Лабораторна робота № 15. Дослідження та застосування мультимодальних моделей для аналізу та генерації візуально-текстового контенту

Мета: ознайомитися з принципами роботи та можливостями мультимодальних моделей, навчитися аналізувати зображення, генерувати підписи до них, створювати історії на основі зображень, а також експериментувати з різними моделями та їх параметрами.

Завдання для всіх варіантів

1. ***Завдання 1:*** *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*
2. ***Завдання 2:*** *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*
3. ***Завдання 3:*** *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*
4. ***Завдання 4:*** *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*
5. ***Завдання 5:*** *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

5.1. ***Завдання 5.1:*** *оберіть 5-7 зображень, які можуть скласти послідовну історію.*

5.2. ***Завдання 5.2:*** *використайте функцію `image_captioning` для створення підпису до кожного зображення.*

5.3. ***Завдання 5.3:*** *об'єднайте ці підписи в коротку історію, додаючи сполучні фрази між ними.*

** За необхідності внесіть додаткові зміни в програмний код*

Базова структура роботи

Блок 1. Встановлення та імпорт бібліотек

```
!pip install gradio transformers pillow requests

import torch
from PIL import Image
import requests
from transformers import pipeline, CLIPProcessor, CLIPModel,
AutoProcessor, AutoModelForCausalLM
import gradio as gr
```

Блок 2. Дослідження VQA

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def visual_question_answering(image, question, model_name):
    vqa_pipeline = pipeline("visual-question-answering", model=model_name)
    result = vqa_pipeline(image, question, top_k=1)
    return result[0]['answer']

def process_image1(image, question, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    answer = visual_question_answering(image, question, model_name)
    return f"Відповідь: {answer}"

vqa_models = [
    "Salesforce/blip-vqa-capfilt-large",
    "microsoft/git-base-vqav2",
    "dandelin/vilt-b32-finetuned-vqa",
    "Salesforce/blip-vqa-capfilt-large"
]

iface = gr.Interface(
    fn=process_image1,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть питання"),
        gr.Dropdown(choices=vqa_models, label="Виберіть модель VQA")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей візуального питання-відповіді (VQA)",
    description="Завантажте зображення, введіть питання та виберіть модель
для тестування."
)
```

```
iface.launch()
```

Завдання 1: *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 3. Створення описів до зображень

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_captioning(image, model_name):
    captioning_pipeline = pipeline("image-to-text", model=model_name)
    result = captioning_pipeline(image)
    return result[0]['generated_text']

def process_image2(image, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    caption = image_captioning(image, model_name)
    return f"Підпис до зображення: {caption}"

vit_models = [
    "nlpconnect/vit-gpt2-image-captioning",
    "microsoft/git-base-coco",
    "Salesforce/blip-image-captioning-base",
    "ydshieh/vit-gpt2-coco-en"
]

iface = gr.Interface(
    fn=process_image2,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Dropdown(choices=vit_models, label="Виберіть ViT модель")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування ViT моделей для створення підписів до зображень",
    description="Завантажте зображення та виберіть модель для створення
підпису."
)
iface.launch()
```

Завдання 2: *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 4. Пошук зображень за текстом

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_text_similarity(image, text_queries, model_name):
    model = CLIPModel.from_pretrained(model_name)
    processor = CLIPProcessor.from_pretrained(model_name)

    inputs = processor(text=text_queries, images=image,
return_tensors="pt", padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
    logits_per_image = outputs.logits_per_image
    probs = logits_per_image.softmax(dim=1)

    return probs.tolist()[0]

clip_models = [
    "openai/clip-vit-large-patch14",
    "laion/CLIP-ViT-H-14-laion2B-s32B-b79K"
]

def process_image_and_text(image, text_queries, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."

    queries = [q.strip() for q in text_queries.split(',')]
    if not queries:
        return "Будь ласка, введіть текстові запити."

    similarities = image_text_similarity(image, queries, model_name)

    result = ""
    for query, similarity in zip(queries, similarities):
        result += f"Подібність до '{query}': {similarity:.4f}\n"

    return result

iface = gr.Interface(
    fn=process_image_and_text,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть текстові запити (розділені комами)"),
        gr.Dropdown(choices=clip_models, label="Виберіть модель CLIP")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей подібності зображення та тексту",
    description="Завантажте зображення, введіть текстові запити та
виберіть модель для оцінки подібності."
```

```
)

iface.launch()
```

Завдання 3: *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 5. Візуальний детектив

```
model_name = "Salesforce/blip-vqa-capfilt-large"
image_url = "https://www.youngminds.org.uk/media/qkxfwq40/group-of-young-people-talking.jpg"
questions = [
    ("What is in the foreground?", 1),
    ("How many people do you see in the image?", 2),
    ("What mood does this scene convey?", 3)
]

score = 0
for question, points in questions:
    answer = visual_question_answering(image_url, question, model_name)

    print(f"Питання: {question}")
    print(f"Відповідь моделі: {answer}")
    user_correct = input("Чи правильна відповідь? (так/ні): ").lower() == 'так'
    if user_correct:
        score += points
        print(f"Чудово! Ви отримали {points} балів.")
    else:
        print("На жаль, відповідь неправильна.")
    print()

print(f"Ваш загальний рахунок: {score}")
```

Завдання 4: *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*

Блок 6. Створення історії за зображеннями

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Функція для генерації зв'язної історії
def generate_coherent_story(captions):
    # Ініціалізуємо модель BLOOM
    model_name = "bigscience/bloom-560m"  # оберіть модель
    model = AutoModelForCausalLM.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
    # Створюємо промпт для моделі
    prompt = "Create a coherent story based on the following scenes:\n\n"
    for i, caption in enumerate(captions, 1):
        prompt += f"Scene {i}: {caption}\n"
    prompt += "\nStory:"

    # Генеруємо історію
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids

    attention_mask = torch.ones(input_ids.shape, dtype=torch.long,
device=input_ids.device)

    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        max_length=300,
        num_return_sequences=1,
        temperature=0.7,
        no_repeat_ngram_size=2,
        do_sample=True
    )
    story = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return story.split("Story:")[1].strip()

# Основний код
image_urls = [
    "https://t3.ftcdn.net/jpg/02/43/12/34/360_F_243123463_zTooub557xEWABDL
k0jJklDyLSGl2jrr.jpg",
    "http://images.cocodataset.org/val2017/000000039769.jpg",
    "https://www.loveyourlandscape.org/media/19953/mighty-oak-tree-from-
below-000056382958_large.jpg"
    # Додайте додаткових зображень
]

model_name = "nlpconnect/vit-gpt2-image-captioning" # оберіть модель


captions = []
for i, url in enumerate(image_urls, 1):
    caption = image_captioning(url, model_name)
    captions.append(caption)
    print(f"Part {i}: {caption}")

print("\nЗгенеровані підписи до зображень:")
for i, caption in enumerate(captions, 1):
    print(f"Part {i}: {caption}")
```

```
print("\nГенеруємо зв'язну історію...")
coherent_story = generate_coherent_story(captions)

print("\nЗгенерована зв'язна історія:")
print(coherent_story)
```

Завдання 5: *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

Приклад виконання роботи

Блок 1. Встановлення та імпорт бібліотек

```
!pip install gradio transformers pillow requests

import torch
from PIL import Image
import requests
from transformers import pipeline, CLIPProcessor, CLIPModel,
AutoProcessor, AutoModelForCausalLM
import gradio as gr
```

Блок 2. Дослідження VQA

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def visual_question_answering(image, question, model_name):
    vqa_pipeline = pipeline("visual-question-answering", model=model_name)
    result = vqa_pipeline(image, question, top_k=1)
    return result[0]['answer']

def process_image1(image, question, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    answer = visual_question_answering(image, question, model_name)
    return f"Відповідь: {answer}"

vqa_models = [
    "Salesforce/blip-vqa-capfilt-large",
    "microsoft/git-base-vqav2",
    "dandelin/vilt-b32-finetuned-vqa",
    "Salesforce/blip-vqa-capfilt-large"
]

iface = gr.Interface(
    fn=process_image1,
    inputs=[
```

```
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть питання"),
        gr.Dropdown(choices=vqa_models, label="Виберіть модель VQA")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей візуального питання-відповіді (VQA)",
    description="Завантажте зображення, введіть питання та виберіть модель для тестування."
)

iface.launch()
```

Завдання 1: *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

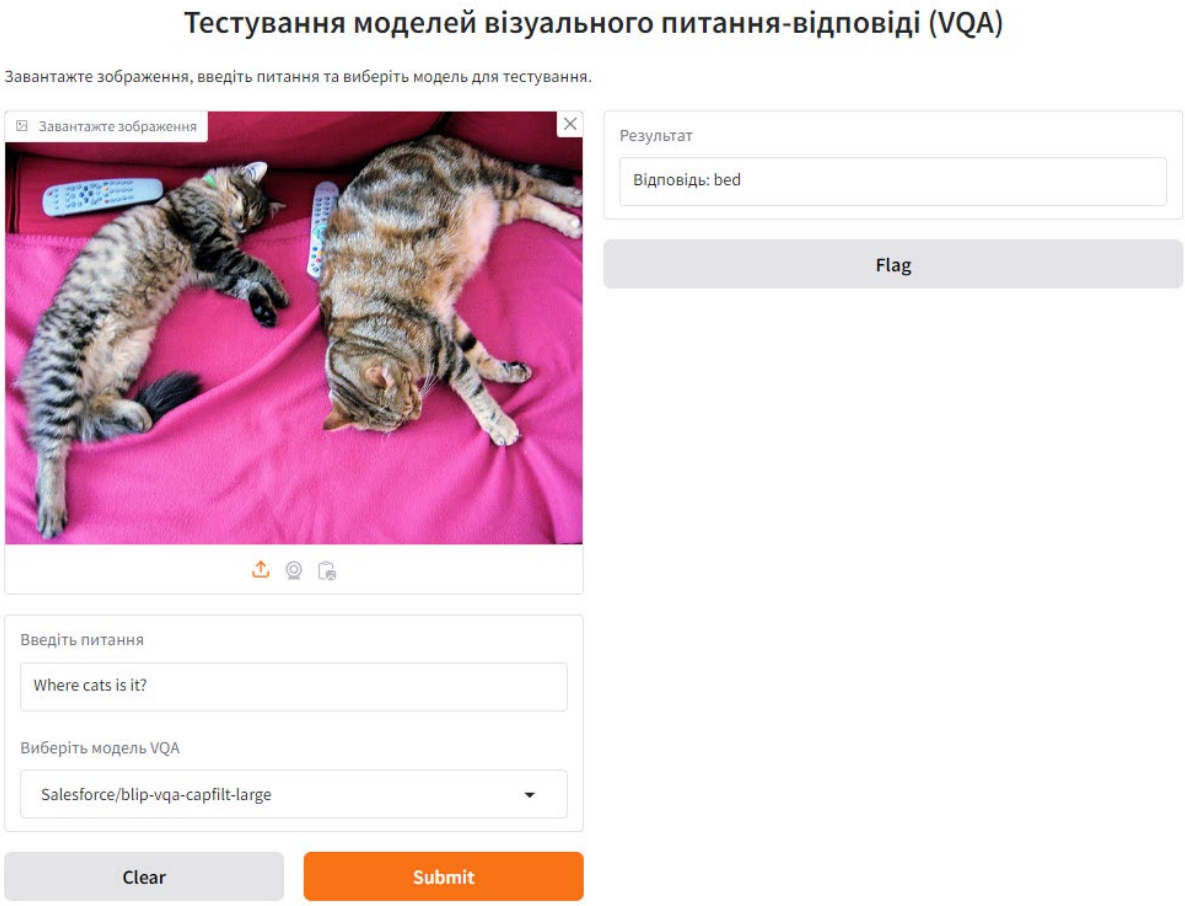

Рисунок 1.1. Результат обробки запитання моделлю blip-vqa-capfilt-large

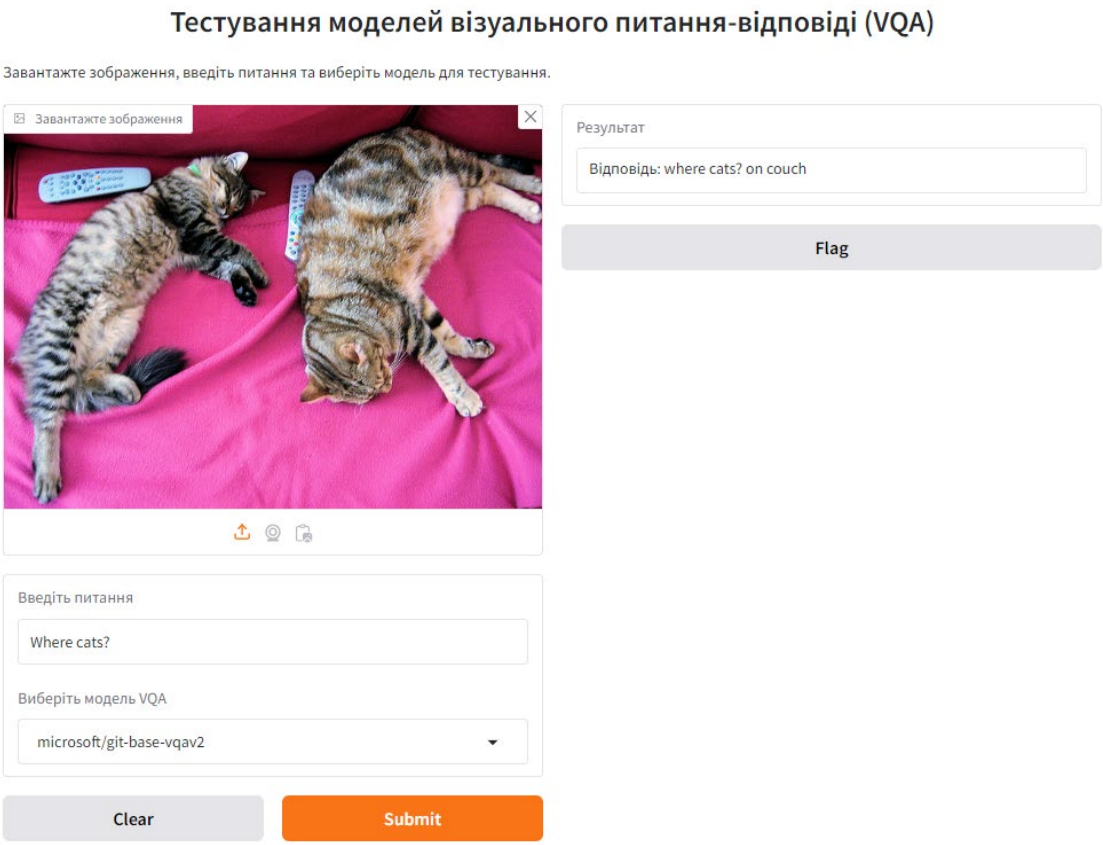

Рисунок 1.2 Результат обробки запитання моделлю git-base-vqav2

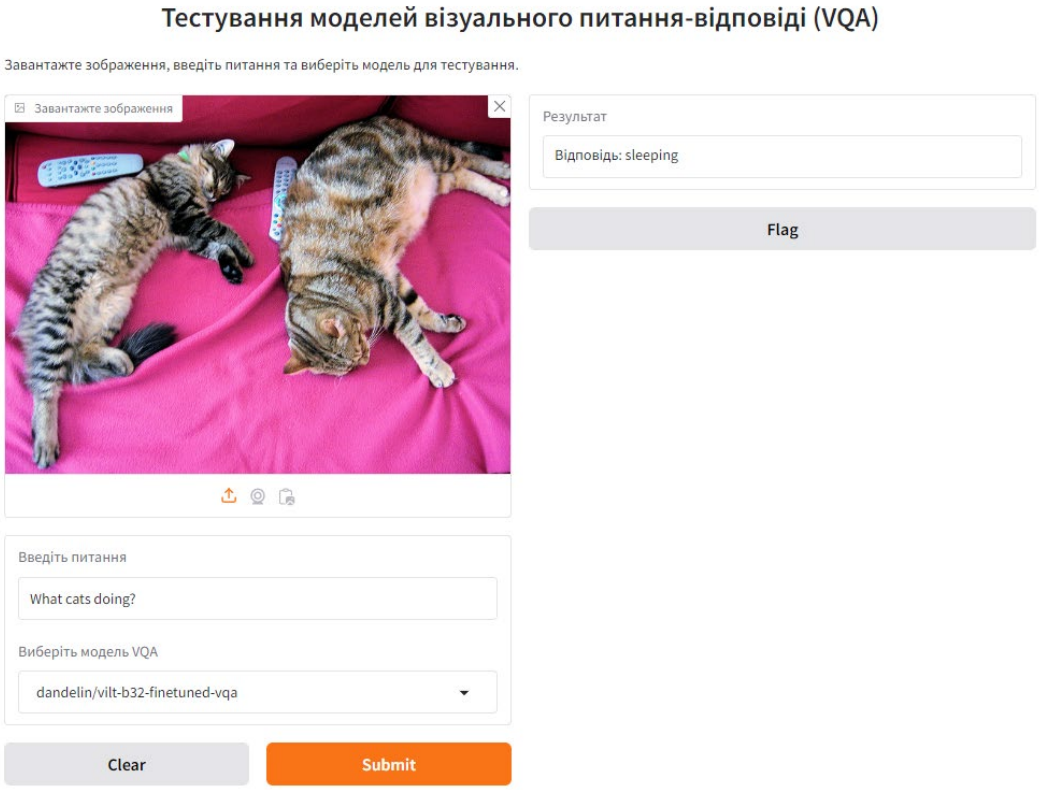

Рисунок 1.3 Результат обробки запитання моделлю vilt-b32-finetuned-vqa

Блок 3. Створення описів до зображень

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_captioning(image, model_name):
    captioning_pipeline = pipeline("image-to-text", model=model_name)
    result = captioning_pipeline(image)
    return result[0]['generated_text']

def process_image2(image, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    caption = image_captioning(image, model_name)
    return f"Підпис до зображення: {caption}"

vit_models = [
    "nlpconnect/vit-gpt2-image-captioning",
    "microsoft/git-base-coco",
    "Salesforce/blip-image-captioning-base",
    "ydshieh/vit-gpt2-coco-en"
]

iface = gr.Interface(
    fn=process_image2,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Dropdown(choices=vit_models, label="Виберіть ViT модель")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування ViT моделей для створення підписів до зображень",
    description="Завантажте зображення та виберіть модель для створення
підпису."
)
iface.launch()
```

Завдання 2: *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

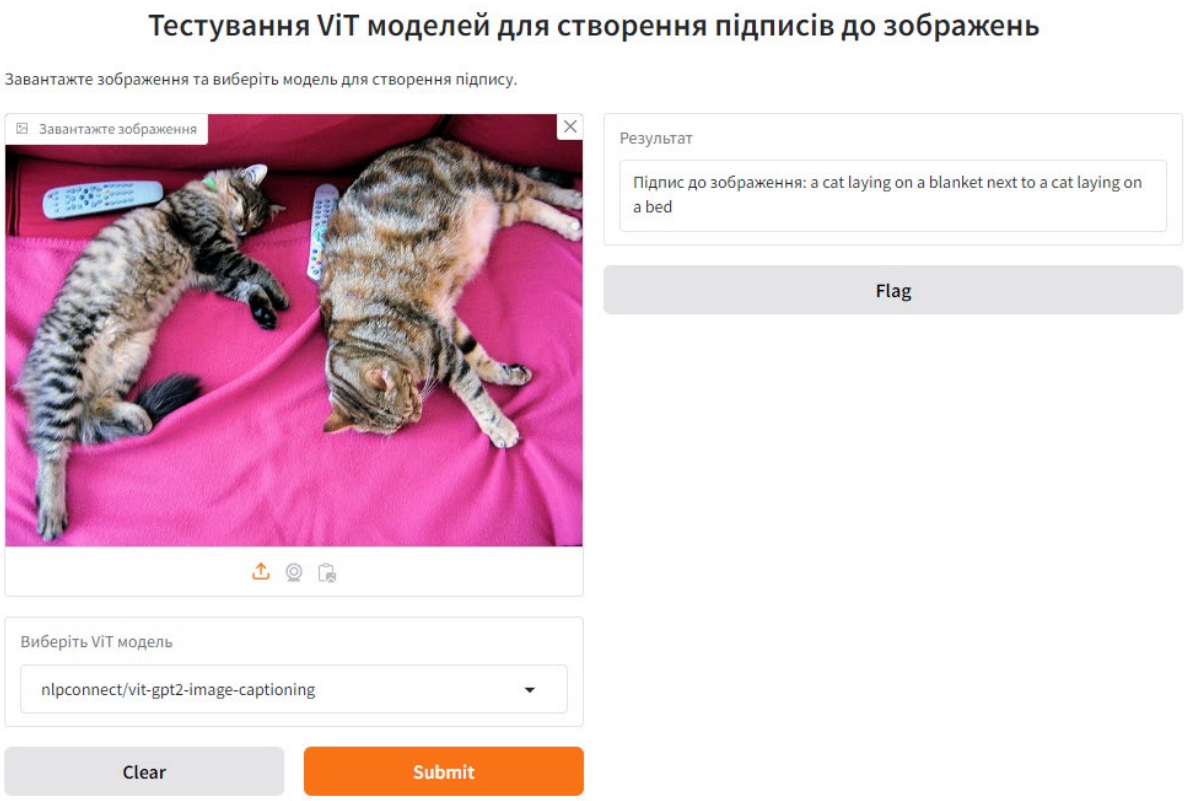

Рисунок 2.1. Результат надання опису моделлю vit-gpt2-image-captioning

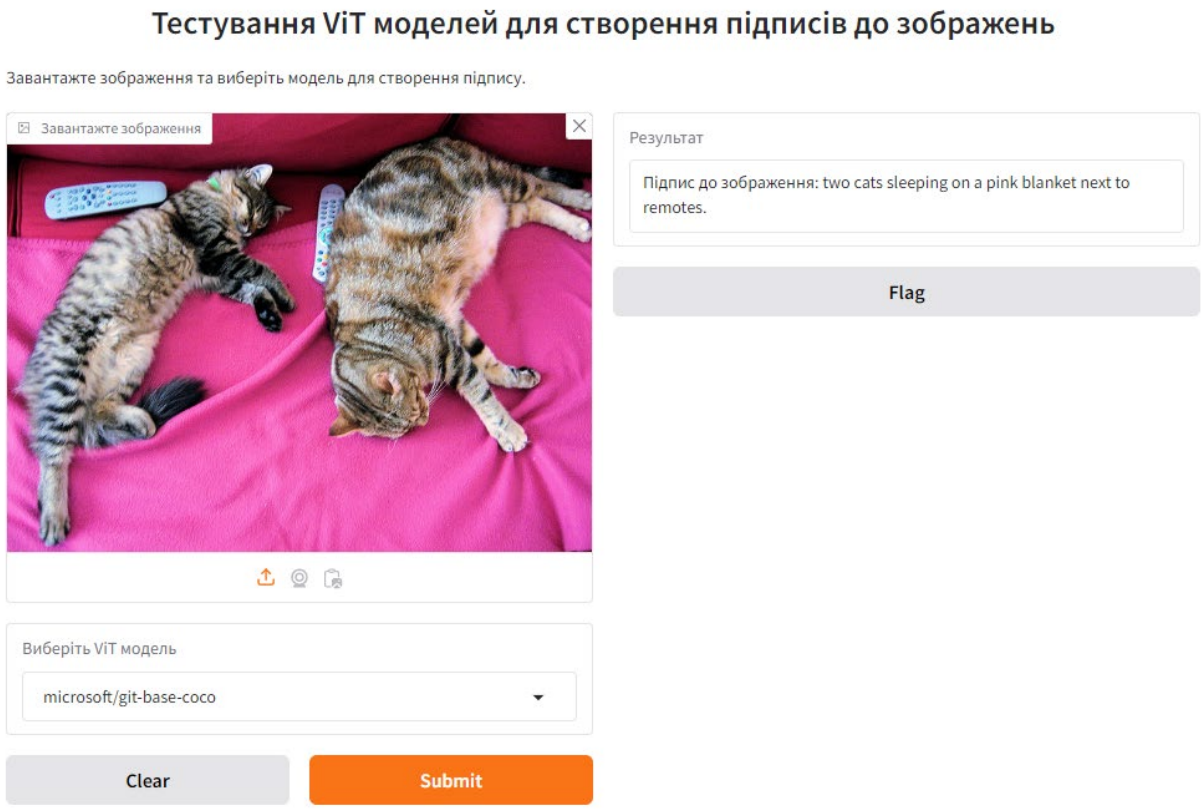

Рисунок 2.2. Результат надання опису моделлю git-base-coco

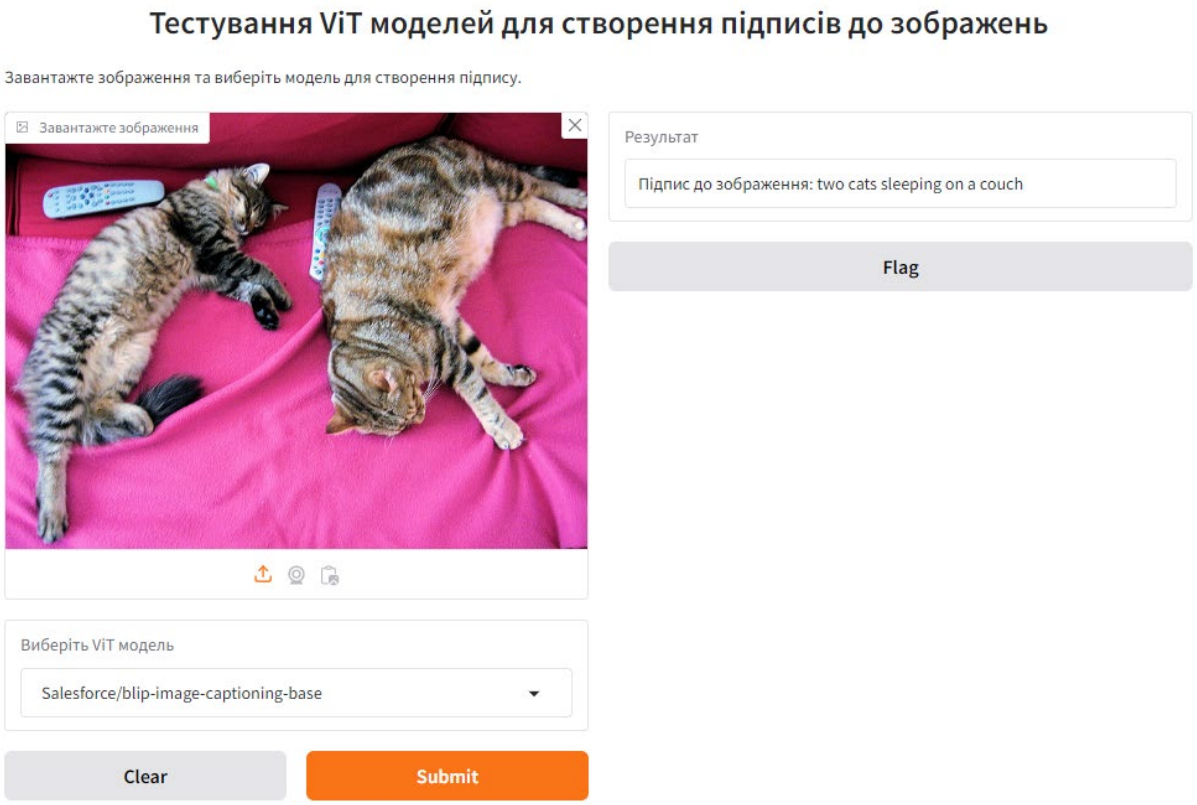

Рисунок 2.3. Результат надання опису моделлю blip-image-captioning-base

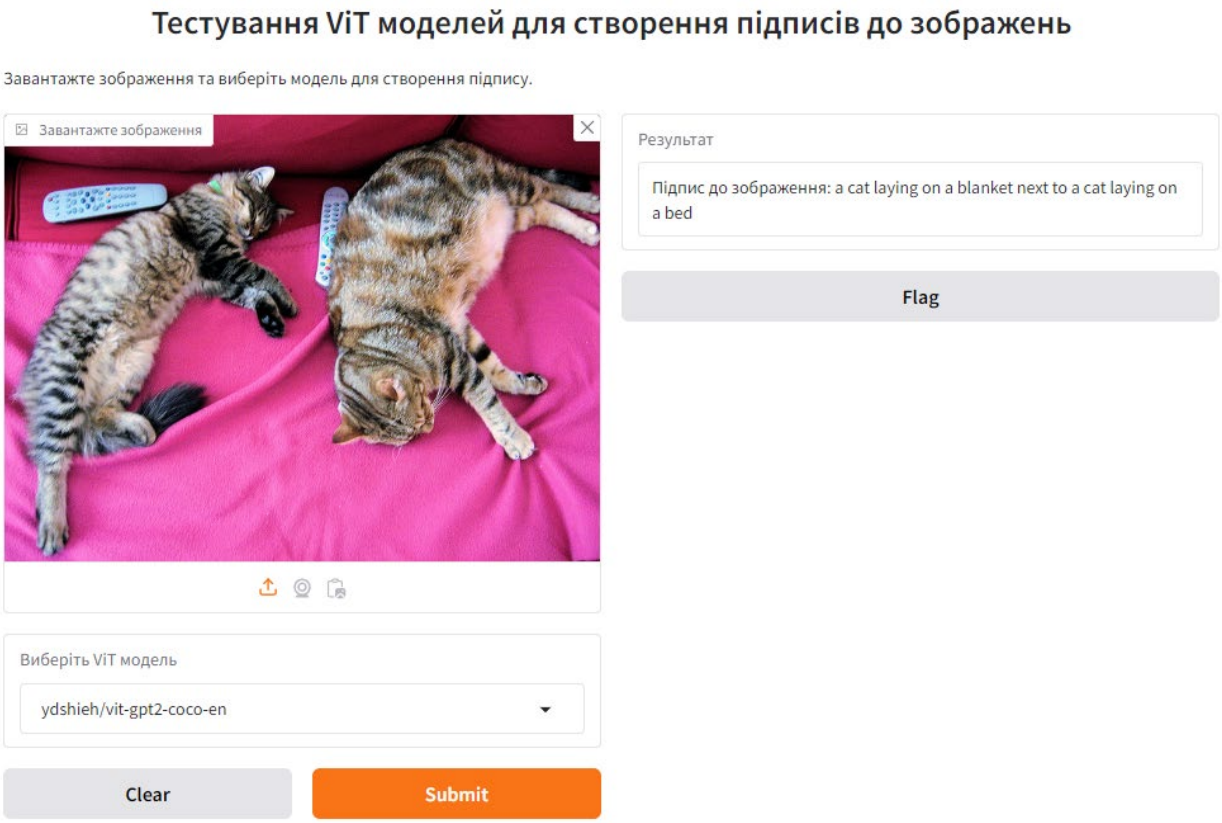

Рисунок 2.4. Результат надання опису моделлю vit-gpt2-coco-en

Блок 4. Пошук зображень за текстом

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_text_similarity(image, text_queries, model_name):
    model = CLIPModel.from_pretrained(model_name)
    processor = CLIPProcessor.from_pretrained(model_name)

    inputs = processor(text=text_queries, images=image,
return_tensors="pt", padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
    logits_per_image = outputs.logits_per_image
    probs = logits_per_image.softmax(dim=1)

    return probs.tolist()[0]

clip_models = [
    "openai/clip-vit-large-patch14",
    "laion/CLIP-ViT-H-14-laion2B-s32B-b79K"
]

def process_image_and_text(image, text_queries, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."

    queries = [q.strip() for q in text_queries.split(',')]
    if not queries:
        return "Будь ласка, введіть текстові запити."

    similarities = image_text_similarity(image, queries, model_name)

    result = ""
    for query, similarity in zip(queries, similarities):
        result += f"Подібність до '{query}': {similarity:.4f}\n"

    return result

iface = gr.Interface(
    fn=process_image_and_text,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть текстові запити (розділені комами)"),
        gr.Dropdown(choices=clip_models, label="Виберіть модель CLIP")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей подібності зображення та тексту",
    description="Завантажте зображення, введіть текстові запити та
виберіть модель для оцінки подібності."
```

```
)

iface.launch()
```

Завдання 3: *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*

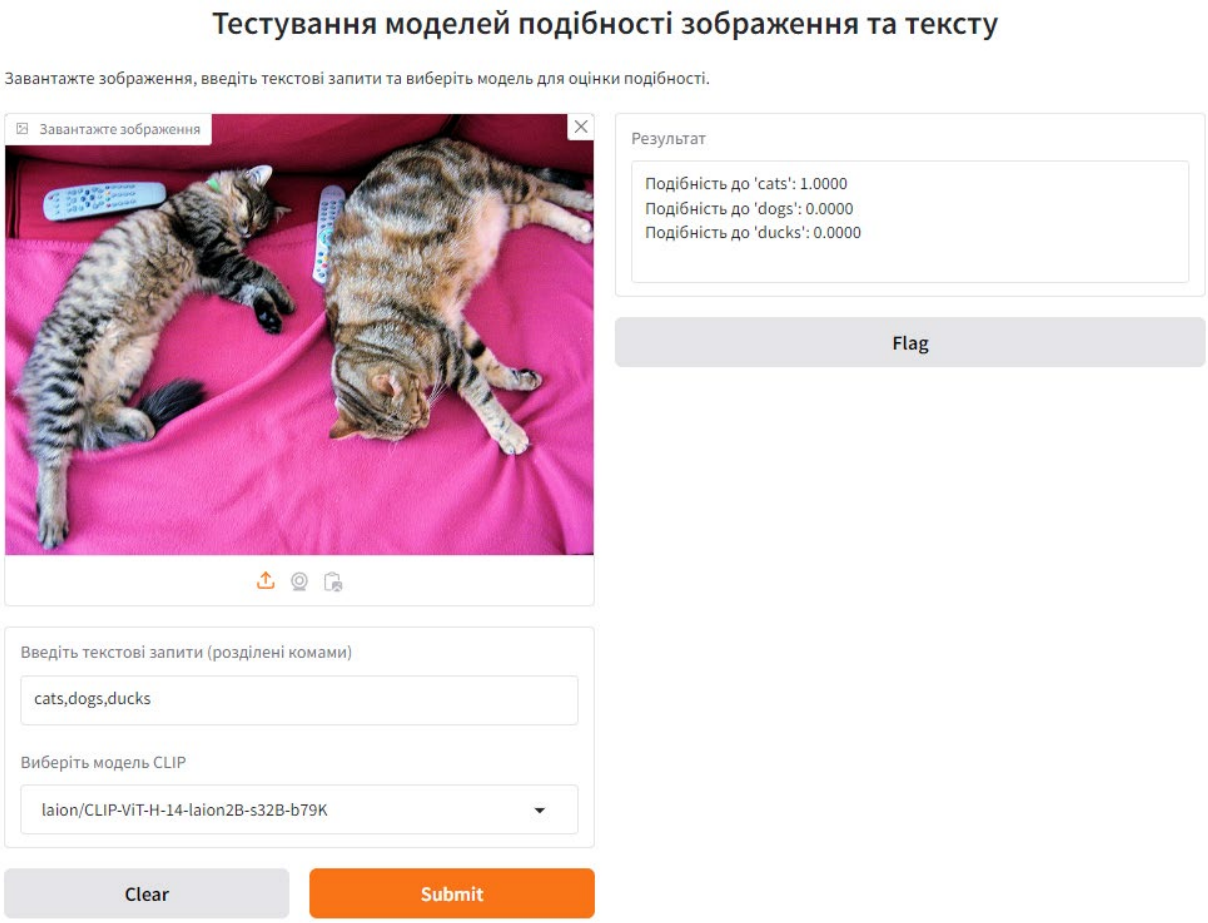

Рисунок 3.1. Результат оцінки подібності моделлю CLIP-ViT-H-14-laion2B-s32B-b79K

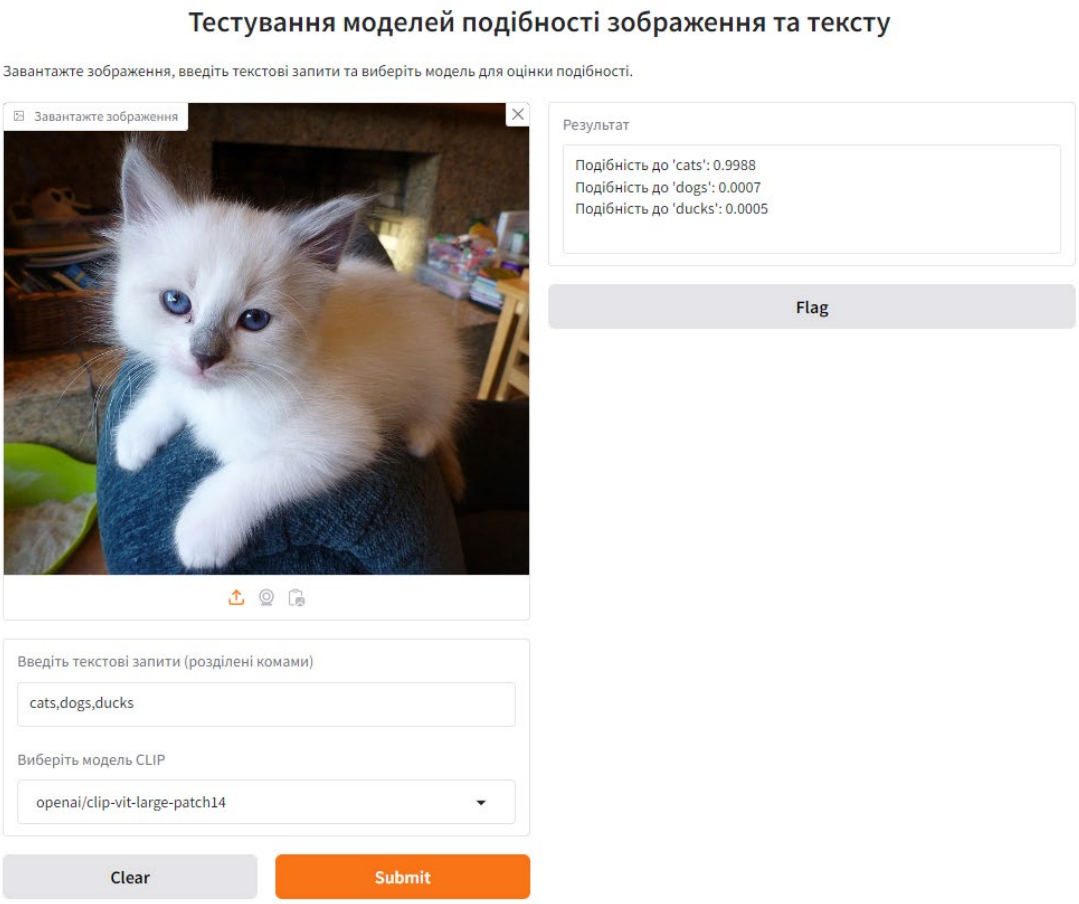

Рисунок 3.2. Результат оцінки подібності моделлю clip-vit-large-patch14

Блок 5. Візуальний детектив

```
model_name = "Salesforce/blip-vqa-capfilt-large"
image_url = "https://www.youngminds.org.uk/media/qkxfwq40/group-of-young-
people-talking.jpg"
questions = [
    ("What is in the foreground?", 1),
    ("How many people do you see in the image?", 2),
    ("What mood does this scene convey?", 3)
]

score = 0
for question, points in questions:
    answer = visual_question_answering(image_url, question, model_name)

    print(f"Питання: {question}")
    print(f"Відповідь моделі: {answer}")
    user_correct = input("Чи правильна відповідь? (так/ні): ").lower() ==
'так'
    if user_correct:
        score += points
        print(f"Чудово! Ви отримали {points} балів.")
```

```
    else:
        print("На жаль, відповідь неправильна.")
    print()

print(f"Ваш загальний рахунок: {score}")
```

Завдання 4: *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1258: UserWarn
  warnings.warn(
Питання: What is in the foreground?
Відповідь моделі: woman
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 1 балів.

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
Питання: How many people do you see in the image?
Відповідь моделі: 4
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 2 балів.

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
Питання: What mood does this scene convey?
Відповідь моделі: happy
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 3 балів.

Ваш загальний рахунок: 6
```

Рисунок 4.1. Результат оцінки роботи «Візуального детективу»

Блок 6. Створення історії за зображеннями

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Функція для генерації зв'язної історії
def generate_coherent_story(captions):
    # Ініціалізуємо модель BLOOM
    model_name = "bigscience/bloom-560m"  # оберіть модель
    model = AutoModelForCausalLM.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Створюємо промпт для моделі
    prompt = "Create a coherent story based on the following scenes:\n\n"
    for i, caption in enumerate(captions, 1):
        prompt += f"Scene {i}: {caption}\n"
    prompt += "\nStory:"

    # Генеруємо історію
```

```
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids

    attention_mask = torch.ones(input_ids.shape, dtype=torch.long,
device=input_ids.device)

    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        max_length=300,
        num_return_sequences=1,
        temperature=0.7,
        no_repeat_ngram_size=2,
        do_sample=True
    )
    story = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return story.split("Story:")[1].strip()

# Основний код
image_urls = [
    "https://t3.ftcdn.net/jpg/02/43/12/34/360_F_243123463_zTooub557xEWABDL
k0jJklDyLSGl2jrr.jpg",
    "http://images.cocodataset.org/val2017/000000039769.jpg",
    "https://www.loveyourlandscape.org/media/19953/mighty-oak-tree-from-
below-000056382958_large.jpg"
    # Додайте додаткових зображень
]

model_name = "nlpconnect/vit-gpt2-image-captioning" # оберіть модель


captions = []
for i, url in enumerate(image_urls, 1):
    caption = image_captioning(url, model_name)
    captions.append(caption)
    print(f"Part {i}: {caption}")

print("\nЗгенеровані підписи до зображень:")
for i, caption in enumerate(captions, 1):
    print(f"Part {i}: {caption}")

print("\nГенеруємо зв'язну історію...")
coherent_story = generate_coherent_story(captions)

print("\nЗгенерована зв'язна історія:")
print(coherent_story)
```

Завдання 5: *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 1: a man in a suit and tie smiling
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 2: a cat laying on a blanket next to a cat laying on a bed
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 3: a tree with a lot of leaves and a few trees

Згенеровані підписи до зображень:
Part 1: a man in a suit and tie smiling
Part 2: a cat laying on a blanket next to a cat laying on a bed
Part 3: a tree with a lot of leaves and a few trees

Генеруємо зв'язну історію...

Згенерована зв'язна історія:
The man sees a couple of cats laying in the bush and he decides to take them to the hospital. He decides that the cat
```

Рисунок 5.1. Результат створення історії за зображеннями

Згенерована історія: «The man sees a couple of cats laying in the bush and he decides to take them to the hospital. He decides that the cat is so small that it is impossible to get it to stay on its bed. The cat has to climb up onto a branch and then it climbs right into the man's suit. As the couple sits down they see a little tree and they decide to come back to this tree to see what the tree looks like and to have a look at the forest. There are some twigs in this forest that are just too small and this cat cann't get to them. They decide that they are going to try a different method of climbing and the sun is shining and so they climb a ladder instead and climb into a huge tree that has a pool of water. This cat gets to its feet and it has some food on it and starts to eat it. She does not even have to leave the water to touch it, it just gets a taste in its mouth and tries to swallow it for a while. Then, when it gets hungry, the two of them try to go over the pool and find a nearby tree, this time the bigger tree».

Висновки: …

Welcome to Google Colab

https://colab.research.google.com/drive/1TCLyv2gXPiIgNNAxr3ljw2FylkI3Ezj-?usp=sharing