Лабораторна робота №4

Згорткові нейроні мережі

Мета: ознайомитись з основними можливостями створення згорткових нейронних мереж за допомогою tensorflow

Виконання роботи

1. Ознайомитись з кодом в додатку 1;

```
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras import layers, models
from PIL import Image
import numpy as np

splits = ['train[:80%]', 'train[80%:90%]', 'train[90%:]'] # train/val/test
(train_ds, val_ds, test_ds), ds_info = tfds.load(
  'cats_vs_dogs',
  split=splits,
  as_supervised=True, # (image, label)
  with_info=True
)

IMG_SIZE = 128
BATCH = 32
AUTOTUNE = tf.data.AUTOTUNE

def preprocess(img, label):
  img = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))
  img = tf.cast(img, tf.float32) / 255.0
  return img, label

train_ds = (train_ds
    .shuffle(10_000)
    .map(preprocess, num_parallel_calls=AUTOTUNE)
    .batch(BATCH)
    .prefetch(AUTOTUNE))

val_ds = (val_ds
    .map(preprocess, num_parallel_calls=AUTOTUNE)
    .batch(BATCH)
    .prefetch(AUTOTUNE))

test_ds = (test_ds
    .map(preprocess, num_parallel_calls=AUTOTUNE)
    .batch(BATCH)
    .prefetch(AUTOTUNE))

model = models.Sequential([
  layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3)),
  layers.Conv2D(16, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(64, activation='relu'),
  layers.Dense(1, activation='sigmoid')
])

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

model.summary()
# history = model.fit(
#   train_ds,
#   validation_data=val_ds,
#   epochs=3
# )
def predict(path):
  img = Image.open(path).convert("RGB").resize((IMG_SIZE, IMG_SIZE))
  x = np.array(img, np.float32)/255.0
  p = model.predict(x[None], verbose=0)[0,0]
  print("dog" if p>=0.5 else "cat", float(p))

predict("cat1.jpg")
```

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 126, 126, 16)	448
max_pooling2d_4 (MaxPooling2D)	(None, 63, 63, 16)	0
conv2d_5 (Conv2D)	(None, 61, 61, 32)	4,640
max_pooling2d_5 (MaxPooling2D)	(None, 30, 30, 32)	0
flatten_2 (Flatten)	(None, 28800)	0
dense_4 (Dense)	(None, 64)	1,843,264
dense_5 (Dense)	(None, 1)	65

```
Total params: 1,848,417 (7.05 MB)
Trainable params: 1,848,417 (7.05 MB)
Non-trainable params: 0 (0.00 B)
cat 0.46675071120262146
```

Рис 1.1

2. Ознайомитись з вхідними даними датасету та вивести по одному зображенню з вибірок train/val/test;

					МІВТ.420.002.002-ЛР4		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Величко В.С.			ПЗІВС	Літ.	Арк. 1 / Акрушів 2
Перевір.		Рижук А.В					
Н. Контр.					Звіт з лабораторних робіт	ДУ "Житомирська політехніка", ІВТ-6	
Затверд.							

```python
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt

splits = ['train[:80%]', 'train[80%:90%]', 'train[90%:]']

(train_ds, val_ds, test_ds), ds_info = tfds.load(
    'cats_vs_dogs',
    split=splits,
    as_supervised=True,
    with_info=True
)

print("Назва датасету:", ds_info.name)
print("Кількість класів:", ds_info.features['label'].num_classes)
print("Мітки класів:", ds_info.features['label'].names)
print("Приклад розміру зображення:", ds_info.features['image'].shape)

print("Кількість прикладів у train/val/test:")
print(" Train:", int(ds_info.splits['train'].num_examples * 0.8))
print(" Val:  ", int(ds_info.splits['train'].num_examples * 0.1))
print(" Test: ", int(ds_info.splits['train'].num_examples * 0.1))

def show_one_sample(dataset, title):
    for img, label in dataset.take(1):
        plt.imshow(img)
        plt.title(f"{title} — {'Собака' if label == 1 else 'Кішка'}")
        plt.axis("off")
        plt.show()
        break

show_one_sample(train_ds, "Train (навчальна вибірка)")
show_one_sample(val_ds, "Validation (перевірочна вибірка)")
show_one_sample(test_ds, "Test (тестова вибірка)")
```

Рис 1.2



Рис 1.3

3. З Запустити процес навчання моделі та вивести графіки точності та втрат для датасету train та val;

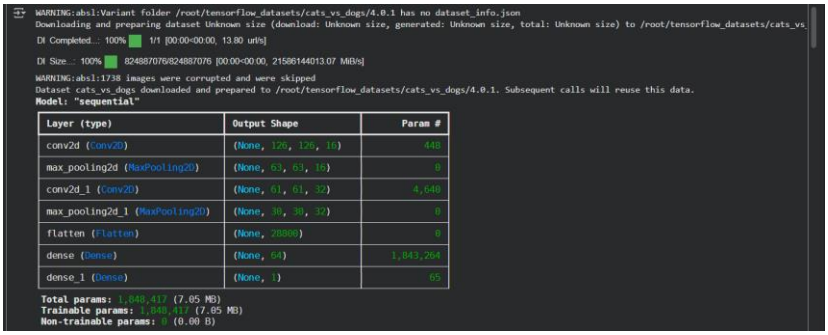


Рис 1.6

4. Завантажити в оперативну пам’ять зображення задане викладачем та зробити передбачення чи правильно навчена модель;

```python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

IMG_SIZE = 128
model_path = '/content/drive/MyDrive/cats_dogs.keras'
image_path = '/content/cat6.jpg'

model = tf.keras.models.load_model(model_path)

filename = os.path.basename(image_path).lower()
if "dog" in filename:
    true_label = "Собака"
elif "cat" in filename:
    true_label = "Кіт"
else:
    true_label = "Невідомо"

img = Image.open(image_path).convert('RGB').resize((IMG_SIZE, IMG_SIZE))
x = np.array(img, dtype=np.float32) / 255.0
x = np.expand_dims(x, axis=0)

pred = model.predict(x, verbose=0)[0, 0]
pred_label = "Собака" if pred >= 0.5 else "Кіт"

correct = (true_label == pred_label)
color = "green" if correct else "red"

plt.imshow(img)
plt.title(f"""
Істина: {true_label}
Прогноз: {pred_label}
Ймовірність: {pred:.4f}
""", fontsize=10, color=color)
plt.axis("off")
plt.show()

print("Істинна мітка:", true_label)
print("Передбачення моделі:", pred_label)
print(f"Ймовірність: {pred:.4f}")
print("Модель вгадала!" if correct else "Модель помилилася.")
```

Рис 1.7



Рис 1.8

5. З тестової вибірки вивести зображення (16 шт) та вивести істину мітку, визначену мітку та прогноз для зображення.

```python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

model = tf.keras.models.load_model('/content/drive/MyDrive/cats_dogs.keras')

loss, acc = model.evaluate(test_ds, verbose=0)
print(f"📊 Точність моделі на тестовій вибірці: {acc*100:.2f}%")

images, labels = next(iter(test_ds))

preds = model.predict(images, verbose=0).flatten()

plt.figure(figsize=(14, 10))
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(images[i])

    true_label = "Собака" if labels[i] == 1 else "Кішка"
    pred_label = "Собака" if preds[i] >= 0.5 else "Кішка"
    prob = preds[i]

    correct = (labels[i] == (preds[i] >= 0.5))
    color = "green" if correct else "red"

    plt.title(f"""
Істина: {true_label}
Прогноз: {pred_label}
Ймовірн.: {prob:.2f}
""", fontsize=9, color=color)
    plt.axis("off")

plt.suptitle("Результати передбачення для 16 зображень із test_ds", fontsize=14)
plt.tight_layout()
plt.show()
```

Рис 1.9



Рис 1.10

Висновок: У ході виконання роботи було створено та досліджено згорткову нейронну мережу з використанням бібліотеки **TensorFlow**. Ознайомлено з процесом підготовки даних, побудови, навчання та тестування моделі на датасеті *Cats vs Dogs*. Отримані результати показали, що мережа здатна розпізнавати зображення котів і собак із прийнятною точністю. Практична реалізація дозволила зрозуміти основні принципи роботи згорткових шарів, роль етапів попередньої обробки даних та методів оцінки якості моделі.

Контрольні запитання

1. Що таке згорткова нейронна мережа (Convolutional Neural Network)?

Згорткова нейронна мережа (CNN) - це спеціалізована архітектура глибоких нейронних мереж, призначена для ефективної обробки даних зі сітковою структурою, таких як зображення. Вона складається з послідовності шарів, що поєднують операції згортки, пулінгу та повного зв'язку для автоматичного виявлення ієрархії особливостей - від простих країв і текстур до складних об'єктів.

2. У чому основна відмінність CNN від звичайних повнозв'язних мереж?

Основна відмінність полягає в принципі локальних зв'язків та спільних ваг. У повнозв'язних мережах кожен нейрон з'єднаний з усіма нейронами попереднього шару, що призводить до величезної кількості параметрів. CNN використовує локальні рецептивні поля - кожен нейрон обробляє лише невелику область вхідних даних, а однакові ваги застосовуються по всьому зображенню, що значно зменшує кількість параметрів та покращує ефективність.

3. Що виконує операція згортки (convolution)?

Операція згортки застосовує спеціальні фільтри (ядра) до вхідного зображення, виконуючи поелементне множення та підсумовування між фільтром і кожною локальною областю вхідних даних. Ця операція дозволяє виявляти просторові залежності, локальні особливості та шаблони в зображенні, такі як краї, кути, текстури та інші характерні ознаки.

4. Яку роль відіграють ядра (filters/kernels) у CNN?

Ядра відіграють роль детекторів особливостей у згорткових мережах. Кожен фільтр навчається розпізнавати специфічні типи просторових шаблонів у вхідних даних. Різні фільтри спеціалізуються на виявленні різних ознак - від простих контурів до складних текстур. Багатошаровість CNN дозволяє створювати ієрархію ознак, де нижчі шари виявляють прості елементи, а вищі - комбінують їх у складніші структури.

5. Що таке stride і як він впливає на розмір вихідного зображення?

Stride - це параметр, що визначає крок переміщення фільтра по вхідному зображенню під час операції згортки. Він безпосередньо впливає на розмір вихідного зображення: більший stride швидше зменшує просторові розмірності, але може призводити до втрати деталей. Математично розмір виходу обчислюється за формулою: (вхідний_розмір - розмір_фільтра) / stride + 1. Менший stride зберігає більше просторової інформації, але вимагає більших обчислювальних ресурсів.