Machine Learning Engineering & AI

Bootcamp Capstone

Step 8: Scale Your Prototype

17 April 2025

Overview:

I had chosen to fine-tune the Depth Anything v2 MDE model, for both indoor and outdoor scenes. For indoor scenes, the Hypersim[1] dataset was augmented with mirror images and 1-D art (Kaggle[2] and Mapillary[3]). For outdoor scenes, the Virtual KITTI 2[4] dataset was augmented with 1-D art. Here are specifics regarding the data augmentation:

- Indoor scenes – Hypersim
    - Mirrors (16,475 images) - Mirror reflections of random portions from the left or right half of the original background image were pasted onto the other half of the background image. Borders were randomly placed around the mirrors which were randomly rotated to simulate perspectives and thus appear as quadrilaterals.
    - Indoor 1-D Art (13,209 images) - Random images were pasted onto random locations of the background image. Borders were randomly placed around the 1-D art which were randomly rotated to simulate perspectives and thus appear as quadrilaterals.
- Outdoor scenes – Virtual KITTI (37,668 images)
    - Outdoor 1-D Art – Random images were pasted onto random locations of the background image. Borders were randomly placed around the 1-D art which were randomly rotated to simulate perspectives and thus appear as quadrilaterals.

This is an outline of what I did to prepare the MDE model for large scale fine-tuning:

1. Determined in general both Depth Anything v2, MiDaS v2.1, and Marigold, leading MDE models, had issues with 1-D flat images (e.g., paintings, mirror reflections)

2. Downloaded Hypersim and vKITTI datasets to augment

[1] https://github.com/apple/ml-hypersim
[2] https://www.kaggle.com/datasets
[3] https://www.mapillary.com/datasets
[4] https://europe.naverlabs.com/proxy-virtual-worlds-vkitti-2/

3. Developed python scripts to augment Hypersim and vKITTI images with embedded flat images, and update the ground truth depth maps (.hdf5, .png). The truth depth maps were altered by masking the regions where features were pasted onto the original image with a gradient of decreasing depth starting at the deepest corner of the region.

4. Refined scripts to produce augmented images that would fool Depth Anything and Marigold into predicting depth where there is no depth in embedded fat images (paintings, mirror images)
*to fool the exisiting MDE models, the embedded images had to be large enough to provide a background scene with sufficient depth clues to predict depth (avoid being overwhelmed by the underlying background scene), have objects in the foreground, and the embedded background scene must have sufficient relative depth: for example, a plaque on a wall would not have any depth to predict

5. Defined the two patterns that the MDE model would be fine-tuned to recognize:
    a. 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
    b. 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 painting)

6. Augmented Hypersim and vKITTI images
    a. Hypersim indoor images augmented with left or right side mirrors, and primarily images from Kaggle dataset for paintings
    b. vKITTI augmented with "wall paintings" of vKITTI images

7. Chose Depth Anything MDE model to fine-tune

8. Fine-tune with laptop with 1 GPU 8GB VRAM
*image size: 518, batch size: 1, LR: 0.000005, epochs: 48, pretrained model: VITL, datasplit: 80% train, 20% validation

9. Fine-tuning produces a .pth model with new weights

10. Restructured fine-tuned .pth model according to the dictionary and keys of the pretrained models

11. Used fine-tuned .pth model to predict depth on augmented images and compared to predictions by pre-trained Depth Anything VITL model
    a. troubleshooted with point-clouds (CloudCompare)
    b. troubleshooted by comparing the norms of keys-values (parameters) of the model's dictionary
    c. had to execute run.py from the /metric_depth directory with an explicit argument for max_depth, for some reason, otherwise the predictions would be a constant maximum depth

12. Switched to the pre-trained models in metric_depth/checkpoints to start fine-tuning

13. Froze the encoder layers and kept the decoder layers as trainable by setting the flag "requires_grad=False" (freeze)

Notes/Deficiencies/Observations:

1. may have to freeze some layers (encoder or decoder) to prevent the depth shifts in the fine-tuned model
2. fine-tuned model seems to be more sensitive and subtly more accurate when both encoder and decoder were trainable
3. fine-tuned model seems to experience a loss of information (depth maps “smeared”) when encoder was frozen and only decoder was trainable
4. the depth gradient applied to the embedded images takes the nearest depth and applies it forward from the deepest corner
5. Because embedded features (mirror reflections, 1-D art) had to be large enough to provide enough background environmental clues to fool the MDE models to predict depth in flat regions, the fine-tuned models “lost” the ability to predict depth of the original image’s background. In other words, fine-tuning with the augmented images over-trained the MDE model because the augmentation overwhelmed the original background.
6. In general, the MDE models do quite well, as-is, if the 1-D embedded features were small enough to lack environmental cues to predict depth. Conversely, to fool the MDE models, embedded 1-D art had to have relative depth with features/objects in the foreground.
7. With the Hypersim-augmented (mirror reflections and 1-D art) dataset, the MDE model was overtrained on quadrilateral regions because the augmenting features were all quadrilateral shapes. For instance, if a window in the background appeared whole, then the fine-tuned model would predict the window is in the foreground when it is actually in the background.