MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Volodymyr Shymkovych

Design andimplementation of software systems with Neural Networks

LABORATORY WORK #8

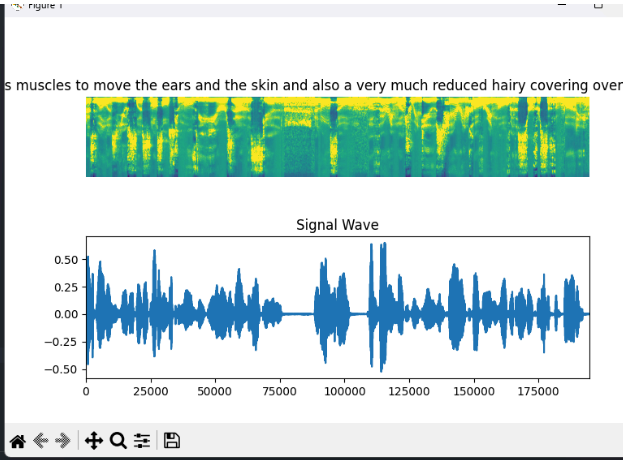
kulubecioglu mehmet
IM-14 FIOT

Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024

```python
# A callback class to output a few transcriptions during training
class CallbackEval(keras.callbacks.Callback):
    """Displays a batch of outputs after every epoch."""

    def __init__(self, dataset):
        super().__init__()
        self.dataset = dataset

    def on_epoch_end(self, epoch: int, logs=None):
        predictions = []
        targets = []
        for batch in self.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), 2):
            print(f"Target    : {targets[i]}")
            print(f"Prediction: {predictions[i]}")
            print("-" * 100)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

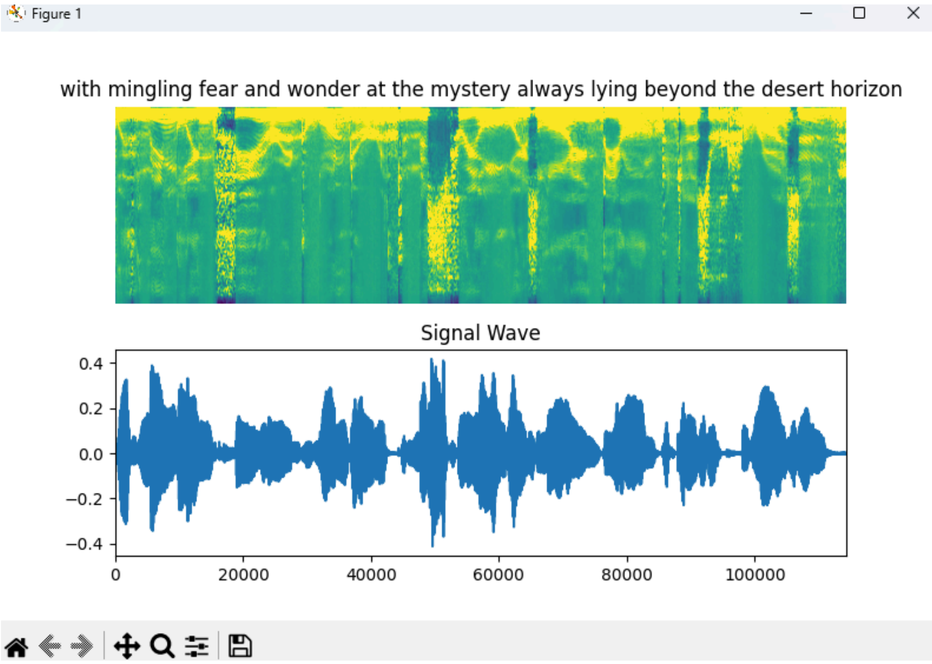
```
2024-05-04 03:15:58.082901: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computat
ion orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-04 03:15:59.705259: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computat
ion orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Size of the training set: 11790
Size of the training set: 1310
2024-05-04 03:16:04.264820: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI AVX512_BF16 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
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)
<IPython.lib.display.Audio object>
2024-05-04 03:16:05.222948: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

Model: "DeepSpeech_2"

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)	6,395,904
dropout (Dropout)	(None, None, 1024)	0
bidirectional_2 (Bidirectional)	(None, None, 1024)	4,724,736
dropout_1 (Dropout)	(None, None, 1024)	0
bidirectional_3 (Bidirectional)	(None, None, 1024)	4,724,736
dropout_2 (Dropout)	(None, None, 1024)	0
bidirectional_4 (Bidirectional)	(None, None, 1024)	4,724,736
dropout_3 (Dropout)	(None, None, 1024)	0
bidirectional_5 (Bidirectional)	(None, None, 1024)	4,724,736
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

Total params: 26,628,480 (101.58 MB)

```
Total params: 26,628,480 (101.58 MB)
Trainable params: 26,628,352 (101.58 MB)
Non-trainable params: 128 (512.00 B)
```



Automated Speech Recognition (ASR) Report

Introduction:
Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that focuses on developing methodologies and technologies enabling computers to recognize and translate spoken language into text. This technology is also known as Automatic Speech Recognition (ASR), Computer Speech Recognition, or Speech to Text (STT). It incorporates knowledge from computer science, linguistics, and computer engineering.

Objective:
This demonstration aims to combine a 2D Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Connectionist Temporal Classification (CTC) loss to build an ASR system similar to DeepSpeech2. The CTC algorithm is particularly useful for training deep neural networks in scenarios where the input does not align directly with the output, such as speech and handwriting recognition.

Dataset:
The LJSpeech dataset, derived from the LibriVox project, is used for this demonstration. It consists of 13,100 short audio clips of a single speaker reading passages from seven non-fiction books. Each audio file is a single-channel 16-bit PCM WAV with a sample rate of 22,050 Hz. The dataset includes a metadata file with normalized transcriptions, which will be used for training and validation.

Data Preparation:

1. Loading Data: The LJSpeech dataset is downloaded, extracted, and loaded into a Pandas DataFrame.
2. Splitting Data: The dataset is split into a training set (90%) and a validation set (10%).
3. Vocabulary Preparation: A set of accepted characters (a-z, punctuation, and space) is defined. Characters are mapped to integers for model input.

Preprocessing:
1. Audio Processing: Audio files are read, decoded, and converted into spectrograms using Short-Time Fourier Transform (STFT). The spectrograms are normalized for consistent input.
2. Label Processing: Transcriptions are converted to lowercase, split into individual characters, and mapped to integers.

Model Architecture:
The ASR model is inspired by the DeepSpeech2 architecture and includes:
1. 2D Convolutional Layers: Extract features from the spectrograms.
2. Recurrent Layers (RNNs): Capture temporal dependencies in the audio data using GRU layers in a bidirectional setup.
3. Dense Layers: Further process the features before the final classification.
4. CTC Loss Function: Custom loss function to handle the alignment between audio and text transcriptions.

Training:
The model is trained for a specified number of epochs using an Adam optimizer. A custom callback function evaluates the Word Error Rate (WER) on the validation set after each epoch, providing insight into the model's performance.

Evaluation:
The WER is used to measure the model's accuracy by comparing the predicted transcriptions with the ground truth. In this demonstration, the initial WER is high, indicating the need for further training. The model achieves a WER of approximately 16-17% after 50 epochs.

Conclusion:
This demonstration illustrates the process of building an ASR system using a combination of CNN, RNN, and CTC loss. With sufficient training, the model achieves a reasonable WER, demonstrating the effectiveness of the DeepSpeech2-inspired architecture for speech recognition tasks. Further training and optimization can improve the model's accuracy, making it suitable for practical ASR applications.

References:
- LJSpeech Dataset
- Speech Recognition Technologies
- Sequence Modeling with CTC
- DeepSpeech2 Model Architecture