National College of Ireland

Configuration Manual

MSc Research Project
MSc Cloud Computing

Johns Thomas
Student ID: 22203389

School of Computing
National College of Ireland

Supervisor: Sai Emani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	JOHNS THOMAS
Student ID:	22203389
Programme:	MSc Cloud Computing **Year:** 2024
Module:	Research Project
Lecturer:	Sai Emani
Submission Due Date:	12th August 2024
Project Title:	Configuring serverless functions using Q learning and Deep Q learning algorithms
Word Count:	1538 **Page Count:** 7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:

Date: 12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Johns Thomas
Student ID: 22203389

1 Introduction

This manual provides detailed explanation and instructions for setting up, configuring, and running Q learning and Deep Q-learning agents to support serverless function configuration. It is intended to ensure the reproducibility of the work and for researchers and developers interested in similar applications. The manual covers the installation of necessary software, configuration of learning parameters, execution of the agents in AWS lambda, and evaluation of results.

2 System Requirements

2.1 Hardware Requirements

- CPU: AMD Ryzen 7 or equivalent.
- Memory: Minimum 16 GB RAM.
- Storage: At least 10 GB of available space.

2.2 Software Requirements

Operating System	Windows 11 or similar
Serverless Environment	AWS Lambda
Programming Language	Python 3.11 or later
Logging & Monitoring Service	AWS CloudWatch
Python libraries	Boto3, Numpy, Pandas, TensorFlow 2.x
Object Storage	AWS S3

Table 1: Components and corresponding software used in project

Table 2 gives the overview of the artifacts developed and available in github repository https://github.com/johns-thomas/ric_implementation.git

qlearning_agent.py	Python code for training the Q-learning agent
dqn.py	Python code for training the Deep Q-Learning agent
helper.py	Utility methods for invoking lambda functions and retrieving logs.

graph.py, graph-2.py	Contains code for evaluating the results
image_processing_tasks/*	Folder containing image processing tasks used for training the RL agents.

Table 2: Details of artifacts

3 Installation and Set Up

This section will guide you through the setting up the environment required for the training of Q-learning and Deep Q-learning agents. Also, instructions on how to install the software dependencies are included.

3.1 AWS Resource Setup

This section assumes that the user has AWS account and appropriate permissions to create and manage AWS Lambda, S3, and CloudWatch resources.

3.1.1 AWS S3 bucket

The purpose of the S3 bucket is to store the images required for processing by serverless functions. AWS S3 bucket *n.d.*

Step 1: Log in to the AWS Management Console.
Step 2: Navigate to the S3 service and create a new bucket, say ‘my_imagebucket’. The bucket name must be unique.
Step 3: To my_imagebucket’, upload images of different sizes from Flickr-Faces-HQ image dataset.

3.1.2 AWS Lambda

In this research project, AWS lambda acts as the serverless environment for the Q learning and Deep Q-learning agent to interact with and learn. You must deploy the image processing functions given in the folder image_processing_tasks under github repository as AWS lambda functions. The following steps will walk you through the process. The steps have been followed as per the documentation given in AWS lambda documentation *n. d.*

Step 1: Navigate to the AWS Lambda Console.
Step 2: From console choose ‘Create function’:
   a. Select "Author from scratch".
   b. Enter a function name denoting the image processing function you are about to deploy.
   c. Choose Python 3.11 for the runtime.
   d. Set up the execution role with appropriate permissions (Lambda, S3, CloudWatch). You can choose ‘Create a new role with basic Lambda permissions’ and later add permissions for S3 and CloudWatch to this role.
   e. Click ‘Create function’ Refer Figure 1.
Step 3: Copy the rotate_image.py function from the folder image_processing_tasks under github repository and paste it into the Code source section of newly created lambda function.

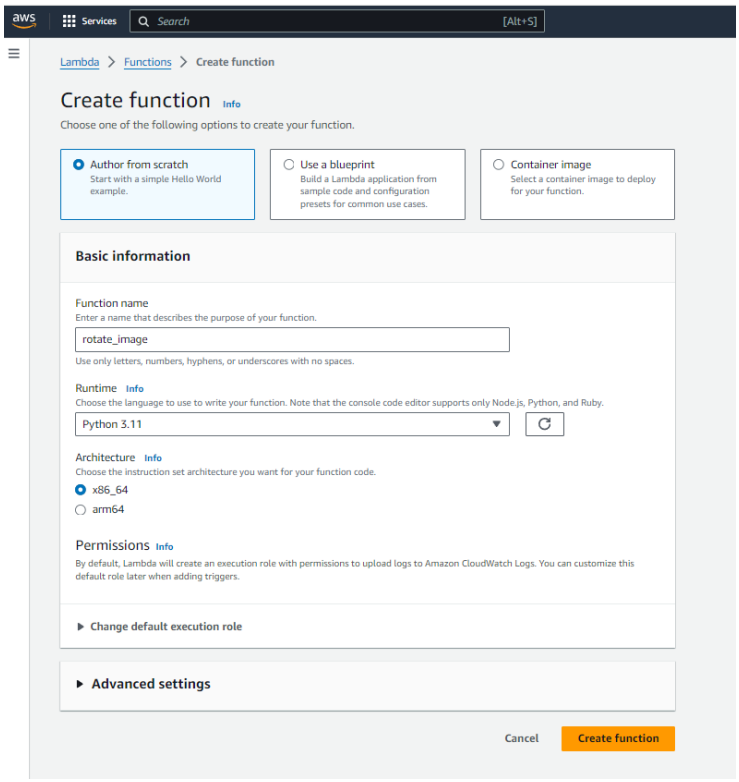

Figure 1: Creating Lambda function on AWS

Step 4: Now for the image processing tasks, python library has to be added as dependency. In AWS lambda, dependency libraries can be added using layers. Under layers section, choose 'Add a Layer'. Refer Figure 2.

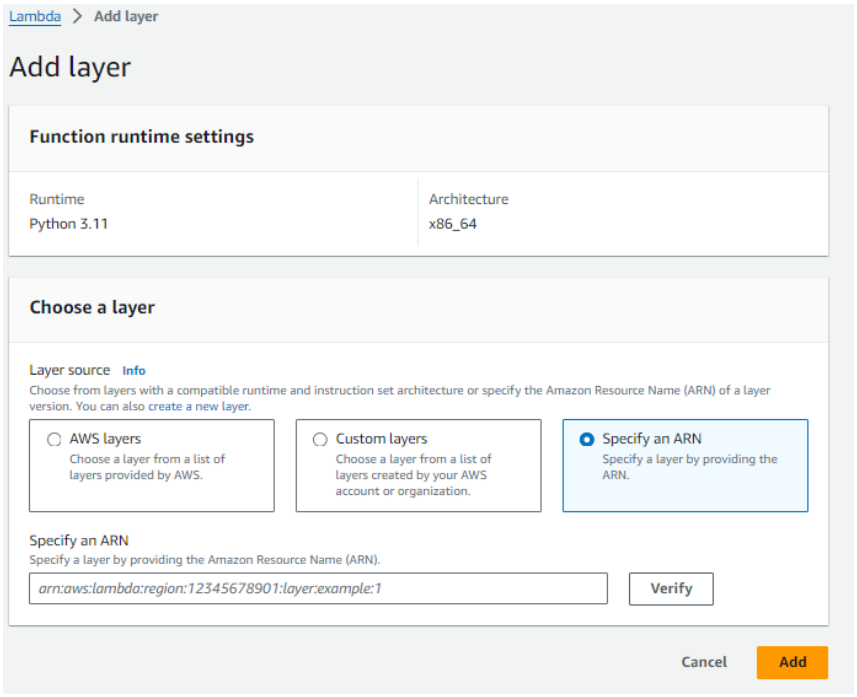

Figure 2: Adding layers in AWS lambda function

Step 5: Choose option Specify an ARN and give the following ARN “arn:aws:lambda:ap-south-1:770693421928:layer:Klayers-p311-Pillow:4”

Step 6: Click verify. Once verified click Add. Now the lambda function is ready for execution and you view the layer in the layers section, Figure 3.

Figure 3: Layer in Lambda function

Repeat the Steps for each of the image processing functions in the image_processing_tasks folder.

3.1.3 AWS CloudWatch

The CloudWatch collects the logs of the execution of lambda function. Follow the given steps as well as consult the AWS CloudWatch documentation *n.d.*

Step 1. Navigate to CloudWatch console.
Step 2. Go to Log groups
Step 3. Create new log group with name /aws/lambda/<lambda funciton name>

Ensure that lambda function has proper rights to write to the CloudWatch log group created.

3.2 Installation guide

Step 1. Install Python 3.11 from python.org.
Step 2. Optional, Create and activate python virtual environment
Step 3. Install TensorFlow using pip : **pip install tensorflow**
Step 4. Install numpy, pandas and matpltlib : **pip install numpy pandas matplotlib**
Step 5. Verify that Tensorflow and other libraries are installed properly
Step 6. Ensure that git is available in the system. Now clone the project artifact from Github repository by executing git clone https://github.com/johns-thomas/ric_implementation.git

4 Configuration Settings

The Q-learning and Deep Q-learning agents can be configured using various parameters. These parameters affect the training time as well as the resultant models.

The following section discusses the parameters used during the training of the Q-learning and Deep Q learning agents.

4.1 Q learning agent configuration

Various configurations available for Q learning agent present in qlearning_agent.py is shown in Figure 4.

Variable q_table_file_path is the name for the Q-table generated during the training, which is saved in .npy format.

Variable state_data_path saves the state data during the training of Q -learning agent for future reference in text format.

```
#Configurations
q_table_file_path = 'q_table-new.npy'
state_data_path='qstate-new.txt'
results_file='new_results_q.json'

learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.01
num_episodes = 1000

log_group_name ='/aws/lambda/x22203389-ric-rotation'
func_name='x22203389-ric-rotation'
bucket_name = 'x22203389-ric'
folder_path = '51000/'
```

Figure 4: Configurations for Q learning agent

Variable results_file represents name of the results of training in a json file, which has data like episode number, reward, memory configurations, timeout, cost incurred.

The learning rate of the q learning can be configured by setting variable learning_rate. In the research learning rate is taken as 0.1. The discount factor for Q learning agent is taken as 0.9 by setting the variable discount_factor. Additionally, you can configure epsilon which controls the exploration rate of the q learning agent. In the project it is taken to be 0.01. Also, depending on the need, the number of episodes can also be increased.

The name of the serverless function and its associated cloudwatch log group name can be configured by setting variable func_name and log_group_name respectively. The S3 bucket containing the images can be configured by setting variable bucket_name and folder_path. folder_path is just the name of the images folder.

4.2 Deep Q-learning agent (DQN) configuration

Various configurations available for Deep Q learning agent present in qlearning_agent.py is shown in Figure 5.

The learning rate of the DQN learning agent can be configured by setting variable learning_rate and is taken as 0.1. Discount factor for Deep Q learning agent is taken as 0.9 by setting the variable gamma. Additionally, you can configure epsilon which controls the exploration rate of the DQN learning agent. In the project it is taken to be 0.1. The exploration rate can be decayed over episodes. epsilon_decay determines how to reduce the

exploration value and epsilon_min determines minimum possible exploration rate for the training.

The name of the serverless function and its associated cloudwatch log group name can be configured by setting variable func_name and log_group_name respectively. The S3 bucket containing the images can be configured by setting variable bucket_name and folder_path. folder_path is just the name of the images folder.

```
# DQN parameters
gamma = 0.9
epsilon = 0.1
epsilon_min = 0.01
epsilon_decay = 0.995
learning_rate = 0.001
batch_size = 32
memory_buffer = deque(maxlen=2000)
num_episodes = 100
max_steps_per_episode = 10

model_filename = 'dqn_new_model.h5'
results_file='dqn_results.json'
q_table_file_path = 'dqn_table.txt'
target_model_file= 'dqn_new_model.h5'

log_group_name ='/aws/lambda/x22203389-ric-rotation'
func_name='x22203389-ric-rotation'
bucket_name = 'x22203389-ric'
folder_path = '51000/'
```

Figure 5: Configuration options for DQN learning agent

The model_filename represents the deep neural network undergoing the training and it can be saved for running long training cycles. The target_model_file variable represents the actual deep neural network which holds the information about the Q-table. When training of DQN agent finishes it is stored as '.h5' file. This deep neural network model file can be deployed to candidate serverless functions to optimize their configurations.

5 Running the Software

To run the training of Q learning agent.
Step 1: Ensure that you have set up AWS credentials to use AWS SDK in your system.
Step 2: Go to command line
Step 3: Run **python qlearning_agent.py**
Step 4: Wait until the training finishes, it may take up to 24 hours to complete 100 episodes
Step 5: After the training, use new_results_q.json for analysis, also you can find the qtable with name q_table-new.py which can be used for configuring other serverless functions.

To run the training of DQN learning agent.

Step 1: Ensure that you have set up AWS credentials to use AWS SDK in your system.
Step 2: Go to command line
Step 3: Run **python dqn.py**
Step 4: Wait until the training finishes, it may take upto 28 hours to complete 100 episodes
Step 5: After the training, use dqn_results.json for analysis.
Step 6: Use the dqn_new_model for configuring other serverless functions.

Note: The q learning and DQN learning agent take some iterations to optimally configure unseen serverless functions not used while training as in the case of any reinforcement learning algorithm.

Using the results.json file obtained after the completion of learning by RL agents, you can analyse the performance of the Q learning and Deep Q learning in terms of rewards colled per episodes, cost incurred , execution duration, memory configured, memory used and timeouts.

References

AWS CloudWatch documentation *n.d.* *https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Working-with-log-groups-and-streams.html* Accessed on August 3, 2024.

AWS Lambda (2024), 'AWS Lambda', https://aws.amazon.com/lambda/. Accessed: July 15, 2024.

AWS Lambda (2024) documentation *n.d.*, https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html ,Accessed on August 3, 2024.

AWS S3 bucket *n.d.* *https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html* , Accessed on August 3, 2024.

Flickr-Faces-HQ Dataset (FFHQ), https://github.com/NVlabs/ffhq-dataset?tab=readme-ov-file#readme Accessed: July 17, 2024.