимальна для класифікації
Transformers (ViT)	Обробка зображень (візуальні трансформери)	Потужні для великих наборів	Вимагають багато даних і ресурсів

8

Вибір програмних засобів

9

Набори даних



10

Фрагменти коду програми. Процес навчання

```
[1] import tensorflow as tf
    from tensorflow.keras import datasets, layers, models
    import matplotlib.pyplot as plt
    import numpy as np
    import seaborn as sns
    from sklearn.metrics import confusion_matrix, Confusion
```

2. Датасет MNIST

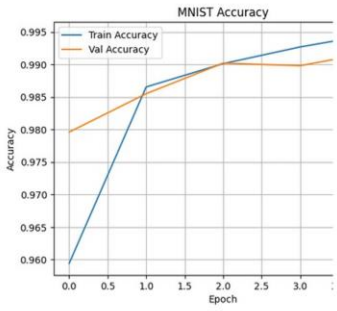
```
[2] # Завантаження
    (train_images, train_labels), (test_images, test_labels
    train_images = train_images[..., np.newaxis] / 255.0
    test_images = test_images[..., np.newaxis] / 255.0
```

```
Downloading data from https://storage.googleapis.com/te
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

Візуалізація прикладів

```
[3] plt.figure(figsize=(5,5))
    for i in range(9):
        plt.subplot(3,3,i+1)
        plt.imshow(train_images[i].squeeze(), cmap='gray')
        plt.title(f'Label: {train_labels[i]}')
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('MNIST Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```




```
history = model.fit(train_images, train_labels, epochs=5,
                    validation_data=(test_images, test_labels))
```

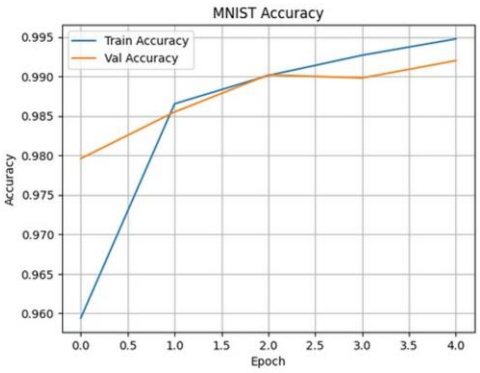
```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 59s 30ms/step - accuracy: 0.9124 - loss: 0.3033 - val_accuracy: 0.9796 - val_loss: 0.0659
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 30ms/step - accuracy: 0.9864 - loss: 0.0462 - val_accuracy: 0.9855 - val_loss: 0.0392
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 30ms/step - accuracy: 0.9908 - loss: 0.0300 - val_accuracy: 0.9902 - val_loss: 0.0289
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 85s 31ms/step - accuracy: 0.9928 - loss: 0.0232 - val_accuracy: 0.9898 - val_loss: 0.0314
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 79s 30ms/step - accuracy: 0.9951 - loss: 0.0156 - val_accuracy: 0.9920 - val_loss: 0.0264
```

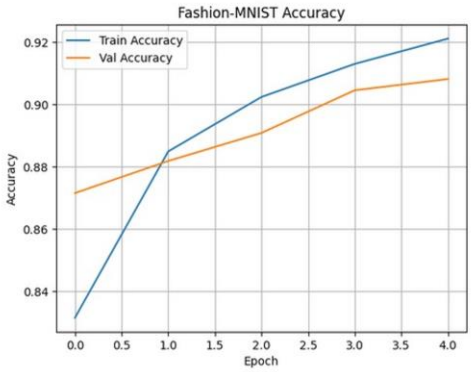
Результати роботи 11

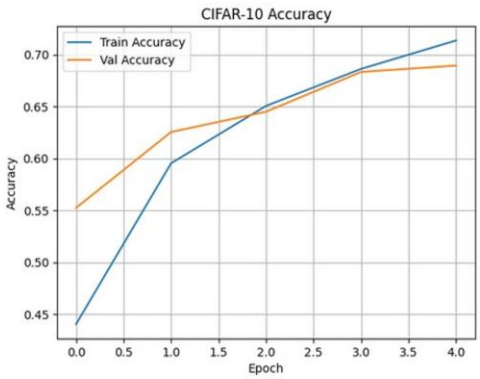
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 64)	102,464
dense_1 (Dense)	(None, 10)	650

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_5 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_6 (Conv2D)	(None, 4, 4, 64)	36,928
flatten_2 (Flatten)	(None, 1024)	0
dense_4 (Dense)	(None, 64)	65,600
dense_5 (Dense)	(None, 10)	650

Результати роботи 12







13 Висновки

У результаті проведеного дослідження було реалізовано програмний засіб для обробки та класифікації зображень з використанням штучних нейронних мереж у середовищі Google Colab. У роботі було поєднано теоретичний аналіз основ глибокого навчання з практичною реалізацією згорткових нейронних мереж, що дозволило системно дослідити ефективність застосування CNN-архітектур для розпізнавання образів на прикладі трьох репрезентативних датасетів: MNIST, Fashion-MNIST та CIFAR-10.

Проведено обґрунтований вибір кожного з наборів даних: MNIST слугував для перевірки базової архітектури, Fashion-MNIST – для оцінки здатності моделі до класифікації об'єктів зі складнішою внутрішньокласовою варіативністю, а CIFAR-10 – як більш складний реалістичний набір зображень, що містить колірні канали та широку предметну категоризацію. Для кожного з наборів було здійснено відповідну попередню обробку, зокрема нормалізацію та приведення розмірності, що забезпечило коректне функціонування моделей.

Архітектура згорткової нейронної мережі була побудована з урахуванням ключових принципів CNN: згорткові шари для автоматичного витягування ознак, шари підвибірки для зменшення розмірності та повнозв'язні шари для класифікації. Навчання проводилось із застосуванням оптимізатора Adam і функції втрат категоріальної крос-ентропії, що дозволило досягти швидкої збіжності моделей. Усі реалізації були протестовані на окремих тестових вибірках, а результати оцінено за допомогою метрик точності, графіків тренування та матриць плутанини.

За результатами експериментів було встановлено, що модель досягає високої точності на простих наборах (MNIST – понад 99%, Fashion-MNIST – понад 91%), тоді як на складнішому CIFAR-10 – на рівні 70% після п'яти епох навчання. Отримані результати підтверджують як здатність згорткових мереж до генералізації, так і їх обмеження в умовах високої варіативності вхідних зображень. Водночас спостереження за динамікою навчання свідчать про відсутність перенавчання, що вказує на правильно обрану глибину мережі та параметри навчального процесу.

ДОДАТОК Б

Програмний код

Б.1 Лістинг коду

Б.1.1 Імпорт бібліотек, початкові налаштування та завантаження MNIST

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
# Завантаження
(train_images, train_labels), (test_images, test_labels) =
datasets.mnist.load_data()
train_images = train_images[..., np.newaxis] / 255.0
test_images = test_images[..., np.newaxis] / 255.0
```

Б.1.2 Візуалізація прикладів

```
plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(train_images[i].squeeze(), cmap='gray')
    plt.title(f'Label: {train_labels[i]}')
    plt.axis('off')
plt.tight_layout()
plt.show()
```

Б.1.3 Побудова CNN для MNIST

```
 model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

```
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=T
rue),
              metrics=['accuracy'])
model.summary()
```

Б.1.4 Навчання моделі та візуалізація процесу навчання

```
history = model.fit(train_images, train_labels, epochs=5,
validation_data=(test_images, test_labels))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('MNIST Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```

Б.1.5 Робота з датасет Fashion-MNIST

```
# Завантаження Fashion-MNIST
(f_train_images, f_train_labels), (f_test_images, f_test_labels)
= datasets.fashion_mnist.load_data()
f_train_images = f_train_images[..., np.newaxis] / 255.0
f_test_images = f_test_images[..., np.newaxis] / 255.0

plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(f_train_images[i].squeeze(), cmap='gray')
    plt.title(f'Label: {f_train_labels[i]}')
    plt.axis('off')
plt.tight_layout()
plt.show()
f_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
```

```
    layers.Dense(10)
])
f_model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=T
rue),
              metrics=['accuracy'])
f_model.summary()
f_history = f_model.fit(f_train_images, f_train_labels,
epochs=5,
                        validation_data=(f_test_images,
f_test_labels))
plt.plot(f_history.history['accuracy'], label='Train Accuracy')
plt.plot(f_history.history['val_accuracy'], label='Val
Accuracy')
plt.title('Fashion-MNIST Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```

Б.1.6 Робота з датасет CIFAR-10

```
# Завантаження CIFAR-10
(c_train_images, c_train_labels), (c_test_images, c_test_labels)
= datasets.cifar10.load_data()
c_train_images = c_train_images / 255.0
c_test_images = c_test_images / 255.0
c_train_labels = c_train_labels.squeeze()
c_test_labels = c_test_labels.squeeze()
plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(c_train_images[i])
    plt.title(f'Label: {c_train_labels[i]}')
    plt.axis('off')
plt.tight_layout()
plt.show()
c_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])
c_model.compile(optimizer='adam',
```

```
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=T
rue),
              metrics=['accuracy'])
c_model.summary()
c_history = c_model.fit(c_train_images, c_train_labels,
epochs=5,
                        validation_data=(c_test_images,
c_test_labels))
plt.plot(c_history.history['accuracy'], label='Train Accuracy')
plt.plot(c_history.history['val_accuracy'], label='Val
Accuracy')
plt.title('CIFAR-10 Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```