Object Detection on Google's Open Images Dataset
A Distributed Computing Approach

Sneha Agrawal

Deepak Swaminathan

Project Overview

Background

This research project focused on implementing an advanced object detection pipeline using a subset of the Google Open Images Dataset. The primary objective was to develop and evaluate object detection models using distributed computing technologies, specifically exploring the performance of Faster R-CNN and Single Shot MultiBox Detector (SSD) object detection models.

Objectives

- Process and filter the Open Images Dataset
- Train and evaluate object detection models
- Compare model performance using standard metrics

Methodology

Data Preparation

Dataset Characteristics

The project utilized the Google Open Images Dataset V7, which contains approximately 1 million images with detailed annotations for various objects, including bounding boxes.

The selective images from the dataset were downloaded using a custom Python script. This script filtered the dataset based on specific requirements and saved the chosen images into a Google Cloud Storage Bucket for further processing.

To prepare the dataset for training, a **custom PySpark script** was developed. This script handled the following steps:

1. **Reading Metadata**:
   - A CSV file containing image names and their associated bounding box annotations was loaded.

2. **Batch Processing of Images**:
    - Images were read in batches from the Google Cloud Storage Bucket.
    - Each image was resized to a standard dimension of (299, 299).
3. **Resizing Bounding Boxes**:
    - Bounding box coordinates in the annotations were proportionally resize to match the new image dimensions.
4. **Storing Resized Data**:
    - The resized images and their updated bounding box annotations were stored back into the Google Cloud Storage Bucket in **parquet format**.
    - Parquet was chosen for its efficiency in handling structured data and compatibility with distributed processing frameworks.

This preprocessing pipeline ensures the data is ready for machine learning workflows, optimizing both storage and retrieval performance for training.

Model Architecture

1. **Faster R-CNN**
    - Region-based Convolutional Neural Network
    - Known for high accuracy in object detection
    - Two-stage detection approach
2. **Single Shot MultiBox Detector (SSD)**
    - Single-stage object detection approach
    - Computationally efficient
    - Simultaneous object localization and classification

Training Infrastructure

- Objective: Leverage parallel processing for faster model training
- Intended Architecture:
    - Multiple GPUs
    - Distributed Data Parallel (DDP) training methodology

Technical Challenges

Computational Limitations

1. **GPU Resource Constraints**
    - Insufficient GPUs force the model to process more data sequentially rather than in parallel, significantly increasing training time.
    - Distributed training frameworks like PyTorch's torch.distributed or TensorFlow's tf.distribute thrive with multiple GPUs. Fewer GPUs lead to bottlenecks in data processing and reduced scalability of your model's architecture
    - Scaled down to training model on CPU, causing significantly longer training duration.

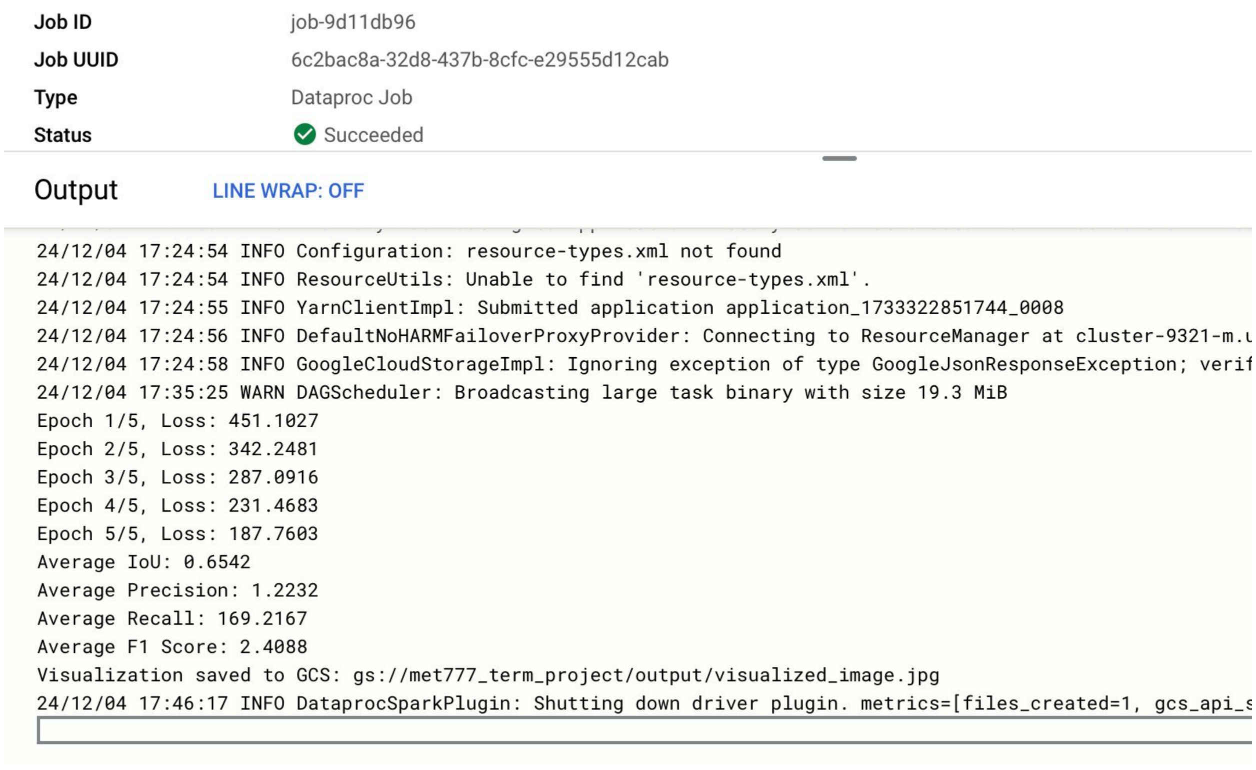
2. **Training Infrastructure Challenges**
   - Attempted Kaggle GPU environment utilization
   - Distributed Data Parallel (DDP) implementation restricted by notebook limitations
   - Extensive training times on CPU infrastructure

Performance Evaluation

Metrics

- Intersection over Union (IoU)
- Precision
- Recall
- F1-Score

Metrics of SSD

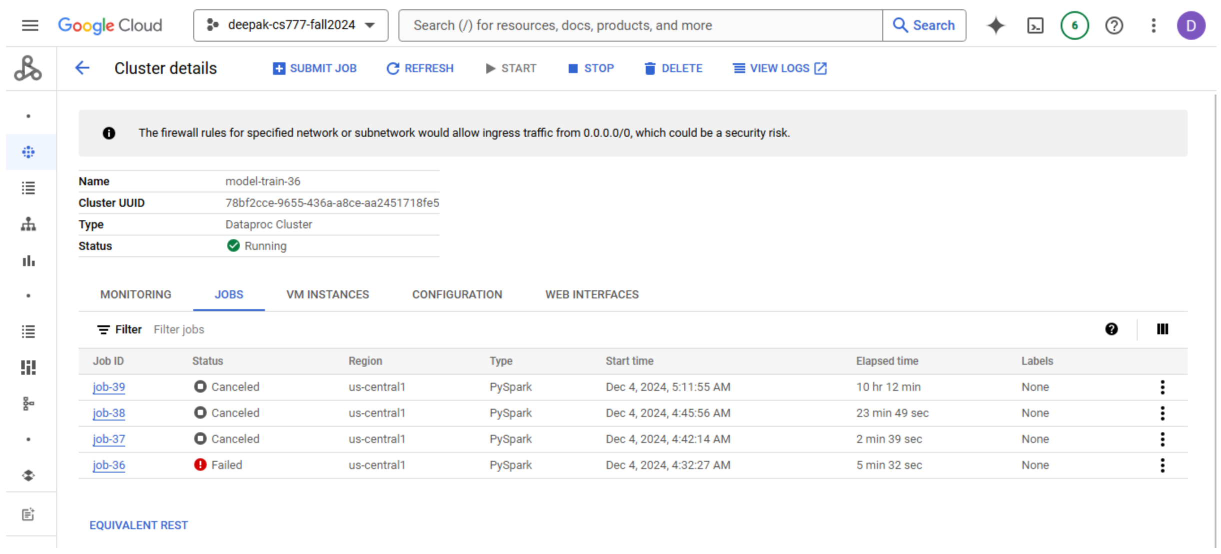


Job ID	job-9d11db96
Job UUID	6c2bac8a-32d8-437b-8cfc-e29555d12cab
Type	Dataproc Job
Status	✓ Succeeded

Output LINE WRAP: OFF

```
24/12/04 17:24:54 INFO Configuration: resource-types.xml not found
24/12/04 17:24:54 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/12/04 17:24:55 INFO YarnClientImpl: Submitted application application_1733322851744_0008
24/12/04 17:24:56 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-9321-m.u
24/12/04 17:24:58 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verif
24/12/04 17:35:25 WARN DAGScheduler: Broadcasting large task binary with size 19.3 MiB
Epoch 1/5, Loss: 451.1027
Epoch 2/5, Loss: 342.2481
Epoch 3/5, Loss: 287.0916
Epoch 4/5, Loss: 231.4683
Epoch 5/5, Loss: 187.7603
Average IoU: 0.6542
Average Precision: 1.2232
Average Recall: 169.2167
Average F1 Score: 2.4088
Visualization saved to GCS: gs://met777_term_project/output/visualized_image.jpg
24/12/04 17:46:17 INFO DataprocSparkPlugin: Shutting down driver plugin. metrics=[files_created=1, gcs_api_s
```

Due to computational constraints Faster RCNN took very long to complete 1 epoch.



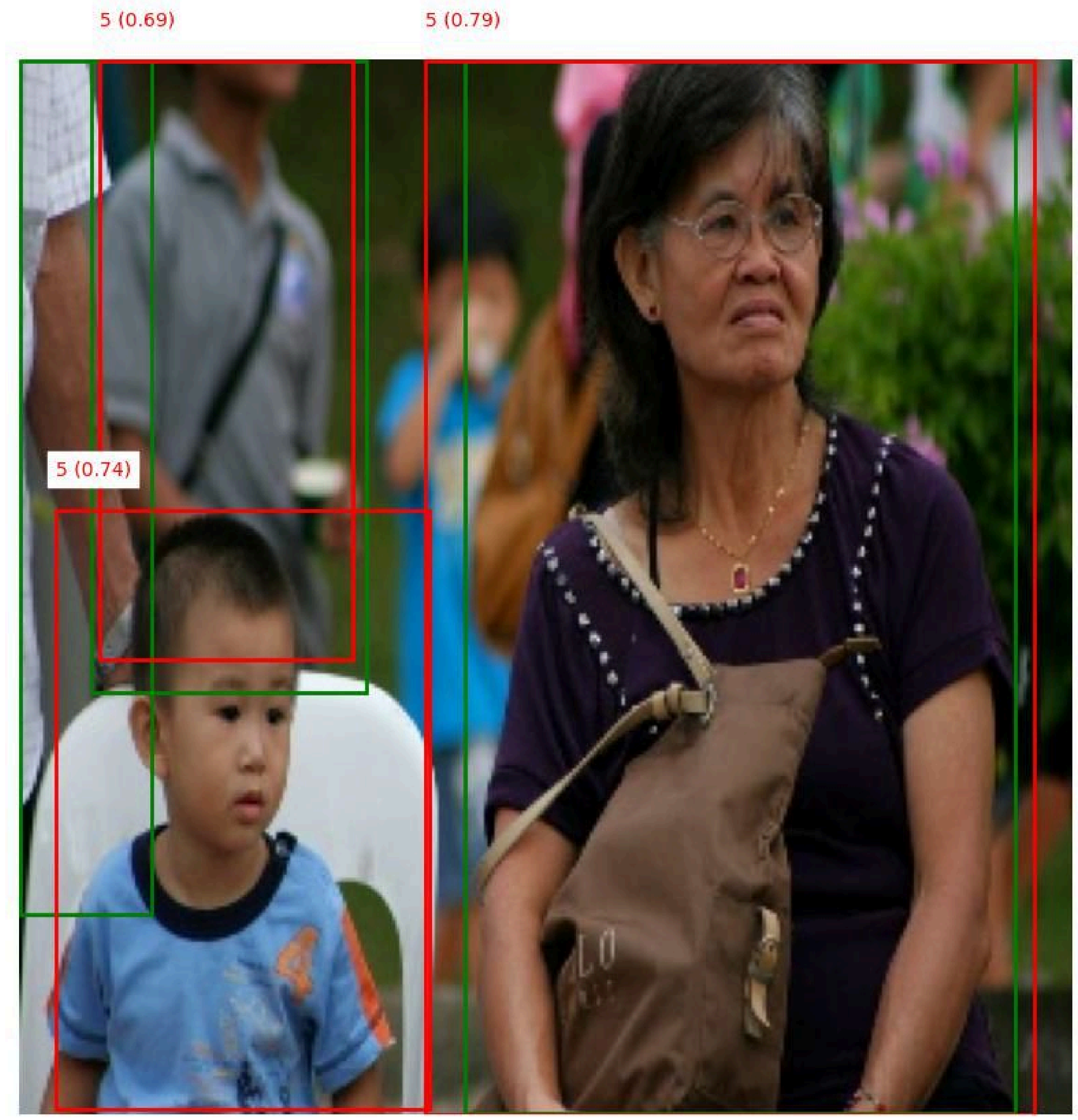
The firewall rules for specified network or subnetwork would allow ingress traffic from 0.0.0.0/0, which could be a security risk.

Name	model-train-36
Cluster UUID	78bf2cce-9655-436a-a8ce-aa2451718fe5
Type	Dataproc Cluster
Status	Running

Job ID	Status	Region	Type	Start time	Elapsed time	Labels
job-39	Canceled	us-central1	PySpark	Dec 4, 2024, 5:11:55 AM	10 hr 12 min	None
job-38	Canceled	us-central1	PySpark	Dec 4, 2024, 4:45:56 AM	23 min 49 sec	None
job-37	Canceled	us-central1	PySpark	Dec 4, 2024, 4:42:14 AM	2 min 39 sec	None
job-36	Failed	us-central1	PySpark	Dec 4, 2024, 4:32:27 AM	5 min 32 sec	None

Visualization

Ground Truth VS Prediction - SSD model



Results and Discussion

Model Performance Highlights

- SSD model showed faster inference times
- Faster R-CNN requires GPU resources for faster computation.
- Computational constraints impacted comprehensive evaluation

Key Insights

- Distributed computing enables parallel processing of image batches, reduced training time and the ability to handle large, complex datasets.
- GPU resources are essential for efficient deep learning workflows, computation CPU resources becomes prohibitively slow
- Key preprocessing techniques:
  - Image resizing
  - Bounding box normalization

Conclusion

The project successfully demonstrated a scalable object detection pipeline, highlighting both the potential and challenges of large-scale machine learning projects. The Single Shot MultiBox Detector (SSD) model emerged as particularly promising, demonstrating significant potential for real-time object detection applications. Unlike two-stage detectors like Faster R-CNN, SSD's single-stage architecture enables rapid inference speeds, making it ideal for time-sensitive scenarios such as autonomous driving, surveillance systems, and mobile robotics.

Future Recommendations

1. Secure dedicated GPU resources.
2. Develop optimized distributed training scripts.
3. Implement more sophisticated model architectures.

Github Repository :
https://github.com/s23deepak/CS777-TermProject