Machine Learning Engineering Bootcamp
Capstone Project: Share Your Project with the World
Step 12

Student: Kenneth Fung
Date: 17 April 2025

OUTLINE

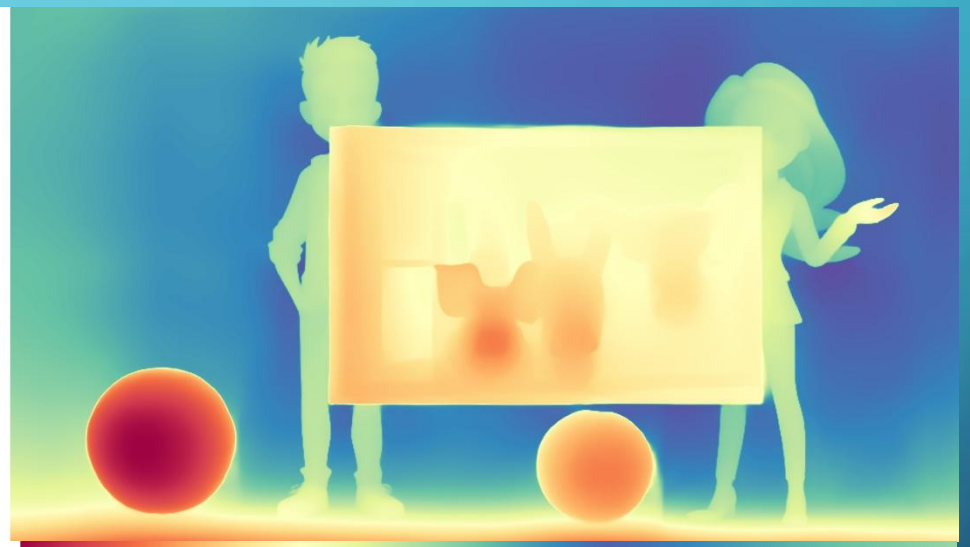
- Overview
  - Problem Statement
  - Existing Solutions
  - Targeted Solution for Exploration
  - Fine-Tuning Process
  - Deployment Steps
- Description & Link to Application Frontend
- Description & Links to GitHub Repositories: scripts, data, checkpoints

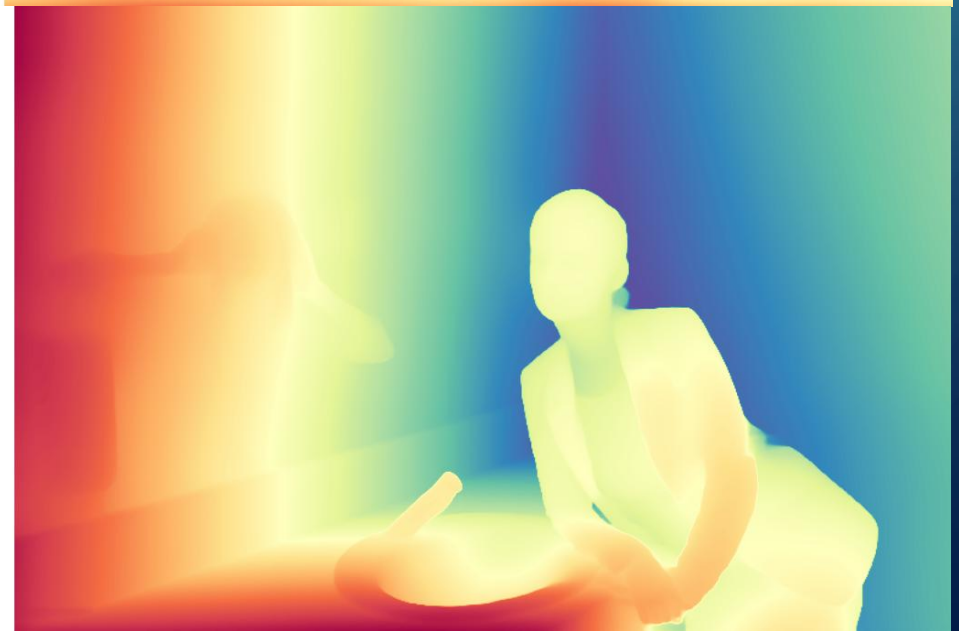
OVERVIEW – PROBLEM STATEMENT, EXISTING SOLUTIONS, TARGETED SOLUTION FOR EXPLORATION

- Problem Statement: Machine learning solutions for estimating depth in scenes are fooled by mirror surfaces and art on flat surfaces in indoor and outdoor environments.
- Existing Solutions:
  - Purely Monocular, Monocular Depth Estimation (MDE) models
    - Single RGB image input
    - Deep-learning based models include Depth Anything, MiDaS, Marigold
  - Monocular + Multi-Sensor Fusion
    - LiDAR, ToF, Radar, Ultrasonic
- Targeted Solution for Exploration:
  - Deep-learning based MDE model
  - Fine-tune Depth Anything v2

OVERVIEW - FIGURES ILLUSTRATING MDE DEFICIENCIES

Depth Anything v2 colormap depth maps (right) of images (left)



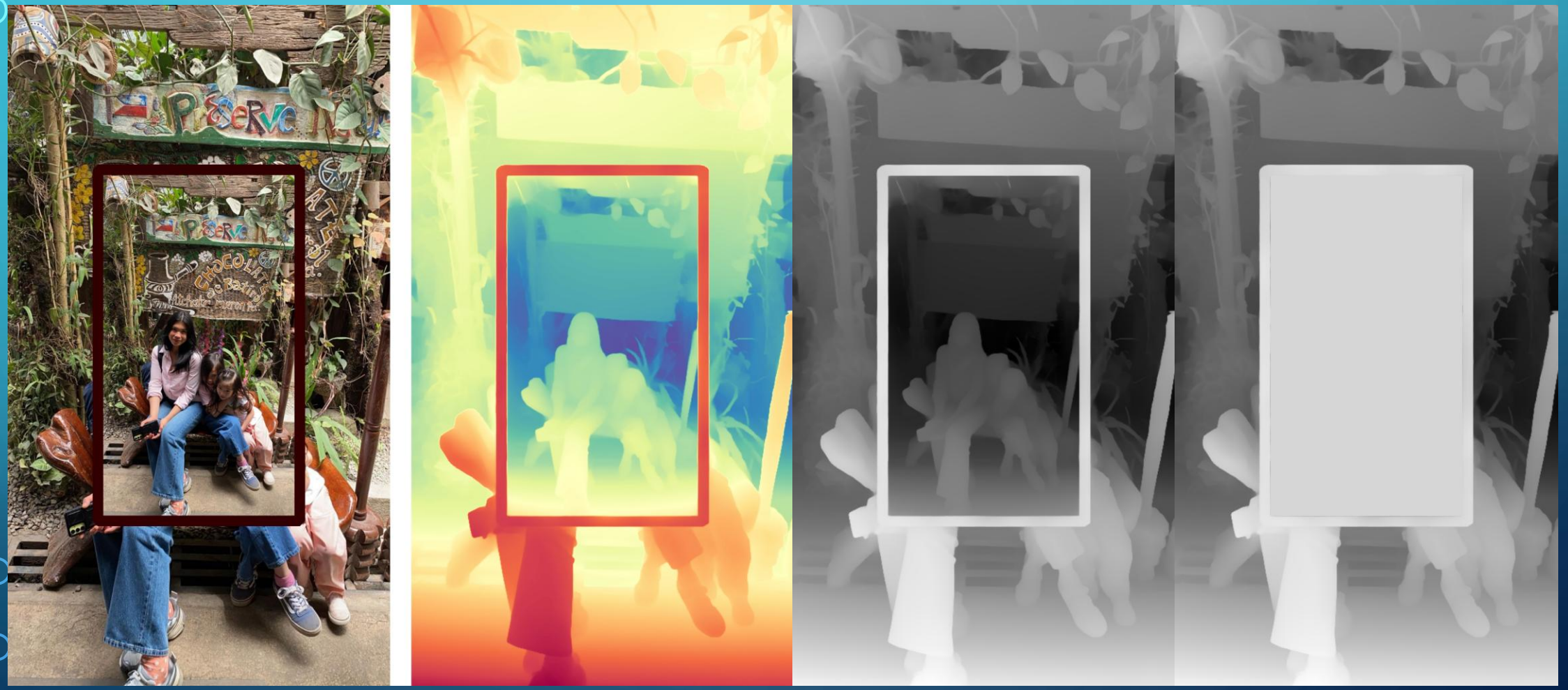


OVERVIEW – FINE-TUNING PROCESS

1. Explore what patterns to fine-tune an MDE model to learn
   - If a scene has a mixture of textures (e.g. photo-realistic and cartoonish), then only the surrounding texture should have depth and the inner embedded image should be flat
   - If an embedded image overlaps a background image such that the overlapped image does not appear in the embedded image, then the embedded image must be flat (e.g. mirror or 1-D art)
2. Augment datasets to fine-tune Depth Anything v2
   - Hypersim (indoor scenes) augmented with mirror images and 1-D art
   - vKITTI (outdoor scenes) augmented with 1-D art
3. Split datasets 80-20 for training and validation subsets
4. Fine-tune Depth Anything v2 using pre-trained "large" Hypersim and vKITTI models
   - Checkpoints are .pth files – integrate new weights into structure of pre-trained model dictionaries
5. Test data validation subsets with fine-tuned checkpoints and assess good and bad patterns
6. Iterate
   1. Refine fine-tuning parameters - epochs, batch-size, image size, learning rate, GPUs
   2. Pick layers to freeze – encoder and decoder
   3. Improve augmentation to achieve the desired patterns while not "unlearning" good patterns

OVERVIEW - SAMPLE AUGMENTED IMAGE

Depth Anything v2



augmented image | colormap depth map | grayscale depth map | augmented grayscale

OVERVIEW - FINE-TUNING PARAMETERS

- Common to Mirror and Art:
  - Image size: (518x518) pixels
  - Batch Size: 1
  - Epochs: 48
  - Initial learning rate: $5x10^{-5}$
  - GPU: 1
  - Maximum depth: indoor is 20m, outdoor is 80m
- Mirror images (indoor environments)
  - nnn total images : nnn training images, nnn validation images
- Indoor art (indoor environments)
  - nnn total images : nnn training images, nnn validation images
- Outdoor art (outdoor environments)
  - nnn total images : nnn training images, nnn validation images

OVERVIEW – DEPLOYMENT STEPS

1. Created a backend application as a Flask application
   - Backend would load MDE model, and process an image to produce a grayscale or colormap depth map
2. Created a frontend user-interface with Javascript
3. Gathered components – scripts, pre-trained and fine-tuned MDE models
4. Assembled deployment package with all components (project structure)
5. Used Docker to build a container for deployment package
6. Deployed through Google Cloud Run
   - 4 CPUs, 16 GB RAM
   - tagged Docker container, pushed container, run container

OVERVIEW - PROJECT STRUCTURE

HTTPS://GITHUB.COM/KENTHEMAN4AI/SB-CAPSTONE-PROJECT-MONOCULAR-DEPTH-ESTIMATION/TREE/88DEE7900D692456579EC9E49B5C8FC50DC92B3F/DEPLOYMENT

```
web_app/
├── Dockerfile
├── requirements.txt
├── DepthAnything_API.py                <-- Flask backend script
├── depth_anything_v2/                  <-- model class definitions – python scripts
├── templates/
│   └── DepthAnything_FT_GUI.html       <-- HTML GUI
├── checkpoints/
│   └── depth_anything_v2<***>.pth      <-- pre-trained and fine-tuned models
├── util/
│   └── <***>.py                        <-- utility scripts
```

SAMPLE RESULT OF FINE-TUNING (MIRROR)

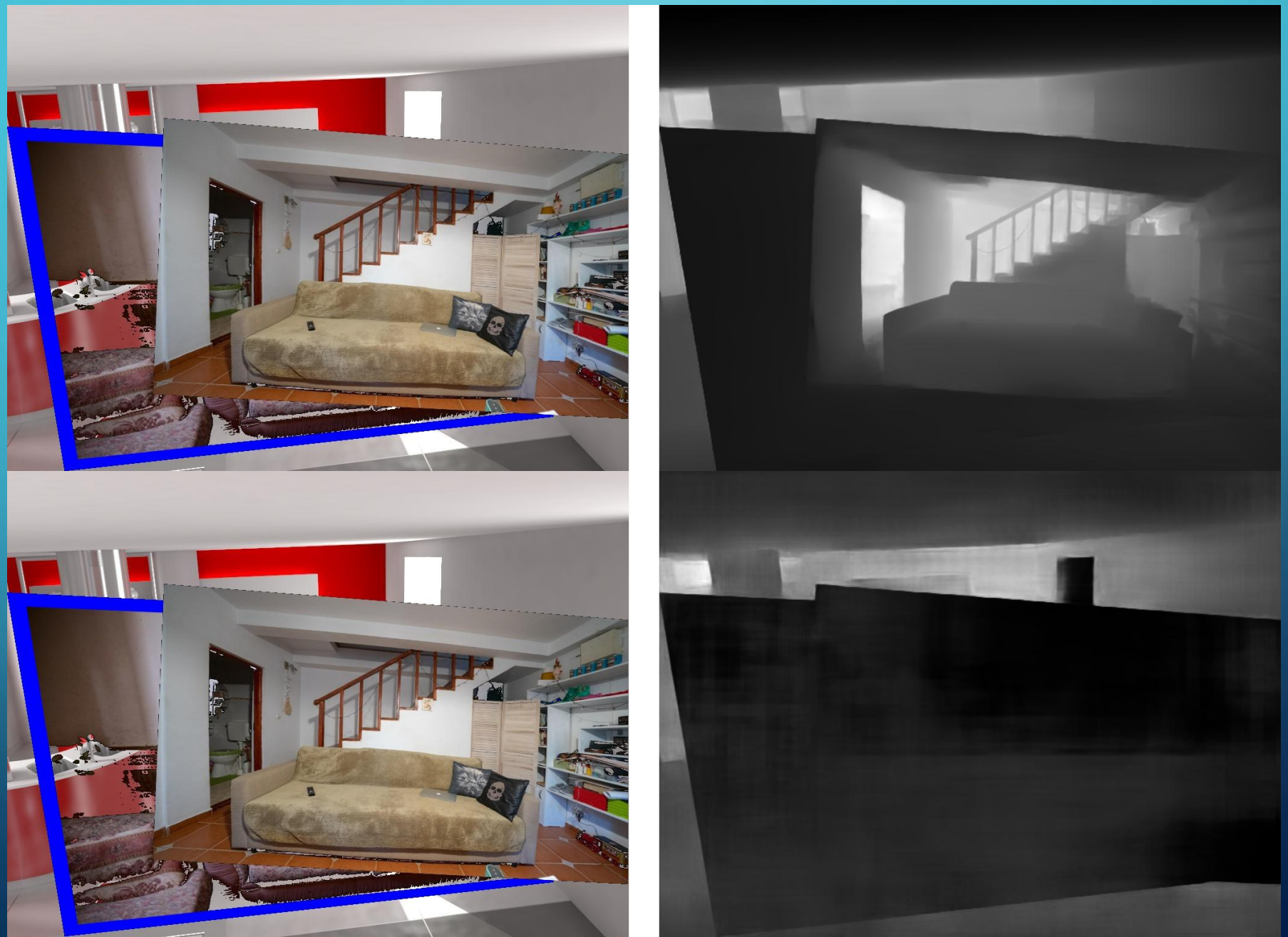
Pre-Trained

Fine-Tuned

SAMPLE RESULT OF FINE-TUNING (INDOOR 1-D ART)

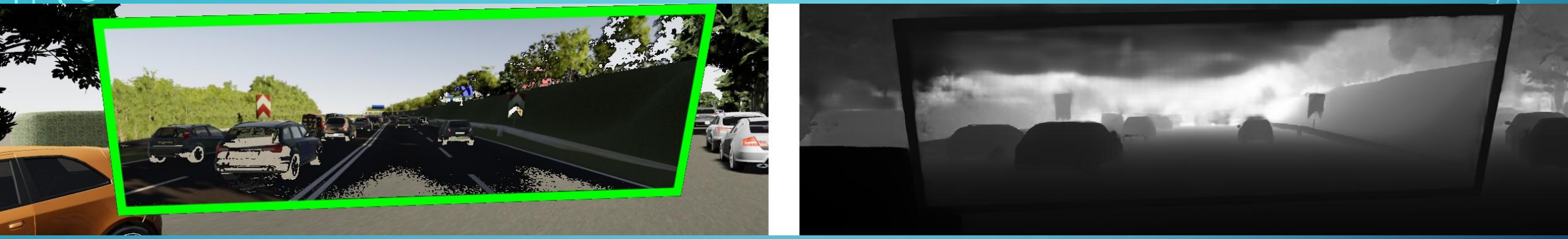
Pre-Trained

Fine-Tuned

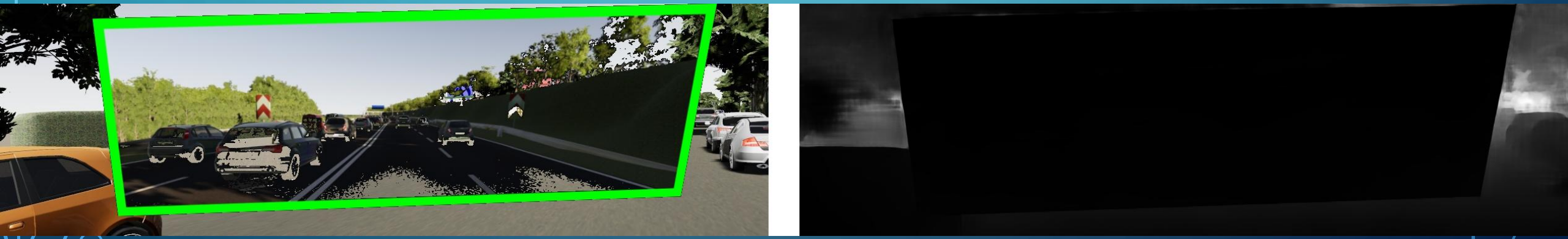


SAMPLE RESULT OF FINE-TUNING (OUTDOOR 1-D ART)

Pre-Trained

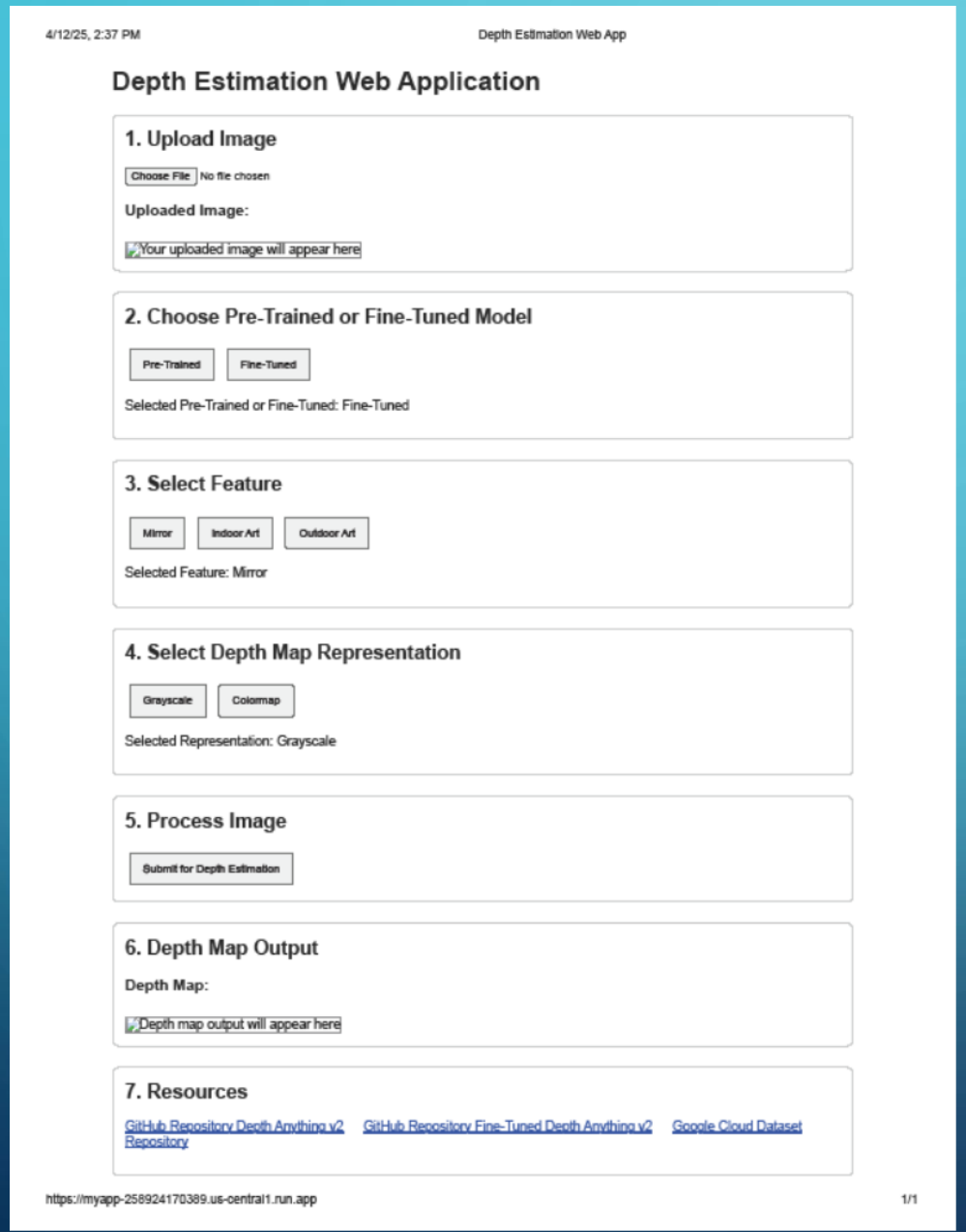


Fine-Tuned



DESCRIPTION & LINK TO APPLICATION FRONTEND

GOOGLE CLOUD RUN LINK: HTTPS://MYAPP-258924170389.US-CENTRAL1.RUN.APP/



1. Choose image to process
2. Select pre-trained or fine-tuned model
3. Select Mirror, Indoor Art, or Outdoor Art
4. Select grayscale or colormap for output
5. Click to process depth map
6. Image of depth map
7. Links to resources

DESCRIPTION & LINKS TO GITHUB REPOSITORIES: SCRIPTS, DATA, CHECKPOINTS

- Depth Anything v2 - https://depth-anything-v2.github.io/
  - Summary
  - Link to paper, GitHub repository with scripts, demo
- Fine-Tuning scripts - https://github.com/kentheman4AI/SB-Capstone-Project-Monocular-Depth-Estimation
  - Scripts used for augmenting images in dataset and for fine-tuning, deployment package
- Datasets with augmented images and ground truth depth maps - https://drive.google.com/drive/folders/13CHoZTuzNndxE6lebUojrV4QNfY8RgOE
  - Hypersim augmented with mirror reflections and 1-D random images as “art”
  - vKITTI augmented with 1-D random images as “art”

CITATIONS

@article{depth_anything_v2,

title={Depth Anything V2},

author={Yang, Lihe and Kang, Bingyi and Huang, Zilong and Zhao, Zhen and Xu, Xiaogang and Feng, Jiashi and Zhao, Hengshuang},

journal={arXiv:2406.09414},

year={2024}

}