GNN + Transformer Model: Quick Start Guide

This quick start guide will help you get up and running with the GNN + Transformer model for keypoint-based score generation.

Environment Setup

First, set up your Python environment:

bash

```bash
# Create and activate a conda environment
conda create -n gnn_transformer python=3.8
conda activate gnn_transformer

# Install PyTorch with CUDA support (adjust for your CUDA version)
conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch

# Install PyTorch Geometric
conda install pyg -c pyg

# Install other dependencies
pip install numpy pandas matplotlib tqdm scikit-learn opencv-python
```

Data Preparation

Ensure your data is organized in the following directories:

- `D:/pickle_dir`: Body keypoints from OpenPose
- `D:/pickle_files`: Alternative OpenPose format
- `D:/pickle_files_hand`: Hand keypoints from MediaPipe
- `D:/pickle_files_object`: Object locations from TridentNet
- `D:/pickle_dir/therapist_labels.csv`: Therapist labels

If your data is in different locations, update the paths in `run_pipeline.py`.

Running the Full Pipeline

The simplest way to get started is to run the full pipeline:

```bash
python run_pipeline.py process_and_train
```

This will:

1. Process the data from all pickle directories
2. Create a combined segment database
3. Train the GNN + Transformer model
4. Generate visualizations and evaluation metrics

Configuration

To change the configuration, edit the `config` dictionary in `run_pipeline.py`:

```python
config = {
    # Data paths
    'pickle_dirs': {
        'body': 'D:/pickle_dir',            # Modify these paths as needed
        'openpose': 'D:/pickle_files',
        'hand': 'D:/pickle_files_hand',
        'object': 'D:/pickle_files_object'
    },
    'output_dir': 'D:/combined_segments',

    # Model parameters
    'model_output_dir': './output/gnn_transformer',
    'view_type': 'top',  # 'top' or 'ipsilateral'
    'epochs': 30,
    'batch_size': 8,
    # ... other parameters
}
```

Step-by-Step Usage

If you prefer to run the steps individually:

1. Process the Data

python

```python
from multi_pickle_processor import MultiPickleProcessor

# Create processor
processor = MultiPickleProcessor(
    pickle_dirs={
        'body': 'D:/pickle_dir',
        'openpose': 'D:/pickle_files',
        'hand': 'D:/pickle_files_hand',
        'object': 'D:/pickle_files_object'
    },
    output_dir='D:/combined_segments',
    num_files_per_dir=10  # Increase this for more data
)

# Process data
processor.process(
    therapist_labels_path='D:/pickle_dir/therapist_labels.csv',
    output_filename='segment_database.pkl'
)
```

2. Train the Model

python

```python
from main_script import train_model

# Train the model
train_model(
    segment_db_path='D:/combined_segments/segment_database.pkl',
    output_dir='./output/gnn_transformer',
    view_type='top',
    epochs=30,
    batch_size=8,
    lr=1e-4,
    balance_classes=True
)
```

3. Evaluate the Model

python

```python
from main_script import test
import torch

# Load the trained model
model_path = './output/gnn_transformer/gnn_transformer_best.pt'
checkpoint = torch.load(model_path)
model.load_state_dict(checkpoint['model_state_dict'])

# Test the model
test(model, test_loader, criterion, device, './output/gnn_transformer/evaluation')
```

Running Cross-Validation

To run cross-validation:

python

```python
from main_script import cross_validate

cross_validate(
    segment_db_path='D:/combined_segments/segment_database.pkl',
    view_type='top',
    output_dir='./output/gnn_transformer_cv',
    num_folds=5,
    epochs=20  # Reduce epochs for faster cross-validation
)
```

Visualization

To generate visualizations for an existing model:

python

```python
from main_script import visualize_keypoints, visualize_attention

# Visualize keypoints
visualize_keypoints(test_loader, output_dir='./output/visualizations')

# Visualize attention weights
visualize_attention(model, test_loader, device, output_dir='./output/visualizations')
```

Expected Outputs

After running the pipeline, you should expect:

1. `D:/combined_segments/segment_database.pkl`: Combined segment database
2. `./output/gnn_transformer/gnn_transformer_best.pt`: Best model weights
3. `./output/gnn_transformer/test_results.json`: Test metrics
4. `./output/gnn_transformer/confusion_matrix.png`: Confusion matrix
5. `./output/gnn_transformer/training_curves.png`: Learning curves
6. `./output/gnn_transformer/attention_visualizations/`: Attention visualizations
7. `./output/gnn_transformer/keypoint_visualizations/`: Keypoint visualizations

Troubleshooting

Common issues and solutions:

- **CUDA out of memory**: Reduce `batch_size` or `seq_length`
- **Model not learning**: Adjust `lr` or increase `epochs`
- **Missing data**: Check the `pickle_dirs` paths
- **Slow processing**: Reduce `num_files_per_dir` for faster prototyping

Next Steps

Once you have a working model, you can:

1. Experiment with different model parameters (GNN layers, transformer heads, etc.)
2. Try different data modalities (include or exclude hand/object data)
3. Analyze attention weights to understand important frames
4. Apply the model to new datasets or activities

For more details, refer to the full documentation in `CODE_DOCUMENTATION.md`.