CloudyPoints

A Monocular Depth Estimation Tool

Amar Hamouma
Wim Looijmans

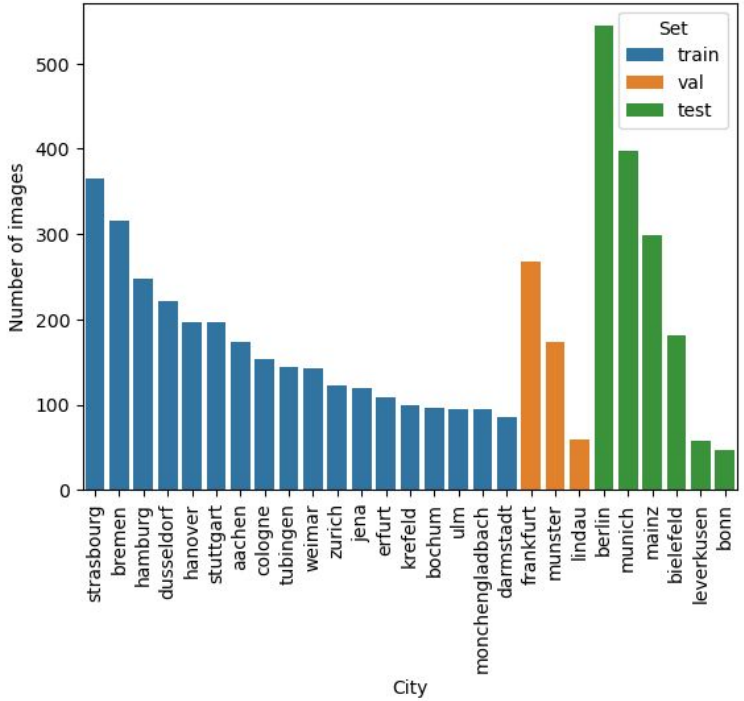
Project Description

- Monocular Depth Estimation (MDE)
  - Estimates depth in RGB image
- Outdoor Scenes
- Cityscapes Dataset
- Applications
  - Autonomous systems
  - Video surveillance
  - Augmented and virtual reality

https://www.cityscapes-dataset.com/

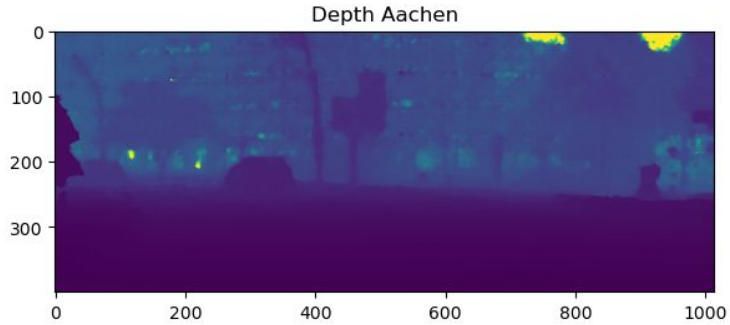
Cityscapes Dataset

- Urban street scenes
  - German cities
- Stereo camera
  - Left and right image
  - Disparity
- 5000 images
  - Training set: 2975
  - Validation set: 500
  - Test set: 1525



Data Preparation

- Depth map
  - Crop
  - Inpainting
  - Blur to decrease noise
  - Clip at max 500
- Resize to height 400
  - Less storage
  - Faster model training
- Stored in GCS
  - cp-bucket-1



Models

Two models tested

- intel-isl/MiDaS/DPT_Hybrid (122M)
- inel-isl/MiDaS_small (22.3M)

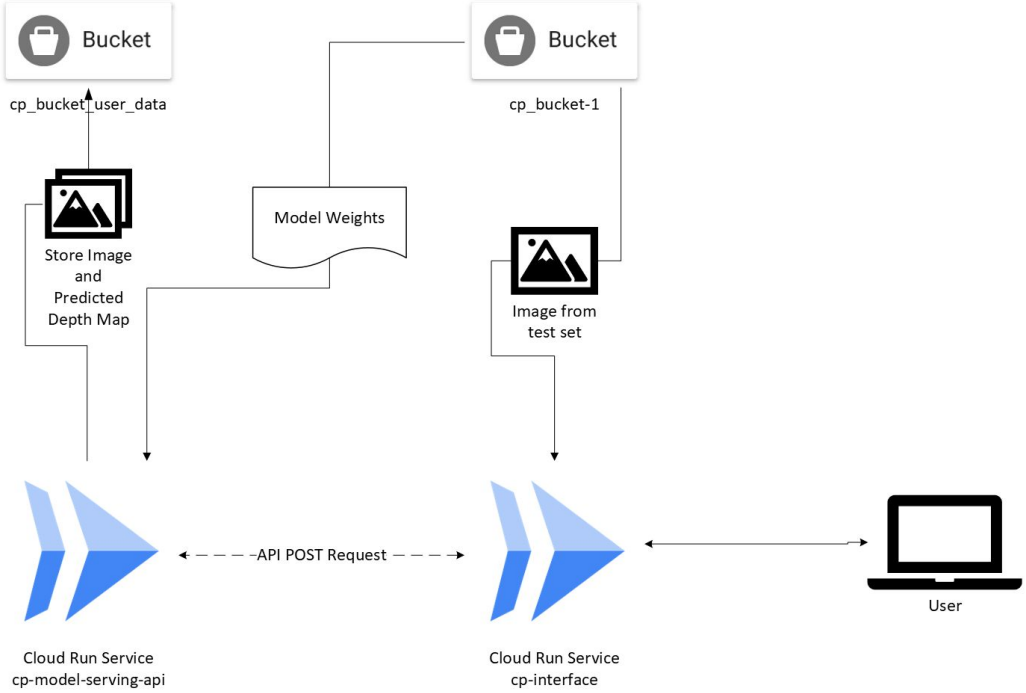
Metrics: (Trained for 15 epochs)

1. Structural Similarity Index Measure (SSIM): (0.90 – 0.88)
2. L1Loss: (0.027 – 0.032)

Trained in PyTorch Lightning

Model Serving API

- Model weights stored in GCS
  - Loaded through URL
- Flask App
  - Loads model
  - /predict
    - POST request
    - Image -> GCS
    - Resize
    - Inference -> depth map
      - -> GCS
      - -> response



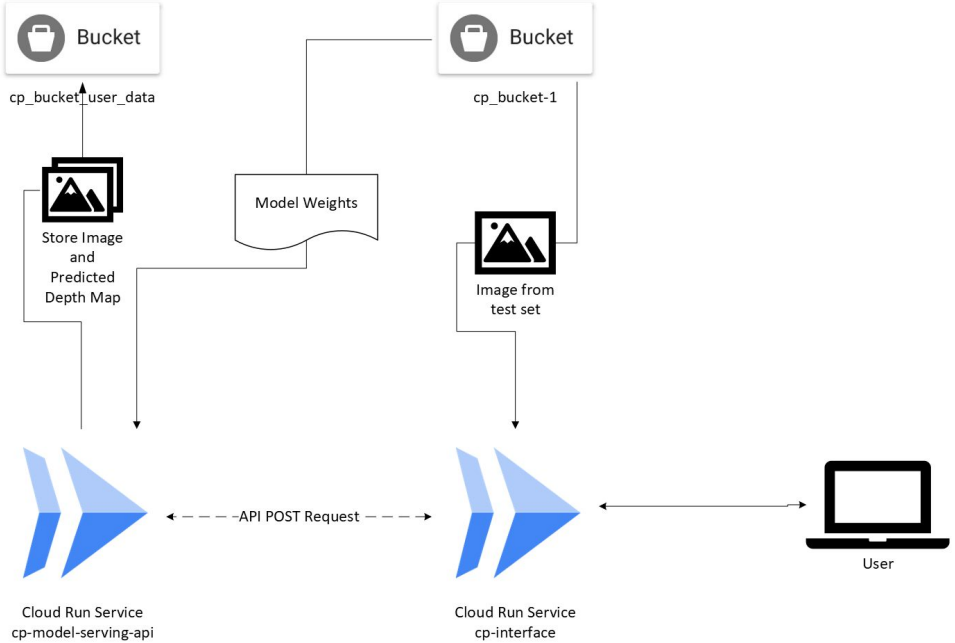
Model Serving API - Deployment

- Google Cloud Run
- Memory: 8 Gi
- CPU: 2
- Mount GCS bucket *cp_bucket_user_data*
  - Store original image and predicted depth map

```
gcloud run deploy ${{ inputs.model-serving-api-name }} \
--region=europe-west1 \
--source=$(pwd) \
--allow-unauthenticated \
--memory=8Gi \
--cpu=2 \
--add-volume name=cp_volume_user_data,type=cloud-
storage,bucket=cp_bucket_user_data \
--add-volume-mount volume=cp_volume_user_data,mount-
path=/mnt/user_data_storage \
--build-service-account "SERVICE ACCOUNT" \
--quiet
```

Streamlit Interface

- Separate Cloud Run Service
- Home page
- Depth Estimation
  - Upload Image
  - Predicted Depth Image
- Select Image from Test Set
  - Select City
  - Select Image with slider
  - Shows Image and Predicted Depth Map



Streamlit Interface - Deployment

- Google Cloud Run
- Base image python:3.9-slim
- Environment variable MODEL_SERVING_BASE_URL

```
gcloud run deploy ${{ inputs.interface-name }} \
--region=europe-west1 \
--source=$(pwd)/interface \
--platform managed \
--allow-unauthenticated \
--memory=512Mi \
--cpu=1 \
--update-env-vars=CP_BASE_URL="${{ env.MODEL_SERVING_BASE_URL }}" \
--build-service-account "SERVICE ACCOUNT" \
--quiet
```

CICD

- Github Actions
- Pipeline defined in:
  - reusable-CI.yml
  - reusable-CD.yml
    - Requires inputs -> service names
- Triggers
  - Pull request and Push
    - CICD-develop.yml
    - CICD-main.yml

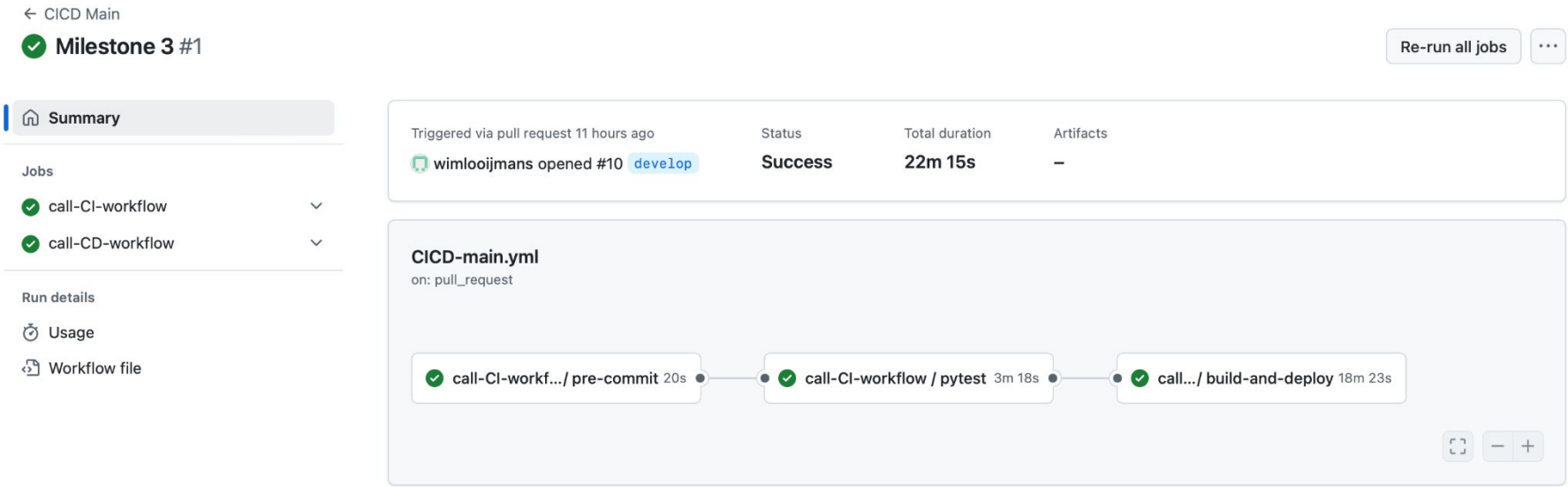
CICD - Continuous Integration

- reusable-CI.yml
- Pre-commit
  - Pre commit hooks
    - check-yaml
    - end-of-file-fixer
    - trailing-whitespace
  - Ruff linter
  - Black formatter
- Pytest
  - Testing Flask app
  - Testing Streamlit helper functions

CICD - Continuous Deployment

- reusable-CD.yml
- Requires inputs:
  - model-serving-api-name
  - interface-name
- Environment variable MODEL_SERVING_BASE_URL
- Workflow
  - Checkout code
  - Google authentication
  - Set up the Google Cloud SDK
  - Deploy Model Serving API to Cloud
  - Deploy Interface

CICD


CICD Main
Milestone 3 #1
Re-run all jobs
Summary
Jobs
call-CI-workflow
call-CD-workflow
Run details
Usage
Workflow file
Triggered via pull request 11 hours ago
wimlooijmans opened #10
develop
Status
Success
Total duration
22m 15s
Artifacts
–
CICD-main.yml
on: pull_request
call-CI-workf.../ pre-commit 20s
call-CI-workflow / pytest 3m 18s
call.../ build-and-deploy 18m 23s

Improvements

- Use GitHub Secrets
  - Instead of service accounts etc. in .yml files
- Implement storage of images on interface side
  - instead of in Model Serving API
- Place code and deployment files of Model Serving API in separate folder
  - instead of partially in src/ and root folder of project

Demo

https://cp-interface-436098836644.europe-west1.run.app