Speech_Recognition_2022

Tongji University · Class of 2022 · School of Computer Science and Technology · Software Engineering · Machine Intelligence Direction · Speech Recognition Coursework

Teacher: Ying Shen

Semester of instruction: 2024-2025, autumn semester

Task: DeepSpeech

Build a DeepSpeech2 speech recognition model based on the MindSpore framework.

Explanation of Key Steps and Evaluation Results

Question:
Training one epoch takes 50 hours. How can the program run faster?

Experiment Overview

Experiment Background

DeepSpeech2, proposed by Baidu Research, is an end-to-end speech recognition model based on the CTC (Connectionist Temporal Classification) loss function. Unlike traditional manually designed pipelines, DeepSpeech2 uses a neural network to directly learn features from speech signals and generate corresponding text outputs. This model is capable of handling various complex speech scenarios, including noisy environments, different accents, and multiple languages. The introduction of DeepSpeech2 greatly simplifies the development process of speech recognition systems and significantly improves recognition performance.

MindSpore is an open-source deep learning framework developed by Huawei that aims to provide efficient and flexible development experiences. MindSpore supports various hardware platforms, including CPU, GPU, and Ascend, and is suitable for a range of scenarios, from research to production. With MindSpore, developers can quickly build and train deep learning models and achieve efficient inference and deployment.

Experiment Objectives

- **Familiarize with the MindSpore training process**: Through this experiment, learners will understand how to use MindSpore to build, train, and evaluate models, gaining proficiency in basic usage of MindSpore.
- **Familiarize with building and training DeepSpeech2 models using MindSpore**: Learners will deeply understand the architecture of the DeepSpeech2 model and implement its training and evaluation using MindSpore, mastering how to perform end-to-end speech recognition tasks.
- **Learn Linux commands**: The experimental platform is based on a Linux environment, allowing learners to become proficient in common Linux commands and enhance their development skills in Linux systems.

Experiment Environment

- **Platform**: ECS (Cloud Server)
- **Framework**: MindSpore 1.6
- **Hardware**: CPU
- **Software Environment**: Python 3.9.0, MindSpore 1.6

Experiment Process

Environment Setup

Purchase ECS resources and log in remotely to set up the development environment.

Code and Data Download

The `deepspeech2` project code is located in the `models/official/audio/deepspeech2` folder.

Training and inference parameters are in the `config.py` file.

The dataset is the LibriSpeech dataset.

- **Training Set**: train-clean-100: [6.3G] (100 hours of clean speech)
- **Validation Set**: dev-clean.tar.gz [337M] (clean), dev-other.tar.gz [314M] (noisy)
- **Test Set**: test-clean.tar.gz [346M] (clean), test-other.tar.gz [328M] (noisy)

Data Preprocessing

Preparation

1. Install Python 3.9.0.
2. Install MindSpore and required dependencies.
3. Download the data preprocessing script from `SeanNaren` .

LibriSpeech Data Preprocessing

1. Upload the local dataset to the MobaXterm server and modify the path.
2. Run the preprocessing script with the following command:

```
python librispeech.py
```

3. Convert `json` files to `csv` files: Create a file `json_to_csv.py` under the `deepspeech.pytorch` directory and copy the code there.

Model Training and Evaluation

Model Training

1. Create a `deepspeech_pytorch` directory under the `DeepSpeech2` directory and create a `decoder.py` file there.
2. Modify the `config.py` file under the `src` directory:
   - Set `batch_size` to 1 (the batch size depends on the server's performance).
   - Set `epochs` to 1 (approximately 48 hours, adjustable).
   - Set `train_manifest` to the actual path of `libri_train_manifest.csv` .
   - Set `test_manifest` to the actual path of `libri_test_clean_manifest.csv` .
   - Change the window type in `eval_config` from `hanning` to `hann` .
3. Install Python dependencies and download the pretrained model with the following command:

```
wget https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/
```

4. Modify the training start script in the `scripts` directory (`run_standalone_train_cpu.sh`) to load the pretrained model:

```
PATH_CHECKPOINT=$1
python ./train.py --device_target 'CPU' --pre_trained_model_path $PATH_CHECKPOINT
```

5. Run the model training in the `DeepSpeech2` directory with the following command:

```
bash scripts/run_standalone_train_cpu.sh PATH_CHECKPOINT
# PATH_CHECKPOINT is the pretrained model path
```

Or run it in the background:

```
nohup bash scripts/run_standalone_train_cpu.sh '/home/work/models/official/audio/DeepSpeech
```

6. View the training log in the current directory (`train.log`):

```
tail -f train.log
```

```
epoch: 2 step: 669, loss is 542.540771484375
epoch: 2 step: 670, loss is 493.9076843261719
epoch: 2 step: 671, loss is 647.4815063476562
epoch: 2 step: 672, loss is 637.4447631835938
epoch: 2 step: 673, loss is 555.0318603515625
epoch: 2 step: 674, loss is 701.7265014648438
epoch: 2 step: 675, loss is 707.91015625
epoch: 2 step: 676, loss is 534.0996704101562
epoch: 2 step: 677, loss is 626.442138671875
epoch: 2 step: 678, loss is 531.4719848632812
epoch: 2 step: 679, loss is 529.4127807617188
epoch: 2 step: 680, loss is 705.5828247070312
epoch: 2 step: 681, loss is 713.3256225585938
epoch: 2 step: 682, loss is 603.5298461914062
epoch: 2 step: 683, loss is 457.8786315917969
epoch: 2 step: 684, loss is 638.672119140625
epoch: 2 step: 685, loss is 505.5020446777344
epoch: 2 step: 686, loss is 566.9273681640625
epoch: 2 step: 687, loss is 307.125244140625
epoch: 2 step: 688, loss is 447.6956481933594
epoch: 2 step: 689, loss is 593.1004028320312
epoch: 2 step: 690, loss is 389.2667541503906
epoch: 2 step: 691, loss is 541.1414794921875
epoch: 2 step: 692, loss is 483.0445861816406
epoch: 2 step: 693, loss is 671.4686889648438
epoch: 2 step: 694, loss is 618.6190795898438
epoch: 2 step: 695, loss is 501.9381408691406
epoch: 2 step: 696, loss is 510.1549072265625
epoch: 2 step: 697, loss is 549.3707275390625
epoch: 2 step: 698, loss is 550.191162109375
epoch: 2 step: 699, loss is 604.9495239257812
epoch: 2 step: 700, loss is 218.06565856933594
epoch: 2 step: 701, loss is 620.7894897460938
epoch: 2 step: 702, loss is 605.3624267578125
epoch: 2 step: 703, loss is 594.2975463867188
epoch: 2 step: 704, loss is 212.2847900390625
epoch: 2 step: 705, loss is 575.0111083984375
epoch: 2 step: 706, loss is 697.1428833007812
epoch: 2 step: 707, loss is 450.92645263671875
epoch: 2 step: 708, loss is 437.4091491699219
epoch: 2 step: 709, loss is 91.18871307373047
epoch: 2 step: 710, loss is 719.0734252929688
```

Model Evaluation

1. To evaluate the model:

```
# Evaluate on CPU
bash scripts/run_eval_cpu.sh [PATH_CHECKPOINT]
# [PATH_CHECKPOINT] is the model checkpoint file
# Example:
bash scripts/run_eval_cpu.sh ./checkpoint/ DeepSpeech-1_140.ckpt
```

2. View the evaluation log:

```
tail -f eval.log
```

3. Model export: Modify the `export.py` code as follows:

```
config = train_config
context.set_context(mode=context.GRAPH_MODE, device_target="CPU", save_graphs=False)
with open(config.DataConfig.labels_path) as label_file:
labels = json.load(label_file)
```

```
Ref: he had no occasion to delay for at the next instant a burst of cries filled the outer air and ran along the whole extent of the village
Hyp: a
WER: 0.9642857142857143 CER: 0.9907407407407407

Ref: the woman seemed thoughtful
Hyp: a
WER: 1.0 CER: 0.9583333333333334

Ref: let us hear the suspicions i will look after the proofs
Hyp: a
WER: 1.0 CER: 0.9777777777777777

Ref: no thanks i am glad to give you such easy happiness
Hyp: a
WER: 1.0 CER: 0.975609756097561

Ref: i shudder as i recall these monsters to my remembrance
Hyp: a
WER: 1.0 CER: 0.9777777777777777

Ref: but not more than whats in the bible aunt said dinah
Hyp: a
WER: 1.0 CER: 0.9761904761904762

Ref: the first of these touches conveyed that the written statement took up the tale at a point after it had in a manner begun
Hyp: a
WER: 0.9583333333333334 CER: 0.9897959183673469

Test Summary    Average WER 98.397      Average CER 98.899
```

4. Convert and export the model file:

```
python export.py --pre_trained_model_path  ./ checkpoint/ DeepSpeech-1_856.ckpt
```

Answer to the Question:

Training one epoch takes 50 hours. How can the program run faster?

To speed up the training process, consider the following optimizations:

Hardware Optimizations

1. **Upgrade Hardware**:
   - Use higher-performance hardware such as GPUs (e.g., NVIDIA V100/A100) or AI-specific chips (e.g., Ascend 910). CPUs have limited performance, especially in deep learning tasks, and GPUs or specialized chips can significantly reduce training time.
   - Consider using distributed training across multiple servers to maximize hardware resources.
2. **Cloud Platform Optimizations**:
   - Use cloud platforms that support GPUs or TPUs (e.g., Huawei Cloud, AWS, Google Cloud) to run the task.
   - Choose more powerful computing instances based on your budget.

Software and Algorithm Optimizations

1. **Adjust Batch Size**:
   - Increase the `batch_size` to process more data at once, reducing the number of iterations.
   - Ensure that the batch size is balanced with memory consumption to avoid overflow.
2. **Optimize Data Loading**:
   - Use multithreading or asynchronous data loading (e.g., MindSpore's `DataLoader` provides parallel features).
   - Ensure that data preprocessing (e.g., data augmentation, decoding) is not a bottleneck.
3. **Mixed-Precision Training**:
   - Enable mixed precision (FP16 and FP32) to speed up floating-point computations using hardware support.
   - MindSpore provides the `mindspore.amp` module to support mixed-precision training.
4. **Use Transfer Learning**:
   - Start training from a pretrained model (e.g., DeepSpeech2's open-source pretrained weights) and only update the last few layers, reducing computational load.
5. **Model Pruning and Distillation**:
   - Prune redundant network weights (model pruning).
   - Use knowledge distillation with a smaller student model to learn.
6. **Optimizer and Learning Rate Adjustments**:
   - Use faster-converging optimizers (e.g., Adam, LAMB).
   - Set up a dynamic learning rate scheduler to adjust the learning rate during training.

Reduce Data Size

1. **Simplify the Dataset**:
    - Use a subset of the dataset (e.g., the train-clean-100 dataset) and improve performance through transfer learning.
    - Data augmentation: Apply transformations (e.g., time-stretching, volume adjustments) to increase variability.
2. **Reduce Epochs or Use Early Stopping**:
    - Adjust the training strategy to reduce unnecessary epochs and stop training early based on validation set performance.