ДОДАТОК А

Вихідний код програмної моделі

```
"""# Imports"""
from os import listdir
import numpy as np
from IPython.core.display import display, HTML
from numpy import array
from tensorflow.keras.utils import to_categorical
from keras.layers import TimeDistributed, Embedding, concatenate, RepeatVector, Reshape, Dense, Flatten, LSTM, Input
from keras.preprocessing.image import load_img, array_to_img, img_to_array
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer, one_hot
from keras.callbacks import ModelCheckpoint
from keras.applications.inception_resnet_v2 import preprocess_input, InceptionResNetV2
from keras.models import Model, Sequential, model_from_json, load_model
from keras.layers.core import Dense, Dropout, Flatten
from tensorflow.keras.optimizers import RMSprop
from keras.layers.convolutional import Conv2D

"""# Functions and constants"""
images_path = "samples_img/"
html_path = "samples/"

# Max number of tokens in sequence
max_caption_len = 265
```

```
    def downloadFile(filename):
        file = open(filename, 'r')
        text = file.read()
        file.close()
        return text

    # convert word to identifier
    def wordToId(integer, tokenizer):
        for word, index in tokenizer.word_index.items():
            if index == integer:
                return word
        return None

    # convert HTML-code from image
    def convertImgToHTML(model, tokenizer, photo):
        features = np.array(IR2.predict(np.array(
                [preprocess_input(np.array(img_to_array(photo),
dtype=float))])))
        in_text = '<html>' # first word of HTML-code
        print(in_text)
        for i in range(5000):
                                                sequence     =
pad_sequences([tokenizer.texts_to_sequences
([in_text])[0][-max_caption_len:]], maxlen=max_caption_len)
            # predict next word
                    yhat  =  model.predict([features,sequence],
verbose=0)
            yhat = np.argmax(yhat)
            # get word identifier
            word = wordToId(yhat, tokenizer)
            # stop when word cannot be predicted
            if word is None:
```

```
                break
            in_text += ' ' + word
            print(' ' + word, end='')
            #print("word is: '" + word + "'")
            # stop if we predict the end of the sequence
            if word.str.contains('</html>'):
                break
        return in_text

    """# Data preparing"""
    # Get images for training
    images = []
    all_filenames = listdir(drive_path+images_path)
    for filename in all_filenames:

images.append(img_to_array(load_img(drive_path+images_path+fil
ename, target_size=(299, 299))))
    images = np.array(images, dtype=float)
    images = preprocess_input(images)

    # Get HTML-code files
    X = []
    all_filenames = listdir(drive_path+html_path)
    for filename in all_filenames:
        X.append(downloadFile(drive_path+html_path+filename))

    # Use InceptionResNetV2 to get features from images
    IR2        =        InceptionResNetV2(weights='imagenet',
include_top=False)
    features = IR2.predict(images)
    tokenizer = Tokenizer(filters='', split=" ", lower=False)
    tokenizer.fit_on_texts(X)
```

```
    # Add space for empty words
    vocab_size = len(tokenizer.word_index) + 1
    # Match words in text to index
    sequences = tokenizer.texts_to_sequences(X)

    max_length = max(len(s) for s in sequences) # max
HTML-code length

    # Input for the model
    X, y, image_data = list(), list(), list()
    for img_no, seq in enumerate(sequences):
        for i in range(1, len(seq)):
            # Add the entire sequence to the input and only
keep the next word for the output
            in_seq, out_seq = seq[:i], seq[i]
            # Add empty words to fill up to the max length
                        in_seq = pad_sequences([in_seq],
maxlen=max_length)[0]
             out_seq = to_categorical([out_seq], num_classes=
vocab_size)[0]
            # Match image and its features
            image_data.append(features[img_no])
            # Cut the input to max_caption_len tokens
            X.append(in_seq[-max_caption_len:])
            y.append(out_seq)

    X, y, image_data = np.array(X), np.array(y), np.array
(image_data)

    """# Model building"""
    #Create the encoder
    model = Sequential()
```

```
    model.add(Conv2D(32,     (3,     3),     padding='valid',
activation='relu', input_shape=(8, 8, 1536,)))
    model.add(Conv2D(32,       (3,3),       activation='relu',
padding='same', strides=2))
    model.add(Conv2D(32,       (3,3),       activation='relu',
padding='same'))
    model.add(Conv2D(64,       (3,3),       activation='relu',
padding='same', strides=2))
    model.add(Conv2D(128,      (3,3),       activation='relu',
padding='same'))
    model.add(Conv2D(128,      (3,3),       activation='relu',
padding='same', strides=2))
    model.add(Conv2D(128,      (3,3),       activation='relu',
padding='same'))
    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.3))
    model.add(RepeatVector(max_length))

    input = Input(shape=(8, 8, 1536,))
    encoder = model(input)

    text_input = Input(shape=(max_length,))
    text_model       =       Embedding(vocab_size,       50,
input_length=max_length, mask_zero=True)(text_input)
    text_model = LSTM(256, return_sequences=True)(text_model)
    text_model = LSTM(256, return_sequences=True)(text_model)

    #Create the decoder
    decoder = concatenate([encoder, text_model])
```

```
    decoder = LSTM(1024, return_sequences=True)(decoder)
    decoder = LSTM(1024, return_sequences=False)(decoder)
    decoder = Dense(vocab_size, activation='softmax')(decoder)

    # Compile the model
    model = Model(inputs=[input, text_input], outputs=decoder)
    optimizer = RMSprop(lr=0.0001, clipvalue=1.0)
    model.compile(loss='categorical_crossentropy',
optimizer=optimizer)

    """# Model training"""
    callbacks_list                                          =
[ModelCheckpoint(drive_path+"checkpoint.hdf5",      verbose=1,
save_weights_only=True, period=5)]

    # Train the neural network
    model.fit([image_data,     X],     y,     batch_size=64,
shuffle=False, epochs=5, callbacks=callbacks_list)
    model.save(drive_path)
    #model = load_model(drive_path)
    model.load_weights(drive_path+"checkpoint.hdf5")

    """# Testing"""
    # Get the image and convert it to HTML-code
    text      =      convertImgToHTML(model,      tokenizer,
load_img(drive_path               +images_path+'sample_1.png',
target_size=(299, 299)))
    print(text)
    display(HTML(text))
```

ДОДАТОК Б

Скріни візуального інтерфейсу

```
# Get the image, preprocess it, extract features and convert them to HTML-code
test_image = preprocess_input(np.array(img_to_array(load_img(drive_path+images_path+'sample_1.png', target_size=(299, 299))), dtype=float))
test_features = IR2.predict(np.array([test_image]))
text = convertImgToHTML(model, tokenizer, np.array(test_features), max_caption_len)
print(text)
```

```
'<html> \n<head> \n    <style> table, th, td {border: 1px solid black} \n    </style>\n</head> \n<body> \n    <h1> Header </h1> \n    <div> \n
p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt.\n                Proin elementum mi ar
vel pharetra metus, a mollis lacus.\n                Vivamus sed lobortis elit. Donec elementum feugiat placerat. Curabitur pharetra elit. </p>
0%> \n            <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt.\n                Pr
ristique non. Proin vel pharetra metus, a mollis lacus.\n                Vivamus sed lobortis elit. Donec elementum feugiat placerat. Curabitur
</div> \n    <table> \n        <tr> \n            <th> Column </th> \n            ...'
```

```
display(HTML(text))
```

Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt. Proin elementum mi arcu, placerat. Curabitur pharetra elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt. Proin elementum mi arcu, placerat. Curabitur pharetra elit.

Column	Column	Column	Column
Value	Value	Value	Value

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt. Proin elementum mi arcu, placerat. Curabitur pharetra elit.

shutterstock.com · 323592404

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eu ante eget dolor fermentum tincidunt. Proin elementum mi arcu, placerat. Curabitur pharetra elit.

Footer

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	60 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Коваль А.Ф.			Інтелектуальний парсер для перетворення макетів web-сторінок у HTML розмітку	Шифр групи	Код напр./спец.
Перевірив		Золотухін О.В.				СШІм-20-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					