УДК 004.932.72
DOI https://doi.org/10.24144/2616-7700.2021.38(1).137-142

Л. М. Дяконюк[1], А. С. Мудрик[2], Я. А. Корольчук[3], М. І. Кондор[4]

[1] Львівський національний університет ім. І. Франка,
доцент кафедри прикладної математики та інформатики,
кандидат фізико-математичних наук
liliya.dyakonyuk@lnu.edu.ua
ORCID: https://orcid.org/0000-0001-5663-0501

[2] Львівський національний університет ім. І. Франка,
магістр 2-го року навчання факультету прикладної математики та інформатики
mudrykandrew@gmail.com
ORCID: https://orcid.org/0000-0002-6608-4425

[3] Львівський національний університет ім. І. Франка,
магістр 2-го року навчання факультету прикладної математики та інформатики
koroltchukjaroslav@gmail.com
ORCID: https://orcid.org/0000-0001-6365-6101

[4] Львівський національний університет ім. І. Франка,
магістр 2-го року навчання факультету прикладної математики та інформатики
martakondor17@gmail.com
ORCID: https://orcid.org/0000-0002-8715-0997

РОЗПІЗНАВАННЯ МАТЕМАТИЧНИХ ФОРМУЛ НА БАЗІ ДАНИХ CROHME

У наш час найбільш точні моделі для розпізнавання об'єктів базуються на двоступеневому підході, популяризованому як R-CNN. На відміну від них, одноступеневі моделі, що застосовуються під час регулярного, детального відбору зразків, можуть бути швидшими та простішими, але вони не досягають точності двоступеневих моделей. Проте з новою функцією втрат, дисбаланс класу, який виникає під час тренування на наборі даних, зникає. Саме тому одноступенева модель має переваги в продуктивності та точності на відміну від двоступеневої. У роботі використано цей дисбаланс класів, щоб переформувати стандартні, перехресні ентропійні втрати таким чином, щоб зменшити їх. В архітектурі RetinaNet [1], функція втрат Focal Loss [1] сфокусовує навчання на наборі даних, які зустрічаються рідше, і запобігає перевантаженню моделі під час тренувань. Архітектура RetinaNet була протестована на наборі даних CROHME [4], що був розширений за допомогою алгоритму Data Augmentation [9] для збільшення частоти входження певних елементів формул. Також було порівняно дві бібліотеки машинного навчання: TensorFlow та Torch. Отримані результати показують, що коли модель тренується з фокальною втратою, RetinaNet показує дуже добрі результати та має хорошу швидкість виконання. Окрім того, отриману модель було інтегровано в веб-застосунок на основі мікросервісної архітектури. Основними веб-фреймворками було використано NodeJs для серверної частини та VueJs для рівня подання. Для роботи з базами даних ми використовуємо MongoDB. Розгортання програми відбувається за допомогою хмарної служби AWS на основі Lambda-функцій, що дає змогу виокремити процеси навчання, обробки, візуалізації та контролювати ресурси серверу окремо для кожного процесу.

Ключові слова: розпізнавання об'єктів, Retina.Net, набір даних, машинне навчання, CROHME.

1. Вступ. Останнім часом системам розпізнавання об'єктів приділяють все більше уваги. Вони досить широко використовуються, але важко створити

систему, яка може розпізнавати об'єкти без належних навчальних даних. Складність розпізнавання математичних виразів залежить від багатьох факторів, до прикладу: кількості об'єктів, набору доступних символів, граматики запису.

Точність отриманих результатів є безпосередньо пов'язаною з навчальними даними, тому використовуються методи збільшення навчальної вибірки для покращення їх якості. Є набір основних архітектур для розв'язування цієї задачі. Як відомо, їх можна класифікувати як однокрокові і двокрокові. Однокрокові є швидкими, проте двокрокові є більш точними. Була використана архітектура RetinaNet, яка є однокроковим методом. Вона має найкращі показники відносно часу виконання/точності на популярних навчальних даних. Після навчання модель готова до використання. Цю модель можна інтегрувати за допомогою Docker контейнеру.

2. Основний результат. Задача полягає у тому, щоб на наборі даних CROHME, навчити детектор розпізнавати математичні символи формули.

Умовно процес можна розділити на 3 частини:

1) формування даних для навчання,
2) навчання та оцінка мережі,
3) імплементація моделі.

1. Скористаємось базою формул CROHME, що поновлюється щорічно та містить деякий базовий набір формул та символів написаних від руки у спеціальному форматі INKML, створеному для цієї вибірки.

Цей формат містить:

- інформацію про розміщення символу на сторінці,
- назву символів формули,
- залежності від інших символів, якщо такі є.

Також додатково він може містити

- інформацію про особу, що писала ці символи (корисно для проблем визначення віку, статі або ж руку, якою символи були написані),
- LaTeX стрічку відповідного символу.

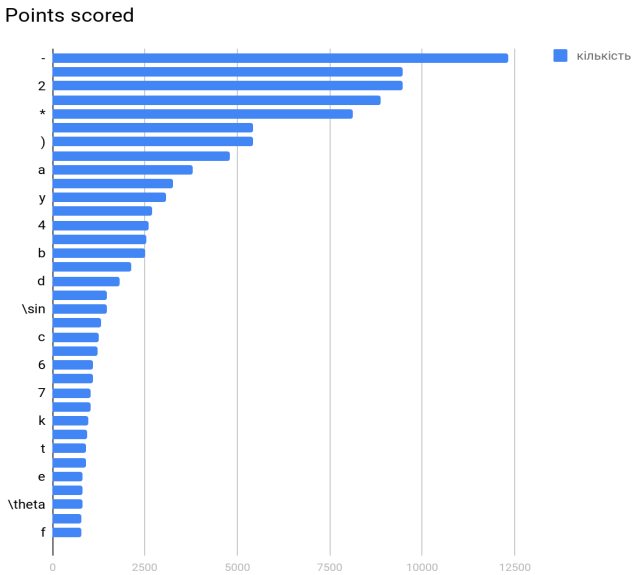

Рис. 1. Розподіл класів CROHME

Детально дослідивши набір даних, можна виділити символи, що мають малу частоту входження у тренувальну вибірку (рис.1). Тому для формул із ними застосуємо наступні перетворення:

- поворот,
- віддзеркалення,
- обрізування,
- зміну фону.

2. RetinaNet [1] був розроблений Facebook AI Research (FAIR) у 2018 році, і є однокроковим детектором. Якщо порівнювати двокрокові і однокрокові детектори, то всі моделі типу R-CNN включали в себе 2 етапи.

1-ий етап пропонує набір регіонів на фотографії.

2-ий етап класифікує регіон до того чи іншого класу.

Однокроковий детектор пропускає виділення регіонів і напряму знаходить оточуючу рамку конкретного об'єкту.

Focal loss - функція втрат, яка буде використовуватись для тренування моделей. Ця функція призначена для однокрокового сценарію виявлення об'єктів, при якому існує надзвичайний дисбаланс між класами переднього плану та фоном під час навчання.

Також було використано ResNet50 [2] як основну частину архітектури RetinaNet.

Для порівняння архітектура була реалізована на бібліотеках TensorFlow [8] і PyTorch [7] на мові програмування Python.

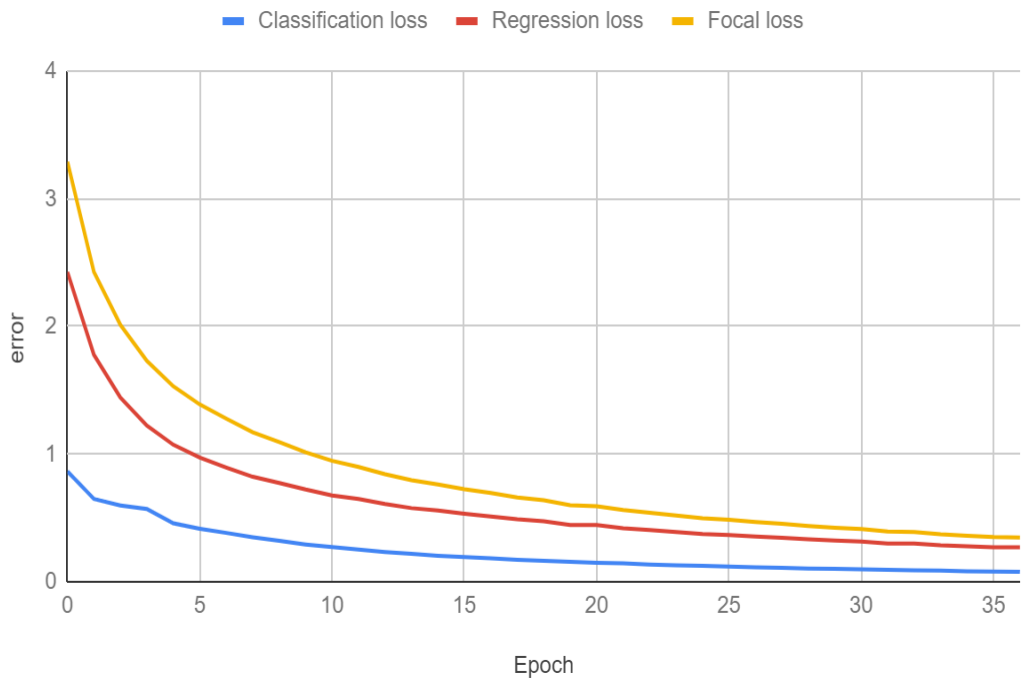

Рис. 2. Результат навчання tensorflow

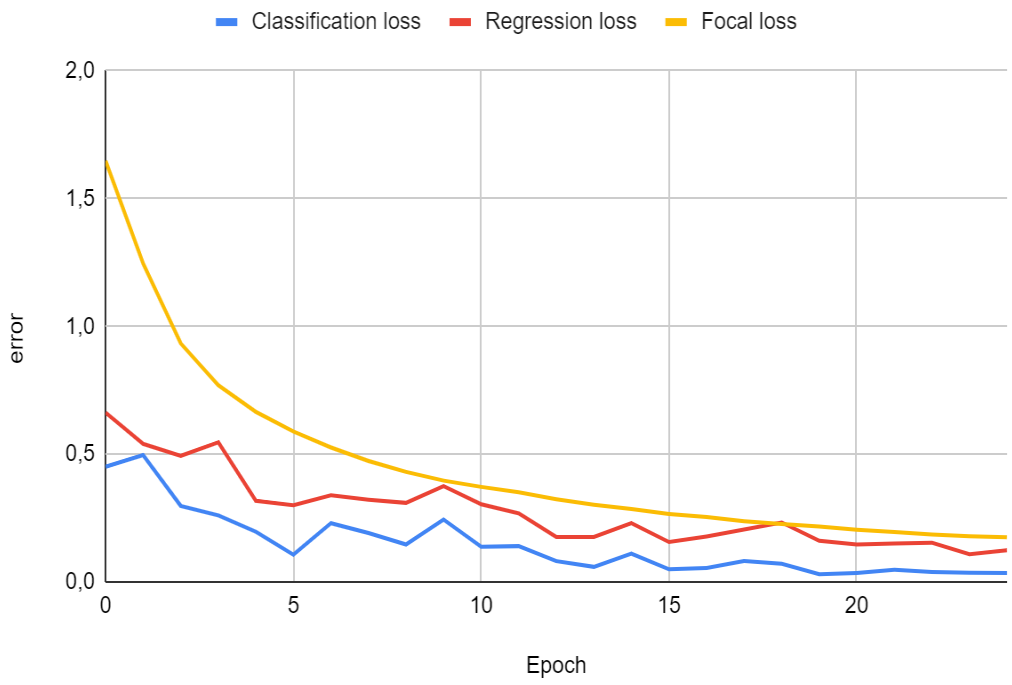

Рис. 3. Результат навчання torch

У таблиці 1 відображені чисельні результати обчислення різних функцій втрат при застосуванні двох бібліотек.

Таблиця 1.

Оцінка моделі

Бібліотека	**Classification loss**	**Regression loss**	**Focal loss**
PyTorch	0,034	0,123	0,174
TensorFlow	0,077	0,261	0,346

Проаналізувавши результати, можемо зробити висновок, що швидкість навчання і краща точність досягається з pytorch, що робить його фаворитом для даної задачі. Через певні особливості паралельного навчання, які є вбудовані у pytorch, і досягається ця різниця.

Час роботи pytorch для однієї картинки складає ∼700 мс, так як у tensorflow ∼1 с. Для тесту було використаний графічний процесор Nvidia 2070 Super.

Модель може також пристосовуватись до почерку конкретної людини. Для цього необхідно включити приклади з цим почерком до навчальної вибірки.

3. Висновки та перспективи подальших досліджень. У нашій роботі однокрокова мережа була навчена розпізнавати написані від руки математичні вирази, які згодом подавались у вигляді формул LaTeX. Основною проблемою було покращення існуючого набору даних. Також було порівняно результати двох бібліотек для роботи з нейронними мережами.

Було застосовано архітектуру RetinaNet на основі ResNet50. Проте для цього

завдання не було порівняно основи для RetinaNet. Також ми не можемо провести порівняння точності результатів, оскільки не було проведене навчання для інших моделей.

На відміну від інших робіт на цю тему, в цій роботі була використана саме однокрокова мережа, що виділяється своєю простотою та швидкістю. Але відомо, що однокрокові мережі поступаються точністю двокроковим. Для того, щоб відслідкувати відхилення в точності потрібно було провести навчання інших моделей.

На основі цього було розроблено додаток на базі мікросервісної архітектури, використовуючи інструменти для розпізнавання та ідентифікації математичного виразу.

Список використаної літератури

1. Focal loss for dense object detection / T. Lin, P. Goyal, R. Girshick та ін. 2018. URL: https://arxiv.org/pdf/1708.02002.pdf.
2. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. 2015. URL: https://arxiv.org/pdf/1512.03385v1.pdf.
3. Puppeteer. URL: https://www.npmjs.com/package/puppeteer-core.
4. CHORME. URL: http://www.iapr-tc11.org/mediawiki/index.php/CROHME:_Competition_on_Recognition_of_Online_Handwritten_Mathematical_Expressions.
5. Шум Гаусса. URL: https://en.wikipedia.org/wiki/Gaussian_noise.
6. AWS Lambda. URL: https://en.wikipedia.org/wiki/AWS_Lambda.
7. PyTorch. URL: https://pytorch.org/.
8. TensorFlow. URL: https://www.tensorflow.org/.
9. Data Augmentation. URL: https://en.wikipedia.org/wiki/Data_augmentation

Diakoniuk L. M., Mudryk A. S., Korolchuk Y. A., Kondor M. I. Object detection of the mathematical symbols based on the CROHME dataset.

The highest accuracy object detectors to date are based on a two-stage approach popularized by R-CNN, where a classifier is applied to a sparse set of candidate object locations. In contrast, one-stage detectors that are applied over a regular, dense sampling of possible object locations have the potential to be faster and simpler but have trailed the accuracy of two-stage detectors thus far. But with new extreme foreground-background class imbalance encountered during training of dense detectors, one-stage detector wins by performance and accuracy. We use this class imbalance by reshaping the standard cross-entropy loss such that it down-weights the loss assigned to well-classified examples. In RetinaNet architecture, Focal Loss focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training. We tested RetinaNet on a CROHME dataset that was increased by the default image augmentation algorithm. Also, we compare two machine learning libraries: TensorFlow and Torch. Our results show that when a model is trained with the focal loss RetinaNet shows very good results and has a good speed. Also, we integrate the inference model into service as a part of microservice architecture. As a database for user clients, we use MongoDB. As the main Web framework, we use NodeJs and VueJs. We use AWS as a cloud service for deploying the application. All functionality we deployed based on the AWS Lambda functions.

Keywords: one-stage detectors, object detection, RetinaNet, dataset, machine learning, CROHME.

Список використаної літератури

1. Lin, T., Goyal, P., Girshick, R., He, K., & Dollar, P. (2018). Focal loss for dense object detection, 1–10. https://arxiv.org/pdf/1708.02002.pdf.
2. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition, 1–12. https://arxiv.org/pdf/1512.03385v1.pdf.

3. *puppeteer* (n.d). Retrieved from: https://www.npmjs.com/package/puppeteer-core.
4. *CROHME* (n.d). Retrieved from: http://www.iapr-tc11.org/mediawiki/index.php/CROHME:_Competition_on_Recognition_of_Online_Handwritten_Mathematical_Expressions.
5. *Gaussian noise* (n.d). Retrieved from: https://en.wikipedia.org/wiki/Gaussian_noise.
6. *AWS Lambda* (n.d). Retrieved from: https://en.wikipedia.org/wiki/AWS_Lambda.
7. PyTorch (n.d). Retrieved from: https://pytorch.org/.
8. TensorFlow (n.d). Retrieved from: https://www.tensorflow.org/.
9. Data Augmentation (n.d). Retrieved from: https://en.wikipedia.org/wiki/Data_augmentation.

Одержано 06.04.2021