Salient Object Detection (SOD) Project

Mentor: Dafina Berisha

Xponian: Diona Muçiqi

genpact

Abstract

This project presents a complete deep-learning pipeline for Salient Object Detection (SOD) using a custom U-Net++–based encoder–decoder architecture. The ECSSD dataset was preprocessed using a TensorFlow data pipeline with resizing, normalization, and augmentation to improve generalization. The model was trained for 25 epochs using the Adam optimizer and Binary Cross-Entropy loss, achieving stable performance on the test set (IoU: 0.69, F1: 0.71). Quantitative metrics and qualitative visualizations demonstrate that the model accurately extracts salient foreground regions from natural images. The final system provides an end-to-end implementation covering dataset preparation, model design, training, evaluation, and result interpretation.

2. Introduction

Salient Object Detection (SOD) is the process of identifying the most important regions in an image — the parts that naturally attract human attention, such as people, animals, or other noticeable objects. Figure 1 shows an example of an input image and the corresponding output after the main object has been detected.

The goal of SOD is to imitate how humans observe a scene by focusing on the most visible and meaningful areas while ignoring unnecessary background details. Identifying the main object in an image is useful for many computer vision tasks, because it makes the processing faster, more accurate, and more efficient.

Figure 1: The input images and outputs of salient object detection

3. Survey of Salient Object Detection Methods

Existing SOD methods are commonly divided into two main groups: **traditional methods** and **deep-learning-based methods**, as shown in Figure 2.

Traditional methods use basic image features such as contrast, color, and simple handcrafted rules to detect the most visible object in a scene.

Deep-learning methods learn richer and more meaningful features from data, which results in significantly better performance. These approaches can be:

- **Fully supervised**, using pixel-level ground truth masks.
- **Weakly supervised**, using simpler or partial labels.

In this project, I focused on a **deep-learning approach**. I implemented a CNN encoder–decoder model from scratch, prepared the dataset, trained the model, and evaluated the performance. This report presents the full workflow, experiments, and results of the developed SOD system.

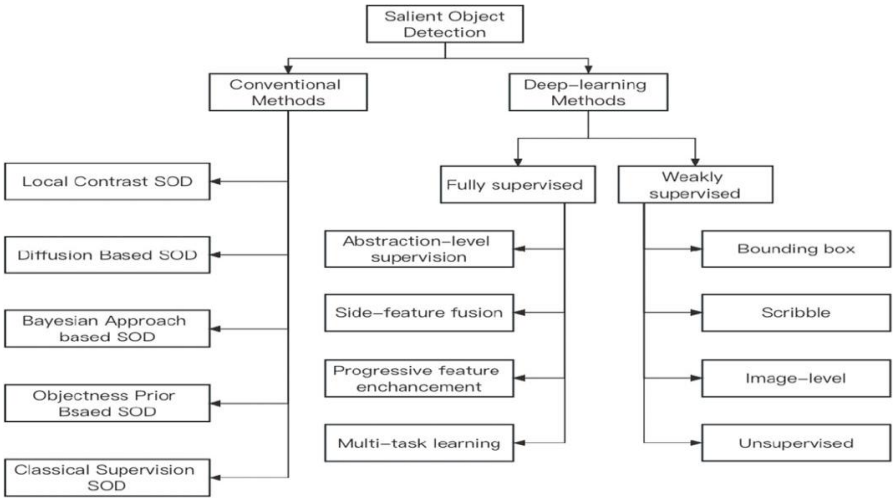

Figure 2: Existing salient object detection methods.

3.1. Dataset and Preprocessing

For this project, the **ECSSD** dataset was used, which contains natural images paired with high-quality pixel-level saliency masks. Each image has a corresponding ground-truth mask with the same filename, enabling automatic matching between image–mask pairs.

All dataset preparation and preprocessing were implemented using a custom TensorFlow data pipeline. Images and masks were resized to **224×224**, normalized to the **0–1** range, and organized into **training, validation, and test** splits using an 70/15/15 ratio. Data augmentation was applied only to the training set to improve generalization.

To provide a clear summary of the full preprocessing pipeline, dataset configuration, and augmentation steps, **Table 1** presents all operations used during this stage of the project.

Table 1: Dataset Configuration and Preprocessing Summary

Component	Details	Notes
Dataset	ECCSD	Public benchmark dataset
Image Size	224 x 224 pixels	Depends on dataset size
Normalization	Pixel values scaled 0-1	Image + mask normalized
Train /Val/Test	70 % / 15% / 15%	Applied during preprocessing
Batch Size	4	As defined in data_loader.py
Mask	Single-channel grayscale	Values between 0-1
Augmentations	Horizontal flip Brightness adjustment Random crop	Applied only in training set
Pipeline Stages	Load – Decode – Resize –Normalize – Augment – Batch - Prefetch	Optimized tf.data pipeline
Loader Framework	TensorFlow	Efficient input pipeline
Final Output	**Train, Validation, Test datasets**	**Ready for model training**

The resulting datasets (train, validation, and test) were then used directly for model training and evaluation.

3.2. Model Architecture

The model developed for this project is a fully custom convolutional encoder–decoder network inspired by the U-Net++ architecture. The goal of this design is to generate accurate saliency masks by learning both local and global image features, while preserving important spatial details through skip connections.

The network consists of three main components:
(1) the encoder, **(2) the bottleneck**, and **(3) the decoder with dense skip connections**.

The model architecture used in this project is based on a U-Net++–style encoder–decoder network with dense skip connections.
As shown in Figure 3, the architecture includes:

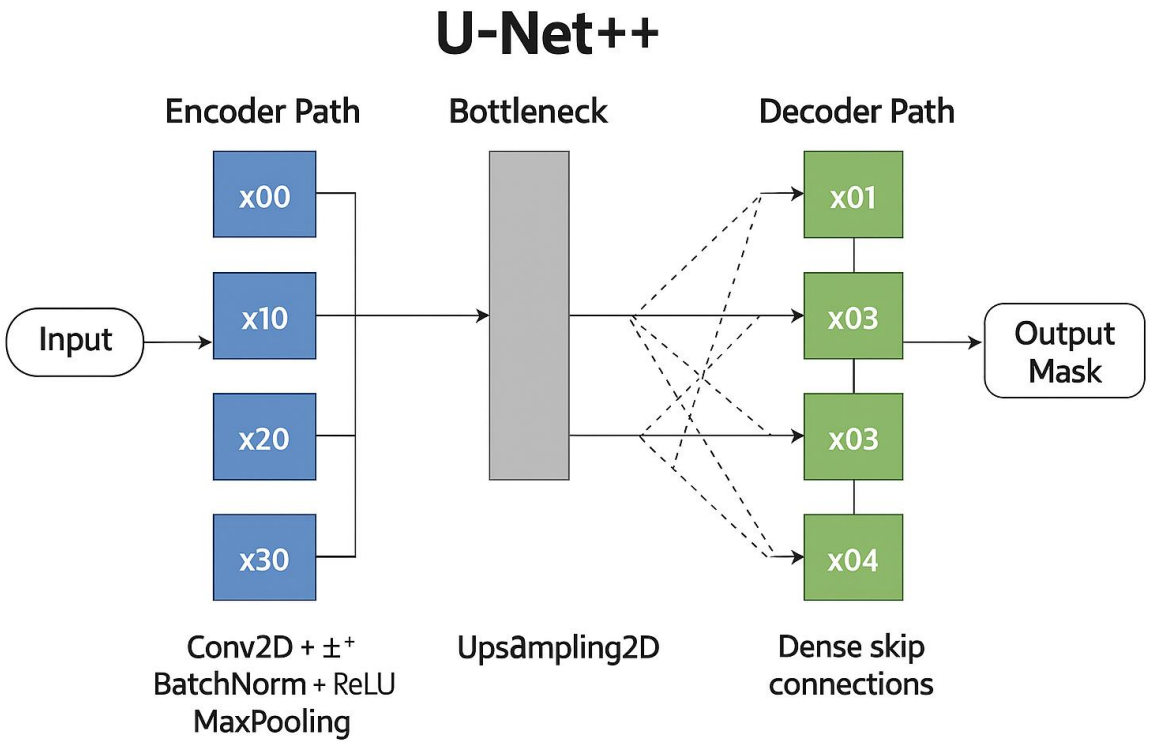

Figure 3: U-Net++ Architecture Used for Salient Object Detection.

3.2.1. Encoder (Feature Extraction)

The encoder path extracts progressively deeper feature representations from the input image.
Each encoder stage contains:

- Two **Conv2D** layers (kernel size 3×3, "same" padding)
- **Batch Normalization**
- **ReLU activation**
- A **MaxPooling** layer for spatial downsampling

This structure allows the model to learn multi-level features, from simple edges to more complex object shapes.

3.2.2. Bottleneck

At the deepest level of the network, the bottleneck captures high-level semantic information needed for identifying the salient regions.
No pooling is applied here — only convolutional blocks — allowing the model to retain as much information as possible before reconstruction.

3.2.3. Decoder

The decoder path reconstructs the spatial dimensions of the image using:

- **UpSampling2D** layers to increase resolution
- **Dense skip connections** (U-Net++ style) that connect multiple encoder layers to multiple decoder layers

These dense connections help the model:

- Preserve fine edges
- Improve gradient flow
- Capture more contextual information

The final output layer is a **1-channel sigmoid activation**, producing a saliency mask with values in the 0–1 range.

3.2.4. Output

The model outputs a grayscale mask, where:

- Values close to **1** represent the salient object
- Values close to **0** represent the background

This mask is used for evaluation and for visualization against ground-truth labels.

3.3. Training Setup

The model was trained for 25 epochs using the ECSSD dataset, with a batch size of 4 and the Adam optimizer (lr = 0.001). The loss function used was Binary Cross-Entropy, and the main evaluation metric during training was IoU. A ModelCheckpoint callback saved the best model based on validation loss.

Training Summary:

Best validation loss: 0.52724 (epoch 18)

Final training loss: 0.2730

Final IoU (train): 0.7795

Final IoU (validation): 0.6060

The training process was stable, and the model achieved consistent segmentation performance without significant overfitting.

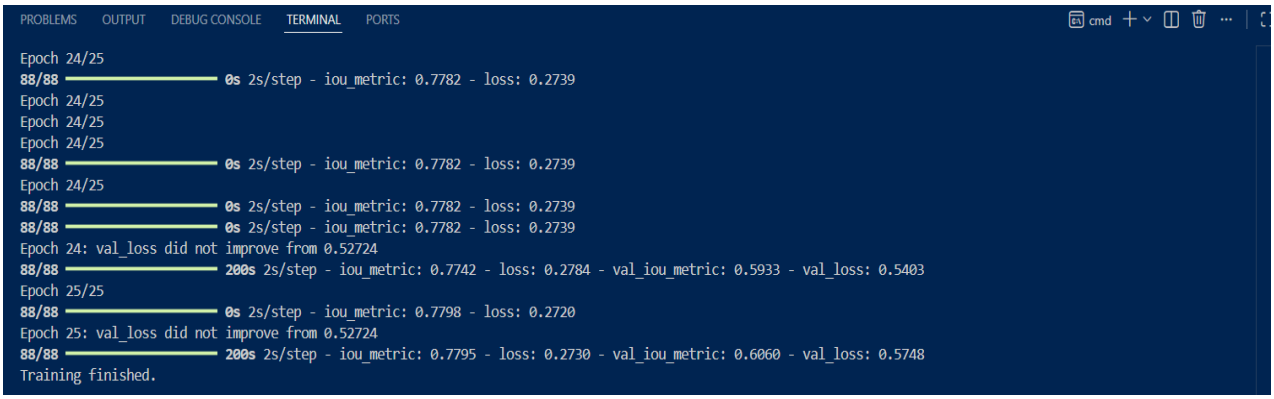

Figure 4: Final training log showing the model's performance across the last epochs.

3.4. Evaluation & Results

After training the model, the final evaluation was performed using the test dataset (15% of ECSSD). The goal of this evaluation is to measure how well the model generalizes to unseen images and how accurately it identifies salient objects.

3.4.1. Evaluation Metrics

The following metrics were used to evaluate the model:

- **Intersection over Union (IoU)**
- **Precision**
- **Recall**
- **F1 Score**

These metrics quantify segmentation accuracy, boundary quality, and foreground–background separation.

Table 2: Qualitative Results

Metric	Score
IoU	0.6944
Precision	0.7071
Recall	0.7446
F1 Score	0.7194

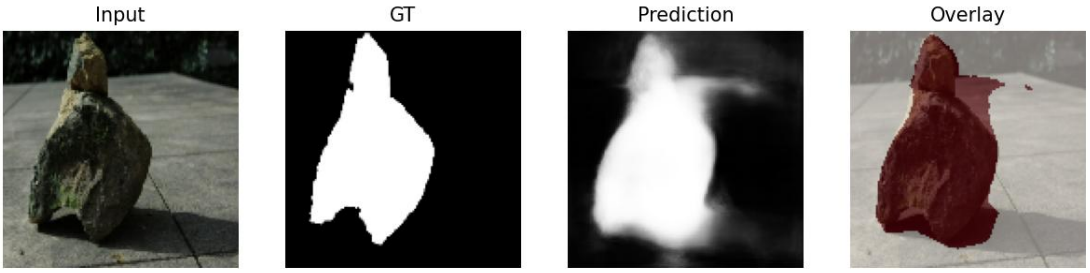

Figure 5: Input, ground-truth mask, predicted mask, and overlay

3.4.2. Summary

The final SOD model demonstrates:

- Good generalization on unseen test images
- Solid performance across standard segmentation metrics
- Clear visual predictions that align with the ground-truth masks

The results validate the effectiveness of the custom U-Net++-style architecture implemented in this project.

4. Conclusion

This project implemented a full end-to-end Salient Object Detection pipeline using a custom U-Net++–style encoder–decoder network. The TensorFlow-based data pipeline enabled efficient preprocessing, augmentation, and batched training.

The model achieved **stable segmentation performance**, confirmed by quantitative metrics on the test set (IoU: 0.69, F1: 0.71). Dense skip connections improved boundary consistency and helped retain fine spatial details in the predicted masks. Qualitative results showed accurate foreground extraction on unseen images.

Overall, the system met its objectives by:

- Designing and training a custom CNN for pixel-level saliency prediction
- Establishing a reproducible preprocessing and training workflow
- Producing reliable masks with clear object–background separation

Future improvements may include larger datasets, attention modules, and pre-trained feature extractors for enhanced accuracy.