Appendix A

User Manual

A.1 Introduction

This user manual provides detailed instructions for using the Equivariant Transformer Capsule Networks (ETCaps) codebase. The codebase implements several neural network architectures, including the Equivariant Transformer (ET), ETCaps, SRCaps (Self Routing Capsules), and ResNet20.

A.2 System Requirements

A.2.1 Hardware Requirements

- High-performance computer with multiple GPUs (for training)
- Local Machine equipped with Nvidia GPU
- 10GB free disk space for datasets and model checkpoints
- At least 16Gb RAM (on local machine)

A.2.2 Software Requirements

- Python 3.8
- PyTorch 1.8 with CUDA support
- Required Python packages: see `req.txt` file in the GitHub repository

A.3 Installation

A.3.1 Setting up the Environment for Model Training

These commands are intended to be run on an HPC environment.
Clone the repository and create a virtual environment:

```
git clone https://github.com/Coeze/Equivariant-Transformer-Capsule-
    Networks.git
cd Equivariant-Transformer-Capsule-Networks
python -m venv venv3.8
```

Warning

Ensure you use Python 3.8 and Pytorch 1.8 for the virtual environment setup. Other versions such as Python 3.7 and Pytorch 1.7-1.9 may work, but have not been tested.

Activate the virtual environment:

```
source venv3.8/bin/activate
```

Install the required packages:

```
pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio
    ==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html
pip install -r req.txt
```

Warning

Install Pytorch first. Installing it from the requirements.txt may not work since it requires an external link for cuda support.

A.4 Dataset Preparation

The code supports the following datasets:

- CIFAR-10
- SVHN (Street View House Numbers)
- smallNORB

Datasets are automatically downloaded when training begins.

A.5 Training Models

A.5.1 Command Line Arguments

Table A.1: Command Line Arguments for Training

Parameter	Default	Description
`--data_dir`	required	Path where datasets are stored

`--dataset`	required	One of: `cifar10`, `svhn`, `smallnorb`
`--batch_size`	64	Training batch size
`--random_seed`	42	Seed for reproducibility
`--epochs`	200	Number of training epochs
`--lr`	2e-3	Learning rate
`--wd`	1e-6	Weight decay
`--save_dir`	models/	Directory to save model checkpoints
`--exp`	full	Experiment type: full, azimuth, elevation
`--model_name`	etcaps	Model architecture
`--encoder`	resnet20	Encoder network
`--num_caps`	8	Number of capsules
`--caps_size`	4	Capsule size
`--depth`	1	Network depth
`--num_workers`	4	Data loader workers
`--world_size`	None	GPUs for distributed training
`--port`	52472	Communication port
`--exp_dir`	experiments	Experiment directory

A.5.2 Basic Training Examples

Train ETCaps on CIFAR-10:

```
python main.py \
  --data_dir ./data \
  --dataset cifar10 \
  --batch_size 64 \
  --model_name etcaps \
  --epochs 150 \
  --num_caps 32 \
  --depth 1
```

Train ETCaps on SVHN:

```
python main.py \
  --data_dir ./data \
  --dataset svhn \
  --batch_size 64 \
  --model_name etcaps \
  --epochs 50 \
  --num_caps 32 \
  --depth 1
```

To run via Slurm:

```
#!/bin/bash
#SBATCH --job-name=etcaps_train
#SBATCH --output=logs/etcaps_%j.out
#SBATCH --error=logs/etcaps_%j.err
```

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:7
#SBATCH --mem=32G
#SBATCH --time=24:00:00
#SBATCH --partition=gpu

source ./venv/bin/activate

python main.py \
  --data_dir ./data \
  --dataset svhn \
  --batch_size 64 \
  --model_name etcaps \
  --epochs 50 \
  --num_caps 32 \
  --depth 1
```

Submit with:

```
sbatch train.sh
```

A.6 Model Evaluation

A.6.1 Evaluating Classification Accuracy

The evaluation scripts are designed to run on a local machine with a single CUDA-enabled GPU. First clone the repository and activate the virual environment as in previous steps. `eval_classification.py` evaluates model accuracy.

Table A.2: Evaluation Arguments

Parameter	Default	Description
`--data_dir`	required	Dataset directory
`--dataset`	required	Dataset name
`--batch_size`	128	Evaluation batch size
`--model_name`	required	Model name
`--path_to_model`	required	Path to trained model
`--num_caps`	32	Capsule count
`--caps_size`	4	Capsule size
`--depth`	1	Model depth
`--exp`	full	Evaluation mode
`--familiar`	False	Use seen data (for smallNORB)

Example:

```
python eval_classification.py \
  --data_dir ./data \
  --dataset cifar10 \
  --model_name etcaps \
  --path_to_model ./cifar10/et/cifar10_best_32_1.pth \
  --num_caps 32 \
  --depth 1
```

A.6.2 Evaluating Equivariance Error

Use `liederiv/exps_e2e.py` to compute equivariance errors:

```
python -m liederiv.exps_e2e \
  --model_path ./cifar10/et/cifar10_best_32_1.pth \
  --output_dir ./equivariance_error \
  --data_dir ./data \
  --dataset cifar10 \
  --modelname et \
  --num_caps 32 \
  --depth 1 \
  --num_datapoints 100
```

Metrics computed:

- `trans_x_deriv`, `trans_y_deriv`
- `rot_deriv`
- `shear_x_deriv`, `shear_y_deriv`
- `stretch_x_deriv`, `stretch_y_deriv`
- `saturate_err`

Output is saved as a CSV to `output_dir`.