ISSN 2707-1898 (print)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2024.01.037
Article received 22.04.2024 p.
Article accepted 30.04.2024 p.
UDK 004.4

Correspondence author
S. S. Ivasiv
slav.ivasiv.mknus.2022@lpnu.ua

УДК 004.02

В. М. Теслюк, С. С. Івасів
Національний університет "Львівська політехніка", м. Львів, Україна

МОДЕЛІ ТА ЗАСОБИ КЛАСИФІКАЦІЇ ПАТЕРНІВ ЕЛЕМЕНТІВ ОДЯГУ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Завдання класифікації патернів не втрачає актуальності у сферах трендів, стилю, моди, персоналізації, виробництва та дизайну. Висвітлено дослідження, спрямоване на проєктування та розроблення моделей і засобів класифікації патернів елементів одягу із використанням машинного навчання. Воно вирішує актуальне питання комп'ютерного зору, а саме підвищення ефективності класифікації патернів елементів одягу. Дослідження виконано з власним набором даних, що містить 600 зображень. Для здійснення класифікації визначено такі патерни: "у клітинку", "у крапку", "рослинність / квітковий", "принт", "суцільний", "у смужку". За допомогою мови програмування Python і фреймворків глибокого навчання Keras та TensorFlow розроблено згорткову нейронну мережу. Використано масштабований фреймворк Keras-Tuner для оптимізації гіперпараметрів розробленої мережі. Структура згорткової нейронної мережі містить вхідну частину, частину визначення характерних ознак у зображенні та частину для визначення типу патерну. Описано архітектуру використаної згорткової нейронної мережі. Застосовано набір інструментів CUDA, бібліотеку cuDNN та прошарок WSL для навчання згорткової нейронної мережі за допомогою графічного процесора, що дало змогу істотно пришвидшити навчання згорткової нейронної мережі. Для оцінювання розробленої згорткової нейронної мережі використано метрики, які охоплюють точність, прогностичну значущість позитивного результату та повноту. Вебзастосунок розроблено мовою програмування Python із фреймворком FastAPI. Вебзастосунок має описаний прикладний програмний інтерфейс для взаємодії зі згортковою нейронною мережею, використано також бібліотеки Pillow (PIL) для роботи із зображеннями та Rembg для видалення фону зображення. Користувацький інтерфейс розроблено мовою програмування JavaScript із використанням мови розмітки, каскадних таблиць стилів та фреймворком React. Інтерфейс користувача представлено як інтуїтивний інструмент для взаємодії із системою. Розроблене програмне забезпечення використовує модульний принцип, що дає змогу швидко модернізувати програмний засіб. У результаті застосування підходу перенесення навчання досягнуто точність тестування 93,33 %, а з використанням підходу тонкого налаштування отримано остаточну версію згорткової нейронної мережі для класифікації патернів елементів одягу із точністю тестування 95 %. Навчену нейронну мережу апробовано на нових зображеннях визначених типів патернів, наведено приклади для двох патернів.

Ключові слова: Visual Geometry Group 16 (VGG16), класифікація патернів, тонке налаштування, перенесення навчання, згорткова нейронна мережа.

Вступ / Introduction

Розпізнавання патернів на елементах одягу є актуальним і важливим аспектом для багатьох сфер, зокрема виробництва, дизайну та технологій, із урахуванням моди. Окремі аспекти актуальності цього питання розкрито у темах трендів та стилю [1], автоматизації виробництва [2], персоналізації [3, 4, 5], розширеної реальності [6, 7], оптимізації виробництва [8], захисту від підробок [9].

Загалом розпізнавання патернів на одязі дає змогу поєднати традиційні елементи з інноваційними технологіями, роблячи модний світ ефективнішим, цікавішим та відповідальнішим.

Основне завдання дослідження – класифікація патернів елементів одягу, що залежить від вибраної архітектури нейронної мережі, її гіперпараметрів та підходів, застосованих для навчання нейронної мережі. Іншим важливим аспектом є набір даних, який буде використовуватись для тренування, валідації та тестування отриманої нейронної мережі.

Уже існує низка застосувань розпізнавання та класифікації патернів, а підходи до їх вивчення постійно удосконалюються та застосовуються до різних сфер діяльності, тому досліджувана тема і розроблення моделей та засобів класифікації патернів з використанням машинного навчання є актуальними.

Об'єкт дослідження – процес класифікації патернів елементів одягу із використанням машинного навчання.

Предмет дослідження – моделі та засоби класифікації патернів елементів одягу із використанням машинного навчання.

Мета роботи – дослідження та реалізація моделей та засобів для підвищення ефективності класифікації патернів елементів одягу із використанням машинного навчання.

Для досягнення поставленої мети визначено такі основні *завдання дослідження*:

- здійснення огляду наукових досліджень за тематикою класифікації патернів елементів одягу;

- синтез та оцінювання нових моделей на основі методів машинного навчання;
- розроблення засобів класифікації патернів елементів одягу із використанням машинного навчання та здійснення досліджень.

Матеріали і методи дослідження. У дослідженні застосовано метод об'єктно-орієнтованого підходу – для розроблення застосунку, метод машинного навчання – навчання з учителем, використано також структури даних, а саме: словники, списки та кортежі. Під час застосування штучних нейронних мереж використано засади згорткових нейронних мереж [10], актуальні архітектури згорткових нейронних мереж, окремі розділи лінійної алгебри, математичного аналізу, основ оптимізації та теорії ймовірності.

Дослідження здійснено із використанням мови програмування Python у середовищах PyCharm та Jupyter Notebook [11, 12]. Для організації штучної нейронної мережі залучено фреймворки TensorFlow та Keras [13, 14]. Набір інструментів CUDA toolkit, бібліотеку cuDNN та прошарок WSL використано для навчання штучних нейронних мереж за допомогою графічного процесора [15, 16, 17]. Застосунок побудовано на фреймворку FastAPI [18]. Користувацький інтерфейс розроблено з використанням мови програмування JavaScript та фреймворку React у середовищі WebStorm. Для видалення фону зображень використано бібліотеки Pillow (PIL) та Rembg [19]. Графіки побудовано за допомогою бібліотеки matplotlib [20].

Аналіз останніх досліджень та публікацій. У роботі "Система для розпізнавання елементів одягу та їх кольорів на зображенні" [21] описано підходи до визначення розташування одягу на зображенні, класифікації елементів одягу та визначення кольору елемента одягу. Проте взаємодія із застосованою нейронною мережею для визначення розташування одягу та його класифікації відбувається через прикладний програмний інтерфейс, який надає платформа Clarifai. Платформа пропонує кілька мереж, проте доступу до архітектури моделей не надає, відповідно змінити архітектуру та здійснити донавчання мережі на власних даних неможливо.

Проаналізовано роботу "Використання переносу навчання для розпізнавання патернів одягу за допомогою камери, встановленої на пальці" [22], яка описує проблематику і рішення для класифікації патернів. У цій статті автори досліджують використання камери, встановленої на пальці, для розпізнавання кольорів і візуальних патернів на одязі. Основний акцент зроблено на моделях нейронних мереж, зокрема на застосуванні перенесення навчання для тренування глибокої нейронної мережі. Автори описують збирання даних та створення набору даних для тренування моделі. Вони використовують шість звичайних візуальних малюнків, які важко або неможливо розрізнити за допомогою дотику: суцільний, смугастий, клітинку, крапковий, зигзагоподібний та квітковий. Науковці використали модель глибокої нейронної мережі ResNet-101, попередньо навчену на публічному наборі даних ImageNet. Застосовано техніку перенесення навчання для тренування моделі на власному наборі даних. Точність класифікації набору зображень від камери, встановленої на пальці, досягає 92 %. Дослідники виявили проблеми плутанини через недостатній контекст чи грубі нитки і прагнули вдосконалити модель, використовуючи додаткові зображення з цільового домену. В авторів є плани подальших досліджень, таких як вивчення впливу позиції камери на пальці користувача на точність системи, а також оцінювання деталей патернів.

Робота "Автоматизована система ідентифікації класу дизайну одягу" [23] описує не застосування нейронних мереж, а алгоритми обробки зображень. Автори розглядають проблему ідентифікації класів дизайну одягу на зображеннях із використанням нових дескрипторів зображень. Вони пропонують два нові дескриптори: Completed CENsus Transform hISTogram (cCENTRIST) і Ternary CENsus Transform hISTogram (tCENTRIST), які ґрунтуються на відомих методах оброблення зображень, таких як Completed Local Binary Pattern (CLBP), Local Ternary Pattern (LTP) і CENsus TRanformed hISTogram (CENTRIST). Автори дослідили ефективність запропонованих методів, порівнюючи їх з іншими відомими методами, такими як Histogram of Oriented Gradients (HOG), GIST, Local Gradient Pattern (LGP) й оригінальним методом CENTRIST. Вони також порівняли свої методи із використанням попередньо навчених моделей глибокого навчання, таких як CaffeNet. Дослідники досягли успіхів, підвищивши точність класифікації дизайну одягу на зображеннях за допомогою запропонованих методів. Для набору даних Clothing Attribute Dataset [24] метод cCENTRIST забезпечив точність 74,97 %, а найвищої точності 84,23 % досяг на наборі даних Fashion Dataset [25]. Показано також, що результати запропонованих методів перевершують результати, отримані з використанням попередньо навчених моделей глибокого навчання.

Основна мета роботи "Підхід для класифікації текстур одягу, оснований на згорткових нейронних мережах" [26] – підвищити точність класифікації патернів одягу. Автори запропонували свою модель, яка ґрунтується на моделях AlexNet та VGG_S, де використовують усі дев'ять навчених шарів. Порівняно з AlexNet, до моделі додано новий повністю з'єднаний шар (FC3), а також використано техніку аугментації даних для зменшення перенавчання на етапі навчання. Зображення взято з двох публічно доступних наборів даних: Fashion [25] і Clothing Attribute [24]. Обидва набори даних були категоризовані для навчання та тестування з метою ідентифікації класів дизайну одягу. Автори порівнюють ефективність запропонованої моделі із двома відомими CNN-моделями (AlexNet та VGGNet) та деякими передовими методами витягування ознак, розробленими вручну. Експериментальне оточення для цього дослідження було налаштовано із використанням моделі CaffeNet й операційної системи Ubuntu 12.4. Для прискорення обчислень у дослідженні використано графічний процесор. Запропонована модель досягла точності 77,8 % на наборі Clothing Attribute Dataset із шістьма різними класами, а також точності 84,5 % на наборі Fashion Dataset із п'ятьма класами текстур. Це свідчить про успішність використання глибоких згорткових нейронних мереж у класифікації дизайну текстури одягу та відкриває перспективи для подальших досліджень у цій сфері.

Виконавши аналіз, можна зробити висновок, що точність класифікації патернів елементів одягу необхідно підвищити, тому варто використати сучасні архітектури нейронних мереж, здійснити підбір гіперпар-

метрів для досягнення вищої точності класифікації, ніж продемонстрована в оглянутих джерелах.

Результати дослідження та їх обговорення / Research results and their discussion

Підготовка середовища. Для розроблення використано один персональний комп'ютер з операційною системою Windows. Для того, щоб виконати навчання штучної нейронної мережі за допомогою графічного процесора, використано підсистему Windows для Linux, що дасть змогу запустити Linux як гостьову систему в межах ОС Windows. У гостьову систему встановлено набір інструментів CUDA та бібліотеку cuDNN. Для підготовки середовища, яке дає змогу використовувати графічний процесор для обчислень, виконано таку послідовність дій:

1) встановлено драйвер графічного процесора Nvidia;
2) у Windows завантажено програму WSL;
3) запущено командний рядок і виконано команду для встановлення дистрибутиву Ubuntu як гостьової системи:

```
wsl --install
```

4) вказано назву користувача гостьової системи та пароль гостьової системи. Після цього поточний командний рядок містить запущену гостьову систему Linux Ubuntu і відповідає командному рядку гостьової системи;
5) здійснено видалення старого GPG ключа командою:

```
sudo apt-key del 7fa2af80
```

6) здійснено встановлення набору інструментів CUDA 11.8 (CUDA Toolkit 11.8), виконано такі команди:

```
wget
  https://developer.download.nvidia.com/c
  ompute/cuda/repos/wsl-
  ubuntu/x86_64/cuda-wsl-ubuntu.pin
sudo        mv        cuda-wsl-ubuntu.pin
  /etc/apt/preferences.d/cuda-repository-
  pin-600
wget
  https://developer.download.nvidia.com/c
  ompute/cuda/11.8.0/local_installers/cud
  a-repo-wsl-ubuntu-11-8-local_11.8.0-
  1_amd64.deb
sudo dpkg -i cuda-repo-wsl-ubuntu-11-8-
  local_11.8.0-1_amd64.deb
sudo cp /var/cuda-repo-wsl-ubuntu-11-8-
  local/cuda-*-keyring.gpg
  /usr/share/keyrings/
sudo apt-get update
sudo apt-get -y install cuda
```

7) перевірено, чи додалась змінна середовища, яка вказує на те, де шукати виконувані файли набору інструментів CUDA 11.8, виконано команду:

```
echo $PATH
```

Додано шлях до виконуваних файлів CUDA за допомогою команди:

```
export              PATH="/usr/local/cuda-
  11.8/bin:$PATH"
```

8) перевірено, чи у директорії "/usr/local/cuda-11.8" містяться директорії "include" та "lib64";
9) завантажено архів з бібліотекою глибокої нейронної мережі CUDA (CUDA Deep Neural Network library, cuDNN) версії 8.7 саме для CUDA Toolkit 11.8. Архів переміщено у директорію користувача гостьової системи;
10) видобуто файли з архіву, що містить бібліотеку cuDNN, та перенесено відповідні файли у розташування CUDA Toolkit 11.8. Виконано такі команди:

```
cd /home/YOUR_UBUNTU_USER
tar       -xvf       cudnn-linux-x86_64-
  8.9.6.50_cuda11-archive.tar.xz
sudo       cp       cudnn-linux-x86_64-
  8.9.6.50_cuda11-
  archive/include/cudnn*.h
  /usr/local/cuda-11.8/include
sudo    cp    -P    cudnn-linux-x86_64-
  8.9.6.50_cuda11-archive/lib/libcudnn*
  /usr/local/cuda-11.8/lib64
sudo    chmod    a+r    /usr/local/cuda-
  11.8/include/cudnn*.h   /usr/local/cuda-
  11.8/lib64/libcudnn*
```

виконано команду, щоб перевірити, чи встановлено CUDA:

```
python3 -c "import tensorflow as tf;
  print(tf.config.list_physical_devices('
  GPU'))"
```

Отримано рядок "[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]", що свідчить про успішне налаштування інструментів та бібліотек.

Загальна структура програмного забезпечення. Розроблено програмне забезпечення для класифікації елементів одягу. Зокрема, на рис. 1 зображено діаграму компонентів, яка демонструє залежності між компонентами системи.

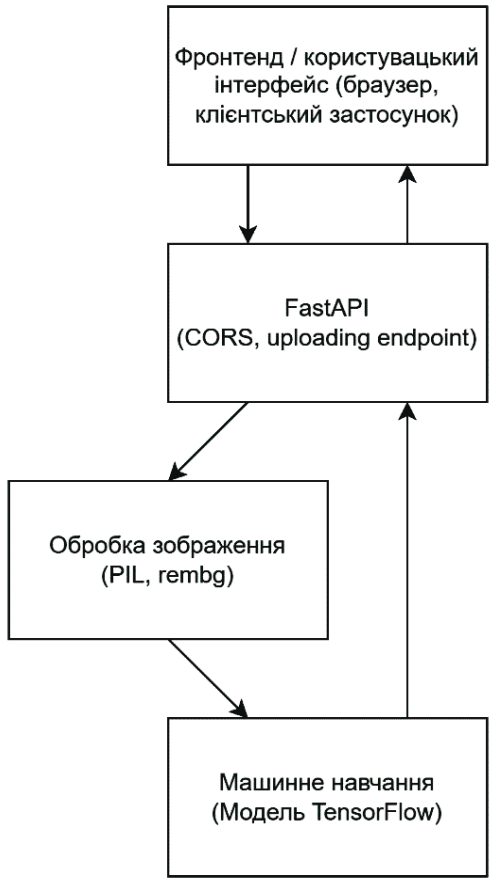

Рис. 1. Діаграма компонентів системи / Diagram of system components

Діаграма компонентів містить такі складові:

- фронтенд/користувацький інтерфейс представляє клієнтську частину вебзастосунку, де користувач взаємодіє із вебзастосунком через браузер або інший клієнтський застосунок;
- FastAPI відповідає за опрацювання HTTP-запитів та відповідей. Він містить модуль CORS, що дає змогу взаємодіяти із додатком із різних джерел. Upload Endpoint є HTTP-маршрутом, який обробляє завантаження файлів. Під час завантаження файлів відбувається обробка зображень для подальшого використання у моделі машинного навчання;
- обробка зображення передбачає бібліотеки для обробки зображень, такі як Pillow (PIL, Python Imaging Library) для роботи із зображеннями та бібліотеку Rembg для видалення фону;
- машинне навчання використовує TensorFlow-модель для класифікації патернів елементів одягу, модель для класифікації та повертає результати FastAPI;
- FastAPI надсилає відповідь користувачеві або іншим клієнтам.

Розроблене програмне забезпечення використовує модульний принцип, що дає змогу швидко модернізувати програмний засіб.

Структура набору вхідних даних. Дослідження здійснюється з одним набором даних, розділеним на три вибірки: “Тренувальна вибірка”, “Валідаційна вибірка” та “Тестувальна вибірка”; кожна вибірка містить зображення певного класу. Структуру набору даних, яка є ієрархією директорій, зображено на рис. 2.

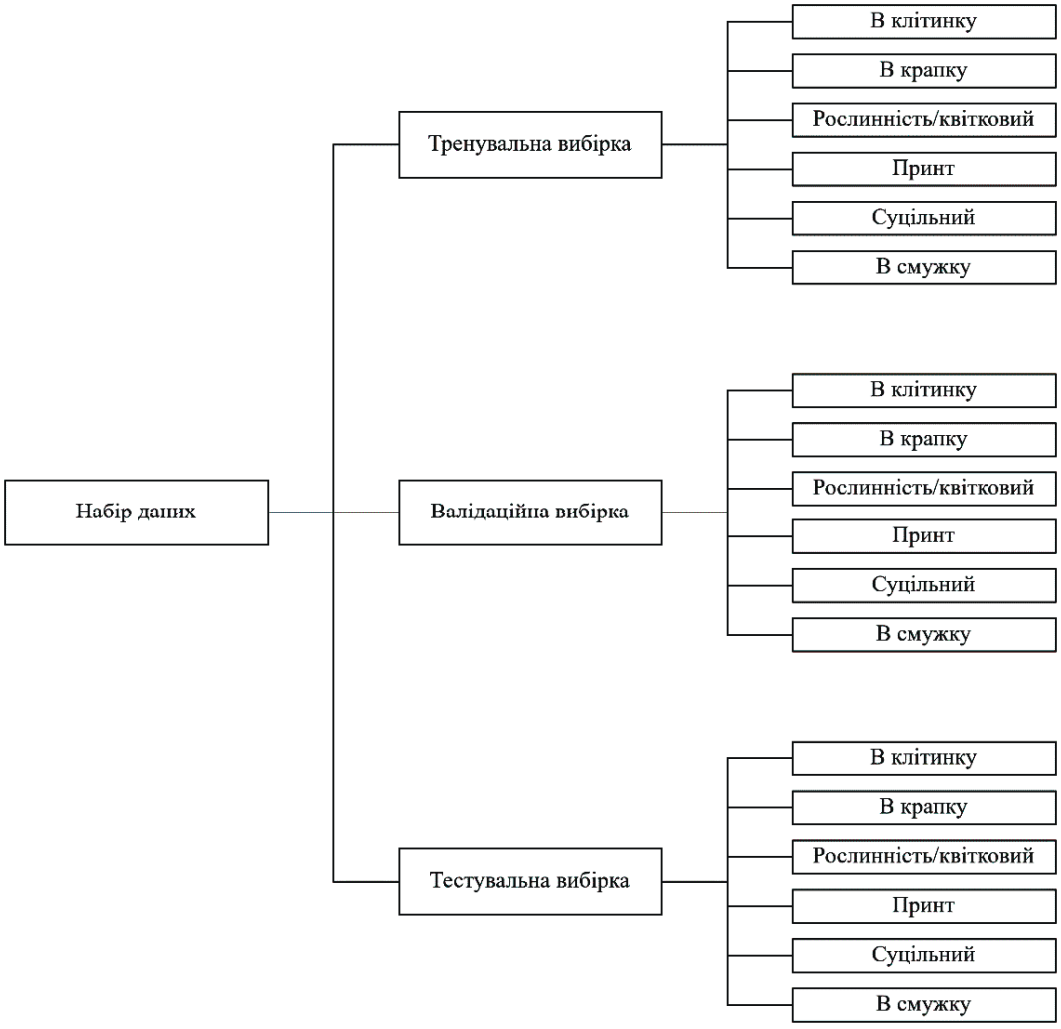

Рис. 2. Структура набору даних / The structure of the dataset

Для здійснення класифікації визначено такі патерни: “в клітинку”, “в крапку”, “рослинність/квітковий”, “принт”, “суцільний”, “в смужку”. Для виконання поставленого завдання вручну для кожного класу вибрано по 100 зображень, загалом 600. Зображення використано для навчання, валідації та тестування моделі із застосуванням генераторів, які під час кроку епохи вибирають мінівибірку (батч) зображень з відповідної директорії, застосовують аугментацію (якщо її задано), а потім передають ці зображення з відповідними мітками мережі. Мітками є директорії, які й позначають класи. Генератори дають змогу не завантажувати всі зображення у пам’ять, а працювати із мінівибірками (батчами).

Архітектура застосованої нейронної мережі. На рис. 3 зображено архітектуру згорткової нейронної мережі.

Рис. 3. Архітектура мережі для класифікації патернів елементів одягу / Network architecture for pattern classification of clothing elements

Опис архітектури мережі

На вхід подається зображення розміром 356 на 356 пікселів у трьох каналах (RGB).

Згорткова частина моделі VGG16:

- блок 1: згортковий шар типу Conv2D, фільтрів 64, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (356, 356, 64), кількість параметрів 1792;
- блок 1: згортковий шар типу Conv2D, фільтрів 64, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (356, 356, 64), кількість параметрів 36926;
- блок 1: шар пулінгу типу MaxPooling2D, фільтрів 64, розмір фільтра 2×2, крок 2, вихідний розмір (178, 178, 64);
- блок 2: згортковий шар типу Conv2D, фільтрів 128, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (178, 178, 128), кількість параметрів 73856;
- блок 2: згортковий шар типу Conv2D, фільтрів 128, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (356, 356, 128), кількість параметрів 147584;
- блок 2: шар пулінгу типу MaxPooling2D, фільтрів 128, розмір фільтра 2×2, крок 2, вихідний розмір (89, 89, 128);
- блок 3: згортковий шар типу Conv2D, фільтрів 256, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (89, 89, 128), кількість параметрів 73856;
- блок 3: згортковий шар типу Conv2D, фільтрів 256, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (89, 89, 128), кількість параметрів 590080;
- блок 3: згортковий шар типу Conv2D, фільтрів 256, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (89, 89, 128), кількість параметрів 590080;
- блок 3: шар пулінгу типу MaxPooling2D, фільтрів 256, розмір фільтра 2×2, крок 2, вихідний розмір (44, 44, 256);
- блок 4: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (44, 44, 512), кількість параметрів 1180160;
- блок 4: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (44, 44, 512), кількість параметрів 2359808;
- блок 4: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (44, 44, 512), кількість параметрів 2359808;
- блок 4: шар пулінгу типу MaxPooling2D, фільтрів 512, розмір фільтра 2×2, крок 2, вихідний розмір (22, 22, 512);
- блок 5: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (22, 22, 512), кількість параметрів 2359808;

- блок 5: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (22, 22, 512), кількість параметрів 2359808;
- блок 5: згортковий шар типу Conv2D, фільтрів 512, розмір фільтра 3×3, крок 1, доповнення Same, функція активації ReLU, вихідний розмір (22, 22, 512), кількість параметрів 2359808;
- блок 5: шар пулінгу типу MaxPooling2D, фільтрів 512, розмір фільтра 2×2, крок 2, вихідний розмір (11, 11, 512).

Частина для класифікації:

- шар типу Flatten, вихідний розмір (61952);
- повнозв'язний шар, функція активації Sigmoid, відключення 0,5, вихідний розмір (384);
- повнозв'язний шар, функція активації SoftMax, вихідний розмір (6).

Проведені експерименти. Набір даних розділено у відношенні: тренувальна вибірка – 80 %, валідаційна вибірка – 10 %, тестова вибірка – 10 %.

Встановлено розмір мінівибірки 32, розмір вхідного зображення – 356×356 пікселів.

Спершу вручну підібрано гіперпараметри для виділення моделі, з якою далі здійснюватиметься дослідження, а параметрами порівняння є точність навчання та точність тестування.

Для тестування використано моделі VGG16 [27], VGG19 [28], InceptionV3 [29], Xception [30], ResNet50 [31]. Для кожної моделі завантажено ваги за набором даних ImageNet, відкинуто повнозв'язні шари, які відповідають за класифікацію, вимкнено навчання для всіх елементів мережі. Засобами бібліотеки TensorFlow створено послідовну мережу, де першим елементом стають мережі VGG16, VGG19, InceptionV3, Xception, ResNet50, а точніше, їх частини, які відповідають за визначення характерних ознак у зображенні, тоді додано шар розгладжування, повнозв'язний шар, що містить 256 нейронів, додано функцію активації ReLU та випадкове відключення зі значенням 0,5, також додано повнозв'язний шар із шести нейронів із функцією активації SoftMax. Формула (1) описує функцію активації SoftMax:

$$\upsilon(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad (1)$$

де $\vec{Z}$ – вектор, який складається з K елементів, z – позначає елемент вектора $\vec{Z}$; *чисельник* –експоненційна функція застосовується до кожного елемента вектора, повертаючи найвище вихідне значення для найвищого вхідного значення, будь-які мінуси також стають позитивними, оскільки їх діапазон становить (0, ∞); *знаменник* – сума нормалізує кожен елемент, гарантуючи, що сума функції дорівнюватиме 1 і створить розподіл ймовірностей. Усі піднесені до степеня елементи додаються разом, тож коли кожен піднесений до степеня елемент розділити на цю суму, це буде її відсоток.

Функцію втрат задано "категоріальна крос-ентропія", оптимізатор Adam зі швидкістю навчання 0.001. Задано кількість епох, яка становить 15.

Для вказаних гіперпараметрів отримано результати, подані у табл. 1.

Подальше дослідження виконано із тестуванням моделі VGG16, яка показала найкращий результат за ручного підбирання гіперпараметрів.

Саме на цьому етапі було використано Keras-Tuner [32]. Для Keras-Tuner встановлено алгоритм пошуку Bayesian Optimization. Завантажено ваги мережі VGG16 за набором даних ImageNet, відкинуто повнозв'язні шари для класифікації, вимкнено навчання для всіх елементів мережі. Засобами бібліотеки TensorFlow створено послідовну мережу, де першим елементом є частина мережі VGG16, яка відповідає за визначення характерних ознак у зображенні, тоді додано шар розгладжування. Кількість повнозв'язаних шарів у діапазоні від 1 до 2, у кожному шарі може бути від 32 до 512 нейронів з кроком збільшення нейронів на 32, функція активації повнозв'язних шарів може бути ReLU, Sigmoid, Tanh, ELU, SELU. Для кожного повнозв'язного шару додається випадкове відключення зі значенням 0,5. Після цього додано повнозв'язний шар з шести нейронів і для нього вказано функцію активації SoftMax. Оптимізатором може бути Adam, RMSprop та SGD. Метрики, які обчислюватимуться, – точність (Accuracy), Precision та Recall.

Табл. 1. Точність моделей за ручного підбору гіперпараметрів / Accuracy of models with manual selection of hyperparameters

№	Назва архітектури	Точність навчання	Точність тестування
1	VGG16	0,9646	0,8667
2	VGG19	0,8534	0,6833
3	InceptionV3	0,7933	0,6333
4	Xception	0,8436	0,4667
5	ResNet50	0,3259	0,1833

Табл. 2. Набори гіперпараметрів, отриманих за допомогою Keras-Tuner / Hyperparameter sets obtained using Keras-Tuner

№	Функція активації	Кількість шарів	Кількість нейронів у шарі	Оптимізатор	Значення функції втрат на валідаційному наборі під час навчання
1	Sigmoid	1	384	RMSprop	0,36
2	ELU	2	1 – 512, 2 – 320	SGD	0,39
3	ReLU	1	192	Adam	0,42

Keras-Tuner здійснив десять тренувань мережі, гіперпараметри для якої вибрано за алгоритмом Bayesian Optimization, із зазначених у функції побудови моделі. Оптимізацію здійснено за мінімізацією значення функції втрат на валідаційному наборі. З-поміж десяти варіантів у табл. 2 наведено три найкращі набори гіперпараметрів.

За набором гіперпараметрів з найнижчим значенням функції втрат побудовано модель. Завантажено ваги мережі VGG16 для набору даних ImageNet, відкинуто повнозв'язні шари для класифікації, вимкнено навчання для всіх елементів мережі. Засобами бібліотеки TensorFlow створено послідовну мережу, першим елементом якої є частина мережі VGG16, що відповідає за визначення характерних ознак у зображенні. Додано шар розгладжування, повнозв'язний шар, що містить 384 нейрони, функцію активації Sigmoid. Формула (2) описує функції активації Sigmoid:

$$s(x) = \frac{1}{\left(1 + e^{-x}\right)}, \qquad (2)$$

де x – вхідне значення; e – стала, яка дорівнює 2,71.

Додано випадкове вимкнення зі значенням 0,5, а також повнозв'язний шар з шести нейронів із функцією активації SoftMax. Задано функцію втрат "категоріальна крос-ентропія", оптимізатор RMSprop зі швидкістю навчання 0,001. Задано кількість епох, яка становить 50. Застосовано засіб бібліотеки Tensorflow з назвою ModelCheckpoint, який дає змогу зберегти найкращу модель за вказаною метрикою. Такою метрикою є функція втрат на валідаційному наборі під час навчання.

У табл. 3 наведено метрики моделі, побудованої з гіперпараметрами Keras-Tuner, які забезпечили найвищий результат.

Табл. 3. Метрики моделі, побудованої за гіперпараметрами Keras-Tuner / Metrics of the model built by Keras-Tuner hyperparameters

Вибірка	Втрати	Точність	Прогностична значущість позитивного результату	Повнота
Тренувальна	0,0755	0,9917	0,9916	0,9812
Валідаційна	0,2432	0,9375	0,9355	0,9062
Тестова	0,3711	0,9333	0,9310	0,9000

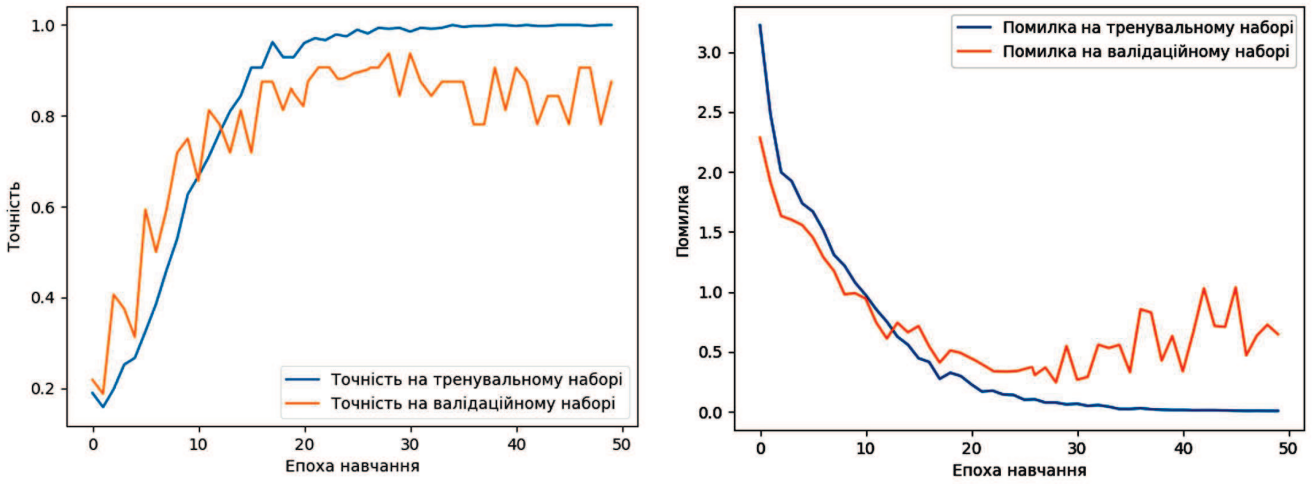

Рис. 4. Графіки точності та функції помилки для моделі, побудованої за гіперпараметрами Keras-Tuner / Plots of accuracy and error functions for the Keras-Tuner hyperparameterized model

На рис. 4 подано графіки точності та функції помилки для моделі, побудованої за гіперпараметрами Keras-Tuner. Видно, що після 30-ї епохи починається процес перенавчання, а вибрана модель отримана внаслідок епохи 29.

На наступному етапі використано тонке налаштування, яке полягає у "розмороженні" шарів п'ятого блока нейронної мережі VGG16, які використовують для визначення характерних ознак зображення. "Розмороження" шарів дасть змогу виділяти характерні ознаки об'єктів, які входять саме у застосований набір даних. Тонке налаштування можна використати, якщо уже натреновано частину мережі, яка відповідає за класифікацію. Без навчання у частині, яка відповідає за класифікацію, початково ваги встановлюють випадково. Звідси випливає, що на початку навчання гарантовано буде велика похибка, а це свідчить про великий градієнт зміни ваг. Згідно з алгоритмом зворотного поширення похибки, сигнал від частини класифікатора дійде до згорткових шарів і, відповідно до цього, у згорткових шарах, які "розморожено", істотно змінюватимуться ваги, а це знизить точність попередньо навченої мережі, яку застосовуємо за методом перенесення навчання. Тому на наступному етапі потрібно працювати із натренованою моделлю, гіперпараметри для якої отримано від Keras-Tuner. Засобами TensorFlow натреновану модель завантажено в пам'ять. Для шару, що містить згорткову частину моделі VGG16, описано цикл, який "розморожує" шари п'ятого блока, а саме три згорткові шари та один шар пулінгу. Задано функцію втрат "категоріальна крос-ентропія", оптимізатор RMSprop зі швидкістю навчання

0.00001. Задано кількість епох, яка становить 50. Застосовано засіб бібліотеки Tensorflow з назвою Model Checkpoint, який дає змогу зберегти найкращу модель за вказаною метрикою; такою метрикою є функція втрат на валідаційному наборі під час навчання.

На рис. 5 зображено графіки точності та функції помилки для моделі із “розмороженням” шарів п’ятого блока згорткової частини шару VGG16. З них випливає, що після п’ятої епохи починається процес перенавчання, а вибрану модель отримано внаслідок епохи 5.

Табл. 4. Метрики моделі із “розмороженням” шарів п’ятого блока згорткової частини шару VGG16 / Metrics of the model with “unfreezing” of the layers of the 5th block of the convolutional part of the VGG16 layer

Вибірка	Втрати	Точність	Прогностична значущість позитивного результату	Повнота
Тренувальна	0,0404	0,9999	0,9999	0,9999
Валідаційна	0,1000	0,9999	0,9999	0,9688
Тестова	0,2078	0,9500	0,9655	0,9333

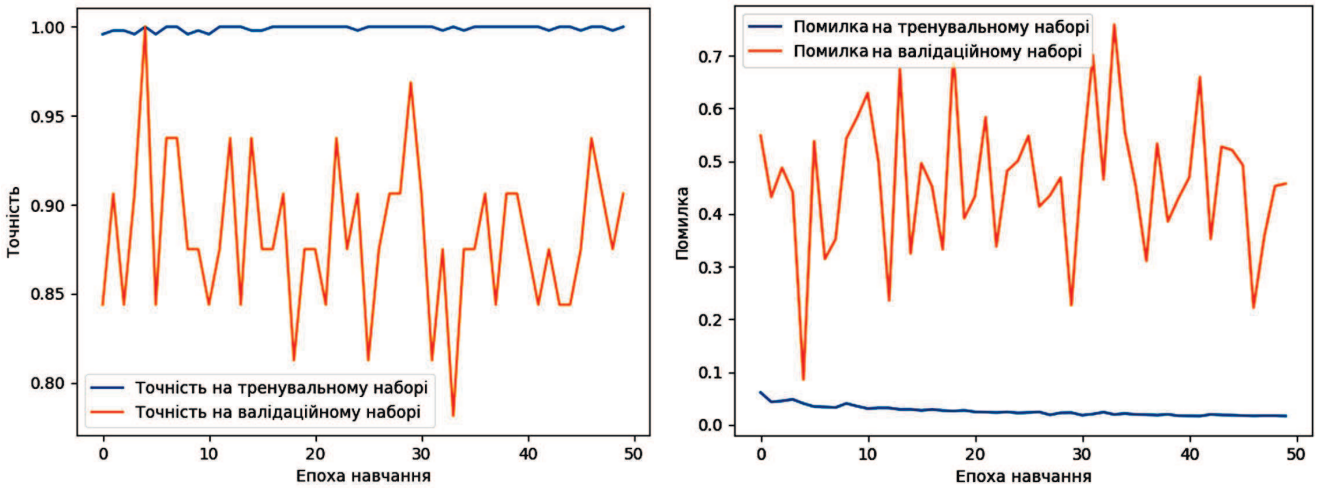

Рис. 5. Графіки точності та функції помилки для моделі із “розмороженням” шарів п’ятого блока згорткової частини шару VGG16 / Plots of accuracy and error functions for the model with “unfreezing” of the layers of the 5th block of the convolutional part of the VGG16 layer

Рис. 6. Тестування користувацького інтерфейсу та натренованої моделі через подавання зображення із типом патерну “в клітинку” / Testing the user interface and the trained model by sending an image with a “checkered” pattern

Рис. 7. Тестування користувацького інтерфейсу та натренованої моделі через подавання зображення з типом патерну "в смужку" / Testing the user interface and the trained model by sending an image with a "striped" pattern

Навчену нейронну мережу апробовано на нових зображеннях, наведено приклади для двох патернів. Спершу використано тестове зображення із патерном "у клітинку" та отримано правильний результат класифікації з точністю 98,64 %, що зображено на рис. 6.

Окрім того, використано тестове зображення з патерном "у смужку" та отримано правильний результат класифікації з точністю 99,41 %, що зображено на рис. 7.

Обговорення результатів дослідження. З використанням методу перенесення навчання отримано згорткову нейронну мережу для класифікації патернів елементів одягу з точністю тренування – 99,17 %, точністю валідації – 93,75 % та точністю тестування – 93,33 %.

З використанням підходу тонкого налаштування одержано остаточну версію згорткової нейронної мережі для класифікації патернів елементів одягу з точністю тренування – 99,99 %, точністю валідації – 99,99 % та точністю тестування – 95 %.

Наукова новизна отриманих результатів дослідження – на основі машинного навчання з використанням методів перенесення навчання та підходів тонкого налаштування розвинено надалі модель класифікації патернів елементів одягу.

Практична значущість результатів дослідження полягає у застосуванні засобів класифікації патернів елементів одягу із використанням машинного навчання, оскільки розроблено програмне забезпечення системи, побудовано архітектуру системи та створено математичне забезпечення. Це підтверджують наведені результати дослідження.

Висновки / Conclusions

Досліджено та реалізовано моделі й засоби для підвищення ефективності класифікації патернів елементів одягу з використанням машинного навчання. Результати дослідження дають користувачеві змогу визначити патерн елементу одягу на зображенні.

Дослідження підтвердило, що для розробленої моделі досягнуто підвищення ефективності класифікації патернів елементів одягу.

Проєкт можна покращити та розширити такими способами:

- збільшувати обсяг та різноманітність набору даних, що підвищить робастність штучної нейронної мережі;
- застосувати аугментацію для збільшення набору даних;
- додавати інші класи патернів;
- продовжувати оптимізацію штучної нейронної мережі, підбираючи гіперпараметри та вносячи зміни в архітектуру; оптимізувати швидкість здійснення передбачення;
- організувати систему анонімізації людей на зображеннях;
- розробити засоби експорту результатів класифікації патернів елементів одягу.

Результати цього дослідження можна використовувати для інтеграції у комплексні системи комп'ютерного зору, застосовувані для оброблення зображень одягу.

References

1. Zakaryan, V. (2022, June 17). AI Clothing Detection: Use Cases for Fashion and E-commerce. Retrieved from: https://postindustria.com/ai-clothing-detection-use-cases-for-fashion-and-e-commerce/
2. Wang, H. (2018, July 30). Rule-free sewing pattern adjustment with precision and efficiency. *ACM Transactions on Graphics, 37*, 1–13. https://doi.org/10.1145/3197517.3201320
3. Liu, L., Xu, X., Lin, Z., Liang, J., & Yan, S. (2023, December). Towards Garment Sewing Pattern Reconstruction from a Single Image. *ACM Transactions on Graphics, 42*(6), Article 200, 15 p. https://doi.org/10.1145/3618319
4. Shen, Y., Liang, J., & Lin, M. (2020). GAN-Based Garment Generation Using Sewing Pattern Images. https://doi.org/10.1007/978-3-030-58523-5_14
5. Mehta, K., & Panda, S. P. (2022). Sentiment Analysis on E-Commerce Apparels using Convolutional Neural Network.

International Journal of Computing, 21(2), 234–241. https://doi.org/10.47839/ijc.21.2.2592
6. El-Nahas, M. M. A. (2021). The Impact of Augmented Reality on Fashion and Textile Design Education. *International Design Journal, 11*(6), Article 3. https://doi.org/10.21608/idj.2021.204886
7. Jadhav, O., Patil, A., Sam, J., & Kiruthika, M. (2021). Virtual Dressing using Augmented Reality. *ITM Web of Conferences, 40*, 03028. https://doi.org/10.1051/itmconf/20214003028
8. Hussain, M. A. I., Khan, B., Wang, Z., & Ding, S. (2020). Woven Fabric Pattern Recognition and Classification Based on Deep Convolutional Neural Networks. *Electronics, 9*, 1048. https://doi.org/10.3390/electronics9061048
9. Birjuk, A. (2023, September 25). Unseen and unheard: The power of anti-surveillance clothing. Retrieved from: https://medium.com/@alinabirjuk/unseen-and-unheard-the-power-of-anti-surveillance-clothing-156570fefb0 e
10. Rajasekhar, K. E. (2020, August 21). Convolutional Neural Network. Retrieved from: https://developersbreach.com/convolution-neural-network-deep-learning/
11. JetBtains (n. d.). PyCharm – The Python IDE for Professional Developers. Retrieved from: https://www.jetbrains.com/pycharm/
12. Jupyter (n. d.). The Jupyter Notebook is a web-based interactive computing platform. Retrieved from: https://jupyter.org/
13. TensorFlow (n. d.). Create production-grade machine learning models with TensorFlow. Retrieved from: https://www.tensorflow.org/
14. Keras (n. d.). Keras – deep learning API. Retrieved from: https://keras.io
15. Nvidia (n. d.). CUDA Toolkit. Retrieved from: https://developer.nvidia.com/cuda-toolkit
16. Nvidia (n. d.). CUDA Deep Neural Network library. Retrieved from: https://developer.nvidia.com/cudnn
17. Loewen, C., Wojciakowski, M., & others. (2023, August 28). Windows Subsystem for Linux. Retrieved from: https://learn.microsoft.com/en-gb/windows/wsl/install
18. Sebastián Ramírez (n. d.). FastAPI framework, high performance, easy to learn, fast to code, ready for production. Retrieved from: https://fastapi.tiangolo.com/
19. Gatis, D. (2020). Rembg – a tool to remove images background. Retrieved from: 20. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90-95. https://doi.org/10.1109/MCSE.2007.55
21. Teslyuk, V. M., & Ivasiv, S. S. (2023). System for recognizing clothing items and their colors in an image. *Ukrainian Journal of Information Technology, 5*(2), 25-32. https://doi.org/10.23939/ujit2023.02.025
22. Stearns, L., Findlater, L., & Froehlich, J. E. (2018). Applying Transfer Learning to Recognize Clothing Patterns Using a Finger-Mounted Camera. In Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '18) (pp. 349-351). Association for Computing Machinery. https://doi.org/10.1145/3234695.3241015
23. Dey, E., Tawhid, M. N. A., & Shoyaib, M. (2015). An Automated System for Garment Texture Design Class Identification. *Computers,* 4, 265–282. https://doi.org/10.3390/computers4030265
24. Chen, H., Gallagher, A., & Girod, B. (2012). Describing Clothing by Semantic Attributes. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 609–623). https://doi.org/10.1007/978-3-642-33712-3_44
25. Manfredi, M., Grana, C., Calderara, S., & et al. (2014). A complete system for garment segmentation and color classification. *Machine Vision and Applications, 25*, 955-969. https://doi.org/10.1007/s00138-013-0580-3
26. Islam, S. S., Dey, E. K., Tawhid, M. N. A., & Hossain, B. M. M. (2017). A CNN Based Approach for Garments Texture Design Classification. *Advances in Technology Innovation, 2*(4), 119-125. Retrieved from: https://ojs.imeti.org/index.php/AITI/article/view/366
27. Datagen (n. d.). Understanding VGG16: Concepts, Architecture, and Performance. Retrieved from: https://datagen.tech/guides/computer-vision/vgg16/
28. Dr. Info Sec. (2021, March 6). VGG-19 Convolutional Neural Network. Retrieved from: https://blog.techcraft.org/vgg-19-convolutional-neural-network/
29. Narein, A. T. (2021). Inception V3 Model Architecture. Retrieved from: https://iq.opengenus.org/inception-v3-model-architecture/#google_vignette
30. Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1800–1807). Honolulu, HI, USA. https://doi.org/10.1109/CVPR.2017.195
31. Datagen (n. d.). ResNet-50: The Basics and a Quick Tutorial. Retrieved from: https://datagen.tech/guides/computer-vision/resnet-50/
32. O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & others. (2019). KerasTuner. Retrieved from: https://github.com/keras-team/keras-tuner

V. M. Teslyuk, S. S. Ivasiv
Lviv Polytechnic National University, Lviv, Ukraine

MODELS AND MEANS OF CLOTHING ELEMENTS PATTERNS CLASSIFICATION USING MACHINE LEARNING

The task of pattern classification remains relevant in the fields of trends, style, fashion, personalization, manufacturing, and design. Research aimed at the design and development of models and means of classification of patterns of clothing elements using machine learning is highlighted. The study addresses a pertinent issue in computer vision, namely: increasing the efficiency of classification of patterns of clothing elements. The research was conducted with a proprietary dataset comprising 600 images. The following patterns are defined for classification: "checkered", "dotted", "vegetation/floral", "print", "solid", "striped". A convolutional neural network was developed using the Python programming language and deep learning frameworks Keras and TensorFlow. The scalable Keras-Tuner framework was used to optimize the hyperparameters of the developed network. The structure of the convolutional neural network includes an input layer, a feature extraction part, and a pattern type determination part. The architecture of the applied convolutional neural network is described. The CUDA Toolkit, the cuDNN library and the WSL layer are applied to train a convolutional neural network using a GPU, significantly speeding up the training process. Metrics including accuracy, precision, and recall were used to evaluate the developed convolutional neural network. The web application is developed in the Python programming lan-

guage with the FastAPI framework. The web application has a described API for interacting with a convolutional neural network, and uses the Pillow (PIL) libraries for working with images and Rembg for image background removal. The user interface is developed in the JavaScript programming language with HTML, CSS and the React framework. The user interface is presented as an intuitive tool for interacting with the system. The developed software uses the modular principle, which allows for rapid modernization of the software. As a result of applying transfer learning, a testing accuracy of 93.33 % was achieved, and with fine-tuning, the final version of the convolutional neural network for the classification of patterns of clothing elements with a test accuracy of 95 % was obtained. The trained neural network was tested on new images of the specified types of patterns, examples for two patterns are given.

Ключові слова: Visual Geometry Group 16 (VGG16), patterns classification, fine-tuning, transfer learning, convolutional neural network.

Інформація про авторів:

Теслюк Василь Миколайович, д-р техн. наук, професор, завідувач кафедри автоматизованих систем управління. **Email:** vasyl.m.teslyuk@lpnu.ua; https://orcid.org/0000-0002-5974-9310

Івасів Станіслав Степанович, магістр, кафедра автоматизованих систем управління. **Email:** stanislav.ivasiv.mknus.2022@lpnu.ua; https://orcid.org/0009-0007-3406-7376

Цитування за ДСТУ: Теслюк В. М., Івасів С. С. Моделі та засоби класифікації патернів елементів одягу з використанням машинного навчання. *Український журнал інформаційних технологій*. 2024, т. 6, № 1. С. 37–47.

Citation APA: Teslyuk, V. M., & Ivasiv, S. S. (2024). Models and means of clothing elements patterns classification using machine learning. *Ukrainian Journal of Information Technology, 6*(1), 37–47. https://doi.org/10.23939/ujit2024.01.037