МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

РОЗРОБКА КІБЕРФІЗИЧНИХ СИСТЕМ

Методичні вказівки
до виконання практичних занять
для здобувачів другого (магістерського рівня)
вищої освіти спеціальності 174 «Автоматизація,
комп`ютерно-інтегровані технології та робототехніка»

Київ 2025

УДК 31.264
Р64

Укладач В. Ю. Луценко, канд. техн. наук, доцент

Рецензент С. В. Іносов канд. техн. наук., доцент

Відповідальний за випуск А. В. Запривода, канд. техн. наук, доцент

Затверджено на засіданні кафедри автоматизації технологічних процесів, протокол № 7 від 25 лютого 2025 року.

В авторській редакції.

Розробка кіберфізичних систем [Електронний ресурс] :
Р64 методичні вказівки до виконання практичних занять/ уклад.: В. Ю. Луценко. – Київ : КНУБА, 2025. – 44 с.

Розглянуто питання розробки та проєктування апаратного, алгоритмічного, методичного та програмного забезпечення кіберфізичних систем в будівельній галузі. Окрему увагу приділено застосуванню методів штучного інтелекту, зокрема машинного навчання, для вирішення задач прогнозування та класифікації.

Призначено для здобувачів другого (магістерського рівня) вищої освіти спеціальності 174 «Автоматизація, комп`ютерно-інтегровані технології та робототехніка».

ЗМІСТ

Загальні положення

Сучасний етап розвитку технологій характеризується інтенсивним впровадженням кібер-фізичних систем (КФС) у різні галузі промисловості, зокрема в будівництві. Дисципліна "Розробка кібер-фізичних систем" відіграє ключову роль у підготовці фахівців з автоматизації, які зможуть проєктувати, впроваджувати та ефективно використовувати інтелектуальні технології для оптимізації будівельних процесів.

Курс базується на знаннях з «Програмно-технічних комплексів», «Технічних засобів автоматизації», «Об’єктно-орієнтованого програмування», «Технології будівельного виробництва» та тісно пов’язаний із дисциплінами «Штучний інтелект у будівництві» та «Метрологія, технологічні вимірювання та проилади». Він доповнює професійну підготовку студентів, надаючи їм практичні навички роботи з сучасними інструментами автоматизації.

Практичні заняття спрямовані на формування у студентів навичок роботи з сучасними системами автоматизації.

Метою практичних занять є формування в студентів компетенцій, необхідних для розробки та впровадження кібер-фізичних систем у будівельній галузі. Основними завданнями є:

- вивчення принципів побудови КФС та їх компонентів.
- опанування інструментів розробки програмно-апаратних рішень (Arduino, Raspberry Pi, PLC, SCADA).
- розвиток навичок проєктування автоматизованих систем для моніторингу та управління будівельними процесами.
- розвиток навичок розробки та впровадження елементів штучного інтелекту у реальних будівельних проєктах.

Практичні заняття спрямовані на закріплення теоретичного матеріалу, що розглядається на лекційних заняттях та під час самостійної роботи. Під час навчання студенти отримують досвід, необхідний для подальшої роботи в галузі індустрії 4.0, де автоматизація та цифровізація стають основою ефективного будівництва.

Практична робота №1. Інтерактивне середовище розробки програмного забезпечення Google colaboratory

Також ця мова часто використовується для створення систем штучного інтелекту, що пояснюється в першу чергу наявністю необхідних бібліотек, простотою та лаконічністю.

Google colaboratory

Блокнот Colab – це безкоштовне інтерактивне хмарне середовище для роботи з кодом від Google. Принцип у нього такий самий, як і у решти онлайн-офісів компанії: воно дозволяє одночасно з колегами працювати з даними.

В основі «Google Colab» є блокнот Jupyter для роботи на Python, тільки з базою на Google Диску, а не на комп'ютері.

Головна особливість «Колабораторії» – безкоштовні потужні графічні процесори GPU та TPU, завдяки яким можна займатися не лише базовою аналітикою даних, а й складнішими дослідженнями у галузі машинного навчання.

CPU – центральний процесор – мозок комп'ютера, який виконує операції з даними.

GPU – графічний процесор. Обробляє дані швидше, оскільки завдання виконує паралельно, а не послідовно, як робить CPU. GPU розроблений для роботи виключно з графікою, тому на ньому зручніше працювати із зображенням та відео, наприклад займатися 3D-моделюванням або монтажем.

TPU – тензорний процесор, розробка Google. Він призначений для тренування нейромереж. У цього процесора в рази вища продуктивність при великих обсягах обчислювальних завдань.

Самі процесори дорогі, і не кожен може їх собі дозволити. Google Colaboratory дає можливість безкоштовно та безперервно користуватися ними протягом 12 годин. Будьте уважні: як тільки цей час закінчиться, Colab зітре всі дані і доведеться починати спочатку.

Крім того, Google відключає блокноти після приблизно 30 хвилин бездіяльності, щоб не перевантажувати процесори.

Для встановлення Google лабораторії необхідно натиснути кнопку «Створити» ліворуч на Диску Google (рис. 1.1).

Потім необхідно перейти в меню до пункту Ще «- -» і натиснути «+» Підключити інші програми.

Потім у рядку пошуку необхідно ввести «Colaboratory » та натиснути Enter.

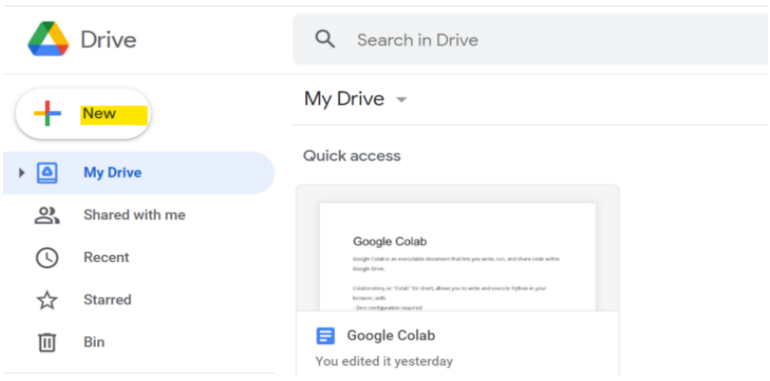


Рис.1.1. Діалог створення документу Google Colab

Створений документ розміщено не на статичній веб-сторінці, а в інтерактивному середовищі під назвою блокнот Colab, що дозволяє писати та виконувати код.

Наприклад, ось комірка з коротким скриптом Python, який дозволяє розрахувати значення, записати його у змінну та роздрукувати результат:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

    86400
```

Для виконання коду послідовно у всіх комірках можна скористатися поєднанням клавіш ctrl+F9.

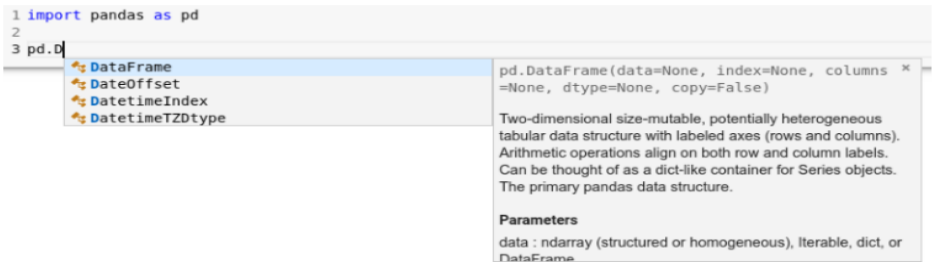
Щоб зберегти файл з новим ім'ям на жорсткий диск необхідно в меню вибрати File->download .ipynb.

Щоб додати новий осередок, виберіть Insert->Code cell.

Доповнення коду та підказки з документації відбуваються автоматично під час введення. Для цього використовуйте такі клавіші:

Ctrl-пробіл, щоб заново відкрити завершення коду.

Ctrl-shift-пробіл, щоб знову відкрити підказки параметрів.



Практична робота №2. Застосування методів машинного навчання для вирішення задач прогнозування (регресія)

Машинне навчання – це навчання, що засноване на досвіді. Наприклад, це схоже на людину, яка вчиться грати у шахи через спостереження, як грають інші. Таким чином, комп'ютери можуть бути запрограмовані шляхом надання інформації, яку вони вивчають, набуваючи здатності ідентифікувати елементи або характеристики з високою ймовірністю.

У процесі машинного навчання можна виділити наступні етапи:

- збір інформації;
- сортування даних;
- аналіз даних;
- розробка алгоритм перевірки;
- використання алгоритму для подальших досліджень.

Розглянемо приклад.

Є дві числові послідовності. Необхідно визначити наступний член однієї з них.

0; 8; 15;22; 38
32; 46.4; 59; 71.6; ?

Скоріш за все друга послідовність описується формулою, що переводить значення температури в градусах Цельсія в градуси шкали Фаренгейта:

$$F=1.8*C+32$$

Перевірте, що це так.

Розв’язок цієї задачі за допомогою машинного навчання основується на застосуванні нейронних мереж, які потрібно попередньо навчити. Цей процес називають тренуванням.

Існує безліч типів архітектур нейронних мереж. Проте, незалежно від того, яка архітектура обрана, алгоритми моделі (які обчислення

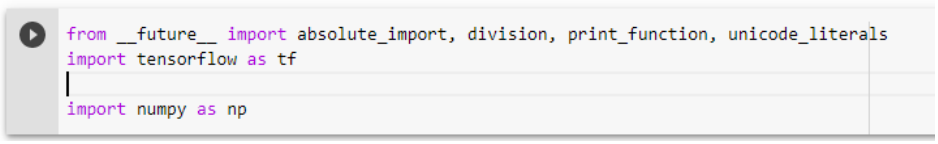
виконуються й у якому порядку) залишаться незмінними у процесі тренування. Замість їх зміни, змінюються внутрішні змінні (вагові коефіцієнти та зміщення) під час тренування.

Наприклад, у задачі конвертації з градусів Цельсія в шкалу Фаренгейта, модель починає з множення вхідного значення на деяке число (ваговий коефіцієнт) та додавання іншого значення (зміщення). Навчання моделі полягає в знаходженні відповідних значень для цих змінних, без зміни операцій множення та додавання.

Розробіть нейронну мережу, що буде вирішувати задачі прогнозу. У нашому випадку ця мережа буде передбачати наступне значення температури за шкалою Фаренгейта. Для цього необхідно виконати наступні кроки.

1. Імпорт залежностей. Імпортуйте TensorFlow. Тут і надалі скорочено будемо називати його tf. Налаштуйте рівень логування – лише помилки.

2.Імпортуйте NumPy як np. Numpy допомагає представляти наші дані у вигляді високоефективних списків.



```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf

import numpy as np
```

3. Підготовка даних для тренування.

Методика машинного навчання з вчителем ґрунтується на пошуку алгоритму перетворення вхідних даних у вихідні.

Створіть два списки – celsius_q і fahrenheit_a, які будемо використовувати під час навчання нашої моделі.

```
import tensorflow as tf

import numpy as np

celsius_q    = np.array([-40, -10, 0, 8, 15, 22, 38], dtype=float)
fahrenheit_a = np.array([-40, 14, 32, 46, 59, 72, 100], dtype=float)

for i,c in enumerate(celsius_q):
  print("{} градусів Цельсія = {} градусів Фаренгейта".format(c, fahrenheit_a[i]))
```

4. Створіть модель нейромережі. Будемо використовувати максимально спрощену модель - модель повнозв'язної мережі (Dense-мережа). Оскільки завдання досить тривіальне, то й мережа скрадатиметься з єдиного шару з єдиним нейроном. Давайте назвемо цей

шар l0 (layer і нуль). Створення та ініціалізація цього шару виконується завдяки методу

```
l0 = tf.keras.layers.Dense(units=1, input_shape=[1])
```

з наступними параметрами:

• input_shape=[1] – цей параметр визначає розмір вхідної величини – одиничне значення. Матриця розміром 1×1 із єдиним значенням. Так як це перший (і єдиний) шар, то розмірність вхідних даних відповідає розмірності всієї моделі. Єдине значення – значення з плаваючою комою, що становить градуси Цельсія.

• units=1 – цей параметр визначає кількість нейронів у шарі. Кількість нейронів визначає те, як багато внутрішніх змінних шару буде використано для навчання під час пошуку рішення поставленого завдання. Так як це останній шар, то його розмірність дорівнює розмірності результату - вихідного значення моделі – число з плаваючою комою, що представляє собою градуси Фаренгейта (у багатошаровій мережі розміри та форма шару input_shape повинні відповідати розмірам та формам наступного шару).

Як тільки шари визначені їх необхідно перетворити на модель. Sequential-модель приймає в якості аргументів перелік шарів у тому порядку в якому їх необхідно застосовувати - від вхідного значення до вихідного значення.

У нашій моделі лише один шар - l0.

model = tf.keras.Sequential([l0])

5. Скомпілюйте модель з функцією втрат та оптимізації.

Перед тренуванням модель має бути скомпільована (зібрана).

```
model.compile(loss='mean_squared_error',
              optimizer=tf.keras.optimizers.Adam(0.1))
```

При компіляції для тренування потрібні:

• функція втрат — спосіб вимірювання того, наскільки відрізняється очікуване значення від вихідного значення (вимірна різниця називається «втратою»).

• Оптимізація — спосіб коригування внутрішніх змінних для зменшення втрат.

6. Проведіть тренування (навчання) нейромережі.

Функція втрат і оптимізації використовуються під час тренування моделі для виконання первинних обчислень у кожній точці і подальшої оптимізації значень.

Під час тренування функція оптимізації використовується для коригування значень внутрішніх змінних. Мета - підігнати значення внутрішніх змінних таким чином у моделі (а це, по суті, математична функція), щоб ті відображали максимально наближений вираз конвертації градусів Цельсія в градуси Фаренгейта.

Функція втрат (середньоквадратична помилка) та функція оптимізації (Adam), що використовуються в цьому прикладі, є стандартними для подібних простих моделей, але крім них є і інші.

Зверніть увагу, на функцію оптимізації та її параметр – коефіцієнт швидкості навчання (learning rate), який у нашому прикладі дорівнює 0.1. Це величина кроку при коригуванні внутрішніх значень змінних. Якщо значення занадто мале. Це означає, що знадобиться занадто багато ітерацій для навчання моделі. Занадто велике значення буде супроводжуватися падінням точності. Знаходження "гарного" значення коефіцієнта швидкості навчання вимагає певних спроб і помилок, воно зазвичай знаходиться в інтервалі від 0.01 (за замовчуванням) до 0.1.

Тренування моделі здійснюється методом «fit». Під час тренування модель отримує на вхід значення градусів Цельсія, виконує перетворення, використовуючи значення внутрішніх змінних (які називаються «вагами») і повертає значення, які повинні відповідати градусам Фаренгейтом. Так як початкові значення вагових коефіцієнтів встановлені довільними, то і результуючі значення першої ітерації будуть далекі від коректних значень. Різниця між необхідним і фактичним результатами обчислюється з використанням функції втрат, а функція оптимізації визначає, яким чином повинні бути підкориговані вагові коефіцієнти.

Цей цикл обчислень, порівнянь та коригування контролюється всередині методу fit.

```
history = model.fit(celsius_q, fahrenheit_a, epochs=500, verbose=False)
```

Перший аргумент методу fit – вхідні значення, другий аргумент – бажані вихідні значення.

Аргумент epochs визначає скільки разів цей навчальний цикл повинен бути виконаний.

Аргумент verbose контролює рівень логування.

7. Відобразіть статистику тренувань

Метод fit повертає об'єкт, який містить інформацію про зміну втрат із кожною наступною ітерацією. Можемо скористатися цим об'єктом для побудови відповідного графіка втрат. Висока втрата означає, що значення градусів Фаренгейта, які пророкувала модель, далекі від справжніх значень у масиві fahrenheit_a.

Для візуалізації скористаємося модулем Matplotlib.

```
import matplotlib.pyplot as plt
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.plot(history.history['loss'])
```

Наша модель покращується дуже швидко на самому початку, а потім приходить до стабільного та повільного покращення до тих пір, поки результати не стають близькими до ідеальних в самому кінці навчання.

Результуюча статистика тренувань має наступний вигляд (рис.1.2):

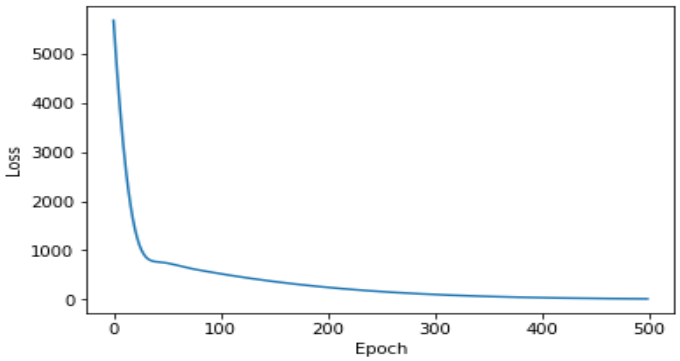


Рис.1.2. Залежність функції втрат від кількості епох тренувань

8. Використайте розроблену нейромережу для передбачення нових результатів.

Тепер у нас є модель, яка була навчена на вхідних значеннях celsius_q та вихідних значеннях fahrenheit_a для визначення взаємозв'язку між ними. Ми можемо скористатися методом передбачення для обчислення нових значень градусів Фаренгейта, що відповідають заданим градусам Цельсія.

Наприклад, скільки буде 100.0 градусів Цельсія за Фаренгейтом?

```
print(model.predict([100.0]))
```

```
[[211.33124]]
```

Перевірте отриманий результат із результатом обрахованим за формулою перерахунку температури в градусах Цельсія в шкалу Фаренгейта.

Практична робота №3. Розробка нейромережі для вирішення задач класифікації

1. Встановіть набір даних datasets tensorflow:

```
!pip install -U tensorflow_datasets
```

2. Імпортуйте «майбутні» методи:

```
from __future__ import absolute_import, division, print_function, unicode_literals
```

3. Імпортуйте TensorFlow та набір даних TensorFlow:

```
import tensorflow as tf
import logging
import tensorflow_datasets as tfds
#tf.logging.set_verbosity(tf.get_logging.ERROR)
```

4. Підключіть додаткові бібліотеки:

```
import math
import numpy as np
import matplotlib.pyplot as plt
```

5. Підключіть додаткові бібліотеки для відображення прогресбару:

```
import tqdm
import tqdm.auto
tqdm.tqdm=tqdm.auto.tqdm
print(tf.__version__)
#tf.enable_eager_execution()
```

6. Імпортуйте набір даних Fashion MNIST:

```
dataset,metadata=tfds.load('fashion_mnist', as_supervised=True,with_info=True)
```

Параметр as_supervised=True забезпечує отримання із зображенням мітки, що відповідає назві зображення, наприклад сорочці відповідає мітка – 7;

with_info=True забезпечує отримання метаданних-додаткову інформацію про dataset)

7. Сформуйте тренувальний набір даних та набір даних для тестування:

```
train_dataset, test_dataset=dataset['train'], dataset['test']
```

Зображення представляє собою двомірний масив 28х28 пікселя. Значення, що можуть бути записані в кожну комірку знаходяться в інтервалі [0,255].

Мітки – це масив цілих чисел в інтервалі [0,9](0-футболка; 1-шорти; 2-светр; 3–сукня; 4- плащ; 5-сандалі; 6-сорочка; 7–кросівки; 8-сумка; 9-ботинки).

8. Створимо змінну для збереження назв міток:

```
class_names=['футболка','шорти','светр', 'сукня','плащ','сандалі','сорочка','кросівки','сумка','ботинки']
```

9. Дослідить формат та структуру даних, що представлені в навчальному наборі перед тренуванням моделі:

```
num_train_examples=metadata.splits['train'].num_examples
num_test_examples=metadata.splits['test'].num_examples
print('Кількість тренувальних екземплярів:{}'.format(num_train_examples))
print('Кількість тестувальних екземплярів:{}'.format(num_test_examples))
```

10.Виконайте добробку даних.

Значення кожного пікселя зображення знаходиться в інтервалі[0,255]. Для того, щоб модель працювала коректно ці значення необхідно нормалізувати, тобто привести до значень, що будуть належати інтервалу [0,1]. Розробимо функцію нормалізації та застосуємо її до даних дата сету.

```
def normalize (images,labels):
  images=tf.cast(images, tf.float32)
  images/=255
  return images, labels
```

Сторено метод normalize, якому передається зображення та мітка. За допомогою метода cast бібліотеки TensorFlow цілочисленне значення пікселя перетворюється в змінну типу float 32, що забезпечує збереження дробної частини. Нормування здійснюється в результаті ділення значення пікселю на 256.

За допомогою метода map() функція нормування застосовується для кожного елементу масивів train_dataset та test_dataset:

```
train_dataset=train_dataset.map(normalize)
test_dataset=test_dataset.map(normalize)
```

11. Дослідіть оброблені дані. Виведіть зображення одного з елементів:

```
for image, label in test_dataset.take(1):
  break;
image=image.numpy().reshape((28,28))
plt.figure()
plt.imshow(image, cmap=plt.cm.binary)
plt.colorbar()
plt.grid (False)
plt.show()
```

На рис. 1.3 представлено один з елементів тестового dataset.

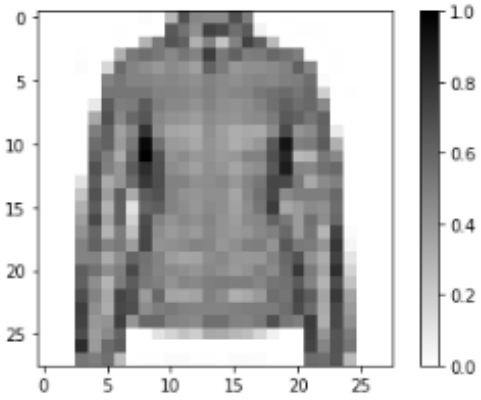


Рис.1.3. Елемент тестового dataset

Зробіть висновки про відповідність отриманого результату постановці задачі.

12. Виведіть перші 25 зображень із мітками з тренувального набору (результат такого виводу має вигляд представлений на рис.1.4). Під кожним зображенням вкажіть до якого класу одягу відноситься зображення:

```
plt.figure(figsize=(10,10))
i=0
for image, label in test_dataset.take(25):
  image=image.numpy().reshape((28,28))
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid (False)
```

```
  plt.imshow(image, cmap=plt.cm.binary)
  plt.xlabel(class_names[label])
  i+=1
plt.show()
```

Рис.1.4. Зображення тренувального набору

Зробіть висновки про відповідність отриманого результату постановці задачі.

13. Розробіть модель нейронної мережі. Мережа містить три нейронних шари:

```
model=tf.keras.Sequential([
   tf.keras.layers.Flatten(input_shape=(28,28,1)),
   tf.keras.layers.Dense(128, activation=tf.nn.relu),
   tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Мережа складається з трьох шарів:

• вхідного tf.keras.layers.Flatten – цей шар перетворює зображення розміром 28x28 пікселів у 1D масив розміром 784 (28*28). На цьому шарі ми не маємо жодних параметрів для навчання, тому що цей шар займається тільки перетворенням вхідних даних.

• tf.keras.layers.Dense – щільно зв’язаний шар з 128 нейронів. Кожен нейрон (вузол) приймає на вхід усі 784 значення з попереднього шару,

змінює вхідні значення згідно з внутрішньою вагою та зсувом під час тренування та повертає єдине значення на наступний шар.

•вихідний шар tf.keras.layers.Dense softmax-шар складається з 10 нейронів, кожен з яких представляє певний клас елементу одягу. Як і попередньому шарі, кожен нейрон приймає на вхід значення всіх 128 нейронів попереднього шару. Вага та зміщення кожного нейрона на цьому шарі змінюються при навчанні таким чином, щоб результуюче значення було в інтервалі [0,1) і представляло собою ймовірність того, що зображення відноситься до цього класу. Сума всіх вихідних значень 10 нейронів дорівнює 1.

Опишіть та дайте визначення для функцій активації «relu» та «softmax».

14. Скомпілюйте модель. Перед тим, як ми приступимо до тренування моделі, варто ще виконати кілька налаштувань.

• функція втрат – алгоритм вимірювання того, наскільки далеко знаходиться бажане значення від спрогнозованого.

• функція оптимізації агоритм 'підгонки' внутрішніх параметрів (ваги та зсуву) моделі для мінімізації функції втрат.

• метрики – використовуються для моніторингу процесу тренування та тестування. Використовує таку метрики як точність та відсоток зображень, які були коректно класифіковані.

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

15.Проведіть тренування моделі. Визначимо послідовність дій під час навчання на тренувальному наборі даних:

1. Повторюємо нескінченну кількість разів набір вхідних даних використовуючи метод dataset.repeat(). Параметр epochs, який буде описано нижче, визначає кількість всіх навчальних ітерацій для виконання.

2. Метод dataset.shuffle(60000) перемішує всі зображення для того, щоб на навчання нашої моделі не впливав порядок подачі вхідних даних.

3. Метод dataset.batch(32) повідомляє методу тренування model.fit використовувати блоки по 32 зображення та мітки при оновленні внутрішніх змінних моделі.

```
BATCH_SIZE=32
train_dataset=train_dataset.repeat().shuffle(num_train_examples).batch(
BATCH_SIZE)
```

test_dataset=test_dataset.batch(BATCH_SIZE)

Тренування відбувається за допомогою виклику методу model.fit:

1. Надсилає train_dataset на вхід моделі.

2. Модель вчиться зіставляти вхідне зображення з міткою

3. Параметр epochs=5 обмежує кількість тренувань до 5 повних навчальних ітерацій з набору даних, що у результаті дає нам тренування на 5*60000 > 300 000 прикладах

model.fit(train_dataset, epochs=5,steps_per_epoch=math.ceil(num_train_examples/BATCH_SIZE))

```
[16] model.fit(train_dataset, epochs=5,steps_per_epoch=math.ceil(num_train_examples/BATCH_SIZE))

Epoch 1/5
1875/1875 [==============================] - 11s 3ms/step - loss: 0.4963 - accuracy: 0.8251
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3759 - accuracy: 0.8640
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3393 - accuracy: 0.8748
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3169 - accuracy: 0.8830
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2901 - accuracy: 0.8931
<keras.callbacks.History at 0x7f288b043990>
```

Зробіть висновки як змінюється функція втрат та точність в процесі тренування моделі.

15. Визначте точність, яку забезпечує модель на тестових даних. Використайте для цього усі приклади, що є в тестових наборах даних

test_loss,test_accurancy=model.evaluate(test_dataset, steps=math.ceil(num_test_examples/BATCH_SIZE))

print('Точність на тестовому наборі даних', test_accurancy)

```
313/313 [==============================] - 1s 4ms/step - loss: 0.3487 - accuracy: 0.8751
Точність на тестовому наборі даних 0.8751000165939331
```

16. Використайте отримані результати для визначення мітки зображення. Використайте тестовий набір даних та метод predict:

```
for test_images, test_labels in test_dataset.take(1):
  test_images=test_images.numpy()
  test_labels=test_labels.numpy()
  predictions=model.predict(test_images)
predictions.shape
```

Метод predictions.shape повертає массив з 32 зображень, кожному з яких відповідає массив вірогідностей міток, що містить 10 значень.

17. Виведіть масив вірогідностей міток для «0-го» зображення:

predictions [0]

18. Виведіть індекс максимального елемента масиву вірогідностей:

np.argmax(predictions[0])

Зробіть висновок відносно відповідності передбачення та реального значення мітки. Для цього виведіть значення мітки зображення. Використайте для цього наступний код:

test_labels[0].

Практична робота №4. Застосування нейромережі для вирішення задач класифікації

Скористайтесь результатами розробки нейромережі, що були отримані на попередньому практичному занятті.

1.Відобразіть всі вихідні зображення та відповідні передбачення моделі за 10-ма класами. Для цього розробіть дві методи: def plot_image та def plot_value_array:

```
def plot_image(i, predictions_array, true_labels, images):
  predictions_array, true_label, img=predictions_array[i], true_labels[i], i
mages[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(img[...,0], cmap=plt.cm.binary)
  predicted_label=np.argmax(predictions_array)
  if predicted_label == true_label:
   color='blue'
  else:
   color='red'
  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                 100*np.max(predictions_array),
                                 class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_labels):
  predictions_array, true_label=predictions_array[i], true_labels[i]
  plt.grid(False)
```

```
  plt.xticks([])
  plt.yticks([])
  thisplot=plt.bar(range(10), predictions_array,color="#777777")
  plt.ylim([0,1])
  predicted_label=np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

2. Виведіть «0» зображення, результат передбачення моделі та масив передбачень (рис.1.5):

```
i=0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
```

Що означає синій колір підпису та діаграми?

3. Розглянемо ще один приклад:

```
i=3
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
```

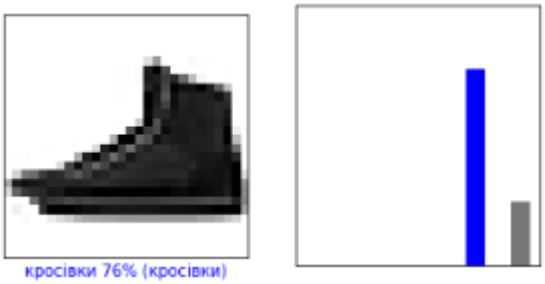


Рис.1.5.Результат передбачення моделі та масив передбачень

Зверніть увагу на неоднозначність ситуації: є досить висока вірогідність що це не кросівки.

4. Виведіть декілька зображень та вірогідності їх передбачення. Коректні результати виведіть, наприклад, синім кольором, а помилкові – червоним. Значення під зображенням відображає відсоток впевненості моделі в тому,що вхідне зображення відповідає цьому класу. Приклад такого виводу представлено на рис.1.6. Поясніть випадок, коли результат може бути помилковим, навіть коли значення «впевненості» досить велике.

```
num_rows=8
num_cols=4
num_images=num_rows*num_cols
plt.figure(figsize=(2*2*num_cols,2*num_rows))
for i in range (num_images):
  plt.subplot(num_rows,2*num_cols,2*i+1)
  plot_image(i, predictions, test_labels, test_images)
  plt.subplot(num_rows,2*num_cols,2*i+2)
  plot_value_array(i, predictions, test_labels)
```

Рис.1.6. Результати передбачень нейромережі

Зробіть висновки відносно отриманих результатів.

Який результат виконання коду:

```
plt.figure(figsize=(2*2*num_cols,2*num_rows))
```

5. Дослідіть окреме зображення (наприклад, зображення з індексом «0»):

```
img=test_images[0]
print(img.shape)
```

Що означають виведені дані?

6. Перетворіть параметр моделі в список. Моделі в keras оптимізовані для передбачень блоками(колекціями). Тому навіть, коли ми використовуємо єдиний елемент, його необхідно додати до списку.

```
img=np.array([img])
print(img.shape)
```

7. Отримайте результат передбачення для цього зображення:

```
predictions_singl=model.predict(img)
print(predictions_singl)
```

Поясніть отриманий результат.

8. Виведіть розподіл вірогідностей результатів (рис.1.7):

```
plot_value_array(0,predictions_singl,test_labels)
_=plt.xticks(range(10),class_names,rotation=45)
```

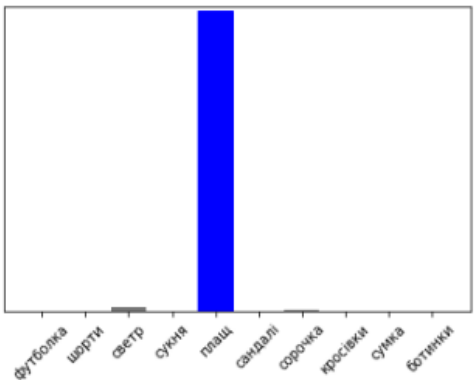


Рис.1.7. Розподіл вірогідностей результатів

9. Виведіть максимальне значення вірогідностей результатів:

```
np. argmax(predictions_singl[0])
```

Поясніть отриманий результат.

10. Задайте значення параметру epochs=1 та дослідіть роботу нейромережі. Зробіть необхідні висновки відносно точності класифікації зображень. Порівняйте отримане значення з попереднім.

11. Задайте значення параметру epochs=20 та дослідіть роботу нейромережі. Зробіть необхідні висновки відносно точності класифікації зображень. Порівняйте отримане значення з попереднім.

12. Визначте кількість ітерацій, за якої модель «запам’ятає» навчальну вибірку. Яке значення в цьому випадку буде мати функція втрат? Чому дорівнює точність на тестовому наборі даних?

13. Змініть кількість нейронів в прихованому (другому)шарі (метод Sequential)наприклад від 10 до 512. Встановіть як буде змінюватися точність прогнозу моделі. Зробіть необхідні висновки відносно точності

класифікації зображень на тестовому наборі даних. Порівняйте отримане значення з попереднім.

14. Додайте додатковий шар нейронів між flatten шаром(шар, що згладжує) та кінцевим dense шаром. Дослідіть як впливає кількість нейронів в цьому шарі на точності класифікації зображень на тестовому наборі даних. Зробіть необхідні висновки.

15. Не проводьте нормалізацію значень пікселів. Який результат в цьому випадку буде отримано?

Підказка: Для цього необхідно закоментувати два останніх рядка:

train_dataset=train_dataset.map(normalize)

test_dataset=test_dataset.map(normalize)

За що відповідає ця ділянка коду? Чи зміняться при цьому зображення елементів? Як зміниться значення функції втрат та точність класифікації зображень на тестовому наборі даних? Яку роль відіграє функція активації «relu» та «softmax».

Практична робота №5. Розробка та застосування згорткової нейромережі

Згортка (анг. Convolution) зображення – матриця, зазвичай, малих розмірів, що використовується в обробці зображень як фільтр для розмиття, підвищення різкості, виділення границь тощо. Обробка зображення полягає у обчисленні нового значення обраного пікселя з врахуванням значення оточуючих його пікселів.

В залежності від елементів матриці згортка може викликати різні ефекти.

Згортка - це процес додавання кожного елемента зображення до його сусідів, зважених ядром. Важливо зауважити, що виконувана матрична операція - згортка - це не звичайне множення, хоча й позначається *.

Наприклад, якщо ми маємо дві 3x3 матриці, перша - ядро, друга - шматок зображення, згортка - це процес транспонування рядків і стовпчиків ядра з наступним множенням і додаванням. Елемент з координатами [2, 2] (тобто, центральний елемент) отриманого зображення буде зваженою комбінацією всіх елементів матриці зображення, з вагами взятими з ядра:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

Згортка дозволяє додати структурованості даних в самій мережі, тим самим реалізуючи інваріантність. Інваріант – (лат. *invariant* — незмінюваний) або інваріантність — термін, що позначає щось незмінне.

Мета згортки зображення – виділити ряд ознак, що будуть ефективно використані наприклад під час вирішення задачі класифікації.

Наприклад, за допомогою згортки на зображенні виділяється комірець в якості ознаки, яка потім застосовується для класифікації інших зображень.

Оперція підвибірки POOLING.

Оперція підвибірки pooling забезпечує зменшення розмірів зображення.

Види pooling: sum-, avg-, max-pooling.

Приклад дії на зображення операції max-pooling представлено на рис. 1.8.

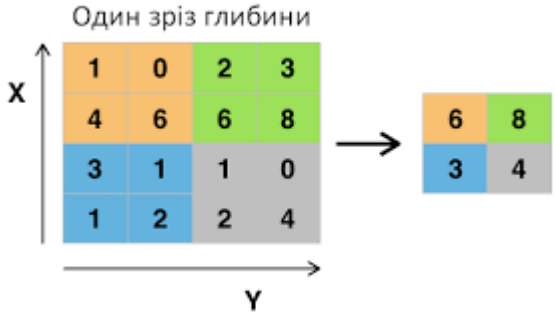


Рис. 1.8. Результат виконання операції max-pooling

1. Використайте результати розробки нейромережі отримані на попередніх практичних заняттях. Повнозвязна мережа забезпечувала точність класифікації елементів одягу на рівні:

```
model.fit(train_dataset, epochs=5, steps_per_epoch=math.ceil(num_train_examples/BATCH_SIZE))

Epoch 1/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.4888 - acc: 0.8279
Epoch 2/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.3727 - acc: 0.8651
Epoch 3/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.3334 - acc: 0.8781
Epoch 4/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.3093 - acc: 0.8878
Epoch 5/5
1875/1875 [==============================] - 15s 8ms/step - loss: 0.2936 - acc: 0.8914
<tensorflow.python.keras.callbacks.History at 0x7fe0dbdf6b38>
```

Точність на тестовому наборі даних склала:

```
[76] test_loss, test_accuracy = model.evaluate(test_dataset, steps=math.ceil(num_test_examples/BATCH_SIZE))
     print('Точность на тестовом наборе данных: ', test_accuracy)

313/313 [==============================] - 2s 7ms/step - loss: 0.3660 - acc: 0.8693
Точность на тестовом наборе данных:  0.8693
```

2. Змініть тестовий набір даних. Для цього розробіть окремий метод:

```
def  mirror(images, labels):
        return tf.image.flip_up_down(images), labels
```

Виклик метода забезпечує рядок коду:

```
test_dataset=test_dataset.map(normalize). map(mirror)
```

Визначте точність розпізнавання на змінуному наборі даних та зробіть висновки.

3. Розробіть структуру загорткової нейромережі:

1-ий шар – це шар згортки. 32 фільтри. Ядро 3x3; (padding-заповнення нулями позицій, що вишли за межі зображення)

2-ий шар виконує операцію max-pooling. Ядро 2x2, крок 2.

3-ий шар – це шар згортки. 64 фільтри. Ядро 3x3; (padding-заповнення нулями позицій, що вишли за межі зображення).

4-ий шар виконує операцію max-pooling. Ядро 2x2, крок 2.

```
model=tf.keras.Sequential([
   tf.keras.layers.Conv2D(32,(3,3),padding='same',activation=tf.nn.relu,
                          input_shape=(28,28,1)),
   tf.keras.layers.MaxPooling2D((2,2),strides=2),
   tf.keras.layers.Conv2D(64,(3,3),padding='same',activation=tf.nn.relu),
   tf.keras.layers.MaxPooling2D((2,2),strides=2),
   tf.keras.layers.Flatten(input_shape=(28,28,1)),
   tf.keras.layers.Dense(128, activation=tf.nn.relu),
   #tf.keras.layers.Dense(256, activation=tf.nn.relu),
   tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

4. Дайте відповідь на наступні питання:

1. Яку точність забезпечує згорткова нейромережа на навчальному наборі даних?

2. Яку точність забезпечує згорткова нейромережа на тестовому наборі даних?

3. Віддзеркальте зображення тестового набору даних та визначте точність класифікації предметів одягу. Зробіть висновки.

Практична робота № 6. Застосування згорткової нейромережі для обробки кольорових зображень

Реальні зображення характеризуються різними розмірами та наявністю кольору. Такі зображення потребують попередньої обробки таким чином, щоб на вхід згорткової нейромережі поступали дані з однаковою структурою.

Під час обробки зображень в градаціях сірого використовувався окремий вхідний шар Flatten, завданням якого було перетворення вхідних даних, що представляли собою 2D масив, в вектор. Створення цього шару відбувалося з урахуванням параметру input_shape=(28,28,1).

Під час роботи з зображеннями різного розміру використовується ресайзінг – зміна розміру. Тобто, всі зображення, з якими працює нейромережа приводяться до одного розміру, наприклад 150x150 пікселів.

Кольорове зображення складається з трьох кольорів: синього, червоного та зеленого (RGB)

Кольорове зображення можна уявити у вигляді шарів синіх, червоних та зелених пікселів. Результуюче зображення отримується в результаті додавання значень пікселів з цих шарів.

Таким чином, результуюче зображення представляє собою нашарування трьох двовимірних масивів, кожен з яких відповідає за свій кольоровий канал (рис. 1.9).

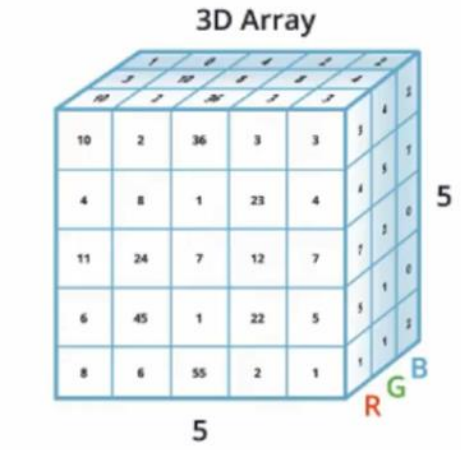


Рис. 1.9. Формування кольорового зображення

Фільтр згортки застосовується до кожного шару окремо (рис. 1.10). При цьому фільтри як правило обираються для кожного шару свій.

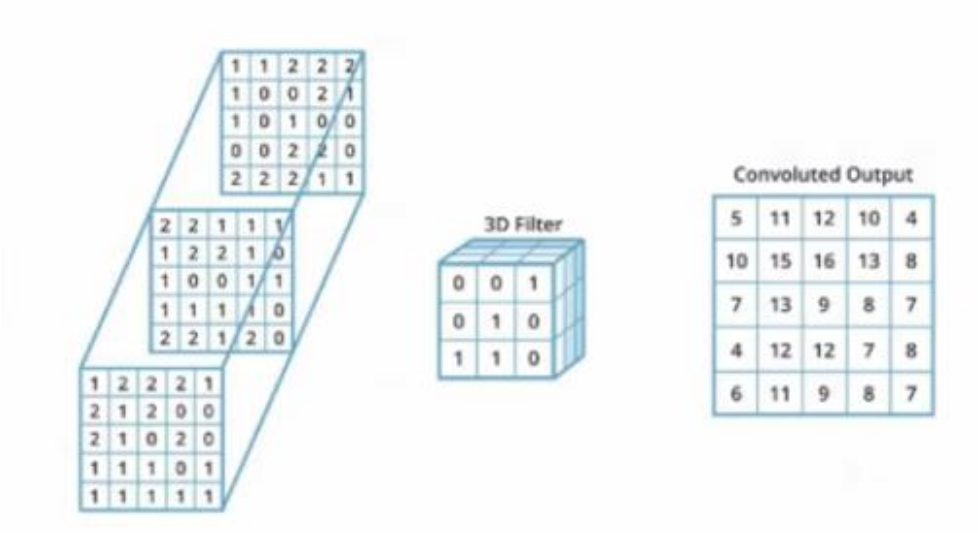


Рис. 1.10. Застосування фільтрів згортки для кольорового зображення

Кількість фільтрів визначає розмірність вихідного зображення. Наприклад, якщо до вхідного зображення застосовано 16 фільтрів, то вихідне зображення матиме 16 шарів зображень перетворених операцією згортки (рис. 1.11).

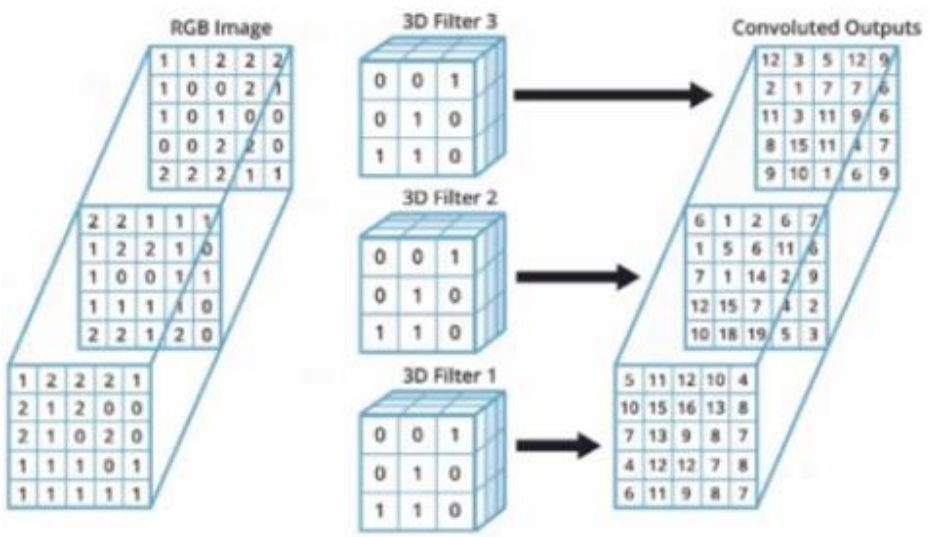


Рис. 1.11. Приклад застосування 16 фільтрів згортки для кольорового зображення

Наступним кроком обробки кольорового зображення є операція підвибірки (наприклад за максимальним значенням.). Ця операція застосовується в кожному кольоровому каналі (рис. 1.12).

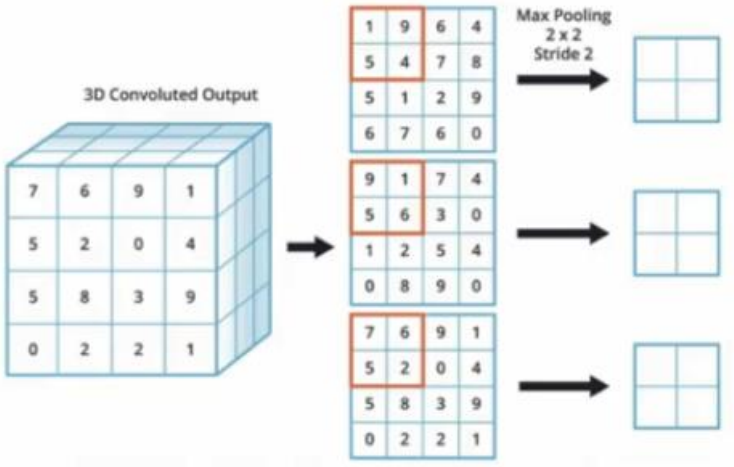


Рис. 1.12. Приклад застосування операції Pooling для кольорового зображення.

Практична частина

1.Імпортуйте необхідні пакети:

```
from __future__ import absolute_import, division, print_function, unicode_literals
import os
import numpy as np
import matplotlib.pyplot as plt
```

Пакет os необхідний для доступу до файлів та директорій, що знаходяться на диску.

Для чого використовується пакет numpy?

Для чого використовується пакет matplotlib?

2.Імпортуйте TensorFLOW:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

3. Завантажте архів з даними (datasets). Використайте для цього метод get_file

```
_URL='https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'
zip_dir=tf.keras.utils.get_file('cats_and_dogs_filtered.zip',origin=_URL, extract=True)
```

4.Створіть змінні, в яких будуть зберігатися адреси директорій з наборами даних для тренувань та валідації:

```
base_dir=os.path.join(os.path.dirname(zip_dir),'cats_and_dogs_filtered')
train_dir=os.path.join(base_dir,'train')
validation_dir=os.path.join(base_dir,'validation')
```

```
train_cats_dir=os.path.join(train_dir,'cats')
train_dogs_dir=os.path.join(train_dir,'dogs')
validation_cats_dir=os.path.join(validation_dir,'cats')
validation_dogs_dir=os.path.join(validation_dir,'dogs')
```

5.Визначте кількість зображень в тестовому та валідаційному наборах даних. Застосуйте для цього метод len

```
num_cats_tr=len(os.listdir(train_cats_dir))
num_dogs_tr=len(os.listdir(train_dogs_dir))
num_cats_val=len(os.listdir(validation_cats_dir))
num_dogs_val=len(os.listdir(validation_dogs_dir))

total_train=num_cats_tr+num_dogs_tr
total_val=num_cats_val+num_dogs_val
print ('Котиків в тренувальному наборі даних:',num_cats_tr)
print ('Собачок в тренувальному наборі даних:',num_dogs_tr)
print ('Котиків в валідаційному наборі даних:',num_cats_val)
print ('Собачок в валідаційному наборі даних:',num_dogs_val)
print('--------')
print ('Всього зображень в тренувальному наборі даних:',total_train)
print ('Всього зображень в валідаційному наборі даних:',total_val)
```

```
Котиків в тренувальному наборі даних: 1000
Собачок в тренувальному наборі даних: 1000
Котиків в валідаційному наборі даних: 500
Собачок в валідаційному наборі даних: 500
--------
Всього зображень в тренувальному наборі даних: 2000
Всього зображень в валідаційному наборі даних: 1000
```

6. Створіть змінні, в яких будуть зберігатися параметри моделі:

```
BATCH_SIZE=100 # кількість зображень в блоці, який оброблює модель за одну ітерацію
IMG_SHAPE=150 #розмір, до якого буде приведено вхідне зображення
```

7. Підготуйте дані. Для цього необхідно виконати наступні дії:

1. Прочитати зображення з диску.

2. Декодувати зображення та перетворити його в необхідний формат з урахуванням RGB-профілю.

3. Привести зображення до тензорів із значеннями з плаваючою комою.

4. Провести нормалізацію значень тензору із інтервалу 0..255 до інтервалу 0..1

Усі ці операції виконуються за допомогою класу tensorflow.keras.preprocessing.image.ImageDataGenerator.

Створіть два генератори

train_image_generator=ImageDataGenerator(rescale=1./255)

validation_image_generator=ImageDataGenerator(rescale=1./255)

За допомогою методу flow_from_directory зробіть необхідні перетворення:

```
train_data_gen=train_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                         directory=train_dir,
                                                         shuffle=True,
                                                         target_size=(IMG_SHAPE,IMG_SHAPE),
                                                         class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
```

```
val_data_gen=validation_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                            directory=validation_dir,
                                                            shuffle=False,
                                                            target_size=(IMG_SHAPE,IMG_SHAPE),
                                                            class_mode='binary')
```

```
Found 1000 images belonging to 2 classes.
```

8. Візуалізуйте зображення із тренувального набору даних. Для цього скористайтеся генератором, що буде повертати блоки зображень. Кожен блок представляє собою кортеж, що містить зображення та мітки. Розробіть функцію, що намалює зображення в сітці розміром 1x5:

```
def plotImages(images_arr):
  fig,axes=plt.subplots(1,5,figsize=(20,20))
  axes=axes.flatten()
  for img,ax in zip (images_arr,axes):
    ax.imshow(img)
  plt.tight_layout()
  plt.show()
plotImages(sample_training_images[:5])
```

9. Розробіть модель. Модель буде містити 4 згорткових шари та 4 шари підвибірки за максимальним значенням. Flatten шар переводить дані з 2D в 1D формат. Модель також буде містити один повнозвязний шар з 512 нейронів та вихідний шар з двох нейронів.

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32,(3,3), activation='relu', input_shape=(IMG_
SHAPE,IMG_SHAPE,3)),
```

```
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
```

10. Проведіть компіляцію моделі:

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

11. Дослідіть структуру моделі за рівнями, використовуючи метод summary для розробленої моделі:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

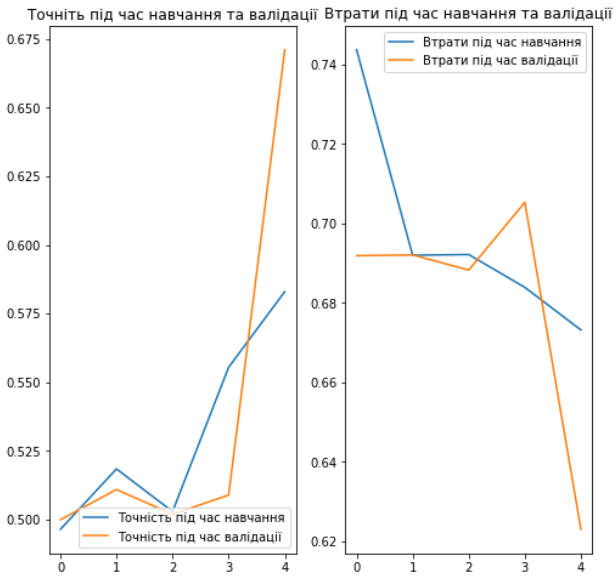
12. Проведіть тренування моделі:

```
EPOCHS=5
history=model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train/float(BATCH_SIZE))),
    epochs=EPOCHS,
```

```
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val/float(BATCH_SIZE))),
)
```

13. Візуалізуйте результати тренування моделі:

```
acc=history.history['accuracy']
val_acc=history.history['val_accuracy']
loss=history.history['loss']
val_loss=history.history['val_loss']
epochs_range=range(EPOCHS)
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(epochs_range,acc,label='Точність під час навчання')
plt.plot(epochs_range,val_acc,label='Точність під час валідації')
plt.legend(loc='lower right')
plt.title ('Точніть під час навчання та валідації')
plt.subplot(1,2,2)
plt.plot(epochs_range,loss,label='Втрати під час навчання')
plt.plot(epochs_range,val_loss,label='Втрати під час валідації')
plt.legend(loc='upper right')
plt.title ('Втрати під час навчання та валідації')
plt.savefig('./foo.png')
plt.show()
```



14. Отримайте аналогічні залежності для параметру EPOCHS=100

15. Дайте характеристику цим залежностям та визначте за графіками значення EPOCHS, що відповідає початку «перенавчання».

16. Дайте пояснення отриманим результатам та зробіть висновки.

Практична робота №. 7. Показники динаміки навчання нейромережі

Інтегральним показником динаміки навчання є критерій «Bias-variance-tradeoff» (відхилення-розкид-компроміс).

Bias – це відхилення між середнім значенням результатів, яке пропонує модель машинного навчання та дійсним значенням. Чим більше значення відхилення, тим більше модель спрощує дані (ігнорується більше факторів, які дозволяють створити коректну модель для цих даних та використовувати її для подальших передбачень). За великих значень відхилення маємо значну похибку на тренувальних та тестувальних наборах даних.

Відхилення визначає здатність моделі відповідати реальній функції. Причина великого відхилення у тому, що модель не враховує існуючі в даних закономірності. Наприклад, реальна функція є квадратичною, а прогнозована модель – першого порядку. Таким чином помилка, викликана відхиленням, зазвичай проявляється під час навчання.

Variance – це варіативність прогнозів, що пропонує модель. Цей показник характеризує наскільки модель є чутливою до шумів та завад, що присутні в даних. Чим більша варіативність моделі тим кращі результати вона демонструє на навчальному наборі даних, у той же час результати на тестовому наборі можуть бажати кращого.

Життєвий цикл моделі машинного навчання.

Етап 1. Характеризується високим відхиленням та низьким значенням розкиду. Цей стан відповідає моделі, що недовчилась (рис.1.13).

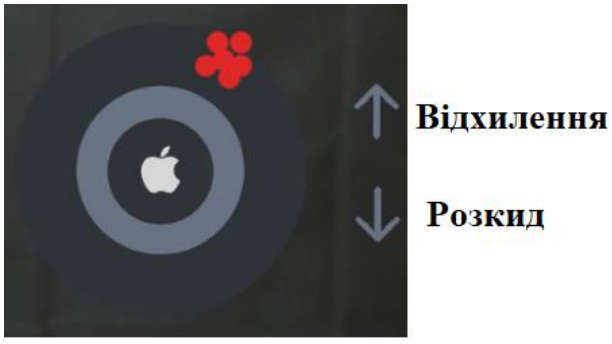


Рис. 1.13. Приклад моделі, що потребує додаткового навчання

Етап 2. На цьому етапі навчання характерною ознакою є збільшення розкиду результатів передбачень (хмаринка результатів має значні розміри). Це проміжний етап навчання моделі, коли вона тільки починає вдало підбирати відповідні вагові коефіцієнти (рис. 1.14).

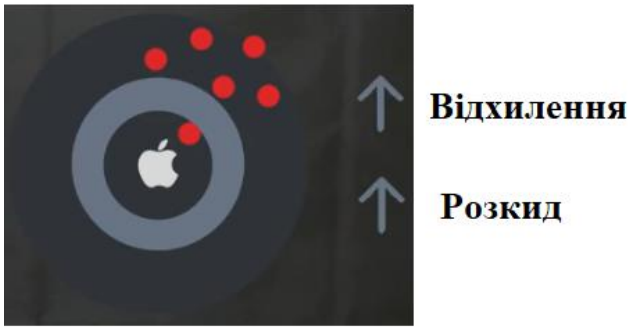


Рис. 1.14. Приклад моделі на проміжному етапі навчання

Етап 3. На цьому етапі модель не реагує на несуттєві зміни в даних (шуми) і дає вдалі передбачення, що характеризуються низькими значеннями відхилення. Модель навчилася (рис. 1.15).

Рис. 1.15. Фінальний етап навчання моделі

Подальше навчання нейромережі характеризується низьким значенням відхилення та значними розкидом результатів (висока варіативність результатів). На результати передбачення моделі починають впливати завади та шуми в даних (рис. 1.15). Модель перевчилася, вона почала брати до уваги ознаки, що не є ключовими. При цьому на тестовому наборі даних результати передбачень будуть суттєво гіршими порівняно з результатами навчання.

Рис. 1.16. Приклад моделі, що «перевчилась»

Таким чином, навчання нейромережі передбачає досягнення компромісу (Tradeoff) між відхиленням та розкидом. Збільшення складності алгоритмів (збільшення параметрів нейромережі) забезпечує одночасне зменшення відхилення та зростання чутливості моделі до шумів даних. Таким чином, існує мінімальне значення сумарної похибки, що відповідає оптимальній складності алгоритму (рис. 1.17).

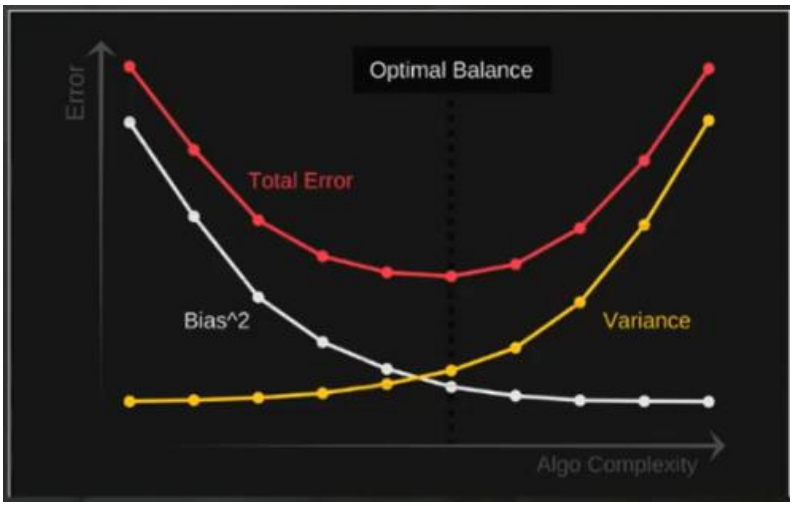


Рис. 1.17. Вплив кількості параметрів нейромережі на точність прогнозування

Практична частина

1.Імпортуйте необхідні пакети:

```
from __future__ import absolute_import, division, print_function, unico
de_literals
import os
import numpy as np
import matplotlib.pyplot as plt
```

Для чого використовується пакет os?

Для чого використовується пакет matplotlib?

2.Імпортуйте TensorFLOW та клас ImageDataGenerator:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

3. Завантажте архів з даними (datasets) з сервіса Kaggle. Використайте для цього метод get_file

```
_URL='https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'
zip_dir=tf.keras.utils.get_file('cats_and_dogs_filtered.zip',origin=_URL, extract=True)
```

Отримайте повний список директорій:

```
zip_dir_base=os.path.dirname(zip_dir)
!find $zip_dir_base -type d -print
```

```
/root/.keras/datasets
/root/.keras/datasets/cats_and_dogs_filtered
/root/.keras/datasets/cats_and_dogs_filtered/train
/root/.keras/datasets/cats_and_dogs_filtered/train/cats
/root/.keras/datasets/cats_and_dogs_filtered/train/dogs
/root/.keras/datasets/cats_and_dogs_filtered/validation
/root/.keras/datasets/cats_and_dogs_filtered/validation/cats
/root/.keras/datasets/cats_and_dogs_filtered/validation/dogs
```

4.Створіть змінні, в яких будуть зберігатися шляхи до директорій з наборами даних для тренувань та валідації:

```
base_dir=os.path.join(os.path.dirname(zip_dir),'cats_and_dogs_filtered')
train_dir=os.path.join(base_dir,'train')
validation_dir=os.path.join(base_dir,'validation')
train_cats_dir=os.path.join(train_dir,'cats')
train_dogs_dir=os.path.join(train_dir,'dogs')
validation_cats_dir=os.path.join(validation_dir,'cats')
validation_dogs_dir=os.path.join(validation_dir,'dogs')
```

5.Визначте кількість зображень в тестовому та валідаційному наборах даних. Застосуйте для цього метод len:

```
num_cats_tr=len(os.listdir(train_cats_dir))
num_dogs_tr=len(os.listdir(train_dogs_dir))

num_cats_val=len(os.listdir(validation_cats_dir))
num_dogs_val=len(os.listdir(validation_dogs_dir))
total_train=num_cats_tr+num_dogs_tr
total_val=num_cats_val+num_dogs_val
```

```
print ('Котиків в тренувальному наборі даних:',num_cats_tr)
print ('Собачок в тренувальному наборі даних:',num_dogs_tr)
print ('Котиків в валідаційному наборі даних:',num_cats_val)
print ('Собачок в валідаційному наборі даних:',num_dogs_val)
print('--------')
print ('Всього зображень в тренувальному наборі даних:',total_train)
print ('Всього зображень в валідаційному наборі даних:',total_val)
```

```
Котиків в тренувальному наборі даних: 1000
Собачок в тренувальному наборі даних: 1000
Котиків в валідаційному наборі даних: 500
Собачок в валідаційному наборі даних: 500
--------
Всього зображень в тренувальному наборі даних: 2000
Всього зображень в валідаційному наборі даних: 1000
```

6. Створіть змінні, в яких будуть зберігатися параметри моделі:

```
BATCH_SIZE=100 # кількість зображень в блоці, який оброблює модель за одну ітерацію
IMG_SHAPE=150 #розмір, до якого буде приведено вхідне зображення
```

7. Проведіть розширення даних. Перенавчання з'являється, коли в наборі даних для навчання мало прикладів. Один із способів вирішення цієї проблеми – це розширення даних до необхідної кількості та варіативності.

Розширення даних представляє собою процес генерації даних із існуючих екземплярів шляхом певних трансформацій початкового набору даних. Метою таких дій є розширення (збільшення) кількості унікальних вхідних екземплярів даних, які раніше не зустрічалися. Опрацювання додаткових вхідних даних дозволить моделі краще їх узагальнити та демонструвати вищу точність на валідаційному наборі даних.

Використання класу tf.keras надає можливість реалізації випадкових перетворень та генерації нових зображень за допомогою класу ImageDataGenerator, якому достатньо передати у вигляді параметру різні трансформації, що будуть застосовуватися до зображень.

8. Розробіть функцію, що буде відображати зображення в сітці розміром 1x5:

```
def plotImages(images_arr):
  fig,axes=plt.subplots(1,5,figsize=(20,20))
  axes=axes.flatten()
  for img,ax in zip (images_arr,axes):
```

```
    ax.imshow(img)
  plt.tight_layout()
  plt.show()
plotImages(sample_training_images[:5])
```

9. Поверніть (відобразіть) зображення по горизонталі. Для цього передайте додатковий параметр horizontal_flip=True генератору зображень:

```
image_gen=ImageDataGenerator(rescale=1./255, horizontal_flip=True)
```

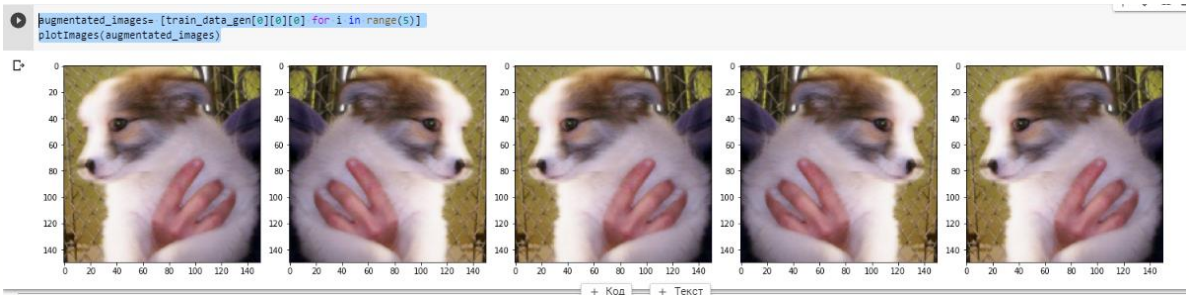
10. За допомогою методу flow_from_directory створіть потік даних з директорії :

```
[15] train_data_gen=image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                                  directory=train_dir,
                                                  shuffle=True,
                                                  target_size=(IMG_SHAPE,IMG_SHAPE))

     Found 2000 images belonging to 2 classes.
```

11. Відобразіть п’ять трансформованих зображень:

```
augmentated_images= [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmentated_images)
```



12. Виконайте трансформацію повороту зображень на заданий кут (наприклад 45 град.):

```
image_gen=ImageDataGenerator(rescale=1./255, rotation_range=45)
```

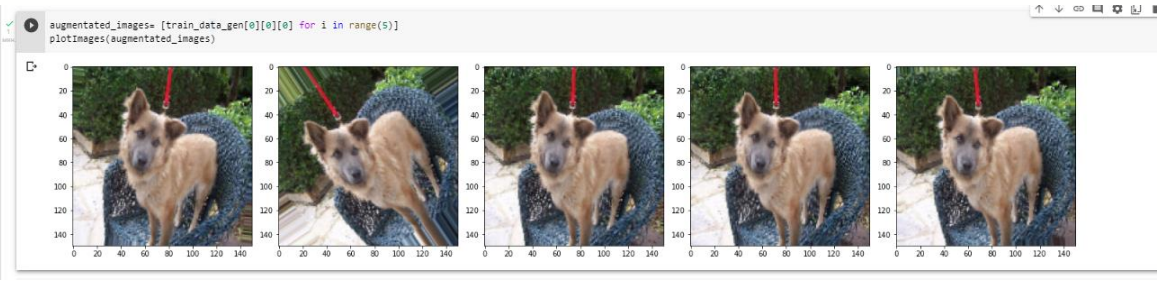
```
train_data_gen=image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                             directory=train_dir,
                                             shuffle=True,
                                             target_size=(IMG_SHAPE,IMG_SHAPE))

Found 2000 images belonging to 2 classes.
```

Та відобразіть п’ять трансформованих зображень:

```
augmentated_images= [train_data_gen[0][0][0] for i in range(5)]
```

plotImages(augmentated_images)



13. Виконайте трансформацію довільного збільшення зображення (до x50%):

image_gen=ImageDataGenerator(rescale=1./255, zoom_range=0.5)

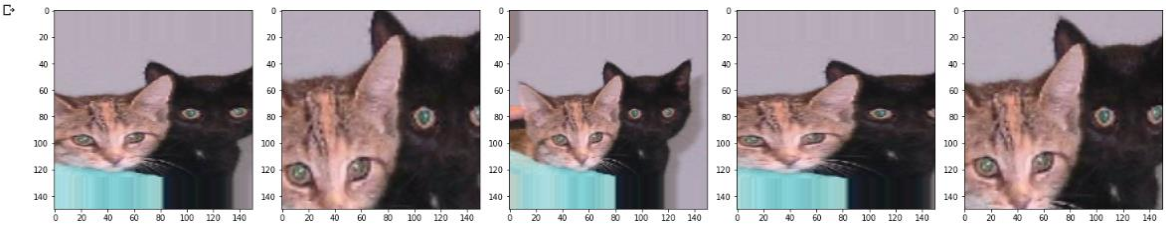
```
train_data_gen=image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                             directory=train_dir,
                                             shuffle=True,
                                             target_size=(IMG_SHAPE,IMG_SHAPE))
```

```
Found 2000 images belonging to 2 classes.
```

Та відобразіть п’ять трансформованих зображень:

augmentated_images= [train_data_gen[0][0][0] for i in range(5)]

plotImages(augmentated_images)



14. Об’єднайте розглянуті трансформації зображень та включіть їх до навчального набору даних:

```
image_gen_train=ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.5,
    horizontal_flip=True,
    fill_mode='nearest'
```

)

```
[23] train_data_gen=image_gen_train.flow_from_directory(batch_size=BATCH_SIZE,
                                                        directory=train_dir,
                                                        shuffle=True,
                                                        target_size=(IMG_SHAPE,IMG_SHAPE),
                                                        class_mode='binary'
                                                       )
```

```
Found 2000 images belonging to 2 classes.
```

За що відповідають параметри:

width_shift_range=0.2,

height_shift_range=0.2,

shear_range=0.2,

fill_mode='nearest' ?

15. Створіть аналогічний генератор для валідаційного набору даних та застосуйте його для їхньої обробки:

validation_image_generator=ImageDataGenerator(rescale=1./255)

```
[25] val_data_gen=image_gen_val.flow_from_directory(batch_size=BATCH_SIZE,
                                                    directory=validation_dir,
                                                    target_size=(IMG_SHAPE,IMG_SHAPE),
                                                    class_mode='binary')
```

```
Found 1000 images belonging to 2 classes.
```

16. Розробіть модель, що буде містити:

- 4 згорткових шари та 4 шари підвибірки за максимальним значенням.
- шар Dropout 50 % нейронів попереднього шару (MaxPoooling) випадковим чином обнулити. Цей процес буде відбуватися на кожній ітерації навчання.
- Flatten шар переводить дані з 2D в 1D.
- Повнозвязний шар з 512 нейронів.
- Вихідний шар з двох нейронів.

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32,(3,3), activation='relu', input_shape=(IMG_SHAPE,IMG_SHAPE,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
```

```
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
```

17. Проведіть компіляцію моделі:

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

18. Дослідіть структуру моделі за рівнями, використовуючи метод summary (model.summary():

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 dropout (Dropout)           (None, 7, 7, 128)         0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

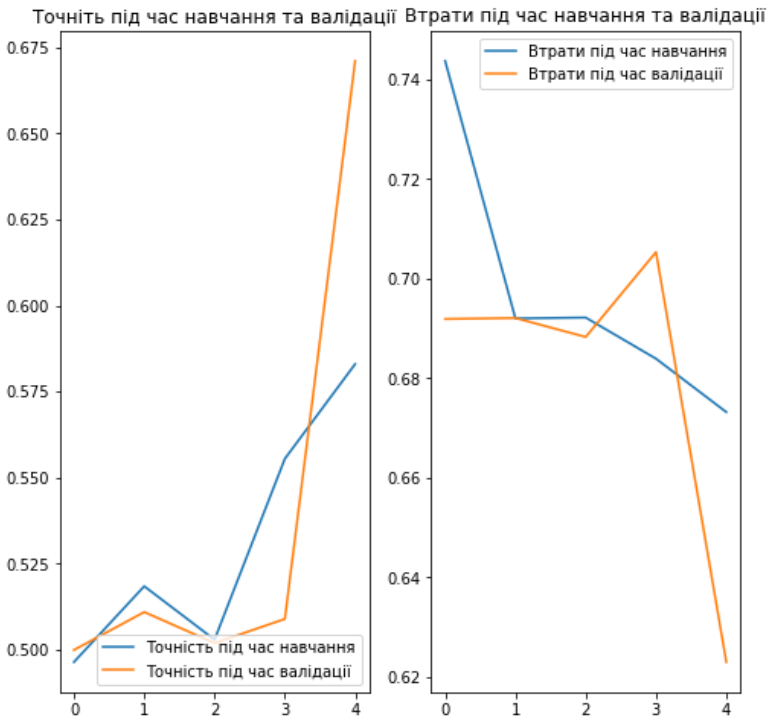
12. Проведіть тренування моделі:

```
EPOCHS=5
history=model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train/float(BATCH_SIZE))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val/float(BATCH_SIZE))),
```

)

13. Візуалізуйте результати тренування моделі.

```
acc=history.history['accuracy']
val_acc=history.history['val_accuracy']
loss=history.history['loss']
val_loss=history.history['val_loss']
epochs_range=range(EPOCHS)
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(epochs_range,acc,label='Точність під час навчання')
plt.plot(epochs_range,val_acc,label='Точність під час валідації')
plt.legend(loc='lower right')
plt.title ('Точніть під час навчання та валідації')
plt.subplot(1,2,2)
plt.plot(epochs_range,loss,label='Втрати під час навчання')
plt.plot(epochs_range,val_loss,label='Втрати під час валідації')
plt.legend(loc='upper right')
plt.title ('Втрати під час навчання та валідації')
plt.savefig('./foo.png')
plt.show()
```



14. Отримайте аналогічні залежності для параметру EPOCHS=100

15. Дайте характеристику цим залежностям та визначте за графіками значення EPOCHS, що відповідає початку «перенавчання».

16. Поясніть як вплинуло розширення набору даних навчальної та тестувальної вибірок на ефект «перенавчання».

17. Дайте пояснення отриманим результатам та зробіть висновки.

2. Перелік додаткових тем практичних занять

Таблиця 1

№	Назва теми
1	Основи організації будівельного виробництва на основі кіберфізичних систем.
2	Архітектура кіберфізичних систем.
3	Елементна база мікропроцесорної техніки для вбудованих та кіберфізичних систем.
4	Датчики та виконавчі механізми кіберфізичних систем.
5	Вимірювання, обробка та генерація сигналів в сенсорних підсистемах кіберфізичних систем
6	Цифрові та аналогові інтерфейси введення-виведення в кіберфізичних систем.
7	Основні протоколи бездротового зв'язку в Інтернеті речей: LoRa / LoRaWAN, 6LoWPAN, NB-IoT, GSM, Wi-Fi, Bluetooth. Фізичні основи, основні параметри і умови застосування.
8	Моделювання кіберфізичних систем у віртуальних середовищах.
9	Аналіз та верифікація кіберфізичних систем.

Список літератури

1. *Sanfelice R. G.* Analysis and Design of Cyber-Physical Systems: A Hybrid Control Systems Approach / R. G. Sanfelice. – CRC Press, 2016.
2. *Мельник А.О.* Кіберфізичні системи: багаторівнева організація та проєктування / А.О. Мельник, В.А. Мельник, В.С. Глухов, А.М. Сало. – Магнолія.–2023. – 238с.
3. *Бочкарьов О.Ю.* Кіберфізичні системи: технології збору даних / О.Ю. Бочкарьов, В.А. Голембо, Я.С. Парамуд. – Магнолія.–2023. –176 с.
4. *Khaitan S.K.* Design Techniques and Applications of Cyber Physical Systems: A Survey / *S.K. Khaitan*, J. McCalley – IEEE Systems Journal, 2014, 9(2), pp.1-16.
5. *Шикалов В.С.* Технологічні вимірювання: навч. посіб. / В.С. Шикалов. – Київ :Кондор, 2007. – 168 с.

Навчально-методичне видання

РОЗРОБКА КІБЕРФІЗИЧНИХ СИСТЕМ

Методичні вказівки
до виконання практичних занять
для здобувачів другого (магістерського рівня)
вищої освіти спеціальності 174 «Автоматизація,
комп`ютерно-інтегровані технології та робототехніка»

Укладач **Луценко** Вадим Юрійович

Комп’ютерне верстання *А. П. Селівестрової*

Ум. друк. арк. 2,56. Обл.-вид. арк. 2,75
Електронний документ. Вид № 58/V-25.

Виконавець і виготовлювач

Київський національний університет будівництва і архітектури
Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб’єктів
видавничої справи ДК № 808 від 13.02.2002 р