Total Variation Loss for Compact Objects' Segmentation Applications to Satellite Images

Giacomo Fraccaroli
Stanford University
giacomof@stanford.edu

Abstract

This paper proposes a custom loss function to improve deep learning performance in image segmentation tasks involving 'compact' objects—those with sharp boundaries and spatial continuity. To promote spatial coherence in predicted masks, I introduce total variation loss as a regularization penalty during training. Applied to a U-Net architecture on the AICrowd Mapping Challenge dataset, this method improves the Intersection-over-Union by 2.2*pp on a held-out test set. This method also showcases better results in classifying noisy synthetic data, with a margin of up to* 3.6*pp on a held-out test set.*

1. Introduction

In computer vision, *semantic segmentation* refers to the process of assigning a class label to each pixel in an image. In the binary case, this involves generating a probability mask that associates each pixel with a value representing the likelihood of it belonging to a particular class or not. For an image I, this means that:

$$\forall (i,j) \in I, \quad \exists \Pr(y_{i,j} = c) = p_{i,j} \in [0,1] \qquad (1)$$

where (i,j) denotes a pixel in the image I, $y_{i,j}$ is the random variable representing the class label at pixel (i,j), and $p_{i,j}$ is the associated probability that this pixel belongs to class c.

In this paper, I consider a U-Net architecture [8] which I train on classifying pixels in an image as either part of a roof or not. This has an encoder/decoder structure such that multiple, connected convolutional layers can process both high- and low-resolution information about each image. A U-Net—and other state-of-the-art classifiers, *e.g.*, SegNet [10]—are typically agnostic to the shape of the object. They rely on the weights of the model, trained on the spatial composition of the training masks, to detect similar patterns in new images. However, when dealing with *compact objects*,[1] it is clear *ex ante* that bigger clusters of pixels classified as such objects are likelier to be correct predictions than smaller disjoint regions (possibly due to overfitting).

To account for this, I introduce a novel objective function that integrates this notion that disjoint regions in the probability mask should be penalized—distinct from the penalty regarding whether they are accurate or not. When training a U-Net on the AICrowd dataset to predict roof masks with this adjustment, IoU performance increases by 2.2pp on the test set, and so does robustness to synthetically manipulated test images. This is driven by more compact and coherent masks being predicted since small, disjoint roof regions are strongly penalized.

The rest of the paper is structured as follows. Section 2 provides a brief overview of related work in computer vision and image processing. Section 3 introduces the data I will be using to develop the model architectures for roof segmentation. Section 4 and 5 describe the models I develop, namely a vanilla U-Net architecture [8], one with softmax regularization [4], and—my original contribution—one with total variation regularization by changing the objective function. Section 6 presents the results with regards to robustness to noisy images, Section 7 presents a method to select the relative weight of total variation in the objective as a learnable parameter, and Section 8 concludes.

2. Related Work

2.1. Past Literature

The intuition that objects appear as bounded, discrete regions in images has long been exploited in image processing through diffusion-based techniques that denoise or regularize an image while preserving sharp edges. A prominent example is total variation (TV) regularization [9]. Given an

[1] I define as compact objects all classes which tend to present themselves as coherent, bounded regions in images, such as satellite snapshots of rooftops.

image $I \in \mathbb{R}^{H \times W}$, its total variation is defined as the sum of differences between neighboring pixels:

$$\mathcal{L}_{\text{TV}}(I) = \sum_{i,j} |I_{i,j} - I_{i+1,j}| + |I_{i,j} - I_{i,j+1}|, \quad (2)$$

where (i, j) indexes a pixel, and the sum is over all horizontal and vertical neighboring pixel pairs' real float values. This penalty encourages local smoothness across the image while preserving sharp transitions at object boundaries.

To minimize (2) for an image I while preserving its general characteristics, *Chambolle's algorithm* [2] provides a computationally efficient solution to the following problem:

$$\underset{I_{\text{TV}}}{\text{minimize}} \quad \frac{1}{2\lambda}\|I_{\text{TV}} - I\|^2 + \mathcal{L}_{\text{TV}}(I_{\text{TV}}), \quad (3)$$

where I_{TV} is the denoised image and $\lambda \in \mathbb{R}_+$ a weight on the TV loss term—$\mathcal{L}_{\text{TV}}$—relative to maintaining the features of the ground truth image I.

[4] incorporate this notion that objects appear as compact clusters of pixels by customizing the *softmax activation layer*. In particular, they redefine it by adding a TV loss regularization term. The standard softmax layer for binary classification solves the following optimization problem:

$$\underset{\mathcal{A}}{\text{minimize}} \quad -\langle \mathcal{A}, \mathbf{o} \rangle + \langle \mathcal{A}, \log \mathcal{A} \rangle \quad (4)$$

$$\text{subject to} \quad \sum_{c=1}^{2} \mathcal{A}_{ci} = 1, \quad \forall i = 1, \ldots, HW \quad (5)$$

where $\mathcal{A} \in [0, 1]^{2 \times HW}$ is the prediction mask for classes $c = \{1, 2\}$ across all pixels in an image of size $H \times W$. The model's output, $\mathbf{o} \in \mathbb{R}^{2 \times HW}$, is fed into the softmax, which converts it into per-pixel probabilities. This leads to the well-known closed-form solution:

$$\hat{\mathcal{A}}_c^* = \frac{\exp(\mathbf{o}_c)}{\sum_{c=1}^{2} \exp(\mathbf{o}_c)} \in [0, 1]^{1 \times HW} \quad (6)$$

i.e., the softmax function applied pointwise across the class logits. [4] add a penalty for non-smooth output masks by modifying the activation layer to instead solve:

$$\underset{\mathcal{A}}{\text{minimize}} \quad -\langle \mathcal{A}, \mathbf{o} \rangle + \langle \mathcal{A}, \log \mathcal{A} \rangle + \lambda \mathcal{L}_{\text{TV}}(\mathcal{A}) \quad (7)$$

$$\text{subject to} \quad \sum_{c=1}^{2} \mathcal{A}_{ci} = 1, \quad \forall i = 1, \ldots, HW \quad (8)$$

where $\lambda \in \mathbb{R}_+$ is a regularization hyperparameter, and $\mathcal{L}_{\text{TV}}$ is defined as at (2). This modified activation (which enters the backpropagation) encourages spatial coherence by embedding regularization within the final layer of the network.

Figure 1: Predicted Segmentation Masks with and without TV Denoising

(a) No Denoising

0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	1	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(b) With TV Denoising

0	0	.2	0	0	0	0	0	0	0
0	0	0	0	0	0	.2	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	.8	1	1	1	.8	0	0
0	.2	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	.8	1	1	1	.8	0	0
0	0	0	0	0	0	0	0	0	.2
0	.2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Notes: The figure displays the predicted probabilities that each pixel is part of a given class in a 10×10 image. The green contours represents the ground truth mask. Panel (b) manipulates the output of Panel (a) by optimizing (3) with $\lambda = 1/4$ via L-BFGS.

2.2. Original Contribution

In this work, I incorporate this intuition—that objects appear as compact clusters of pixels—directly into the training objective of my segmentation model, as opposed to the activation layer. Let $f_\theta(I)$ be a network that maps an image $I \in \mathbb{R}^{H \times W \times C}$ to a predicted probability mask $\hat{P} \in [0, 1]^{H \times W}$ for a binary class of interest, I define its total loss as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}}(\hat{P}, P) + \lambda \mathcal{L}_{\text{TV}}(\hat{P}), \quad (9)$$

where $\mathcal{L}_{\text{BCE}}$ is the standard binary cross-entropy loss,[2] P is the ground-truth mask, and $\mathcal{L}_{\text{TV}}$ acts as a spatial regularizer with weight $\lambda \in \mathbb{R}_+$. Indeed, the added TV term penalizes local irregularities in the predicted mask, promoting compact, coherent segmentations. Figure 1 provides an example where TV denoising greatly benefits predictions in the presence of outliers.

Unlike [4], the penalty term is added directly to the loss function of the network as opposed to any activation layer(s). Thus, it solely affects models' weights via backpropagation. This approach has two main advantages:

1. **Simplicity:** It requires no modification to the architecture or inference procedure, unlike variational activations that involve iterative solvers.
2. **Efficiency:** Adding the TV term as part of the loss avoids the computational overhead of solving optimization problems inside the forward pass.

3. Data

The AICrowd Mapping Challenge offers circa $400,000$ labeled, colored images with a resolution of 300×300 pix-

[2] Binary cross-entropy loss is defined as:
$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$,
where N is the number of samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability.

Figure 2: AICrowd Image downsampling, with Ground Truth Mask

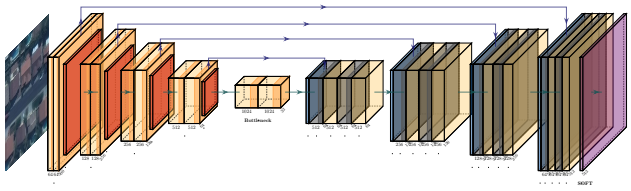
(a) 300 × 300 Resolution (b) 100 × 100 Resolution

els to train models for roof segmentation. For compuational constraints, I limit my analysis to $5,000$ of such images,[3] which I downsample at a resolution of 100×100 pixels. Figure 2 presents an example of the original resolution and the downsampled one, where the latter clearly contains enough visual information to detect the boundaries of the roofs. As standard in the literature, I split the sample into train, validation, and test sets in proportions of 80%, 10%, and 10%, respectively.

4. Model Architectures

Figure 3 showcases the baseline architecture of my U-Net. This entails a symmetric encoder/decoder structure with residual connections, like [8]. The encoder consists of two downsampling blocks, each containing two 3×3 convolutional layers followed by batch normalization and a 2×2 max pooling operation. The number of filters doubles with each downsampling step, starting from 64. The bottleneck block includes two convolutional layers with 256 filters. The decoder mirrors the encoder with two upsampling blocks, each beginning with a 2×2 transposed convolution (for upsampling), followed by concatenation with the corresponding encoder feature maps. Thereafter, two 3×3 convolutional layers with batch normalization are applied. The network ends with a 1×1 convolution and sigmoid activation to produce the final binary segmentation mask.

Figure 3: U-Net Architecture with Residualized Connections



[3] Specifically, I draw them randomly from the AICrowd validation images' dataset given known issues with the training data [1].

From this, I build three variations:

i) **U-Net + TV**: Applies TV denoising to the U-Net predictions post-training, solving an analogous to (3), where I is the predicted mask instead.

ii) **RU-Net**: Adds a *softmax layer* solving (7).

iii) **LU-Net**: Trains the network with (9) as the total loss, as opposed to $\mathcal{L}_{\text{BCE}}$ only.

The performance of each model is assessed as per the average Intersection-over-Union (IoU) they achieve. This evaluates the overlap between the predicted segmentation and the ground truth:

$$\text{IoU} = \frac{|\text{Prediction} \cap \text{Ground Truth}|}{|\text{Prediction} \cup \text{Ground Truth}|} \tag{10}$$

Table 1: IoU across models and datasets

		Intersection-over-Union		
Model	λ^*	Train	Validation	Test
U-Net	–	77.16%	61.63%	61.53%
U-Net + TV	0.09	77.26%	61.73%	61.69%
RU-Net	10^{-6}	75.63%	62.97%	63.47%
LU-Net	10^{-6}	76.90%	63.09%	63.72%

Notes: The table displays the IoU performance of 4 different models on the AICrowd dataset for roof segmentation. λ^* is the optimal TV loss weight, determined by the performance on the validation set.

Figure 4: Prediction Masks on a Test Image

(a) Original Image with Mask (b) U-Net Prediction

(c) RU-Net Prediction (d) LU-Net Prediction

Notes: The figure showcases the predictions on a test image depending on the model employed, with the specifications in Table 1. Panels (a), (b), and (c) achieve, respectively, an IoU of 48%, 68%, and 72%.

$$= \frac{TP}{TP + FP + FN} \tag{11}$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively. I favor it over other metrics because of the high level of class imbalances in the AICrowd data.[4]

Table 1 presents the results across all four models when λ is picked to maximize IoU performance in the validation set, as explained in detail in Section 5.

Figure 4 displays the predictions on a test image where *LU-Net* and *RU-Net* vastly outperform the benchmark model by over 20pp on the IoU metric. The U-Net model performs poorly (48% IoU), as it predicts a mask filled with roofs composed of few, scattered pixels that do not match the original mask (outlined in blue) of Panel (a). The *RU-Net* and *LU-Net*, on the other hand, achieve an average IoU of 70% by providing more concentrated predictions—which better match the ground truth.

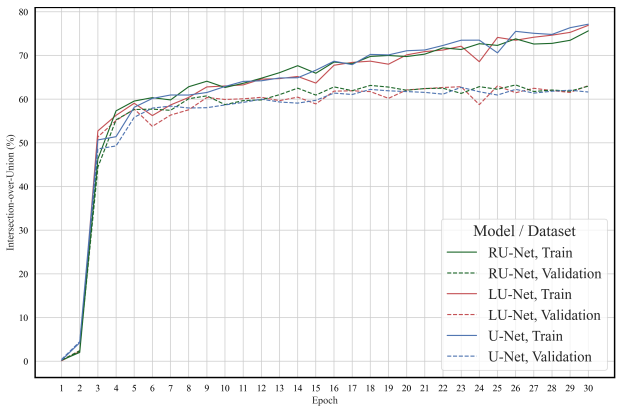
5. Hyperparameters Tuning

To compare models, I set a fixed learning rate of 5×10^{-5} with an Adam optimizer [5] across 30 epochs and a batch size of 16 images. Given the (relative) shallowness of my overall network, all three models converge quickly with rather stable validation performance after the first 20 epochs. Figure 5 displays the training and validation IoU across epochs for all three architectures.

The remaining hyperparameter for my models is λ, which controls the relative weight of the TV loss in either the post-processing (*U-Net + TV*), the softmax layer (*RU-Net*), or the model's objective (*LU-Net*). I choose λ from a predefined set Λ by maximizing the validation IoU:

$$\lambda^* = \arg\max_{\lambda \in \Lambda} \text{IoU}_{\text{val}}(\lambda) \tag{12}$$

Figure 5: IoU Performance by Epoch, Dataset, and Model



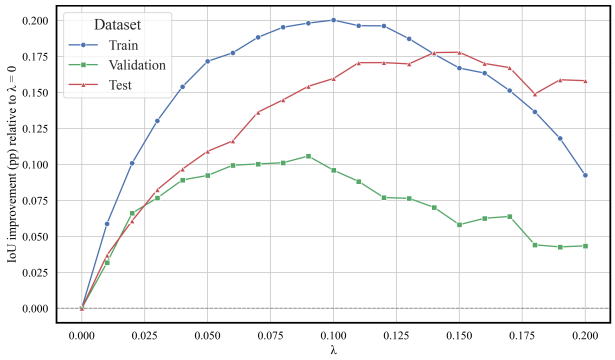
[4] In the training data, 77.3% of the pixels are not roofs – hence, a naive model classifying everything as not-a-roof would achieve high accuracy.

That is, λ^* is the value of λ yielding the highest Intersection-over-Union on the validation set.

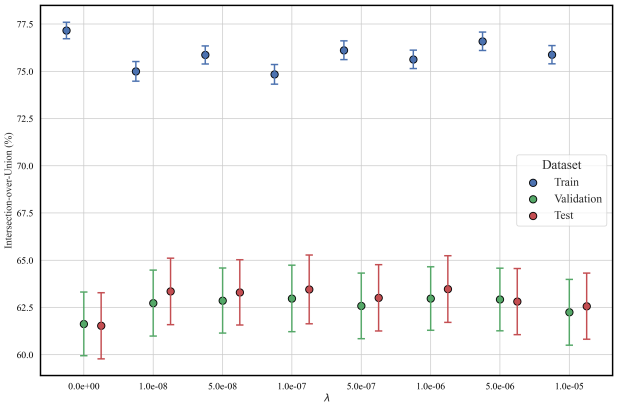
U-Net + TV When applying denoising to the predicted

Figure 6: Selecting λ_1^*, λ_2^*, and λ_3^*

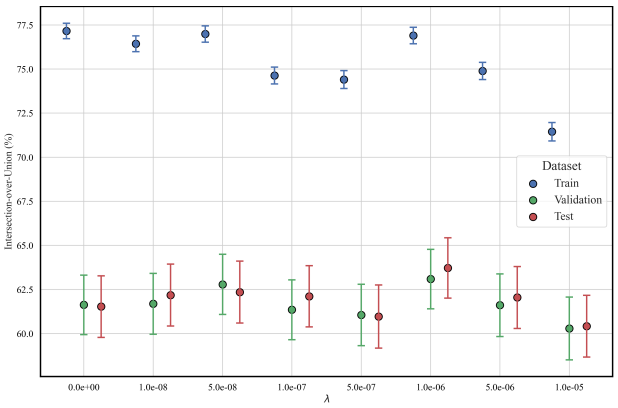
(a) U-Net + TV



(b) RU-Net



(c) LU-Net



Notes: The three panels display the train, validation, and test IoU performance for three different variations of the model. Panel (a) applies post-processing to the vanilla U-Net predictions for all $\lambda_1 \in \Lambda_1$. Panel (b) reports the results at the final epoch of the *RU-Net* architecture for all $\lambda \in \Lambda_2$. Panel (c) displays analogous results for the *LU-Net* architecture for all $\lambda \in \Lambda_3$

mask by solving (3), I sample from:

$$\Lambda_1 = \{0.01, 0.02, \ldots, 0.20\} \tag{13}$$

and select $\lambda_1^* = 0.09$ as per (12). Figure 6a displays the IoU performance given $\lambda \in \Lambda_1$ for the training, validation, and test sets separately. The gains from this architecture are negligible relative to the vanilla U-Net.

RU-Net For this architecture too, the choice of λ_2^* is data-driven. I sample from:

$$\Lambda_2 = \left\{10^{-8}, 5 \times 10^{-8}, 10^{-7}, \cdots, 10^{-5}\right\}$$

and pick $\lambda_2^* = 10^{-6}$ according to (12). To solve (7), I apply the same iterative algorithm as [4], with one single iteration at the final softmax layer. In fact, I slightly tweak the architecture of the U-Net in Figure 3 to have a softmax layer rather than a sigmoid function for each pixel at the end.

LU-Net To select λ_3^*, I sample from:

$$\Lambda_3 = \Lambda_2$$

and pick $\lambda_3^* = 10^{-6}$ according to (12).

Figures 6b and 6c display the average IoU performance on the training, validation, and test datasets of the *RU-Net* and *LU-Net* architectures as λ_2 and λ_3 are sampled from Λ_2 and Λ_3, respectively.

6. Robustness to Noise

The primary application of TV denoising is to remove noise from corrupted images. In this section, I demonstrate that incorporating a TV term into the loss function of my architecture improves its robustness to noise. Specifically, it outperforms the standard U-Net architecture when evaluated on synthetically corrupted data, where noise is added to the RGB values from a mean-zero Gaussian distribution.

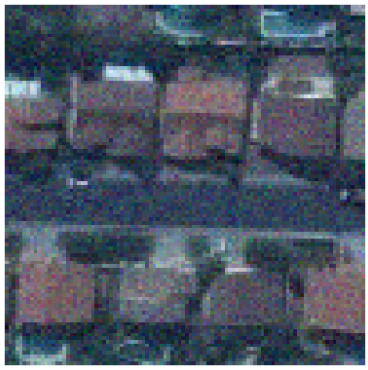
The noise injection process is defined as follows for the test data:

$$\tilde{v}_{i,j,c} = v_{i,j,c} + \varepsilon_{i,j,c}, \quad \varepsilon_{i,j,c} \sim \mathcal{N}\left(0, \sigma^2\right), \tag{14}$$

where $v_{i,j,c} \in [0, 1]$ denotes the original pixel value at (i, j) for channel c, and $\varepsilon_{i,j,c}$ represents Gaussian noise with zero mean and variance σ^2. Figure 7 displays an image in the AICrowd dataset when exposed to severe mean-zero Gaussian noise.

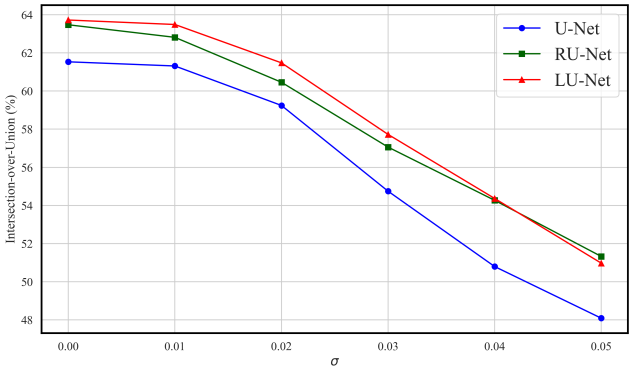
The *RU-Net* and *LU-Net* architectures perform better on a manipulated test set with varying levels of noise—with a margin of up to 3.6pp on the held-out test set (when $\sigma = 0.04$). Figure 8 displays the test IoU across different noise levels.

Figure 7: AICrowd Image corrupted by Gaussian Noise with $\sigma = 0.05$



Notes: The figure displays the corrupted version of Figure 2b when injected with mean-zero Gaussian noise as per (14), with standard deviation $\sigma = 0.05$.

Figure 8: Robustness to Gaussian Noise by Model



7. Gradient-Based Hyperparameter Selection

An alternative to a sweep of values for $\lambda_3 \in \Lambda_3$ is that of gradient-based hyperparameter selection [6, 7, 3]. That is, to consider λ in the model's objective—(9)—as a learnable parameter via gradient descent.[5]

In order to estimate the gradient step for λ in the *LU-Net* architecture, consider the problem as a *bilevel optimization* where the U-Net parameters θ are set to solve:

$$\theta^*(\lambda) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(\theta, \lambda) \tag{15}$$

$$= \arg\min_{\theta} \mathcal{L}_{\text{BCE}}(\theta) + \lambda \mathcal{L}_{\text{TV}}(\theta). \tag{16}$$

where $\mathcal{L}_{\text{train}}$ is defined as at (9) for the training data.

The relative weight λ is, on the other hand, set to minimize validation loss:[6]

$$\lambda^*(\theta) = \arg\min_{\lambda} \mathcal{L}_{\text{val}}(\theta^*(\lambda)) \tag{17}$$

[5] For the *RU-Net* model, [4] provides a derivation of the update rule for λ via the training loss associated with the softmax layer.

[6] Trivially, since λ is a regularization term, its performance must be evaluated on a held-out set and not on the training loss directly.

where $\mathcal{L}_{\text{val}}$ is the binary cross-entropy loss on the validation data.

As per the chain rule:

$$\frac{d\mathcal{L}_{\text{val}}}{d\lambda} = \underbrace{\frac{\partial \mathcal{L}_{\text{val}}}{\partial \lambda}}_{\text{direct}} + \underbrace{\frac{\partial \mathcal{L}_{\text{val}}}{\partial \theta^*}\frac{d\theta^*}{d\lambda}}_{\text{indirect}} \tag{18}$$

The direct gradient is zero ($\mathcal{L}_{\text{val}}$ does not explicitly depend on λ), so:

$$\frac{d\mathcal{L}_{\text{val}}}{d\lambda} = \frac{\partial \mathcal{L}_{\text{val}}}{\partial \theta^*}\frac{d\theta^*}{d\lambda}. \tag{19}$$

Now, consider one gradient descent step for θ with learning rate η_θ:

$$\theta'(\lambda) = \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \lambda) \tag{20}$$

Assuming $\mathcal{L}_{\text{train}}$ is twice continuously differentiable with respect to both θ and λ, by Schwarz's theorem:

$$\frac{d\theta'}{d\lambda} = -\eta_\theta \frac{\partial}{\partial \lambda}\left[\nabla_\theta \mathcal{L}_{\text{train}}\right] \tag{21}$$

$$= -\eta_\theta \nabla_\theta \left(\frac{\partial \mathcal{L}_{\text{train}}}{\partial \lambda}\right) \tag{22}$$

$$= -\eta_\theta \nabla_\theta \mathcal{L}_{\text{TV}} \tag{23}$$

Approximating $\theta^*(\lambda)$ with $\theta'(\lambda)$, the gradient of the validation loss with respect to λ writes:

$$\frac{d\mathcal{L}_{\text{val}}}{d\lambda} \approx \frac{\partial \mathcal{L}_{\text{val}}}{\partial \theta^*}\left(-\eta_\theta \nabla_\theta \mathcal{L}_{\text{TV}}\right) \tag{24}$$

$$\approx -\eta_\theta \left\langle \underbrace{\nabla_\theta \mathcal{L}_{\text{val}}}_{\text{validation BCE gradient}}, \underbrace{\nabla_\theta \mathcal{L}_{\text{TV}}}_{\text{training TV gradient}} \right\rangle \tag{25}$$

Therefore, given a learning rate η_λ, the update rule for λ writes:

$$\lambda^{(t+1)} := \lambda^{(t)} + \eta_\lambda \eta_\theta \left\langle \nabla_\theta \mathcal{L}_{\text{val}}, \nabla_\theta \mathcal{L}_{\text{TV}} \right\rangle \tag{26}$$

Intuitively, if $\langle \cdot, \rangle > 0$, it means that TV regularization improves validation performance and therefore λ should be increased. Algorithm 1 summarizes the training of the *LU-Net* architecture parameters when $\lambda \in \mathbb{R}_+$ is a learnable parameter.[7]

Algorithm 1 Gradient-Based Hyperparameter Update

Require: Initial parameters $\theta(0)$, TV weight $\lambda(0)$, learning rates η_θ, η_λ, training data D_{train}, and validation data D_{val}

for each training iteration $t \in \{1, \cdots, T\}$ **do**

Sample mini-batch $B_{\text{train}} \subset D_{\text{train}}$

Compute training losses:

$$\mathcal{L}_{\text{BCE}}(\theta(t); B_{\text{train}}), \quad \mathcal{L}_{\text{TV}}(\theta(t); B_{\text{train}})$$

Compute total loss:

$$\mathcal{L}_{\text{train}}(\theta(t), \lambda(t)) = \mathcal{L}_{\text{BCE}}(\theta(t)) + \lambda(t) \cdot \mathcal{L}_{\text{TV}}(\theta(t))$$

Update network parameters:

$$\theta(t+1) \leftarrow \theta(t) - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}}(\theta(t), \lambda(t))$$

Sample mini-batch $B_{\text{val}} \subset D_{\text{val}}$

Compute validation gradient:

$$g_{\text{val}} = \nabla_\theta \mathcal{L}_{\text{val}}(\theta(t+1); B_{\text{val}})$$

Compute TV gradient:

$$g_{\text{TV}} = \nabla_\theta \mathcal{L}_{\text{TV}}(\theta(t+1); B_{\text{train}})$$

Update hyperparameter:

$$\lambda(t+1) \leftarrow \lambda(t) + \eta_\lambda \eta_\theta \langle g_{\text{val}}, g_{\text{TV}} \rangle$$

end for

return $\theta(T), \lambda(T)$

8. Conclusion

This paper introduces an image processing heuristic—TV denoising—to train image segmentation networks on customized loss functions cognizant of the shape of the objects they seek to detect. Specifically, I experiment with U-Net architectures that incorporate TV loss in different ways for a roof segmentation task. The novel model I introduce, *LU-Net*, improves the IoU on the test set by 2.2pp relative to an analogous network trained only on binary cross-entropy loss. Further, it maintains a higher performance when tested on noisy images—with a margin of up to 3.6pp relative to a vanilla U-Net. This method improves existing architectures accounting for TV loss in segmentation tasks [4] by providing a methodology which results in (i) comparable performance, (ii) robustness to noise, and overall (iii) lower computational costs. Additionally, to reduce the computational burden of a hyperparameter sweep for the relative weight of the total variation loss when training the model, I build on past literature to provide an implicit differentiation procedure; essentially setting λ as a learnable parameter.

[7] While not explicitly stated in the algorithm, it may be preferable in practice to parameterize the hyperparameter as $\alpha = \log \lambda$ and optimize for it instead. This ensures that $\lambda = \exp(\alpha)$ remains non-negative throughout training.

References

[1] Y. K. Adimoolam, B. Chatterjee, C. Poullis, and M. Averkiou. Efficient deduplication and leakage detection in large scale image datasets with a focus on the crowdai mapping challenge dataset, 2023.

[2] J. Duran, B. Coll, and C. Sbert. Chambolle's projection algorithm for total variation denoising. *Image Processing On Line*, 3:74–90, 2013. Submitted on 2013-01-17, accepted on 2013-10-10, published on 2013-12-17.

[3] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization, 2017.

[4] F. Jia, J. Liu, and X.-C. Tai. A regularized convolutional neural network for semantic image segmentation. *Analysis and Applications*, 19(01):147–165, 2021. Special Issue on Mathematics of Data Science.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[6] D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning, 2015.

[7] F. Pedregosa. Hyperparameter optimization with approximate gradient. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 20–22 Jun 2016. PMLR.

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[9] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[10] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. 2021.