Automated Image Tagging System

Project Documentation

Team members

Ahmed Waheed: Data Engineer

Bedour Fouad: Data Engineer

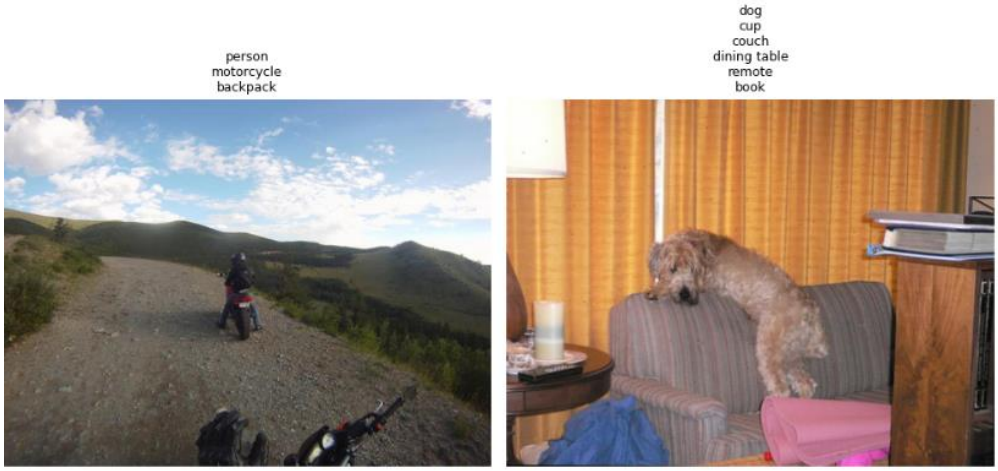
Menna Osama: Model Development

Norhan El-Sayed: Model Development

Mina Farid: Deployment and Documentation

Project Overview

The objective of this project is to create a machine learning application capable of performing multi-label classification for image tagging in order to assign tags to images, 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.

Problem Formulation

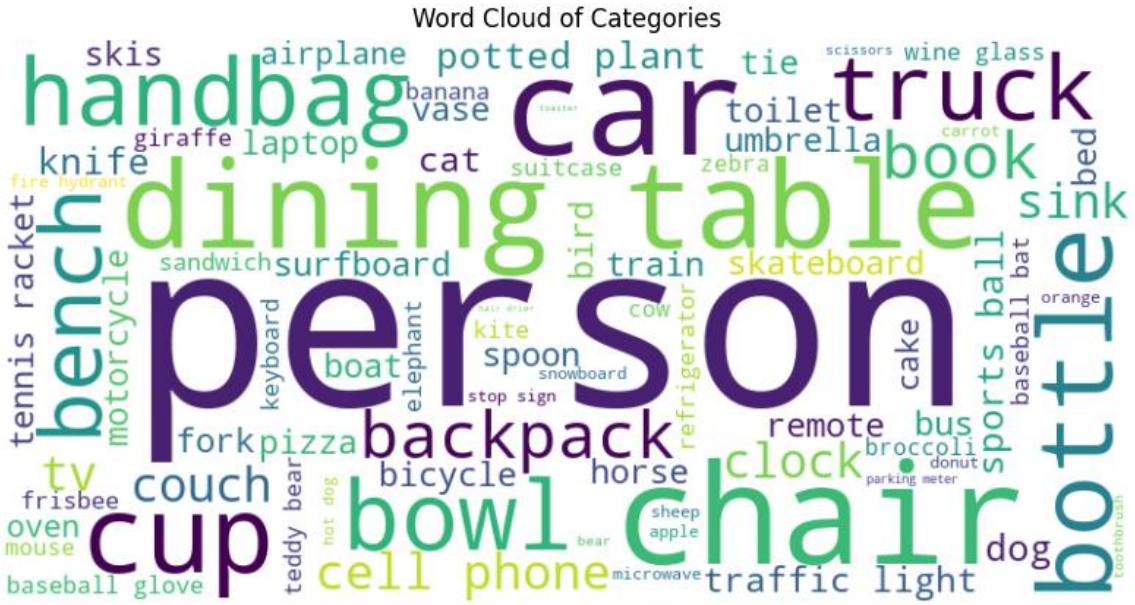
With the rapid increase in visual data, manual image tagging has become inefficient. Many images contain multiple objects, necessitating a multi-label classification approach. This project aims to automate image tagging, offering businesses and developers the following benefits:

- **Enhanced content organization**: Automatically tag images with relevant categories, simplifying content search and retrieval.
- **Improved user experience**: Enable personalized recommendations or content filtering based on accurately tagged data.
- **Reduced manual workload**: Free up human resources from labor-intensive image labeling tasks, reducing both time and costs.

Dataset

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

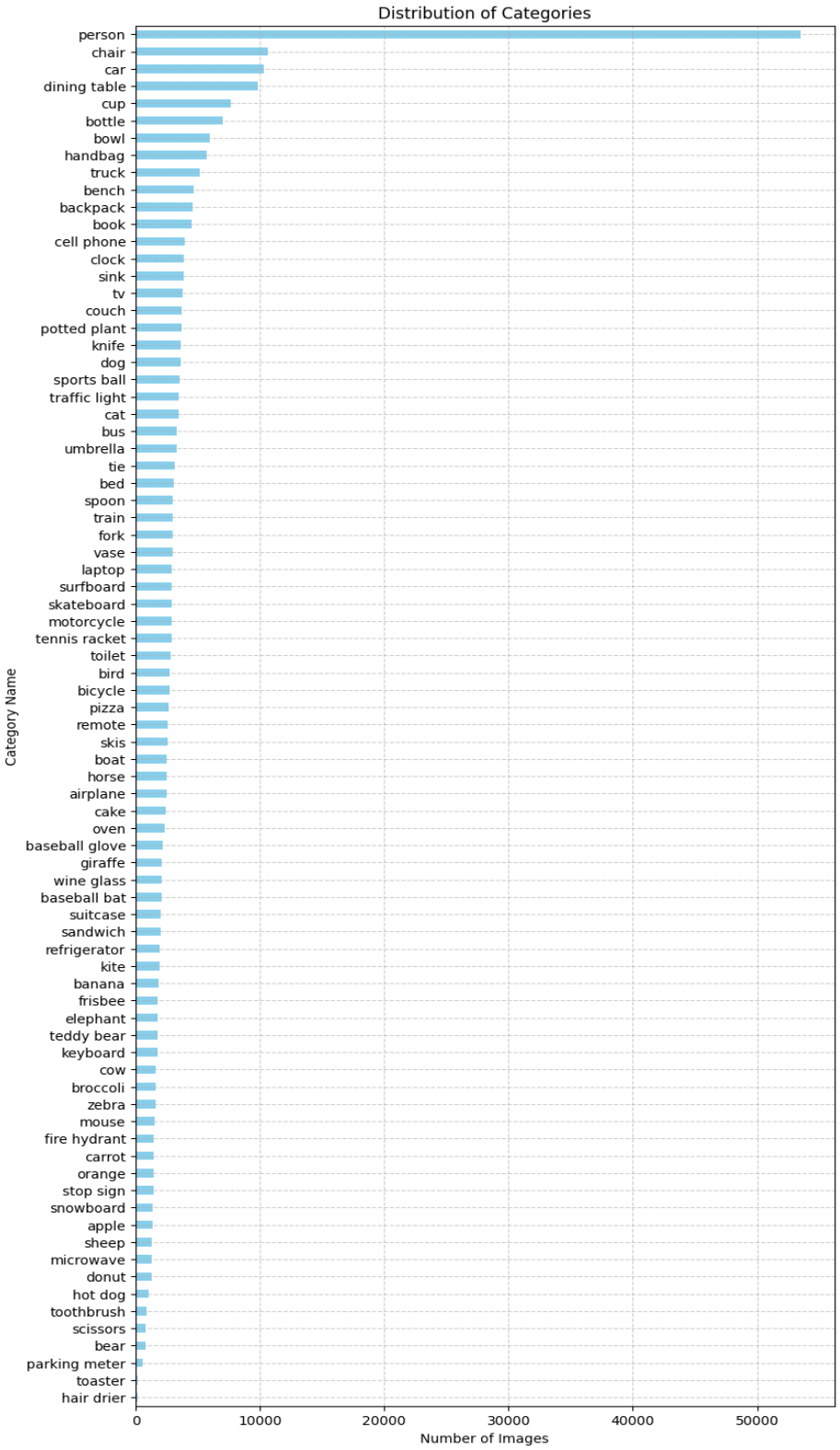
- **80 classes**: Various object categories relevant to the COCO dataset.
- **100,000 training examples**: Diverse images representing the classes. The dataset was uploaded and stored in AWS S3.



Data Consistency Analysis

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 by comparing a random subset of images and labels.
- **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. Minor class imbalances were handled through oversampling and targeted data augmentation.


Distribution of Categories
person
chair
car
dining table
cup
bottle
bowl
handbag
truck
bench
backpack
book
cell phone
clock
sink
tv
couch
potted plant
knife
dog
sports ball
traffic light
cat
bus
umbrella
tie
bed
spoon
train
fork
vase
laptop
surfboard
skateboard
motorcycle
tennis racket
toilet
bird
bicycle
pizza
remote
skis
boat
horse
airplane
cake
oven
baseball glove
giraffe
wine glass
baseball bat
suitcase
sandwich
refrigerator
kite
banana
frisbee
elephant
teddy bear
keyboard
cow
broccoli
zebra
mouse
fire hydrant
carrot
orange
stop sign
snowboard
apple
sheep
microwave
donut
hot dog
toothbrush
scissors
bear
parking meter
toaster
hair drier
Category Name
0
10000
20000
30000
40000
50000
Number of Images

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. Augmentation was particularly important for underrepresented classes.

Model Development

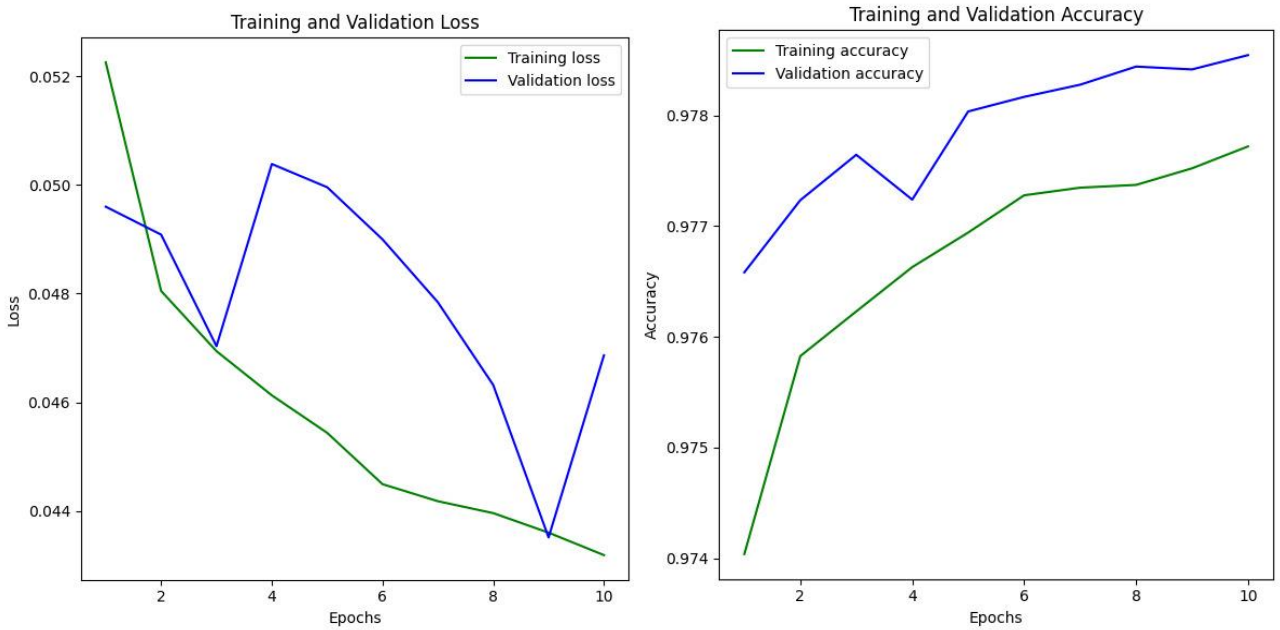
We used transfer learning with a ResNet50 pre-trained model. The rationale for selecting ResNet50 was its effectiveness in handling complex image classification tasks while allowing the model to generalize well for our multi-label classification problem.

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.

After training, the model was saved and stored in AWS S3 for future access.



Evaluation

We evaluated the model's performance and the final model performance per label amounted to a testing set accuracy of: 97.85%

We also managed to increase the model’s recall by 10% by increasing the model’s complexity, introducing focal loss to the loss function and adding weighted sampling.

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. 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/Website Development

To demonstrate the use of the trained model, we built both a simple mobile application and a website that perform the following:

- **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/website interface.

The application was made using Flutter and deployed as an APK for Android and also an executable for windows,

The website was made using simple HTML, CSS, Javascript and hosted on an EC2 instance.

Problems Faced

Throughout the project, we encountered several challenges:

- **Class imbalance**: Certain classes in the dataset were underrepresented, making it difficult for the model to generalize well. We addressed this through targeted data augmentation, class weighted sampling and focal loss.
- **Data leakage**: We discovered that augmenting both the training and validation sets together led to data leakage, which affected test accuracy. After modifying the augmentation process, the model performed better on unseen data.
- **Endpoint performance**: Initial endpoint timeouts and bugs were observed. Debugging the model's inference parameters and leveraging AWS CloudWatch for debugging helped resolve these issues.

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 using the following links:
Demo App (APK)

Demo App (Windows)

Demo Website

Demo Website

GitHub Repository

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

Dataset Link

Link