Install Library and Dependencies

```
!pip install stable-baselines3[extra]
!pip install gymnasium

!pip install huggingface_sb3
!pip install huggingface_hub
!pip install panda_gym
```

Challenge 1: PandaReachDense-v3

- Environment Documentations
  - Visualise environment and usage: https://github.com/qgallouedec/panda-gym
  - Brief explanation on action and rewards: https://panda-gym.readthedocs.io/en/latest/usage/environments.html

Step 1: Visualize the environment

```
import gymnasium as gym
import panda_gym

env = gym.make("PandaReachDense-v3")

print(f"Randomly sample a state: {env.observation_space}")

print(f"Randomly sample an action: {env.action_space}")

```

```
Randomly sample a state: Dict('achieved_goal': Box(-10.0, 10.0, (3,), float32), 'desired_goal': Box(-10.0, 10.0, (3,), float32), 'ob
Randomly sample an action: Box(-1.0, 1.0, (4,), float32)
```

Step 2: Create a vectorised environment with normalization

```
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3.common.vec_env import VecNormalize

# Create a vectorised environment
env_id = "PandaReachDense-v3"

env = make_vec_env(env_id = env_id,
                   n_envs = 4)

# Wrap it to normalize input and reward
env = VecNormalize(venv = env,
                   norm_obs = True,
                   norm_reward = True,
                   clip_obs = 10.0)
```

Step 3: Create an A2C model

```
from stable_baselines3 import A2C

model = A2C(policy = "MultiInputPolicy",
            env = env,
            verbose = 1)
```

```
Using cuda device
```

Step 4: Training

```
model.learn(1000000)
```

Show hidden output

Step 5: Save the model

```
model_save_name = f"a2c-{env_id}"

model.save(model_save_name)
env.save("vec_normalize.pkl")
```

Step 6: Evaluation

6.1 Create the evaluation environment

```
from stable_baselines3.common.vec_env import DummyVecEnv, VecVideoRecorder
from stable_baselines3.common.monitor import Monitor

# Create the environment for both evaluation and pushing to hub (including video recording)
eval_env = DummyVecEnv([lambda : Monitor(gym.make(env_id, render_mode = "rgb_array"))])

# Load the normalization statistics obtained from training
eval_env = VecNormalize.load("vec_normalize.pkl", eval_env)

# Use a wrapper to manually record videos (due to errors in `package_to_hub`)
# Once the video is recorded, it can be manually uploaded to the repository and renamed as "replay.mp4"
# This video recording feature only get triggered when used in `package_to_hub`
eval_env = VecVideoRecorder(eval_env,
                            video_folder = "./videos/",
                            record_video_trigger = lambda x: x ==0,
                            video_length = 2000,
                            name_prefix = model_save_name)

# Do not update the agent during evaluation
eval_env.training = False

# No need to normalize the reward during the evaluation
eval_env.norm_reward = False
```

6.2 Evaluate the agent

```
from stable_baselines3.common.evaluation import evaluate_policy

# Load the agent
model = A2C.load(model_save_name)

# Evaluate
mean_reward, std_reward = evaluate_policy(model, eval_env)

# Print results
print(f"The mean_reward: {mean_reward:.2f} | The standard deviation: {std_reward:.2f}")
```

```
The mean_reward: -45.00 | The standard deviation: 15.00
```

Step 7: Push to Hub

7.1 Login to Hub

- https://huggingface.co/settings/tokens

```
from huggingface_hub import notebook_login

notebook_login()
!git config --global credential.helper store
```

Show hidden output

7.2 Push to Hub

- P/S: Need to upload the video manually to the repository by renaming to "replay.mp4"

```
from huggingface_sb3 import package_to_hub

```

```
package_to_hub(model = model,
               model_name = model_save_name,
               model_architecture = "A2C",
               env_id = env_id,
               eval_env = eval_env,
               repo_id = f"wengti0608/{model_save_name}",
               commit_message = "Initial Commit")
```

Show hidden output

Challenge 2: PandaPickAndPlace-v3

- Environment Documentations
  - Visualise environment and usage: https://github.com/qgallouedec/panda-gym
  - Brief explanation on action and rewards: https://panda-gym.readthedocs.io/en/latest/usage/environments.html

Step 1: Visualize the environment

```
import gymnasium as gym
import panda_gym

env = gym.make("PandaPickAndPlace-v3")

print(f"Randomly sample a state: {env.observation_space}")

print(f"Randomly sample an action: {env.action_space}")

```

```
Randomly sample a state: Dict('achieved_goal': Box(-10.0, 10.0, (3,), float32), 'desired_goal': Box(-10.0, 10.0, (3,), float32), 'ob
Randomly sample an action: Box(-1.0, 1.0, (4,), float32)
```

Step 2: Create a vectorised environment with normalization

```
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3.common.vec_env import VecNormalize

# Create a vectorised environment
env_id = "PandaPickAndPlace-v3"

env = make_vec_env(env_id = env_id,
                   n_envs = 4)

# Wrap it to normalize input and reward
env = VecNormalize(venv = env,
                   norm_obs = True,
                   norm_reward = True,
                   clip_obs = 10.0)
```

Step 3: Create an A2C model

```
from stable_baselines3 import A2C

model = A2C(policy = "MultiInputPolicy",
            env = env,
            verbose = 1)
```

```
Using cuda device
```

Step 4: Training

```
model.learn(1000000)
```

Show hidden output

Step 5: Save the model

```
model_save_name = f"a2c-{env_id}"

model.save(model_save_name)
env.save("vec_normalize.pkl")
```

Step 6: Evaluation

6.1 Create the evaluation environment

```
from stable_baselines3.common.vec_env import DummyVecEnv, VecVideoRecorder
from stable_baselines3.common.monitor import Monitor

# Create the environment for both evaluation and pushing to hub (including video recording)
eval_env = DummyVecEnv([lambda : Monitor(gym.make(env_id, render_mode = "rgb_array"))])

# Load the normalization statistics obtained from training
eval_env = VecNormalize.load("vec_normalize.pkl", eval_env)

# Use a wrapper to manually record videos (due to errors in `package_to_hub`)
# Once the video is recorded, it can be manually uploaded to the repository and renamed as "replay.mp4"
# This video recording feature only get triggered when used in `package_to_hub`
eval_env = VecVideoRecorder(eval_env,
                            video_folder = "./videos/",
                            record_video_trigger = lambda x: x ==0,
                            video_length = 2000,
                            name_prefix = model_save_name)

# Do not update the agent during evaluation
eval_env.training = False

# No need to normalize the reward during the evaluation
eval_env.norm_reward = False
```

6.2 Evaluate the agent

```
from stable_baselines3.common.evaluation import evaluate_policy

# Load the agent
model = A2C.load(model_save_name)

# Evaluate
mean_reward, std_reward = evaluate_policy(model, eval_env)

# Print results
print(f"The mean_reward: {mean_reward:.2f} | The standard deviation: {std_reward:.2f}")
```

```
The mean_reward: -45.00 | The standard deviation: 15.00
```

Step 7: Push to Hub

7.1 Login to Hub

- https://huggingface.co/settings/tokens

```
from huggingface_hub import notebook_login

notebook_login()
!git config --global credential.helper store
```

Show hidden output

7.2 Push to Hub

- P/S: Need to upload the video manually to the repository by renaming to "replay.mp4"

```
from huggingface_sb3 import package_to_hub

package_to_hub(model = model,
               model_name = model_save_name,
               model_architecture = "A2C",
               env_id = env_id,
```

```
                eval_env = eval_env,
                repo_id = f"wengti0608/{model_save_name}",
                commit_message = "Initial Commit")
```

Show hidden output