Gait Preprocessing - Quick Reference for Your Team

🚀 Getting Started (5 minutes)

1. Install Dependencies

```bash
cd gait_preprocessing
pip install -r requirements.txt
```

2. Preprocess CASIA-B (Basic)

```bash
python casia_b_loader.py \
    --dataset_root /path/to/CASIA-B \
    --output_root preprocessed_data \
    --create_splits
```

3. Test with Small Subset (Recommended First!)

```bash
# Process just 3 subjects, one view, normal walking only
python casia_b_loader.py \
    --dataset_root /path/to/CASIA-B \
    --output_root test_output \
    --subjects 001 002 003 \
    --views 090 \
    --sequences nm \
    --create_splits
```

📦 What You Get After Preprocessing

For each sequence, you'll have:

1. **Silhouettes** (`*_silhouettes.npy`)
    - Shape: `(T, 64, 128)` - T frames of 64x128 silhouettes
    - Use for: CNN-based sequence models
2. **GEI** (`*_gei.npy` and `*_gei.png`)
    - Shape: `(64, 128)` - Single averaged image
    - Use for: Simple CNN classification
3. **Pose** (`*_pose.npy`)
    - Shape: `(T, 33, 3)` - T frames of 33 joints with [x, y, visibility]
    - Use for: RNN/LSTM models
4. **Metadata** (`*_metadata.json`)
    - Contains: frame count, dimensions, statistics

🤖 For Your Teammate (Model Training)

Load GEI Data (Simplest - Start Here!)

python

```python
from data_loader import GaitDataLoader

# Creates train/val/test loaders automatically
loaders = GaitDataLoader.create_loaders(
    data_root='preprocessed_data',
    data_type='gei',  # Start with this!
    batch_size=32
)

# Use in training
for gei_images, labels, metadata in loaders['train']:
    # gei_images: (32, 1, 64, 128) - batch of GEI images
    # labels: (32,) - subject IDs as integers
    # Train your CNN here!
    pass
```

Load Pose Data (For RNN/LSTM)

python

```
from data_loader import SequenceDataset

dataset = SequenceDataset(
    data_root='preprocessed_data',
    data_type='pose',
    sequence_length=100  # Fixed length for batch processing
)

# pose shape: (batch, 100, 33, 3)
# Feed to LSTM: reshape to (batch, 100, 99) for 33*3=99 features
```

Load Silhouette Sequences (For 3D CNN)

python

```
loaders = GaitDataLoader.create_loaders(
    data_root='preprocessed_data',
    data_type='silhouettes',  # Temporal sequences
    batch_size=16  # Smaller batch due to memory
)

# silhouettes shape: (batch, T, 1, 64, 128)
```

🔍 Visualize Your Data (Important - Check Quality!)

python

```
from visualization import visualize_sample

visualize_sample(
    data_root='preprocessed_data',
    subject_id='001',
    sequence_id='nm-01',
    view_angle='090',
    save_dir='viz'
)
```

This creates:

- GEI visualization
- Silhouette grid
- Pose skeleton
- Joint trajectory plots

📊 Dataset Info

CASIA-B Structure

- **124 subjects** (IDs: 001-124)
- **11 view angles**: 0°, 18°, 36°, 54°, 72°, 90°, 108°, 126°, 144°, 162°, 180°
- **10 sequences per subject:**
  - `nm-01` to `nm-06`: Normal walking (6 sequences)
  - `bg-01` to `bg-02`: With bag (2 sequences)
  - `cl-01` to `cl-02`: In coat (2 sequences)

Recommended Training Setup

1. **Start simple**: Use 90° view, normal walking only
2. **Then expand**: Add more views for view-invariant models
3. **Finally**: Include bag/coat for robustness

🎯 Model Recommendations

1. Start: GEI + Simple CNN

📋

python

```python
# Easiest to implement and debug
# Good baseline: ~70-80% accuracy on CASIA-B

import torch.nn as nn

model = nn.Sequential(
    nn.Conv2d(1, 32, 3, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Conv2d(32, 64, 3, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Flatten(),
    nn.Linear(64 * 16 * 32, 128),
    nn.ReLU(),
    nn.Linear(128, num_subjects)  # num_subjects from dataset
)
```

2. Advanced: Pose + LSTM

python

```python
# Better for temporal patterns
# Can detect limping, gait abnormalities

model = nn.LSTM(
    input_size=99,  # 33 joints * 3 coordinates
    hidden_size=128,
    num_layers=2,
    batch_first=True
)
```

3. Expert: 3D CNN for Silhouettes

python

```
# Best performance but more complex
# Captures spatio-temporal features

from torch.nn import Conv3d
# Your 3D CNN implementation
```

💡 Tips & Tricks

Data Splits Already Done!

The preprocessor creates `data_splits.json` with 70/15/15 train/val/test split by subject (not sequence). This prevents data leakage!

Batch Size Recommendations

- GEI: 32-64 (small memory footprint)
- Silhouettes: 8-16 (larger due to temporal dimension)
- Pose: 16-32 (moderate memory)

Quick Dataset Check

python

```python
from data_loader import get_dataset_info

info = get_dataset_info('preprocessed_data')
print(info)  # Shows: subjects, sequences, splits
```

Common Issues

Q: "No such file or directory"

- Make sure to preprocess first with `casia_b_loader.py`

Q: "Out of memory"

- Reduce batch size
- Use fewer workers: `num_workers=2`

Q: "MediaPipe not working"

bash

```
pip install mediapipe --no-cache-dir
```

📁 GitHub Integration

Add to `.gitignore` (Already Included!)

```
preprocessed_data/
*.npy
*.png
*.mp4
```

What to Commit

✅ All `.py` files ✅ `requirements.txt` ✅ `README.md` ✅ `.gitignore`

What NOT to Commit

❌ Preprocessed data (too large) ❌ CASIA-B dataset (license restrictions) ❌ Model checkpoints (until final)

🔗 Connecting Preprocessing → Training

```
1. PREPROCESSING (You)       2. TRAINING (Teammate)
   ↓                              ↓
   casia_b_loader.py            from data_loader import GaitDataLoader
   ↓                              ↓
   preprocessed_data/           loaders = GaitDataLoader.create_loaders(...)
   ├── 001/                          ↓
   │   ├── *_gei.npy        →    for data, labels, _ in loaders['train']:
   │   ├── *_pose.npy        →        model(data)
   │   └── *_silhouettes.npy →
   ├── 002/
   └── data_splits.json
```

🆘 Need Help?

1. **Check examples.py** - Has working code for all scenarios
2. **Read README.md** - Full documentation
3. **Run visualization** - Always visualize to check quality!

⚡ Pro Tips for Your Teammate

1. **Start with 3-5 subjects** for quick iteration
2. **Use 90° view only** for initial model development
3. **Check GEI quality** - if silhouettes are bad, pose will be worse
4. **Use data_splits.json** - prevents accidentally training on test subjects
5. **Try GEI first** - simplest and fastest to get results

📊 Expected Performance (Ballpark)

- **GEI + CNN**: 70-85% accuracy (depending on # subjects and views)
- **Pose + LSTM**: 75-90% accuracy (better temporal modeling)
- **3D CNN**: 85-95% accuracy (state-of-the-art but complex)

These are rough estimates for CASIA-B with proper train/test splits!

Good luck! 🚀