U-Net with Conditional Random Fields

Abstract

Pollution in the form of litter, especially plastics, is a global concern. Litter originating from rivers and land frequently accumulates along coastlines, making beaches an important location for recovering the litter, before it enters the ocean or sea.

This project explores the potential of detecting and segmenting small litter objects in coastal environments using the U-Net model architecture with Conditional Random Fields as a post-processing layer.

This project is focused around the new BeachLitterv2022 dataset, because of its prevalence of small litter objects that are far away from the camera, having complex edges/boundaries. The main performance metric used is mIoU.

The results indicate that the U-Net architecture has significant potential for detecting litter in coastal environments. However, the CRF post-processing layer contributed, at best, to slight improvements in the mIoU score.

Contents

1 Introduction

In this research project, the existing U-Net [12] architecture will be comparatively evaluated using performance assessment metrics [8] and will be further improved using tuning techniques. There have been many examples of impressive segmentation using the lightweight U-Net model [1] [4]. The final goal of the project is to obtain segmentation masks that will more efficiently detect the complex boundaries that some small pieces of litter have.

A standard U-Net will be trained on the BeachLitterv2022 dataset [6]. After this, a U-Net with a CRF [5] post-processing layer is going to be trained on the same dataset. The results of these 4 models (binary and multi-class) are going to be compared.

1.1 Research Questions

The questions that this research project aims to answer are:

- What techniques can be used to make the U-Net model struggle less with segmenting fine, complex boundaries or small structures?
- How do these techniques compare with the standard version of the U-Net model?

2 Background

2.1 Datasets Used

The datasets that will be used are: the full size **BeachLitter Dataset v2022** [6] and a smaller size partition of the same datatset. This dataset consists of 3500 beach litter images and their corresponding masks, taken from 2011 to 2019 in Yamagata Prefecture, Japan. This dataset has 8 classes: Artificial Litter, Background, Living, Sea. Natural Litter, Sky, Sand Beach, Non Living. The smaller dataset, used for initial testing, has 111 images and masks.

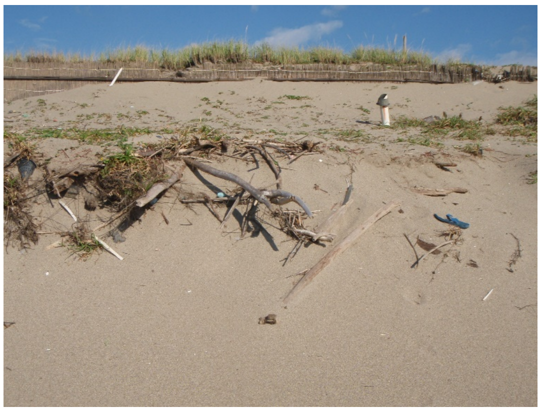
Figure 1: Original image example

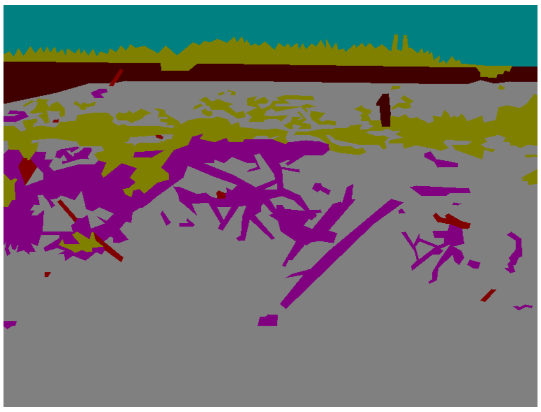
Figure 2: Ground truth mask example

2.2 Related Work

The calculated IoU [8] and DSC score varies across different datasets and architectures, ranging from 73% for TACO using a U-Net [1], up to 79.8% for the improved, Attention-U-Net [11]. Despite this results, the datasets used were not ideal for this task. TACO: Trash Annotations in Context [9] and in TrashNet [15] do not contain many examples of litter that is hard to see or damaged litter. In contrast, the dataset used for this research project is more suitable, having many far-away pieces of litter and some litter with complex edges.

2.3 Models Used

The models used are going to be:

- A standard **binary** segmentation U-Net [12] model.
- A standard **multi-class** segmentation U-Net [12] model.
- The **binary** segmentation U-Net model with an added CRF [16] layer.
- The **Multi-Class** segmentation U-Net model with an added CRF [16] layer.

2.4 Overview of the architecture

This research project focuses on trying to reduce the drawbacks caused by the specific issues mentioned below. The fixes and/or enhancements are also specified below.

The U-Net architecture uses following key concepts:

- The Down-Convolution and Up-Convolution layers
- The Pooling and Concatenation layers
- The Batch Normalization Layers
- The Output layer
- The Activation and Cost Functions

2.4.1 Down-Convolution and Up-Convolution Layers

The Down-Convolution layers are responsible for extracting the features from the images. Each layer is going to have 2 convolution operations. Both the convolution operations are going to have double the filters used at the previous layer, therefore doubling the number of channels of the image. Each filter is going to be 3x3 pixels.

$$\mathbf{f}_{\text{down}}^{(i)} = \text{ReLU}(\mathbf{W}^{(i)} \star \mathbf{x}^{(i-1)} + \mathbf{b}^{(i)})$$

Explanation: $\mathbf{W}^{(i)}$ are the filter parameters at this layer, i, $\mathbf{x}^{(i-1)}$ is the input, vector of $\mathbf{i} * \mathbf{i}$ elements, from the previous layer, and $\mathbf{b}^{(i)}$ is the bias term. $\star$ is the convolution operation.

The Up-Convolutions are responsible for extracting the location of the features from the images. Each layer is going to have one up-convolution operation. This operation is going to have half of the filters from the previous layer, therefore reducing the number of channels by 50%. Each filter is going to be 2x2 pixels. Note the lack of an activation function, this operation is used only for up-scaling.

$$\mathbf{f}_{\text{up}}^{(i)} = \mathbf{W}^{(i)} \star \mathbf{x}^{(i-1)} + \mathbf{b}^{(i)}$$

2.4.2 Pooling and Concatenation Layers

Each max-pooling operation will be the last operation done in every layer, once per layer. A kernel of size 2x2 and a stride of 2 pixels will be used.

$$\mathbf{f}_{\text{pooled}}^{(i)} = Max(\mathbf{W}^{(i)} \star \mathbf{x}^{(i)})$$

The concatenation operation is going to be used right after every up-convolution. This operation is going to concatenate the result from the contrasting path, from the corresponding layer, with the result of the up-convolution. This operation adds data that was lost during the up-sampling phase back into the image being processed.

$$\mathbf{f}_{\text{concat}}^{(i)} = \text{Concat}(\mathbf{f}_{\text{down}}^{(i)}, \mathbf{f}_{\text{up}}^{(i)})$$

2.4.3 Batch Normalization Layers

Batch normalization [14] operations are going to be used after each of the first down-convolutions in each layer. This is used to normalize the activations of the neurons during training for faster convergence and better gradient flow. After this scaling and shifting are performed on these values, using **learnable** parameters.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Explanation: $\hat{x}_i$ is the normalized feature, x_i is the input feature, μ_B is the batch mean, σ_B^2 is the batch variance, and ϵ is a small constant for numerical stability.

$$y_i = \gamma \hat{x}_i + \beta$$

Explanation: y_i is the scaled and shifted output, γ is the learnable scaling parameter, and β is the learnable shifting parameter.

$$\mathbf{y} = \mathrm{BN}(\mathbf{x}) = \gamma \tfrac{\mathbf{x}-\mu_B}{\sqrt{\sigma_B^2+\epsilon}} + \beta$$

Explanation: $\mathbf{y}$ represents the final output of batch normalization applied to $\mathbf{x}$, with μ_B and σ_B^2 calculated for every batch.

2.4.4 CRF Layer

$$P(\mathbf{y}|\mathbf{x}) = \tfrac{1}{Z(\mathbf{x})} \exp\left(\textstyle\sum_{t=1}^{T} \psi_t(y_t, \mathbf{x}) + \sum_{t=1}^{T-1} \psi_{t,t+1}(y_t, y_{t+1}, \mathbf{x})\right) \quad [7]$$

Explanation: $P(\mathbf{y}|\mathbf{x})$ is the conditional probability of the label sequence $\mathbf{y}$ given the input $\mathbf{x}$

$$\psi_t(y_t, \mathbf{x}) = \mathbf{w}_t^\top \phi_t(y_t, \mathbf{x})$$

Explanation: $\psi_t(y_t, \mathbf{x})$ is the local feature function, it evaluates how well a particular label y matches the input features at position t.

$$\psi_{t,t+1}(y_t, y_{t+1}) = \mathbf{A}_{y_t, y_{t+1}}$$

Explanation: $\psi_{t,t+1}(y_t, y_{t+1})$ is the transition function and it evaluates the compatibility between adjacent labels in the sequence. This is crucial for structured prediction tasks where neighboring labels often depend on each other.

$$Z(\mathbf{x}) = \textstyle\sum_{\mathbf{y}} \exp\left(\sum_{t=1}^{T} \psi_t(y_t, \mathbf{x}) + \sum_{t=1}^{T-1} \psi_{t,t+1}(y_t, y_{t+1}, \mathbf{x})\right)$$

Explanation: It ensures that the sum of all probabilities over all possible label sequences equals 1. This makes the CRF a valid probability distribution.

2.4.5 Output Layer

This is a simple convolution operation, with 1x1 filters, with a number of filters equal to the number of classes expected. There are two scenarios: binary and multi-class segmentation. For the binary segmentation, a sigmoid activation function is used. For the multi-class segmentation, the softmax function is used.

$$\textbf{prediction} = \text{Sigmoid}(\mathbf{W}_{\text{out}} \cdot \mathbf{x}_{\text{final}} + \mathbf{b}_{\text{out}})$$

$$\textbf{prediction} = \text{Softmax}(\mathbf{W}_{\text{out}} \cdot \mathbf{x}_{\text{final}} + \mathbf{b}_{\text{out}})$$

Explanation: $\mathbf{W}^{(out)}$ are the filter parameters from the last layer, i, $\mathbf{x}^{(final)}$ is the input from the final/last layer and $\mathbf{b}^{(out)}$ is the bias term from the last layer. $\cdot$ is a scalar product.

2.4.6 Cost Functions

Likewise, two different cost functions can be used here. **B**inary **C**ross **E**ntropy for binary segmentation and **C**ategorical **C**ross **E**ntropy for multi-class segmentation.

$$\mathcal{L}_{\mathbf{BCE}} = -\frac{1}{N} \sum_i \left[y_{\text{true}} \log(y_i) + (1 - y_{\text{true}}) \log(1 - y_i) \right]$$

$$\mathcal{L}_{\mathbf{CCE}} = -\frac{1}{N} \sum_{i,c} y_{\text{true},i,c} \log(y_{i,c})$$

Explanation: **N** is the number of samples, y_{true} is the ground-truth mask.

2.5 Issues

Some issues with the standard U-Net model are:

- The **resolution loss** that the up-sampling [13] layers fail to fully account for.
- The **blurring** of the predictions at the boundaries, caused by the lack of mechanisms to highlight edge regions.

2.6 Proposed Fixes and Enhancements

- Using Conditional Random Fields (CRFs) in a post-processing layer.

2.7 Performance Metrics

Overlap metrics:

- IoU (intersection over union)

The proposed model's results are going to be compared to the results found in other papers, highlighted down below.

Model	IoU Score	DataSet
U-Net [16]	78%	TrashNet
U-Net CRF [16]	79.8%	TrashNet
U-Net [1]	73.88%	TACO
AA U-Net [11]	72.3%	TACO

Table 1 Comparison Metrics

3 Methodology

The following section introduces the approach used in this research project, including specific information about the issues and enhancements proposed, structure of the dataset, image pre-processing steps and the individual techniques used for every model.

Reference: The code for the following research project is hosted on github here.

3.1 Data collection and structuring

The dataset is split into an 80:10:10 ratio for the training, validation and the testing sets. 5-fold cross validation [3] is applied on the combined training/validation set later on.

Both the multi-class and binary datasets contain 3500 images. Upon closer inspection, a significant amount (around 9%) of the images either: don't contain litter at all or are of very low resolution. This is significant since, for the images not containing litter, the model is not being trained to recognize litter. For images that are of low resolution, when re-scaling the images to 512x512 they are actually being up-sampled. This up-sampling turns out to be either very helpful, some of these images have a very high IoU score, or detrimental, most of them have really low scores.

For the binary masks, each pixel has a value of 0 (black) for litter and 1 (white) for anything else.

For the multi-class masks, each pixel has a value from 0 to 7, one value corresponds to each class. The classes (labels) and their corresponding pixel values, are:

- 0 -> Artificial Litter
- 1 -> Background
- 2 -> Living
- 3 -> Sea

- 4 -> Natural Litter
- 5 -> Sky
- 6 -> Sand Beach
- 7 -> Non Living

3.2 Image Pre-processing

For **binary** segmentation:

- New binary masks are created, all being scaled to 512x512 pixels, where the pixel value of 0 (black) is litter and 1 (white) is anything else
- The input image (rgb) is kept at 3 channels, masks have 1 channel each
- The input image is **scaled** to 512x512

For **multi-class** segmentation:

- The masks that are used are scaled to have values in the range $[0, \mathbf{nrClasses} - 1]$ [10]
- The input image (rgb) is kept at 3 channels, masks have 1 channel each
- The input image is **scaled** to 512x512

3.3 Standard U-Net

The resulting images are going to be processed by the binary U-Net model mentioned above.

3.3.1 Hyper-Parameters

- The model has 18 convolution operations, 4 down-sample and 4 up-sample operations and one final 1x1 convolution, for a total of 4 layers.
- Learning-rate: 0.0001

- Batch size: 2
- Epochs: 20

3.3.2 Optimizer

The optimizer used was Adam. [2]

4 Results

These are the resulting scores from training a the U-Net models on the whole BeachLitterv2022 dataset.

The multi-class results are slightly better than binary results, this likely has to do with the fact that some of the images did not contain a significant amount of litter, so the background class scores are very high.

Model	IoU Score	DataSet
U-Net [16]	78%	TrashNet
U-Net CRF [16]	79.8%	TrashNet
U-Net [1]	73.88%	TACO
AA U-Net [11]	72.3%	TACO
U-Net Binary*	71.3%	BeachLitterv2022
U-Net Multi-Class*	75.6%	BeachLitterv2022
U-Net Binary CRF*	71%	BeachLitterv2022
U-Net Multi-Class CRF*	75.8%	BeachLitterv2022

Table 2 Final Results

Both the binary and multi-class models, being trained on only 20 epochs, have not yet converged. Training for closer to 100 epochs is needed for better results.

The results fluctuate a lot, with the mean IoU score not revealing the whole picture. Here are some very good results (>85% IoU score).

*These are the proposed models

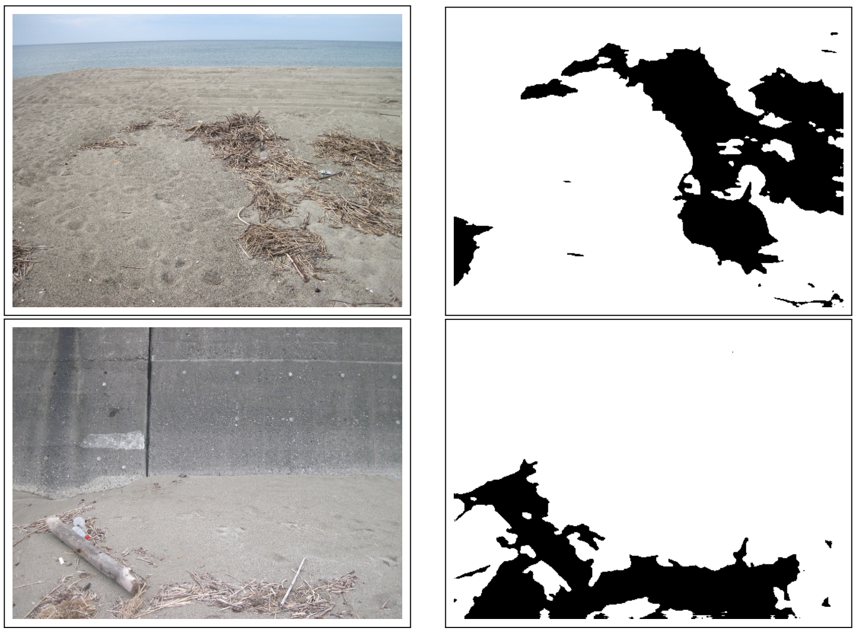
Figure 3 Good binary predictions.

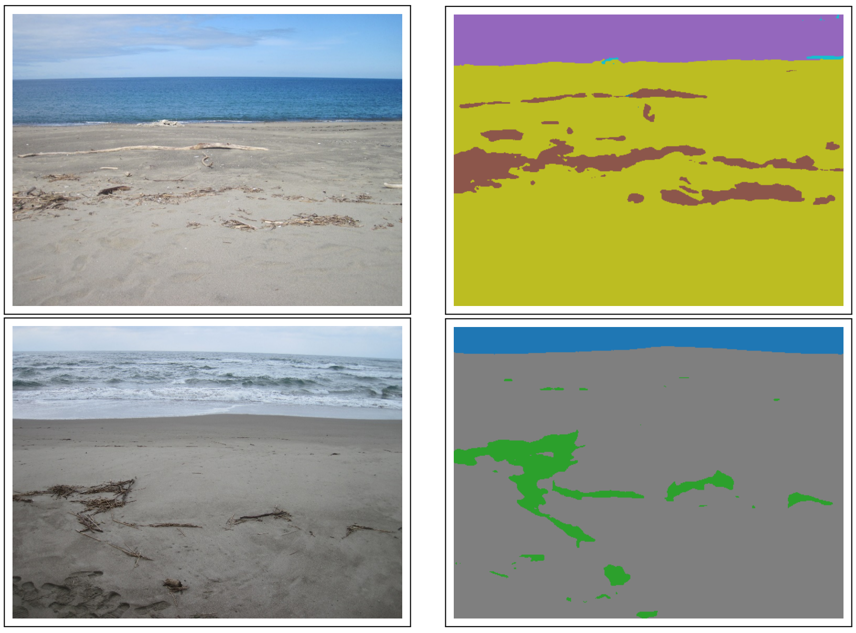
Figure 4 Good multi-class predictions.

5 Future Work

More research is needed, as these method can be easily improved. If I were to undertake this project again, I would place greater emphasis on the dataset preparation, and I would start running model experiments earlier on in the project due to the tight resource and time requirements.

Dataset future work:

- The dataset can be merged with others that have only 1 litter class, such as: TACO or TrashNet
- Eliminating the previously discussed, "bad" images
- Having better, more accurate, ground truth masks

Training process future work:

- The training needs to be more extensive, at over 100 epochs
- The use of transfer learning will only improve the model
- Because of batch normalization, using a batch size bigger than 64 would be advised

6 Conclusions

This research project was conducted by reviewing previous papers and articles in the field of semantic segmentation and litter detection. Firstly, some relevant datasets were reviewed. These datasets vary from single-class (with only one "litter" category) to multi-class (with up to 60 categories and 28 super-categories), including both bounding annotations and pixel-wise instance masks, and containing between 772 and 3,722 images. After reviewing them, the newest dataset, BeachLitterv2022 was chosen for the experiments.

For the evaluation of the segmentation algorithms, both binary and multi-class variants of the U-Net architecture were used. Furthermore, a CRF post-processing layer was added in an attempt to get cleaner segmentation for pieces of litter that were either far away from the camera or had complex boundaries.

Some key challenges encountered were the lack of computational resources, with both of the models not fully converging.

The experiments yielded promising results, with the binary U-Net model getting a 71.3 mean IoU score and 71 after the CRF post-processing. The multi-class result are better, due to the other, more prominent, classes such as: background, sky getting generally good scores.

At a high level, these methods can be improved significantly by: merging more specialized datasets together, eliminating some "bad" images from the BeachLitterv2022 dataset, using transfer learning and training the models more extensively.

References

[1] D. Akkoyun, “Identification and detection of beach litter using computer vision techniques,” 2024. [Online]. Access: https://uu.diva-portal.org/smash/get/diva2:1899954/FULLTEXT01.pdf

[2] J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Access: https://arxiv.org/abs/1412.6980

[3] J. Brownlee, “A gentle introduction to k-fold cross validation,” 2023. [Online]. Access: https://machinelearningmastery.com/k-fold-cross-validation/

[4] D. Cherney, “Beach litter computer vision via image segmentation,” 2022. [Online]. Access: https://github.com/davidcherney/trashnet

[5] P. W. Code, “Crf explained,” 2010. [Online]. Access: https://paperswithcode.com/method/crf

[6] S. Daisuke, “The beachlitter dataset v2022,” 2022. [Online]. Access: https://www.seanoe.org/data/00743/85472/

[7] A. McCallum, “An introduction to conditional random fields,” 2012. [Online]. Access: https://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

[8] R. Padilla, “A survey on performance metrics for object-detection algorithms,” 2020. [Online]. Access: https://ieeexplore.ieee.org/document/9145130

[9] P. F. Proença, “Taco: Trash annotations in context for litter detection,” 2020. [Online]. Access: https://arxiv.org/abs/2003.06975

[10] Pytorch, “Pytorch cross entropy loss,” 2023. [Online]. Access: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html#torch.nn.CrossEntropyLoss

[11] K. Rajamani, “Attention-augmented u-net (aa-u-net) for semantic segmentation,” 2022. [Online]. Access: https://link.springer.com/article/10.1007/s11760-022-02302-3

[12] O. Ronneberger, “U-net: Convolutional networks for biomedical image segmentation,” 2015. [Online]. Access: https://arxiv.org/pdf/1505.04597

[13] N. Shibuya, “Up-sampling with transposed convolution,” 2017. [Online]. Access: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

[14] C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015. [Online]. Access: https://arxiv.org/abs/1502.03167

[15] G. Thung, “Trashnet,” 2017. [Online]. Access: https://github.com/garythung/trashnet

[16] E. Yi, “U-net with a crf-rnn layer,” 2019. [Online]. Access: https://github.com/EsmeYi/UNet-CRF-RNN?tab=readme-ov-file