Face to ball

Aswin

Pol

Tatsuki

Damien

Our topic

Topic 9

Face to Ball

- Create your dataset
- Create your own architecture
- Train it

Our topic

Goal

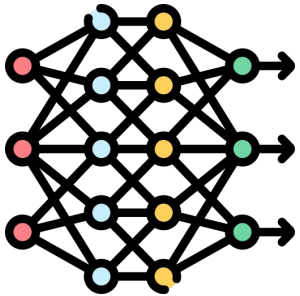
- Create a synthetic dataset.
- Capture the lighting effects on the face of people and potentially recreate it.

Methodology

Dataset

Image of **Face 3D model with light**

DL model



Output

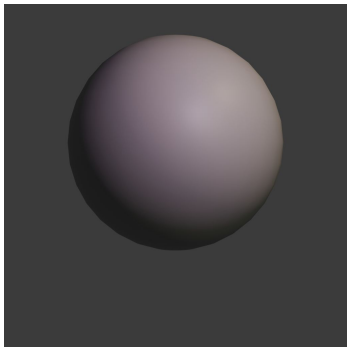
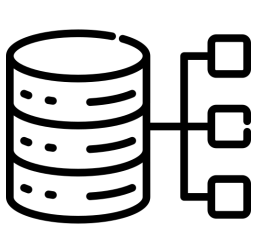
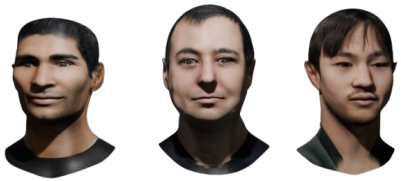


Image of **Grey ball with same light**

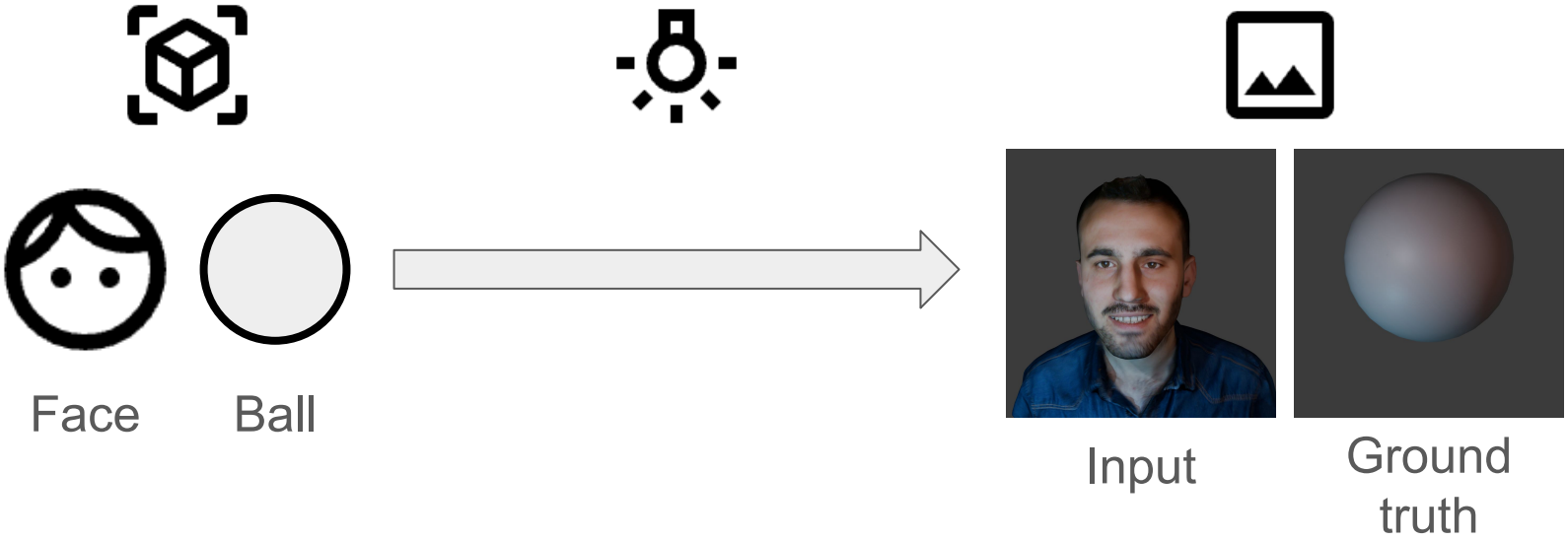
Workflow

Generate dataset → **Explore state of the art models** → **Design the architecture** → **Train and evaluate a model**





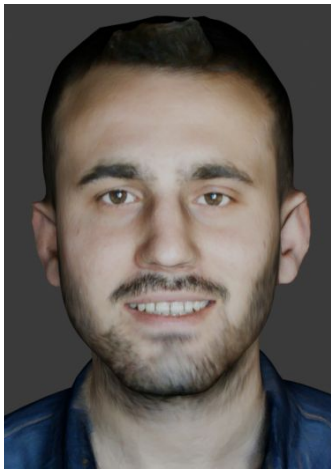
Dataset generation Pipeline



Generate 3D face model

Way 1 1 face

Free model from sketchfab



3D model

Way 2 3 faces

Keen Tools FaceBuilder

Real photo

3D model

Generate 3D face model

Way 3 1 face

TRELLIS

Real photo

3D model

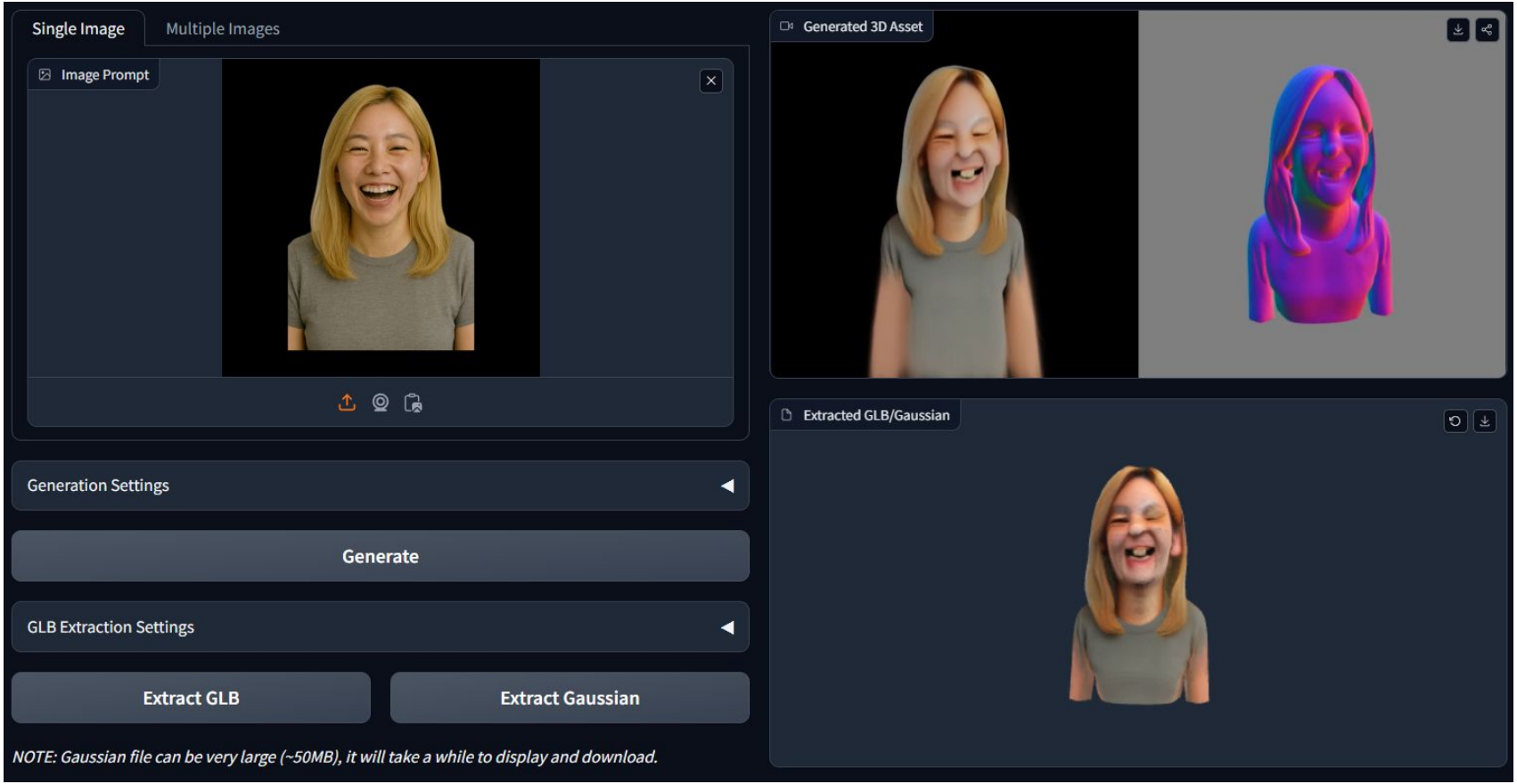
Way 4 11 faces

TRELLIS

AI photo

3D model

TRELLIS



https://huggingface.co/spaces/innoai/TRELLIS

Dataset



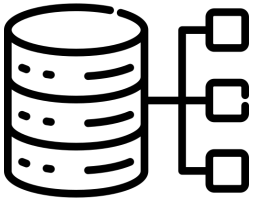

16 faces × 100 conditions (lighting and angle) = **1600 set of images**

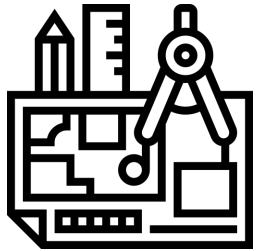
1 set = 1 face image + 1 ball image

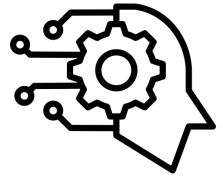
- **Training: 1200 set**
- Validation: 200 set
- Test: 200 set

Workflow

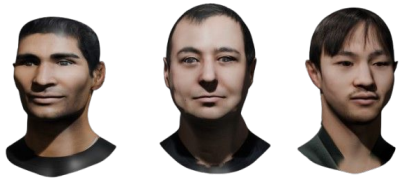
Generate dataset → **Explore state of the art models** → **Design the architecture** → **Train and evaluate a model**







✅



Exploring models

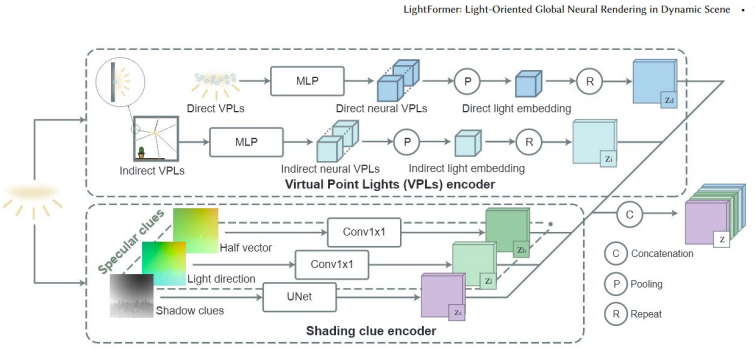
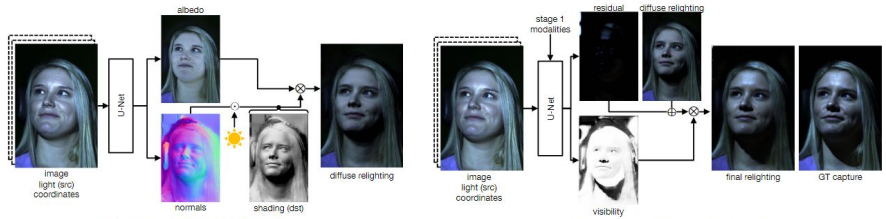


Fig. 2. Light Encoding Stage of LightFormer. For each light in the scene, we apply a series of VPL encoders and shading clue encoders to encode different light properties. These neural features are then concatenated to obtain a unified light embedding for each light.



(a) Stage 1: Diffuse Rendering. (b) Stage 2: Non-Diffuse Residual.

Figure 2: **Physics-guided relighting with structured generators.** Our generator consists of two stages modeling diffuse and non-diffuse effects. All intrinsic predictions are guided by losses w.r.t. photometric stereo reconstructions. **(a)** We use a U-Net with grouped convolutions to make independent predictions of the intrinsic components. Predicted normals are always re-normalized to unit vectors. Given a desired output lighting, we compute shading from normals and render a diffuse output. **(b)** Conditioned on all modalities inferred in (a), we predict a non-diffuse residual and binary visibility map to model specularities, cast shadows, and other effects not captured by our instance of the rendering equation.

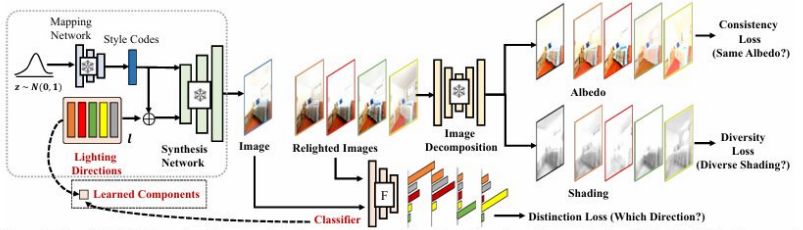


Figure 2. How StyLitGAN works: We generate an image from a random Gaussian noise using a pretrained StyleGAN. We also generate novel relighted images (16 in our case) of the same using randomly initialized latent directions (**d**) that are added to $\mathbf{w}^+$ latent style codes. We train a classifier (F) that takes in all the pairs of relighted and original images and predicts the relighting direction applied to them. We apply a distinction loss and jointly update the latent directions and the classifier. Next, we generate the decomposition of these images from a pretrained decomposition model (D). We then apply losses that force StyLitGAN to find latent directions so that the albedo does not change (consistency loss), but the image does (diversity loss).

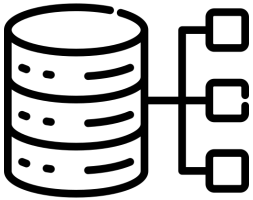
Exploring models

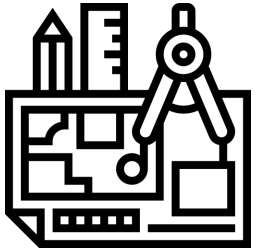
We found some potential candidates:

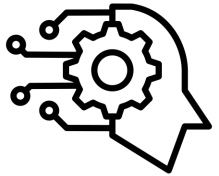
1. **UNet** [2]: U-Net can capture light information from a reference object and transfer it to a gray ball by using its encoder-decoder architecture with skip connections to preserve spatial lighting details. The model would be trained on paired images showing the same scene under different lighting conditions, learning to extract lighting features and apply them to new objects while maintaining geometric consistency. (supervised)
2. **GAN** [1],[3]: A GAN approach uses a generator (often U-Net-based) to create realistic lighting effects on a gray ball, while a discriminator ensures the results look authentic by comparing them to real lighting examples. This adversarial process excels at producing realistic specular highlights, shadows, and subtle lighting nuances that are difficult to achieve with traditional methods, resulting in more photorealistic light transfers. (unsupervised)

Workflow

Generate dataset → **Explore state of the art models** → **Design the architecture** → **Train and evaluate a model**

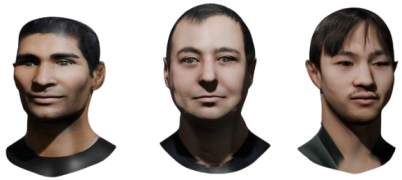




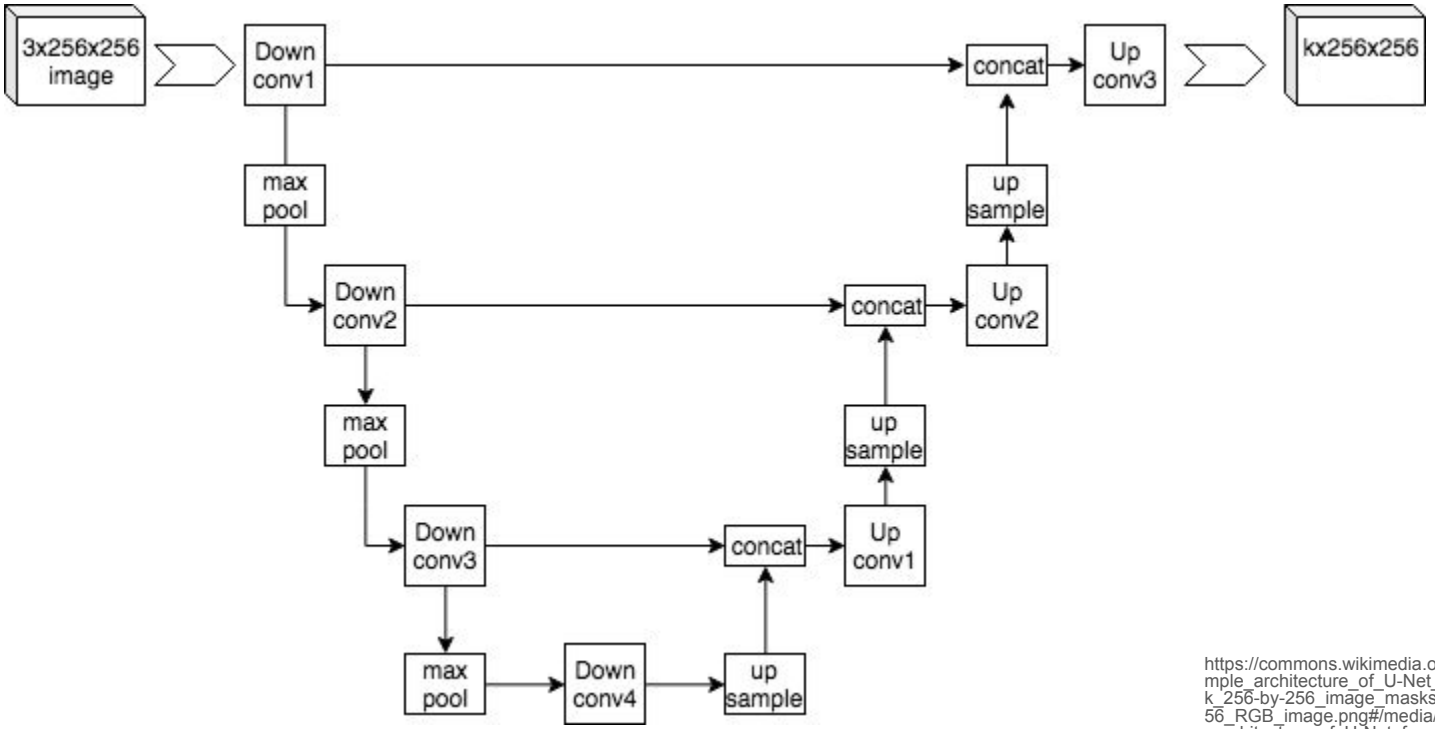


✅

✅



Design the architecture- UNet



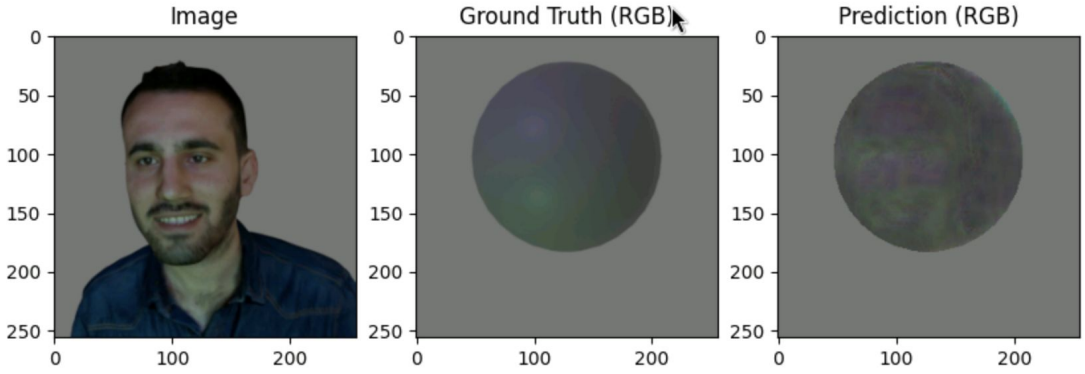
*For reference

https://commons.wikimedia.org/wiki/File:Example_architecture_of_U-Net_for_producing_k_256-by-256_image_masks_for_a_256-by-256_RGB_image.png#/media/Fichier:Example_architecture_of_U-Net_for_producing_k_256-by-256_image_masks_for_a_256-by-256_RGB_image.png

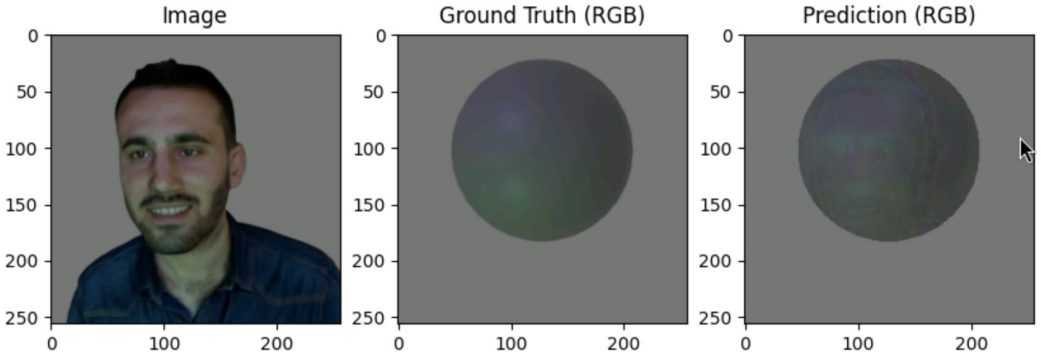
Design the architecture- UNet

Two approaches:

1. Freezing encoder and training decoder
    - Weights- Imagenet
    - Encoder- Resnet34
2. Training both encoder and decoder
    - Encoder- Resnet34



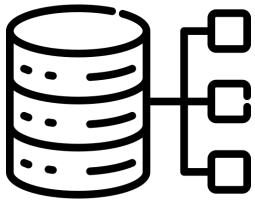
Freezing encoder and training decoder (50 epochs)

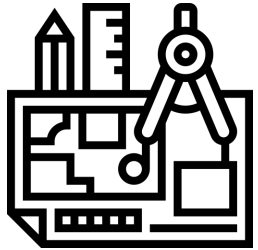


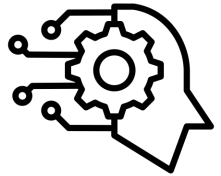
Training both encoder and decoder (50 epochs)

Workflow

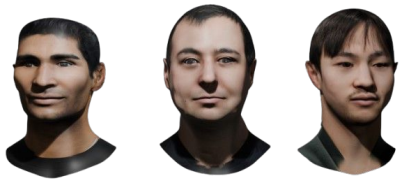
Generate dataset → **Explore state of the art models** → **Design the architecture** → **Train and evaluate a model**







✅ ✅ ✅

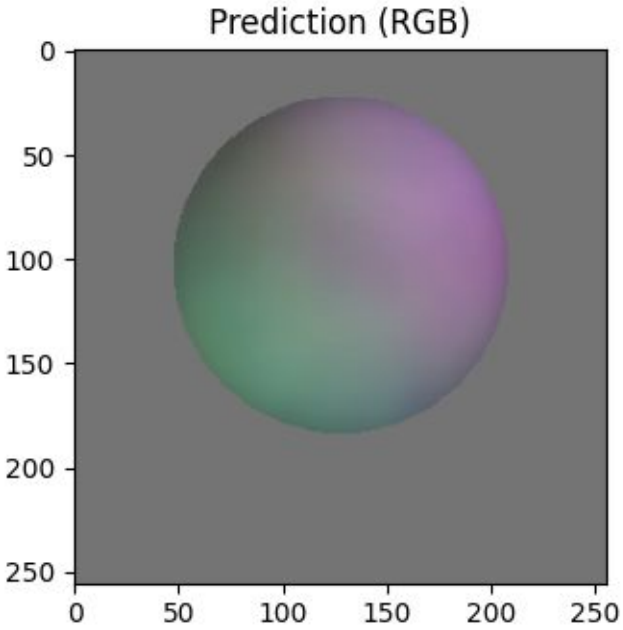


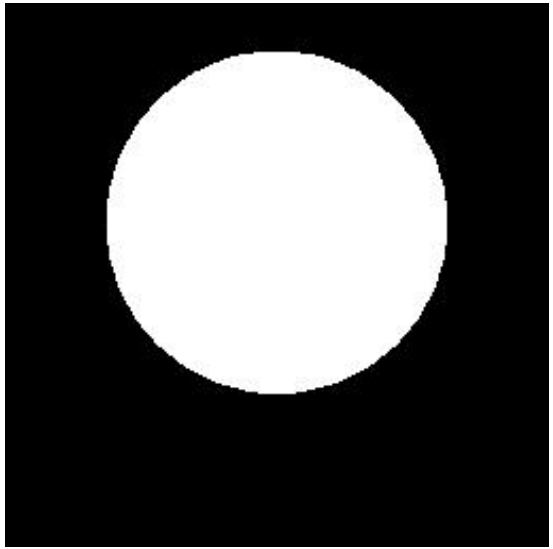
Training setup

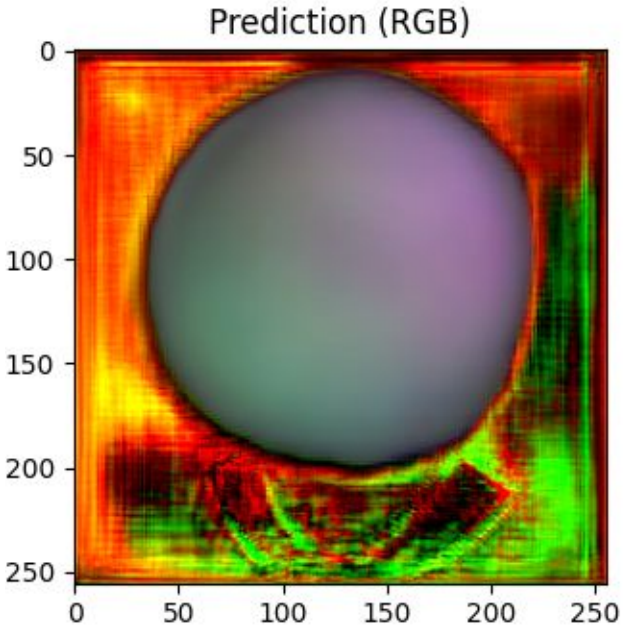
- Loss functions
  - **L1**
  - **Mean Squared Error (MSE)**
  - 1 - SSIM
  - Combination loss = MSE * 0.5 + (1 - SSIM) * 0.5

- Learning rate = 1e-4 ~ 1e-5
- Batch size = 50
- Epochs = 500

Computing loss

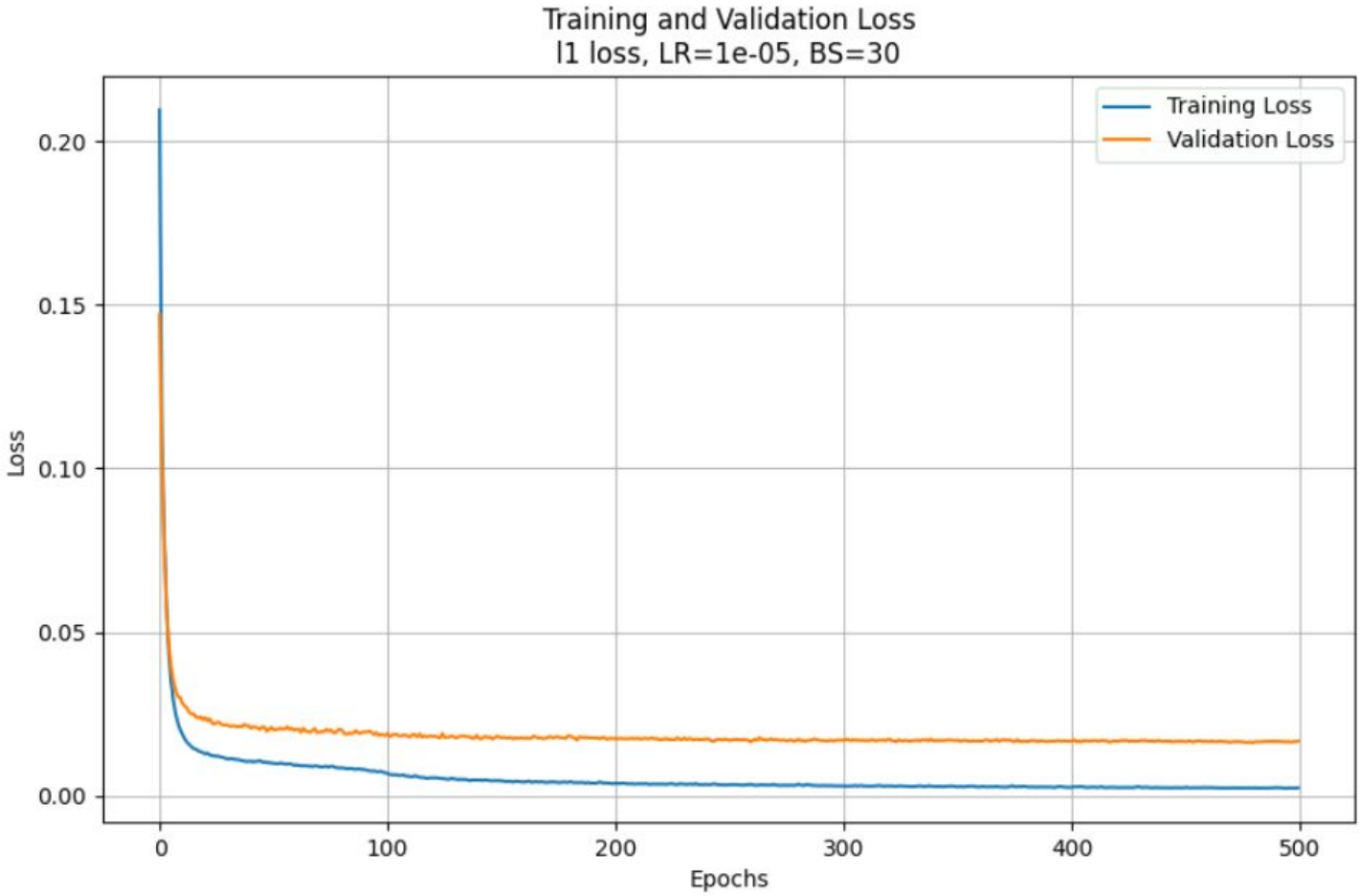






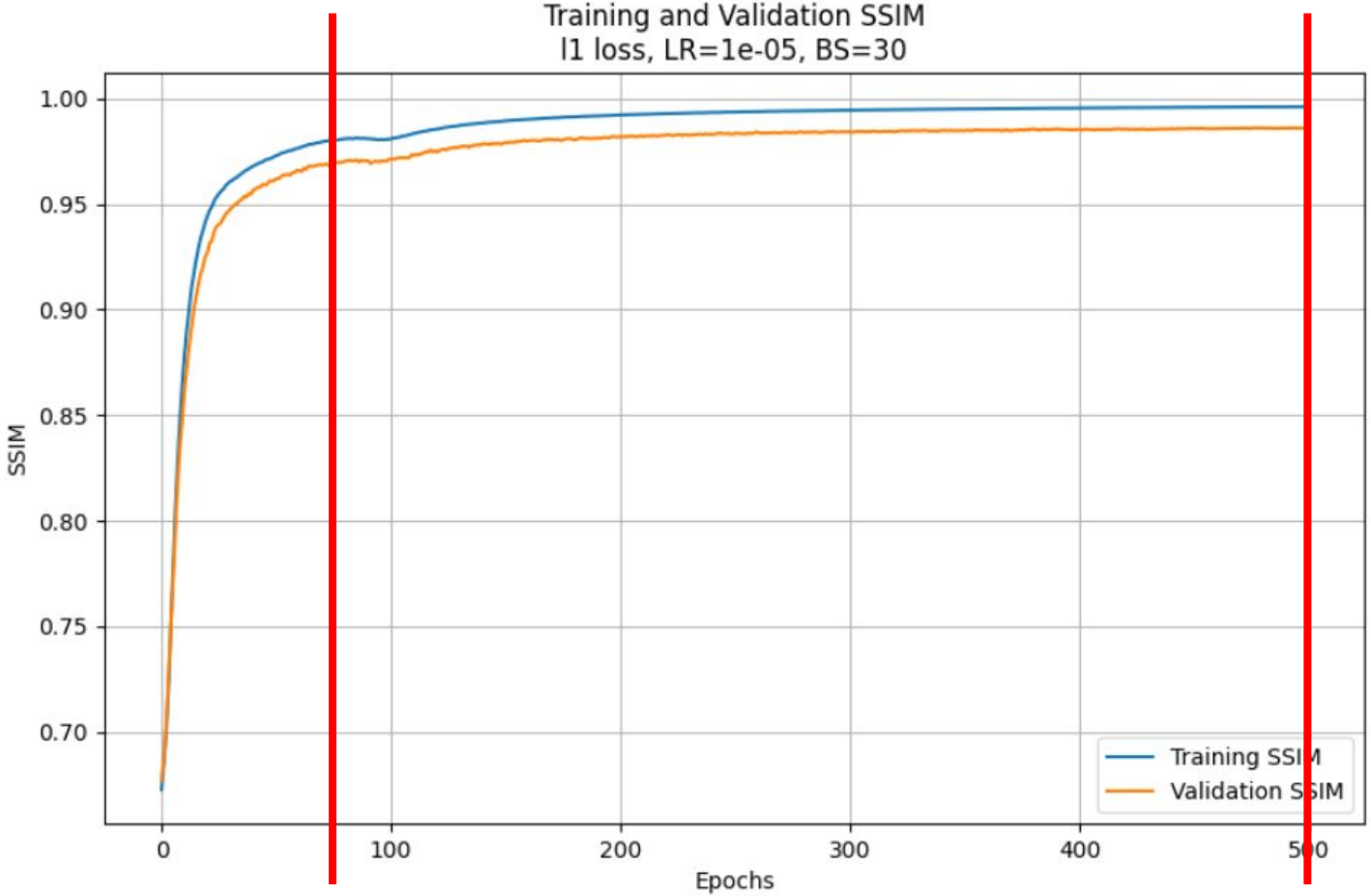
Training metrics

Loss



Training metrics

SSIM



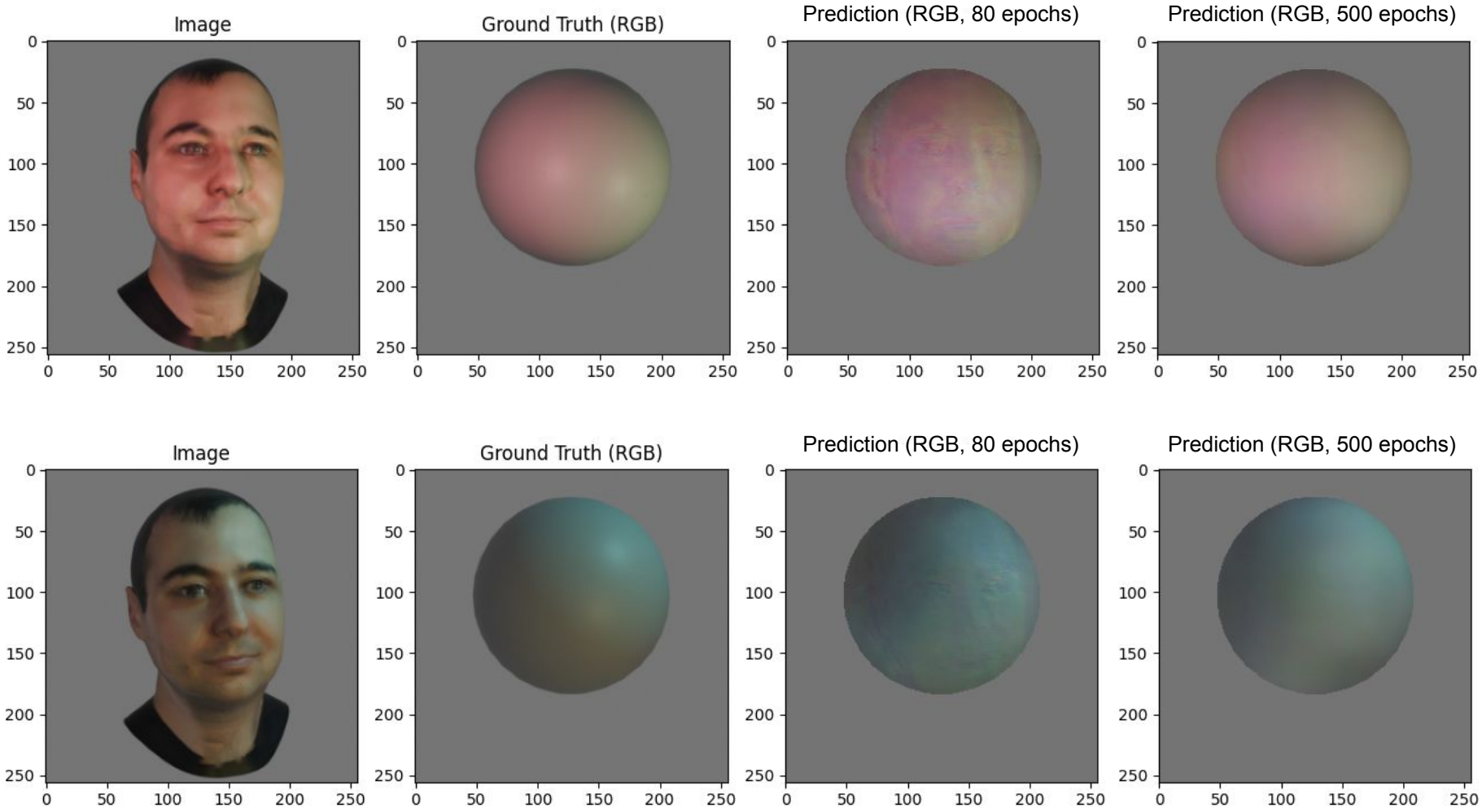


Image
Ground Truth (RGB)
Prediction (RGB, 80 epochs)
Prediction (RGB, 500 epochs)
Image
Ground Truth (RGB)
Prediction (RGB, 80 epochs)
Prediction (RGB, 500 epochs)

Real Data

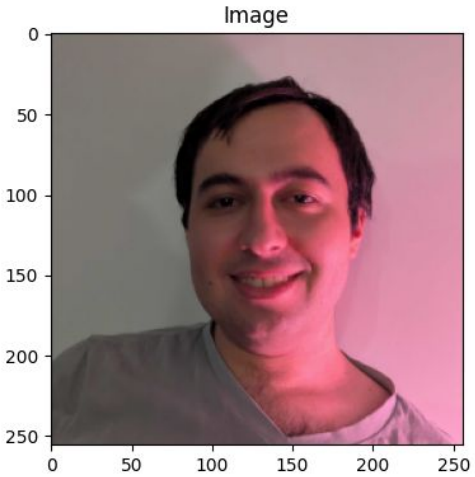
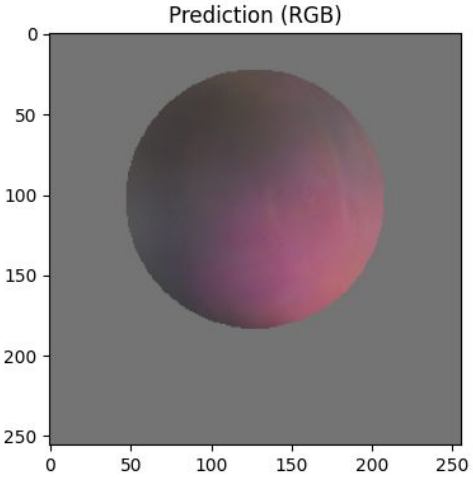


Image
0
50
100
150
200
250
0
50
100
150
200
250


Prediction (RGB)
0
50
100
150
200
250
0
50
100
150
200
250

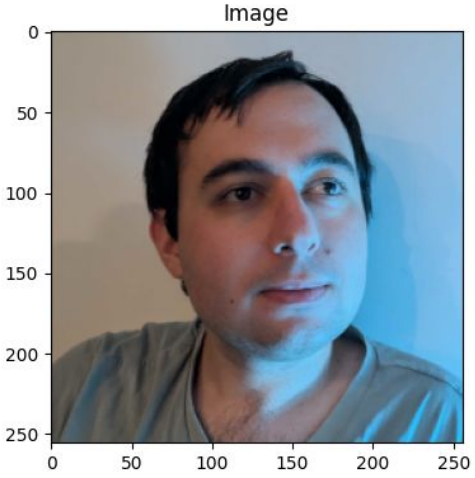
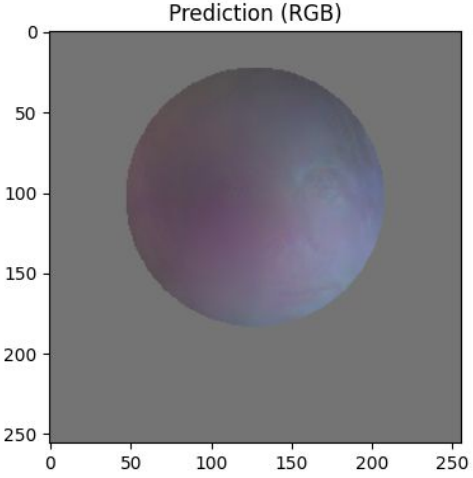


Image
0
50
100
150
200
250
0
50
100
150
200
250


Prediction (RGB)
0
50
100
150
200
250
0
50
100
150
200
250

Real Data

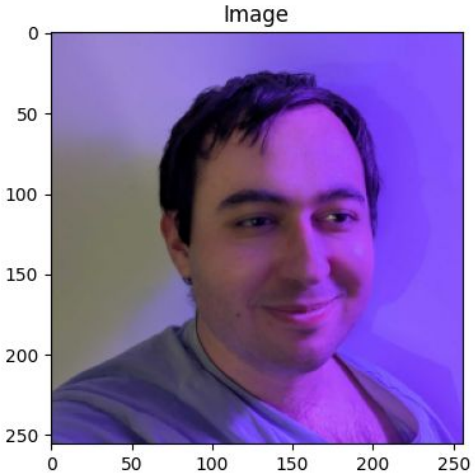
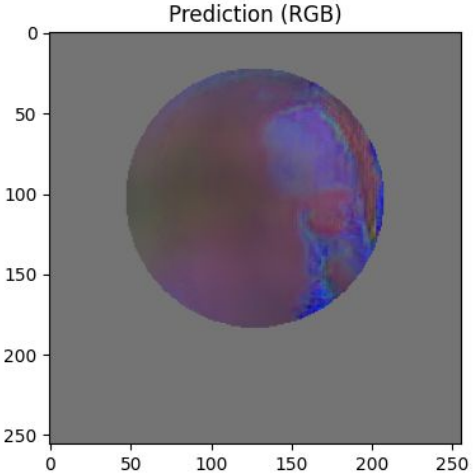


Image
0
50
100
150
200
250


Prediction (RGB)
0
50
100
150
200
250

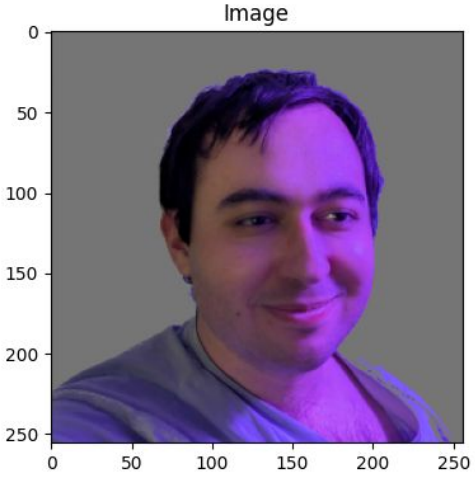
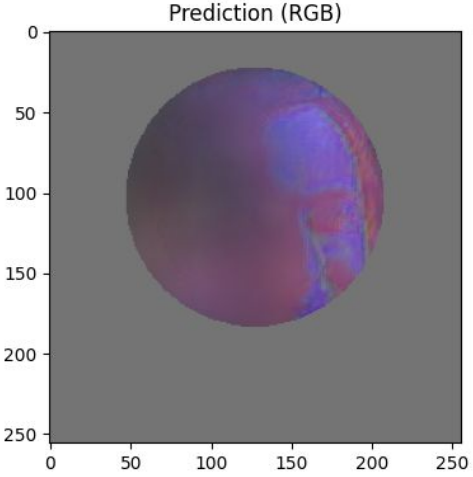
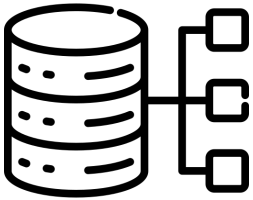


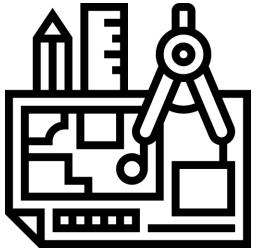
Image
0
50
100
150
200
250

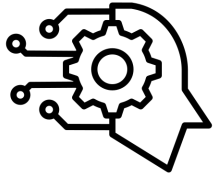

Prediction (RGB)
0
50
100
150
200
250

Workflow

Generate dataset → **Explore state of the art models** → **Design the architecture** → **Train and evaluate a model**







✅ ✅ ✅ ✅

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” Nov. 26, 2018, arXiv: arXiv:1611.07004. doi: 10.48550/arXiv.1611.07004.

[2] T. Nestmeyer, J.-F. Lalonde, I. Matthews, and A. Lehrmann, “Learning Physics-Guided Face Relighting Under Directional Light,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA: IEEE, Jun. 2020, pp. 5123–5132. doi: 10.1109/CVPR42600.2020.00517.

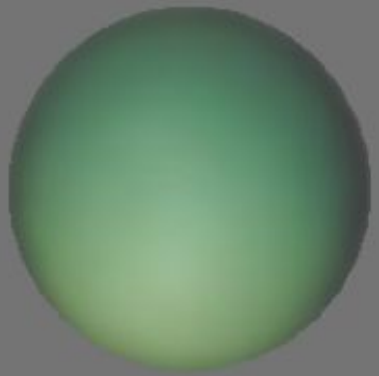
[3] H. Ren et al., “LightFormer: Light-Oriented Global Neural Rendering in Dynamic Scene,” ACM Trans. Graph., vol. 43, no. 4, pp. 1–14, Jul. 2024, doi: 10.1145/3658229.

[4] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang, “Structured 3D Latents for Scalable and Versatile 3D Generation,” arXiv preprint arXiv:2412.01506, 2024.

Face to ball

Aswin

Pol



Tatsuki

Damien