SwarmAI

20190196 TaeHyung Kim

20190321 Seunghwan Song

20190456 SangHyun Lee

Intro

- Complex interactions
- Behaviors for survival
- Model ecosystems

Intro

- Swarm behavior
- Disturbing predator
- Increase chances of survival

Background

- Fish agent (Discrete Action Multi Agent Actor-Critic)

Algorithm 1 Discrete Action Multi-Agent Actor Critic for N agents

for episode $= 1$ to M **do**
  Receive initial state x
  for $t = 1$ to max-episode-length **do**
    for each agent i **do**
      Select action $a_i \sim \mu_{\theta_i}(o_i)$ w.r.t. the current policy and exploration
    end for
    Execute actions $\mathbf{a} = (a_1, \ldots, a_N)$ and observe reward r and new state x'
    Store (x, a, r, x') in replay buffer $\mathcal{D}$
    $x \leftarrow x'$
  end for
  for agent $i = 1$ to N **do**
    Sample a random minibatch of S samples (x^j, a^j, r^j, x'^j) from $\mathcal{D}$
    Set $y^j = r_i^j + \gamma Q^i_{\theta_i'}(x', a_1, \ldots, a_N)|_{a_k = \mu_{\theta_k}(o_k)}$
    Update critic by minimizing the loss

$$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\right)^2$$

    Update actor using the sampled policy gradient

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_{\theta_i}(o_i^j)\nabla_{a_i} Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)|_{a_i = \mu_{\theta_i}(o_i^j)}$$

  end for
  Update target network parameters for each agent i:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

end for

Background

- Fish agent (Discrete Action Multi Agent Actor-Critic)

Algorithm 1 Discrete Action Multi-Agent Actor Critic for N agents

for episode $= 1$ to M **do**
  Receive initial state x
  for $t = 1$ to max-episode-length **do**
    for each agent i **do**
      Select action $a_i \sim \mu_{\theta_i}(o_i)$ w.r.t. the current policy and exploration ⟸ Discrete Action select
    end for
    Execute actions $\mathbf{a} = (a_1, \ldots, a_N)$ and observe reward r and new state x'
    Store (x, a, r, x') in replay buffer $\mathcal{D}$
    $x \leftarrow x'$
  end for
  for agent $i = 1$ to N **do**
    Sample a random minibatch of S samples (x^j, a^j, r^j, x'^j) from $\mathcal{D}$
    Set $y^j = r_i^j + \gamma Q^i_{\theta'_i}(x', a_1, \ldots, a_N)|_{a_k=\mu_{\theta_k}(o_k)}$
    Update critic by minimizing the loss

$$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\right)^2$$

    Update actor using the sampled policy gradient

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_{\theta_i}(o_i^j)\nabla_{a_i} Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\big|_{a_i=\mu_{\theta_i}(o_i^j)}$$

  end for
  Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

end for

Background

- Fish agent (Discrete Action Multi Agent Actor-Critic)

Algorithm 1 Discrete Action Multi-Agent Actor Critic for N agents

```
for episode = 1 to M do
    Receive initial state x
    for t = 1 to max-episode-length do
        for each agent i do
            Select action a_i ~ μ_θi(o_i) w.r.t. the current policy and exploration
        end for
        Execute actions a = (a_1, ..., a_N) and observe reward r and new state x'
        Store (x, a, r, x') in replay buffer D
        x ← x'
    end for
    for agent i = 1 to N do
        Sample a random minibatch of S samples (x^j, a^j, r^j, x'^j) from D
        Set y^j = r_i^j + γ Q^i_θ'i (x', a_1, ..., a_N)|_{a_k = μ_θk(o_k)}
        Update critic by minimizing the loss
```

$$\boxed{\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\right)^2}$$

```
        Update actor using the sampled policy gradient
```

$$\boxed{\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_{\theta_i}(o_i^j)\nabla_{a_i} Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\big|_{a_i=\mu_{\theta_i}(o_i^j)}}$$

```
    end for
    Update target network parameters for each agent i:
```

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

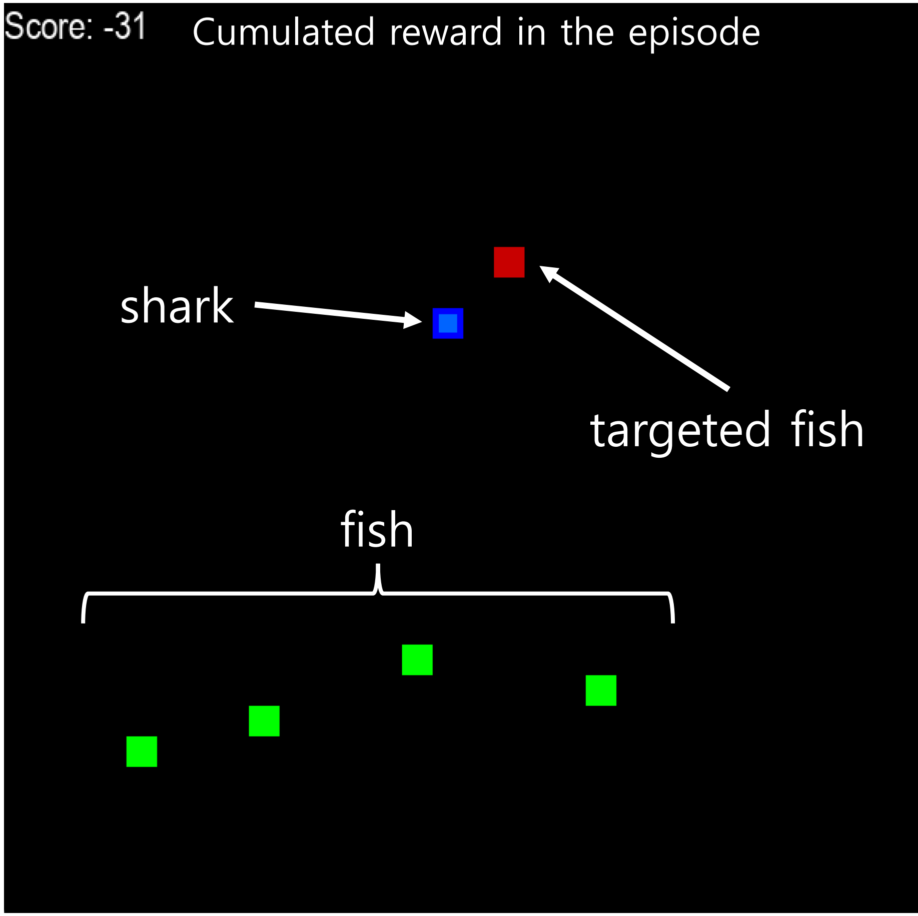
```
end for
```

Discrete Action select

Using Softmax

Method

- Simulation in 2D pixel space
- Periodic boundary
- Partially observable

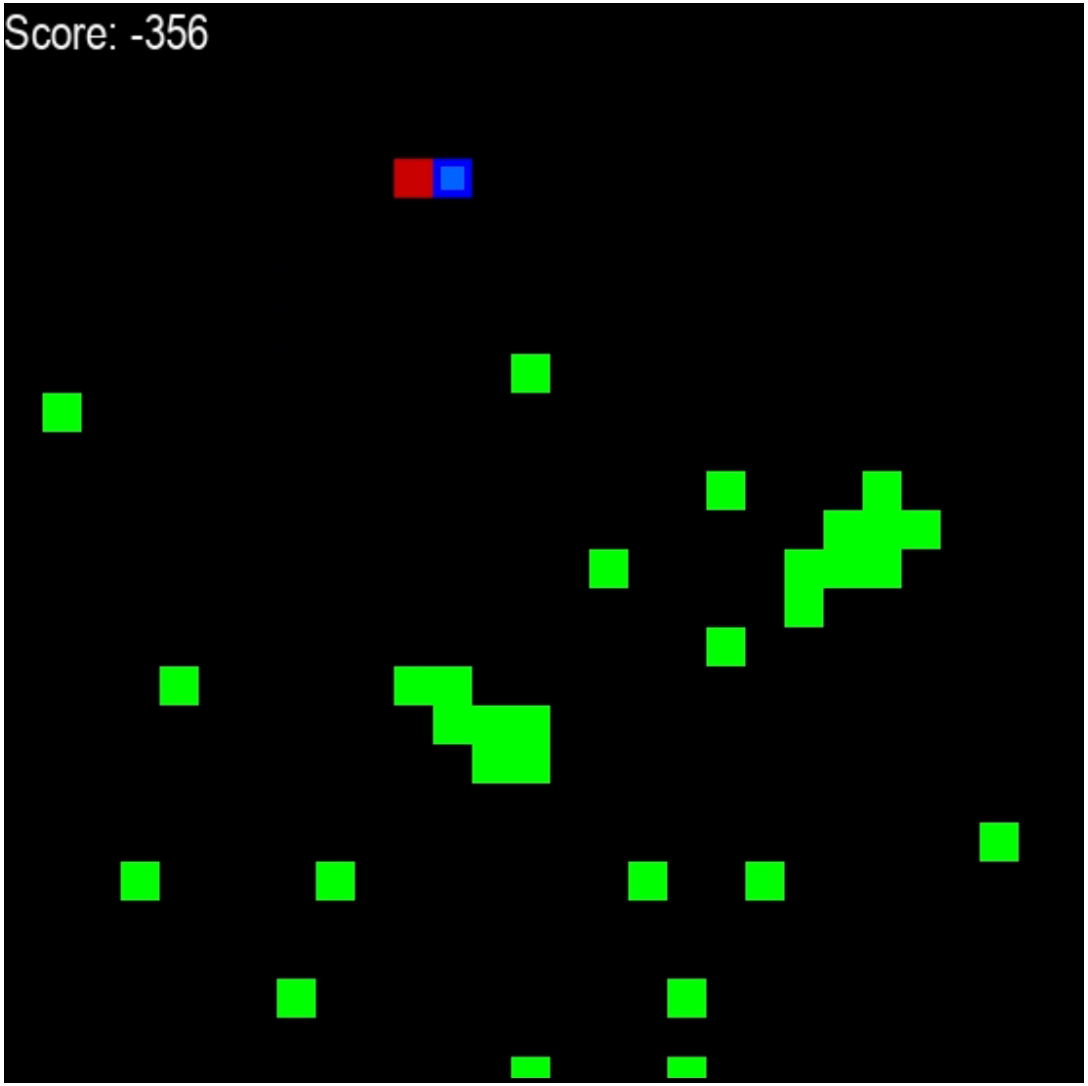


Method

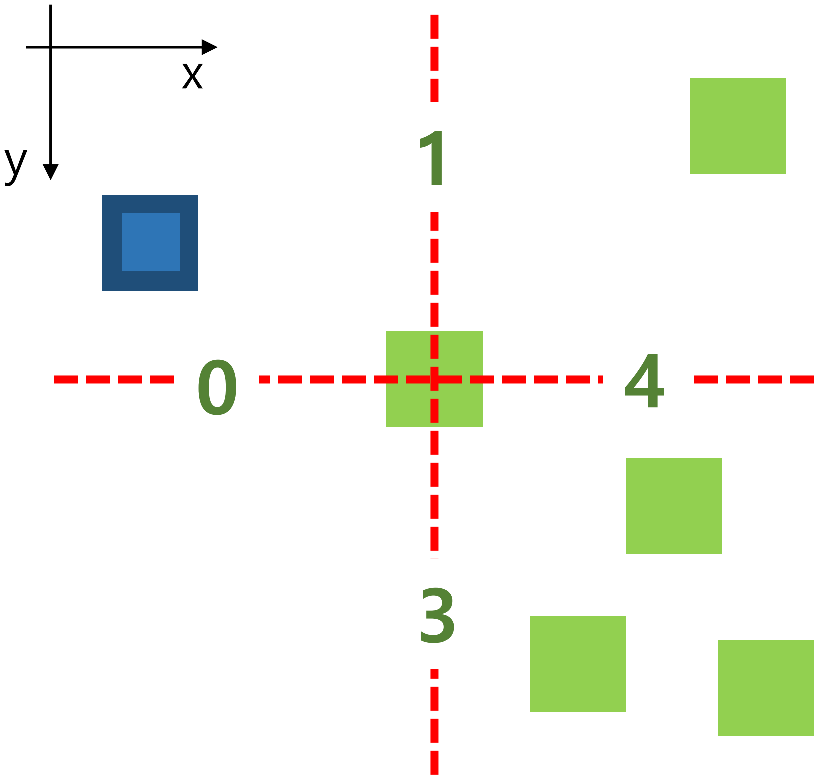
- Game loop (2000 steps per episode)

1. Fish moves w.r.t policy
2. Shark moves
3. Check fish eaten => -1 **reward** per fish eaten
4. Reset shark target to closest fish

Example run

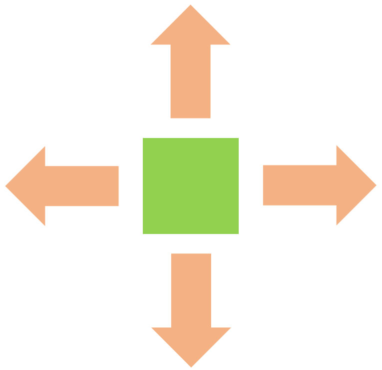


State space



- Number of fish in each direction: (1,3,0,4)
- Shark direction: (1,0,1,0) w.r.t. (up, down, left, right)

Action space



Go one block among four directions: up down left right

Method

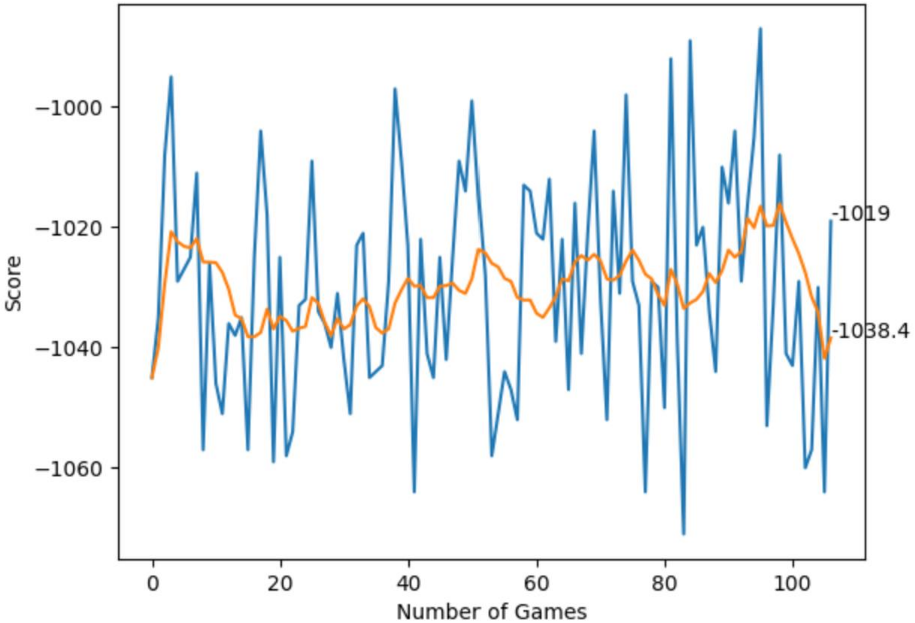
- Shark (challenge)

Algorithm Measuring the swarm size of a fish by the predator

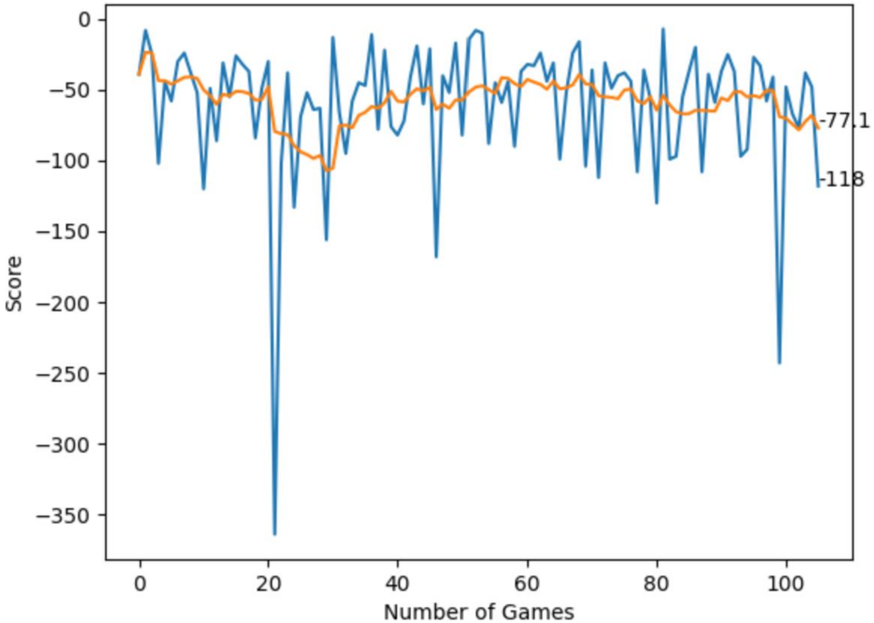
```
Get the target fish and its x,y coordinates x, y
Initialize the swarmsize = 0
for each fish do
    if fish is dead or fish is current target fish then
        continue
    else
        Take care of periodic boundary as follows
        dx = min(abs(x − fish.x), WIDTH − abs(x − fish.x))
        dy = min(abs(y − fish.y), HEIGHT − abs(y − fish.y))
        distance = sqrt(dx ∗ ∗2 + dy ∗ ∗2)
        if distance <= SWARM RADIUS then
            swarmsize += 1
        end if
    end if
end for
return swarmsize + 1
```

Baselines

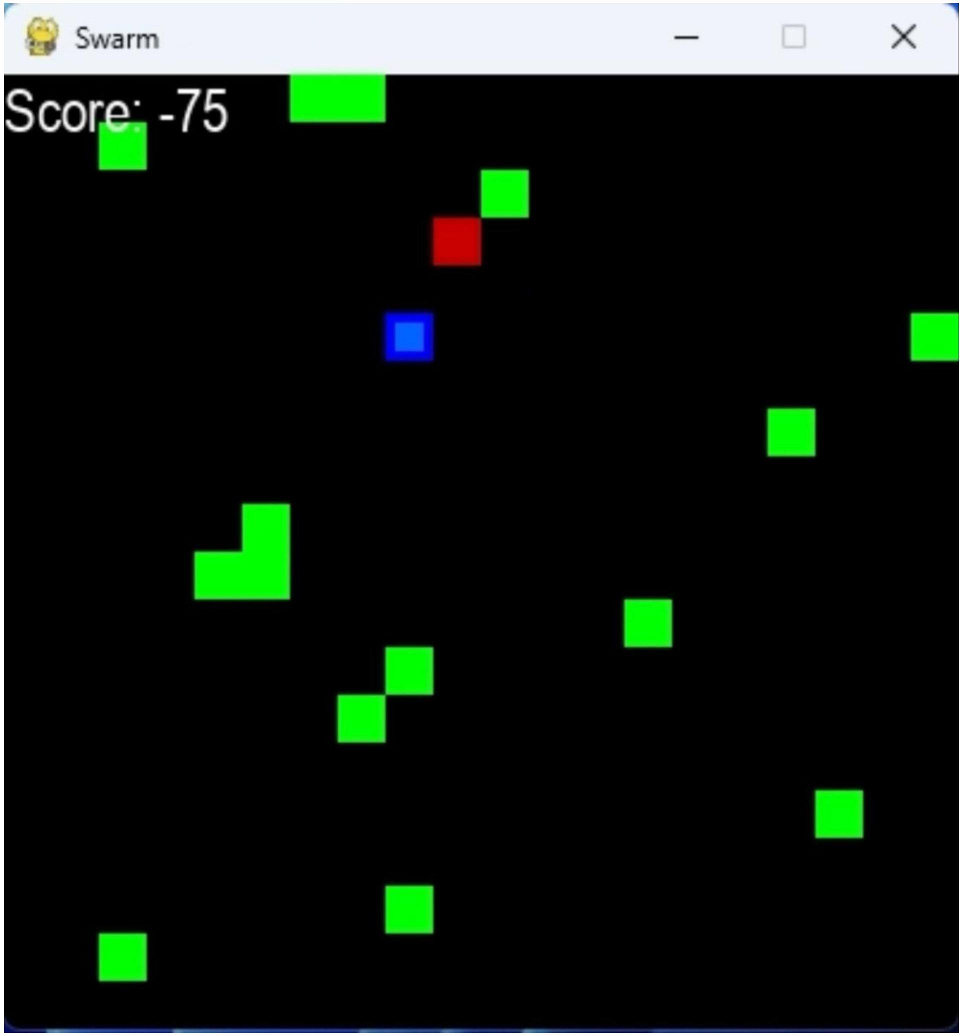
random



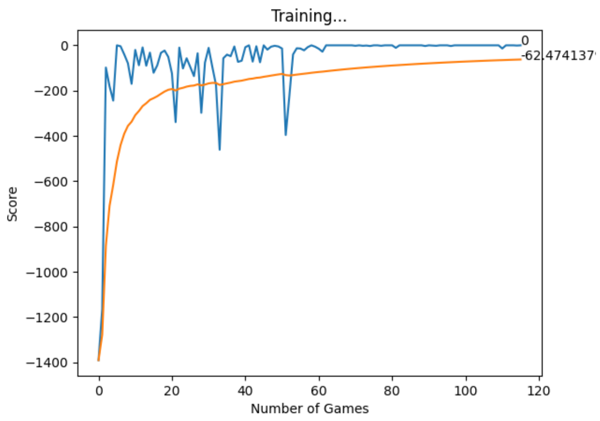
One direction



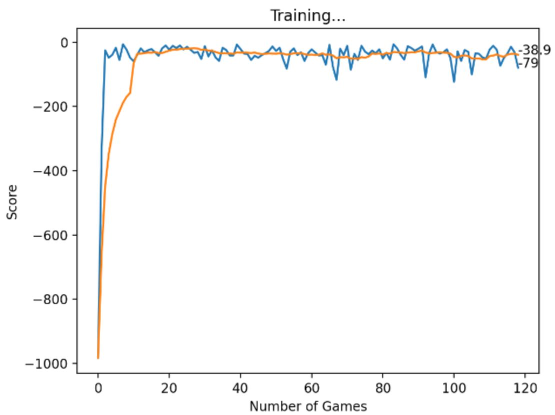
Experiment



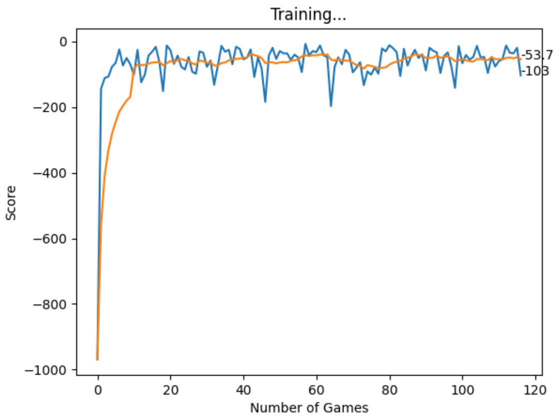
Result



200 x 200



400 x 400



600 x 600

Discussion

- **Environmental Variations and Robustness**
  - ➢ Effective group behavior across different environmental conditions.
- **Behavioral Observations**
  - ➢ Tend to form cohesive groups
  - ➢ Some fish remained alone and moved in a single direction

Conclusion

- Potential of MADDPG in **predator-prey** ecosystem
- Future works will explore more complex and realistic environments.
  - ➢ Refining the model to minimize human intervention.

Thank you