Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №8
«Нейронні мережі CNN-bi-LSTM для розпізнавання звуку»

з дисципліни

«Програмні засоби проектування та реалізації нейромережевих систем»

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку: 63

Перевірив:
Шимкович В. М.

Київ 2024

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:

За основу був узятий приклад, наданий в методичці: https://keras.io/examples/audio/ctc_asr/

Спершу було визначено низку констант, серед яких: кількість епох навчання, розмір тренувального пакету:

```
epochs = 5
batch_size = 32
```

Далі відбувається завантаження датасету за наданим у завданні посиланням:

```
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"
```

Потім відбувається обробка метаданих файлів та поділ їх на тренувальні (90%) та тестувальні (інші 10%):

```
# Read metadata file and parse it
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)

split = int(len(metadata_df) * 0.9)
df_train = metadata_df[:split]
df_val = metadata_df[split:]
```

Наступним кроком визначається словник літер та описано перетворення символи в числа та навпаки:

```
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 )
```

Додатково визначаються параметри для спектрограми аудіо за допомогою перетворень Фур'є:

```
frame_length = 256
frame_step = 160
fft_length = 384
```

Функція, яка відповідає за попередньої обробки деякого прикладу аудіо з набору даних:

```
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
```

Предобробка аудіофайлів та транскрипції і пакування їх у пакети. Таким чином створюються тренувальний та тестовий датасет:

```
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))
```

Відображення спектограми прикладу деякого аудіо та відповідної йому хвильової форми відбувається наступним чином:

```
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()
```

Наступна функція відповідає за обчислення навчальних втрат:

```
def CTCLoss(actual, prediction):
    batch_len = tf.cast(tf.shape(actual)[0], "int64")
    input_len = tf.cast(tf.shape(prediction)[1], "int64")
    label_len = tf.cast(tf.shape(actual)[1], "int64")
    input_len = input_len * tf.ones(shape=(batch_len, 1),
dtype="int64")
    label_len = label_len * tf.ones(shape=(batch_len, 1),
dtype="int64")
    loss = keras.backend.ctc_batch_cost(actual, prediction, input_len,
label_len)
    return loss
```

Сама нейронна мережа реалізована таким чином:

```
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.GRU( units=rnn_units, activation="tanh",
recurrent_activation="sigmoid", use_bias=True, return_sequences=True,
reset_after=True, name=f"gru_{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",
name="dense_output")(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)
```

Функція нижче відповідає за декодування розпізнаного моделлю тексту.

```
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
```

Навчання мережі відбувається за допомогою .fit впродовж визначених на початку 5 епох:

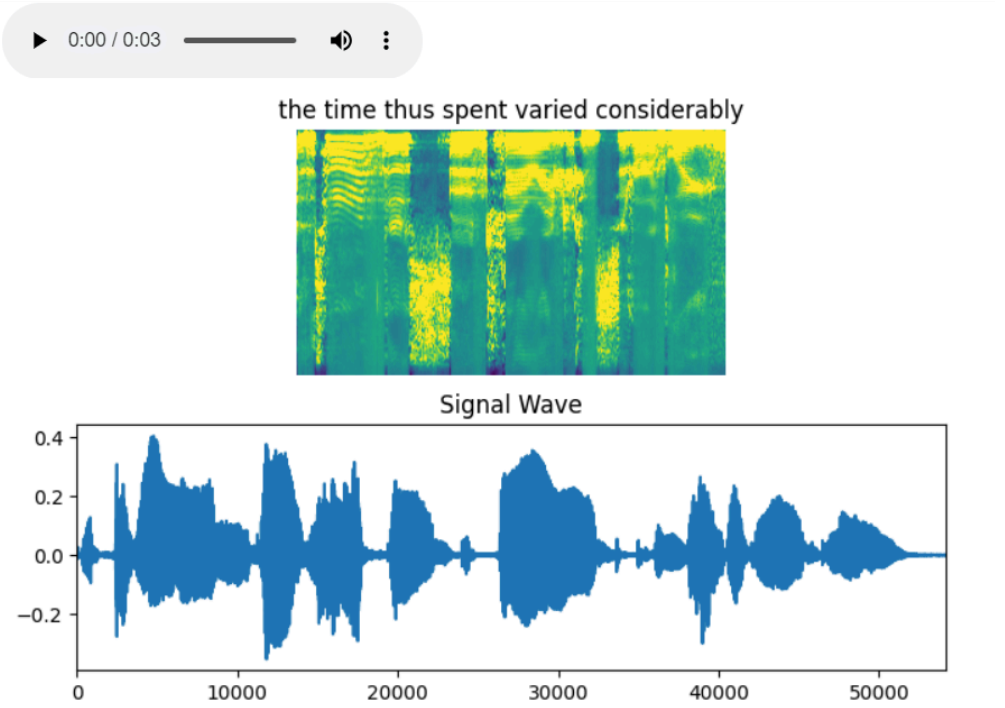
```
history = model.fit(train_dataset, validation_data=validation_dataset,
epochs=epochs)
```

Перевірка якості навченої моделі на валідаційних даних:

```
predictions = []
targets = []
for batch in validation_dataset:
    X, y = batch
    batch_predictions = model.predict(X)
    batch_predictions = decode_batch_predictions(batch_predictions)
    predictions.extend(batch_predictions)
    for label in y:
        label =
tf.strings.reduce_join(num_to_char(label)).numpy().decode("utf-8")
        targets.append(label)
wer_score = wer(targets, predictions)
print("-" * 100)
print(f"Word Error Rate: {wer_score:.4f}")
print("-" * 100)
for i in np.random.randint(0, len(predictions), 5):
    print(f"Target : {targets[i]}")
    print(f"Prediction: {predictions[i]}")
    print("-" * 100)
```

Результати навчання:

Аудіо, текстовий варіант, спектограма та хвильова форма виглядають наступним чином:



Через 5 епох обрахований рівень помилки становив 62%, що є все ще досить високим показником. Втім, враховуючи невелику тривалість навчання моделі, в перспективі є можливість зменшити цей показник до зазначених у референсі 15-17%.

```
----------------------------------
Word Error Rate: 0.6232
----------------------------------
```

Вивід проміжних результатів по кожній з епох підтверджує позитивні зміни у роботі спроектованої моделі:

```
Epoch 1/5
369/369 [==============================] - 2162s 6s/step - loss: 301.9385 - val_loss: 285.9377
Epoch 2/5
369/369 [==============================] - 934s 3s/step - loss: 201.9794 - val_loss: 149.0404
Epoch 3/5
369/369 [==============================] - 934s 3s/step - loss: 139.2334 - val_loss: 107.3772
Epoch 4/5
369/369 [==============================] - 929s 3s/step - loss: 110.5409 - val_loss: 89.0674
Epoch 5/5
369/369 [==============================] - 926s 3s/step - loss: 93.8212 - val_loss: 78.9211
```

Вивід отриманих результатів передбачення мережі на валідаційних даних наступний:

```
--------------------------------------------------------------------------------------------------------
Target : on april twentyone nineteen sixtythree the fbi field office in new york
Prediction: on ait was wenty one nineteen sixtythre the efbi feuld office in new ork
--------------------------------------------------------------------------------------------------------
Target : already a strong dislike to the reckless and almost indiscriminate application of the extreme penalty was apparent in all classes
Prediction: al redy a strongdisligh to the recle sond olmust indiscrimminit aptication of the etrim penlty was aparint on al clae
--------------------------------------------------------------------------------------------------------
Target : and after he had walled the city and adorned its gates he built another palace before his father's palace but so that they joined to it
Prediction: and affer he had wal the sity and edorn its gates he bilt unother pelas befor his fothers palas but so that tha joing toit
--------------------------------------------------------------------------------------------------------
Target : and thirteen states which contain only five percent of the voting population can block ratification
Prediction: an thirtein stas which containe oly fif prsent of the voting oulation cand  ok rettoffication
--------------------------------------------------------------------------------------------------------
Target : his attempt to express himself through his fair play for cuba activities however
Prediction: is attemp toxpres im sel throh his far ply for cubt ac tivitis hoever
--------------------------------------------------------------------------------------------------------
```

Висновок:

Під час виконання даної лабораторної роботи було спроектовано згорково-рекурентну нейронну мережу CNN-bi-LSTM. Ця архітектура поєднує у собі згорткові шари для розпізнання ознак з аудіосигналу та багатошарові рекурентні шари LSTM для аналізу послідовностей. Бінаправленість LSTM дозволяє моделі аналізувати контекст у двох напрямках, що може покращити точність розпізнавання.

У результаті навчання впродовж 5 епох було досягнуто показник WER у 0.6232, що є досить непоганим результатом, як на таку малу тривалість навчання. Вивід проміжних результатів по кожній з епох підтверджує позитивні зміни у роботі спроектованої моделі.