Salient Object Detection (SOD) Project Report

Author: Ridon Maliqi
Date: November 2025

Project Report – Salient Object Detection (SOD)

1. Project Overview

- Goal: Develop a **Salient Object Detection system from scratch** using CNNs.
- Saliency focuses on identifying the **most visually important objects** in an image.
- Implemented the **full pipeline**: dataset preprocessing, model design, training, evaluation, and interactive demo.

2. Learning Objectives

By completing this project, I achieved:

- Strengthened understanding of **deep learning fundamentals**.
- Implemented a **complete ML pipeline** without pre-trained models.
- Handled image datasets efficiently, including **augmentation**.
- Designed, trained, and debugged a **CNN encoder-decoder architecture**.
- Computed **IoU, Precision, Recall, F1-score, and MAE**.
- Visualized model predictions and interpreted results critically.
- Built a **Gradio app** for interactive demonstration.

3. Technical Requirements

3.1 Dataset

- Dataset: **DUTS** (22000+ images, pixel-accurate saliency masks)
- Preprocessing:
    - Resize to 224×224
    - Normalize pixels to 0–1
    - Train/Validation/Test split: 70% / 15% / 15%
    - Augmentations: horizontal flip, random crop, brightness adjustment

3.2 Model Architecture

- Input: RGB image (224×224×3)
- Encoder: 4 Conv2D layers with ReLU, each followed by MaxPooling
- Decoder: 4 ConvTranspose2D layers with ReLU
- Output: 1-channel Sigmoid mask
- Loss: Binary Cross-Entropy + 0.5 × (1 − IoU)
- Optimizer: Adam (lr = 1e-3)
- Epochs: 20, Early stopping applied

3.3 Training

- Custom **training loop** with:
    - Forward/backward pass
    - Logging of loss and metrics per epoch
    - Validation per epoch
    - Checkpoint saving and resuming capability

3.4 Evaluation

- Metrics: IoU, Precision, Recall, F1-score, MAE
- Visualizations: Input → Ground Truth → Predicted Mask → Overlay

3.5 Experiments & Improvements

- Improvements over baseline:
    - Added **Batch Normalization** and **Dropout**
    - Increased depth of convolutional layers

    - Enhanced augmentation (horizontal flip, random crop, brightness shift)
- Compared **baseline vs improved** in table:

Model	IoU	Precision	Recall	F1
Baseline	0.6504	0.7831	0.8484	0.8144
Improved	0.6449	0.8238	0.8049	0.8143

3.6 Demo / Visualization

- **Gradio web app** implemented:
    - Upload an image → display predicted saliency mask → overlayed visualization
    - Measures **inference time per image (~0.025 sec)**
    - User-friendly interface for interactive demonstration

4. Tools & Environment

- Language: Python 3.9+
- Framework: TensorFlow/Keras
- Libraries: NumPy, OpenCV, Matplotlib, scikit-learn, tqdm, Gradio
- Environment: Google Colab (T4 GPU), Local Jupyter Notebook for testing
- Version Control: Git/GitHub

5. Experiments and Results

- Training progress monitored via **loss and IoU curves**
- Sample visual results:

Example Visualizations:

1. Input Image
2. Ground-truth Mask
3. Predicted Mask
4. Overlay

- Observed that the **baseline model** performed better than the improved version in IoU; training further is needed for improvements to surpass baseline

6. Learnings

- Building SOD from scratch enhances **deep learning intuition**
- Proper **dataset preprocessing and augmentation** is critical
- Visualization helps **debug model predictions**
- Gradio provides **easy deployment** for interactive demos
- Early stopping and checkpoints ensure **training stability and reproducibility**

7. Deliverables

- **Code files:** `data_loader.py`, `sod_model.py`, `train.py`, `evaluate.py`, `app.py`
- **Report:** 8 pages PDF
- **Presentation Slides:** 5 slides summarizing key results