Chapter 1

Maintenance Manual

1.1 System Overview

This maintenance manual provides detailed information about the ETCAPS implementation and offers guidance for future development. The system encompasses several neural network architectures, including the Equivariant Transformer (ET), ETCaps, Self Routing Capsules (SRCaps), and ResNet20.

1.2 Installation Instructions

1.2.1 System Requirements

- **Operating System**: Windows, Linux, or macOS
- **Hardware**: CUDA-compatible GPU with at least 8GB VRAM (recommended)
- **RAM**: Minimum 16GB
- **Disk Space**: 10GB for code, datasets, and model checkpoints

1.2.2 Software Dependencies

- Python 3.8
- PyTorch 1.8 with CUDA support
- Python packages:
  - torchvision 0.10.0 or later
  - numpy 1.19.0 or later
  - pandas 1.1.0 or later
  - matplotlib 3.3.0 or later
  - seaborn 0.11.0 or later
  - tqdm 4.50.0 or later

1.2.3 Installation Steps

For detailed installation instructions, please refer to the User Manual.

1.3 System Architecture

1.3.1 Directory Structure

```
Equivariant-Transformer-Capsule-Networks/
 main.py                  # Main entry point for training models
 eval_classification.py   # Script for evaluating classification accuracy
 class_acc.py             # Classification accuracy utility functions
 train.sh                 # SLURM job submission script
 src/                     # Core source code
    coordinates.py       # Coordinate manipulation utilities
    datasets.py          # Dataset loading and preprocessing
    grid_sampler.py      # Grid sampling for transformations
    loss.py              # Loss functions
    models.py            # Model architectures
    networks.py          # Network components
    norb.py              # smallNORB dataset handling
    resnet.py            # ResNet implementation
    train.py             # Training functions
    transformers.py      # Transformer implementations
 liederiv/                # Lie derivative evaluation code
    exps_e2e.py          # End-to-end equivariance evaluation
    exps_layerwise.py    # Layer-wise equivariance evaluation
    lee/                 # Lie derivative implementation
```

1.3.2 Model Checkpoints Structure

Model checkpoints are organized by dataset, model, and experimental condition:

```
cifar10/                     # Dataset name
 et/                      # Model name
    cifar10_best_32_1.pth  # Best checkpoint (32 capsules, depth 1)
 resnet20/                # ResNet20 model
    cifar10_best_32_1.pth
 srcaps/                  # SRCaps model
     cifar10_best_32_1.pth
```

1.3.3 Temporary Files

During training and evaluation, the following temporary files may be generated:

- `*.pth.tmp` - Temporary checkpoints during model saving

- `*.log` - Log files for training progress

1.3.4 Dataset Preparation

Datasets are automatically downloaded upon the first execution of the training script. Alternatively, they can be pre-downloaded to the `data/` directory.

1.4 Source Code Documentation

1.4.1 Key Source Files

Table 1.1: Source Code Files and Their Roles

File	Role
`main.py`	Entry point for training models with command-line argument parsing
`eval_classification.py`	Script for evaluating classification accuracy on test sets
`src/models.py`	Model architecture definitions for ET, ETCaps, SRCaps, and ResNet20
`src/transformers.py`	Implementations of equivariant transformers
`src/networks.py`	Equivariant Transformer components
`src/train.py`	Training loop and optimization functions
`src/datasets.py`	Dataset loading and preprocessing utilities
`liederiv/exps_e2e.py`	End-to-end equivariance evaluation using Lie derivatives

1.4.2 Crucial Constants

Table 1.2: Important Constants in the Codebase

Constant	Location
Dataset configurations	`src/datasets.py:DATASET_CONFIGS`
Viewpoint experiment types	`src/datasets.py:VIEWPOINT_EXPS` Coordinate system limits
`src/transformers.py` (in each transformer class)	
Learning rate schedules	`main.py` (in main_worker function)

1.5 Memory and Space Requirements

1.5.1 Disk Space Requirements

- CIFAR-10 Dataset: 340MB
- SVHN Dataset: 61MB
- smallNORB Dataset: 1.67GB
- Model Checkpoints: 3-6MB per model
- Total: 1GB for basic setup, 10GB with multiple trained models and results

1.5.2 Memory Requirements

- Training ET model: 2-3GB GPU memory
- Training ETCaps model: 5-6GB GPU memory
- Training SRCaps model: 4-5GB GPU memory
- Training ResNet20 model: 1-2GB GPU memory
- RAM usage: 8GB during training with batch size 64

1.6 Main Classes and Methods

1.6.1 Model Architectures

- `ETCaps`: Equivariant Transformer Capsule Network (`src/models.py`)
- `ET`: Equivariant Transformer with ResNet backbone (`src/models.py`)
- `SRCaps`: Self Routing Capsule model (`src/models.py`)
- `ResNet20`: Standard ResNet-20 classifier (`src/resnet.py`)

1.6.2 Other Components

- `Transformer`: Base class for equivariant transformers (`src/transformers.py`)
- `TransformerSequence`: Stacks multiple transformer layers (`src/transformers.py`)
- `TransformerLayer`: Implements a transformer block in ETCAPS (`src/networks.py`)

1.6.3 Training and Evaluation Functions

- `train_epoch` (`src/train.py`): Executes one training epoch
- `validate` (`src/train.py`): Computes validation metrics
- `test` (`src/train.py`): Evaluates classification accuracy
- `get_metrics` (`liederiv/lee/lie_derivs.py`): Computes equivariance errors

1.7 Future Improvements

- Implement additional equivariant transformations for more groups
- Add self-supervised learning methods
- Introduce learned coordinate transformations
- Expand dataset support (e.g., 3DIEBench)
- Add automated hyperparameter tuning tools

1.8 Troubleshooting and Tips

1. **Training Instability**:
   - Reduce learning rate
   - Increase weight decay
2. **CUDA Out of Memory**:
   - Reduce batch size
   - Reduce capsule number or depth
3. **Dependency Errors**:

```
ModuleNotFoundError: No module named 'typing_extensions'
ImportError: numpy.core.multiarray failed to import
```

   - Install missing packages using pip
   - Rebuild virtual environment if needed

1.9 Extending the Framework

1.9.1 Adding New Models

1. Create class in `src/models.py` inheriting from `nn.Module`
2. Implement `__init__` and `forward`
3. Register model in the factory
4. Update argument parsing in `main.py`

1.9.2 Adding New Datasets

1. Add loader functions in `src/datasets.py`
2. Update dataset registry in same file
3. Handle argument in `main.py`

1.9.3 Adding Equivariance Metrics

1. Implement new metrics in `liederiv/lee/lie_derivs.py`
2. Add metric logic to `exps_e2e.py`
3. Update CSV output handling