МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

*Кафедра автоматизації, комп’ютерних наук і технологій*

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

До курсової роботи з дисципліни

**«Методі і технології інтелектуального аналізу даних»**

на тему

«Розпізнавання тексту за допомогою технологій інтелектуальних систем розпізнавання»

Виконав: студент групи КН-22м

Криницький В. О.

Керівник: доцент кафедри АКНТ

Тронь В. В

Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кривий Ріг – 2022

|  |
| --- |
| *Криворізький національний університет* |
| (назва вищого закладу освіти) |

|  |  |
| --- | --- |
| Кафедра | *автоматизації, комп’ютерних наук і технологій* |

|  |  |
| --- | --- |
| Дисципліна | *Методі і технології інтелектуального аналізу даних ь* |

|  |  |
| --- | --- |
| Напрям підготовки | *122 Комп’ютерні науки* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Курс | *перший* | Група | *КН-22м* | Семестр | *перший* |

ЗАВДАННЯ

на курсову роботу студента

|  |
| --- |
| *Криницький Владислав Олександрович* |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1. Тема роботи | *Розпізнавання тексту за допомогою технологій інтелектуальних систем розпізнавання.* |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи | *19.11.2022* |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | *Тема: Розпізнавання тексту за допомогою технологій інтелектуальних систем розпізнавання.* |
|  | |
|  | |
|  | |
|  | |
|  | |

|  |  |
| --- | --- |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) |  |
| ВСТУП, РОЗДІЛ 1 Аналіз предметної області і особливостей застосування методів аналізу | |
| даних і машинного навчання, 1.1 Особливості застосування методів аналізу даних і | |
| машинного навчання, РОЗДІЛ 2 Розроблення моделі аналізу даних і машинного навчання | |
| та алгоритм її роботи, 2.1 Визначення моделі та основних етапів обробки даних, 2.2 Первин- | |
| на обробка даних та бібліотека OpenCV, 2.3 Розпізнавання тексту. Створення та тренування | |
| нейронної мережі за допомогою TensorFlow та Keras, РОЗДІЛ 3 Реалізація моделі та | |
| апробація інтелектуальної системи , ВИСНОВОК, СПИСОК ВИКОРСТАННОЇ | |
| ЛІТЕРАТУРИ, ДОДАТОК А ВИХІДНИЙ КОД ПРОГРАМИ | |

|  |  |  |
| --- | --- | --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) | |  |
| 6. Дата видачі завдання | 04.09.2022 | |

# ЗМІСТ

[ВСТУП 4](#_Toc119660737)

[РОЗДІЛ 1 Аналіз предметної області і особливостей застосування методів аналізу даних і машинного навчання 5](#_Toc119660738)

[1.1 Аналіз предметної області 5](#_Toc119660739)

[1.2 Особливості застосування методів аналізу даних і машинного навчання 8](#_Toc119660740)

[РОЗДІЛ 2 Розроблення моделі аналізу даних і машинного навчання та алгоритм її роботи 10](#_Toc119660741)

[2.1 Визначення моделі та основних етапів обробки даних 10](#_Toc119660742)

[2.2 Первинна обробка даних та бібліотека OpenCV 11](#_Toc119660743)

[2.3 Розпізнавання тексту. Створення та тренування нейронної мережі за допомогою TensorFlow та Keras 13](#_Toc119660744)

[РОЗДІЛ 3 Реалізація моделі та апробація інтелектуальної системи 16](#_Toc119660745)

[ВИСНОВОК 23](#_Toc119660746)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 24](#_Toc119660747)

[ДОДАТОК А ВИХІДНИЙ КОД ПРОГРАМИ 25](#_Toc119660748)

# ВСТУП

Більшість робочих процесів пов'язана з отриманням інформації з друкованих видань. Будь-який бізнес-процес передбачає бланки, рахунки, відскановані юридичні документи та контракти, надруковані на паперовому носії. Такі великі обсяги паперової роботи потребують багато часу та місця для зберігання й опрацювання. Хоча безпаперовий документообіг - це шлях уперед, сканування документа в зображення створює певні труднощі. Цей процес вимагає ручного втручання і може бути виснажливим та повільним.

Оптичне розпізнавання символів (OCR) - це процес перетворення зображення тексту в текстовий формат, який можна прочитати на комп'ютері. Наприклад, під час сканування бланка або квитанції, комп'ютер зберігає скан у вигляді файлу зображення. Текстовий редактор неможливо використовувати для редагування, пошуку або підрахунку слів у файлі зображення. OCR допомагає перетворити зображення в текстовий документ, вміст якого зберігається у вигляді текстових даних.

Під час оцифрування вмісту документа створюються файли зображень із прихованим у них текстом. Програми обробки тексту не можуть обробити текст у зображеннях. Технологія OCR вирішує цю проблему шляхом перетворення зображення в текстові дані, які можуть бути проаналізовані офісним ПЗ. Потім такі дані можна використовувати для аналітики, оптимізації операцій, автоматизації процесів і підвищення продуктивності.

Ще одним широко досліджуваним завданням є розпізнавання рукописного тексту. Нині досягнута точність навіть нижча, ніж для рукописного "друкованого" тексту. Для розв'язання складніших завдань у сфері розпізнавання використовують, як правило, інтелектуальні системи розпізнавання, такі як штучні нейронні мережі. Для калібрування систем розпізнавання тексту створено стандартну базу даних MNIST, що складається із зображень рукописних цифр.

# РОЗДІЛ 1 Аналіз предметної області і особливостей застосування методів аналізу даних і машинного навчання

## 1.1 Аналіз предметної області

Оптичне розпізнавання символів - механічне або електронне переведення зображень рукописного, машинописного або друкованого тексту в текстові дані, які використовуються для представлення символів у комп'ютері (наприклад, у текстовому редакторі). Розпізнавання широко застосовується для перетворення книг і документів в електронний вигляд, для автоматизації систем обліку в бізнесі або для публікації тексту на веб-сторінці. Оптичне розпізнавання символів дає змогу редагувати текст, здійснювати пошук слів або фраз, зберігати його в компактнішій формі, демонструвати або роздруковувати матеріал, не втрачаючи якості, аналізувати інформацію, а також застосовувати до тексту електронний переклад, форматування або перетворення на мову. Оптичне розпізнавання тексту є досліджуваною проблемою в галузях розпізнавання образів, штучного інтелекту та комп'ютерного зору.

Системи оптичного розпізнавання тексту вимагають калібрування для роботи з конкретним шрифтом; у ранніх версіях для програмування було необхідне зображення кожного символу, програма одночасно могла працювати тільки з одним шрифтом. Нині найбільше поширені так звані "інтелектуальні" системи, які з високим ступенем точності розпізнають більшість шрифтів. Деякі системи оптичного розпізнавання тексту здатні відновлювати вихідне форматування тексту, включно із зображеннями, колонками та іншими нетекстовими компонентами.

На рисунку 1.1 зображено основний принцип роботи оптичних систем розпізнавання тексту. Як можна бачити з цієї схеми, текст, з різних джерел нетекстової інформації, за допомогою системи оптичного розпізнавання перетворюється у текстову інформацію, та може у подальшому редагуватися стандартними текстовими програмами.

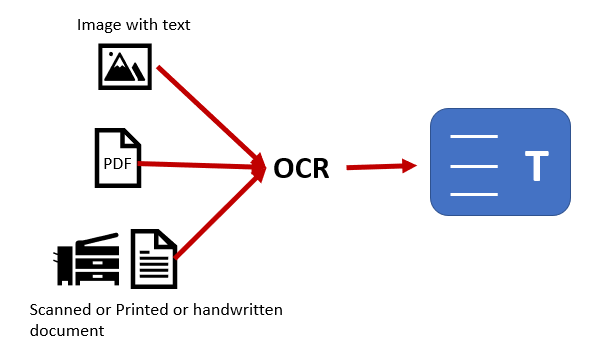


Рисунок 1.1 – Принцип роботи систем оптичного розпізнавання тексту

У чому полягають основні переваги OCR?

Текст із можливістю пошуку. Підприємства можуть перетворювати наявні та нові документи в базу знань з можливістю повноцінного пошуку. ПЗ для автоматичного опрацювання текстової бази дає змогу вдосконалювати базу знань підприємства.

Ефективність роботи. Застосування ПЗ OCR дає змогу підвищити ефективність роботи шляхом автоматичної інтеграції документообігу та цифрових робочих процесів. Ось кілька прикладів того, що може зробити ПЗ OCR:

Сканування заповнених вручну форм для автоматизованої перевірки, розгляду, редагування та аналізу. Такий підхід скорочує час ручного опрацювання документів і введення даних.

* Пошук необхідних документів за допомогою швидкого пошуку терміна в базі даних, замість ручного перебору файлів у скриньці.
* Перетворення рукописних нотаток на редаговані тексти та документи.

Нижче перелічено деякі поширені випадки використання OCR у різних галузях.

Банківська сфера використовує OCR для опрацювання та перевірки документів за кредитами, депозитних чеків та інших фінансових операцій. Така перевірка дала змогу підвищити ефективність боротьби з шахрайством і зміцнити безпеку транзакцій.

У системі охорони здоров'я OCR використовується для опрацювання історій хвороби пацієнтів, включно з лікувальними процедурами, аналізами, лікарняними картками та страховими виплатами. OCR допомагає оптимізувати робочий процес і скоротити обсяг ручної роботи в лікарнях, а також підтримувати актуальність записів. Наприклад, компанія забезпечує медичне страхування і щодня отримує тисячі заявок на виплату страхового відшкодування за отримання медичних послуг. Клієнти компанії можуть сфотографувати свій медичний рахунок і надіслати його через мобільний додаток. Технології розпізнавання тексту автоматично оброблюють ці зображення, що дозволяє компанії набагато швидше розглядати заявки.

Логістичні компанії використовують OCR для більш ефективного відстеження етикеток на упаковках, рахунків, квитанцій та інших документів. Наприклад, компанія може використовувати OCR для автоматизації обробки рахунків у SAP. Введення таких документів вручну забирало багато часу і призводило до помилок, оскільки співробітникам компанії доводилося вводити дані в кілька систем бухгалтерського обліку.

## 1.2 Особливості застосування методів аналізу даних і машинного навчання

Фахівці з аналізу даних класифікують різні види технологій OCR на основі їх використання та застосування. Наведемо лише деякі приклади:

* Програми простого оптичного розпізнавання символів.
* Програми інтелектуального розпізнавання символів.
* Інтелектуальне розпізнавання слів.
* Оптичне розпізнавання знаків

Розглянемо кожен видів окремо.

Програми простого оптичного розпізнавання символів. Простий механізм OCR застосовує безліч різних збережених шаблонів шрифтів і зображень тексту як шаблони. Програмне забезпечення OCR використовує алгоритми зіставлення шаблонів для посимвольного порівняння зображень тексту з внутрішньою базою даних. Підхід, за якого система зіставляє текст слово за словом, називається оптичним розпізнаванням слів. Він має свої обмеження, оскільки існує практично необмежена кількість шрифтів і стилів почерку, і кожен окремий тип не може бути врахований і збережений у базі даних.

Програми інтелектуального розпізнавання символів. Сучасні системи OCR використовують технологію інтелектуального розпізнавання символів (ICR) для зчитування тексту так само, як це робить людина. Вони використовують передові методи машинного навчання людських навичок читання. Система машинного навчання, звана нейронною мережею, аналізує текст на багатьох рівнях, багаторазово обробляючи зображення. Вона шукає різні атрибути зображення (криві, лінії, перетини і петлі) і об'єднує результати різних рівнів аналізу для отримання остаточного результату. Незважаючи на те, що ICR обробляє зображення за символами, процес не займає багато часу, а результати виходять за лічені секунди.

На рисунку 1.2 зображено умовну схему OCR на основі нейронної мережі та методів інтелектуальної обробки даних.

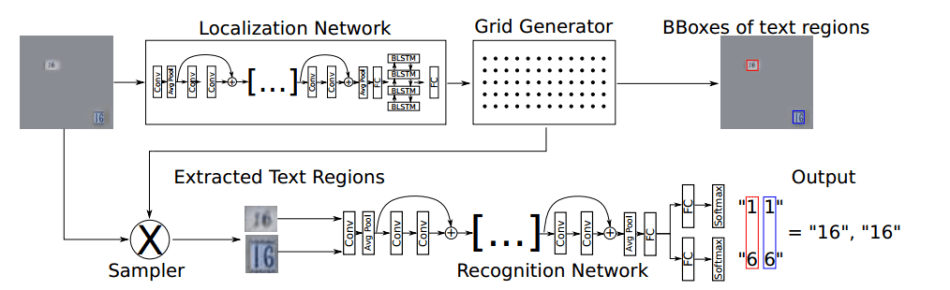


Рисунок 1.2 - приклад схеми OCR на основі інтелектуальних методів обробки даних

Інтелектуальне розпізнавання слів. Інтелектуальні системи розпізнавання слів працюють за тим самим принципом, що й ICR, але обробляють зображення цілих слів без попереднього виділення символів у зображенні.

Оптичне розпізнавання знаків. Оптичне розпізнавання знаків дає змогу ідентифікувати логотипи, водяні знаки та інші позначення в документі.

Також необхідно відмітити, що OCR часто є складовою інших рішень у сфері штучного інтелекту, які можуть впроваджувати підприємства. Наприклад, OCR може застосовуватися для сканування і розпізнавання номерних знаків і дорожніх покажчиків у самокерованих автомобілях, виявлення логотипів брендів у повідомленнях у соціальних мережах або ідентифікації упаковки продукту в рекламних зображеннях. Такі технології штучного інтелекту допомагають підприємствам ухвалювати більш ефективні маркетингові та операційні рішення, які дають змогу скоротити витрати і поліпшити якість обслуговування клієнтів.

# РОЗДІЛ 2 Розроблення моделі аналізу даних і машинного навчання та алгоритм її роботи

## 2.1 Визначення моделі та основних етапів обробки даних

Як було описано в попередньому розділі, сучасні системи оптичного розпізнавання тексту покладаються на методи інтелектуального оброблення даних, а зокрема на методи машинного навчання. Найчастіше застосовуються різні нейронні мережі для цих цілей.

У більшості подібних систем на вхід надходить зображення, яке містить текст, а на виході отримуємо сам текст, який система змогла виявити на даному зображенні. Однак однієї нейронної мережі недостатньо для виконання цього роду завдань. Перед тим як зображення надійде на вхід нейронної мережі, його необхідно попередньо обробити. І від типу вхідного зображення, а також типу системи розпізнавання тексту, кроки попереднього процесингу можуть сильно відрізняться. На рисунку 2.1 зображена умовна схема цієї обробки зображення.

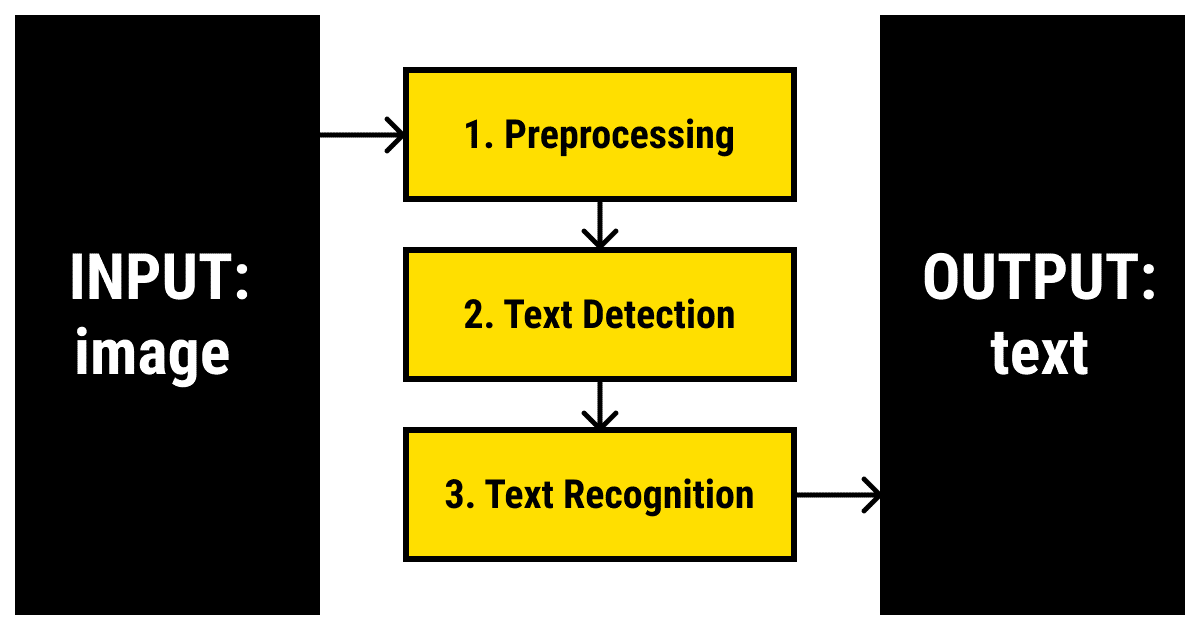


Рисунок 2.1 – Обробка зображення у OCR системі

Для підготовки зображення, перед передачею його в нейронну мережу, часто необхідно виконати такі кроки: знайти на зображенні текст, відокремити його від заднього фону, розділити текст на окремі слова, слова розділити на літери, поліпшити різкість окремих букв. Залежно від зображення і необхідних результатів для цих цілей також можуть застосовуватися нейронні мережі.

Однак у цій курсовій роботі було ухвалено рішення для попередньої обробки зображення використовувати простіші методи комп'ютерного зору, а саме бібліотеку OpenCV. За допомогою цієї технології буде виконано первинне опрацювання зображення, визначення літер та їхню подальшу модифікацію і коригування для передавання в нейронну мережу.

Для самого визначення тексту буде створена і навчена спеціальна нейронна мережа. Для створення нейронної мережі буде використовуватися широко використовувана бібліотека машинного навчання tensorflow. Саму ж нейронну мережу навчатимуть на спеціальному датасеті, що містить набір рукописних букв і цифр.

Для програмної реалізації моделі буде використовуватись мова програмування Python, тому що існує велика кількість готових та зручних рішень для обробки даних та створення нейронних мереж.

## 2.2 Первинна обробка даних та бібліотека OpenCV

OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом) - бібліотека алгоритмів комп'ютерного зору, опрацювання зображень і чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних і комерційних цілях - поширюється в умовах ліцензії BSD.

На рисунку 2.2 зображено приклад роботи бібліотеки OpenCV.



Рисунок 2.2 – Приклад використання OpenCV

За допомогою цієї бібліотеки можна виконувати значну кількість складних операції з зображеннями. На рисунку 2.3 зображено функцію, що зчитує картинку з файлу у форматі GRAYSCALE, відображає її на екрані, та запису у інший файл.

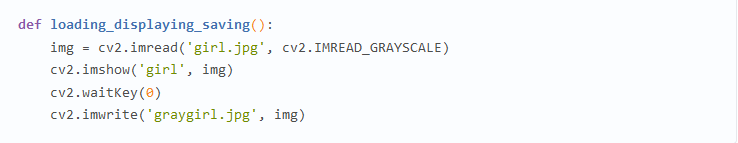


Рисунок 2.3 – Приклад функції з використанням OpenCV

Також для маніпуляцій та перетворення вхідних даних необхідно буде використовувати дуже розповсюджену бібліотеку у мові Python – NumPy. Необхідно відмітити, що реалізація OpenCV у мові Python у деяких випадках повертає об’єкти NumPy, таким чином ми можемо маніпулювати та змінювати дані.

NumPy - це open-source модуль для python, який надає загальні математичні та числові операції у вигляді пре-компільованих, швидких функцій. Вони об'єднуються у високорівневі пакети. Вони забезпечують функціонал, який можна порівняти з функціоналом MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з великими масивами та матрицями.

## 2.3 Розпізнавання тексту. Створення та тренування нейронної мережі за допомогою TensorFlow та Keras

Для розпізнавання тексту буде створено нейронну мережу, на вхід якої подаватиметься попередньо оброблене зображення букви, а на виході нейронної мережі буде відповідь у вигляді коду букви (рис. 2.4).

Для створення, використання та тренування нейронної мережі буде використовуватись бібліотека машинного навчання Tensorflow.

TensorFlow - відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для розв'язання завдань побудови та тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізовано для Python, також існують реалізації для R, C Sharp, C++, Haskell, Java, Go і Swift.

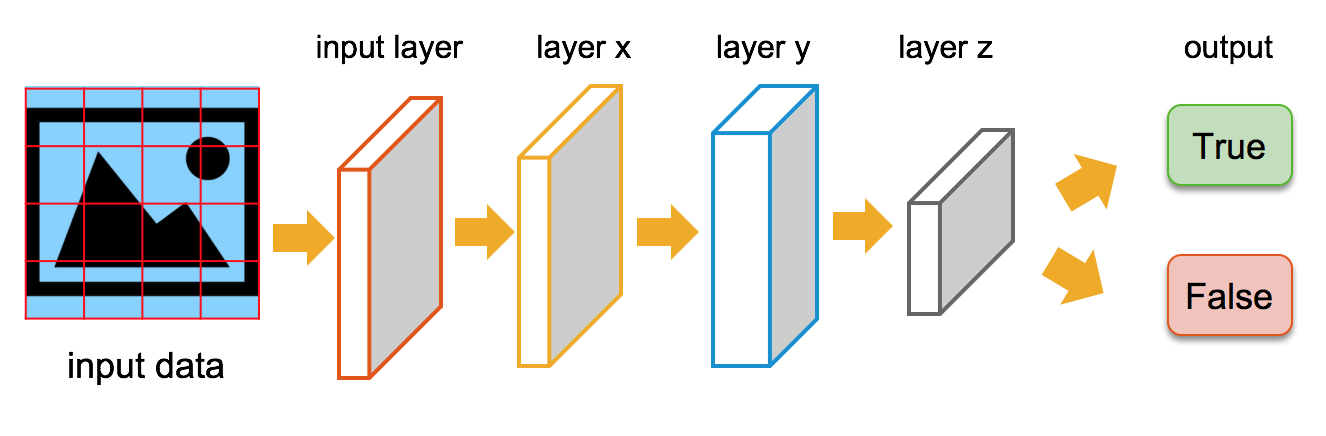


Рисунок 2.4 – Приклад нейронної мережі для обробки зображення

На рисунку 2.5 наведено приклад використання бібліотеки Tenserflow.

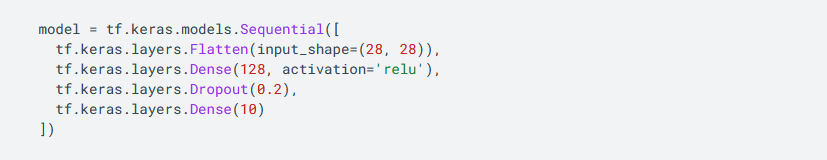


Рисунок 2.5 – Приклад створення простої нейронної мережі за допомогою Tenserflow

Keras - це програмна бібліотека з відкритим вихідним кодом, яка надає інтерфейс Python для штучних нейронних мереж. Keras виступає в якості інтерфейсу для бібліотеки TensorFlow.

До версії 2.3 Keras підтримував декілька бекендів, включаючи TensorFlow, Microsoft Cognitive Toolkit, Theano та PlaidML. Починаючи з версії 2.4, підтримується тільки TensorFlow. Розроблений для швидкого експериментування з глибокими нейронними мережами, він зосереджений на тому, щоб бути зручним для користувача, модульним та розширюваним. Вона була розроблена в рамках дослідницького проекту ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором і супровідником є Франсуа Шолле (François Chollet), інженер Google. Шолле також є автором моделі глибокої нейронної мережі Xception.

Keras містить численні реалізації загальновживаних нейромережевих будівельних блоків, таких як шари, цілі, функції активації, оптимізатори, а також безліч інструментів для полегшення роботи з графічними та текстовими даними для спрощення кодування, необхідного для написання коду глибокої нейронної мережі. Код розміщено на GitHub, а форуми підтримки спільноти включають сторінку питань GitHub та канал Slack.

На додаток до стандартних нейронних мереж, Keras має підтримку згорткових та рекурентних нейронних мереж. Він підтримує інші поширені утиліти, такі як відсіювання, пакетна нормалізація та об'єднання.

Keras дозволяє користувачам створювати глибокі моделі на смартфонах (iOS та Android), в Інтернеті або на віртуальній машині Java.[3] Він також дозволяє використовувати розподілене навчання моделей глибокого навчання на кластерах графічних процесорів (GPU) та тензорних процесорів (TPU).

Для навчання нейронної мережі було вирішено використати датасет із набором рукописних літер EMNIST. Згодом це може обмежити створену модель під час розпізнавання друкованих шрифтів, але під час виконання цієї роботи це обмеження не є суттєвим. Хоча в разі поліпшення або зміни моделі, можна використовувати інший датасет.

# РОЗДІЛ 3 Реалізація моделі та апробація інтелектуальної системи

Як було розглянуто в попередньому розділі, процес інтелектуального опрацювання даних складатиметься з кількох пов'язаних етапів. Перший етап - це завантаження і попереднє опрацювання зображення, виділення тексту і літер, приведення виділених літер у необхідний формат для подальшого опрацювання. Наступним кроком буде передача вихідних даних із попереднього етапу до нейромережі, однак необхідно зазначити, що на цьому етапі вже має бути навчена нейромережа. Отже, одним із кроків перед створенням повноцінної інтелектуальної системи оброблення даних є створення моделі нейромережі та навчання її на тестових даних.

Першим кроком розіб'ємо текст на окремі літери. Для цього знадобиться OpenCV, точніше його функція findContours. Відкриємо зображення (cv2.imread), переведемо його в ч/б (cv2.cvtColor + cv2.threshold), злегка збільшимо (cv2.erode).

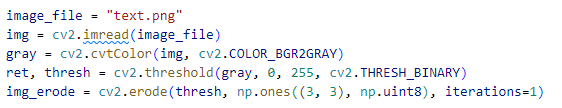


Рисунок 3.1 – Завантаження та первинна обробка зображення

Наступним кроком знайдемо контуру в зображенні, виділимо їх, та виведімо на екран користувача для більшої наочності.

Ми отримуємо ієрархічне дерево контурів (параметр cv2.RETR\_TREE). Першим іде загальний контур картинки, потім контури букв, потім внутрішні контури. Нам потрібні тільки контури букв, тому перевіряється, що батьківським є загальний контур. Це спрощений підхід, і для реальних сканів це може не спрацювати, хоча для розпізнавання скріншотів це некритично, тому було вирішено використовувати цей підхід.

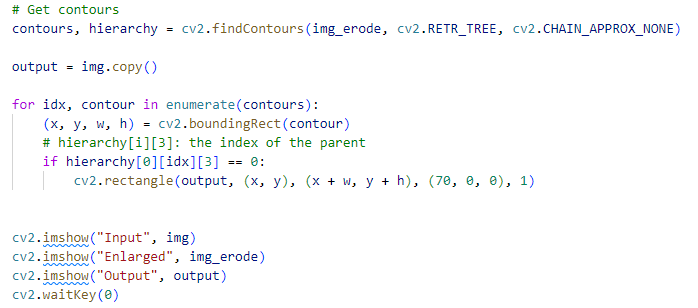


Рисунок 3.2 – Пошук контурів у зображені

Результат виконання віще зазначеного коду показано на рисунку 3.3.

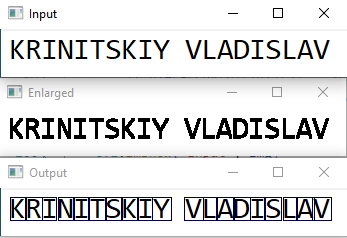


Рисунок 3.3 – Результат пошуку контурів

Наступним кроком збережемо кожну літеру, попередньо відмасштабувавши її до квадрата 28х28 (надалі цей крок буде необхідний, адже саме в такому форматі зберігається база MNIST, на основі якої буде навчена нейромережа). OpenCV побудований на базі numpy, тож ми можемо використовувати функції роботи з масивами для кропу і масштабування.

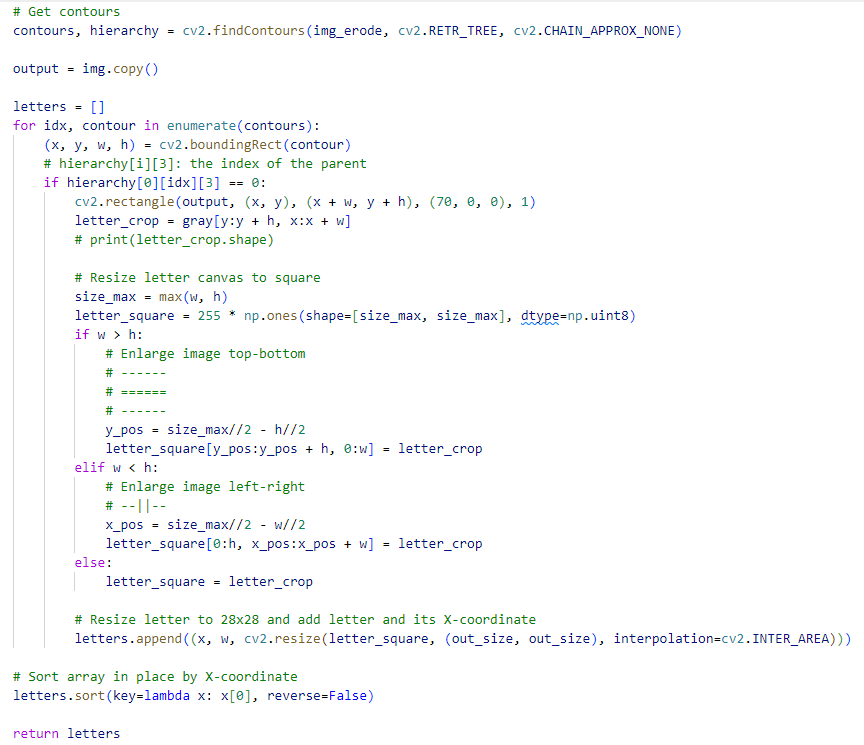


Рисунок 3.4 – Код обробки кожної окремої літери

Для перегляду отриманих результатів виведемо перши шість літер та будимо впевнені, що алгоритм обробки зображення працює правильно.



Рисунок 3.5 – Окремі літери після попередньої обробки зображення

Для навчання нейронної мережі буде використовуватися EMNIST датасет, який містить зображення рукописних латинських літер і цифр (A..Z, 0..9) та інших часто використовуваних символів. Загальна кількість символів дорівнює 62.

Перед створенням нейромережі необхідно визначити спеціальний масив, який відповідатиме виходам нейронної мережі та датасету EMNIST.



Рисунок 3.6 - Оголошення необхідних даних

Нейронна мережа відповідно, має 62 виходи, на вході вона отримуватиме зображення 28х28, після розпізнавання "1" буде на відповідному виході мережі.

Створюємо модель мережі.

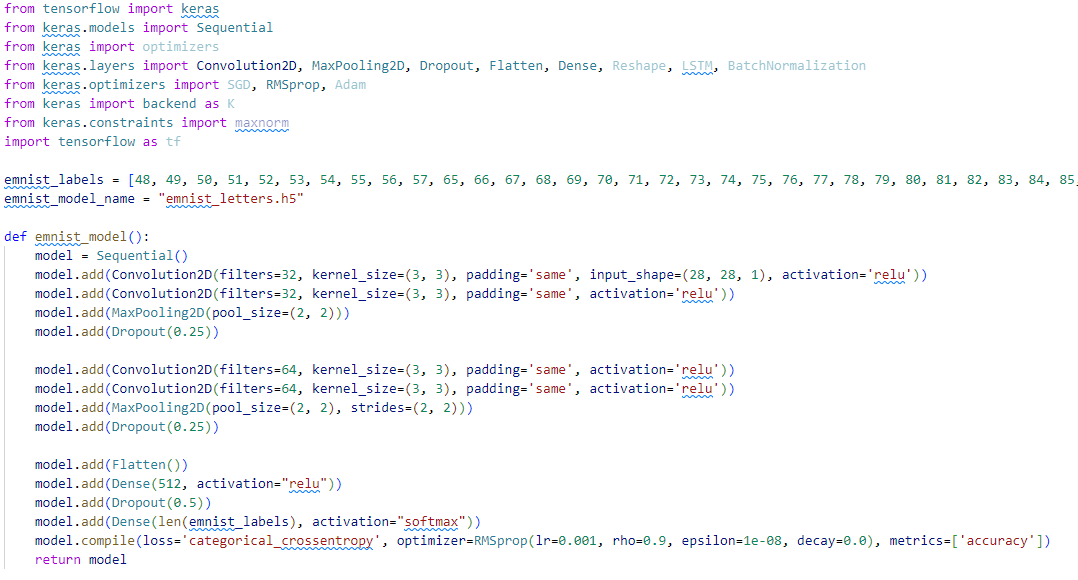


Рисунок 3.7 – Функція створення моделі нейронної мережі

Як можна бачити, це класична згорткова мережа, що виокремлює певні ознаки зображення (кількість фільтрів 32 і 64), до "виходу" якої приєднана "лінійна" мережа MLP, що формує остаточний результат.

Наступним етапом є навчання нейронної мережі. Для цього необхідно використовувати датасет для навчання, як було раніше зазначено, у даній курсовій роботі буде використано датасет із рукописними символами EMNIST.

Для читання бази скористаємося бібліотекою idx2numpy. Підготуємо дані для навчання і валідації.



Рисунок 3.8 – Підготовка даних для навчання

Також з метою економії часу використовується лише 1/10 датасету для навчання (параметр k), інакше процес займе не менше 10 годин. Для виконання цієї курсової роботи отримана точність після обробки 1/10 від набору даних є задовільною. Запускаємо навчання мережі, наприкінці процесу зберігаємо навчену модель на диск.

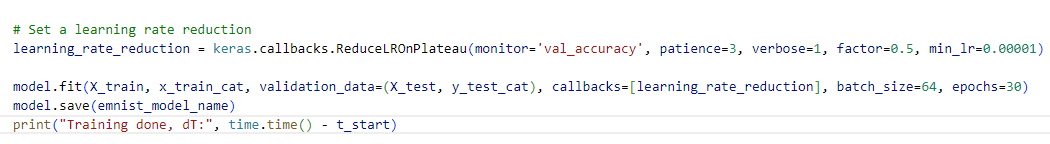


Рисунок 3.9 – Навчання нейронної мережі та збереження її на диск

Сам процес навчання займає близько півгодини:

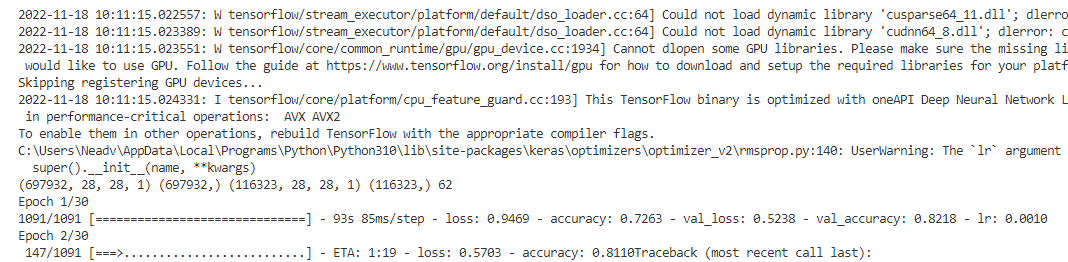


Рисунок 3.10 – Процес навчання нейронної мережі

Наступним кроком необхідно створити функцію для визначення літери на зображенні. Зображення в датасеті спочатку були повернуті, тож нам доводиться повернути картинку перед розпізнаванням.

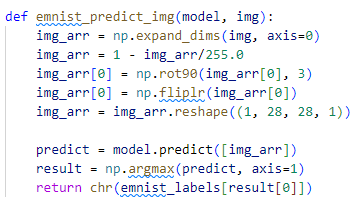


Рисунок 3.11 – Функція визначення літери по зображенню

Останнім кроком необхідно створити функцію, яка отримує зображення, обробляє його, а далі передає нейронній мережі для визначення тексту. На виході цієї функції буде рядок із текстом, що містився на зображенні.

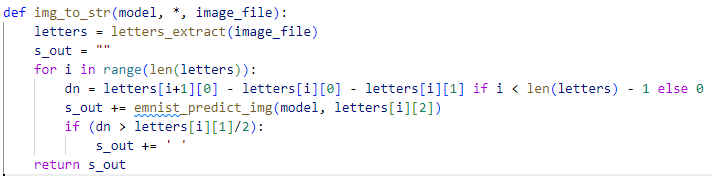


Рисунок 3.13 – Функція для перетворення зображення в текст

Приклад використання:

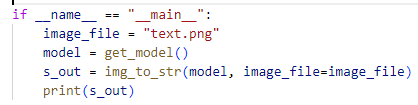


Рисунок 3.14 – Приклад використання отриманої моделі

Результат:



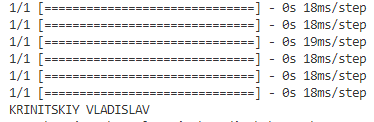


Рисунок 3.15 – Результат роботи

# ВИСНОВОК

У результаті курсової роботи було створено модель інтелектуальної системи обробки даних для розпізнавання тексту із зображення. У межах роботи було проведено дослідження предметної області, сформовано вимоги для розроблюваної системи, створено інтелектуальну модель оброблення даних, розроблено та протестовано саму систему за цією моделлю.

Під час цієї курсової роботи було розглянуто застосування принципів машинного навчання під час розпізнавання тексту. Глибоке навчання почало набирати популярність в останньому десятилітті XX століття. Сьогодні це один з найбільш затребуваних методів ML, який надає можливості, недоступні іншим архітектурам. Алгоритми OCR, побудовані за допомогою нейронних мереж, дають змогу по-новому поглянути на стару проблему і відродити інтерес як власників бізнесу, так і інженерів ML.

У результаті було створено модель інтелектуальної обробки даних для оптичного розпізнавання тексту. Було створено і навчено нейромережу для виконання даного завдання. Так само були розглянуті методи подальшого поліпшення і модифікації системи.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Тронь В.В. Методичні вказівки до виконання курсової роботи з дисципліни «Методи і технології інтелектуального аналізу даних» для студентів спеціальності 122 “Комп’ютерні науки”, 2022. 20с.
2. Лутц М. Изучаем Python. СПб: Символ-Плюс, 2011. 1280 с.
3. Саттон Р. С., Барто Э. Г.Обучение с подкреплением. М.: Бином, 2012. 399 с.
4. Мюллер А., С. Гвідо. Введення в машинне навчання за допомогою Python. М .: Вільямс. 2017. 465 с.
5. TensorFlow Guide. URL: <https://www.tensorflow.org/guide> (Дата звертання: 11.11.2022)
6. NumPy Documentation. URL: <https://numpy.org/> (Дата звертання: 11.11.2022)
7. OpenCV Python Tutorials. URL: <https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html> (Дата звертання: 12.11.2022)
8. Keras Getting started. URL: <https://keras.io/getting_started/> (Дата звертання: 12.11.2022)
9. STN-OCR: A single Neural Network for Text Detection and Text Recognition. URL: <https://medium.com/syncedreview/stn-ocr-a-single-neural-network-for-text-detection-and-text-recognition-220debe6ded4> (Дата звертання: 13.11.2022)
10. OCR with Keras, TensorFlow, and Deep Learning. URL: <https://pyimagesearch.com/2020/08/17/ocr-with-keras-tensorflow-and-deep-learning/> (Дата звертання: 13.11.2022)
11. OCR with Deep Learning: The Curious Machine Learning Case. URL: <https://labelyourdata.com/articles/ocr-with-deep-learning>   
    (Дата звертання: 13.11.2022)

# ДОДАТОК А ВИХІДНИЙ КОД ПРОГРАМИ

Посилання на GitHub-репозиторій: <https://github.com/Neadv/MTIAD--Terms>

**main.py:**

import cv2

import numpy as np

from emnist\_model import emnist\_labels, get\_model

def letters\_extract(image\_file: str, out\_size=28):

    img = cv2.imread(image\_file)

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY)

    img\_erode = cv2.erode(thresh, np.ones((3, 3), np.uint8), iterations=1)

    # Get contours

    contours, hierarchy = cv2.findContours(img\_erode, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

    output = img.copy()

    letters = []

    for idx, contour in enumerate(contours):

        (x, y, w, h) = cv2.boundingRect(contour)

        # hierarchy[i][3]: the index of the parent

        if hierarchy[0][idx][3] == 0:

            cv2.rectangle(output, (x, y), (x + w, y + h), (70, 0, 0), 1)

            letter\_crop = gray[y:y + h, x:x + w]

            # print(letter\_crop.shape)

            # Resize letter canvas to square

            size\_max = max(w, h)

            letter\_square = 255 \* np.ones(shape=[size\_max, size\_max], dtype=np.uint8)

            if w > h:

                # Enlarge image top-bottom

                # ------

                # ======

                # ------

                y\_pos = size\_max//2 - h//2

                letter\_square[y\_pos:y\_pos + h, 0:w] = letter\_crop

            elif w < h:

                # Enlarge image left-right

                # --||--

                x\_pos = size\_max//2 - w//2

                letter\_square[0:h, x\_pos:x\_pos + w] = letter\_crop

            else:

                letter\_square = letter\_crop

            # Resize letter to 28x28 and add letter and its X-coordinate

            letters.append((x, w, cv2.resize(letter\_square, (out\_size, out\_size), interpolation=cv2.INTER\_AREA)))

    # Sort array in place by X-coordinate

    letters.sort(key=lambda x: x[0], reverse=False)

    return letters

def emnist\_predict\_img(model, img):

    img\_arr = np.expand\_dims(img, axis=0)

    img\_arr = 1 - img\_arr/255.0

    img\_arr[0] = np.rot90(img\_arr[0], 3)

    img\_arr[0] = np.fliplr(img\_arr[0])

    img\_arr = img\_arr.reshape((1, 28, 28, 1))

    predict = model.predict([img\_arr])

    result = np.argmax(predict, axis=1)

    return chr(emnist\_labels[result[0]])

def img\_to\_str(model, \*, image\_file):

    letters = letters\_extract(image\_file)

    s\_out = ""

    for i in range(len(letters)):

        dn = letters[i+1][0] - letters[i][0] - letters[i][1] if i < len(letters) - 1 else 0

        s\_out += emnist\_predict\_img(model, letters[i][2])

        if (dn > letters[i][1]/2):

            s\_out += ' '

    return s\_out

if \_\_name\_\_ == "\_\_main\_\_":

    image\_file = "text.png"

    model = get\_model()

    s\_out = img\_to\_str(model, image\_file=image\_file)

    print(s\_out)

**emnist\_model.py:**

import numpy as np

import idx2numpy

import time

from tensorflow import keras

from keras.models import Sequential

from keras import optimizers

from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten, Dense, Reshape, LSTM, BatchNormalization

from keras.optimizers import SGD, RMSprop, Adam

from keras import backend as K

from keras.constraints import maxnorm

import tensorflow as tf

emnist\_labels = [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122]

emnist\_model\_name = "emnist\_letters.h5"

def emnist\_model():

    model = Sequential()

    model.add(Convolution2D(filters=32, kernel\_size=(3, 3), padding='same', input\_shape=(28, 28, 1), activation='relu'))

    model.add(Convolution2D(filters=32, kernel\_size=(3, 3), padding='same', activation='relu'))

    model.add(MaxPooling2D(pool\_size=(2, 2)))

    model.add(Dropout(0.25))

    model.add(Convolution2D(filters=64, kernel\_size=(3, 3), padding='same', activation='relu'))

    model.add(Convolution2D(filters=64, kernel\_size=(3, 3), padding='same', activation='relu'))

    model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2)))

    model.add(Dropout(0.25))

    model.add(Flatten())

    model.add(Dense(512, activation="relu"))

    model.add(Dropout(0.5))

    model.add(Dense(len(emnist\_labels), activation="softmax"))

    model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0), metrics=['accuracy'])

    return model

def emnist\_train(model, \*, path):

    t\_start = time.time()

    X\_train = idx2numpy.convert\_from\_file(path + 'emnist-byclass-train-images-idx3-ubyte')

    y\_train = idx2numpy.convert\_from\_file(path + 'emnist-byclass-train-labels-idx1-ubyte')

    X\_test = idx2numpy.convert\_from\_file(path + 'emnist-byclass-test-images-idx3-ubyte')

    y\_test = idx2numpy.convert\_from\_file(path + 'emnist-byclass-test-labels-idx1-ubyte')

    X\_train = np.reshape(X\_train, (X\_train.shape[0], 28, 28, 1))

    X\_test = np.reshape(X\_test, (X\_test.shape[0], 28, 28, 1))

    print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape, len(emnist\_labels))

    # Test:

    k = 10

    X\_train = X\_train[:X\_train.shape[0] // k]

    y\_train = y\_train[:y\_train.shape[0] // k]

    X\_test = X\_test[:X\_test.shape[0] // k]

    y\_test = y\_test[:y\_test.shape[0] // k]

    # Normalize

    X\_train = X\_train.astype(np.float32)

    X\_train /= 255.0

    X\_test = X\_test.astype(np.float32)

    X\_test /= 255.0

    x\_train\_cat = keras.utils.to\_categorical(y\_train, len(emnist\_labels))

    y\_test\_cat = keras.utils.to\_categorical(y\_test, len(emnist\_labels))

    # Set a learning rate reduction

    learning\_rate\_reduction = keras.callbacks.ReduceLROnPlateau(monitor='val\_accuracy', patience=3, verbose=1, factor=0.5, min\_lr=0.00001)

    model.fit(X\_train, x\_train\_cat, validation\_data=(X\_test, y\_test\_cat), callbacks=[learning\_rate\_reduction], batch\_size=64, epochs=30)

    print("Training done, dT:", time.time() - t\_start)

def get\_model():

    return keras.models.load\_model(emnist\_model\_name)

if \_\_name\_\_ == "\_\_main\_\_":

    model = emnist\_model()

    emnist\_train(model, path="./training\_data/")

    model.save(emnist\_model\_name)