Image Classification using AWS

Team-4: Members

Aditya Goverdhana (agoverdh@kent.edu)

Balakrishna Phani Kommanaboina (bkommana@kent.edu)

Naveena Kanderi (nkanderi@kent.edu)

Jagadeesh Karri (jkarri1@kent.edu)

April 29, 2023

Instructor: Prof. Debobroto Das Robin

CAPSTONE PROJECT (CS-69099-001)

KENT STATE UNIVERSITY

Department of Computer Science

Introduction

Problem Statement

The project aims at building an elastic web application that can automatically scale out and scale in on-demand and cost-effectively by using cloud resources. The resources used were from Amazon Web Services. It is an image classification application exposed as a Rest Service for the clients to access.

Objectives

The application takes the images and returns the predicted output using an image classification model through the AWS resources, an IaaS provider. AWS as an IaaS provider offers a variety of compute, storage and message services. So the tasks involved designing the architecture, implementing RESTful Web Services, a load balancer that scales in and scales out EC2 instances at App Tier according to the demand of the user.

Milestones

The initial milestones involve

- Design and build an interactive system for User.

- Explore and understand the AWS services and improve the infrastructure.

- Enable AWS Services and run some tests to see storage and computation performance.

Tools Used

- AWS services (EC2, SQS, S3).

- Pretrained Image classification model.

- Web Services (HTML, CSS, Flask).

- Testing for resources.

- Python for backend of the architecture.

Literature Review

Image Recognition is an essential component of various applications, such as facial recognition, object detection, and product recommendation. With the growing demand for image recognition, many researchers have focused on developing image recognition systems using cloud and edge computing. This literature review summarizes and analyzes five research papers that discuss image recognition as a service.

In the paper "Cloud Strategies for Image Recognition," da Costa and Pisa [3] discuss various cloud strategies for image recognition. The authors analyze the features of different cloud providers, such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure, and evaluate their suitability for image recognition tasks. The paper emphasizes the importance of cloud computing in image recognition and provides insights into cloud-based image recognition services.

In "Real-Time Object Detection with TensorFlow Model Using Edge Computing Architecture," N et al. [4] propose a real-time object detection system using a TensorFlow model and edge computing architecture. The paper presents a system architecture that utilizes edge computing to reduce latency and improve response time. The authors evaluate the system's performance using a dataset of real-world images and demonstrate the effectiveness of the system in detecting objects in real-time.

In "The Utilization of Cloud Computing for Facial Expression Recognition using Amazon Web Services," Rafael et al. [5] discuss the utilization of cloud computing for facial expression recognition. The authors use Amazon Web Services to develop a facial expression recognition system and evaluate the system's performance using a dataset of facial expressions. The paper demonstrates the effectiveness of cloud-based facial expression recognition systems and highlights the potential of such systems in various applications, such as security and entertainment.

In "Object Detection and Recognition using Amazon Rekognition with Boto3," Sharma

[6] discusses object detection and recognition using Amazon Rekognition with Boto3, a software development kit for Python developers. The paper presents an overview of the Amazon Rekognition service and explains how to use the service with Boto3. The author demonstrates the effectiveness of the Amazon Rekognition service in detecting and recognizing objects in real-time, providing a practical guide for developers.

Finally, Suguna et al. [7] present "An Efficient Real-time Product Recommendation using Facial Sentiment Analysis." The paper proposes an efficient product recommendation system that utilizes facial sentiment analysis to determine a user's emotional state. The system uses the Microsoft Azure Face API to extract facial features and sentiment scores, which are then used to recommend products. The authors evaluate the system's performance using a dataset of user reviews and demonstrate the system's effectiveness in generating relevant product recommendations.

In conclusion, these five research papers provide insights into image recognition as a service using cloud and edge computing. The papers demonstrate the effectiveness of cloud-based and edge-based image recognition systems and provide practical guidance for developers. The papers also highlight the potential of image recognition in various applications, such as facial recognition, object detection, and product recommendation.

Our project aims to create an elastic web application that uses cloud resources to develop an image recognition application. The project involves designing the architecture and implementing RESTful Web Services to provide access to the application. The use of cloud resources to provide scalable and cost-effective solutions is a common theme in the above 5 research papers. The above research papers emphasize the use of cloud-based services, including AWS services, for scalable and cost-effective image recognition solutions.

Project Background

Computer vision is a field in artificial intelligence that deals with the analysis and interpretation of visual data, particularly images. One of the primary tasks in computer vision is image classification, where images are categorized into predefined classes or labels through machine learning algorithms. Although similar to image recognition, which also uses machine learning algorithms to interpret visual data, image classification differs in that it categorizes entire images, while image recognition identifies and labels objects within an image.
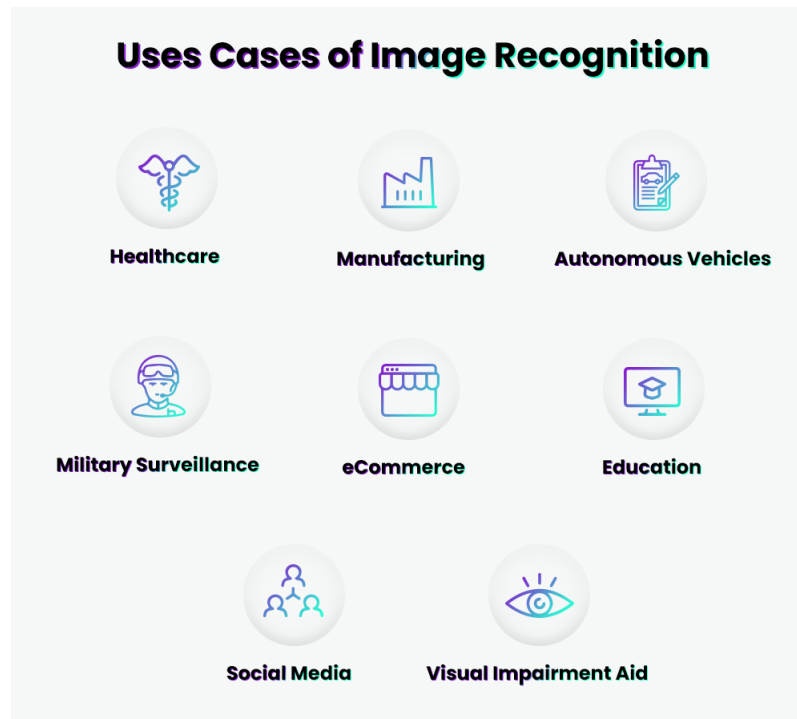

Figure 1: Some use-cases of Image Recognition.Ariwala [1]

Image classification, like image recognition, also has a wide range of applications across various industries such as healthcare, automotive, retail, security, and entertainment as shown in the Figure 1. Image classification technology can be used for various tasks such as object detection, medical diagnosis, and product recognition. Some examples include:

- **Medical Diagnosis:** Image classification can be used for medical imaging analysis, identifying and diagnosing diseases like cancer, or detecting anomalies in medical images.

- **Agriculture:** Image classification can be used for crop and soil analysis, plant disease detection, and monitoring crop growth and health.

- **Manufacturing:** Image classification can be used for product quality control, identifying defects, and ensuring consistency in production lines.

- **Retail:** Image classification can be used for product recognition, inventory management, and customer behavior analysis, such as tracking the popularity of certain products.

- **Security:** Image classification can be used for object detection, facial recognition, and surveillance monitoring, such as identifying security threats and detecting suspicious behavior.

- **Environmental Monitoring:** Image classification can be used for land use and land cover mapping, identifying deforestation or other changes in the environment, and monitoring wildlife populations.

IaaS (Infrastructure as a Service) is a cloud computing model in which cloud service providers offer virtualized computing resources such as servers, storage, and networking to users on a pay-per-use basis. IaaS allows users to scale up or down their computing resources as needed, without having to invest in their own infrastructure. Examples of IaaS providers include Amazon Web Services, Microsoft Azure, and Google Cloud Platform.

IaaS services have a wide range of applications across various industries, such as:

- **Web hosting**: IaaS providers can host websites and web applications in the cloud, providing scalable and reliable infrastructure.

- **Big data**: IaaS providers can provide the computing resources needed to process and analyze large volumes of data.

- **Disaster recovery**: IaaS providers can provide backup and recovery solutions in the event of a disaster or outage.

- **DevOps**: IaaS providers can provide the infrastructure needed for software development and testing.

- **Machine learning**: IaaS providers can provide the computing resources needed to train and run machine learning models.

Auto scaling is a key feature of IaaS providers that allows for the automatic adjustment of computing resources based on user demand. Auto scaling ensures that users are not paying for resources that they do not need, and that they have sufficient resources during periods of high demand. This feature can be particularly useful for applications with unpredictable usage patterns, as it enables the system to automatically adjust to changing needs without requiring manual intervention.

Infrastructure

The infrastructure described is a common architecture for building web applications that require image classification capabilities. The infrastructure is divided into two tiers: the Web-Tier and the Application-Tier.

The Web-Tier of the system consists of a simple User Interface, where a user can interact with our system. In our system, user can upload images they want to classify. The Application-Tier contains core functionality of the system, image classification. It also handles business logic, database manipulation functions (CRUD), and communicates with other resources. The overall system architecture of our project can be viewed in the Figure 2.
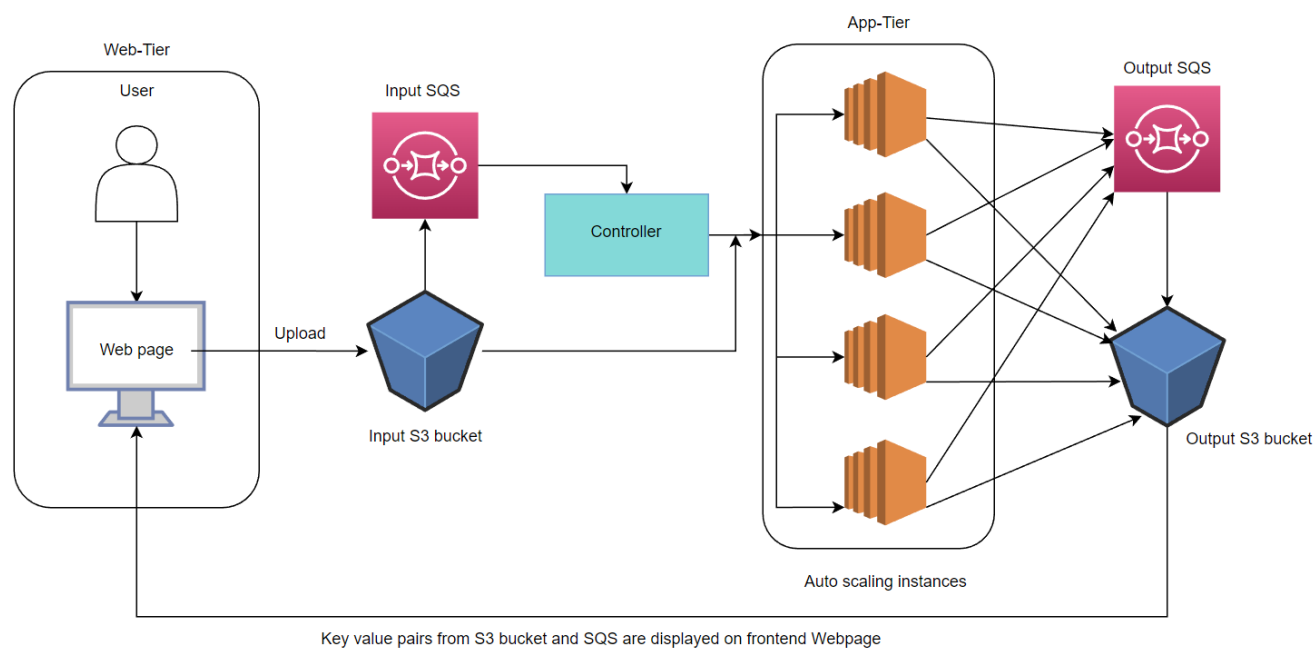

Figure 2: System Architecture.

The number of EC2 instances needed depends on the frequency of data uploads. Scaling in and scaling out of instances are determined based on the number of input images

uploaded, which are fed into the SQS queue and the number of messages are calculated. Scaling in is managed by the controller, which automatically terminates instances when there are no more messages left in the input SQS. Scaling out is implemented using the number of running instances and the estimated number of visible SQS messages. Instances are created with the AMI, and the scaling out logic is designed not to create more than 15 instances. To summarize, the number of instances created depends on the number of SQS messages. If there are less than 15 messages, the number of instances created will be the number of messages minus the number of currently running instances. If there are more than 15 messages, 17 instances will run to handle the increased load and avoid errors during the processing of larger images.

This infrastructure utilizes AWS' EC2, SQS, and S3 resources. Specifically, the App Tier relies on S3 and SQS to store and transmit data, respectively. Once the classification results are generated, they are sent as key-value pairs to the output SQS and output S3 bucket to be displayed on the web page.

The infrastructure is a standard two-tier architecture used to develop web applications that incorporate image classification features. The Web-Tier provides a user interface for uploading images, while the Application-Tier contains core functionality, such as image classification, business logic, and database manipulation functions. AWS EC2, SQS, and S3 resources are utilized to create this infrastructure. Scaling in and out of resources is determined by the number of incoming images, which are processed through SQS, and the number of running instances. The infrastructure is designed to create up to 15 instances, and the number of instances created is based on the number of incoming images, with automatic termination of instances when the queue is empty. By leveraging AWS services and this scalable infrastructure, web applications with image classification capabilities can be built in a cost-effective and efficient manner.

Implementation

The following is a detailed description of the implementation steps involved in building a web application with image classification capabilities on AWS using a two-tier architecture with a Web-tier and App-tier.

Setup

Web-tier:

To implement the front-end part of the application, we utilized HTML and CSS to design a visually appealing static web page, with HTML providing the structural framework and CSS enabling customization of the layout, font, and color schemes. We integrated the Flask framework to manage communication between the front-end and back-end components, with essential tools and libraries provided to handle various user requests. Using the boto3 libraries, we established a seamless connection between the front-end and App-tier EC2 instance, allowing us to interact with the instance and upload images to the input S3 bucket for processing by the image classification model. Our approach was effective in creating a user-friendly interface while ensuring seamless integration with the App-tier EC2 instance.

App-tier EC2 instances:

App-tier EC2 instances were created following the instructions Barr [2] provided by AWS to create an AMI with a t2.small instance type and a root key attached. The default location was set to us-east-1 and the results were stored in JSON format. On the App-tier instances, we installed the necessary libraries listed in "requirements.txt" and added "worker.py" and "image classification.py", which contains a pre-trained ResNet model with ImageNet labels from the Keras library used for image classification. The model has been

pre-trained on the ImageNet dataset, which includes over a million labeled images, making it a powerful tool for quickly and accurately classifying images without extensive training on a custom dataset. To run the scripts automatically every minute, we used cron tab and executed the command `* * * * * python3 /home/ubuntu/worker.py > ./result.txt`.

Execution Steps

1. First, we started the Web-tier application manually on the terminal using the command `python3 app2.py`, which started the Web-tier.

2. Next, the controller is started on the terminal using the command
   `python3 controller.py`. The script retrieves information about an SQS queue, counts the number of running instances, and starts new instances if necessary. It is designed to be run continuously in a loop as a background process on a local machine.

3. As the local IP address is static and does not change over time, we associated the Web-tier with it. Then, the images are uploaded using the upload button.

4. The required number of instances started running depending on the input number of images.

5. These instances can be on the EC2 dashboard, images were uploaded to the input S3 bucket, and the classification results were uploaded to the output S3 bucket via message passing through SQS.

6. Finally, the results were forwarded to the frontend webpage in key-value pair from SQS and S3 respectively in the format: `image.jpeg, image_label`.

Results

The Results section of the document showcases the functionality of the AWS-based image classification system. The user interface of the web application comprises a simple web page that allows users to upload any number of images for classification as shown in Figure 3 . The UI has a "Choose file" button that enables users to select the image files to be uploaded, and an "Upload" button to upload them to the AWS services.

Image Classification with AWS

Choose one or more images to upload to AWS:
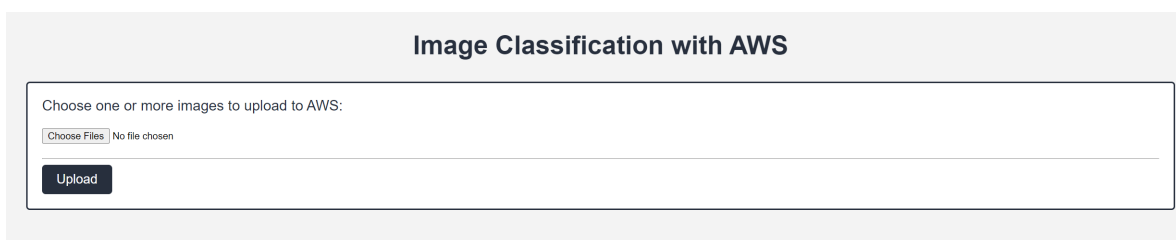
Choose Files No file chosen

Upload

Figure 3: User Interface.

Upon clicking the "Upload" button, the web application automatically creates EC2 instances based on the number of uploaded images. Figure 4 displays the number of instances running based on the number of images uploaded. This dynamic scaling of the App-tier instances ensures that the system can handle a large number of requests efficiently.

After the images have been processed, the system returns the classification results in the form of key-value pairs. The screenshot of Figure 5 displays the results beneath the "Upload" button on the same webpage. Each image is labeled with its corresponding class and displayed in the format: `test_0.JPEG, hair_spray`. This user-friendly interface enables users to quickly and easily classify multiple images with high accuracy, thanks to the use of a pre-trained ResNet model.
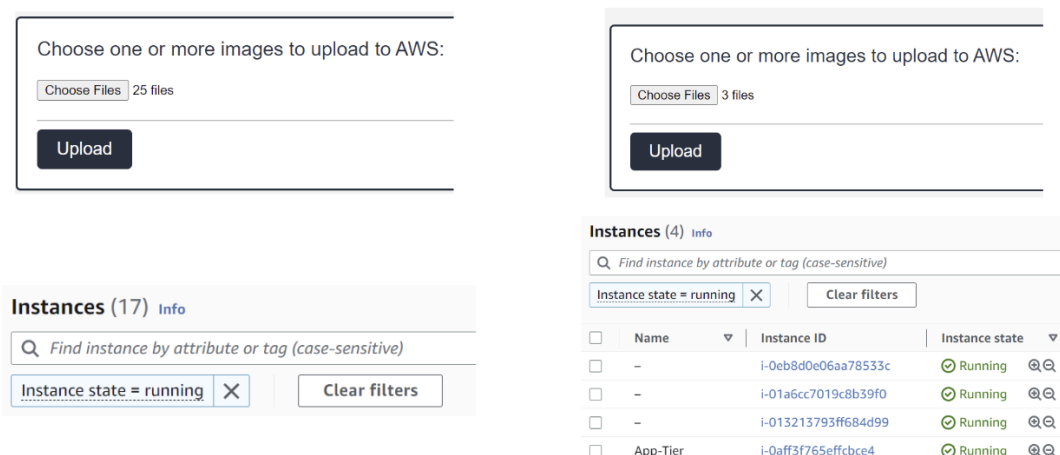
Figure 4: If the number of uploaded images is over 15, 17 instances will run, otherwise, the app-tier instance and the number of messages will run for less than 15 uploaded images.

Figure 5: Sample output displayed on the UI.

Evaluation based on Metrics:

In order to evaluate the performance of the system, we measured the response time and boot time for entire processing of the system and booting time of ec2 instance.

- Response time for uploading and classifying images was measured using the Python 'time' module.

- **Medium sized images (≈ 80 kb):**

  - 20 images took an average of 230.15 seconds to display results on the webpage.

  - 30 images took an average of 215.67 seconds to display results on the webpage.

  - Medium sized images were classified perfectly.

- **Lower sized images ($<$2 kb):**

  - 27 images took approximately 600 seconds to upload.

  - Lower sized images did not get accurate classification results.

- **Issues faced while uploading images:**

  - Memory errors were encountered while uploading 60 images.

  - Looping occurred after the 26th image while uploading 40 images.

  - Investigation needed to determine the maximum number of images that can be uploaded at a time - To determine root cause at varying levels.

- Average boot time for App-Tier's EC2 instance (excluding the delay of 30 seconds) was calculated from multiple test runs:

  - Average boot time was found to be 71.53 seconds based on the results from the several test runs.

Conclusion

The image classification application implemented in this project using AWS as an IaaS provider was successful in providing users with a platform to upload images for classification. The use of AWS services such as EC2, SQS, and S3, along with Python for the backend, allowed for the scaling of resources based on user demand, providing a cost-effective and scalable infrastructure for image classification applications.

Auto-scaling was implemented as a key feature of the infrastructure, which proved to be useful for applications with unpredictable usage patterns. The project demonstrated the importance of a scalable infrastructure in the field of image classification, which has various applications across industries such as medical diagnosis, agriculture, retail, security, environmental monitoring, and manufacturing.

The evaluation of the image classification system was based on the metrics of response time, boot time and accuracy. The response time for medium-sized images was found to be acceptable, with an average of 215.67 seconds for 30 images. However, the lower-sized images took a longer time to upload, with an average of approximately 600 seconds for 27 images. Additionally, lower-sized images did not get accurate classification results. Some issues were faced while uploading a large number of images, such as memory errors while uploading 60 images and looping after the 26th image while uploading 40 images. Investigation is needed to determine the maximum number of images that can be uploaded at a time. The average boot time for the App-Tier instance (excluding the delay of 30 seconds) was found to be 71.53 seconds based on the results from multiple test runs.

Overall, this project demonstrated the successful implementation of an image classification application using AWS, and provided valuable insights into the architecture and infrastructure required for such applications. The scalability and cost-effectiveness of the infrastructure make it a suitable choice for various image classification applications across different industries.

References

[1] Pinakin Ariwala. What is the working of image recognition and how is it used?, 2022. URL `https://marutitech.com/working-image-recognition/`.

[2] Jeff Barr. Choosing the right ec2 instance type for your application, 2013. URL `https://aws.amazon.com/blogs/aws/choosing-the-right-ec2-instance-type-for-your-application/`.

[3] Bernardo Botelho Antunes da Costa and Pedro Silveira Pisa. Cloud strategies for image recognition. In *2020 4th Conference on Cloud and Internet of Things (CIoT)*, pages 57–58, 2020. doi: 10.1109/CIoT50422.2020.9244200.

[4] Mahiban Lindsay N, Alla Eswara Rao, and Madaka Pavan Kalyan. Real-time object detection with tensorflow model using edge computing architecture. In *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, pages 01–04, 2022. doi: 10.1109/ICSSS54381.2022.9782169.

[5] Gregorius Rafael, Hendra Kusuma, and Tasripan. The utilization of cloud computing for facial expression recognition using amazon web services. In *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pages 366–370, 2020. doi: 10.1109/CENIM51130.2020.9297974.

[6] Vivek Sharma. Object detection and recognition using amazon rekognition with boto3. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 727–732, 2022. doi: 10.1109/ICOEI53556.2022.9776884.

[7] R. Suguna, M. Shyamala Devi, Akash Kushwaha, and Puja Gupta. An efficient real time product recommendation using facial sentiment analysis. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–6, 2019. doi: 10.1109/ICECCT.2019.8869300.