Automatic Reflection Removal: noreflections.ai

This project aims to develop an AI model that can automatically detect and remove reflections from photographs taken through reflective surfaces, such as glass windows or water. By doing so, the solution will reveal the true scene behind the reflection, enhancing the clarity and quality of the image.

Team: weshowspeed
Tead id-Qual-230388

Shaman Shetty
Aryan Tiwari
Krishna Shetty
Dishank Vyas

Project Overview: Motivation and Objectives

1 Overcoming Reflections

Reflections in photographs can obscure important details and distort the intended subject, hindering the usefulness of the image.

2 Enhancing Image Quality

By removing reflections, the AI model will improve the clarity and fidelity of the underlying scene, making the image more valuable for various applications.

3 Improving Computer Vision

Developing a robust reflection removal algorithm will contribute to the advancement of computer vision and image processing technologies.

The Challenge of Reflections in Photographs

Distortion

Reflections can cause unwanted distortions in the image, such as blurring, ghosting, and overlapping of objects.

Occlusion

Reflections can obscure important details and features of the intended subject, making it difficult to discern the true scene.

Complexity

Reflections can vary in intensity, position, and shape, making it challenging to develop a one-size-fits-all solution.

Designing the AI Model: U-Net Architecture

1

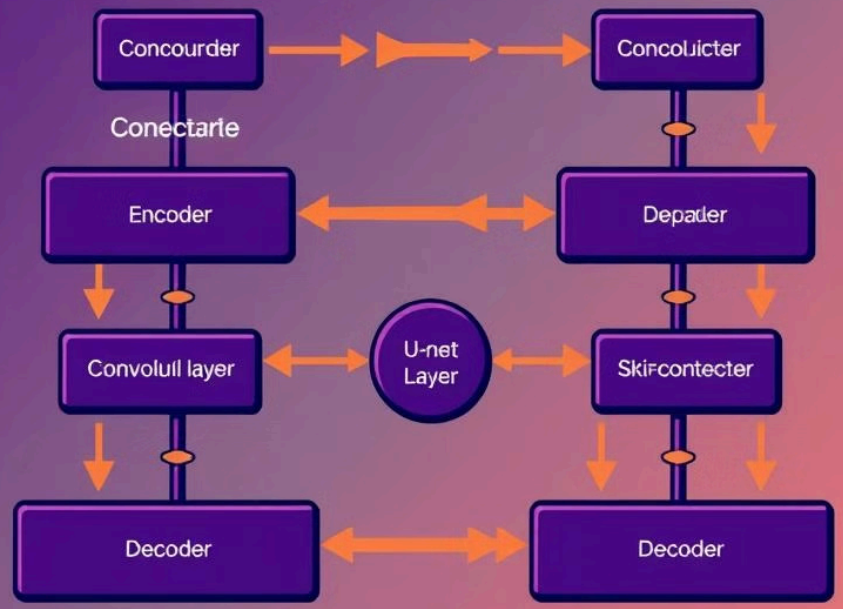
Encoder

The encoder portion of the U-Net model learns to extract relevant features from the input image.

2

Decoder

The decoder portion of the model uses the bottleneck features to reconstruct the final output image with reflections removed.



Encoder

```
# Enhanced U-Net architecture with padding to maintain spatial dimensions
def build_unet(input_shape):
    inputs = Input(shape=input_shape)

    # Encoder (Downsampling)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1)
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(512, 3, activation='relu', padding='same')(pool3)
    conv4 = Conv2D(512, 3, activation='relu', padding='same')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(1024, 3, activation='relu', padding='same')(pool4)
    conv5 = Conv2D(1024, 3, activation='relu', padding='same')(conv5)
```

Decoder

```
# Decoder (Upsampling)
up1 = Conv2DTranspose(512, (2, 2), strides=(2, 2), padding='same')(conv5)
up1 = concatenate([up1, conv4])
conv6 = Conv2D(512, 3, activation='relu', padding='same')(up1)
conv6 = Conv2D(512, 3, activation='relu', padding='same')(conv6)

up2 = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv6)
up2 = concatenate([up2, conv3])
conv7 = Conv2D(256, 3, activation='relu', padding='same')(up2)
conv7 = Conv2D(256, 3, activation='relu', padding='same')(conv7)

up3 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv7)
up3 = concatenate([up3, conv2])
conv8 = Conv2D(128, 3, activation='relu', padding='same')(up3)
conv8 = Conv2D(128, 3, activation='relu', padding='same')(conv8)

up4 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv8)
up4 = concatenate([up4, conv1])
conv9 = Conv2D(64, 3, activation='relu', padding='same')(up4)
conv9 = Conv2D(64, 3, activation='relu', padding='same')(conv9)

# Final layer
outputs = Conv2D(3, 1, activation='sigmoid', padding='same')(conv9)

model = Model(inputs=inputs, outputs=outputs)
return model
```

Data Collection and Preprocessing

Dataset Acquisition

Gathering a diverse dataset of photographs with reflections, including both indoor and outdoor scenes.

Data Augmentation

Applying techniques like flipping, rotation, and noise addition to expand the dataset and improve model robustness.

Preprocessing

Carefully cleaning and preprocessing the data to ensure consistent input format and quality for the model.

Ground Truth

Generating ground truth images with reflections removed to serve as the target output for the model.

Dataset Acquisition

```python
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Conv2DTranspose, concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
import matplotlib.pyplot as plt

def is_image_file(filename):
    """
    Check if the file is an image file
    """
    image_extensions = ['.jpg', '.jpeg', '.png', '.bmp', '.gif', '.tiff', '.webp']
    return any(filename.lower().endswith(ext) for ext in image_extensions)

def load_images(folder):
    images = []
    skipped_files = []

    # List all files in the folder
    all_files = os.listdir(folder)
    print(f"Total files in {folder}: {all_files}")

    # Filter image files
    image_files = [f for f in all_files if is_image_file(f)]
    print(f"Image files found: {image_files}")

    for filename in image_files:
        try:
            # Full path to the image
            img_path = os.path.join(folder, filename)

            # Read image with error handling
            img = cv2.imread(img_path)

            # Check if image is None (failed to read)
            if img is None:
                print(f"Failed to read image: {img_path}")
                skipped_files.append(filename)
                continue

            # Convert color space from BGR to RGB
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Normalize pixel values
            img = img.astype(np.float32) / 255.0

            # Resize image
            img = cv2.resize(img, (256, 256))

            images.append(img)

        except Exception as e:
            print(f"Error processing {filename}: {e}")
            skipped_files.append(filename)

    # Print summary
    print(f"Successfully loaded {len(images)} images")
    if skipped_files:
        print(f"Skipped files: {skipped_files}")

    return np.array(images)

# Define folder paths
with_reflection_folder = '/content/drive/MyDrive/Train/TRAIN/with reflection images'
without_reflection_folder = '/content/drive/MyDrive/Train/TRAIN/without reflection images'
test_cases_folder = '/content/drive/MyDrive/Test /TEST'

# Load images from each folder
with_reflection_images = load_images(with_reflection_folder)
without_reflection_images = load_images(without_reflection_folder)
test_cases_images = load_images(test_cases_folder)

# Split the dataset into training and validation subsets
train_images = with_reflection_images[:8]
train_labels = without_reflection_images[:8]
val_images = with_reflection_images[8:]
val_labels = without_reflection_images[8:]

# Print dataset sizes
print(f"Training images shape: {train_images.shape}")
print(f"Training labels shape: {train_labels.shape}")
print(f"Validation images shape: {val_images.shape}")
print(f"Validation labels shape: {val_labels.shape}")
```

Data Augmentation

```
# Data augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Model Training and Optimization

1

Model Architecture

Refining the U-Net model design to optimize performance for the reflection removal task.

2

Loss Function

Defining an appropriate loss function to guide the model towards accurate reflection removal.

3

Hyperparameter Tuning

Carefully adjusting hyperparameters like learning rate, batch size, and regularization to improve model convergence.

Model Architecture

```
# Build and compile the U-Net model
unet_model = build_unet(input_shape=(256, 256, 3))
unet_model.compile(optimizer='adam', loss=combined_loss, metrics=['mae'])

# Print model summary
unet_model.summary()
```

```
# Learning rate scheduler
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6)

# Train the model
history = unet_model.fit(
    datagen.flow(train_images, train_labels, batch_size=1),
    epochs=35,
    validation_data=(val_images, val_labels),
    callbacks=[lr_scheduler]
)
```

Evaluating the Model's Performance

Quantitative Metrics

Measuring the model's performance using objective metrics like PSNR, SSIM, and L1 loss.

Visual Assessment

Conducting human evaluation of the model's output to ensure the removal of reflections is visually pleasing and faithful to the original scene.

Edge Case Analysis

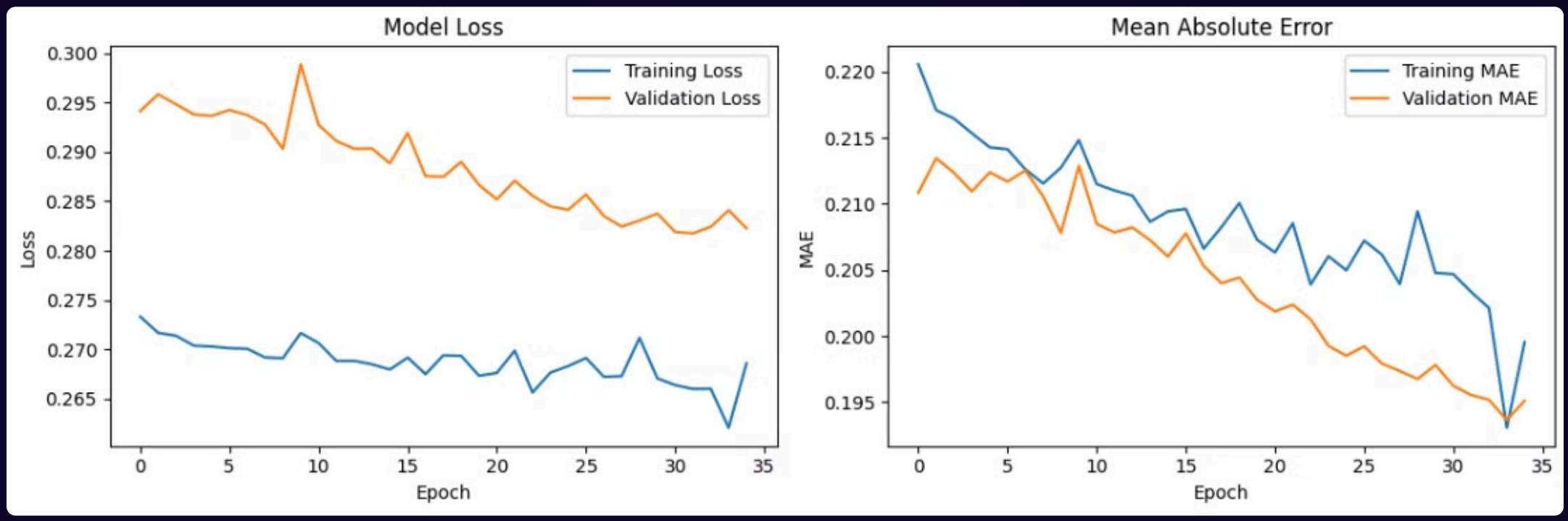
Identifying and addressing challenging edge cases, such as complex reflections or low-light conditions.

Performance:

```
# Optional: Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['mae'], label='Training MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.title('Mean Absolute Error')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.legend()

plt.tight_layout()
plt.show()
```



improvement of code:

model architecture visualization:

```
# Build and compile the U-Net model
unet_model = build_unet(input_shape=(256, 256, 3))
unet_model.compile(optimizer='adam', loss=combined_loss, metrics=['mae'])

# Print model summary
unet_model.summary()
```

Key Improvements Highlighted:

Advanced U-Net Architecture

- Enhanced network depth
- Added more convolutional layers
- Improved feature extraction capabilities
- Increased model complexity from 4 to 5 encoding/decoding blocks

Sophisticated Loss Function

- Implemented Combined Loss Approach
- Integrated Structural Similarity Index (SSIM)
- Balanced Mean Squared Error with SSIM
- Improved image quality preservation

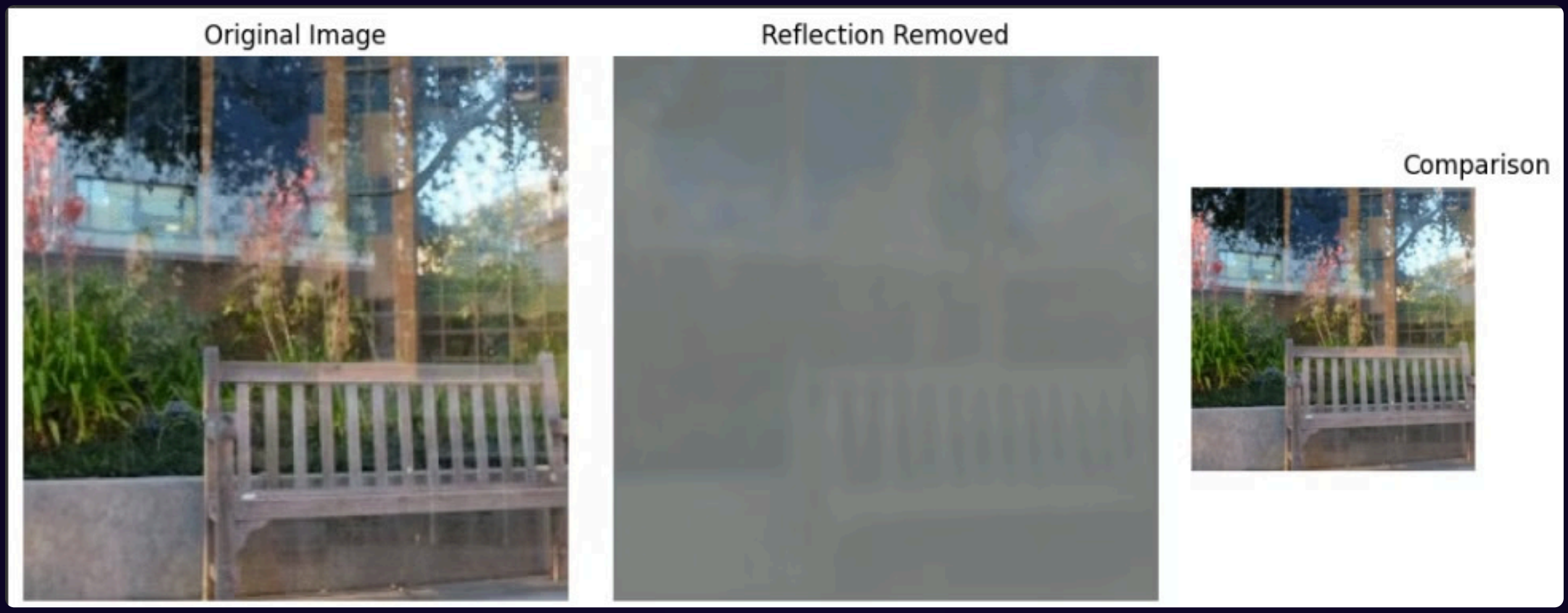
Data Augmentation Techniques

- Rotation range: ±20 degrees
- Width/Height shift: ±20%
- Shear range: ±20%
- Zoom capabilities
- Horizontal flipping

Comprehensive Evaluation Metrics

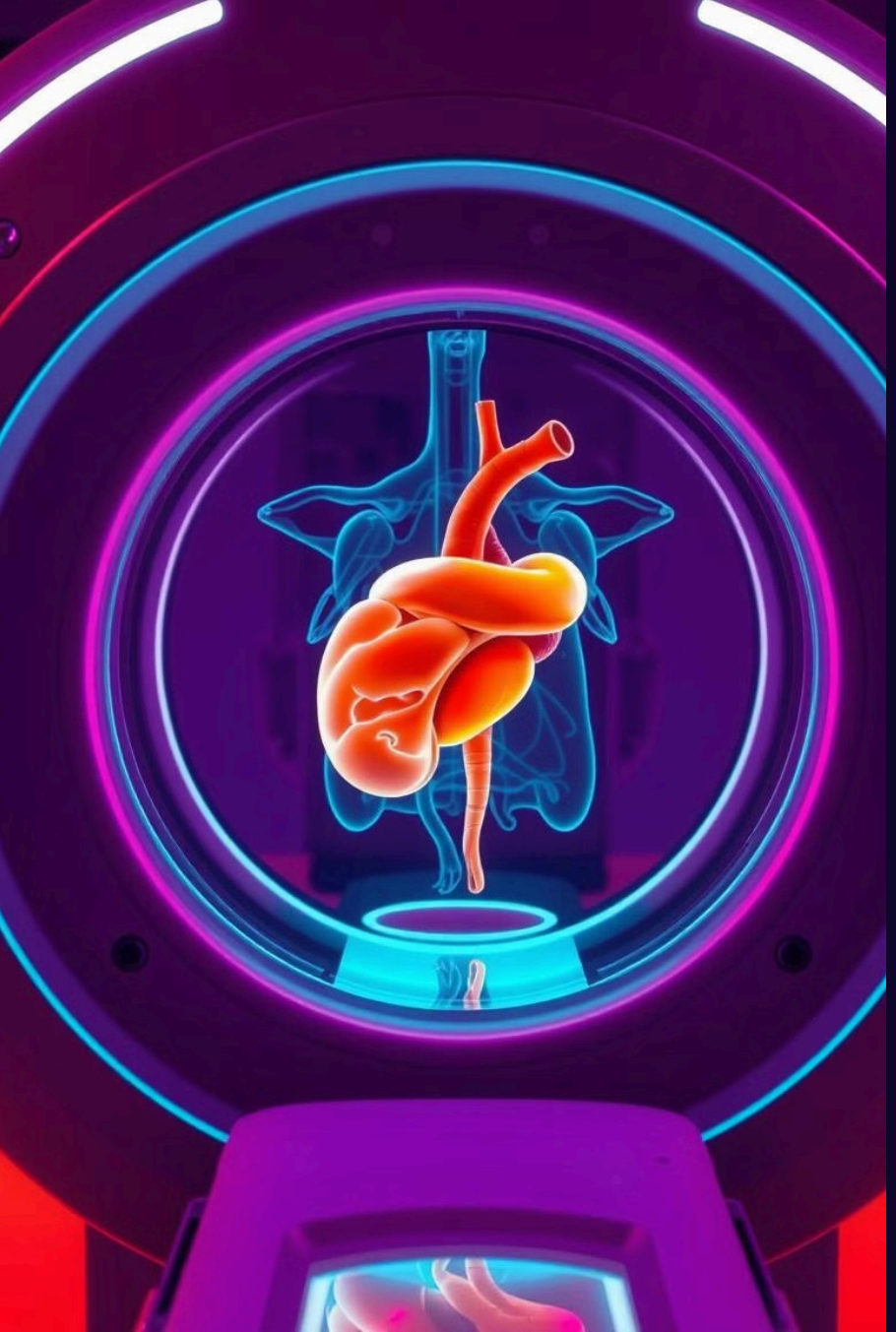
- Added validation loss tracking
- Mean Absolute Error (MAE) monitoring
- Detailed performance assessment
- Visualization of training progression

image output:



Deployment and Real-World Applications

Photography	Removing reflections from photos taken through windows or water surfaces to enhance image quality.
Surveillance	Improving the accuracy of object detection and tracking in surveillance systems by eliminating reflections.
Autonomous Vehicles	Enabling better perception and scene understanding for self-driving cars by addressing reflections on windshields.



Medical Imaging

- "Enhancing Medical Diagnostics"
- Impact:
  - Removes reflections from medical scans
  - Improves diagnostic accuracy
  - Helps medical professionals see clearer images
  - Potential to detect subtle medical details

Forensic Analysis

- "Bringing Clarity to Critical Evidence"
- Key Benefits:
  - Improves surveillance camera footage
  - Reveals hidden details in reflective surfaces
  - Assists in criminal investigations
  - Enhances image quality for legal proceedings

Photography

- landscape
  Challenges:
  - Reflections in water bodies
  - Glass glare in greenhouse or museum settings
  - Unwanted reflections in glass-covered displays

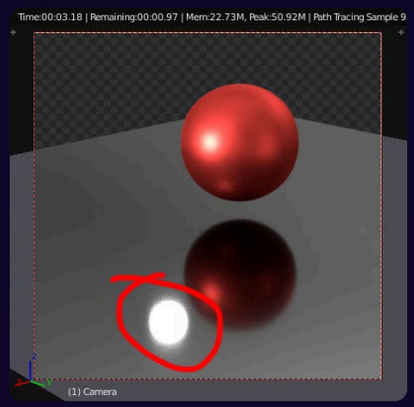
Product and Commercial Photography

- Professional Use Cases:
  - Remove reflections from glossy products
  - Enhance jewelry and metallic surface photography
  - product shots
  - Eliminate glare from glass or polished surfaces
  - Eliminate reflections from eyeglasses
  - Remove background glass reflections
  - Enhance studio and outdoor portrait clarity

Future Scope

Blurring reflections or object

Blurring reflections or objects in an image can create a sense of depth and focus, drawing the viewer's attention to the main subject. This technique is often used in photography and graphic design to enhance the visual appeal and storytelling of the scene. By selectively blurring certain elements, artists can mimic the effects of shallow depth of field, adding a professional touch to their work.



Selective removal or blurring

Selective removal or blurring is a powerful technique used in image editing and photography to emphasize certain elements while de-emphasizing others. By strategically removing or blurring background details, distractions, or unwanted objects, the main subject of the image becomes more prominent. This approach not only enhances the visual focus but also helps in creating a cleaner, more impactful composition. It is commonly employed in portraiture to highlight the subject and in landscape photography to simplify complex scenes.

Mixing and matching reflections and objects

Mixing and matching reflections and objects in a composition can create intriguing and dynamic visuals. By strategically incorporating reflections alongside physical objects, artists and photographers can add depth, symmetry, and a sense of surrealism to their work. This technique often plays with the viewer's perception, blending reality and illusion to create a more engaging and thought-provoking piece. Whether in photography, graphic design, or fine art, the interplay between reflections and objects can enhance storytelling and aesthetic appeal.

Team

Shaman Shetty

Team Leader, noreflections.ai

Aryan Tiwari

Developer,

noreflections.ai

Krishna Shetty

Developer,

noreflections.ai

Dishank Vyas

Developer,

noreflections.ai