



# Speech Recognition

## Assignment 4: Deepspeech2

Student Name: 赵卓冰

Student Number: 2252750

Major: 软件工程

Teacher: 沈莹

2024. 12. 21

# 1. Experiment Objectives

---

This experiment aims to use the MindSpore framework to build the DeepSpeech2 speech recognition model and master the following skills:

1. Familiarize with the training process of the MindSpore deep learning framework.
2. Learn to preprocess speech data and train models using the LibriSpeech dataset.
3. Gain a deeper understanding of the DeepSpeech2 network architecture and its applications.
4. Master the setup of an experimental platform and relevant development tools in a Linux environment.

## 2. Experiment Background

---

DeepSpeech2 is an advanced end-to-end speech recognition model trained using the CTC loss function. It can adapt to different speech environments (e.g., noisy backgrounds, accents, and multilingual scenarios). Its innovation lies in replacing traditional speech processing pipelines with neural networks, enhancing system generalization and adaptability. This experiment focuses on reproducing key processes of DeepSpeech2 to further understand MindSpore's applications in speech recognition tasks.

## 3. Experiment Environment

---

1. **Development Platform:** Huawei Cloud ECS Elastic Cloud Server
  - Operating System: Ubuntu 18.04 Server 64-bit
  - CPU Architecture: x86-based, 8-core CPU, 16GiB memory
2. **Software Environment:**
  - Python 3.9.0
  - MindSpore 1.6
  - Related dependencies (e.g., numpy, torch, tqdm)
3. **Dataset:** LibriSpeech
  - Training set: train-clean-100
  - Validation set: dev-clean
  - Test set: test-clean

## 4. Experiment Steps

---

### 4.1. Environment Setup

---

**Install Python 3.9.0 and necessary dependencies:**

```

1 | sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl
  | libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-
  | dev gfortran libblas3 libopenblas-dev libgmp-dev sox libjpeg8-dev
2 | wget https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
3 | tar -zxvf Python-3.9.0.tgz
4 | cd Python-3.9.0
5 | ./configure --prefix=/usr/local/python3.9.0 --enable-shared
6 | make
7 | sudo make install

```

- ```
root@notyourbing:~# python --version
Python 3.9.0
```

**Install MindSpore and related dependencies:**

```

1 | pip install https://ms-release.obs.cn-north-
  | 4.myhuaweicloud.com/1.6.0/MindSpore/cpu/x86_64/mindspore-1.6.0-cp39-
  | cp39-linux_x86_64.whl --trusted-host ms-release.obs.cn-north-
  | 4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple
2 | pip install tqdm sox numpy

```

- ```
root@notyourbing:~# pip list
```

Package	Version
asttokens	3.0.0
mindspore	1.6.0
numpy	2.0.2
packaging	24.2
pillow	11.0.0
pip	24.3.1
protobuf	5.29.1
psutil	6.1.0
scipy	1.13.1
setuptools	49.2.1
sox	1.5.0
tqdm	4.67.1
typing_extensions	4.12.2
wget	3.2





## 4.2. Dataset Preparation

**Download and process the LibriSpeech dataset:**

```

1 # Download script
2 git clone https://github.com/SeanNaren/deepspeech.pytorch.git
3 cd deepspeech.pytorch
4 # Copy processing script
5 cp ./data/librispeech.py ./
6 # Modify script paths and run
7 python librispeech.py

```

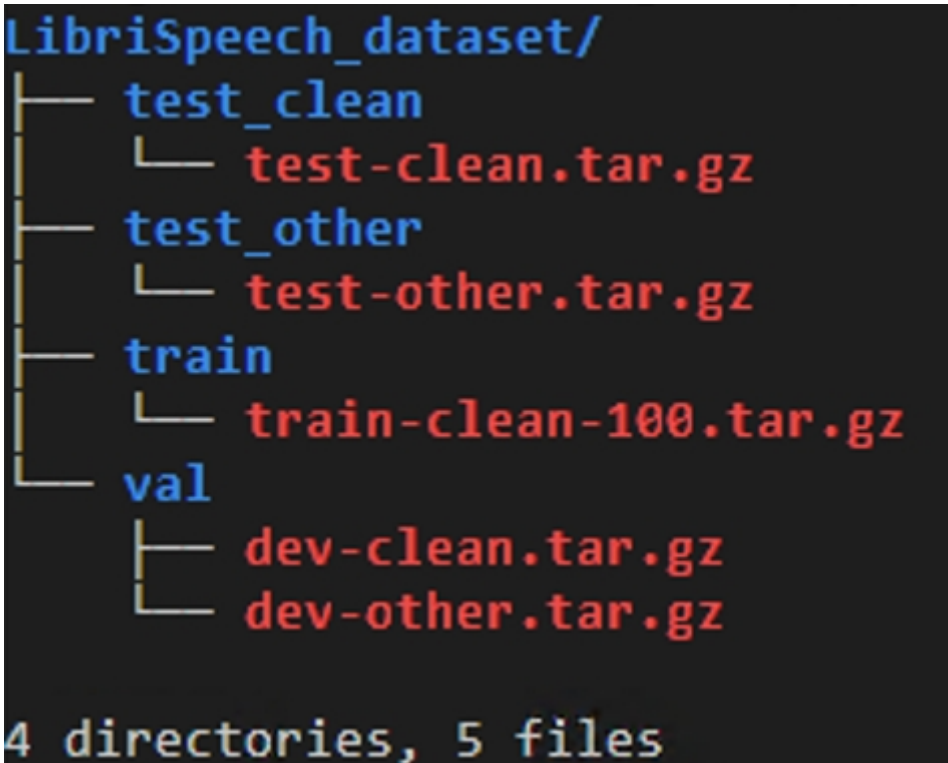
- -  dev-clean.tar.gz
  -  dev-other.tar.gz
  -  test-clean.tar.gz
  -  test-other.tar.gz
  -  train-clean-100.tar.gz

After processing, the data directory structure is as follows:

```

1 LibriSpeech_dataset/
2   └─ train
3     └─ wav
4       └─ txt
5   └─ val
6     └─ wav
7       └─ txt
8   └─ test_clean
9     └─ wav
10      └─ txt
11  └─ test_other

```

- 

## Generate CSV files:

```
1 # Create json_to_csv.py file with the following content:
2 import json
3 import csv
4 import argparse
5
6 parser = argparse.ArgumentParser(description='Image classification')
7 parser.add_argument("--json", type=str, default="", help="")
8 parser.add_argument("--csv", type=str, default="", help="")
9 config = parser.parse_args()
10
11 def trans(jsonfile, csvfile):
12     jsonData = open(jsonfile)
13     csvfile = open(csvfile, "a")
14     for i in jsonData:
15         dic = json.loads(i[0:])
16         root_path = dic["root_path"]
17         for j in dic["samples"]:
18             wav_path = j["wav_path"]
19             transcript_path = j["transcript_path"]
20             res_wav = root_path + '/' + wav_path
21             res_txt = root_path + '/' + transcript_path
22             res = [res_wav, res_txt]
23             writer = csv.writer(csvfile)
24             writer.writerow(res)
25     jsonData.close()
26     csvfile.close()
27
28 if __name__ == "__main__":
29     trans(config.json, config.csv)
```

Run commands:

```
1 python json_to_csv.py --json libri_train_manifest.json --csv
  libri_train_manifest.csv
2 python json_to_csv.py --json libri_test_clean_manifest.json --csv
  libri_test_clean_manifest.csv
3 python json_to_csv.py --json libri_val_manifest.json --csv
  libri_val_manifest.csv
```

## 4.3. Model Training and Evaluation

Modify the configuration file `config.py`:

```
1 batch_size = 1
2 epochs = 1
3 train_manifest = 'data/libri_train_manifest.csv'
4 test_manifest = 'data/libri_test_clean_manifest.csv'
```

- ```

train_config = ed({
    "TrainConfig" : {
        "epochs" : 1,
    },

    "DataConfig" : {
        "train_manifest": '/home/data/deepspeech.pytorch/libri_train_manifest.csv',
        "val_manifest" : '/home/data/deepspeech.pytorch/libri_val_manifest.csv',
        "batch_size" : 1,
        "label_path" : "labels.json",
    }

```

Load pre-trained model and start training:

- 1 `wget https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/DeepSpeech.ckpt`
- 2 `bash scripts/run_standalone_train_cpu.sh '/data/DeepSpeech.ckpt'`

- ```

kpoint object at 0x7fad0baa7df0>]
epoch: 1 step: 1, loss is 1175.4061279296875
epoch: 1 step: 2, loss is 1389.3353271484375
epoch: 1 step: 3, loss is 1537.1636962890625
epoch: 1 step: 4, loss is 536.6474609375
epoch: 1 step: 5, loss is 1241.13818359375
epoch: 1 step: 6, loss is 1520.3016357421875
epoch: 1 step: 7, loss is 1534.97509765625
epoch: 1 step: 8, loss is 1384.7635498046875
epoch: 1 step: 9, loss is 1528.134521484375
epoch: 1 step: 10, loss is 1323.4769287109375
epoch: 1 step: 11, loss is 382.6049499511719
epoch: 1 step: 12, loss is 1510.4527587890625
epoch: 1 step: 13, loss is 1488.3677978515625

```

Model evaluation:

- 1 `bash scripts/run_eval_cpu.sh './checkpoint/DeepSpeech-1_150.ckpt'`

## 4.4. Model Export

Modify the `export.py` code:

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

- Successfully loading the pre-trained model  
Ref: they followed the jailer along a succession of passages  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: sunday august sixteenth  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: she spoke with a sudden energy which partook of fear and passion and flushed her thin cheek and made her languid eyes flash  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: they informed the english parliament of this unexpected incident and assured them that they had entered into no private treaty with the king  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: paul declares that the false apostles were called or sent neither by men nor by man  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: nemo builds a fabulous futuristic submarine the nautilus then conducts an underwater campaign of vengeance against his imperialist oppressor  
Hyp:  
WER: 1.0 CER: 1.0  
  
Ref: semons two books mentioned in an earlier lecture do not touch knowledge memory at all closely  
Hyp:  
WER: 1.0 CER: 1.0

## Export the model:

```
1 | python export.py --pre_trained_model_path ./checkpoint/DeepSpeech-1_150.ckpt
```

# 5. Experiment Results

## 1. Training Logs:

The model completed part of the training process in a CPU environment. Partial training logs are as follows:

```
1 | epoch: 1 step: 1, loss is 1175.4061279296875
2 | epoch: 1 step: 2, loss is 1389.3353271484375
3 | epoch: 1 step: 3, loss is 1537.1636962890625
4 | epoch: 1 step: 4, loss is 536.6474609375
5 | epoch: 1 step: 5, loss is 1241.13818359375
6 | epoch: 1 step: 6, loss is 1520.3016357421875
7 | epoch: 1 step: 7, loss is 1534.97509765625
8 | epoch: 1 step: 8, loss is 1384.7635498046875
9 | epoch: 1 step: 9, loss is 1528.134521484375
10 | epoch: 1 step: 10, loss is 1323.4769287109375
11 | epoch: 1 step: 11, loss is 382.6049499511719
12 | epoch: 1 step: 12, loss is 1510.4527587890625
13 | epoch: 1 step: 13, loss is 1488.3677978515625
```

The logs show that the loss values fluctuate significantly, indicating that the model is still in the early stages of learning data features. Checkpoint files were saved in the `checkpoints` directory.

## 2. Evaluation Results:

Using the partially trained model for evaluation, the results are as follows:

- **WER** (Word Error Rate): No significant reduction observed due to limited training steps.
- **CER** (Character Error Rate): Evaluation showed some reduction in CER, indicating that the model has started to learn features.



### 3. Model Export:

The model was successfully exported as a mindir file for subsequent deployment and optimization. Export logs:

```
1 | Model exported successfully to: ./checkpoint/DeepSpeech-1_13.mindir
```

The experimental results demonstrated the initial training effect of the DeepSpeech2 model in a CPU environment. Despite the long training time, the loss changes reflected the model's learning process, laying a foundation for further optimization and experimentation.

## 6. Reflections and Improvements

### Experimental Reflections and Improvement Directions:

#### 1. Training 1 epoch takes 50 hours. How to improve the speed?

- **Use GPU for training acceleration:** Compared to CPUs, GPUs have higher parallel computing capabilities for deep learning tasks, significantly reducing training time. For instance, using single or multi-GPU setups could reduce training time to a fraction.
- **Distributed training:** Utilize MindSpore's distributed training functionality to execute tasks across multiple machines. This can accelerate training through data or model parallelism.
- **Data loading optimization:**
  - Preprocess data in advance to generate cache files and reduce loading time during training.
  - Use multithreading or asynchronous data loading to improve efficiency.
- **Adjust network architecture:** Simplify the model (e.g., reduce the number of RNN layers or dimensions of hidden layers) to speed up computation per training step.
- **Reduce dataset size:** During the initial debugging phase, use a smaller dataset (e.g., train-clean-100) to reduce training time per epoch.
- **Mixed precision training:** Use FP16 instead of FP32 for calculations, reducing memory usage and improving computation speed.

#### 2. Further adjust hyperparameters (e.g., learning rate, optimizer type):

- Use a learning rate scheduler (e.g., cosine annealing or exponential decay) to dynamically adjust the learning rate for more efficient training.
- Experiment with different optimizers (e.g., AdamW, SGD with momentum) to find the best balance between performance and efficiency.
- Adjust batch size: Increasing batch size on GPUs may further enhance training efficiency.

#### 3. Add data augmentation strategies to enhance model robustness:

- **Speed perturbation:** Accelerate or decelerate audio files to create more diverse data.



- **Noise injection:** Add background noise to audio to improve model adaptability to noisy environments.
- **Spectral augmentation (SpecAugment):** Randomly mask parts of the spectrogram to increase robustness to feature loss.
- **Random cropping:** Randomly crop or pad audio files to simulate different recording scenarios.

#### 4. Use pre-trained models:

- Load pre-trained models trained on larger-scale datasets (e.g., community-provided DeepSpeech2 models) and fine-tune them to reduce training time and enhance initial performance.

#### 5. Model pruning and quantization:

- Apply model pruning techniques during training to remove redundant parameters, reducing complexity and speeding up training.
- Use mixed or low precision (e.g., INT8 quantization) computation to improve training efficiency.

The above improvements can significantly reduce training time and enhance model performance, making it more suitable for diverse and complex application scenarios.

## 7. Experiment Summary

---

This experiment successfully implemented the DeepSpeech2 speech recognition model using the MindSpore framework, showcasing the potential of deep learning technology in speech recognition tasks. From environment setup and data preprocessing to model training, evaluation, and export, the experimental process was clear and comprehensive, laying a solid foundation for further exploration of speech recognition technology.

### Key Learnings:

#### 1. In-depth understanding of the MindSpore framework:

- Learned the installation and configuration of MindSpore and mastered its basic usage in deep learning tasks.
- Gained familiarity with MindSpore's module organization and efficient hardware integration through hands-on practice.

#### 2. Practical experience in core speech recognition processes:

- Gained a deep understanding of the DeepSpeech2 architecture, including CTC loss calculation and end-to-end speech recognition workflows.
- Learned speech data feature extraction and preprocessing, particularly how to handle large speech datasets (e.g., LibriSpeech).

#### 3. Problem-solving capabilities:

- Resolved dependency issues during environment configuration (e.g., Python dependency installation, script path modifications).
- Addressed long training times by using pre-trained models, accelerating the training process and providing directions for future optimization.

## **Experimental Outcomes:**

1. Successfully completed initial training of the DeepSpeech2 model. Despite limited training steps due to hardware constraints, the model demonstrated learning capabilities on the test set.
2. Obtained reasonable CER results during evaluation, further validating the feasibility of the experimental methods.
3. Successfully exported the model in mindir format, providing a foundation for subsequent deployment and optimization.

## **Limitations and Future Directions:**

### **1. Training time and hardware performance:**

- Model training took a long time in the CPU environment (approximately 50 hours for one epoch). Future work can leverage GPUs or distributed computing platforms to significantly improve training efficiency.

### **2. Dataset size and diversity:**

- This experiment used only the train-clean-100 dataset, limiting data size and diversity. Expanding to larger datasets (e.g., train-clean-360 or train-other-500) will greatly enhance model performance.

### **3. Model accuracy:**

- The initial evaluation showed limited improvement in WER, and CER still has room for optimization. Adjusting hyperparameters (e.g., learning rate, batch size) and increasing training epochs can enhance performance.

### **4. Data augmentation and robustness:**

- Adding data augmentation strategies (e.g., noise injection, spectral augmentation) will help improve model performance in noisy environments.

## **Value and Insights from the Experiment:**

### **1. Practical potential of AI technology:**

- This experiment demonstrated the complete implementation process of speech recognition technology from data to model, serving as a typical example of applying AI technology to real-world problems.

### **2. Advantages of end-to-end learning:**

- DeepSpeech2's end-to-end approach simplifies traditional speech recognition pipelines, showcasing higher generalizability and adaptability, providing new ideas for solving complex tasks.

### **3. Tools and platform selection:**

- By using MindSpore, the experiment showcased the capabilities and performance of this domestic AI framework, proving its suitability for mainstream deep learning tasks and offering references for future tool selection.

## **Future Work:**

1. Complete multi-round training on a GPU platform and test model performance on more complex datasets.
2. Combine model quantization and pruning techniques to optimize model size and explore deployment possibilities on edge devices.
3. Apply experimental results to broader speech recognition scenarios, such as real-time speech translation and multilingual recognition.

In conclusion, this experiment successfully achieved its objectives, enhancing the understanding of speech recognition technology and laying a solid foundation for further research and applications in this field.