Automated Image Tagging System

Project Documentation

Team members

Ahmed Waheed: Data Engineer

Bedour Fouad: Data Engineer

Menna Osama: Model Development

Norhan El-Sayed: Model Development

Mina Farid: Deployment and Documentation

Project Overview

The objective was to create a machine learning application capable of performing multi-label classification for image tagging, leveraging a subset of the COCO dataset and AWS services to train and deploy the model.

Key Features

- **Dataset**: Subset of COCO dataset, optimized for multi-label classification with 80 classes and 100,000 training examples.
- **Data Processing**: Data integrity checks, preprocessing, and augmentation.
- **Model Architecture**: Transfer learning using ResNet50.
- **Training**: Conducted on AWS SageMaker using a SageMaker notebook.
- **Deployment**: Model deployed on AWS using SageMaker Inference Endpoints and API Gateway.
- **Monitoring and Debugging**: AWS CloudWatch Logs were utilized to debug and monitor both the SageMaker endpoint and API Gateway.
- **App Integration**: Simple Flutter application to demo the model via API calls.

Dataset

We selected a subset of the COCO dataset that is specifically optimized for multi-label classification. The dataset contains:

- **80 classes**: Various object categories relevant to the COCO dataset.
- **100,000 training examples**: Diverse images representing the classes. The dataset was uploaded and stored in AWS S3.

Data Integrity Checks

Before training the model, we conducted a thorough review of the dataset to ensure the following:

- **Consistency of labels**: We ensured that the labels are correct and represent the images accurately.
- **Completeness of data**: We ensured that the images had corresponding labels and no missing data.
- **Data distribution**: The dataset was checked for class imbalance to ensure diversity in training.

Data Preprocessing and Augmentation

Data preprocessing was done to standardize image sizes and enhance model performance. Techniques included:

- **Resizing images** to match the input size for the ResNet50 model.
- **Normalization** to ensure pixel values are between 0 and 1.
- **Data Augmentation**: Applied techniques like horizontal flipping, random cropping, and rotations to increase the diversity of the training set.

Model Development

We used **transfer learning** with a **ResNet50** pre-trained model. The rationale for selecting ResNet50 was its effectiveness in handling complex image classification tasks while allowing the model to generalize well for our multi-label classification problem.

Training

Training was carried out on **AWS SageMaker** using a SageMaker notebook instance. Key details:

- **Pre-trained model**: Transfer learning with ResNet50.
- **Validation**: The model's performance was validated using a separate test set.

During the training process, **AWS CloudWatch Logs** were used to monitor the model's performance, track potential errors, and review training outputs. The logs provided valuable debugging information that helped fine-tune the training pipeline. After training, the model was saved and stored in AWS S3 for future access.

Deployment

We deployed the trained model using **AWS SageMaker Serverless Inference Endpoints**. To facilitate interaction with the model, we integrated:

- **API Gateway**: Set up to create RESTful API endpoints for the model.
- **API Keys**: Generated to control access to the model and enable authentication for requests.

AWS CloudWatch Logs were crucial for debugging and monitoring both the **SageMaker Endpoint** and **API Gateway**.

Specifically, the logs helped us:

- Debug API errors and failures, such as request failures and endpoint timeouts.
- Monitor the **number of inference requests** sent to the API Gateway, ensuring smooth operation and tracking usage. This setup allows for easy scalability and access to the model for inference purposes.

Application Development

To demonstrate the use of the trained model, we built a simple **Flutter application**. The app:

- **Connects to the API**: Uses API calls to communicate with the deployed model on AWS.
- **Image input**: Allows users to upload images for tagging and classification.
- **Displays results**: The model returns the top predicted labels, which are displayed on the app interface

---

Conclusion

This project successfully developed a machine learning solution for **multi-label image classification** using a subset of the COCO dataset, **ResNet50** for transfer learning, and **AWS SageMaker** for training and deployment. We ensured data integrity, applied preprocessing and augmentation, and deployed the model with **SageMaker Inference Endpoints** and **API Gateway**.

A **Flutter app** was created to demonstrate the model's real-time tagging capabilities. Throughout the process, **AWS CloudWatch Logs** were used for debugging and monitoring API requests, ensuring smooth model access and performance. The project highlights the scalability and effectiveness of integrating cloud-based infrastructure for machine learning solutions.

Additional Resources

Demo App

You can try the demo app online using the following links:
Demo App (APK)

Demo App (Windows)

GitHub Repository

The full source code and additional information can be found in the GitHub repository:
GitHub Repository