COMP4651 Project (Group 20)

LOHANAKAKUL Sasatorn (20805810), NG Josiah Jian Wei (20817875), YAP William Khin Hoong (20838324), FAN Tsz Ho (20852552)
Github Code: https://github.com/Hardy0611/Clould_Project

Project Description

In the realm of image processing—widely used in applications such as facial recognition—large and complex AI models often require cloud hosting. When a user submits a request, the model processes the data in the cloud and returns the results. However, this approach raises concerns about the privacy of uploaded images and the costs associated with cloud computing. Our project aims to develop a secure and cost-effective platform that addresses these issues, ensuring that sensitive information is processed locally while leveraging the cloud for more demanding tasks. This hybrid model not only improves user experience through seamless integration but also adapts to evolving demands, creating a robust framework for AI deployment across diverse environments.

To effectively harness the growing capabilities of AI while minimizing costs, a strategic approach involves dynamically splitting workloads between personal devices and cloud computing. Many AI models are also too complex for less advanced devices, necessitating the partitioning of these models. This allows lightweight tasks to run locally while offloading more resource-intensive computations to the cloud. Hence, we can optimize resource allocation based on device capabilities and network conditions, enhancing performance and reducing cloud computing costs.

Application Description

The application operates as follows: the user uploads a photo to the local platform, which assesses the computational power of the local device. Based on the available resources, the application dynamically determines how many layers of processing should run locally and how many should be handled in the cloud. If the local device has higher computational capacity, more layers will be processed on-site. After the initial layers are executed locally, the output is sent to the cloud to complete the remaining processing. Since the cloud model accepts input in the form of encoded numbers, the original photo is never directly exposed, minimizing the risk of data leakage. Once the cloud completes the processing, the results are sent back to the application, which then displays the identified emotion from the photo.

Technologies Used

We implement our core technology using the following libraries. These libraries facilitate dynamic allocation, building user-friendly interface and model splitting. Based on the computational power of each user's machine, we intelligently distribute the workload between the local environment and cloud functions. This approach ensures that every client enjoys a similar experience while also maximizing cost efficiency.

- **Streamlit** [https://streamlit.io/]
  Streamlit offers a powerful and user-friendly framework for building data-driven web applications quickly and effortlessly. It allows developers to create interactive dashboards and applications using pure Python, making it accessible for data scientists and machine learning practitioners. With features like real-time updates and seamless integration with popular data libraries, Streamlit enables the rapid prototyping of data visualizations and machine learning models. Additionally, its compatibility with various data sources and styling options helps streamline the development process, ensuring that applications are both functional and visually appealing.

- **Tensorflow** [https://www.tensorflow.org/]
  TensorFlow is an open-source machine learning framework that provides a comprehensive ecosystem for building and deploying machine learning models. It offers a flexible architecture that allows developers to create complex neural networks and perform large-scale numerical computations efficiently. With features such as automatic differentiation, robust support for deep learning, and a wide range of pre-trained models, TensorFlow empowers researchers and practitioners to innovate in fields like computer vision, natural language processing, and reinforcement learning. Additionally, its integration with tools like TensorFlow Serving and TensorFlow Lite facilitates the deployment of models across various platforms, making it easier to bring machine learning solutions to production.

- **Psutil [https://psutil.readthedocs.io/]**

  psutil is a powerful cross-platform library for retrieving information on system utilization, including CPU, memory, disk, and network statistics. It provides an easy-to-use interface for monitoring system performance and resource usage, making it invaluable for developers and system administrators. With features such as process management, system uptime tracking, and the ability to access system information in real time, psutil enables efficient performance monitoring and troubleshooting. Its lightweight nature and compatibility with various operating systems allow for seamless integration into Python applications, helping users optimize system resource management effectively.

Deployment and Testing Technologies

- **API Gateway** [https://aws.amazon.com/api-gateway/]
  Amazon API Gateway is a fully managed service that makes it easy to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a gateway to your backend services, enabling developers to create RESTful APIs and WebSocket APIs that can handle thousands of concurrent requests.. In our project, we used Amazon API Gateway to expose an HTTP endpoint for our AWS Lambda function. This setup allowed the Streamlit application to send intermediate results (processed image data) from the local computation to the Lambda function hosted in the cloud.

- **AWS Lambda Function** [https://aws.amazon.com/pm/lambda/]

AWS Lambda is a serverless compute service that enables you to run code without provisioning or managing servers. With AWS Lambda, you can execute code in various programming languages, integrate with other AWS services, and only pay for the compute time you consume. In our project, AWS Lambda played a critical role in executing the cloud-based portion of our hybrid image processing model. After preprocessing the image locally and determining the partial model output, the Streamlit application sent the intermediate results to the Lambda function for further processing. The Lambda function completed the remaining layers of the machine learning model and returned the final predicted emotion back to the application.

- **Elastic Container Registry** [https://aws.amazon.com/ecr/]
  Amazon Elastic Container Registry (ECR) is a fully managed container registry service provided by AWS. It allows developers to store, manage, and deploy container images in a secure and scalable manner. With ECR, you can seamlessly integrate with AWS services, making it an ideal solution for deploying containerized applications on platforms like Amazon ECS, EKS, or even Lambda.

  In our project, the TensorFlow package and the associated machine learning model exceeded the size limits imposed by AWS Lambda's traditional deployment options. While Lambda Layers can increase the size limit, they introduced complexities and were not sufficient for our requirements. Elastic Container Registry was chosen as the most suitable solution because it allowed us to package the entire application, including TensorFlow, dependencies, and the model, into a single container image. This container image could then be deployed seamlessly as a Lambda function, bypassing size restrictions and ensuring smooth integration with our workflow. Using ECR also made it easier to manage and version our containerized application, simplifying future updates and scalability.

- **S3 Bucket** [https://aws.amazon.com/s3/]
  Amazon S3 (Simple Storage Service) is a highly scalable, secure, and durable cloud storage service provided by AWS. It is designed for storing and retrieving data of any size and format, making it ideal for a wide range of use cases, from backup storage to hosting large-scale datasets. S3 offers features such as object versioning, access control, and event-driven triggers, which make it particularly useful for applications that rely on file uploads and event-based processing.

  To test our hypothesis, we needed to replicate the standard AI image processing workflow, where a user uploads an image to an S3 bucket, triggering a Lambda function for processing. By using S3 as our storage solution, we were able to simulate this process and validate the interactions between the image upload mechanism, the Lambda trigger, and the subsequent processing pipeline. This setup allowed us to evaluate the performance of our prototype under realistic conditions and confirm its ability to optimize resource usage, enhance privacy, and streamline image processing workflows.

Evaluation

In most AI image processing workflows, the standard approach involves a user uploading an image to an S3 bucket, which then triggers a Lambda function to process the image and return the prediction as a result. However, our proposed prototype takes this a step further by addressing key inefficiencies in this traditional setup by processing data locally in your own device using dynamic splitting.. We evaluate the prototype across three key aspects: computing resources, storage cost, and security

- **Computing Resource**
  Through our project, the Lambda function execution time decreases as the number of layers processed in the cloud decreases. This dynamic distribution of processing reduces the Computational overhead in the cloud, as illustrated in the table below. However, there are some notable findings. Cloud processing is impacted by cold start and hot start conditions, which significantly affect execution time. Cold starts are longer due to initialization delays, while hot starts to benefit from container reuse. Additionally, the execution time for five layers is longer than the full model due to the increased data transmitted between local and cloud components.

Model Layers in Cloud	Execution Speed (s)
Full Model + Preprocessing (Cold Start)	18
Full Model + Preprocessing (Hot Start)	3.2
5 layers	3.7
4 layers	3
3 layers	2.5

- **Storage Cost**
  Traditional workflows rely on S3 buckets for temporary image storage, which incurs storage and retrieval costs. By processing the image locally and transmitting encoded numerical vectors to the cloud, our project eliminates the need for intermediate image storage. We also are able to reduce data transmission, as vectors, instead of full images, require less bandwidth, further optimizing cost efficiency

- **Security**
  One of the primary concerns in AI image processing is the privacy of user data, especially when sensitive images are transmitted over the internet. Through this hybrid model, we are able to preprocess the images locally, ensuring the raw images are never uploaded to the cloud. Not only that, through preprocessing, we convert raw images into encoded numerical vectors, making it not only challenging to reconstruct the original image, but also significantly lowering the risk of data breaches or misuse

Conclusion

As shown by our project, our project aims to reduce cloud computing resources by performing partial training locally using dynamic splitting based on the computing resource power of the device. By combining local and cloud processing, our prototype achieves measurable improvements in computing efficiency, cost reduction, and security. This hybrid model offers a scalable, efficient, and secure framework for AI image processing, setting the foundation for more privacy-aware and resource-efficient applications.

Future Work

As this project is currently a prototype, the machine learning model used is relatively basic. In the future, we plan to explore more advanced machine learning algorithms and implement larger, more sophisticated models to enhance the accuracy and performance of the application.

Additionally, an area worth considering is the implementation of federated learning. This approach would allow multiple devices to collaboratively train and machine learning while keeping data localized, further improving privacy and reducing the need for centralized data collection. Federated learning could be especially valuable in applications where user data security and compliance with privacy regulations are critical.

By advancing these aspects, the project could evolve into a more robust and scalable solution for real-world AI image processing challenges.