ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Комп'ютерна система обробки зображень з використанням Deep Learning

Кваліфікаційна робота

Виконав:
студент гр. КІУКІ-21-2
Карножицький А.О.

Керівник:
доц. каф. ЕОМ, к.т.н.,
Бугрій А.М.

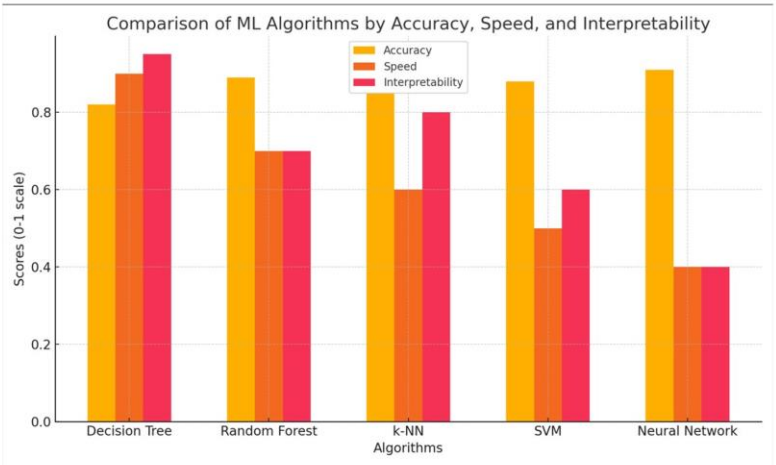
Мета роботи та завдання 2

Метою кваліфікаційної роботи є розробка комп'ютерної системи обробки зображень з використанням Deep Learning.

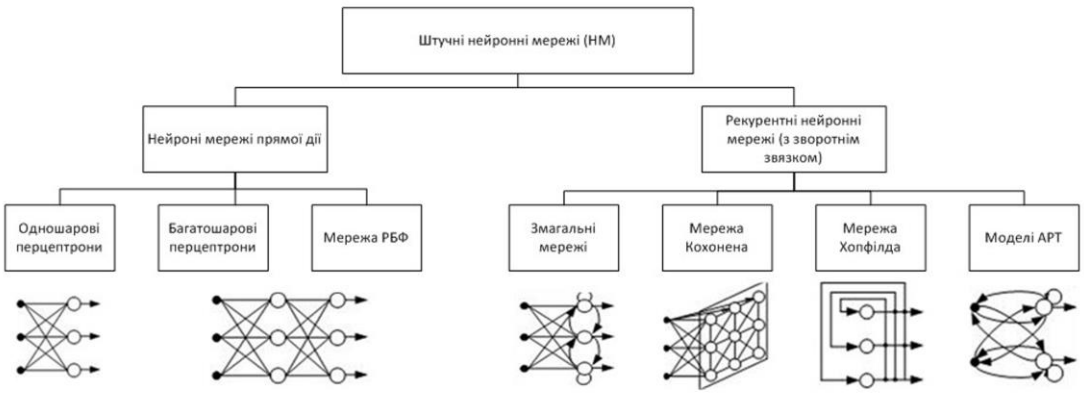
Завдання:

- ❑ здійснення аналізу теоретичних основ обробки зображень та машинного навчання;
- ❑ дослідження сучасних методів та архітектур нейронних мереж;
- ❑ вибір засобів реалізації ПЗ;
- ❑ практична реалізація алгоритму обробки зображень;
- ❑ аналіз результатів

Порівняльний аналіз методів машинного навчання 3



Класифікація ШНМ 4

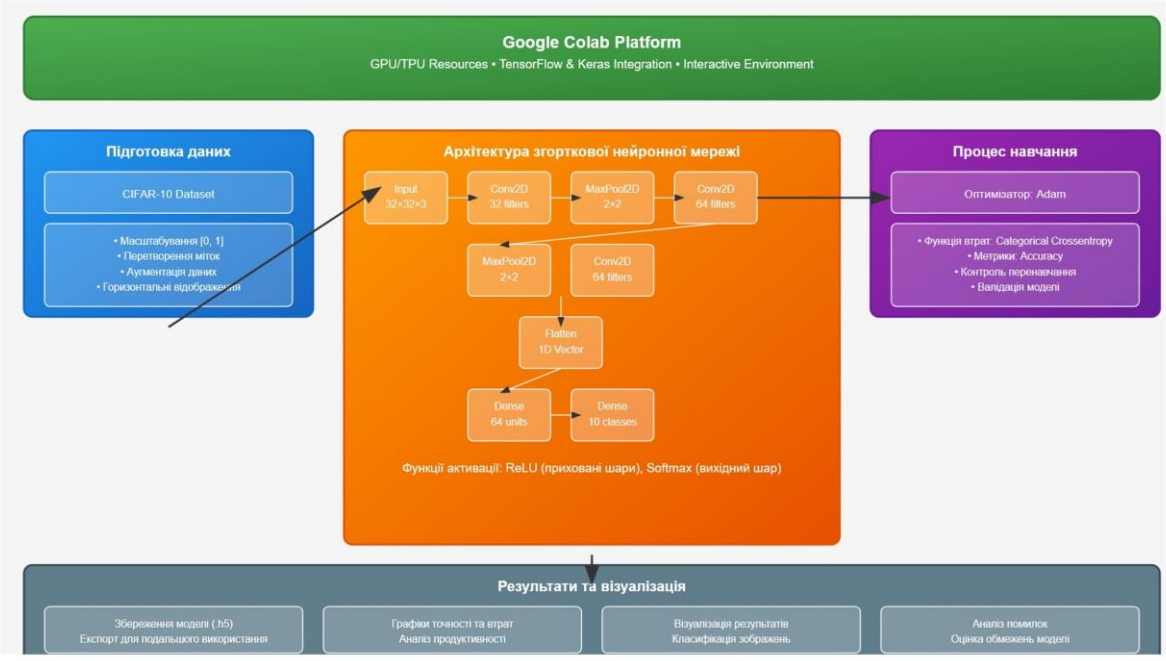


Огляд архітектур та інструментів реалізації 5

Архітектура	Призначення	Ключові особливості
CNN	Обробка зображень, відео, просторових даних	Згорткові фільтри, локальність, ієрархічне вилучення ознак
RNN	Обробка послідовностей, часових рядів, мови	Зворотні зв'язки, пам'ять про попередні стани
GAN	Генерація нових даних, зокрема зображень	Архітектура з двома мережами: генератор і дискримінатор
Архітектура	**Переваги**	**Недоліки**
CNN	Висока точність при класифікації зображень	Не пристосована до обробки послідовностей
RNN	Добре працює з часовими залежностями	Складність тренування, проблема з градієнтами
GAN	Здатність створювати реалістичні нові дані	Важке налаштування, нестабільність під час навчання

Фреймворк	Основне призначення	Сумісність з Google Colab
TensorFlow	Глибоке навчання, розробка нейронних мереж	Повна інтеграція, попередньо встановлений
PyTorch	Глибоке навчання, наукові дослідження та візуалізація	Повна інтеграція, попередньо встановлений
OpenCV	Обробка зображень, комп'ютерний зір	Повна інтеграція, попередньо встановлений
Фреймворк	**Переваги**	**Недоліки**
TensorFlow	Підтримка мобільних та веб-платформ, TPU	Складніша відлагодження, громіздкий синтаксис
PyTorch	Гнучкість, зручність для дослідників, динамічне обчислення	Менш зручний у виробничих середовищах
OpenCV	Швидкість, простота базових операцій зображень	Не призначений для глибокого навчання

Архітектура системи обробки зображень 6

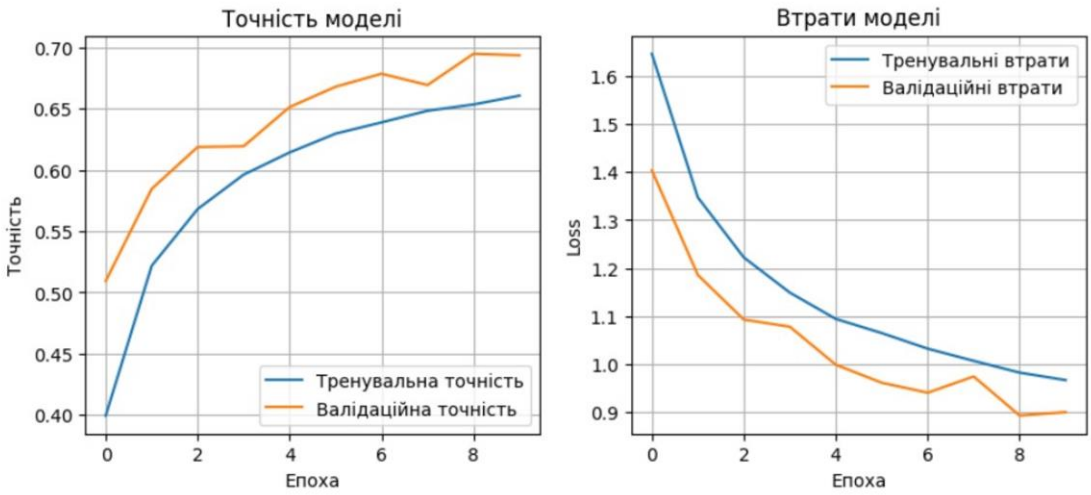


Результати оцінювання якості роботи 7

```
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_classes))
print(confusion_matrix(y_test, y_pred_classes))
```

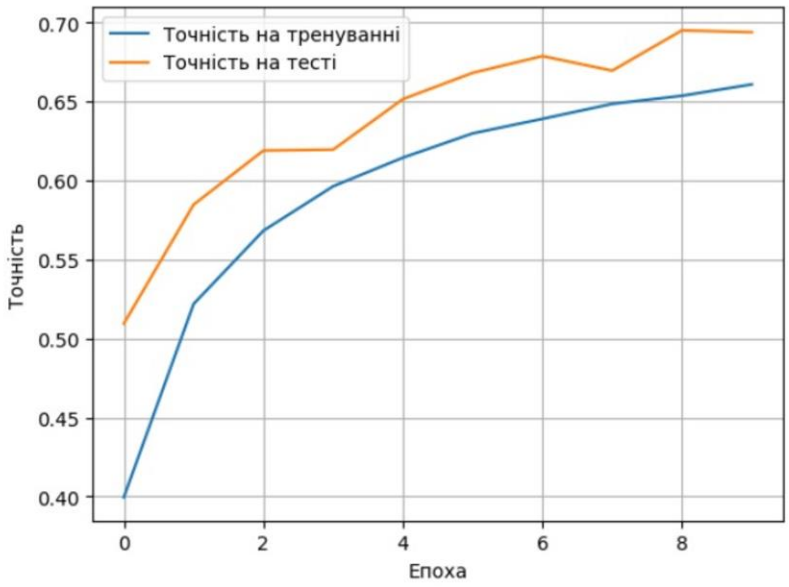
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step
              precision    recall  f1-score   support

           0       0.75      0.73      0.74      1000
           1       0.74      0.88      0.80      1000
           2       0.63      0.58      0.60      1000
           3       0.56      0.46      0.50      1000
           4       0.69      0.59      0.64      1000
           5       0.70      0.45      0.55      1000
           6       0.65      0.86      0.74      1000
           7       0.71      0.78      0.74      1000
           8       0.76      0.82      0.79      1000
           9       0.72      0.79      0.75      1000

    accuracy                           0.69     10000
   macro avg       0.69      0.69      0.69     10000
weighted avg       0.69      0.69      0.69     10000

[[733  35  44  10   7   2  14  10  97  48]
 [ 11 882   1   2   1   2  11   2  19  69]
 [ 63  15 578  43  69  38  93  47  26  28]
 [ 23  29  81 461  57  90 121  62  32  44]
 [ 24  13  77  38 589  19 121  90  21   8]
 [ 23  19  67 199  55 449  64  85  13  26]
 [ 10  13  36  33  18   5 857  10   7  11]
 [ 17  11  25  25  49  32  16 779  11  35]
 [ 57  53  10   7   5   1   2   5 819  41]
 [ 19 122   4   8   3   5  15   4  29 791]]
```

Аналіз результатів 8



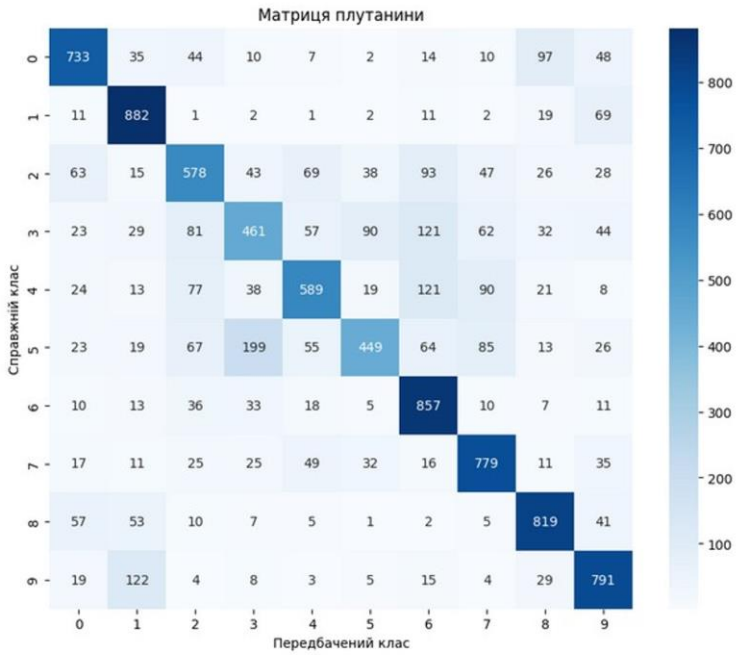
9

Аналіз результатів навчання



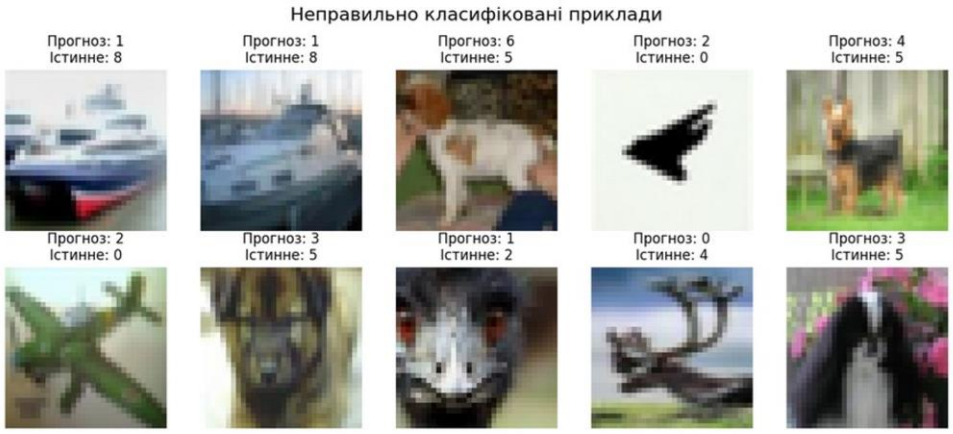
10

Матриця плутанини



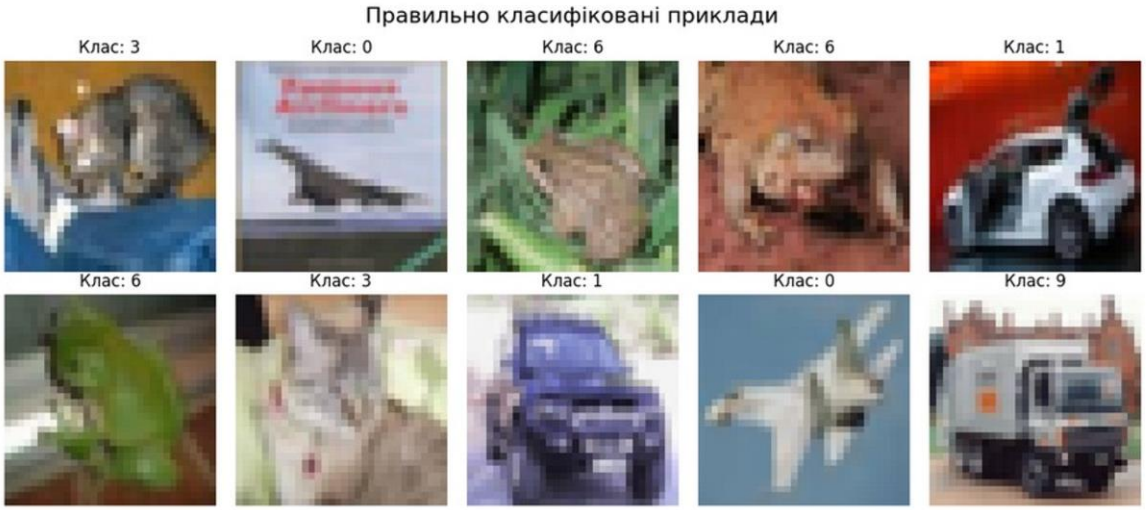
11

Результати роботи



12

Результати роботи



Висновки

13

У процесі виконання кваліфікаційної роботи було розроблено та реалізовано комп’ютерну систему обробки зображень із використанням технологій глибинного навчання, зокрема згорткових нейронних мереж. Робота охоплює повний цикл побудови моделі – від аналізу предметної області й постановки задачі до реалізації, навчання, тестування та візуалізації результатів у середовищі Google Colab.

Проведене дослідження підтвердило доцільність використання CNN у задачах класифікації зображень. Реалізована модель досягла прийнятного рівня точності на тестовій вибірці та продемонструвала здатність до узагальнення при обмеженій кількості навчальних прикладів. Було встановлено, що правильна підготовка даних – нормалізація, аугментація – істотно впливає на якість результатів. Візуалізація роботи моделі дала змогу виявити помилки класифікації, які, у свою чергу, дозволили зробити обґрунтовані висновки про можливі напрями подальшого вдосконалення.

Практична цінність розробленої системи полягає в її модульності, відкритості коду, простоті інтеграції в інші проєкти та адаптивності до нових наборів зображень. Окрім того, реалізація в середовищі Google Colab робить її доступною для широкого кола користувачів без потреби у спеціалізованому обладнанні.

ДОДАТОК Б

Програмний код

Б.1 Лістинг коду

```
# Завантаження бібліотек
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report,
confusion_matrix
import numpy as np
# Завантаження CIFAR-10
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
# Аугментація
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
datagen.fit(x_train)
# Побудова моделі
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.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# Навчання моделі
history = model.fit(datagen.flow(x_train, y_train,
batch_size=64),
                    epochs=10,
                    validation_data=(x_test, y_test))
# Оцінка точності
y_pred = model.predict(x_test)
```

```
y_pred_classes = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_classes))
print(confusion_matrix(y_test, y_pred_classes))
# Візуалізація результатів навчання
plt.plot(history.history['accuracy'], label='Точність на
тренуванні')
plt.plot(history.history['val_accuracy'], label='Точність на
тесті')
plt.xlabel('Епоха')
plt.ylabel('Точність')
plt.legend()
plt.grid(True)
plt.show()
# Графік точності
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Тренувальна
точність')
plt.plot(history.history['val_accuracy'], label='Валідаційна
точність')
plt.title('Точність моделі')
plt.xlabel('Епоха')
plt.ylabel('Точність')
plt.legend()
plt.grid(True)

# Графік втрат
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Тренувальні втрати')
plt.plot(history.history['val_loss'], label='Валідаційні
втрати')
plt.title('Втрати моделі')
plt.xlabel('Епоха')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
import seaborn as sns

conf_mat = confusion_matrix(y_test, y_pred_classes)
plt.figure(figsize=(10,8))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
plt.title('Матриця плутанини')
plt.xlabel('Передбачений клас')
plt.ylabel('Справжній клас')
plt.show()
# Індекси помилок
errors = np.where(y_pred_classes != y_test.reshape(-1))[0]

# Показати 10 помилок
plt.figure(figsize=(12,5))
for i, idx in enumerate(errors[:10]):
    plt.subplot(2, 5, i+1)
```

```
    image = x_test[idx]
    if image.shape[-1] == 1:
        plt.imshow(image.reshape(image.shape[0],
image.shape[1]), cmap='gray')
    else:
        plt.imshow(image)

    true_label = y_test[idx][0] if y_test.ndim == 2 else
y_test[idx]
    plt.title(f"Прогноз: {y_pred_classes[idx]}\nІстинне:
{true_label}")
    plt.axis('off')

plt.tight_layout()
plt.suptitle("Неправильно класифіковані приклади", y=1.05,
fontsize=16)
plt.show()

# Індекси правильно класифікованих прикладів
corrects = np.where(y_pred_classes == y_test.reshape(-1))[0]

# Показати 10 правильно класифікованих прикладів
plt.figure(figsize=(12,5))
for i, idx in enumerate(corrects[:10]):
    plt.subplot(2, 5, i+1)
    image = x_test[idx]

    if image.shape[-1] == 1:
        plt.imshow(image.reshape(image.shape[0],
image.shape[1]), cmap='gray')
    else:
        plt.imshow(image)

    true_label = y_test[idx][0] if y_test.ndim == 2 else
y_test[idx]
    plt.title(f"Клас: {true_label}")
    plt.axis('off')

plt.tight_layout()
plt.suptitle("Правильно класифіковані приклади", y=1.05,
fontsize=16)
plt.show()
```