МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

з лабораторної роботи №8

з навчальної дисципліни «Проектування та реалізація програмних систем з нейронними мережами»

Виконав:
студент групи ІП-15
Мєшков Андрій Ігорович

Перевірив:
Шимкович В.М.

Київ 2024

ЛАБОРАТОРНА РОБОТА №8

Тема: Нейронні мережі CNN-bi-LSTM для розпізнавання звуку

Завдання – Написати програму, що реалізує нейронну мережу типу CNN-bi-LSTM для розпізнавання мови в текст. Використати датасет LJ-Speech: https://keithito.com/LJ-Speech-Dataset/

Хід роботи

```
# Завантаження необхідних бібліотек
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from IPython import display
from jiwer import wer

# Завантаження набору даних LJSpeech
data_url = "https://data.keithito.com/data/speech/LJSpeech-1.1.tar.bz2"
data_path = keras.utils.get_file("LJSpeech-1.1", data_url, untar=True)
wavs_path = data_path + "/wavs/"
metadata_path = data_path + "/metadata.csv"


metadata_df = pd.read_csv(metadata_path, sep="|", header=None, quoting=3)
metadata_df.columns = ["file_name", "transcription", "normalized_transcription"]
metadata_df = metadata_df[["file_name", "normalized_transcription"]]
metadata_df = metadata_df.sample(frac=1).reset_index(drop=True)
metadata_df.head(3)
```

	file_name	normalized_transcription
0	LJ035-0056	Meanwhile, Truly had run up several steps towa...
1	LJ046-0025	This planning document has been made a part of...
2	LJ017-0004	I have attempted to draw of crime in connectio...

```
# Розбиття даних на навчальні та тренувальні дані.
split = int(len(metadata_df) * 0.90)
df_train = metadata_df[:split]
df_val = metadata_df[split:]

print(f"Size of the training set: {len(df_train)}")
print(f"Size of the validation set: {len(df_val)}")
```

```
Size of the training set: 11790
Size of the validation set: 1310
```

```
# Попередня обробка
characters = [x for x in "abcdefghijklmnopqrstuvwxyz'?! "]
char_to_num = keras.layers.StringLookup(vocabulary=characters, oov_token="")
num_to_char = keras.layers.StringLookup(
    vocabulary=char_to_num.get_vocabulary(), oov_token="", invert=True
)

print(
    f"The vocabulary is: {char_to_num.get_vocabulary()} "
    f"(size ={char_to_num.vocabulary_size()})"
)
```

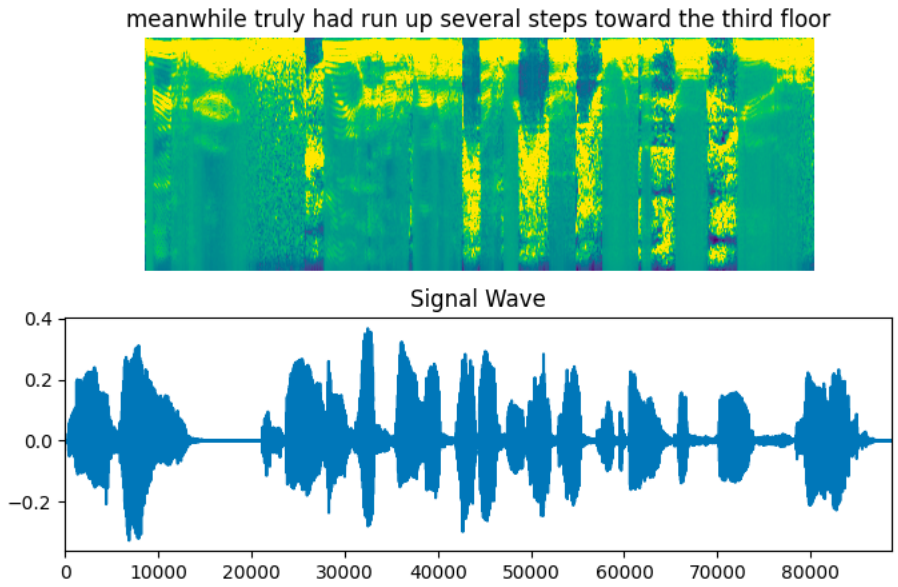
```
The vocabulary is: ['', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', "'", '?', '!', ' ']
(size =31)
```

```python
# Функція, яка описує перетворення, яке ми застосовуємо до кожного елемента нашого
набору даних.
frame_length = 256
frame_step = 160
fft_length = 384


def encode_single_sample(wav_file, label):

    file = tf.io.read_file(wavs_path + wav_file + ".wav")

    audio, _ = tf.audio.decode_wav(file)
    audio = tf.squeeze(audio, axis=-1)
    audio = tf.cast(audio, tf.float32)

    spectrogram = tf.signal.stft(
        audio, frame_length=frame_length, frame_step=frame_step, fft_length=fft_length
    )
    spectrogram = tf.abs(spectrogram)
    spectrogram = tf.math.pow(spectrogram, 0.5)

    means = tf.math.reduce_mean(spectrogram, 1, keepdims=True)
    stddevs = tf.math.reduce_std(spectrogram, 1, keepdims=True)
    spectrogram = (spectrogram - means) / (stddevs + 1e-10)

    label = tf.strings.lower(label)
    label = tf.strings.unicode_split(label, input_encoding="UTF-8")
    label = char_to_num(label)

    return spectrogram, label
```

```python
# Створення об'єктів набору даних
batch_size = 32

train_dataset = tf.data.Dataset.from_tensor_slices(
    (list(df_train["file_name"]), list(df_train["normalized_transcription"]))
)
train_dataset = (
    train_dataset.map(encode_single_sample, num_parallel_calls=tf.data.AUTOTUNE)
    .padded_batch(batch_size)
    .prefetch(buffer_size=tf.data.AUTOTUNE)
)

validation_dataset = tf.data.Dataset.from_tensor_slices(
    (list(df_val["file_name"]), list(df_val["normalized_transcription"]))
)
validation_dataset = (
    validation_dataset.map(encode_single_sample, num_parallel_calls=tf.data.AUTOTUNE)
```

```
    .padded_batch(batch_size)
    .prefetch(buffer_size=tf.data.AUTOTUNE)
)
```

```
# Візуалізуйте дані
fig = plt.figure(figsize=(8, 5))
for batch in train_dataset.take(1):
    spectrogram = batch[0][0].numpy()
    spectrogram = np.array([np.trim_zeros(x) for x in np.transpose(spectrogram)])
    label = batch[1][0]

    label = tf.strings.reduce_join(num_to_char(label)).numpy().decode("utf-8")
    ax = plt.subplot(2, 1, 1)
    ax.imshow(spectrogram, vmax=1)
    ax.set_title(label)
    ax.axis("off")

    file = tf.io.read_file(wavs_path + list(df_train["file_name"])[0] + ".wav")
    audio, _ = tf.audio.decode_wav(file)
    audio = audio.numpy()
    ax = plt.subplot(2, 1, 2)
    plt.plot(audio)
    ax.set_title("Signal Wave")
    ax.set_xlim(0, len(audio))
    display.display(display.Audio(np.transpose(audio), rate=16000))
plt.show()
```




```
# Модель
def CTCLoss(y_true, y_pred):
    batch_len = tf.cast(tf.shape(y_true)[0], dtype="int64")
    input_length = tf.cast(tf.shape(y_pred)[1], dtype="int64")
    label_length = tf.cast(tf.shape(y_true)[1], dtype="int64")
```

```
    input_length = input_length * tf.ones(shape=(batch_len, 1), dtype="int64")
    label_length = label_length * tf.ones(shape=(batch_len, 1), dtype="int64")

    loss = keras.backend.ctc_batch_cost(y_true, y_pred, input_length, label_length)
    return loss
def build_model(input_dim, output_dim, rnn_layers=5, rnn_units=128):
    input_spectrogram = layers.Input((None, input_dim), name="input")
    x = layers.Reshape((-1, input_dim, 1), name="expand_dim")(input_spectrogram)
    x = layers.Conv2D(
        filters=32,
        kernel_size=[11, 41],
        strides=[2, 2],
        padding="same",
        use_bias=False,
        name="conv_1",
    )(x)
    x = layers.BatchNormalization(name="conv_1_bn")(x)
    x = layers.ReLU(name="conv_1_relu")(x)
    x = layers.Conv2D(
        filters=32,
        kernel_size=[11, 21],
        strides=[1, 2],
        padding="same",
        use_bias=False,
        name="conv_2",
    )(x)
    x = layers.BatchNormalization(name="conv_2_bn")(x)
    x = layers.ReLU(name="conv_2_relu")(x)
    x = layers.Reshape((-1, x.shape[-2] * x.shape[-1]))(x)
    for i in range(1, rnn_layers + 1):
        recurrent = layers.LSTM(
            units=rnn_units,
            activation="tanh",
            recurrent_activation="sigmoid",
            use_bias=True,
            return_sequences=True,
            name=f"lstm_{i}",
        )
        x = layers.Bidirectional(
            recurrent, name=f"bidirectional_{i}", merge_mode="concat"
        )(x)
        if i < rnn_layers:
            x = layers.Dropout(rate=0.5)(x)
    x = layers.Dense(units=rnn_units * 2, name="dense_1")(x)
    x = layers.ReLU(name="dense_1_relu")(x)
    x = layers.Dropout(rate=0.5)(x)
    output = layers.Dense(units=output_dim + 1, activation="softmax")(x)
    model = keras.Model(input_spectrogram, output, name="DeepSpeech_2")
    opt = keras.optimizers.Adam(learning_rate=1e-4)
    model.compile(optimizer=opt, loss=CTCLoss)
    return model
```

```
model = build_model(
    input_dim=fft_length // 2 + 1,
    output_dim=char_to_num.vocabulary_size(),
    rnn_units=512,
)
model.summary()
```

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, None, 193)	0
expand_dim (Reshape)	(None, None, 193, 1)	0
conv_1 (Conv2D)	(None, None, 97, 32)	14,432
conv_1_bn (BatchNormalization)	(None, None, 97, 32)	128
conv_1_relu (ReLU)	(None, None, 97, 32)	0
conv_2 (Conv2D)	(None, None, 49, 32)	236,544
conv_2_bn (BatchNormalization)	(None, None, 49, 32)	128
conv_2_relu (ReLU)	(None, None, 49, 32)	0
reshape (Reshape)	(None, None, 1568)	0
bidirectional_1 (Bidirectional)	(None, None, 1024)	8,523,776
dropout (Dropout)	(None, None, 1024)	0
bidirectional_2 (Bidirectional)	(None, None, 1024)	6,295,552
dropout_1 (Dropout)	(None, None, 1024)	0
bidirectional_3 (Bidirectional)	(None, None, 1024)	6,295,552
dropout_2 (Dropout)	(None, None, 1024)	0
bidirectional_4 (Bidirectional)	(None, None, 1024)	6,295,552
dropout_3 (Dropout)	(None, None, 1024)	0
bidirectional_5 (Bidirectional)	(None, None, 1024)	6,295,552
dense_1 (Dense)	(None, None, 1024)	1,049,600
dense_1_relu (ReLU)	(None, None, 1024)	0
dropout_4 (Dropout)	(None, None, 1024)	0
dense (Dense)	(None, None, 32)	32,800

```
def decode_batch_predictions(pred):
    input_len = np.ones(pred.shape[0]) * pred.shape[1]
    results = keras.backend.ctc_decode(pred, input_length=input_len, greedy=True)[0][0]
    output_text = []
    for result in results:
        result = tf.strings.reduce_join(num_to_char(result)).numpy().decode("utf-8")
        output_text.append(result)
    return output_text
```

```
epochs = 10
history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
)
```

```
--------------------------------------------------------------------------------------------------------
Word Error Rate: 0.4546
--------------------------------------------------------------------------------------------------------
Target : there were no eyewitnesses although a fourteenyearold boy in a neighboring house claimed that immediately after the shooting
Prediction: there were no i witneces although forteen yearald boy in a naboring house clamed that im mediaely after the shoting
--------------------------------------------------------------------------------------------------------
Target : and which contributed in no small degree to the introduction of private executions a great crowd was expected and a great crowd came
Prediction: and which contributed in no smal degre to the introduction of privit executions agreat crowd was expected and agreat crow kaim
--------------------------------------------------------------------------------------------------------
369/369 [==============================] - 947s 3s/step - loss: 60.0124 - val_loss: 61.2968
```

Висновок:

Під час виконання лабораторної роботи було реалізовано нейронну мережу типу CNN-bi-LSTM для розпізнавання мови в текст.