Homework 5 Template

Use this template to record your answers for Homework 5. Add your answers using LaTeXand then save your document as a PDF to upload to Gradescope. You are required to use this template to submit your answers. **You should not alter this template in any way** other than to insert your solutions. You must submit all 11 pages of this template to Gradescope. Do not remove the instructions page(s). Altering this template or including your solutions outside of the provided boxes can result in your assignment being graded incorrectly.

You should also export your code as a .py file and upload it to the **separate** Gradescope coding assignment. Remember to mark all teammates on **both** assignment uploads through Gradescope.

Instructions for Specific Problem Types

On this homework, you must fill in blanks for each problem. Please make sure your final answer is fully included in the given space. **Do not change the size of the box provided.** For short answer questions you should **not** include your work in your solution. Only provide an explanation or proof if specifically asked.

Fill in the blank: What is the course number?

10-703

Problem 0: Collaborators

Enter your team members' names and Andrew IDs in the boxes below. If you worked in a team with fewer than three people, leave the extra boxes blank.

Name 1:	Ethan Cheong	Andrew ID 1:	echeong
Name 2:	Al Hassan	Andrew ID 2:	akhassan
Name 3:	Clement Ou	Andrew ID 3:	clemento

Problem 1: MuZero (100 pt)

1.1.1: MCTS child selection (10 pt)

Insert code for MCTS child selection.

```
def select_child(config, node: Node, min_max_stats):
    scores = []
    for action, child in node.children.items():
        score = ucb_score(config, node, child, min_max_stats)
        scores.append((score, action, child))
    # Select action-child pair with max score
    _, action, child = max(scores, key=lambda x: x[0])
    return action, child
```

1.1.2: MCTS expand root/child (20 pts)

Insert code for root and child expansion.

```
def expand_root(node, actions, network, current_state):
    value, reward, policy_logits, hidden_representation =
        network.initial_inference(np.expand_dims(current_state, axis=0))

    node.hidden_representation = hidden_representation
    node.reward = reward
    node.expanded = True

    policy = tf.nn.softmax(policy_logits).numpy()[0]

    for a, p in zip(actions, policy):
        node.children[a] = Node(prior=p)

    return value

def expand_node(node, actions, network, parent_state, parent_action):
    value, reward, policy_logits, hidden_representation =
        network.recurrent_inference(parent_state, parent_action)

    node.hidden_representation = hidden_representation
    node.reward = reward
    node.expanded = True

    policy = tf.nn.softmax(policy_logits).numpy()[0]

    for a, p in zip(actions, policy):
        node.children[a] = Node(prior=p)

    return value
```

1.1.3: MCTS backpropagation (5 pts)

Insert code for MCTS backpropagation.

```
def backpropagate(path, value, discount, min_max_stats):
    for node in reversed(path):
        node.value_sum += value
        node.visit_count += 1
        value = node.reward + discount * value
```

1.1.4: MCTS softmax sampling (5 pts)

Insert code for MCTS softmax sampling.

```
def softmax_sample(visit_counts, temperature):
    # visit_counts = [(visit_count, action), ...]
    counts = np.array([vc for vc, a in visit_counts])
    if temperature == 0:
        # Argmax selection
        max_index = np.argmax(counts)
        return visit_counts[max_index][1]
    else:
        # Compute distribution p(a) = N(a)^(1/T) / sum(N(b)^(1/T))
        counts = counts ** (1.0 / temperature)
        probs = counts / np.sum(counts)
        return np.random.choice([a for _, a in visit_counts], p=probs)
```

1.1.5: Network weight updates (20 pts)

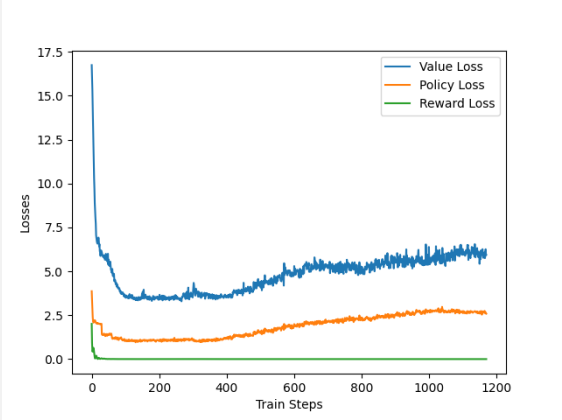
Insert code for network weight updates.

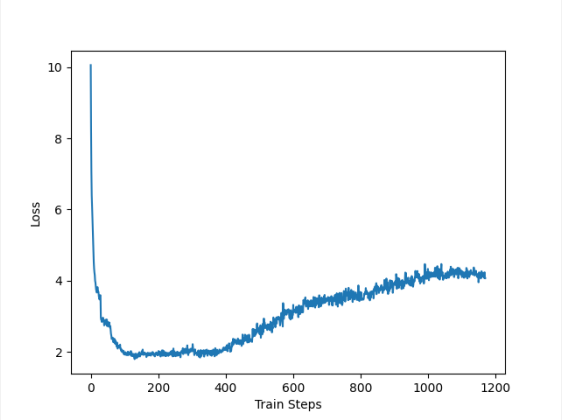
```
def update_weights(config, network, optimizer, batch, train_results):
    (state_batch, targets_init_batch,
        targets_recurrent_batch, actions_batch) = batch
    state_batch = tf.convert_to_tensor(state_batch, dtype=tf.float32)
    init_values, init_rewards, init_policies = zip(*targets_init_batch)
    init_values = tf.convert_to_tensor(init_values, dtype=tf.float32)
    init_policies = tf.convert_to_tensor(init_policies, dtype=tf.float32)
    init_values_support = network._scalar_to_support(init_values)
    num_unroll_steps = len(targets_recurrent_batch)
    with tf.GradientTape() as tape:
        hidden_rep, pred_value, pred_policy_logits =
            network.initial_model(state_batch)
        value_loss = tf.nn.softmax_cross_entropy_with_logits(
            logits=pred_value, labels=init_values_support)
        value_loss = tf.reduce_mean(value_loss)
        policy_loss = tf.nn.softmax_cross_entropy_with_logits(
            logits=pred_policy_logits, labels=init_policies)
        policy_loss = tf.reduce_mean(policy_loss)
        reward_loss = 0.0
        total_loss = 0.25 * value_loss + policy_loss
        current_hidden = hidden_rep
        for step_idx, (step_values) in enumerate(targets_recurrent_batch):
            step_values, step_rewards, step_policies = zip(*step_values)
            step_values = tf.convert_to_tensor(step_values,
                dtype=tf.float32)
            step_rewards = tf.convert_to_tensor(step_rewards,
                dtype=tf.float32)
            step_policies = tf.convert_to_tensor(step_policies,
                dtype=tf.float32)
            step_values_support = network._scalar_to_support(step_values)
            step_actions = actions_batch[step_idx]
            step_actions = tf.convert_to_tensor(step_actions,
                dtype=tf.int32)
            action_one_hot = tf.one_hot(step_actions,
                depth=config.action_space_size, dtype=tf.float32)
            conditioned_hidden = tf.concat([current_hidden,
                action_one_hot], axis=1)
            next_hidden, pred_reward, pred_value, pred_policy_logits =
                network.recurrent_model(conditioned_hidden)
```

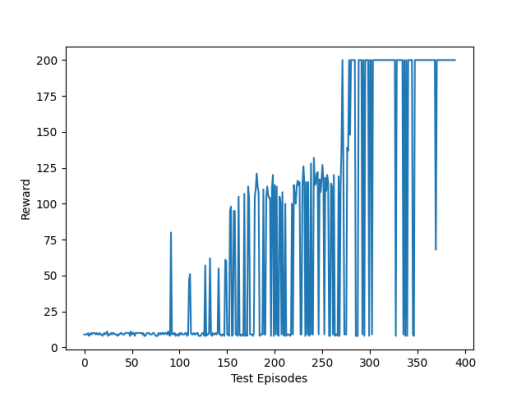
```
            step_value_loss = tf.nn.softmax_cross_entropy_with_logits(
                logits=pred_value, labels=step_values_support)
            step_value_loss = tf.reduce_mean(step_value_loss)
            step_reward_loss = tf.reduce_mean((tf.squeeze(pred_reward,
                axis=1) - step_rewards) ** 2)
            step_policy_loss = tf.nn.softmax_cross_entropy_with_logits(
                logits=pred_policy_logits, labels=step_policies)
            step_policy_loss = tf.reduce_mean(step_policy_loss)
            step_loss = 0.25 * step_value_loss + step_policy_loss +
                step_reward_loss
            step_loss = scale_gradient(step_loss, 1.0 / num_unroll_steps)
            total_loss += step_loss
            value_loss += step_value_loss
            reward_loss += step_reward_loss
            policy_loss += step_policy_loss
            current_hidden = scale_gradient(next_hidden, 0.5)
        total_value_loss = value_loss
        total_policy_loss = policy_loss
        total_reward_loss = reward_loss
        train_results.total_losses.append(total_loss.numpy())
        train_results.value_losses.append(total_value_loss.numpy())
        train_results.policy_losses.append(total_policy_loss.numpy())
        train_results.reward_losses.append(total_reward_loss.numpy())
        variables = network.cb_get_variables()()
        grads = tape.gradient(total_loss, variables)
        optimizer.apply_gradients(zip(grads, variables))
    network.train_steps += 1
```

1.2: Running MuZero (20 pts)

Run MuZero, provide the three plots, reason about policy loss behavior.





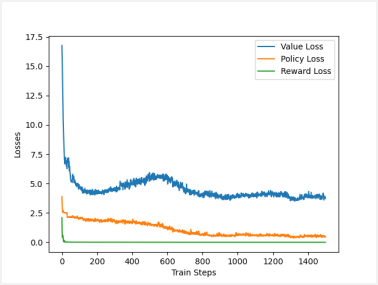


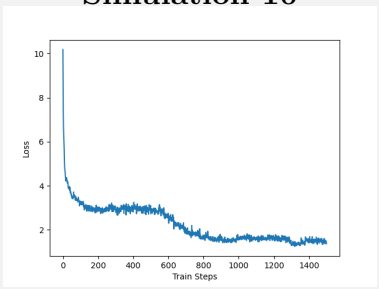
The policy loss initially decreases sharply in the first 100 training steps, suggesting the network quickly learns basic action preferences for the task. After the initial decline, we observe a steady increase in policy loss from around step 200 to 1200. This behavior can be attributed to more exploration as the value estimates become more reliable, the policy network adapting to complex situations as the agent learns to balance longer, and potential distribution shift as the agent encounters states it hasn't seen before. Despite rising policy loss, the test rewards plot shows consistent improving performance, reaching the maximum reward of 200 consistently by the end. This suggests that the increasing policy loss doesn't necessarily indicate worse performance, but rather reflects the complexity of the learned policy.

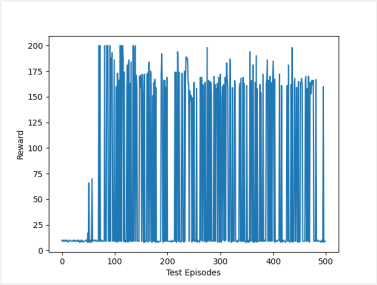
1.3: Effects of Hyperparameters (10 pts)

Run MuZero with different hyperparameters and provide three plots for each value (nine total). Describe and explain the effects of the parameter.

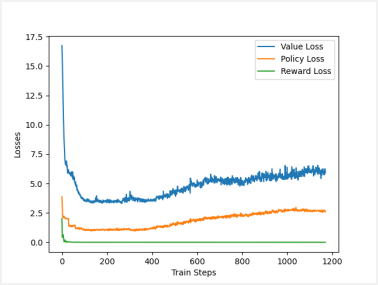
Simulation 10

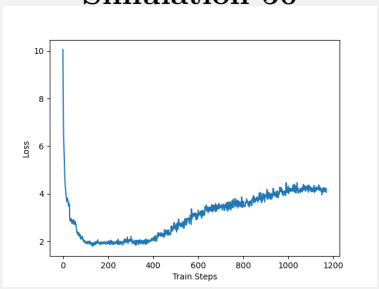


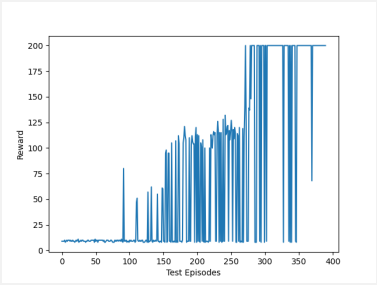




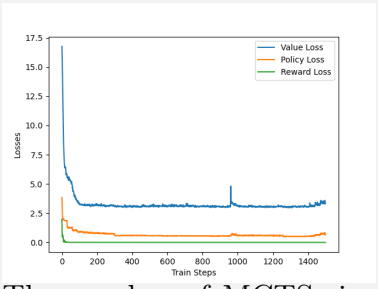
Simulation 50

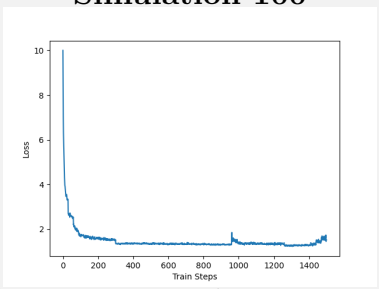


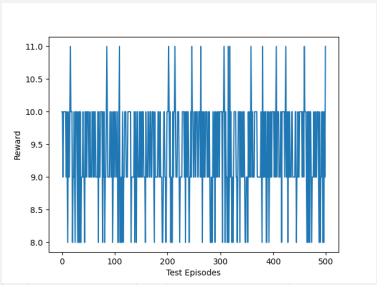




Simulation 100







The number of MCTS simulations impacted training stability, with less simulations showing more volatile training and higher loss values. More simulations led to better stability with smoother loss curves. The number of simulations also impacted performance, with 50 being a good balance between exploration and exploitation, and the only one that reached the maximum reward of 200 consistently. Lower simulations had high variance in test rewards, suggesting inconsistent policy, and high simulations had lower average rewards, possibly due to over-exploration or excessive computational budget leading to conservative behavior.

1.4: Conceptual Questions (10 pts)

Respond to the three questions.

Feedback

Feedback: You can help the course staff improve the course for future semesters by providing feedback. What was the most confusing part of this homework, and what would have made it less confusing?

Solution

The gym environment should be updated to a more recent version, as well as packages. Two of my teammates had M1 Macs and were unable to run the code as is.

Time Spent: How many hours did you spend working on this assignment? Your answer will not affect your grade. Please average your answer over all the members of your team.

Alone	6
With teammates	2
With other classmates	0
At office hours	0