GAIL Pipeline Implementation

Clean, modular implementation of Generative Adversarial Imitation Learning (GAIL) using the HumanCompatibleAI/imitation library.

Under Development

For environments without pre-trained HuggingFace experts, custom expert training will be added in future versions.

Currently tested environments:

- `CartPole-v0` ✅
- `MountainCar-v0` ✅

Features

- **Pre-trained Expert Loading**: Automatically loads pre-trained experts from HuggingFace
- **Vectorized Environments**: Parallel environment execution for faster training
- **Modular Pipeline**: Clear separation of setup → expert loading → demonstration collection → GAIL training
- **CLI Interface**: Easy configuration without code changes
- **Vector Observations Only**: Supports standard Gym environments (CartPole, MountainCar, etc.)

Installation & Setup

Windows (Command Prompt)

First-time setup:

```
# 1. Install Python 3.12 from python.org (if not installed)
python --version
```

```
# Should show: Python 3.12.x

# 2. Navigate to your GAIL folder
cd C:\Users\YourUsername\GAIL

# 3. Create a virtual environment
python -m venv venv

# 4. Activate environment
venv\Scripts\activate

# 5. Set version variable
set SETUPTOOLS_SCM_PRETEND_VERSION_FOR_IMITATION=0.0.0

# 6. Install the package
pip install -e .

# 7. Run your first test
python gail_pipeline.py --env CartPole-v0
```

Subsequent runs:

```
# 1. Navigate to your GAIL folder
cd C:\Users\YourUsername\GAIL

# 2. Activate environment (previously installed)
venv\Scripts\activate

# 3. Run your first test
python gail_pipeline.py --env CartPole-v0
```

AWS/Cloud Environments [!! Read the Note !!]

Note: Untested - based on typical AWS EC2 setup patterns.

First-time setup (Amazon Linux 2 / Ubuntu):

```
# 1. Connect to your EC2 instance
ssh -i your-key.pem ec2-user@your-instance-ip

# 2. Install Python 3.12 (if not available, use Python 3.11)
sudo yum update -y                    # Amazon Linux
```

```
# OR
sudo apt update && sudo apt upgrade   # Ubuntu

# For Amazon Linux 2
sudo amazon-linux-extras install python3.11 -y
# OR for Ubuntu
sudo apt install python3.12 python3.12-venv -y

# 3. Upload your GAIL package to EC2 (from local machine)
# scp -i your-key.pem -r GAIL/ ec2-user@your-instance-ip:~/

# 4. Navigate to GAIL folder
cd ~/GAIL

# 5. Create virtual environment
python3.12 -m venv venv
# OR
python3.11 -m venv venv

# 6. Activate environment
source venv/bin/activate

# 7. Set environment variable
export SETUPTOOLS_SCM_PRETEND_VERSION_FOR_IMITATION=0.0.0

# 8. Install package
pip install -e .

# 9. Run test (use nohup for long training)
nohup python gail_pipeline.py --env CartPole-v0
```

Subsequent runs:

```
ssh -i your-key.pem ec2-user@your-instance-ip
cd ~/GAIL
source venv/bin/activate
export SETUPTOOLS_SCM_PRETEND_VERSION_FOR_IMITATION=0.0.0

# Run in background
nohup python gail_pipeline.py --env CartPole-v0
```

Quick Start

Basic Usage (Default Settings)

```
python gail_pipeline.py --env CartPole-v0
```

What happens:

1. Loads pre-trained CartPole expert from HuggingFace
2. Collects 60 expert demonstration episodes
3. Trains GAIL for 100,000 timesteps
4. Evaluates learner performance on 100 episodes

Expected output:

```
============================================================
GAIL Pipeline - CartPole-v0
============================================================
[1/4] Setting up environment: CartPole-v0
  ✓ Environment ready: seals:seals/CartPole-v0 (8 parallel envs)
[2/4] Loading expert policy from HuggingFace
  ✓ Expert loaded | Mean reward: 500.00 ± 0.00
[3/4] Collecting 60 expert demonstrations
  ✓ Collected 64 episodes, 32064 transitions
[4/4] Setting up GAIL trainer
  ✓ GAIL trainer initialized
  → Minimum training timesteps: 8192

Training GAIL (100000 steps)
============================================================
Mean reward before GAIL training:  100.35 ± 14.77
round: 100%|
```

```
| 6/6 [00:29<00:00,  4.92s/it]

Learner reward after: 29.20 ± 6.62
Improvement: -71.15
============================================================

============================================================
```

```
Pipeline complete!
============================================================
```

'Note that this run results in an under-trained learner model and is just for demonstration.'

CLI Parameters

Environment & Basic Settings

Parameter	Default	Description
`--env`	`CartPole-v0`	Gym environment name (CartPole-v0)
`--seed`	`42`	Random seed for reproducibility
`--n-envs`	`8`	Number of parallel environments

Demonstration Collection

Parameter	Default	Description
`--demo-episodes`	`60`	Number of expert episodes to collect for training dataset

What it means: More episodes = better representation of expert behavior, but diminishing returns after ~100 episodes.

GAIL Training

Parameter	Default	Description
`--gail-steps`	`100000`	Total training timesteps for GAIL
`--demo-batch-size`	`2048`	Batch size for demonstration samples
`--gen-buffer-capacity`	`512`	Generator replay buffer capacity
`--n-disc-updates`	`16`	Discriminator updates per training round

Minimum timesteps: `demo_batch_size × n-disc-updates` (default: 2048 × 16 = 32,768)

Evaluation

Parameter	Default	Description
`--n-eval-episodes`	`100`	Number of episodes for performance evaluation

What it means: Used only for measuring reward before/after training. More episodes = more accurate estimate.

Usage Examples

Example 1: Quick Test Run

Test GAIL with minimal training:

```
python gail_pipeline.py --env CartPole-v0 --gail-steps 50000 --demo-episc
```

```
python gail_pipeline.py --env  MountainCar-v0 --gail-steps 50000 --demo-e
```

Use case: Fast prototyping, debugging, or verifying setup works.

Example 2: Full Training to Match Expert

Train until the learner matches the expert performance:

```
python gail_pipeline.py --env CartPole-v0 --gail-steps 800000 --demo-epis
```

Expected results:

- Expert reward: 500 ± 0.00
- Learner before: 100.35 ± 14.77
- Learner after: 500.00 ± 0.00

```
python gail_pipeline.py --env MountainCar-v0 --gail-steps 2000000 --demo-
```

Expected results:

- Expert reward: -98.94 ± 7.61
- Learner before: -200.00 ± 0.00
- Learner after: -120.70 ± 18.75

Use case: Production-quality imitation learning.

Example 3: Custom GAIL Configuration

Adjust GAIL hyperparameters:

```
python gail_pipeline.py \
    --seed 32
    --n-envs 10
    --env CartPole-v0 \
    --demo-episodes 100
    --gail-steps 500000 \
    --demo-batch-size 1024 \
    --gen-buffer-capacity 2048 \
    --n-disc-updates 8
```

Note: Minimum timesteps becomes 2048 × 8 = 16,384 [Check]

Use case: Hyperparameter tuning for specific environments.

Example 4: More Parallel Environments [Not Tested]

Speed up training with more parallel environments:

```
python gail_pipeline.py --env CartPole-v0 --n-envs 16 --gail-steps 100000
```

Use case: Faster training on multi-core systems.

Understanding the Output

Stage 1: Environment Setup

```
[1/4] Setting up environment: CartPole-v0
  ✓ Environment ready: seals:seals/CartPole-v0 (8 parallel envs)
```

- Creates vectorized environment for parallel execution
- Wraps with `RolloutInfoWrapper` for demonstration collection

Stage 2: Expert Loading

```
[2/4] Loading expert policy from HuggingFace
  ✓ Expert loaded | Mean reward: 500.00 ± 0.00
```

- Loads pre-trained expert from HuggingFace Hub

- Expert reward shows the performance goal for GAIL

Stage 3: Demonstration Collection

```
[3/4] Collecting 60 expert demonstrations
  ✓ Collected 67 episodes, 33500 transitions
```

- Collects expert trajectories (observations, actions, next_obs, done)
- May collect more episodes than requested to reach minimum timesteps
- Transitions = individual (state, action, next_state) tuples

Stage 4: GAIL Setup

```
[4/4] Setting up GAIL trainer
  ✓ GAIL trainer initialized
  → Minimum training timesteps: 32768
```

- Initializes learner policy (untrained PPO)
- Creates a reward network for GAIL
- Shows minimum timesteps required (auto-calculated)

Training & Evaluation

```
Training GAIL (100000 steps)
============================================================
Mean reward before GAIL training:  100.35 ± 14.77

Training for 100000 timesteps...

Learner reward after: 29.20 ± 6.62
Improvement: -71.15
============================================================
```

Interpreting results:

- **Before training**: Random/untrained policy performance (~20-100 for CartPole)
- **After training**: Learned policy performance (should approach expert ~500)
- **Improvement**: Difference showing learning effectiveness
- **Standard deviation (±)**: Variance across evaluation episodes

Success criteria:

- ✅ After training ≈ Expert reward (within ~10%)

- ✅ Improvement > 80% of gap between before and expert
- ❌ After training << Expert reward → Need more training steps or demos

Troubleshooting

Error: Timesteps too low

`AssertionError: No updates (need at least 16384 timesteps, have only total_timesteps=10000)!`

Solution: Increase `--gail-steps` to at least the minimum shown in setup:

```
python gail_pipeline.py --env CartPole-v0 --gail-steps 16384
```

Performance doesn't match the expert

Possible causes:

1. **Not enough training**: Increase `--gail-steps` (try 1-2M)
2. **Not enough demos**: Increase `--demo-episodes` (try 100+)
3. **Evaluation variance**: Increase `--n-eval-episodes` to 100+

Advanced: Loading Custom Experts [Under Development]

For environments without pre-trained HuggingFace experts, use the separate loader:

```
from load_cheetah_expert import train_cheetah_expert

# Train custom expert
expert, env = train_cheetah_expert(
    total_timesteps=1_000_000,
    save_path="my_expert.zip"
)

# Integrate with pipeline
pipeline = GAILPipeline(env_name="HalfCheetah-v3")
```

```
pipeline.venv = env
pipeline.expert = expert
pipeline.collect_demonstrations()
pipeline.setup_gail()
pipeline.train_gail()
```

Key Concepts

Demo Episodes vs Eval Episodes

- **Demo Episodes** (`--demo-episodes`): Training data for GAIL
  - Used to teach the learner what expert behavior looks like
  - More = better training signal
- **Eval Episodes** (`--n-eval-episodes`): Testing/measurement only
  - Used to calculate performance metrics
  - More = more stable/accurate estimates
  - Does NOT affect training

GAIL Hyperparameters

- **demo_batch_size**: How many expert transitions per batch
  - Larger = more stable discriminator training
  - Should be > 1024 for most environments
- **gen_replay_buffer_capacity**: Buffer size for learner experience
  - Larger = more diverse training data
  - Trade-off with memory usage
- **n_disc_updates**: Discriminator training frequency
  - Higher = better reward learning, but slower
  - Typical range: 8-32

Citation

```
@misc{gleave2022imitation,
  author = {Gleave, Adam and Taufeeque, Mohammad and Rocamonde, Juan and
  title = {imitation: Clean Imitation Learning Implementations},
  year = {2022},
  howPublished = {arXiv:2211.11972v1 [cs.LG]},
  archivePrefix = {arXiv},
  eprint = {2211.11972},
  primaryClass = {cs.LG},
  url = {https://arxiv.org/abs/2211.11972},
}
```

License

This implementation uses the imitation library which is licensed under MIT.