AI Assessment Report

Task 1:

Monocular Depth Estimation: A simple baseline model is implemented having an encoder-decoder architecture model and training + evaluation script on NYU Depth V2 (using TFDS).

Task 2:

I have chosen U-Net for this task. I have used a small subset of the NYU Depth V2 (downloaded from Kaggle) dataset for this task, since my office server went out of memory and apparently crashed after completing the baseline training. Please refer to the GitHub repository for more details.

Task 3:

I have chosen the top 3 approaches based on several factors including its proven effectiveness in minimizing RMSE and the alignment with state-of-the-art (SOTA) or recent advancements in the field.

Bibliographic Research

The top 3 approaches that I found out after some research are:

1. Depth Anything
   Paper: *https://arxiv.org/abs/2401.10891*
   Github: *https://github.com/LiheYoung/Depth-Anything*
2. ZoeDepth
   Paper: *https://arxiv.org/abs/2302.12288*
   Github: *https://github.com/isl-org/ZoeDepth.git*
3. DepthFormer
   Paper: *https://arxiv.org/abs/2203.14211*
   Github: *https://github.com/zhyever/Monocular-Depth-Estimation-Toolbox/tree/main/configs/depthformer*

Out of all these 3 approaches, 'Depth Anything' is the most recent innovation in the depth estimation field and it has surpassed all the other previously established state-of-the-art techniques.

It is also available in HuggingFace's Transformer library. So, a simple way to implement this would be to install the transformer library in a python environment and use the pretrained model.

```
pip install -q transformers
```

There are 2 main ways to use Depth Anything: either using the pipeline API, which abstracts away all the complexity, or by using the DepthAnythingForDepthEstimation class yourself.

```
from transformers import pipeline

>>> from PIL import Image
```

```
>>> import requests



>>> # load pipe

>>> pipe = pipeline(task="depth-estimation", model="LiheYoung/depth-
anything-small-hf")



>>> # load image

>>> url = 'http://images.cocodataset.org/val2017/000000039769.jpg'

>>> image = Image.open(requests.get(url, stream=True).raw)



>>> # inference

>>> depth = pipe(image)["depth"]
```

If you want to do the pre- and postprocessing yourself, here's how to do that. I have already integrated this approach in the project. Run the depthAnything.py python script from the 'script/' directory of the Github repository after cloning the repository and installing the dependencies. The content of the file is as follows:

```
from transformers import AutoImageProcessor,
AutoModelForDepthEstimation

>>> import torch

>>> import numpy as np

>>> from PIL import Image

>>> import requests
```

```
>>> url = "http://images.cocodataset.org/val2017/000000039769.jpg"

>>> image = Image.open(requests.get(url, stream=True).raw)



>>> image_processor =
AutoImageProcessor.from_pretrained("LiheYoung/depth-anything-small-hf")

>>> model =
AutoModelForDepthEstimation.from_pretrained("LiheYoung/depth-anything-
small-hf")



>>> # prepare image for the model

>>> inputs = image_processor(images=image, return_tensors="pt")



>>> with torch.no_grad():

...     outputs = model(**inputs)

...     predicted_depth = outputs.predicted_depth



>>> # interpolate to original size

>>> prediction = torch.nn.functional.interpolate(

...     predicted_depth.unsqueeze(1),

...     size=image.size[::-1],

...     mode="bicubic",

...     align_corners=False,

... )
```

```
>>> # visualize the prediction
>>> output = prediction.squeeze().cpu().numpy()
>>> formatted = (output * 255 / np.max(output)).astype("uint8")
>>> depth = Image.fromarray(formatted)
```

Time to finish: 4 days

Thank you.