ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН

КВАЛІФІКАЦІЙНА РОБОТА

«Комп'ютерна система класифікації та сегментації зображень з використанням машинного навчання»

Студент гр. КІУКІ-21-1 **Абрагамович В.О.**

Керівник **ст.викл. Радченко В.О.**

Харків 2025

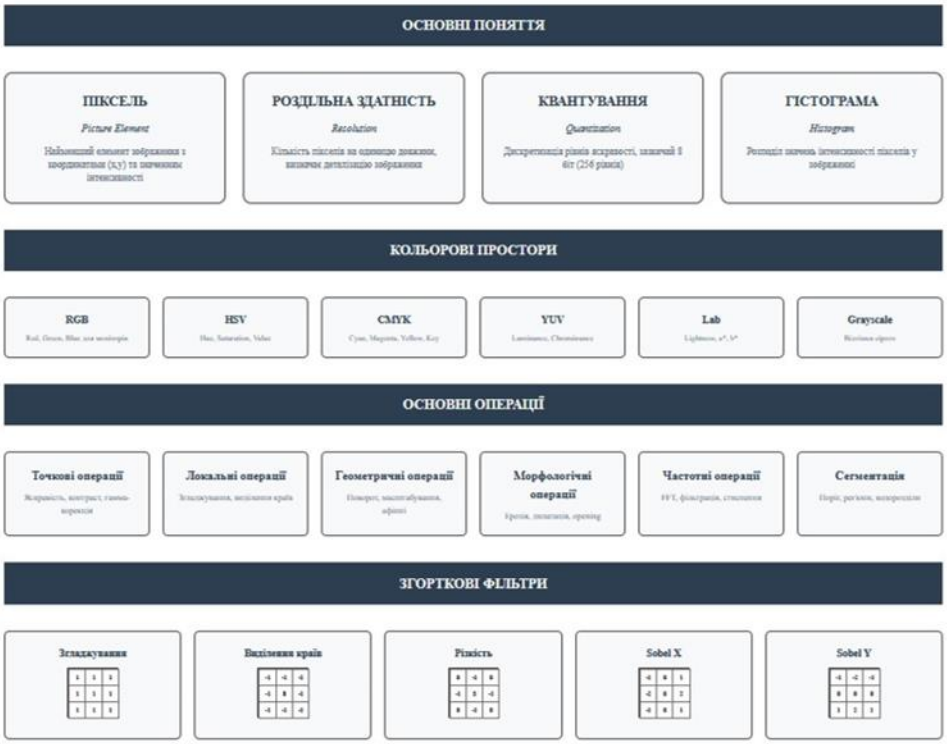
Мета та завдання кваліфікаційної роботи

Мета кваліфікаційної роботи: розробка та реалізація комп'ютерної системи, здатної виконувати класифікацію та сегментацію зображень на основі методів машинного навчання, зокрема глибоких згорткових нейронних мереж, з використанням відкритих платформ, таких як Google Colab.

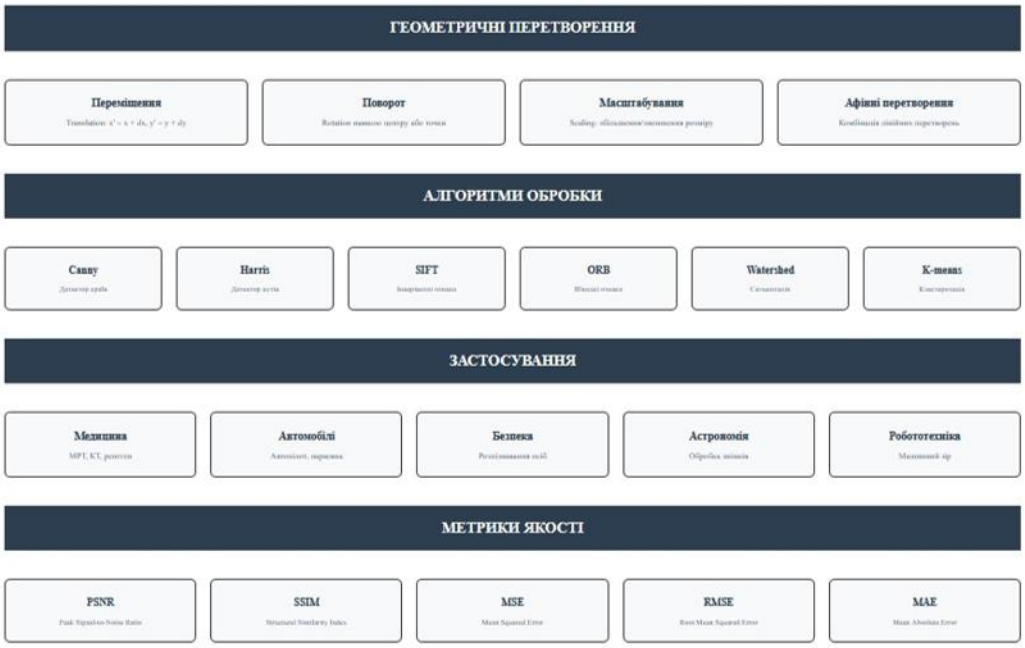
Завдання:

- провести огляд сучасних підходів до класифікації та сегментації зображень із використанням машинного навчання;
- обґрунтувати вибір середовища розробки та інструментів, зокрема Google Colab, TensorFlow та Keras;
- реалізувати модель класифікації зображень на базі згорткової нейронної мереж;
- реалізувати модель сегментації зображень із використанням архітектури U-Ne;
- провести навчання моделей на відкритих датасетах та проаналізувати отримані результати;
- оцінити точність роботи системи, здійснити її візуальне тестування та вивести ефективні метрики.

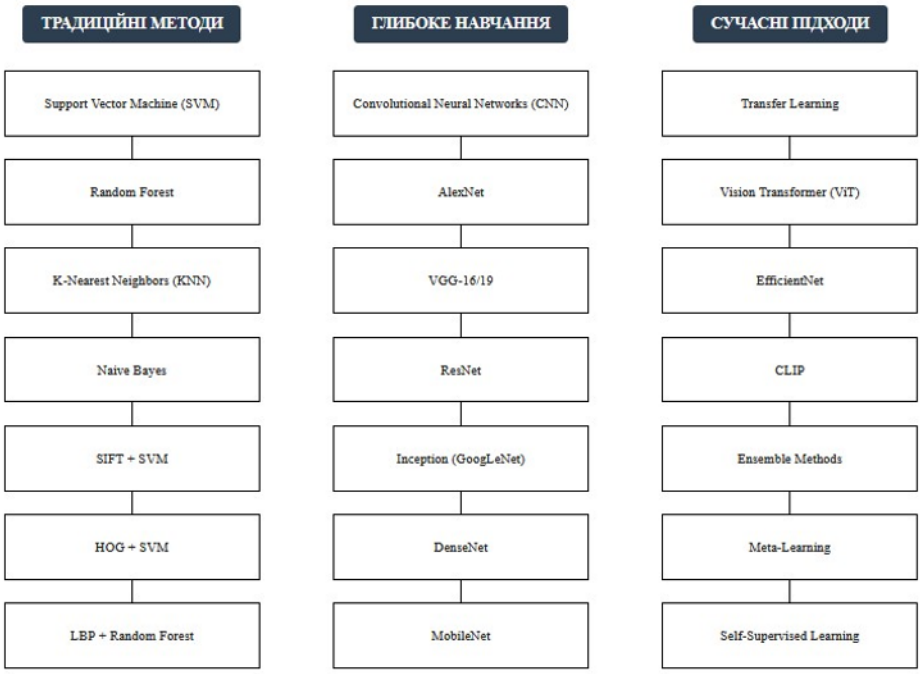
Основи обробки зображень



Основи обробки зображень

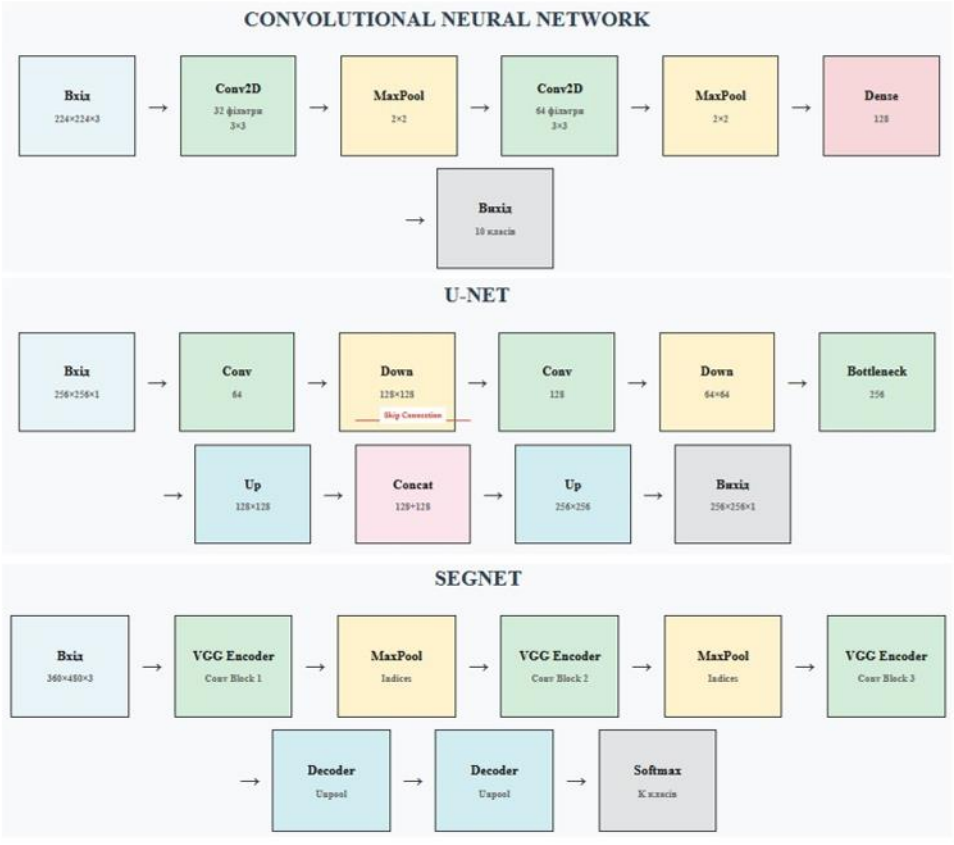


Методи машинного навчання для класифікації зображень



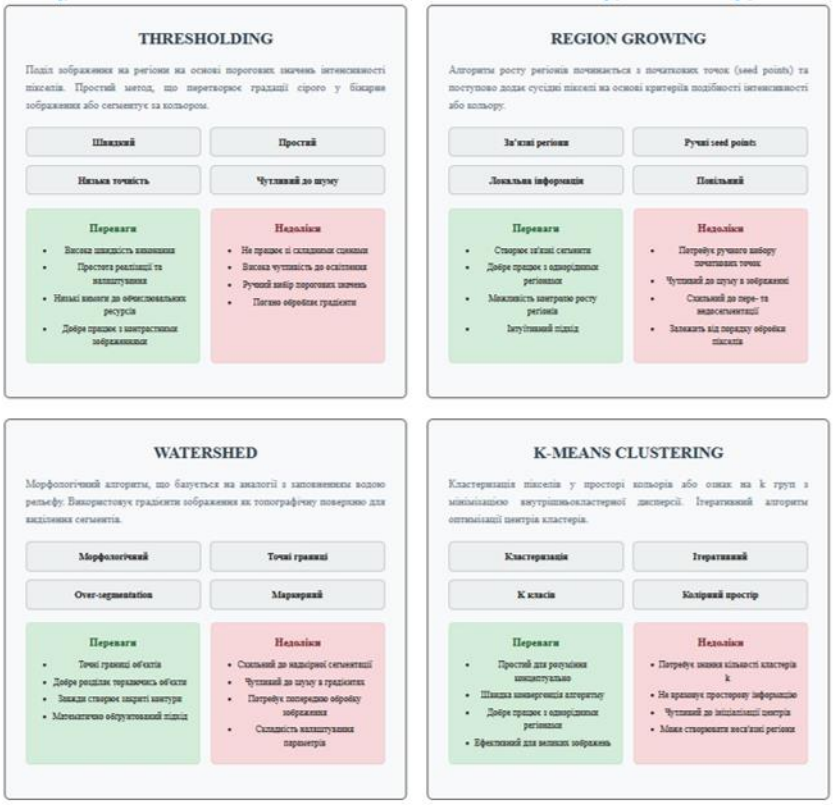
5

CNN, U-Net та SegNet



6

Традиційні методи сегментації зображень

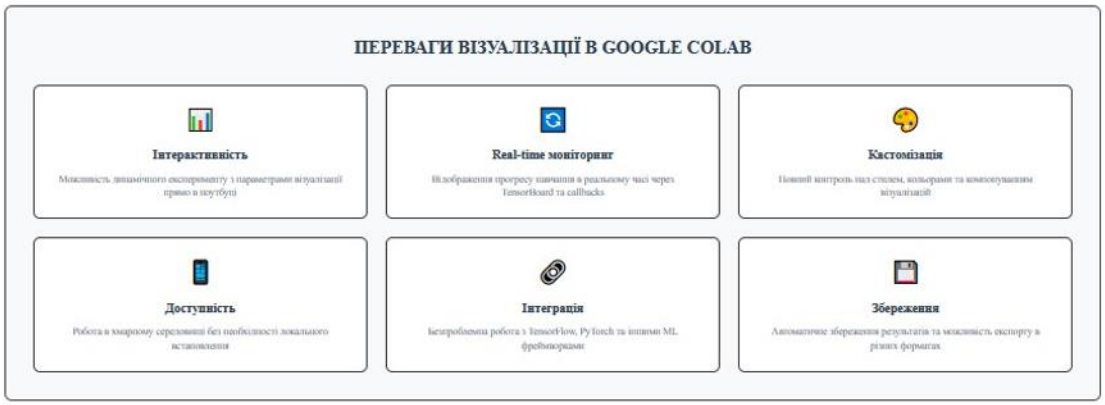


7

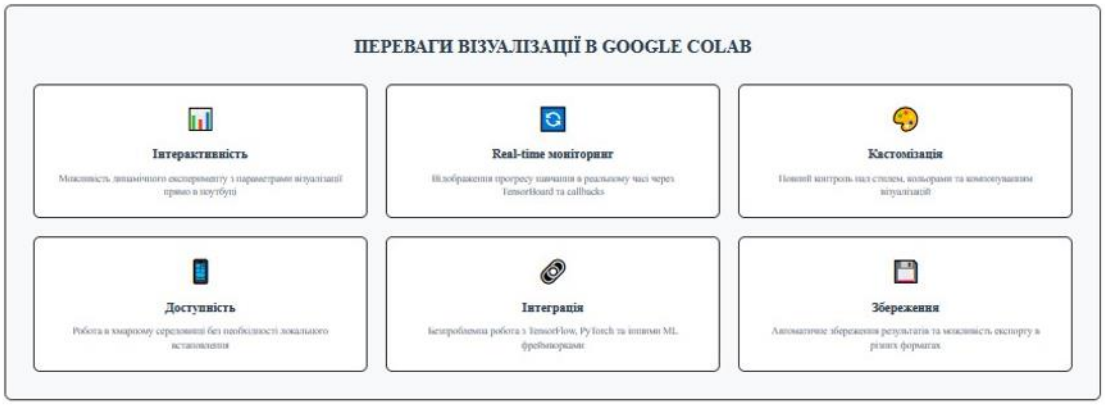
Глибокі методи сегментації зображень

8

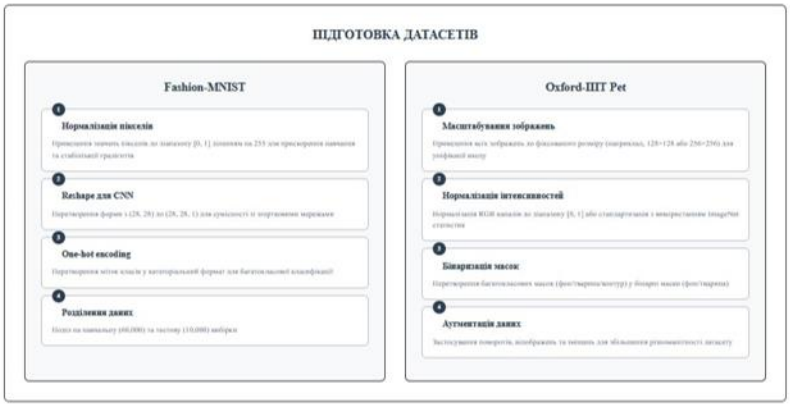
Вибір програмних засобів для реалізації системи



Вибір програмних засобів для реалізації системи



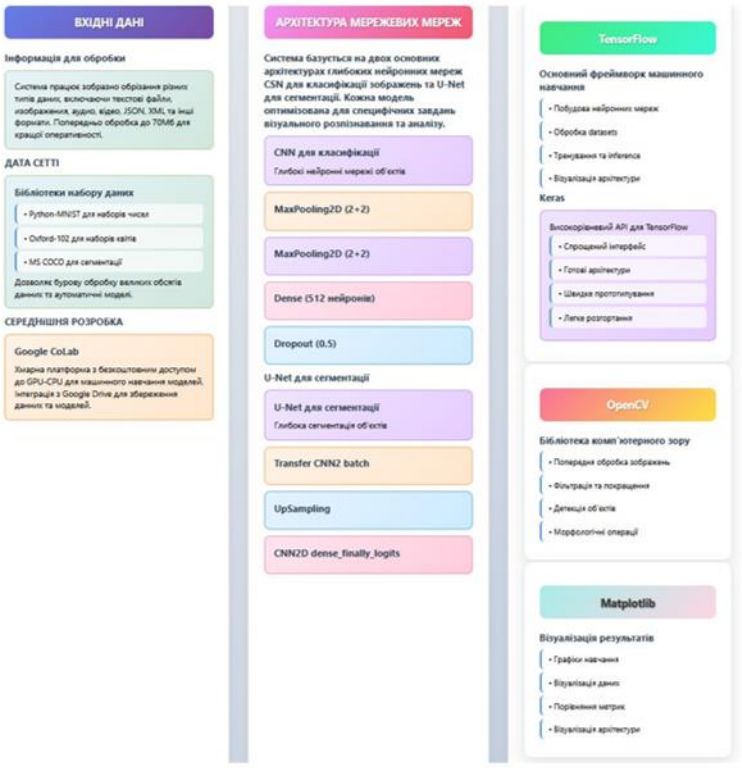
Підготовка датасетів



ТЕХНІЧНІ СПЕЦИФІКАЦІЇ ДАТАСЕТІВ

Характеристика	Fashion-MNIST	Oxford-IIIT Pet
Задача	Класифікація зображень	Семантична сегментація
Розмір датасету	70,000 зображень	7,349 зображень
Розмір зображення	28×28 пікселів	Різний (після resize: 256×256)
Кольорові канали	1 (відтінки сірого)	3 (RGB)
Кількість класів	10 типів одягу	2 класи (фон/тварина)
Тип анотацій	Мітки класів	Піксельні маски
Розмір файлу	~30 MB	~800 MB
Джерело	Zalando Research	Oxford University

Реалізація комп'ютерної системи



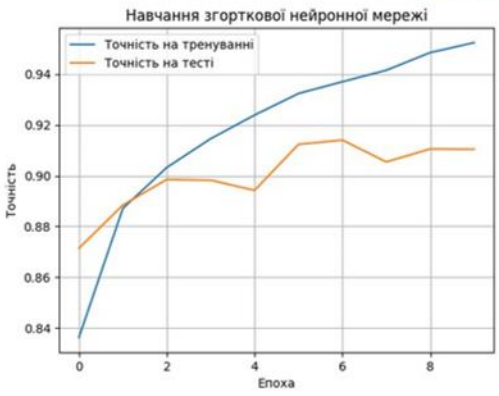
Аналіз результатів

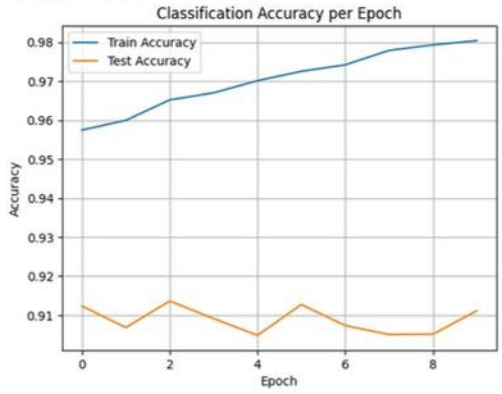
Перші 10 зображень Fashion-MNIST

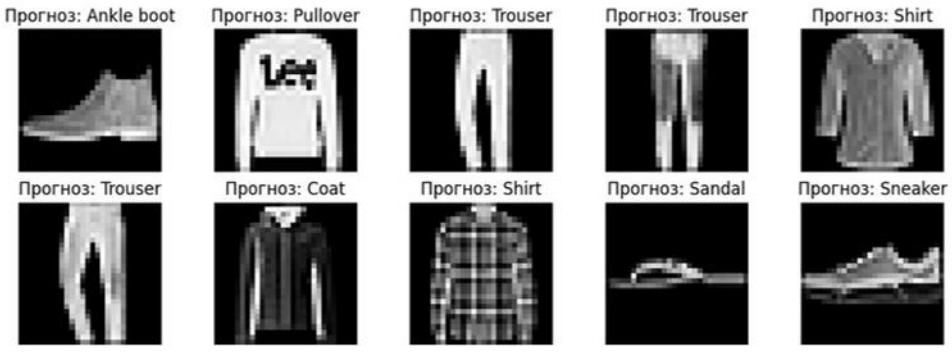
```
        plt.subplot(2, 3, i + 4)
        plt.imshow(tf.squeeze(mask[i]), cmap='gray')
        plt.axis('off')
        plt.title("Маска сегментації")
    plt.suptitle("Приклади з датасету Oxford Pets")
    plt.show()
```

13

Аналіз результатів







14

Висновки

У ході виконання кваліфікаційної роботи було розроблено, реалізовано та проаналізовано комп'ютерну систему, що поєднує задачі класифікації та семантичної сегментації зображень на основі методів глибокого навчання. Дослідження охопило як теоретичні аспекти побудови архітектур нейронних мереж для обробки візуальної інформації, так і практичну реалізацію повного циклу моделювання у середовищі Google Colab із використанням сучасних бібліотек машинного навчання, таких як TensorFlow, Keras та OpenCV.

У рамках реалізації було розроблено згорткову нейронну мережу для класифікації об'єктів зображень із використанням датасету Fashion-MNIST, а також архітектуру U-Net для задачі піксельної сегментації зображень домашніх тварин із набору Oxford-IIIT Pet. Проведено навчання обох моделей із подальшим аналізом функцій втрат, точності класифікації та якості сегментації. Результати підтвердили правильність обраних архітектурних рішень і гіперпараметрів, оскільки обидві моделі досягли високих показників точності та візуальної відповідності з еталонними розмітками.

Особливу увагу приділено етапу візуалізації, що дозволив підтвердити ефективність моделей з точки зору інтерпретованості. Було представлено графіки динаміки навчання, результати класифікації на тестових зображеннях та маски сегментації у зіставленні з реальними. Це дало змогу не лише кількісно, а й якісно оцінити здатність моделей до генералізації та відтворення просторових характеристик об'єктів.

Ключовим досягненням стало інтегрування класифікаційної та сегментаційної моделей у єдиний обчислювальний конвеєр, що забезпечує послідовну та узгоджену обробку зображення. Така інтеграція дозволяє одночасно отримувати глобальну інформацію про категорію зображення і деталізовану карту об'єкта на рівні пікселів. Система може бути масштабована до багатокласових або мультимодальних задач і адаптована до специфічних галузей застосування.

Загалом, результати роботи підтвердили доцільність застосування глибокого навчання для багаторівневої обробки зображень і відкривають перспективи для подальших досліджень у напрямі вдосконалення архітектур, оптимізації процесів навчання та розширення функціональних можливостей системи.

ДОДАТОК Б

Програмний код

Б.1 Лістинг коду

Б.1.1 Імпорт бібліотек

```
import tensorflow as tf
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = x_train / 255.0
x_test = x_test / 255.0
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

plt.figure(figsize=(10, 3))
for i in range(10):
    plt.subplot(1, 10, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.axis('off')
plt.suptitle("Перші 10 зображень Fashion-MNIST")
plt.show()
import tensorflow_datasets as tfds

dataset, info = tfds.load('oxford_iiit_pet', with_info=True)

def preprocess_sample(sample):
    image = tf.image.resize(sample['image'], (128, 128)) / 255.0
    mask = tf.image.resize(sample['segmentation_mask'], (128,
128))
    mask = tf.cast(mask > 0, tf.int32)
    return image, mask

train_ds =
dataset['train'].map(preprocess_sample).batch(32).prefetch(tf.da
ta.AUTOTUNE)
test_ds =
dataset['test'].map(preprocess_sample).batch(32).prefetch(tf.dat
a.AUTOTUNE)
for image, mask in train_ds.take(1):
    plt.figure(figsize=(8, 4))
    for i in range(3):
        plt.subplot(2, 3, i + 1)
```

```
        plt.imshow(image[i])
        plt.axis('off')
        plt.title("Зображення")

        plt.subplot(2, 3, i + 4)
        plt.imshow(tf.squeeze(mask[i]), cmap='gray')
        plt.axis('off')
        plt.title("Маска сегментації")
    plt.suptitle("Приклади з датасету Oxford Pets")
    plt.show()
```

Б.1.2 Побудова моделі для класифікації зображень

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',
input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=10,
validation_data=(x_test, y_test))
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.title('Навчання згорткової нейронної мережі')
plt.legend()
plt.grid(True)
plt.show()
```

Б.1.3 Реалізація моделі сегментації зображень

```
from tensorflow.keras import layers, Model
def unet_model(input_size=(128, 128, 3)):
    inputs = layers.Input(input_size)

    # Енкодер
```

```
    c1 = layers.Conv2D(16, 3, activation='relu',
padding='same')(inputs)
    c1 = layers.Conv2D(16, 3, activation='relu',
padding='same')(c1)
    p1 = layers.MaxPooling2D()(c1)

    c2 = layers.Conv2D(32, 3, activation='relu',
padding='same')(p1)
    c2 = layers.Conv2D(32, 3, activation='relu',
padding='same')(c2)
    p2 = layers.MaxPooling2D()(c2)
    # Ботлнек
    c3 = layers.Conv2D(64, 3, activation='relu',
padding='same')(p2)
    c3 = layers.Conv2D(64, 3, activation='relu',
padding='same')(c3)
    # Декодер
    u1 = layers.UpSampling2D()(c3)
    u1 = layers.Concatenate()([u1, c2])
    c4 = layers.Conv2D(32, 3, activation='relu',
padding='same')(u1)
    c4 = layers.Conv2D(32, 3, activation='relu',
padding='same')(c4)
    u2 = layers.UpSampling2D()(c4)
    u2 = layers.Concatenate()([u2, c1])
    c5 = layers.Conv2D(16, 3, activation='relu',
padding='same')(u2)
    c5 = layers.Conv2D(16, 3, activation='relu',
padding='same')(c5)

    outputs = layers.Conv2D(1, 1, activation='sigmoid')(c5)

    model = Model(inputs, outputs)
    return model

model_unet = unet_model()
model_unet.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
for images, masks in test_ds.take(1):
    preds = model_unet.predict(images)
    plt.figure(figsize=(10, 6))
    for i in range(3):
        plt.subplot(3, 3, i*3+1)
        plt.imshow(images[i])
        plt.title("Зображення")
        plt.axis('off')

        plt.subplot(3, 3, i*3+2)
        plt.imshow(tf.squeeze(masks[i]), cmap='gray')
        plt.title("Реальна маска")
        plt.axis('off')

        plt.subplot(3, 3, i*3+3)
```

```
        plt.imshow(tf.squeeze(preds[i]) > 0.5, cmap='gray')
        plt.title("Передбачена маска")
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

Б.1.4 Навчання моделей і обробка результатів

```
# Навчання класифікаційної моделі (уже виконано раніше, повторно
для графіка)
cnn_history = model.fit(x_train, y_train, epochs=10,
validation_data=(x_test, y_test))
plt.plot(cnn_history.history['accuracy'], label='Train
Accuracy')
plt.plot(cnn_history.history['val_accuracy'], label='Test
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Classification Accuracy per Epoch')
plt.legend()
plt.grid(True)
plt.show()
plt.plot(history_unet.history['loss'], label='Train Loss')
plt.plot(history_unet.history['val_loss'], label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Segmentation Loss per Epoch')
plt.legend()
plt.grid(True)
plt.show()
```

Б.1.5 Візуалізація результатів класифікації і сегментації

```
class_names = ['T-shirt', 'Trouser', 'Pullover', 'Dress',
'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle
boot']

predictions = model.predict(x_test[:10])

plt.figure(figsize=(10, 4))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_test[i].squeeze(), cmap='gray')
    plt.title(f"Прогноз:
{class_names[predictions[i].argmax()]}")
    plt.axis('off')
plt.suptitle("Результати класифікації")
```

```
plt.tight_layout()
plt.show()
for images, masks in test_ds.take(1):
    preds = model_unet.predict(images)
    plt.figure(figsize=(10, 6))
    for i in range(3):
        plt.subplot(3, 3, i*3+1)
        plt.imshow(images[i])
        plt.title("Зображення")
        plt.axis('off')

        plt.subplot(3, 3, i*3+2)
        plt.imshow(tf.squeeze(masks[i]), cmap='gray')
        plt.title("Маска-еталон")
        plt.axis('off')

        plt.subplot(3, 3, i*3+3)
        plt.imshow(tf.squeeze(preds[i]) > 0.5, cmap='gray')
        plt.title("Передбачена маска")
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

Б.1.6 Інтеграція класифікації та сегментації в єдину систему

```
def combined_analysis(image):
    # Зміна розміру до 128x128
    img_resized = tf.image.resize(image, (128, 128))

    # Якщо зображення grayscale, перетворити в RGB
    if img_resized.shape[-1] == 1:
        img_resized = tf.image.grayscale_to_rgb(img_resized)

    # Додати batch-розмір
    img_input = tf.expand_dims(img_resized, axis=0)

    # Класифікація
    class_pred = cnn_model.predict(img_input)
    predicted_class = tf.argmax(class_pred, axis=1).numpy()[0]

    # Сегментація
    seg_pred = unet_model.predict(img_input)
    seg_mask = tf.argmax(seg_pred[0], axis=-1)

    # Візуалізація
    plt.figure(figsize=(12, 4))
    plt.subplot(1, 3, 1)
    plt.imshow(tf.cast(image, tf.uint8))
    plt.title("Original Image")
    plt.axis("off")
```

```
plt.subplot(1, 3, 2)
plt.imshow(seg_mask, cmap='gray')
plt.title("Predicted Segmentation")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.bar([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], class_pred[0])
plt.title(f"Predicted Class: {predicted_class}")
plt.show()
```