Transformer-Based Behaviour Cloning Using PPO Expert Demonstrations

1. Abstract

This project shows a full imitation learning pipeline in which a Transformer-based policy learns to balance the cartpole by copying a high-quality PPO expert. The PPO agent learns to act almost perfectly through reinforcement learning. After that, its paths are collected and used to train a Behaviour Cloning (BC) model in a supervised way. The BC model uses a Transformer encoder to find time-based connections between stacked observations. The last model is tested in the environment and can accurately mimic the expert's behaviour, showing that Transformers work well for imitation-based control tasks.

2. Introduction to the Algorithm

Imitation Learning (IL) emphasises acquiring decision-making policies through the observation of an expert, rather than through direct interaction with the environment. We use Proximal Policy Optimisation (PPO), a reliable and stable reinforcement learning algorithm, to create expert actions for this project. These expert demonstrations are used to teach a Transformer-based Behaviour Cloning model how to turn a stack of observations into expert actions.
The pipeline is made up of:

1. PPO training to figure out the best way to control a Cart-Pole.
2. A collection of expert demonstrations (state–action pairs).
3. Teaching a Transformer model to copy the expert policy (BC).
4. Testing how well the BC model works and making a video of it in action.

This method is like modern robotics pipelines like ACT, RT-1, and RT-2, where big models learn from expert demonstrations instead of expensive RL interactions.

3. Mathematical Interpretation

3.1 PPO Expert Training

Cart-Pole is formulated as an MDP:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$$

PPO optimizes the clipped surrogate loss:

$$L^{CLIP}(\theta) = E[\min(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

This ensures controlled, stable policy updates.

3.2 Behaviour Cloning (Supervised Imitation)

Once PPO has converged, we collect dataset:

$$\mathcal{D} = \{(x_t, a_t)\}_{t=1}^{N}$$

where

$$x_t \text{ } is \text{ } a \text{ } \boldsymbol{stacked\ observation} \text{ } and \text{ } a_t \text{ } is \text{ } the \text{ } expert \text{ } action.$$

BC minimizes cross-entropy:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{N} \log \pi_\theta (a_t | x_t)$$

3.3 Transformer Policy Model

Each stacked observation x_t is passed through:

Embedding

$$h_0 = W_{\text{embed}} x_t$$

Self-Attention

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Action Prediction

$$\pi_\theta(a|x_t) = softmax(W_{\text{out}} \cdot h_L)$$

The model learns to mimic the PPO expert without interacting with the environment.

4. Pseudocode for the Complete Algorithm

Algorithm: Transformer-Based Behaviour Cloning from PPO Expert

Step 1: PPO Expert Training

Initialize PPO policy π_E

for t in 1 … T:

s_t ← environment state

a_t ← sample from π_E(a | s_t)

Execute a_t and store transition

Update π_E using PPO clipped objective

Save π_E

Step 2: Dataset Collection

Initialize dataset D = {}

for each episode:

Reset env → s

Initialize a stack of last K observations

while episode not done:

a = π_E(s)

x = stacked observation

Add (x, a) to D

s ← next state

Save D

Step 3: Train Transformer for Behavior Cloning

Load dataset D & Normalize observations

Initialize Transformer policy π_θ

for epoch = 1 to E:

for each mini-batch (x_b, a_b):

logits = π_θ(x_b)

loss = CrossEntropy(logits, a_b)

Backpropagate loss and update θ

Save π_θ

Step 4: Evaluate and Record

Load model π_θ

for each evaluation episode:

Reset env

while not done:

x ← stacked observation

a_pred = argmax π_θ(a | x)

Step environment and record frame

Save frames as MP4/GIF

5. Output in Terms of Iterations

This section interprets the detailed training logs obtained during execution, particularly the **PPO expert's iteration-by-iteration progress**.

The file shows around **75 iterations**, each corresponding to **2048 environment steps**, producing approximately **150,000 PPO timesteps** overall.

Key Observations from Iteration Logs

Iterations increase steadily from 1 to 75, showing consistent PPO training progression. At each iteration, PPO prints metrics such as:

- approx_kl → how much the new policy diverges from old policy
- clip_fraction → how often the objective is clipped
- entropy_loss → how random the policy is
- value_loss → critic network error
- explained_variance → how well value function predicts returns

Over time:

- **approx_kl decreases** → policy becomes stable
- **entropy_loss decreases** → policy becomes deterministic (typical for CartPole)
- **explained_variance increases** → critic becomes accurate
- **value_loss drops** → critic predictions improve

These results show **convergence** of PPO before collecting demonstrations.

BC Training Output (Iterations as Epochs)

BC training shows:

Epoch	**Loss**
1	0.08368
10	0.02442
20	0.01648
30	0.01506
40	0.01323
50	0.01148
60	0.01163

This shows:

- Sharp improvement in the first 10 epochs
- Loss plateauing after epoch 40
- Model has **successfully learned** the expert policy

Evaluation Output (Iterations as Episodes)

During evaluation:

```
BC eval rewards: [500.0, 500.0]
```