ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.9+616.24

М. В. Барабан, С. В. Барабан, В. В. Гармаш

РОЗРОБКА ПРОГРЕСИВНОГО ВЕБ-ДОДАТКУ ЗІ ЗГОРТКОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Вінницький національний технічний університет, Вінниця

Анотація. В даній роботі проаналізовано технології створення веб-додатків, внаслідок чого обрано Progressive Web App як найбільш відповідну для розв'язання поставлених задач. Досліджено особливості використання інтелектуальних технологій для проблеми розпізнавання зображень. Акцент зроблено на методах, які використовують бібліотеку для нейронних мереж TensorFlow. Створено власну модель згорткової нейронної мережі для розпізнавання зображень. Для навчання моделі було обрано набір даних «The Quick, Draw! Dataset» від корпорації Google. Визначено, що прогресивний веб-додаток надає можливість швидше за аналоги надавати результуючу вибірку користувачеві. Проілюстровано результат порівняння швидкодії розробленого та додатків-аналогів.
Ключові слова: Progressive Web App, веб-додаток, згорткова нейронна мережа, розпізнавання зображень.
Аннотация. В данной работе проанализированы технологии создания веб-приложений, в результате чего избран Progressive Web App как наиболее подходящая для решения поставленных задач. Исследовано особенности использования интеллектуальных технологий для проблемы распознавания изображений. Акцент сделан на методах, которые используют библиотеку для нейронных сетей TensorFlow. Создана собственная модель сверточной нейронной сети для распознавания изображений. Для обучения модели был выбран набор данных «The Quick, Draw! Dataset» от корпорации Google. Определено, что прогрессивноей веб-приложение предоставляет возможность быстрее аналогов предоставлять результирующую выборку пользователю. Проиллюстрировано результат сравнения быстродействия разработанного и приложений-аналогов.
Ключевые слова: Progressive Web App, веб-приложение, сверточная нейронная сеть, распознавание изображений.
Abstract. In this paper the technologies for creating web applications are analyzed, in result Progressive Web App as the most suitable for solving the tasks is selected. The peculiarities of the use of intelligent technologies for the problem of image recognition are investigated. Emphasis is placed on methods that use the TensorFlow neural network library. The own model of convolutional neural network for image recognition has been created. The dataset «The Quick, Draw! Dataset» from Google is selected for model training. It has been determined that a progressive web application provides the ability to provide the resulting sample to the user faster than analogues. The result of comparing the speed of the developed and analog applications is illustrated.
Key words: Progressive Web App, web application, convolutional neural network, image recognition.
DOI: https://doi.org/10.31649/1999-9941-2021-50-1-7-14.

Вступ

Технологічні інновації завжди впливають на розробку продуктів і послуг. В останні десятоліття спостерігається безпрецедентне зростання в області Web. Завдяки впровадженню нових і вдосконаленню існуючих технологій у світі Інтернету, те, що раніше не було можливим, почало існувати. Використання технологій можна знайти в усіх областях, від освіти до охорони здоров'я, досліджень і сільського господарства. Більшість компаній вже прийняли Інтернет як свою бізнес-платформу, і тенденція зростає. Таким чином, існує потреба в розробці швидких, привабливих і надійних додатків.

Native і Web – два основних типи додатків. Native додатки платформно залежні (Android, IOS, Windows), вони побудовані за допомогою спеціальних мов програмування та комплектів розробки програмного забезпечення, тоді як Web додатки є незалежними від платформи веб-сайтами, які багато в чому виглядають і відчуваються як Native додатки [1]. Web додатки запускаються браузером і зазвичай написані на HTML5.

Протягом останніх кількох років використання нативних мобільних додатків показало зростання у порівнянні з мобільними веб-додатками [1]. Веб-додатки програють нативним додаткам з точки зору продуктивності, надійності та взаємодії. Підприємства та розробники часто бачать необхідність розробки власних мобільних додатків для подолання обмежень, які веб-сайт накладає на мобільні пристрої. Однак у нативних додатків є свої недоліки. Коли йдеться про доступність для користувача, нативні додатки програють веб-додаткам [2]. Нативні додатки витрачають більше ресурсів і часу для розробки та підтримки. Крім того, публікація нативного додатку – це досить складний процес. Також для використання нативного додатку користувачам необхідно пройти багато кроків, які включають в себе реєстрацію у відповідному магазині, перевірку пам'яті, завантаження, встановлення та, нарешті, відкриття для його використання. Дослідження показало, що в середньому додаток втрачає близько 20% своїх користувачів на кожному кроці між першим контактом користувача і початком його використання [3]. Багато користувачів також вважають цей процес складним і затратним. Це величезний недолік як для компаній, так і для розробників.

Веб і нативні платформи мають свої власні недоліки, тому існувала потреба в платформі, яка може поєднувати можливості та досвід нативних додатків з доступністю вебу. Progressive Web App об'єднує в собі все найкраще з світу веб та світу нативних додатків. Вони корисні для користувачів з першого

відкриття сторінки в браузері, не потребують інсталяції. По мірі того як користувач крок за кроком будує відносини з додатком, він стає все більш та більш корисним. Progressive Web App швидко завантажується, навіть у нестабільних мережах, відправляє пуш-повідомлення, має кнопку на домашньому екрані і дарує повноцінний повноекранний досвід [4].

Популярність Progressive Web App швидко зростає не лише серед користувачів а й серед дослідників. Progressive Web App дає можливість швидко розгорнути, перевірити та протестувати різноманітні моделі нейронних мереж. Сьогодні однією з провідних тем досліджень є розпізнавання зображень за допомогою нейронних мереж, що підтверджують наукові праці таких авторів як P. Haffner, Y. Bengio, R. Fergus, J. Johnson [5]. Частиною даної теми є розпізнавання зображень. Зображення простіше всього створити за допомогою смартфону чи планшету. Тому, актуальною є задача створення прогресивного веб-додатку, адже дана технологія є зручною та працює на усіх можливих платформах: у веб-браузері, на IOS та Android [6].

Актуальність

Полягає в тому, що отримав подальший розвиток нейромережевий метод розпізнавання зображень шляхом використання згорткових нейронних мереж для аналізу зображень, що підвищило швидкодію та точність розпізнавання.

Практичне значення результатів роботи полягає в розробці алгоритмічних та програмних засобів, які реалізують веб-додаток для розпізнавання зображень, що може працювати на усіх доступних платформах: у веб-браузері, на IOS та Android.

Методи дослідження ґрунтуються на теорії обчислювального інтелекту, а саме, на методах теорії штучних нейронних мереж, які дозволили синтезувати нейромережеві моделі для подальшого використання у веб-додатку через бібліотеку tensorflow; методи теорії розпізнавання образів, на основі яких були синтезовані моделі аналізованих об'єктів, що використовуються для розпізнавання зображень; експериментальне дослідження для перевірки достовірності отриманих результатів.

Мета

Метою статті є підвищення швидкодії та точності розпізнавання зображень на основі загорткової нейронної мережі за рахунок застосування технології Progressive Web App.

Задачі

1. Проаналізувати особливості прогресивних веб додатків.
2. Дослідити особливості використання прогресивних технологій для проблеми розпізнавання зображень.
3. Розробити програмні засоби у вигляді прогресивного веб-додатку та оцінити ефективність роботи запропонованої технології.

Розв’язання задач

Основним призначенням програми є дослідження можливості розпізнавання зображень автономно за рахунок застосування технології Progressive Web App.

В програмі реалізовані наступні функції:

1) Можливість створення зображення в додатку.

2) Розпізнавання створеного зображення.

3) Розпізнавання зображення автономно.

4) Відображення результатів. Виведення можливих результатів у вигляді таблиці співпадінь.

Базовим будівельним блоком нейронної мережі є шар. Шари витягають образи з даних, які в них подаються, як зображено на рис. 1.

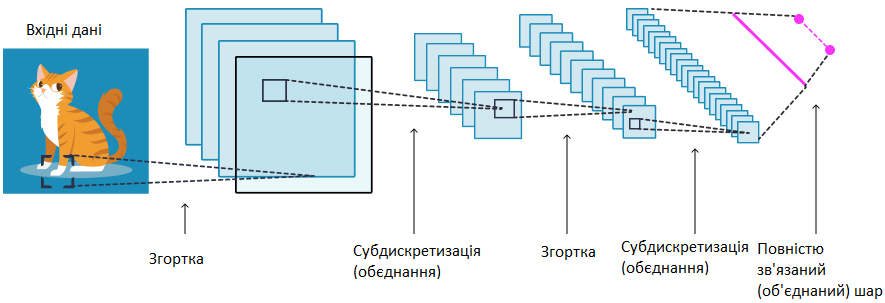

Рисунок 1 – Створення шарів нейронної мережі

Розроблено модель нейронної мережі для розпізнавання зображення. Згорткова нейронна мережа має три важливі будівельні блоки:

- згорнутий шар, який витягує ознаки із зображення або його частин;
- рівень субдискретизації (об'єднання), який зменшує розмірність кожної функції, щоб зосередитись на найважливіших елементах (зазвичай існує кілька кроків згортки та об'єднання);
- повністю зв’язаний шар, який приймає вирівняну форму ознак, визначених у попередніх шарах, і використовує їх для прогнозування зображення.

Велика частина глибокого навчання складається із з'єднання в послідовність простих шарів. Більшість шарів, таких як tf.layers.conv2d, мають параметри, які налаштовуються під час навчання. Наприклад, шар tf.layers.conv2d створює ядро згортки, яке перехресно корелюється із вхідним шаром для отримання тензора вихідних даних.

Шар згортки витягує елементи з вихідного зображення, «скануючи» зображення за допомогою фільтра, наприклад, 5 × 5 пікселів. Для кожної області 5 × 5 пікселів у зображенні операція згортки обчислює точкові добутки між значеннями пікселів зображення та вагами, визначеними у фільтрі, як хображено на рис. 2.

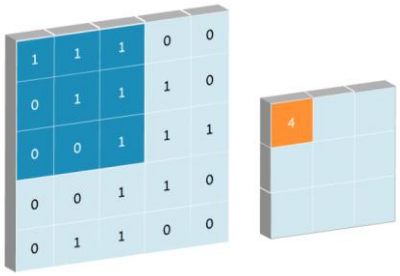

Рисунок 2 – Шар згортки (conv2d)

Двовимірний рівень згортки означає, що вхідні дані операції згортки є тривимірними. «2D» у «2D-згортці» відноситься до руху фільтра, який проходить зображення у двох вимірах. Наприклад, кольорове зображення, яке має значення для кожного пікселя у трьох шарах: червоному, синьому та зеленому. Потім фільтр запускається по зображенню тричі, по одному для кожного шару.

tf.layers.dense реалізує операцію: output = activation(dot(input, kernel) + bias) де activation – це елементна функція активації, передана як аргумент активації, kernel – це матриця ваг, створена шаром, а bias – вектор зміщення, створений за шаром (застосовується лише якщо use_bias має значення True).

tf.layers.dropout полягає у випадковому встановленні частоти частки вхідних одиниць до 0 під час кожного оновлення під час навчання, що допомагає запобігти перенапруженню. Одиниці, що зберігаються, масштабуються на 1 / (1 – ставка), так що їхня сума не змінюється під час навчання та часу висновку.

Метою об’єднання шарів у згорткових нейронних мережах зменшення розмірності вхідного зображення. Об’єднання шарів робить виявлення функцій незалежним від шуму та незначних змін, таких як обертання або нахил зображення.

Об’єднання базується на концепції «розсувного вікна». Він застосовує статистичну функцію до значень у межах певного розміру вікна, відомого як фільтр згортки або ядро.

Шар max_pooling (об'єднання) приймає максимальне значення у фільтрі згортки. На рис. 3 показано max_pooling в дії.

Рисунок 3 – Шар max_pooling

На наведеній вище схемі кольорові рамки представляють функцію максимального об’єднання з розсувним вікном (розмір фільтра) 2 × 2. Просте максимальне значення береться з кожного вікна на вихідну карту ознак. Іншими словами, максимальне значення в синьому полі – 3. Це значення буде представляти чотири вузли в синьому полі. Те саме стосується зеленого та червоного поля.

Об’єднання невеликих зображень із невеликою кількістю ознак може допомогти запобігти перенавчанню. На великих зображеннях об’єднання може допомогти уникнути величезної кількості вимірів. Складність оптимізації зростає в геометричній прогресії із зростанням розмірності. Таким чином вийде надзвичайно повільна конвергенція, яка може спричинити перенавчання. Не має значення, чи відображається значення 4 у комірці 4 x 2 або комірці 3 x1, ми все одно отримуємо те саме максимальне значення з цієї комірки після операції максимального об’єднання, як зображено на рис. 4. Саме цей процес надає згортковій нейронній мережі можливість «просторової варіативності».

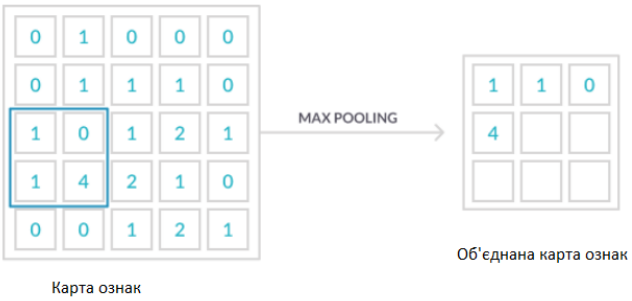

Рисунок 4 – Просторова варіативність згорткових нейронних мереж

Перш ніж модель буде готова для навчання, потрібно вказати ще кілька параметрів: функцію втрат, оптимізатор та метрики. Вони додаються на етапі компіляції моделі.

Функція втрат (Loss function) – вимірює точність моделі під час навчання. Необхідно мінімізувати цю функцію щоб «направити» модель в правильному напрямку.

Оптимізатор (Optimizer) – показує яким чином оновлюється модель на основі вхідних даних і функції втрат.

Метрики (Metrics) – використовуються для моніторингу тренування і тестування моделі.

Навчання моделі нейронної мережі вимагає виконання наступних кроків:

1. Потрібно подати тренувальні дані в модель.
2. Модель вчиться асоціювати зображення з правильними класами.
3. Модель робить прогнози для перевірочних даних.

Для початку навчання, потрібно викликати метод model.fit, який називається так, оскільки «тренує (fits)» модель на тренувальних даних. В процесі навчання моделі відображаються метрики втрати (loss) і точності (accuracy), як зображено на рис. 5. Ця модель досягає на тренувальних даних точності приблизно 0.88 (88%).

```
Epoch 1/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.4996 - accuracy: 0.8239
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3764 - accuracy: 0.8646
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3342 - accuracy: 0.8781
Epoch 4/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.3124 - accuracy: 0.8861
Epoch 5/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2921 - accuracy: 0.8926
Epoch 6/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2794 - accuracy: 0.8964
```

Рисунок 5 – Тренування моделі

Далі необхідно порівняти яку точність модель покаже на перевірочному наборі даних, рис. 6.

```
313/313 - 0s - loss: 0.3253 - accuracy: 0.8854
```

Рисунок 6 – Точність моделі на перевірочному наборі даних

Отримана на перевірочному сеті точність виявилася трохи нижче, ніж на тренувальному. Цей розрив між точністю на тренуванні і тесті є прикладом перенавчання (overfitting). Перенавчання виникає, коли модель машинного навчання показує на нових даних найгірший результат, ніж на тих, на яких вона навчалася. Після навчання моделі та перевірки метрик модель готова.

Для правильного функціонування додатку розроблено архітектуру системи та його логіку. Попередньо навчена модель TensorFlow перетворюється на веб-формат TensorFlow.js та інтегрується з додатком. Конвертер TensorFlow.js складається з двох компонентів: утиліта командного рядка, що перетворює модель TensorFlow для використання в TensorFlow.js та API для завантаження і виконання моделі в браузері за допомогою TensorFlow.js.

В процесі перетворення переглядається граф моделі і перевіряється, чи підтримується кожна операція TensorFlow.js. Якщо це так, записуємо графік у форматі, який може використовувати браузер. Щоб оптимізувати модель для обслуговування в Інтернеті, сегментовано ваги в файли розміром 4 МБ – таким чином вони можуть бути кешованими браузерами. Якщо під час оптимізації виявляється, що операція не підтримується, то процес завершується помилкою, і система виводить ім'я операції для користувача. Після перетворення моделі у веб-формат TensorFlow.js її можна використовувати за призначенням. Дана модель набуває вигляду таблиці з даними та зберігається у форматі json.

Як тільки користувач запускає прогресивний веб-додаток, активи програми та файли моделі TensorFlow.js завантажуються з Інтернету. Активи та модель зберігаються локально за допомогою кешу браузера. Користувач в спеціальному вікні робить зображення за допомогою комп’ютерної миші або тач скріну. Зображення розпізнається та даються найкращі прогнози. Схема роботи додатку зображена на рис. 7.

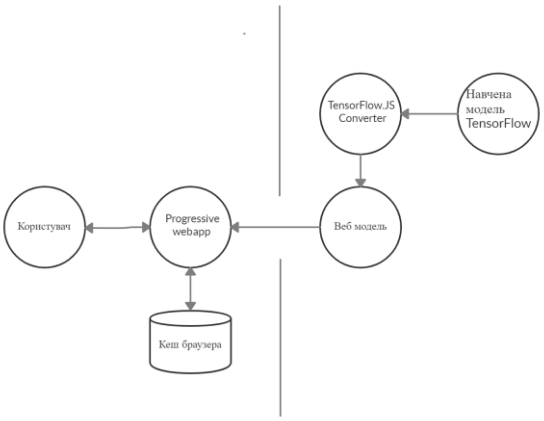

Рисунок 7 – Схема роботи додатку

Схема роботи системи використовує процедурний підхід. Дана схема зображено на рис. 8.

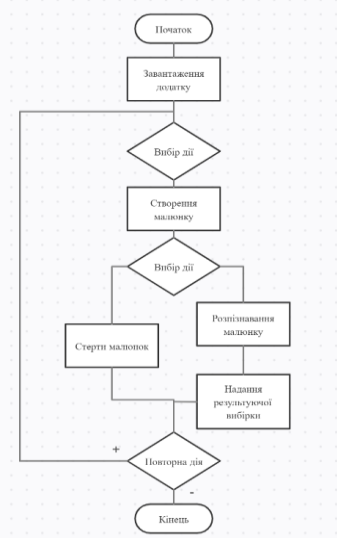

Рисунок 8 – Схема роботи системи використовуючи процедурний підхід

Для проведення аналізу та результатів дослідження, порівняємо розроблений додаток із вже існуючими на ринку. Одними із найпопулярніших на даний момент є Quickdraw, Autodraw, та Draw it. Розглянемо порівняльну таблицю даних програм (табл. 1), щоб дізнатись перевеги розробленого PWA додатку.

Таблиця 1 – Порівняльний аналіз програм аналогів

Додаток/Особливості	Quickdraw	Autodraw	Draw it	PWA Draw
Надає доступ до тачскріну	Так	Так	Так	Так
Працює без підключення до інтернету	Ні	Ні	Так	Так
Використовує нейронні мережі для класифікації	Так	Так	Так	Так
Працює на усіх доступних платформах: IOS, Android, браузер	Працює на всіх платформах у вигляді сайту	Працює на всіх платформах у вигляді сайту	Встановлюється як додаток на смартфон з ОС Android	Працює на усіх доступних платформах
Класифікує зображення	Так	Так	Так	Так
Підтримує технологію PWA	Ні	Ні	Ні	Так

В якості аналога для розробки було обрано Draw it. Основними недоліками аналога є: висока вартість розробки, залучення великої команди розробників для створення програми та розробка додатку під кожну платформу окремо. У розробці дана проблема вирішується використанням технології Progressive Web App. Також програмний продукт випереджає аналог за швидкістю розпізнавання зображень.

Застосування розробленої програми дозволяє підвищити швидкодію та стабільність роботи розпізнавання зображень за рахунок застосування технології Progressive Web App та TensorFlow.js.

Після порівняльної таблиці протестуємо усі додатки та заміряємо час виконання розпізнавання зображень в кожному з них. Занесемо час виконання операцій у кожному із додатків в порівняльний графік. Даний графік наведено на рис. 9. Найбільший час виконання розпізнавання зображень 0.016 припадає на додаток Draw it. Найменший час 0.007 у розробленого PWA додатку.

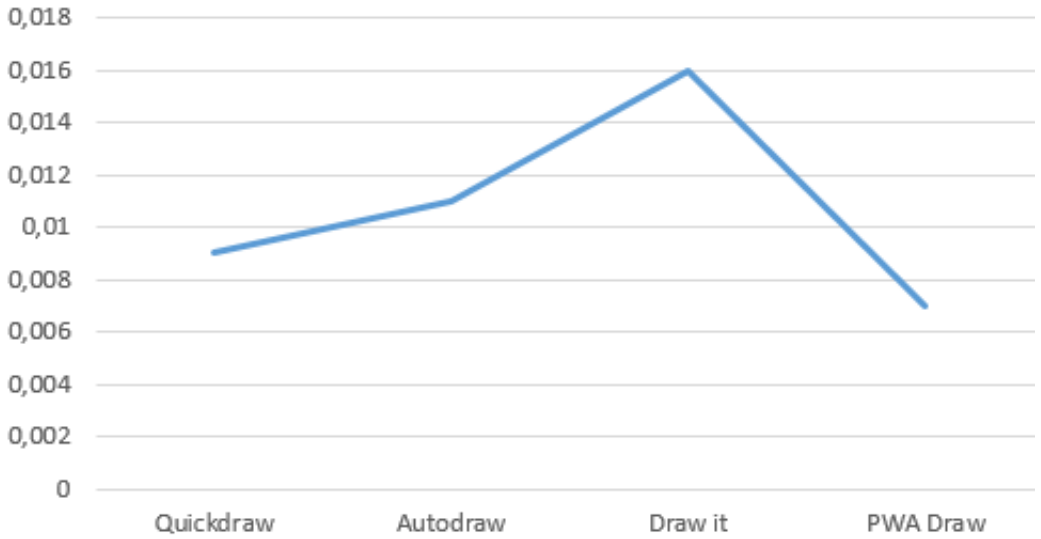

Рисунок 9 – Графік швидкості виконання операцій

З графіку можна зробити висновок, що створена програма на 0,02 секунди швидше розпізнає зображення в порівнянні з найшвидшим з представлених аналогів Quickdraw.

Висновки

1. У статті проведено аналіз технологій та підходів для створення мобільних додатків. Для створення додатку було обрано технологію Progressive Web App. Дана технологія має ряд суттєвих переваг, а саме: простоту в створенні, малу затрату ресурсів, високу швидкість роботи, підтримка різних операційних систем та можливість працювати офлайн.

2. Створено власну модель згорткової нейронної мережі для розпізнавання зображень. Для навчання моделі було обрано набір даних «The Quick, Draw! Dataset» від корпорації Google. Модель показала високу точність та швидкість розпізнавання зображень.

3. Розроблено архітектуру програми та усіх необхідних складових для конвертації моделей нейронних мереж у необхідний формат для роботи із зображеннями у веб-додатку.

4. Розроблено програмне забезпечення, в якому реалізовано усі необхідні функції, що притаманні PWA додатку. Завдяки даному програмному забезпеченню проведено розпізнавання зображень з використанням згорткової нейронної мережі і бібліотеки TensorFlow. Наведено успішні приклади розпізнавання різних зображень. Продемонстровано роботу програми на різних пристроях, а також у браузері. Визначено, що прогресивний веб-додаток покращений у багатьох параметрах і надає можливість швидше за аналоги надавати результуючу вибірку користувачеві. Проілюстровано результат порівняння швидкодії програм аналогів.

Список літератури

[1] Making Progressive Web Apps (PWAs) with React, 2019. [Online]. Available: https://alligator.io/react/react-progressive-web-apps/. Accessed on: February 01, 2021.

[2] Прогресивний веб-застосунок, 2018. [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki /Progressive_Web_Apps. Дата звернення: Лют. 01, 2021.

[3] Progressive Web, 2018. [Online]. Available: https://codelabs. developers.google.com/codelabs/your-first-pwapp-ru/index.html?index=..%2F..%2Flangru#0. Accessed on: February 01, 2021.

[4] M. D. Zeiler, R. Fergus, «Visualizing and understanding convolutional networks», in *European conference on computer vision*, Springer International Publishing, pp. 818–833, 2014.

[5] Fei-Fei Li, A. Karpathy, J. Johnson, *CS231n Convolutional Neural Networks for Visual Recognition*, 2016. [Online]. Available: https://cs231n.github.io/convolutional-networks. Accessed on: Mar. 21, 2017.

[6] Т. О. Ковтун, М. В. Барабан, В. В. Гармаш, «Прогресивний веб додаток для розпізнавання малюнків» на *XV Міжнародній науковій конференції «Контроль і управління в складних системах»*, Вінниця: ВНТУ, 8 –10 жовтня, 2020. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mccs/mccs2020/paper/view/10665/. Дата звернення: Лют. 01, 2021.

Стаття надійшла: 08.02.2021.

References

[1] Making Progressive Web Apps (PWAs) with React, 2019. [Online]. Available: https://alligator.io/react/react-progressive-web-apps/. Accessed on: February 01, 2021.

[2] Prohresyvnyi veb-zastosunok, 2018. [Elektronnyi resurs]. Rezhym dostupu: https://en.wikipedia.org/wiki/Progressive_Web_Apps. Data zvernennia: Liut. 01, 2021.

[3] Progressive Web, 2018. [Online]. Available: https://codelabs. developers.google.com/codelabs/your-first-pwapp-ru/index.html?index=..%2F..%2Flangru#0. Accessed on: February 01, 2021.

[4] M. D. Zeiler, R. Fergus, «Visualizing and understanding convolutional networks», in *European conference on computer vision*, Springer International Publishing, pp. 818–833, 2014.

[5] Fei-Fei Li, A. Karpathy, J. Johnson, *CS231n Convolutional Neural Networks for Visual Recognition*, 2016. [Online]. Available: https://cs231n.github.io/convolutional-networks. Accessed on: Mar. 21, 2017.

[6] T. O. Kovtun, M. V. Baraban, V. V. Harmash, «Prohresyvnyi veb dodatok dlia rozpiznavannia maliunkiv» na XV Mizhnarodnii naukovii konferentsii «Kontrol i upravlinnia v skladnykh systemakh», Vinnytsia: VNTU, 8–10 zhovtnia, 2020. [Elektronnyi resurs]. Rezhym dostupu: https://conferences.vntu.edu.ua/index.php/mccs/mccs2020/paper/view/10665. Data zvernennia: Liut. 01, 2021.

Відомості про авторів

Барабан Марія Володимирівна – кандидат технічних наук, доцент кафедри автоматизації та інтелектуальних інформаційних технологій.

Барабан Сергій Володимирович – кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук.

Гармаш Володимир Володимирович – кандидат технічних наук, доцент, доцент кафедри автоматизації та інтелектуальних інформаційних технологій.

М. В. Барабан, С. В. Барабан, В. В. Гармаш

РАЗРАБОТКА ПРОГРЕСИВНОГО ВЕБ-ПРИЛОЖЕНИЯ СО СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТЬЮ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ

Винницкий национальный технический университет, Винница

M. V. Baraban, S. V. Baraban, V. V. Garmash

DEVELOPMENT OF A PROGRESSIVE WEB APPLICATION WITH A CONVOLUTIONAL NEURAL NETWORK FOR IMAGE RECOGNITION

Vinnytsia National Technical University, Vinnytsia