УДК 004.415.5
DOI: 10.18523/2617-3808.2020.3.4-6

Гороховський С. С., Франків О. О.

МАШИННЕ НАВЧАННЯ ТА ДОПОВНЕНА РЕАЛЬНІСТЬ НА ПРИСТРОЯХ НА БАЗІ iOS ІЗ ФРЕЙМВОРКОМ MLARKit

Створено фреймворк MLARKit, що дає змогу легко користуватися складними для використання у мобільних пристроях алгоритмами машинного навчання та доповненої реальності, враховуючи їхні особливості. Фреймворк створено максимально гнучким, тож сторонні розробники зможуть максимально задовольнити свої потреби без самостійної реалізації.

Ключові слова: машинне навчання, Core ML, доповнена реальність, AR Kit, мобільні пристрої, iOS.

Вступ

У 2017 р. майже одночасно на своїх конференціях розробників два головні виробники операційних систем для мобільних платформ – Apple та Google – оголосили про визначення вектора розвитку технологій у напрямі елементів штучного інтелекту на наступну декаду. На теперішній момент на базі обох найпопулярніших мобільних операційних систем – iOS та Android – існують інструменти для роботи в галузі машинного навчання, а також доповненої реальності, однак ці технології ще перебувають у стадії активного розроблення, а значна кількість інженерів-розробників мобільних додатків не мають достатньо знань для роботи з ними. Через це своєю чергою виникають певні перешкоди для впровадження нових технологій у промисловому створенні мобільних додатків. Саме для розв'язання цієї проблеми було розроблено спеціальний фреймворк, що на базі представлених Apple Core ML та AR Kit створює певний прошарок абстракції.

Особливості мобільних пристроїв

Донедавна використання алгоритмів машинного навчання та доповненої реальності на мобільних пристроях було неможливим через серйозні виклики, що стоять перед будь-яким мобільним програмним забезпеченням. Проблеми були пов'язані, зокрема, із швидкодією, використанням мережі, кількістю спожитої енергії та приватністю.

Користувач схильний використовувати швидкі й красиві додатки, адже це передбачає сама концепція мобільного пристрою. Пристрій, який не здатний отримати вхідний виклик через роботу неоптимізованої програми, перестає бути телефоном, а саме це є головним його завданням. Потрібно зазначити, що естетична складова програми, зокрема, складне розташування елементів, анімації та кольори різної прозорості, також залежить від швидкодії, оскільки на побудову користувацького інтерфейсу теж витрачається процесорний час.

Питання використання мережі, особливо мобільної, є також дуже важливим. По-перше, мобільний трафік є дорогим і користувач мало зацікавлений у програмах, що збільшують його використання заради додаткових та інколи непомітних функцій. По-друге, надмірне використання трафіку дуже знижує енергоефективність, важливість якої в мобільних системах є очевидною. Окрім фінансового та екологічного аспектів цього питання важливу роль відіграє мінімізація залежності пристрою від зовнішніх джерел енергії. Ба більше, кількість циклів заряду-розряду є суттєвим показником уживаності мобільного пристрою, зокрема з огляду на те, що для певних пристроїв заміна акумулятора є нерентабельною.

Конфіденційність є чи не найважливішим аспектом для мобільного програмного забезпечення. Окрім того, що постійний потік даних між сервером і мобільним клієнтом через мережу сам по собі пов'язаний більшою мірою із проблемою використання трафіку, він так само стає мішенню для кіберзлочинців, якщо є вразливим для перехоплення. Розглянемо на прикладі підказок під час введення тексту на клавіатурі. Надсилання тексту повідомлення на сервер після введення кожного слова заради підказки наступного є сумнівною ідеєю, і навіть не так через залежність від швидкості доступу до мережі, як через те, що всі повідомлення користувача завантажують у мережу. Важливо завжди пам'ятати, що немає абсолютно надійного способу захисту інформації, а навіть один прецедент витоку даних може відіграти вирішальну роль для іміджу програмного забезпечення та його розробника.

Фреймворк MLARKit

Особливості застосування MLARKit

Цей фреймворк розв'язує задачу розпізнавання образів і створення міток для розпізнаних об'єктів у доповненій реальності. Більше того, він простий у налаштуванні, однак достатньо гнучкий для задоволення потреб інженерів у конкретній задачі. Найбільшим плюсом у використанні цього фреймворку є те, що для його реалізації не потрібно заглиблюватися у деталі роботи Core ML та ARKit.

MLARkit написаний на Swift 5 і використовує лише системні бібліотеки та фреймворки iOS:

1) Foundation;
2) UIKit;
3) Core ML;
4) Vision;
5) ARKit.

Наразі фреймворк підтримує три режими розпізнавання: швидке розпізнавання QR-кодів і прочитання їхнього вмісту, розпізнавання заготованих наперед зображень за допомогою рушія ARKit та розпізнавання образів за заданою декларативно моделлю нейронної мережі. Для відображення міток у доповненій реальності передбачено такі опції:

1. Задання текстового заповнення для мітки, що використовується за замовчуванням.
2. Задання URL для програвання відеоконтенту на мітці за допомогою віртуального відеоплеєра.
3. Задання власної мітки, яку можна відрендерити на базі UIKit для кожного конкретного розпізнаного об'єкта.
4. Задання власного тривимірного об'єкта для використання як мітки.

Нова версія фреймворку також дає змогу визначати такий собі «безпечний окіл» об'єктів – окіл об'єкта у просторі, в якому інші об'єкти не можуть перебувати. Це використовують для запобігання дублювання міток у разі повторного розпізнавання того самого об'єкта.

Також фреймворк дає змогу задавати в метрах зміщення для відображення міток. Це допомагає використовувати розпізнані об'єкти як орієнтири в просторі.

Потрібно зазначити, що хоча користувачу фреймворку і надано можливість виконувати власні обчислення, вклинюючись у стандартний потік роботи фреймворку, всі користувацькі обчислення мають бути якомога лаконічнішими та якомога більш оптимізованими.

Особливості будови фреймворку

MLARkit працює на базі двох системних фреймворків – Vision, який своєю чергою працює на базі Core ML, та ARKit. Головний сценарій роботи передбачає два етапи: розпізнавання та відображення. Спершу користувач задає свою модель нейронної мережі, або ж може цього не робити. В такому разі розпізнавання відбуватиметься у стандартному режимі розпізнавання QR-кодів. Зауважимо, що завдяки використанню Vision таке розпізнавання є досить точним і менш чутливим до паралаксу. Сам процес розпізнавання відбувається у багатопотоковому середовищі з використанням Grand Central Dispatch. Після цього користувач має визначити дані для відображення, залежно від постановки своєї задачі. Користувач також має підготувати для відображення дані, які повертає його нейронна мережа. Після цього фреймворк розраховує місця в просторі, де потрібно відображати мітку. Користувач може задати власні опції відображення, реалізувавши відповідні методи делегата. Після цього фреймворк рендерить зображення або тривимірний об'єкт та відображає його в доповненій реальності.

Головним об'єктом фреймворку є контролер MLARController, який відповідає за відображення ARSCNView, що своєю чергою відповідає за відображення відеопотоку головної камери смартфона та накладення віртуальних об'єктів поверх нього. Також у класі визначено наперед заготований запит до нейронної мережі, що економить ресурси за постійного до неї звертання для розпізнавання. Цей контролер має два розширення. Розширення MLARController+ML відповідає за створення запитів для розпізнавання та правильне їх трактування. Друге розширення – MLARController+AR – відповідає за правильну генерацію віртуальних об'єктів та відображення їх на ARSCNView. У цій реалізації мітки в доповненій реальності ставлять у центр прямокутника, в межах якого перебуває розпізнаний об'єкт на тій самій відстані від користувача, на якій перебуває і сам розпізнаний об'єкт. Цю відстань визначають за допомогою функції hitTest рушія ARKit. Очевидно, що це визначення має нестабільну точність, адже чутливість до світла та надто інтенсивних рухів є наразі нерозв'язаною проблемою самого ARKit.

Для передавання даних використовують структуру MLARDetectionData та агрегований тип кортежа MLARDisplayInfo, що містять усі необхідні дані про розпізнавання та відображення.

Керування процесом реалізовано за стандартним підходом для iOS систем із використанням шаблона делегування. Цей шаблон є чи не найпопулярнішим у контексті POP (англ. Protocol Oriented Programming – протокольно-орієнтоване програмування), де за основу беруть протоколи. У такий спосіб на кожному етапі процесу залучається делегат із метою отримання додаткових настанов. Відповідно для керування достатньо відповідати протоколу MLARControllerDelegate, реалізувавши необхідні методи відповідно до поставленої задачі.

Висновки

За допомогою розробленого фреймворку здійснено спробу полегшити впровадження машинного навчання та доповненої реальності у промисловому створенні мобільних додатків. Використання готового рішення може позитивно вплинути на собівартість продукту, додавши натомість непростий у реалізації функціонал. Завдяки високій гнучкості та відмові від використання бібліотек та модулів сторонніх розробників, що позитивно впливає на підтримуваність і продуктивність, цей фреймворк може допомогти у легкій реалізації додаткових функцій і в нових, і в уже наявних додатках.

У майбутньому цей фреймворк можна розвивати, додаючи нові можливості, а через те, що за основу взято системні модулі, кожне оновлення рушіїв у рамках оновлення операційної системи автоматично оновлюватиме і сам MLARkit.

Список літератури

1. Buck E. Cocoa Design Patterns / E. Buck, D. Yacktman. – Upper Saddle River, NJ: Addison-Wesley, 2009. – 456 p.
2. Hollemans M. Core ML Survival Guide [Electronic resource] / Matthiijs Hollemans. – 2018. – 363 p. – Mode of access: https://leanpub.com/coreml-survival-guide.
3. Newnham J. Machine Learning with Core ML: An iOS developer's guide to implementing machine learning in mobile apps [Electronic resource] / Joshua Newnham. – 2018. – 378 p. – Mode of access: https://www.amazon.com/Machine-Learning-Core-developers-implementing/dp/1788838297.
4. Wolfensparger D. Apple Arkit Revealed: Augmented and Mixed Reality for Iphone and Ipad / Dell Wolfensparger. – 2018. – 180 p.

References

Buck, E. M., & Yacktman, D. A. (2010). *Cocoa design patterns*. Upper Saddle River, NJ: Addison-Wesley.

Hollemans, M. (2018). *Core ML Survival Guide*. Leanpub. Retrieved from https://leanpub.com/coreml-survival-guide.

Newnham, J. (2018). Machine Learning with Core ML: An iOS developer's guide to... Retrieved from https://www.amazon.com/Machine-Learning-Core-developers-implementing/dp/1788838297.

Wolfensparger, D. (2018). *Apple Arkit Revealed: Augmented and mixed reality for iphone and ipad*. Place of publication not identified: APRESS.

S. Gorokhovskyi, O. Frankiv

MACHINE LEARNING AND AUGMENTED REALITY ON iOS DEVICES WITH MLARKIT FRAMEWORK

Machine learning is a technology that requires a significant amount of computing power and therefore is not efficient in terms of energy consumption. The equivalent may be applied to the technology of augmented reality which impacts in contrast not only the processor but also the camera, accelerometer, and gyroscope. Moreover, due to the work with the visual effects, the speed of computing truly matters to the latter. In recent years it has become possible to use these technologies on mobile devices due to the grand efficiency optimizations. The problem these days is that it requires an imposing amount of resources for engineers to spend on the development of the experiences with the use of technologies mentioned above.

The solution developed is the next step in simplifying the process of integration of such experiences in mobile applications for the iOS operating system. While Core ML and AR Kit provide a simple and comprehensive interface for the basic machine learning and augmented reality routines, the new framework – Machine Learning Augmented Reality Kit (MLARKit) – provides the same simplicity and flexibility for the engineers in terms of the two tasks: replacing real objects with the virtual ones and marking of recognized objects or images in augmented reality which are widely used in modern mobile applications as the additional functionality. With the use of the above-mentioned framework, the integration of such experience comes down to the conformance of the only protocol which provides a full range of settings.

This article covers the problems that may be faced while solving power-consuming and energy inefficient tasks on mobile devices as well as the approaches used to reduce such impact. The manner in which the newly developed framework is designed is described in the article as well. As an improvement, it is important to consider tracking the moving objects which may reduce the amount of computing power consumed significantly.

Keywords: machine learning, Core ML, augmented reality, AR Kit, mobile devices, iOS.

Матеріал надійшов 10.06.2020