ctc-e-branchformer-asr

Problem statement

This project implements a CTC-only Automatic Speech Recognition (ASR) system using the E-Branchformer encoder, trained on the LibriSpeech 100-hour dataset. The model integrates a Language Model (LM) during inference to enhance decoding accuracy and reduce Word Error Rate (WER).

Repository structure

```
ctc-e-branchformer-asr/
|-- requirements.txt
|-- data/
|   |-- (place LibriSpeech files or links here)
|-- scripts/
|   |-- build_lm.sh
|   `-- prepare_manifest.py
|-- src/
|   |-- features.py
|   |-- model.py
|   |-- train.py
|   |-- decode.py
|   `-- evaluate.py
|-- configs/
|   `-- config.yaml
`-- assets/
    `-- (place training curves / sample outputs)
```

Requirements

```
torch>=1.12
torchaudio
numpy
scipy
librosa
pyctcdecode
kenlm
jiwer
tqdm
PyYAML
soundfile
```

Example configuration (configs/config.yaml)

```
dataset:
  sample_rate: 16000
  n_mels: 80
  n_fft: 512
  win_length: 400
  hop_length: 160
```

```
train:
  batch_size: 16
  epochs: 60
  lr: 1e-4
  device: cuda

model:
  input_dim: 80
  d_model: 256
  num_layers: 12
  nhead: 4
  conv_expansion: 2
  dropout: 0.1

decode:
  beam_width: 100
  lm_weight: 2.0
  word_score: -1.0
```

Quick start (minimal instructions)

1. Install dependencies:
   pip install -r requirements.txt

2. Prepare manifests (example):
   python scripts/prepare_manifest.py --librispeech_root /path/to/train-clean-100 --out data/train_manifest.json

3. (Optional) Build LM:
   jq -r '.text' data/train_manifest.json > data/train_transcripts.txt
   ./scripts/build_lm.sh data/train_transcripts.txt models/kenlm.arpa

4. Train:
   python src/train.py --manifest data/train_manifest.json --config configs/config.yaml --out checkpoints

5. Decode / evaluate:
   python src/decode.py --wav sample.wav --ckpt checkpoints/checkpoint_epoch10.pt --lm models/kenlm.arpa
   python src/evaluate.py --manifest data/test_manifest.json --ckpt checkpoints/checkpoint_epoch50.pt --lm models/ke

Notes / tips

- Ensure LibriSpeech files are placed under data/ and manifests point to correct paths.
- If you get OOM errors, reduce batch_size or num_layers.
- Building KenLM (lmplz) requires system build tools; use Docker or follow KenLM instructions if building fails.
- Keep feature extraction parameters consistent across train/val/test.

Deliverables

- Training logs and validation curves (place under assets/)
- Model explanation and configuration (configs/config.yaml, src/model.py)
- WER comparison (with and without LM) — include results file in assets/