УДК 004.9+004.8+004.93+003.24

ТОКЕНІЗАЦІЯ ВЕКТОРНОЇ ГРАФІКИ У КОНТЕКСТІ СИНТЕЗУ ТАКТИЛЬНОЇ ГРАФІКИ

Є. А. Джуринський, В. З. Маїк

Українська академія друкарства,
вул. Під Голоском, 19, Львів, 79020, Україна

Актуальним напрямом дослідження в галузі інклюзивного видавництва є розробка методів синтезу тактильної графіки, яка може використовуватися як ілюстрації у виданнях для осіб із проблемами зору. Метою синтезу тактильної графіки є автоматизація процесу підготовки інклюзивної ілюстрації до друку, що значно прискорює виробничий процес. В роботі розглянуто один з етапів вирішення завдання синтезу тактильної графіки, а саме токенізація векторної графіки, яка чудово підходить як формат представлення тактильної графіки. Токенізація має на увазі представлення вихідної інформації (в нашому випадку – векторної графіки) в іншому – більш оптимальному представленні, яке може бути використано моделлю на базі штучного інтелекту. Проаналізовано дві методики токенізації, які відрізняються архітектурою моделі штучного інтелекту: модель на базі VAE та модель на базі трансформера. Визначено переваги та недоліки обраних моделей та формалізовано принцип роботи цих рішень. З'ясовано, що розглянуті моделі нівелюють переваги векторного представлення тактильної графіки, що передусім пов'язано із малою пропускною здатністю розглянутих моделей (водночас існують інші обставини, що наводяться в основній частині). Наприкінці наведено висновок дослідження, який полягає у неможливості використання проаналізованих підходів.

Ключові слова: *інформаційна технологія, штучний інтелект, модель, методика токенізації, вимоги до ілюстрації, обробка зображень, тактильна графіка, інклюзивна ілюстрація, інклюзивна література, шрифт Брайля.*

Постановка проблеми. Одним з аспектів проблематики галузі виробництва інклюзивних видань є процес розробки тактильної графіки, який, порівняно із «традиційними» ілюстраціями, потребує більше виробничих ресурсів [1]. Синтез тактильної графіки за допомогою моделі на основі штучного інтелекту вирішує виробничі труднощі, автоматизуючи процес створення зображення.

Розглядаючи приклади тактильного зображення, можна дійти висновку, що у найкращий спосіб таке зображення представляється у форматі векторного зображення, який має низку переваг у контексті галузі інклюзивної ілюстрації:

- масштабування зображення без втрати якості, що значно спрощує сумісність векторного зображення із будь-яким виданням або тифлопристроєм;
- можливість взаємодії з окремими елементами зображення, що дозволяє ілюстратору швидко адаптувати окремі елементи під вимоги видання;

– можливість повторного використання, що дає змогу друкарні використовувати наявні ілюстрації в інших виданнях, адаптуючи їх під цільового читача й докладаючи при цьому мінімальних зусиль.

Однак, зважаючи на принцип роботи моделей на базі штучного інтелекту, векторне зображення не може бути опрацьовано у вихідному представленні (принаймні з огляду на методику, яка буде розглянута далі). Натомість векторне зображення має пройти попередню обробку, яка в нашій роботі виконується за допомогою токенізації вихідного представлення зображення.

Аналіз останніх досліджень та публікацій. Сучасні засоби штучного інтелекту [2–4], частина з яких є дифузними моделями [5–6], дозволяють синтезувати растрове зображення за текстовою підказкою [7]. Однак, згідно з дослідженням [8], наведені моделі є непридатними для використання як інформаційні системи синтезу тактильної графіки, проте методика їхньої роботи, яка, зокрема, полягає у токенізації растрового зображення за допомогою VAE [9], може бути взята за основу в цьому дослідженні.

Мета статті – дослідити застосування методики токенізації векторної графіки у контексті завдання синтезу тактильної графіки. Наприкінці дослідження необхідно визначити переваги та недоліки такого підходу, які сформують висновок про можливість застосування методики на практиці.

Виклад основного матеріалу дослідження. SVG-VAE [10] – це модель штучного інтелекту, що синтезує векторне зображення, представляючи його не у пікселях, а у дискретних інструкціях, які в результаті перетворюються на SVG шляхи, утворюючи таким чином векторне зображення. Результат моделі може бути намальований будь-якою програмою, що здатна відобразити векторну графіку. Варто зазначити, що в оригіналі [10] таке рішення не вміє синтезувати зображення з тексту, але воно може бути взято за основу.

На рисунку 1 зображено архітектуру цієї моделі, яка складається з Variational Autoencoder [9] та авторегресійного [12] SVG декодера, реалізованого за принципом Tensor2Tensor [13].

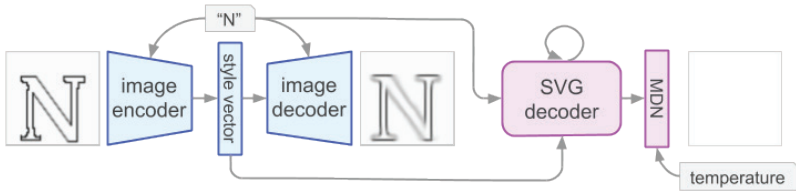

Рис. 1. Архітектура моделі SVG-VAE [10]

VAE складається зі згорткового (від англ. convolution) кодера і декодера (див. на рис. image encoder та image decoder). Їхньою проміжною ланкою є латентний простір [14] (див. на рис. style vector), елементи якого складаються із двох значень μ та σ, що означають середнє значення та стандартне відхилення багатовимірного розподілу Гауса:

$$z = \mu + \sigma, \tag{1}$$

де z – латентний вектор, μ та σ – відповідно середнє значення та стандартне відхилення багатовимірного розподілу Гауса.

При навчанні згортковий кодер на вхід отримує растрове зображення, яке в результаті відобразиться в латентний простір, утворюючи латентний вектор цього зображення $z_x = E(x)$, де x – вихідне растрове зображення, E – згортковий енкодер, z_x – відображення вихідного зображення у латентному просторі.

Згортковий декодер працює у подібний спосіб, але у протилежному напрямі: $x' = D(z_x)$, де z_x – відображення вихідного зображення у латентному просторі, D – згортковий декодер, x' – декодоване вихідне растрове зображення: $x' \approx x$. SVG декодер складається з чотирьох послідовних моделей LSTM [15] та MDN [16] наприкінці. При навчанні SVG декодер отримує три значення на вхід: попередній результат роботи MDN, дискретну мітку (англ. label) та представлення стилю у латентному просторі.

У цьому підході одним з найголовніших випробувань є оптимізація недиференційованого векторного представлення фігур, що утворюють зображення. Автори моделі вирішують цю проблему за допомогою, по-перше, проміжного представлення SVG шляхів у вигляді диференційованих дискретних інструкцій, які в кінцевому підсумку перетворюються на SVG команди та шляхи; по-друге, автори визначають такі обмеження:

– обмежена множина SVG команд, що може опрацювати модель: $c_i^j \in c^* = \{$*move to, line to, cubic bezier, close path*$\}$. При цьому функціональні можливості такого рішення категорично не зменшуються, оскільки решта команд може бути виведена із комбінації доступних команд;
– обмежена можлива загальна кількість команд, які утворюють векторне зображення: $\sum_i^{N_p} N_{Ci} \le 50$;
– необхідність попередньої нормалізації навчальних прикладів, які сортуються від найвищої лівої точки координат за годинниковою стрілкою.

У підсумку автори демонструють здібність цієї моделі в автоматичний спосіб маніпулювати властивостями шрифту, змінюючи окремі значення латентного вектора, наприклад, нахил та товщину символів та ін. Крім того, автори стверджують, що їхня модель здатна синтезувати шрифт на основі декількох прикладів символів. Однак, беручи до уваги вищенаведені обмеження, можна стверджувати, що таке рішення не є придатним у використанні в галузі інклюзивної ілюстрації.

Насамперед наявність обмежень як у множині доступних команд c^*, так і в можливій загальній кількості команд $\sum_i^{N_p} N_{Ci}$ значно ускладнює застосування цього рішення для синтезу ілюстрацій. Якщо символи шрифту здебільшого складаються з невеликої кількості шляхів і команд, то кількість команд в ілюстрації, як правило, значно вища за встановлений ліміт у 50 команд. Крім того, обмеження множини c^* збільшує фактичну загальну кількість команд, оскільки в ілюстраціях можуть використовуватися команди, що не входять до множини c^*, тому вони будуть попередньо перетворені на сукупність більш простих команд, які підтримуються рішенням. Такі обставини роблять це рішення ще більш непридатним для синтезу інклюзивних ілюстрацій.

До того ж таке рішення нівелює переваги векторного зображення з точки зору галузі інклюзивної ілюстрації, а саме можливість взаємодії з окремими елементами зображення та повторного використання наявних зображень. Вислідні шляхи будуть мати такий самий вигляд, як і нормалізовані навчальні приклади, – оптимальний для моделі, але не для ілюстратора. На практиці елемент зображення, який людина розділила б на окремі шляхи, модель буде відображати у незрозумілий спосіб, з яким неможливо проводити маніпуляції (модель навіть може представити складне векторне зображення у вигляді єдиного шляху, бо з її точки зору це оптимальне представлення). Звичайно, за потреби можливо змінити правила нормалізації навчальних даних, позбувшись небажаного ефекту, але автори моделі стверджують, що їхній метод є оптимальним, а інші способи, особливо ті, які не містять сортування, проявили себе гірше.

Перейдемо до розгляду наступного рішення – DeepSVG [11]. DeepSVG – це модель штучного інтелекту, побудована у вигляді ієрархічної синтезуючої мережі для анімації векторної графіки. На відміну від SVG-VAE, автори даної моделі фокусуються саме на векторному зображенні, а не на шрифтах, хоча й із акцентом на анімацію. Варто зазначити, що, як і у випадку із SVG-VAE, в оригіналі [11] це рішення не вміє синтезувати зображення з тексту, проте воно може бути взято за основу.

Мережа DeepSVG побудована за допомогою трансформерів [17] (англ. transformer), які призначені для обробки обмеженої послідовності дискретних даних (див. рис. 2). Їхньою головною відмінністю є те, що кожний конкретний елемент послідовності обробляється з урахуванням контексту усієї послідовності, через що трансформери часто використовують у завданнях машинного перекладу та автоматичного реферування.

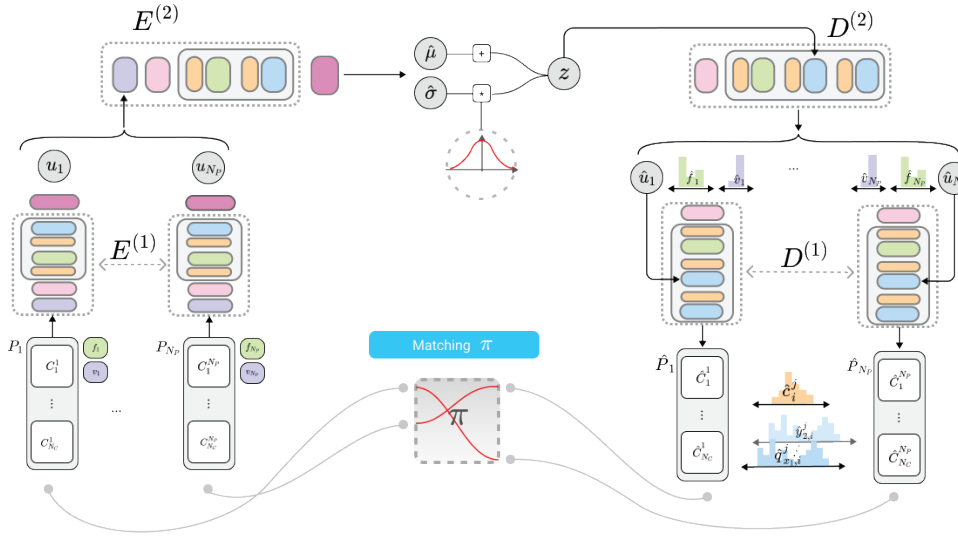

Рис. 2. Архітектура моделі DeepSVG [11]

Для того, щоб модель мала змогу розпізнавати команди, кожна C_i^j відображається у латентний простір розмірності d_E. У такий спосіб модель має можливість навчитися виконувати операції між латентними векторами, що утворюються з команд, вивчаючи при цьому складні залежності між властивостями команд: координатних значень, відносної послідовності команд тощо. Після кодування шляхів вихідного векторного зображення кодером воно перетворюється у латентний вектор $e_i^j \in \mathbb{R}^{d_E}$, який представляється як сума трьох складових:

$$e_i^j = e_{cmd,i}^j + e_{coord,i}^j + e_{ind,i}^j, \tag{2}$$

де e_i^j – латентний вектор команди C_i^j, $e_{cmd,i}^j$ – складова латентного вектора, що відображається з ідентифікатора команди c_i^j, $e_{coord,i}^j$ – складова латентного вектора, що відображається з аргументів команди X_i^{Cj}, $e_{ind,i}^j$ – складова латентного вектора, що відображається з порядкового номера команди C_i^j, тобто з j.

Кожна складова латентного вектора обчислюється за допомогою окремих матриць, що навчаються під час тренування моделі:

$$e_{cmd,i}^j = W_{cmd}\ \delta_{c_i^j}, \tag{3}$$

де $W_{cmd} \in \mathbb{R}^{d_E \times |c^*|}$ – матриця обрахунку латентної складової ідентифікаторів команд c_i^j, що здатна навчатися під час тренування моделі, $\delta_{c_i^j} \in \mathbb{R}^{|c^*|}$ – унітарний вектор, що дорівнює одиниці в єдиному елементі, який відповідає порядковому номеру ідентифікатора команди c_i^j в множині c^*, розмірності, що дорівнює загальній кількості множини доступних команд c^*.

$$e_{coord,i}^j = W_{coord}\, vec\left(W_X X_i^{Cj}\right), \tag{4}$$

де $W_{coord} \in \mathbb{R}^{d_E \times 6 d_E}$ – матриця обрахунку латентної складової аргументів команд X_i^{Cj}, що здатна навчатися під час тренування моделі, $W_X \in \mathbb{R}^{d_E \times 257}$ – матриця обрахунку кожного значення латентної складової аргументів команд X_i^{Cj}, що здатна навчатися під час тренування моделі, $X_i^{Cj} \in \mathbb{R}^{257 \times 6}$ – матриця аргументів команди C_i^j, яка містить її значення, $vec(\cdot)$ – операція, що означає векторизацію матриці: $vec\left(A^{\alpha \times \beta}\right) = a^{\alpha\beta}$.

$$e_{ind,i}^j = W_{ind}\ \delta_j, \tag{5}$$

де $W_{ind} \in \mathbb{R}^{d_E \times N_S c}$ – матриця обрахунку латентної складової порядкового номера команд C_i^j, що здатна навчатися під час тренування моделі, $\delta_j \in \mathbb{R}^{N_S c}$ – унітарний вектор, що дорівнює одиниці в єдиному елементі, який відповідає порядковому номеру позиції команди C_i^j в послідовності команд S_i^C, розмірності $N_S c$.

Під час розробки прикладного рішення, що базується на використанні трансформерів, варто врахувати особливості їхньої роботи:

- значна перевага, що полягає у можливості роботи із послідовністю з урахуванням її контексту, компенсується вагомим недоліком з практичної точки зору, а саме – час обчислення. Цей фактор сповільнює процес тренування таких моделей, особливо в умовах з обмеженими ресурсами;
- трансформери працюють з обмеженими послідовностями, тому модель DeepSVG також має встановити лімітування розмірності загальної кількості шляхів $\exists M_1 \forall N_p \Rightarrow N_p \le M_1$ та команд $\exists M_2 \forall n \left(n \in \left\{ N_{S_i^C} \right\}_1^{N_p} \right) \Rightarrow n \le M_2$;

– область значень трансформерів є також обмеженою, що пов'язано із лімітованою розмірністю латентного простору, який трапляється в архітектурі трансформерів.

Враховуючи наведені особливості, автори з метою як покращення продуктивності, так і пришвидшення процесу тренування моделі пішли на такі обмеження:

– обмежена множина SVG-команд, що може обробити модель: $c_i^j \in c^* = \{SOS, move\ to, line\ to, cubic\ bezier, close\ path, EOS\}$. При цьому функціональні можливості такого рішення не зменшуються, оскільки решта команд може бути виведена з комбінації доступних команд. Ідентифікатори *SOS* та *EOS* є службовими (вони не входять до ідентифікаторів команд, що попередньо визначені специфікацією SVG), а їхня наявність обґрунтована принципом роботи трансформерів;
– автори лімітували максимальний розмір аргументів команди: $X_i^{Cj} \in \mathbb{R}^6$. При цьому якщо команда, наприклад, використовує лише два аргументи, то інші аргументи будуть заповнені наперед визначеним значенням -1, які не будуть впливати на результат тренування моделі;
– аргументи команд X_i^{Cj} квантизуються. Оскільки для визначеної множини доступних команд c^* аргументами є дійсні числа, їх необхідно дискретизувати у визначеному діапазоні. Автори квантизують значення аргументів до 8 біт, що дає змогу зберігати $2^8 + 1 = 257$ (одне значення відводиться на те, щоб відрізняти аргументи, які не використовуються);
– обмеженню також піддалася можлива загальна кількість шляхів у векторному зображенні $N_p \leq 8$ та можлива загальна кількість команд в окремому шляху $\forall n\left(n \in \left\{N_{S_i^C}\right\}_1^{N_p}\right) \Rightarrow n \leq 30$, лімітуючи таким чином можливу загальну кількість команд, які утворюють векторне зображення: $\sum_i^{N_p} N_{S_i^C} \leq 240$;
– автори застосовують попередню обробку SVG зображення для більш продуктивного навчання моделі. По-перше, команди, що не входять до множини c^*, перетворюються на більш прості команди, при цьому збільшуючи вислідну кількість команд (тобто одна команда може перетворитися, наприклад, на чотири). По-друге, шляхи піддаються апроксимації за допомогою або алгоритму Рамер-Дуглас-Пюкер [18], або алгоритму Філіпа Дж. Шнайдера [19], залежно від геометричного примітиву, що апроксимується. По-третє, автори нормалізують SVG зображення, сортуючи вершини шляхів, починаючи від найвищої лівої координати за годинниковою стрілкою.

У підсумку автори моделі демонструють здатність їхнього рішення в автоматичний спосіб створювати анімацію між ключовими кадрами, які надаються у вигляді двох або більше векторних зображень. Також вони відображають можливість моделі здійснювати інтерполяцію між двома різними зображеннями, утворюючи при цьому проміжні нові векторні форми.

Незважаючи на те, що DeepSVG працює саме із зображенням, доводиться констатувати, що таке рішення не є придатним у використанні в галузі інклюзивної ілюстрації. Причиною непридатності є низка обмежень, що ґрунтуються на особливостях архітектури мережі, яка базується на трансформерах.

До того ж нормалізація вхідних даних змушує споживача моделі отримувати в результаті векторні зображення, які складаються із шляхів, що піддалися оптимізації (така проблема також притаманна SVG-VAE). У такий спосіб модель нівелює переваги векторного зображення в контексті інклюзивної ілюстрації: можливість роботи з окремими елементами зображення та повторного використання зображень в інших виданнях, оскільки їхнє представлення є оптимальним з точку зору моделі DeepSVG, але непридатним у використанні ілюстратором.

Висновки. Отже, нами було досліджено методику токенізації векторного зображення з огляду на завдання синтезу тактильної графіки, а також розглянуто та проаналізовано дві методики токенізації векторної графіки – на основі моделі VAE та на основі моделі трансформера. Наведені рішення є непридатними до застосування у завданні токенізації векторної тактильної графіки, що насамперед пов'язано із низькою пропускною здатністю. Обидві методики нівелюють переваги векторного зображення, які були зазначені на початку роботи, саме у площині практичного застосування такого підходу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джуринський Є. А., Маїк В. З. Аналіз процесу підготовки ілюстрацій для інклюзивної літератури. Квалілогія книги. 2022. № 1 (41). С. 7−15.
2. Midjourney AI model tool for text-to-image conversion. URL: https://www.midjourney.com/ (access date: 04/05/2023).
3. Stable Diffusion AI model tool for text-to-image conversion. URL: https://stablediffusionweb.com/ (access date: 04/05/2023).
4. DALL·E 2 AI system that can create realistic images and art from a description in natural language. URL: https://openai.com/product/dall-e-2/ (access date: 04/05/2023).
5. Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. Ludwig Maximilian University of Munich & IWR, 2022.
6. Ramesh A., Dhariwal P., Nichol A., Chu C., Chen M. Hierarchical Text-Conditional Image Generation with CLIP Latents. 2022. Doi: https://doi.org/10.48550/arXiv.2204.06125.
7. Oppenlaender J. The Creativity of Text-to-Image Generation. In 25th International Academic Mindtrek conference (Academic Mindtrek 2022), November 16–18, 2022, Tampere, Finland. ACM, New York, NY, USA, 11 p. 2022. Doi: https://doi.org/10.1145/3569219.3569352.
8. Джуринський Є. А., Маїк В. З. Підготовка ілюстрацій для інклюзивної літератури за допомогою моделей штучного інтелекту синтезу зображення з тексту. Наукові записки [Української академії друкарства]. 2023. № 1 (66). С. 155−163.
9. Diederik P., Welling M. Auto-Encoding Variational Bayes. Universiteit van Amsterdam. 2013. 14 p. Doi: https://doi.org/10.48550/arXiv.1312.6114.
10. Gontijo Lopes. R, Ha D., Eck D., Shlens J. A Learned Representation for Scalable Vector Graphics. Google Brain. 2019. 13 p. Doi: https://doi.org/10.48550/arXiv.1904.02632.
11. Carlier A., Danelljan M., Alahi A., Timofte R. DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation. Ecole Polytechnique Fédérale de Lausanne. ETH Zurich. 2020. 19 p. Doi: https://doi.org/10.48550/arXiv.2007.11301.

12. Graves A. Generating Sequences With Recurrent Neural Networks. University of Toronto. 2014. 43 p. Doi: https://doi.org/10.48550/arXiv.1308.0850.
13. Vaswani A., Bengio S., Brevdo E., Chollet F., N. Gomez A., Gouws S., Jones L. Tensor2Tensor for Neural Machine Translation. Google Brain. DeepMind. 2018. 9 p. Doi: https://doi.org/10.48550/arXiv.1803.07416.
14. Hu T., Chen F., Wang H., Li J., Wang W., Sun J., Li Z. Complexity Matters: Rethinking the Latent Space for Generative Modeling. Hong Kong University of Science and Technology. National University of Singapore. 2023. 22 p. Doi: https://doi.org/10.48550/arXiv.2307.08283.
15. Hochreiter Sepp, Schmidhuber Jürgen. Long Short-term Memory. Neural computation. 1997. 9.
16. Hjorth L. U., Nabney Ian. Regularisation of mixture density networks. 1999. Vol. 2. 521−526.
17. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., N. Gomez A., Kaiser L., Polosukhin I. Attention Is All You Need. Google Brain. Google Research. 2023. 15 p. Doi: https://doi.org/10.48550/arXiv.1706.03762.
18. Ramer Urs. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*. 1972. 1 (3). 244–256. doi: 10.1016/S0146-664X(72)80017-0.
19. Weaver W. P. A More Excellent Way: Philip Melanchthon's Corinthians Lectures of 1521-22. Renaissance and Reformation. 2014. no. 1. 31–63. URL: http://www.jstor.org/stable/43446567.

REFERENCES

1. Dzhurynskyi, Ye. A., & Maik, V. Z. (2022). Analiz protsesu pidhotovky iliustratsii dlia inkliuzyvnoi literatury: Kvalilohiia knyhy, 1 (41), 7−15 (in Ukrainian).
2. Midjourney AI model tool for text-to-image conversion. Retrieved from https://www.midjourney.com/ (access date: 04/05/2023) (in English).
3. Stable Diffusion AI model tool for text-to-image conversion. Retrieved from https://stablediffusionweb.com/ (access date: 04/05/2023) (in English).
4. DALL·E 2 AI system that can create realistic images and art from a description in natural language. Retrieved from https://openai.com/product/dall-e-2/ (access date: 04/05/2023) (in English).
5. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. Ludwig Maximilian University of Munich & IWR (in English).
6. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. Doi: https://doi.org/10.48550/arXiv.2204.06125 (in English).
7. Oppenlaender, J. (2022). The Creativity of Text-to-Image Generation. In 25th International Academic Mindtrek conference (Academic Mindtrek 2022), November 16–18, 2022, Tampere, Finland. ACM, New York, NY, USA. Doi: https://doi.org/10.1145/3569219.3569352 (in English).
8. Dzhurynskyi, Ye. A., & Maik, V. Z. (2023). Pidhotovka iliustratsii dlia inkliuzyvnoi literatury za dopomohoiu modelei shtuchnoho intelektu syntezu zobrazhennia z tekstu: Naukovi zapysky [Ukrainskoi akademii drukarstva], 1 (66), 155−163 (in Ukrainian).
9. Diederik, P., & Welling, M. (2013). Auto-Encoding Variational Bayes. Universiteit van Amsterdam. Doi: https://doi.org/10.48550/arXiv.1312.6114 (in English).

10. Gontijo, Lopes R., Ha, D., Eck, D., & Shlens, J. (2019). A Learned Representation for Scalable Vector Graphics. Google Brain. Doi: https://doi.org/10.48550/arXiv.1904.02632 (in English).
11. Carlier, A., Danelljan, M., Alahi, A., & Timofte, R. (2020). DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation. Ecole Polytechnique Fédérale de Lausanne. ETH Zurich. Doi: https://doi.org/10.48550/arXiv.2007.11301 (in English).
12. Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. University of Toronto. Doi: https://doi.org/10.48550/arXiv.1308.0850 (in English).
13. Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., N. Gomez, A., Gouws, S., & Jones, L. (2018). Tensor2Tensor for Neural Machine Translation. Google Brain. DeepMind. Doi: https://doi.org/10.48550/arXiv.1803.07416 (in English).
14. Hu, T., Chen, F., Wang, H., Li, J., Wang, W., Sun, J., & Li, Z. (2023). Complexity Matters: Rethinking the Latent Space for Generative Modeling. Hong Kong University of Science and Technology. National University of Singapore. Doi: https://doi.org/10.48550/arXiv.2307.08283 (in English).
15. Hochreiter, Sepp, Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation (in English).
16. Hjorth, L. U., & Nabney, I. (1999). Regularisation of mixture density networks, 2. 521−526 (in English).
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., N. Gomez, A., Kaiser, L., & Polosukhin, I. (2023). Attention Is All You Need. Google Brain. Google Research. Doi: https://doi.org/10.48550/arXiv.1706.03762 (in English).
18. Ramer, Urs. (1972). An iterative procedure for the polygonal approximation of plane curves: Computer Graphics and Image Processing, 1 (3), 244–256. doi: 10.1016/S0146-664X(72)80017-0 (in English).
19. Weaver, W. P. (2014). A More Excellent Way: Philip Melanchthon's Corinthians Lectures of 1521-22. Renaissance and Reformation, 1, 31–63. Retrieved from http://www.jstor.org/stable/43446567 (in English).

doi: 10.32403/1998-6912-2023-2-67-11-20

TOKENIZATION OF VECTOR GRAPHICS IN THE CONTEXT OF TACTILE GRAPHICS SYNTHESIS

Y. A. Dzhurynskyi, V. Z. Mayik

Ukrainian Academy of Printing,
19, Pid Holoskom St., Lviv, 79020, Ukraine
vol_maik@meta.ua

In the field of inclusive publishing, the development of tactile illustration requires an appropriate set of competencies from the designer. Due to the relative shortage of personnel on the labor market among fine arts specialists who have knowledge related to the technical execution of convex-tactile graphics, the process of finding and placing

such an employee at a publishing house is a non-trivial task, because it requires both time and financial costs. At the same time, publishing houses are forced to include an additional cost item, which is the training of such workers. The above applied problems can be solved with the help of information systems for the synthesis of tactile graphics, which, using the means of artificial intelligence, will partially or completely replace the designer of tactile illustrations. The work considers one of the stages of solving the problem of synthesis of tactile graphics, namely, the tokenization of vector graphics, which is perfectly suited as a format for presenting tactile graphics. Tokenization implies the representation of the original information (in this case – vector graphics) in another – more optimal representation, which can be used by a model based on artificial intelligence. The purpose of this research is to determine the expediency of using the technique of tokenization of tactile graphics in vector representation in the task of synthesizing tactile graphics. The paper considers two methods of tokenization, which differ in the architecture of the artificial intelligence model: a VAE-based model and a transformer-based model. Despite the fact that both models were primarily developed to solve other problems, nevertheless, the approach they use can be borrowed and adapted to the problem that is the subject of this study. The work provides an analysis of the listed models, with the determination of their advantages and disadvantages, and with the formalization of the principle of operation of these solutions. During the analysis, it is found that the considered models nullify the advantages of the vector representation of tactile graphics, which is primarily due to the low bandwidth of the considered models (at the same time, there are other circumstances that are given in the main part). At the end, the conclusion to which this study led is given, which is the impossibility of using the given approaches.

***Keywords**: information technology, artificial intelligence, model, tokenization technique, illustration requirements, image processing, tactile graphics, inclusive illustration, inclusive literature, Braille.*

Стаття надійшла до редакції 21.08.2023.
Received 21.08.2023.