ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Метод розпізнавання рукописних математичних формул

КВАЛІФІКАЦІЙНА РОБОТА

Другий (магістерський)

Автор
Шаповалов М.І.
студ. гр. КСМм 21-1

Керівник
Кучук Н.Г.
проф. каф. ЕОМ

Актуальність розробки

Застосунок для розпізнавання рукописних математичних формул дає змогу оптимізувати час шляхом надання можливості швидкого і зручного переносу формул з рукописного у друкований вигляд для подальших обчислень

Об’єкт та предмет дослідження

Об’єктом дослідження є процес розпізнавання рукописного тексту.

Предметом дослідження є засоби та методи побудови нейронної мережі із використанням сучасних технологій

Мета роботи

Метою роботи є розробка додатку для розпізнавання рукописних математичних формул, що полегшить переведення рукописів у друкований вигляд та допоможе людям зекономити власний час.

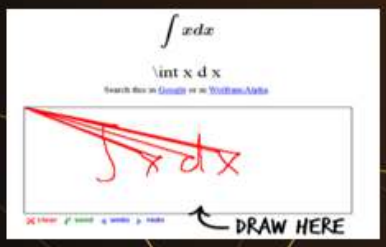
Для досягнення поставленої мети були поставлені такі задачі:
1) провести аналітичний огляд існуючих додатків;
2) обрати мову програмування, підходи до розробки застосунку та сучасні технології програмування;
3) реалізувати застосунок з розпізнавання рукописних математичних формул;
4) виконати тестування розробленого застосунку.

Огляд аналогів

Detexify

Google Docs

Photomath

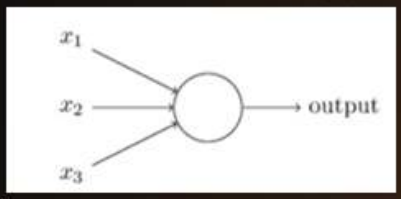


Mathematical Expression Recognit

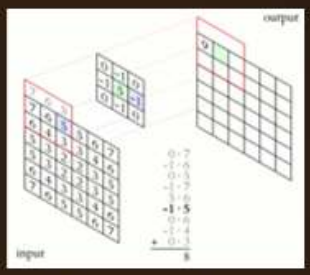
Mathpix

5

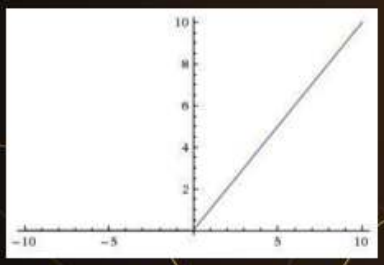
Огляд теорії нейронної мережі



Стандартний вигляд персептрона



Згортання зображення



Активаційна функція ReLU

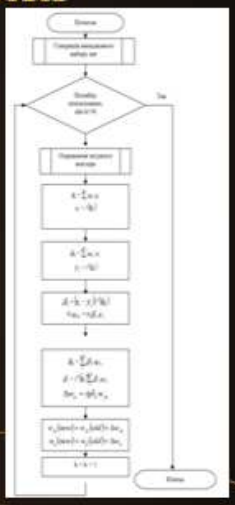
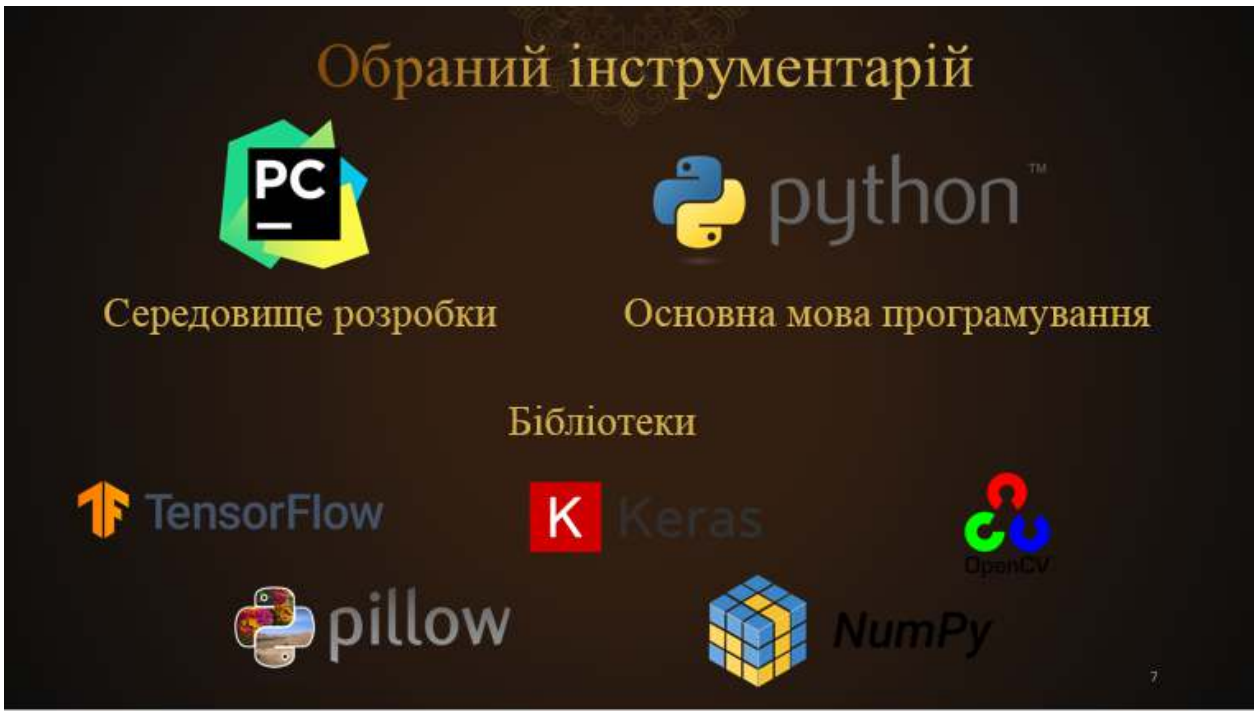
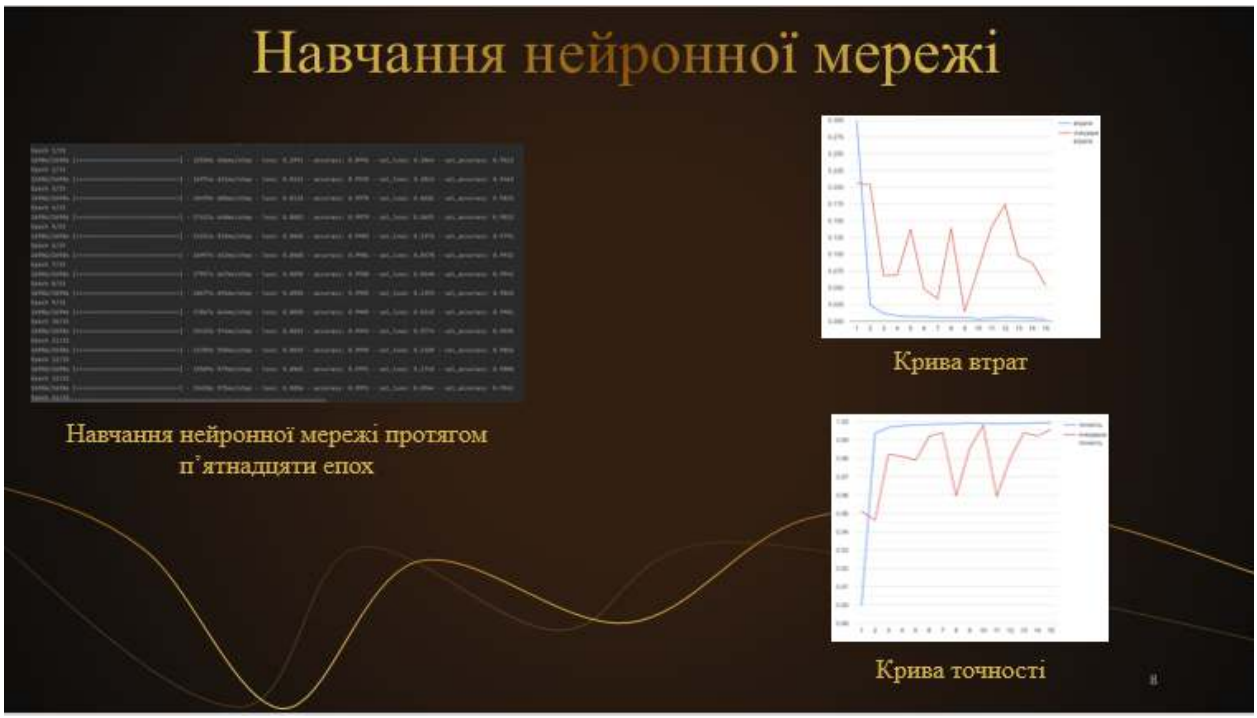


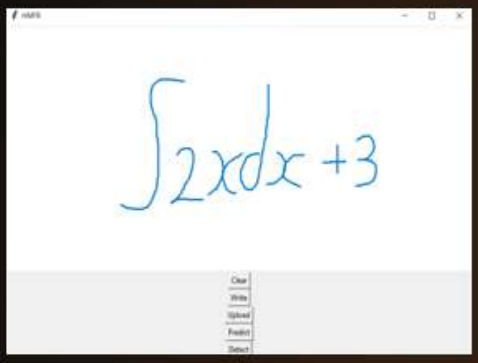
Схема алгоритму зворотнього поширення помилки

6

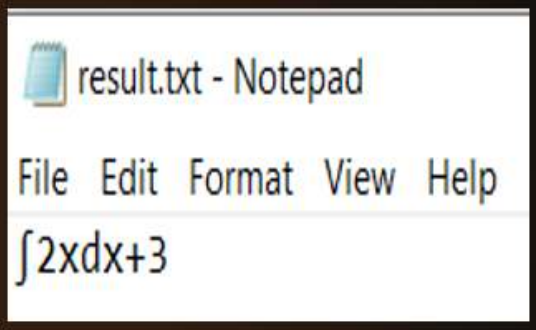

Обраний інструментарій
Середовище розробки
Основна мова програмування
Бібліотеки
TensorFlow
Keras
OpenCV
pillow
NumPy
7


Навчання нейронної мережі
Навчання нейронної мережі протягом п'ятнадцяти епох
Крива втрат
Крива точності
8

Результати розробки



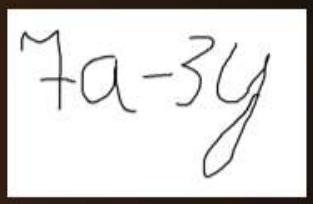
Введений за допомогою мишки математичний вираз



Розпізнаний математичний вираз

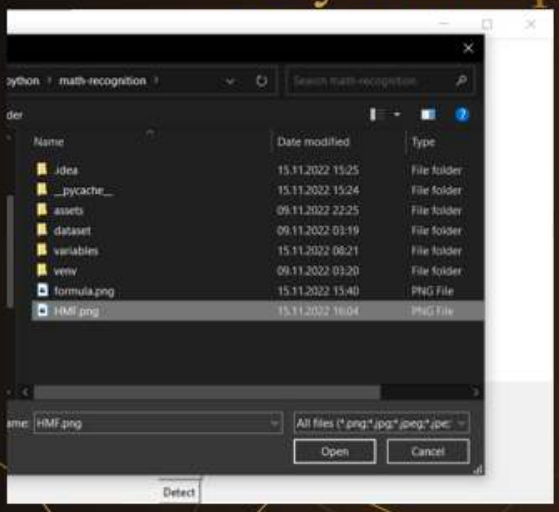
Результати розробки

Застосунок після натискання кнопки “Clean”

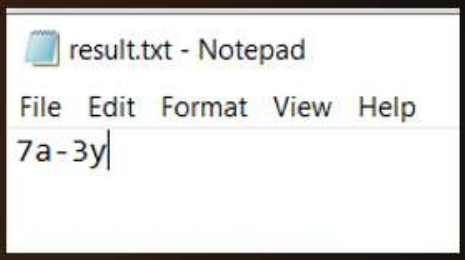


Зображення рукописного математичної формули

Результати розробки



Вибір файлу HMF.png із рукописним математичним виразом для подальшого розпізнавання



Розпізнаний математичний вираз у файлі result.txt

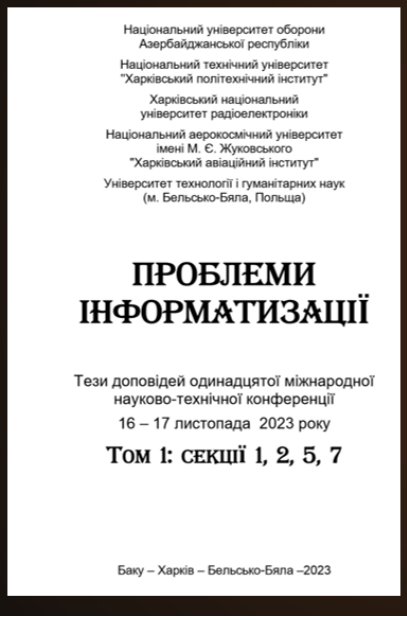
11

Висновки

- було проведено аналіз існуючих застосунків з розпізнавання рукописних математичних формул;
- було обрано інструментарій розробки програмного продукту;
- було створено та навчено нейронну мережу;
- було розроблено застосунок з розпізнавання рукописних математичних формул.

12

Апробація результатів кваліфікаційної роботи



Національний університет оборони
Азербайджанської республіки

Національний технічний університет
"Харківський політехнічний інститут"

Харківський національний
університет радіоелектроніки

Національний аерокосмічний університет
імені М. Є. Жуковського
"Харківський авіаційний інститут"

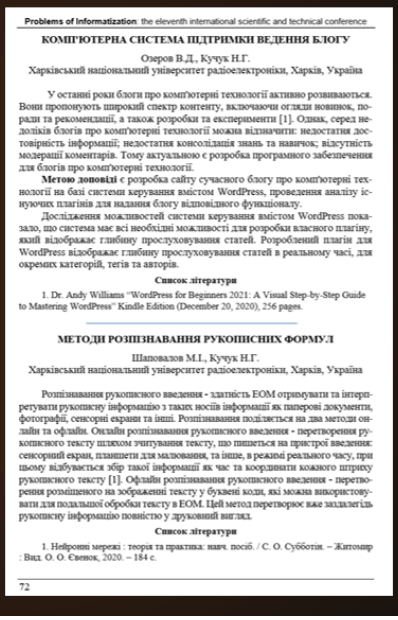
Університет технології і гуманітарних наук
(м. Бельсько-Бяла, Польща)

ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ

Тези доповідей одинадцятої міжнародної
науково-технічної конференції
16 – 17 листопада 2023 року

Том 1: секції 1, 2, 5, 7

Баку – Харків – Бельсько-Бяла –2023



КОМП'ЮТЕРНА СИСТЕМА ПІДТРИМКИ ВЕДЕННЯ БЛОГУ

Озеров В.Д., Кучук Н.Г.
Харківський національний університет радіоелектроніки, Харків, Україна

У останні роки блоги про комп'ютерні технології активно розвиваються. Вони пропонують широкий спектр контенту, включаючи огляди новинок, поради та рекомендації, а також розробки та експерименти [1]. Однак, серед недоліків блогів про комп'ютерні технології можна відзначити: недостатня достовірність інформації; недостатня консолідація знань та навичок; відсутність модерації коментарів. Тому актуальною є розробка програмного забезпечення для блогів про комп'ютерні технології.

Метою доповіді є розробка сайту сучасного блогу про комп'ютерні технології на базі системи керування вмістом WordPress, проведення аналізу існуючих плагінів для надання блогу відповідного функціоналу.

Дослідження можливостей системи керування вмістом WordPress показало, що система має всі необхідні можливості для розробки власного плагіну, який відображає глибину прослуховування статей. Розроблений плагін для WordPress відображає глибину прослуховування статей в реальному часі, для окремих категорій, тегів та авторів.

Список літератури

1. Dr. Andy Williams "WordPress for Beginners 2021: A Visual Step-by-Step Guide to Mastering WordPress" Kindle Edition (December 20, 2020), 256 pages.

МЕТОДИ РОЗПІЗНАВАННЯ РУКОПИСНИХ ФОРМУЛ

Шаповалов М.І., Кучук Н.Г.
Харківський національний університет радіоелектроніки, Харків, Україна

Розпізнавання рукописного введення - здатність ЕОМ отримувати та інтерпретувати рукописну інформацію з таких носіїв інформації як паперові документи, фотографії, сенсорні екрани та інші. Розпізнавання поділяється на два методи онлайн та офлайн. Онлайн розпізнавання рукописного введення - перетворення рукописного тексту шляхом зчитування тексту, що пишеться на пристрої введення: сенсорний екран, планшети для малювання, та інше, в режимі реального часу, при цьому відбувається збір такої інформації як час та координати кожного штриху рукописного тексту [1]. Офлайн розпізнавання рукописного введення - перетворення розміщеного на зображенні тексту у буквені коди, які можна використовувати для подальшої обробки тексту в ЕОМ. Цей метод перетворює вже зазделегідь рукописну інформацію повністю у друковий вигляд.

Список літератури

1. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.

ДОДАТОК Б

Лістинг програми predict_symbols.py

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
import cv2
from class_to_latexCommand_dictionary import class_to_latexCommand
filepath = "."  # file path of the model file (.bp file)
# The model is moved here, to not load the model at every time we initialize the object.
model = load_model(filepath, compile=True)  # Load the model

class SymbolPredictor:
    def __init__(self, symbol_detector):

        self.symbol_detector = symbol_detector
        self.image_of_symbol_to_predict = None
        self.np_array_of_symbols_to_predict = []
        self.symbol_classes_list = []
        self.latex_commands_list = []

    def preprocess_images(self, images):
        # img_array = cv2.imread('dataset/product/15.png', cv2.IMREAD_GRAYSCALE)  #
convert to array
        # TODO: check size before resizing
        # TODO: import image size instead of using the hardcoded value
        # TODO: Why the image here still need to be grayed?
        if images is None:
            raise ValueError("images passed to preprocess_images() is None!")
        for image in images:
            print(f"Number of images to preprocess: {len(images)}")
            resized_image = cv2.resize(image, (50, 50))
            image_gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
            np_array_image = np.array(image_gray).reshape(-1, 50, 50, 1)
            if len(np_array_image) != 1:
                raise RuntimeError(f"The image must be in Gray Scale not RGB.")
            self.np_array_of_symbols_to_predict.append(np_array_image)

    def predict_symbol_class(self):
        # predict
        if self.np_array_of_symbols_to_predict is not None:
            for symbol in self.np_array_of_symbols_to_predict:
                predictions = model.predict(symbol)
                print(f"Prediction: {predictions}")
                # Get classes from predicition
                # np.argmax() returns an array with 1 element which is the index of the
max value
                symbol_class = np.argmax(predictions, axis=1)
                print(f"Symbole Class: {symbol_class}")
                self.symbol_classes_list.append(str(symbol_class[0]))
        else:
            raise ValueError("np_array_of_symbols_to_predict is None! No image has been
passed to the preprocess_image() method")

    def convert_class_to_latex_command(self):
        for symbol_class in self.symbol_classes_list:
            latex_command = class_to_latexCommand[str(symbol_class)]
            self.latex_commands_list.append(latex_command)
            print(latex_command)

    def predict(self):
        self.__init__(symbol_detector=self.symbol_detector)
        self.preprocess_images(images=self.symbol_detector.detected_symbols_list)
        self.predict_symbol_class()
self.convert_class_to_latex_command()
```

ДОДАТОК В

Лістинг програми model.py

```
import numpy as np
import os
import cv2
from tqdm import tqdm
import pickle

from tensorflow.keras.models import Sequential, save_model
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.optimizers import Adam

# Load the dataset
DATADIR = "dataset"
CATEGORIES = ["int", "sum", "product", "sqrt", "a", "b", "x", "y"]
training_data = []

# Model configuration
IMG_SIZE = 50
no_classes = 8

# Check the Data
# for category in CATEGORIES:  # do dogs and cats
#     path = os.path.join(DATADIR, category)  # create path to dogs and
cats
#     for img in os.listdir(path):  # iterate over each image per dogs
and cats
#                     img_array  =  cv2.imread(os.path.join(path,  img),
cv2.IMREAD_GRAYSCALE)  # convert to array
#         print(img_array)
#         plt.imshow(img_array, cmap='gray')  # graph it
#         plt.show()  # display!
#         break
#     break


def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num    =    CATEGORIES.index(category)       #    get    the
classification  (0, 1 or 2)
        for img in tqdm(os.listdir(path)):  # iterate over each image
            try:
                img_array    =    cv2.imread(os.path.join(path,    img),
cv2.IMREAD_GRAYSCALE)  # convert to array
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
# resize to normalize data size
                training_data.append([new_array,  class_num])    #  add
this to our training_data
            except Exception as e:  # in the interest in keeping the
output clean...
                raise RuntimeError(f"Exception happened: {e}")
    return training_data
```

```
create_training_data()

# Create dataset model
X = []
y = []
# y = np.array(y)

for features, label in training_data:
    X.append(features)  # X contains the images
    y.append(label)     # y contains the classes
    # np.array((y, label))
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)   # create an np
array
y = np.array(y).reshape(-1, 1)

print(f"Shape of X: {X.shape}")
print(f"Shape of Y: {y.shape}")

print(f"Type of X: {type(X)}")

# Save the data set ina pickle file
pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()

# Load data set
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)

# Rescale Data
X = X/255.0

# Configure dataset for performace


# Define the Model
model = Sequential()
model.add(Conv2D(2500,     kernel_size=(3,     3),     activation='relu',
input_shape=X.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2)))
# model.add(Dropout(0.25))

model.add(Conv2D(250, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# model.add(Dropout(0.25))
# Now we have a 64x2 Model (64 neurons in the input and 2 layers)

model.add(Flatten())  # Flatten the data because Convolution is 2D and
Dense is 1D dataset layer
model.add(Dense(32, activation='relu'))

# Output layer
model.add(Dense(no_classes, activation='softmax'))

# Compile the Model
model.compile(loss=sparse_categorical_crossentropy,    optimizer=Adam(),
metrics=['accuracy'])
```

```
# Fit data to model
# The Batch size and validation split depend on the size of the dataset
# Epochs are the number of times of repeating the training process, so
the more epochs, the better, upon that the mode
# is not overtrained
model.fit(X,      y,      batch_size=3,      epochs=15,      verbose=1,
validation_split=0.05)

# Evaluate the model
score = model.evaluate(X, y, verbose=0)
print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')

# save the model
print("Saving the model")
filepath = '.'
save_model(model, filepath)
```