Arabic Sign Language Recognition - Experimental Guide

Table of Contents

Overview

This guide provides step-by-step instructions for running Leave-One-Subject-Out (LOSO) cross-validation experiments on the Arabic Sign Language dataset using the SignBart model with right-hand-only keypoints.

Experimental Design

- **Dataset**: 12 users × 10 gestures × 10 repetitions = 1,200 samples
- **Method**: Leave-One-Subject-Out (LOSO) cross-validation
- **Test Users**: user01, user08, user11
- **Input**: Right-hand keypoints only (21 landmarks per frame)
- **Task**: 10-class gesture classification

LOSO Configurations

Experiment	Training Set	Test Set	Train Samples	Test Samples
LOSO-1	Users 2-12	User 1	~1,100	~100
LOSO-2	Users 1-7, 9-12	User 8	~1,100	~100
LOSO-3	Users 1-10, 12	User 11	~1,100	~100

Prerequisites

System Requirements

- **OS**: Linux, macOS, or Windows with WSL
- **Python**: 3.8 or higher
- **GPU**: NVIDIA GPU with CUDA support (recommended)
  - Without GPU: Training will be ~10x slower but still functional
- **RAM**: Minimum 8GB, recommended 16GB
- **Storage**: ~5GB free space

Hardware Specifications Used

- **GPU**: NVIDIA RTX 3090 / V100 / A100 (or similar)
- **CPU**: Intel i7/i9 or AMD Ryzen 7/9
- **RAM**: 16GB+

Installation

Step 1: Clone or Navigate to Project Directory

```bash
cd ~/SignBart/SignBart-main
```

Step 2: Install Dependencies

```bash
# Install PyTorch (with CUDA support)
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

# Install other dependencies
pip install opencv-python
pip install mediapipe
pip install numpy
pip install pyyaml
pip install matplotlib
pip install tqdm
pip install scikit-learn  # For confusion matrix (optional)
```

Step 3: Verify Installation

```bash
python -c "import torch; print(f'PyTorch: {torch.__version__}'); print(f'CUDA available: {torch.cuda.is_available()}')"
```

Expected output:

```
PyTorch: 2.x.x
CUDA available: True
```

Dataset Preparation

Step 1: Verify Video Dataset Structure

Ensure your videos are organized as:

```
data/MLR511-ArabicSignLanguage-Dataset-MP4/
├── user01/
│   ├── G01/
│   │   ├── R01.mp4
│   │   ├── R02.mp4
│   │   └── ... (R03-R10)
│   ├── G02/
│   └── ... (G03-G10)
├── user02/
└── ... (user03-user12)
```

Step 2: Update Data Path (if needed)

Edit `prepare_arabic_asl.py` line 9 if your videos are in a different location:

```python
ROOT_RAW = "data/MLR511-ArabicSignLanguage-Dataset-MP4"  # Update this path
```

Step 3: Run Data Preparation

```bash
python prepare_arabic_asl.py
```

Duration: 10-30 minutes (depends on CPU)

Expected Output:

```
================================================================================
Arabic ASL Data Preparation (Right-Hand Only)
================================================================================

Found 1200 videos to process
Output will be saved to: data/arabic-asl

[50/1200] Processed: user05/G01/R05 (45 frames)
[100/1200] Processed: user09/G02/R10 (38 frames)
...
[1200/1200] Processed: user12/G10/R10 (52 frames)

================================================================================
Processing complete: 1200 successful, 0 failed
================================================================================

LOSO directories created:
  data/arabic-asl_LOSO_user01
    - train: 1100 samples
    - test:  100 samples
  data/arabic-asl_LOSO_user08
    - train: 1100 samples
    - test:  100 samples
  data/arabic-asl_LOSO_user11
    - train: 1100 samples
    - test:  100 samples

================================================================================
Data preparation complete!
================================================================================
```

Step 4: Verify Generated Files

bash

```
# Check total samples
find data/arabic-asl/all -name "*.pkl" | wc -l
# Expected: 1200

# Check LOSO splits
find data/arabic-asl_LOSO_user01/train -name "*.pkl" | wc -l
# Expected: ~1100

find data/arabic-asl_LOSO_user01/test -name "*.pkl" | wc -l
# Expected: ~100

# Check label files exist
ls data/arabic-asl_LOSO_user01/label2id.json
ls data/arabic-asl_LOSO_user01/id2label.json
```

Running Experiments

Option 1: Train All LOSO Models (Recommended)

Train all 3 LOSO models sequentially:

```
bash

python train_loso.py --epochs 80 --lr 2e-4
```

Duration: 6-12 hours total (2-4 hours per model on GPU)

What happens:

1. Trains model on users 2-12, tests on user01
2. Trains model on users 1-7,9-12, tests on user08
3. Trains model on users 1-10,12, tests on user11
4. Evaluates each trained model on its test set
5. Generates logs, checkpoints, and plots for each

Option 2: Train Single LOSO Model

Train only one specific LOSO configuration:

```
bash
```

```
# Train/test on user01
python train_loso.py --holdout_only user01 --epochs 80 --lr 2e-4

# Train/test on user08
python train_loso.py --holdout_only user08 --epochs 80 --lr 2e-4

# Train/test on user11
python train_loso.py --holdout_only user11 --epochs 80 --lr 2e-4
```

Duration: 2-4 hours per model on GPU

Option 3: Train with Custom Hyperparameters

bash

```
python train_loso.py \
  --epochs 100 \
  --lr 1e-4 \
  --seed 42 \
  --holdout_only user01
```

Option 4: Quick Test Run (Few Epochs)

Test the pipeline with fewer epochs:

bash

```
python train_loso.py --epochs 5 --lr 2e-4 --holdout_only user01
```

Duration: ~10-20 minutes **Purpose**: Verify everything works before full training

---

Option 5: Resume Training from Checkpoint

If training was interrupted:

bash

```
python main.py \
  --experiment_name arabic_asl_LOSO_user01 \
  --config_path configs/arabic-asl.yaml \
  --task train \
  --data_path data/arabic-asl_LOSO_user01 \
  --epochs 80 \
  --lr 2e-4 \
  --resume_checkpoints checkpoints_arabic_asl_LOSO_user01
```

Option 6: Evaluation Only (Skip Training)

Evaluate pre-trained models:

```
bash

python train_loso.py --skip_training
```

Prerequisite: Checkpoint directories must exist

Option 7: Evaluate Specific Model

```
bash

python main.py \
  --experiment_name arabic_asl_LOSO_user01_eval \
  --config_path configs/arabic-asl.yaml \
  --task eval \
  --data_path data/arabic-asl_LOSO_user01 \
  --resume_checkpoints checkpoints_arabic_asl_LOSO_user01
```

Experiment Options

train_loso.py Command-Line Arguments

Argument	Type	Default	Description
`--config_path`	str	`configs/arabic-asl.yaml`	Path to model configuration file
`--base_data_path`	str	`data/arabic-asl`	Base path to processed data (without _LOSO suffix)
`--epochs`	int	80	Number of training epochs
`--lr`	float	2e-4	Learning rate
`--seed`	int	379	Random seed for reproducibility

Argument	Type	Default	Description
--pretrained_path	str	""	Path to pretrained model (optional)
--skip_training	flag	False	Skip training, only evaluate
--holdout_only	str	""	Train only specific user (user01/user08/user11)

main.py Command-Line Arguments

Argument	Type	Default	Description
--experiment_name	str	**required**	Name for logs and checkpoints
--config_path	str	**required**	Path to YAML config file
--task	str	**required**	"train" or "eval"
--data_path	str	**required**	Path to dataset directory
--epochs	int	200	Number of training epochs
--lr	float	2e-5	Learning rate
--seed	int	379	Random seed
--pretrained_path	str	""	Path to pretrained weights
--resume_checkpoints	str	""	Path to checkpoint directory
--scheduler_factor	float	0.1	LR reduction factor
--scheduler_patience	int	5	Patience for LR scheduler

Monitoring Progress

Real-Time Monitoring

Watch Training Progress (Linux/Mac)

```bash
# Watch log file
tail -f arabic_asl_LOSO_user01.log

# Monitor GPU usage
watch -n 1 nvidia-smi

# Count completed epochs
grep "TRAIN" arabic_asl_LOSO_user01.log | wc -l
```

Check Current Accuracy

```bash
```

```
# Latest training accuracy
grep "TRAIN" arabic_asl_LOSO_user01.log | tail -1

# Latest validation accuracy
grep "VAL" arabic_asl_LOSO_user01.log | tail -1
```

Expected Training Output

```
Training epoch 1/80: 100%|████████████████| 1100/1100 [02:30<00:00, 7.33it/s]
[1/80] TRAIN  loss: 2.2453 acc: 0.1236 top5: 0.4891 | time: 150.3s
Evaluation epoch 1/80: 100%|████████████████| 100/100 [00:15<00:00, 6.67it/s]
[1/80] VAL    loss: 2.1987 acc: 0.1500 top5: 0.5200

Training epoch 2/80: 100%|████████████████| 1100/1100 [02:28<00:00, 7.40it/s]
[2/80] TRAIN  loss: 1.9876 acc: 0.2782 top5: 0.6345 | time: 148.7s
Evaluation epoch 2/80: 100%|████████████████| 100/100 [00:15<00:00, 6.73it/s]
[2/80] VAL    loss: 1.9234 acc: 0.3100 top5: 0.6700

...

Training epoch 80/80: 100%|████████████████| 1100/1100 [02:25<00:00, 7.55it/s]
[80/80] TRAIN  loss: 0.0234 acc: 0.9945 top5: 1.0000 | time: 145.6s
Evaluation epoch 80/80: 100%|████████████████| 100/100 [00:14<00:00, 6.85it/s]
[80/80] VAL    loss: 0.3456 acc: 0.8700 top5: 0.9800

================================================================================
Training Complete!
================================================================================
Total training time: 3.42 hours (12312.5 seconds)
Average time per epoch: 153.9 seconds
Best train accuracy: 0.9945 (99.45%)
Best validation accuracy: 0.8700 (87.00%)
================================================================================
```

Training Progress Indicators

Epoch 1-10: Model learning basic patterns

- Train acc: 10-40%
- Val acc: 10-30%

Epoch 10-30: Rapid improvement

- Train acc: 40-80%
- Val acc: 30-60%

Epoch 30-60: Refinement

- Train acc: 80-95%
- Val acc: 60-80%

Epoch 60-80: Fine-tuning

- Train acc: 95-100%
- Val acc: 70-90%

Collecting Results

Generated Files Structure

After training completes:

```
SignBart-main/
├── arabic_asl_LOSO_user01.log              # Training log
├── arabic_asl_LOSO_user08.log
├── arabic_asl_LOSO_user11.log
├── checkpoints_arabic_asl_LOSO_user01/     # Model checkpoints
│   ├── checkpoints_X_best_train.pth
│   └── checkpoints_Y_best_val.pth
├── checkpoints_arabic_asl_LOSO_user08/
├── checkpoints_arabic_asl_LOSO_user11/
└── out-imgs/
    ├── arabic_asl_LOSO_user01_loss.png     # Training curves
    ├── arabic_asl_LOSO_user01_lr.png       # Learning rate schedule
    ├── arabic_asl_LOSO_user08_loss.png
    ├── arabic_asl_LOSO_user08_lr.png
    ├── arabic_asl_LOSO_user11_loss.png
    └── arabic_asl_LOSO_user11_lr.png
```

Extract Key Metrics

Final Accuracies

```bash
```

```
echo "=== LOSO User01 ==="
grep "VAL" arabic_asl_LOSO_user01.log | tail -1

echo "=== LOSO User08 ==="
grep "VAL" arabic_asl_LOSO_user08.log | tail -1

echo "=== LOSO User11 ==="
grep "VAL" arabic_asl_LOSO_user11.log | tail -1
```

Training Times

```
bash

grep "Total training time" arabic_asl_LOSO_user01.log
grep "Total training time" arabic_asl_LOSO_user08.log
grep "Total training time" arabic_asl_LOSO_user11.log
```

Best Accuracies

```
bash

grep "Best validation accuracy" arabic_asl_LOSO_user01.log
grep "Best validation accuracy" arabic_asl_LOSO_user08.log
grep "Best validation accuracy" arabic_asl_LOSO_user11.log
```

Generate Comprehensive Report

After all experiments complete, run:

```
bash

python collect_results.py
```

This will generate a formatted report with:

1. Network architecture description
2. All hyperparameters
3. Model size and parameters
4. Accuracy per test signer
5. Training and inference times
6. Summary tables

Troubleshooting

Issue 1: CUDA Out of Memory

Error: `RuntimeError: CUDA out of memory`

Solutions:

1. Use CPU (slower but works):

```bash
export CUDA_VISIBLE_DEVICES=""
python train_loso.py --epochs 80 --lr 2e-4
```

2. Reduce model size in `configs/arabic-asl.yaml`:

```yaml
d_model: 64  # was 128
encoder_layers: 1  # was 2
decoder_layers: 1  # was 2
```

Issue 2: Training is Very Slow

On CPU: 5-10 minutes per epoch is normal **On GPU**: Should be 1-3 minutes per epoch

Check:

```bash
# Verify GPU is being used
python -c "import torch; print(torch.cuda.is_available())"

# Monitor GPU usage during training
nvidia-smi -l 1
```

Issue 3: Loss Becomes NaN

Solutions:

1. Lower learning rate:

```bash
python train_loso.py --lr 1e-5
```

2. Add gradient clipping in `utils.py` after `loss.backward()`:

```python
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
```

Issue 4: Low Validation Accuracy

If validation accuracy < 60% after 80 epochs:

1. Train longer:

```bash
python train_loso.py --epochs 120
```

2. Increase learning rate:

```bash
python train_loso.py --lr 5e-4
```

3. Check if right hand is visible in videos

Issue 5: Training Interrupted

Resume from last checkpoint:

```bash
python main.py \
  --experiment_name arabic_asl_LOSO_user01 \
  --config_path configs/arabic-asl.yaml \
  --task train \
  --data_path data/arabic-asl_LOSO_user01 \
  --epochs 80 \
  --lr 2e-4 \
  --resume_checkpoints checkpoints_arabic_asl_LOSO_user01
```

Issue 6: Device Mismatch Error

Error: `Expected all tensors to be on the same device`

Solution: Ensure you've updated `utils.py` with the device handling fix (lines 70 and 74 should move data to device)

Experimental Variations

Experiment A: Different Learning Rates

Compare different learning rates:

```bash
# Low LR
python train_loso.py --lr 1e-4 --holdout_only user01

# Medium LR (default)
python train_loso.py --lr 2e-4 --holdout_only user01

# High LR
python train_loso.py --lr 5e-4 --holdout_only user01
```

Experiment B: Different Model Sizes

Edit `configs/arabic-asl.yaml` to create variants:

Small Model:

```yaml
d_model: 64
encoder_layers: 1
decoder_layers: 1
encoder_ffn_dim: 128
decoder_ffn_dim: 128
```

Large Model:

```yaml
d_model: 256
encoder_layers: 4
decoder_layers: 4
encoder_ffn_dim: 512
decoder_ffn_dim: 512
```

Experiment C: With/Without Augmentation

Disable augmentation in `dataset.py` line 43:

```python
```

```
if self.augment and np.random.uniform(0, 1) < 0.0:  # Disabled
```

Experiment D: Both Hands

Modify `configs/arabic-asl.yaml`:

```yaml
joint_idx: [33, 34, ..., 74]  # Include left hand (33-53) and right hand (54-74)
```

Quick Reference

Essential Commands

```bash
# 1. Prepare data (once)
python prepare_arabic_asl.py

# 2. Train all models
python train_loso.py --epochs 80 --lr 2e-4

# 3. Monitor progress
tail -f arabic_asl_LOSO_user01.log

# 4. Check results
grep "Best validation" *.log

# 5. Generate report
python collect_results.py
```

Directory Structure Check

```bash
# Verify setup
ls configs/arabic-asl.yaml
ls data/arabic-asl/label2id.json
ls data/arabic-asl_LOSO_user01/label2id.json
find data/arabic-asl/all -name "*.pkl" | wc -l  # Should be 1200
```

Expected Timeline

Phase	Duration	Output
Data preparation	10-30 min	Pickle files + LOSO splits
Training (1 model)	2-4 hours	Checkpoints + logs + plots
Training (all 3)	6-12 hours	Complete results
Evaluation	1-2 min/model	Test accuracies
Report generation	< 1 min	Formatted report

Total time: ~7-13 hours from start to finish

Citation

If you use this implementation, please cite:

```bibtex
@misc{signbart_arabic_asl,
  title={SignBart for Arabic Sign Language Recognition},
  author={Your Name},
  year={2025},
  note={Based on SignBart implementation from https://github.com/tinh2044/SignBart}
}
```

Support

For issues or questions:

1. Check the logs: `cat arabic_asl_LOSO_user01.log`
2. Verify data: Run verification commands above
3. Check GPU: `nvidia-smi`
4. Review TROUBLESHOOTING.md

Document Version: 1.0

Last Updated: 2025

Tested On: Ubuntu 20.04, Python 3.10, PyTorch 2.0, CUDA 11.8