J. Appl. Numer. Optim. 4 (2022), No. 2, pp. 175-201
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.4.2022.2.05

POINT-TO-SET DISTANCE FUNCTIONS FOR OUTPUT-CONSTRAINED NEURAL NETWORKS

BAS PETERS

Department of Mathematics, Emory University, 400 Dowman Drive, Atlanta, Georgia, USA

Abstract. Training a neural network for semantic segmentation with many images and pixel-level segmentations is a well-established computer-vision technique. When pixel-level segmentations are unavailable, weak supervision or prior information like bounding boxes and the size/shape of objects still enables training a network. Directly including prior knowledge on the segmentations means constraining the network output. This complicates the possible optimization strategies because the regularization then acts on the non-linear neural-network function output and not on the optimization variables. We present a new algorithm to include prior information via constraints on the network output, implemented via projection-based point-to-set distance functions, that are differentiable and always have the same functional form for the derivative. Thus, there is no need to adapt penalty functions or algorithms to various constraints. The distance function's differentiability also avoids issues related to constraining properties typically associated with non-differentiable penalties. We show that by explicitly taking a general neural network structure into account, the Lagrangian for the problem 'naturally' decouples the constraints and neural network, which allows for a gradient computation via backpropagation/adjoint-state as is common in deep learning. We present a suite of constraint sets suitable for segmentation problems. The numerical experiments show that learning from constraint sets applies to the broader imaging sciences, including visual and non-visual imagery, even when training a network for a single example.

Keywords. Backpropagation; Constraint sets; Image segmentation; Neural networks; Projection methods; Regularization.

1. INTRODUCTION

Generating large datasets with fully segmented/annotated images for training convolutional neural networks is costly and time-consuming at best. Therefore, researchers developed methods for faster annotation and training approaches that learn from the sparsely annotated images. Such methods include point annotations [1, 2] and annotated slices from a 3D data volume [3]. More extreme labeling mechanisms to speed up the annotation time per image are bounding boxes [4, 5, 6, 7] or an indication whether or not an object is present in the image (image-level supervision) e.g., [8, 9]. Annotation time/cost of many categories of images is not a fundamental problem because most people can annotate street scenes or pictures of everyday objects and animals.

E-mail address: bas.peters@emory.edu.
Received March 16, 2021; Accepted September 21, 2021.
©2022 Journal of Applied and Numerical Optimization

More serious challenges arise in the imaging sciences when just domain experts can annotate. An even more difficult situation occurs when labels need to come from sparsely available ground truth observations. Data types that are difficult/impossible to annotate include certain hyperspectral data, geophysical data, medical images, highly corrupted images, and indirect measurements that typically require solving an inverse problem to create an image. The scope of the work consists exactly of these types of datasets from the imaging sciences, where the dataset contains one or a few examples with partial or no annotation. The general idea is to use weak-supervision, or high-level/prior knowledge, to address the limitations of training neural networks for semantic segmentation with scarce annotation. Examples of high-level information include bounds on the minimum and maximum size of an object, bounding boxes indicating object presence, as well as structural information like limited total-variation or monotonicity properties. The fact that total-variation and information about the size of an object can help solve segmentation problems dates back to at least [10, 11], where size information and total-variation information are used jointly in a level-set framework (without neural network).

This work introduces a new method to learn from high-level prior information that is more general in handling various constraints and easier to implement with fewer hyper-parameters. The approach also avoids alternatingly training a network and solving an auxiliary problem involving latent variables for constraint satisfaction. The new algorithm injects image-level supervision directly into the learning problem via constraints on the network output. Compared to earlier work, we introduce a new approach based on point-to-set distance functions with projection operators. Contrary to some approaches in the literature, we show that our approach does not lead to a very challenging or computationally expensive optimization problem. The distance functions, combined with a closer look at the optimization problem and the corresponding Lagrangian, reveal that the coupling between network parameters, network output, and constraints is not as problematic as it seems from an initial problem formulation. Furthermore, we describe methods that go beyond regularizing a single image property via combinations of various image characteristics constructed as a variant of a Minkowski set. The proposed algorithm enables training in various modes on datasets as small as a single example: without any annotation, with partial annotation available for just one of the classes, and with bounding boxes. The proposed framework can also serve as a regularization method for training with insufficiently many sparse annotations.

2. Related Work

The general concept of constraining an output of a neural network dates back to at least [12]. That work introduces (in)equality constraints encoded via differentiable functions as penalties or Lagrange multipliers. The scope and applications were different from image segmentation. Various applications aimed at natural language, some not in the context of neural networks, used a probabilistic learning framework to introduce posterior constraints [13], see also, e.g., [14, 15] for related work. [13] applied constraints that hold in expectation applied to multiple examples. [16] used an alternating optimization method for differentiable non-linear constraints.

Various works on vision applications are closer to this work and include penalties/constraints that promote the presence or minimum/maximum size of objects [8, 17, 18, 19]. [17] implemented linear inequality constraints on the network output with slack variables via an alternating optimization strategy by introducing auxiliary variables to remove direct coupling between

the network output and the constraints. [18] presented a Lagrangian method for training with non-linear but differentiable constraints, primarily for human-pose estimation. [19] and [7] added penalties or log-barriers on the violation of linear inequalities, respectively. Finally, [20] employed an expectation-maximization approach to ensure that total-variation (TV) constraints are always satisfied when generating physical quantities using a neural network. Except for this last reference, the aforementioned works all require differentiability of the penalty/constraint functions, and if provided, the gradient is specific to the particular constraint or related penalty term. These are limitations that we avoid by using projection-based functions to measure the distance to constraint sets. Furthermore, we show how to work with more complicated prior knowledge like intersections and sums of constraint sets.

Various authors added penalties to include shape priors: [21] added penalties that describe the 'topology' of class-inclusions and boundary smoothness for medical image segmentation; [22] promoted smoothness of the output probabilities in geophysical segmentation; [23] designed a penalty function that promotes star-shaped 'convexity' of objects in medical images; [24, 25] also added penalties for TV, volume, and star-shaped regularization, via a regularized variational formulation of the softmax function. This work focusses on optimization for constraints, particularly to avoid tuning multiple penalty parameters when working with multiple constraints. Moreover, our distance-function approach also provides a unified framework to turn all constraint sets into the same differentiable problem.

Some recent works use datasets with good and degraded images to train a neural network to discriminate between those two types of images. These works also show that the trained network mimics a distance function to the manifold of good images [26, 27, 28]. The trained network can operate as a distance function or projector in regularized image estimation and reconstruction algorithms. In contrast, we train a neural network to segment a given dataset based on prior knowledge and possibly some labels. The distance function is employed to enforce prior knowledge on the output of the network.

3. CONTRIBUTIONS

The primary differences compared to earlier and related works are in the area of 1) the optimization, 2) the range and types of prior knowledge that we include, 3) the target applications. Specifically, the contributions in each of these areas are as follows:

- We train a neural network for the segmentation tasks, using prior knowledge/high-level image information and possibly some labels. We include the prior knowledge in the training procedure via a projection-based distance function that acts on the output of the neural network. This framework does not rely on nested or alternating optimization schemes, does not need auxiliary variables, fits into the commonly used backpropagation algorithm for network gradient computation, and works with any constraint set in a plug-and-play style via its projection.
- Besides prior knowledge such as the size of objects, bounding boxes, and total-variation, we also show how to include multiple pieces of prior knowledge via intersections and sums of constraint sets, thereby preventing an unpractically large number of penalty parameters. Furthermore, we train neural networks via non-convex feasibility problems to include labels as constraint sets, which eliminates one more penalty parameters.

- This work extends the range of applications of output-constrained and regularized network training to single image/data problems without object-background setting, and where annotating requires domain-experts or ground-truth observations. The method applies to settings with *a*) no annotation at all; *b*) bounding boxes; *c*) partial annotation where one or more classes have no annotation; *d*) as a regularization method combined with sparse point-annotations.

The organization of the remainder of this paper is as follows: we introduce our problem formulations for training neural networks with high-level image information. Next, we introduce a novel optimization implementation for training neural nets with constraints on the output. The subsequent section describes how to translate various types of prior knowledge about a segmentation into constraint sets. Finally, five different problems from the imaging sciences illustrate the applicability and generality of our contributions.

4. NOTATION

In this work, images/data are treated in (block-) vectorized form: $\mathbf{d} \in \mathbb{R}^{n_1 n_2 n_3 n_{\text{chan}}}$ denotes the vector corresponding to 3D data input with multiple space/time/frequency coordinates, one channel coordinate (n_{chan}), and $N = n_1 n_2 n_3$. The non-linear and typically non-convex function $g(\mathbf{K}, \mathbf{d}) : \mathbb{R}^{n_{\text{param}}} \times \mathbb{R}^{N n_{\text{chan}}} \to \mathbb{R}^{N n_{\text{class}}}$ represents a neural network for semantic segmentation. Such a network transforms the n_{chan} input data into n_{class} probability maps (output channels) with N elements each. We indicate the output of the network for a single channel using the numbered subscript, i.e., $g(\mathbf{K}, \mathbf{d})_1$ is the first output channel of the network (the probability map for class number 1). The network depends on n_{param} parameters that we indicate by $\mathbf{K}$. The parameters are organized into j network layers, and the subscript j indicates the parameters for neural-network layer number j: $\mathbf{K}_j$. At each layer, the parameters $\mathbf{K}_j$ often represent a matrix, such as a dense or block-convolutional matrix where each block corresponds to a single input/output channel.

Assumption 4.1. The neural network function $g(\mathbf{K}, \mathbf{d})$ is non-linear, non-convex, and at least once differentiable such that gradient descent can minimize differentiable loss functions that include the network. This follows the standard assumptions in deep learning.

Assumption 4.2. For segmentation problems it is common that the neural network outputs class-probability values per pixel that satisfy

$$\sum_{k=1}^{n_{\text{class}}} g(\mathbf{K}, \mathbf{d})_{k,i} = 1 \text{ for all pixel indices } i, \tag{4.1}$$

$$0 \leq g(\mathbf{K}, \mathbf{d})_{\text{k},i} \leq 1 \text{ for all pixel indices } i \text{ and all class indices } k.$$

A neural network's output satisfies these conditions, for instance, if we select the softmax function

$$s(\mathbf{x})_k = \frac{e^{\mathbf{x}_k}}{\sum_{k=1}^{n_{\text{class}}} e^{\mathbf{x}_k}}$$

for the last layer. The softmax $s(\mathbf{x}) : \mathbb{R}^{n_{\text{class}}} \to \mathbb{R}^{n_{\text{class}}}$ maps its input $\mathbf{x} \in \mathbb{R}^{n_{\text{class}}}$ to numbers between zero and one, and that sum up to one. In other words, the network transforms the input image into one probability 'image' per class.

The vector $\mathbf{c}$ contains the labels/annotations, which may be partially known, not available at all for one or more classes, or completely absent. A typical loss function to measure the difference between a vector $\mathbf{x}$ and the labels $\mathbf{c}$ is the multi-class cross-entropy loss. Combined with the softmax, the indexing for this loss is conveniently expressed in matrix form with labels $\mathbf{C} \in \mathbb{R}^{N \times n_{\text{class}}}$ and input $\mathbf{X} \in \mathbb{R}^{N \times n_{\text{class}}}$ as

$$l(\mathbf{X}, \mathbf{C}) = -\sum_{i=1}^{N} \sum_{k=1}^{n_{\text{class}}} \mathbf{C}_{i,k} \log \left(\frac{e^{\mathbf{X}_{i,k}}}{\sum_{k=1}^{n_{\text{class}}} e^{\mathbf{X}_{i,k}}} \right). \tag{4.2}$$

The following sections use of the Euclidean projection operator $P_C(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ onto a constraint set C:

$$P_C(\mathbf{m}) \in \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in C.$$

The indicator function corresponding to a set C is defined as

$$\iota_C(\mathbf{m}) = \begin{cases} 0 & \text{if } \mathbf{m} \in C, \\ +\infty & \text{if } \mathbf{m} \notin C. \end{cases}$$

5. Learning from High-level Image Information via Constraints on the Network Output

When full-supervision in the form of fully annotated pixel-level labels are not available for training deep neural networks for image segmentation, there may still be enough information available for training using weak-supervision. We consider quantitative high-level image information. Previous works include information on the area/volume/number of pixels that a class is expected to occupy in the segmentation. [17, 19] showed that such information is sufficient to obtain pixel-level segmentations for object detection tasks in both general and medical image datasets, while the prior knowledge does not need to be very precise.

This work develops a more general optimization strategy to include multiple pieces of prior knowledge that will allow for training using a wider variety of prior information, using smaller datasets, and learn tasks beyond object segmentation. Our approach is based on a constrained formulation that avoids the trade-off parameters that make penalty methods less practical when dealing with many constraints. To accommodate more complicated constraint sets, we start with the set D, which may be the intersection of p sets C_i,

$$D = \bigcap_{i=1}^{p} C_i \neq \emptyset.$$

This intersection is assumed to be non-empty so that D does not lead to infeasible optimization problems by construction. To allow even more flexible constraint set constructions, one of the sets C_i in the intersection D may be a Minkowski set,

$$V + U = \{\mathbf{m} = \mathbf{u} + \mathbf{v} \,|\, \mathbf{u} \in U, \mathbf{v} \in V\}, \tag{5.1}$$

which enables the implementation of constrained versions of decomposition ideas like morphological component analysis/cartoon-texture decomposition/robust principal component analysis [29, 30, 31, 32, 33, 34, 35, 36, 37]. The sum of two sets is convex if both U and V are convex sets [38, p. 24]. It follows that D is convex as well if all C_i are convex sets.

Using the above constraint set constructions, we state our problem as the minimization of a loss function, l, that measures the discrepancy between labels, $\mathbf{c}$, and the neural-network output $g(\mathbf{K},\mathbf{d})$, subject to constraints on the prediction. For a single example, this statement corresponds to the optimization problem over network parameters $\mathbf{K}$,

$$\min_{\mathbf{K}} \; l(\mathbf{Q}g(\mathbf{K},\mathbf{d}),\mathbf{c}) \text{ s.t. } g(\mathbf{K},\mathbf{d}) \in D, \tag{5.2}$$

where the matrix $\mathbf{Q}$ selects from the output the pixels where labels are available. A typical choice of loss function l is the multi-class cross-entropy loss (4.2).

Recall that $g(\mathbf{K},\mathbf{d})$ denotes the neural network output that depends on the network parameters $\mathbf{K}$ and the input data $\mathbf{d}$. There are two scenarios in which the above optimization problem reduces to a feasibility problem. First, if there are no labels at all. Second, because our formulations use intersections of constraint sets already, an additional set of bound constraints to make the network output fit the labels does not require any problem modifications. The corresponding feasibility problem reads

$$\text{find}\, g(\mathbf{K},\mathbf{d}) \in D \Leftrightarrow \min_{\mathbf{K}} \iota_D(g(\mathbf{K},\mathbf{d})), \tag{5.3}$$

where ι_D is the indicator function for the intersection of constraint sets D. Both problem versions appear in the example section later in this work. In search of an algorithm to solve (5.2), a first idea could be to look at projected gradient descent.

Why does forward-backward splitting not help? To illustrate why problems involving constraints on the output of a neural network (5.2) are not trivial to solve using a forward-backward splitting approach, we look at the projected gradient iteration

$$\begin{aligned}\mathbf{K}^{i+1/2} &= \mathbf{K}^i - \gamma \nabla_{\mathbf{K}} l(\mathbf{Q}g(\mathbf{K}^i,\mathbf{d}),\mathbf{c}),\\ \mathbf{K}^{i+1} &\in \arg\min_{\mathbf{K}} \frac{1}{2}\|\mathbf{K}-\mathbf{K}^{i+1/2}\|_2^2 \text{ s.t. } g(\mathbf{K},\mathbf{d}) \in D.\end{aligned}$$

The first equation is the standard gradient descent step with step size $\gamma > 0$ and is straightforward to compute. However, the constraints in the projection step act on the network output and not on the optimization parameters $\mathbf{K}$ (e.g., convolutional kernels and biasses). The resulting projection step is almost as difficult to solve as the original problem because there is no simple way to compute projections that involve $g(\mathbf{K},\mathbf{d}) \in D$.

It is possible to proceed by introducing auxiliary variables and equality constraints to construct a projection problem that requires $g(\mathbf{K},\mathbf{d})$ to be an element of D indirectly; see, e.g., [17]. However, in the following section, we introduce a different approach based on the distance to the constraint set. We argue that this is a simpler method because there are no auxiliary variables, and no alternating optimization strategy between network variables and auxiliary variables is required. Moreover, we show that standard backpropagation/adjoint-state optimization applies to our formulation such that our approach integrates in established deep learning workflows without modifications.

5.1. **Point-to-set distance functions for networks with output-constraints.** To goal is to construct an algorithm to train a neural network with constraints on the output. The following requirements lead to broader applicability of the method: 1) the algorithm needs to solve both problems (5.2) and (5.3); 2) be compatible with existing training frameworks for deep learning

based on backpropagation; 3) should not be tied to a specific constraint type but accommodate a wide variety of constraints, as well as their intersections and sums.

To arrive at an algorithm in accordance with the above three items, the point-to-set distance function forms the core of our approach. Specifically, consider the version that measures the squared distance from a point in $\mathbb{R}^N$ to the set D,

$$d_D^2(\mathbf{y}) = \min_{\mathbf{x}\in D} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 = \frac{1}{2}\|P_D(\mathbf{y})-\mathbf{y}\|_2^2. \tag{5.4}$$

The evaluation of the distance requires just the projection operator of a vector onto the intersection of constraints, P_D. We could also opt for an exact version by removing the square [39, Thm. 1.2.3] but this is not of paramount importance for our applications. The squared distance function is differentiable [38, Ex. 3.3] and the gradient is given by

$$\nabla_{\mathbf{y}} d_D^2(\mathbf{y}) = \mathbf{y} - P_D(\mathbf{y}). \tag{5.5}$$

The differentiability of the distance function is a valuable property because it holds even if the constraint set corresponds to regularizers that lead to non-differentiable penalty functions, such as sparsity promotion via the ℓ_1-norm. Therefore, the distance function integrates seamlessly with (stochastic) gradient schemes used for training neural networks. All information about the constraint sets is implicit in the projection operator; an advantage over approaches that add a custom penalty function for each property that needs to be constrained.

To finalize this section, we note that a closed-form derivative of the distance function using the projection operator itself is a powerful result that is also at the core of algorithms for (split-) feasibility problems involving one or multiple constraint sets such as the CQ algorithm [40, 41]; see e.g., [42, 43] for some recent advances. These ideas did not see applications to distance functions that include neural networks, however.

5.2. **Gradient of the distance function including a neural network.** The discussion so far focussed on the formulation of training a neural network with output constraints (equations (5.3) and (5.2)), as well as the motivation to tackle these problems using the projection-based point-to-set distance function (5.4). Putting the two together leads to the optimization problems

$$\min_{\mathbf{K}} \frac{1}{2}\|P_D(g(\mathbf{K},\mathbf{d})) - g(\mathbf{K},\mathbf{d})\|_2^2, \tag{5.6}$$

and

$$\min_{\mathbf{K}} l(\mathbf{Q}g(\mathbf{K},\mathbf{d}),\mathbf{c}) + \frac{\alpha}{2}\|P_D(g(\mathbf{K},\mathbf{d})) - g(\mathbf{K},\mathbf{d})\|_2^2. \tag{5.7}$$

The second problem requires the penalty parameter $\alpha > 0$. Moreover, in practice, the value of α likely needs to grow while optimizing (5.7) to ensure that $\|P_D(g(\mathbf{K},\mathbf{d})) - g(\mathbf{K},\mathbf{d})\|_2^2 \approx 0$ and therefore mimics the original constrained problem (5.2). See the numerical experiments for details about the selection of α. It may be appealing to write down the gradient for the above problem by using the closed-form derivative for the distance function (5.5) and use the chain rule to combine with the gradient $\nabla_{\mathbf{K}} g(\mathbf{K},\mathbf{d})$. This strategy, however, does not take into account any known structure of $g(\mathbf{K},\mathbf{d})$. The derivation below shows how this structure will simplify problems (5.6) and (5.7).

Many (convolutional) neural networks $g(\mathbf{K},\mathbf{d})$ have layers of the form $\mathbf{y}_j = f(\mathbf{y}_{j-1},\mathbf{K}_j)$, where $\mathbf{y}_j$ is the network state at layer j, and $f(\mathbf{y}_{j-1},\mathbf{K}_j)$ is a non-linear evolution of the previous network state. An example includes the successful ResNet [44]. Network designs with longer

recurrences share a similar relation. For instance, a hyperbolic network layer has the form $\mathbf{y}_j = f(\mathbf{y}_{j-1}, \mathbf{y}_{j-2}, \mathbf{K}_j)$ [45]. The key property is that the *last* network state (the output) occurs in an explicit form dependent on earlier network states only, such that it can be written explicitly as $g(\mathbf{K}, \mathbf{d}) = \mathbf{y}_n$, where n is the final layer index of the network.

The derivation of the gradient of the distance function with a neural network proceeeds by including the structure of a neural network in the minimization (5.7). To keep the notation compact, the remainder of this section uses a standard ResNet [44]. This choice leads to the constrained optimization problem

$$\begin{aligned} \min_{\{\mathbf{K}\}} \; & l(\mathbf{Q}\mathbf{y}_n, \mathbf{c}) + \frac{\alpha}{2}\|P_D(\mathbf{y}_n) - \mathbf{y}_n\|_2^2 \;\text{ s.t.} \\ & \mathbf{y}_n = \mathbf{y}_{n-1} - \sigma(\mathbf{K}_n \mathbf{y}_{n-1}) \\ & \vdots \\ & \mathbf{y}_j = \mathbf{y}_{j-1} - \sigma(\mathbf{K}_j \mathbf{y}_{j-1}) \\ & \vdots \\ & \mathbf{y}_1 = \mathbf{d}, \end{aligned} \tag{5.8}$$

with one equality constraint for each of the n neural-network layers. The constraint $\mathbf{y}_1 = \mathbf{d}$ sets the initial state equal to the input data. We omit the corresponding equations for the feasibility problem (5.6) because this merely requires leaving out the term $l(\mathbf{Q}\mathbf{y}_n, \mathbf{c})$ and the penalty parameter α. The function $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ is the non-linear point-wise activation, and the numerical experiments employ the ReLU function: $\sigma(\mathbf{x}) = \max(0, \mathbf{x})$. We emphasize that this work does not depend on the ResNet, and most neural networks could replace the ResNet in the above problem statement.

Minimizing a distance function applied to the output of a network does not need to fundamentally change the optimization process for training a neural network based on labels alone. The proposed approach merges seamlessly with most existing neural networks and training algorithms. To explicitly show this, consider the Lagrangian corresponding to (5.8),

$$\begin{aligned} L(\{\mathbf{y}\}, \{\mathbf{p}\}, \{\mathbf{K}\}) &= l(\mathbf{Q}\mathbf{y}_n, \mathbf{c}) + \frac{\alpha}{2}\|P_D(\mathbf{y}_n) - \mathbf{y}_n\|_2^2 \\ &- \sum_{j=2}^{n} \mathbf{p}_j^\top (\mathbf{y}_j - \mathbf{y}_{j-1} + \sigma(\mathbf{K}_j \mathbf{y}_{j-1})) - \mathbf{p}_1^\top (\mathbf{y}_1 - \mathbf{d}), \end{aligned}$$

where $\mathbf{p}_j$ are the vectors of Lagrange multipliers for every layer j. Optimization algorithms need the following partial derivatives of the Lagrangian:

$$\nabla_{\mathbf{y}_n} L = \nabla_{\mathbf{y}_n} l(\mathbf{Q}\mathbf{y}_n, \mathbf{c}) + \alpha(\mathbf{y}_n - P_D(\mathbf{y}_n)) - \mathbf{p}_n$$

$$\begin{aligned} &\text{for } j = n-1, \cdots, 2: \\ &\nabla_{\mathbf{y}_{j-1}} L = -\mathbf{p}_{j-1} + \mathbf{p}_j - \mathbf{K}_j^\top \operatorname{diag}(\sigma'(\mathbf{K}_j \mathbf{y}_{j-1}))\mathbf{p}_j, \\ &\nabla_{\mathbf{p}_j} L = -\mathbf{y}_j + \mathbf{y}_{j-1} - \sigma(\mathbf{K}_j \mathbf{y}_{j-1}), \\ &\nabla_{\mathbf{K}_j} L = \left[\frac{\partial(\mathbf{K}_j \mathbf{y}_{j-1})}{\partial \mathbf{K}_j}\right]^\top \operatorname{diag}(\sigma'(\mathbf{K}_j \mathbf{y}_{j-1}))\mathbf{p}_j. \end{aligned}$$

The function $\sigma'(\cdot)$ is the derivative of the activation function, where we follow the common approach of ignoring the non-differentiability of the ReLU at zero by assigning $\sigma'(0) = 0$. The operator $\text{diag}(\cdot)$ creates a diagonal matrix of its input. The gradient for the network parameters $\mathbf{K}_j$ at every layer follows via the reduced-Lagrangian/adjoint-state/backpropagation algorithm. See [46] for the equivalence between these algorithms.

First, forward propagating through the network computes all $\mathbf{y}_j$ that satisfy the equality constraints so it follows that all $\nabla_{\mathbf{p}_j} L = 0$. Then, propagating backward provides the Lagrange multipliers $\mathbf{p}_j$ that satisfy $\nabla_{\mathbf{y}_j} L = 0$. The last step computes the gradient with respect to the network parameters, $\nabla_{\mathbf{K}_j} L$, using the already computed states and multipliers.

Algorithm 1 summarizes the steps outlined above. The partial derivatives of the Lagrangian and Algorithm 1 show that the constraints, implemented via a distance function, insert information into the final Lagrange multiplier that then backpropagates through all layers and influences the gradient with respect to the network parameters. The penalty parameter α and loss function $l(\mathbf{Q}\mathbf{y}_n, \mathbf{c})$ are absent for the feasibility problem (5.3).

Algorithm 1: Backpropagation to train a network including constraints on the network output via distance-to-set functions.

$\mathbf{y}_1 = \mathbf{d}$, $\alpha > 0$ (penalty parameter), γ_i (iteration dependent decaying step-size/learning rate, like $\gamma_i = \gamma_0/i$), P_D (projection operator onto intersection of constraint sets), $\mathbf{c}$ (labels), $\mathbf{K}_1, \mathbf{K}_2, \cdots, \mathbf{K}_n$ (randomly initialized (parameterized) weight matrices);

for $i = 1 : n_{max\ iter}$ **do**

  for $j = 2, \cdots, n$ **do**

    $\mathbf{y}_j = \mathbf{y}_{j-1} - \sigma(\mathbf{K}_j \mathbf{y}_{j-1})$ // Forward;

  end

  $\mathbf{p}_n = \nabla_{\mathbf{y}_n} l(\mathbf{Q}\mathbf{y}_n, \mathbf{c}) + \alpha(\mathbf{y}_n - P_D(\mathbf{y}_n))$ //Final Lagrange multiplier;

  //Propagate backward and update network parameters for each layer;

  for $j = n, n-1, \cdots, 2$ **do**

    $\nabla_{\mathbf{K}_j} L = \left[\frac{\partial\left(\mathbf{K}_j \mathbf{y}_{j-1} \right)}{\partial \mathbf{K}_j} \right]^\top \text{diag}(\sigma'(\mathbf{K}_j \mathbf{y}_{j-1}) \mathbf{p}_j$;

    $\mathbf{p}_{j-1} = \mathbf{p}_j - \mathbf{K}_j^\top \text{diag}(\sigma'(\mathbf{K}_j \mathbf{y}_{j-1})) \mathbf{p}_j$;

    $\mathbf{K}_j \leftarrow \mathbf{K}_j - \gamma_i \nabla_{\mathbf{K}_j} L$;

  end

end

The distance-function approach makes sure the output of the network satisfies the constraints after a sufficient number of gradient descent iterations if the non-convexity of the problem does cause the minimization to stall prematurely. Numerical examples empirically verify that the distance to the constraint sets decreases to satisfactory levels. Only the training phase uses prior knowledge in this work. After successful training, the network learned how to segment the data of interest, and there should be no need to include prior knowledge for inference.

6. Translating High-level Image Information into Constraint Sets

The following is a discussion of the types of high-level information/prior knowledge/weak-supervision that we propose to use to train neural networks for segmentation tasks, and how to translate the information into constraint sets.

6.1. Constraints on the size/surface area of a class.

To segment an image into two or more classes, knowledge on how many pixels each class will occupy in the segmentation can help train the network. In the case of two classes this means,

$$\begin{aligned} a_1 \leq &\text{area(class 1)} \leq a_2, \\ (1-a_2) \leq &\text{area(class 2)} \leq (1-a_1), \end{aligned} \tag{6.1}$$

where the scalars a_1 and a_2 are bounds on the size or 'area' (surface in 2D or volume in 3D) of a class in the segmented image. Several implementations of this type of prior knowledge are conceivable. These implementations have different convexity properties and also differ in how directly they translate the information into a constraint set.

Cardinality constraints: The cardinality of a vector, $\text{card}(\mathbf{x})$, counts the number of non-zero elements in $\mathbf{x}$. Therefore, a direct translation from (6.1) creates the sets

$$\begin{aligned} &\{g(\mathbf{K},\mathbf{d})_1 \,|\, \text{card}(g(\mathbf{K},\mathbf{d})_1) \leq a_2\}, \\ &\{g(\mathbf{K},\mathbf{d})_2 \,|\, \text{card}(g(\mathbf{K},\mathbf{d})_2) \leq (1-a_1)\}, \end{aligned} \tag{6.2}$$

where $g(\mathbf{K},\mathbf{d})_1$ indicates the first channel of the network output $g(\mathbf{K},\mathbf{d})$. Here it is possible to use upper bounds only because it is assumed that the network normalizes the each pixel in the output to $0-1$ and sums all classes to one, for example, via the softmax function (see (4.1)). The lower bounds are implicit via the normalization of the output, i.e., if $\text{card}(g(\mathbf{K},\mathbf{d})_1) \leq a_2$, it implies that $(1-a_2) \leq \text{card}(g(\mathbf{K},\mathbf{d})_2)$. The projection onto this non-convex set is known in closed form by setting all but the largest values (in absolute sense) to zero. A typical implementation sorts the vector, determines the threshold value, set all elements below the threshold to zero, and finally reverts to the original order.

ℓ_1 constraints: The ℓ_1 norm-ball offers a convex relaxation of the cardinality and results in the constraint sets

$$\begin{aligned} &\{g(\mathbf{K},\mathbf{d})_1 \,|\, \|g(\mathbf{K},\mathbf{d})_1\|_1 \leq a_2\}, \\ &\{g(\mathbf{K},\mathbf{d})_2 \,|\, \|\,g(\mathbf{K},\mathbf{d})_2\|_1 \leq (1-a_1)\}, \end{aligned}$$

corresponding to (6.1). For these upper bounds to have meaning, assume that the network output per pixel is normalized between zero and one, and both channels sum to one (see (4.1)). It follows that the output of the network is non-negative so the ℓ_1 norm reduces to the summations

$$\begin{aligned} &\{g(\mathbf{K},\mathbf{d})_1 \,|\, \sum_{i=i}^{N} g(\mathbf{K},\mathbf{d})_{1,i} \leq a_2\}, \\ &\{g(\mathbf{K},\mathbf{d})_2 \,|\, \sum_{i=i}^{N} g(\mathbf{K},\mathbf{d})_{2,i} \leq (1-a_1)\}. \end{aligned}$$

The quantity $g(\mathbf{K},\mathbf{d})_{1,i}$ represents element i from channel 1 of the network output. These sets are related to the penalty function $a_1 \leq \sum_{i=i}^{N} g(\mathbf{K},\mathbf{d})_{1,i} \leq a_2$ that was used for weakly supervised medical image segmentation [19]. Bounds on the ℓ_1 norm do not directly indicate how many pixels will be assigned to a certain class. The value of $\|g(\mathbf{K},\mathbf{d})_1\|_1$ may concentrate in a minimal

number of pixels, or be spread out evenly among a large number of pixels. For this reason, the numerical examples use the cardinality constraint.

6.2. **Point-annotations and bounding-boxes as bound constraints.** A benefit of including point-annotations and bounding box information as bound constraints instead of a separate loss function, is the absence of the penalty parameter α (as in Equation (5.8)).

Point annotations: The definition of a point annotation is a single pixel that has been assigned a class. The assumption is that only a small number of pixels come with annotation. The collection Ω denotes the pixels labeled 'true', and Ψ corresponds to pixels labeled 'false'. The corresponding bound constraints are

$$\{g(\mathbf{K},\mathbf{d})_1 \,|\, g(\mathbf{K},\mathbf{d})_{1,\Omega} = 1,\, g(\mathbf{K},\mathbf{d})_{1,\Psi} = 0\},$$
$$\{g(\mathbf{K},\mathbf{d})_2 \,|\, g(\mathbf{K},\mathbf{d})_{2,\Omega} = 0,\, g(\mathbf{K},\mathbf{d})_{2,\Psi} = 1\},$$

where channel one and channel two correspond to the 'true' and 'false' classes, respectively. The projection operators for these sets with an input vector $\mathbf{x}$ are special cases of the projection onto the set of (pixel-dependent) bound constraints

$$P_1(\mathbf{x}) = \begin{cases} \mathbf{x}_i = 1 \text{ if } i \in \Omega, \\ \mathbf{x}_i = 0 \text{ if } i \in \Psi \end{cases} \qquad P_2(\mathbf{x}) = \begin{cases} \mathbf{x}_i = 0 \text{ if } i \in \Omega, \\ \mathbf{x}_i = 1 \text{ if } i \in \Psi \end{cases}.$$

Note that we do not need to set the bounds as equalities in the above constraints. If we decide upfront that any output value > 0.5 in channel one will be classified as 'true', then we may also use bounds like $g(\mathbf{K},\mathbf{d})_{1,\Omega} > 0.5$ and $g(\mathbf{K},\mathbf{d})_{1,\Psi} \leq 0.5$ for channel one. The difference is in the 'certainty' that we require from the network output that a given pixel is in class one or class two.

Bounding boxes: Constraints that implement bounding boxes are similar to point annotations. A bounding box indicates an area in which an object may be present and at the same time an area outside of which no object occurs. The bound constraints corresponding to the bounding box are given by

$$\{g(\mathbf{K},\mathbf{d})_1 \,|\, 0 \leq g(\mathbf{K},\mathbf{d})_{1,\Omega} \leq 1,\, g(\mathbf{K},\mathbf{d})_{1,\Psi} = 0\}$$
$$\{g(\mathbf{K},\mathbf{d})_2 \,|\, 0 \leq g(\mathbf{K},\mathbf{d})_{2,\Omega} \leq 1,\, g(\mathbf{K},\mathbf{d})_{2,\Psi} = 1\}.$$

This time, the collection of pixels in Ω defines the inside of the bounding box, and Ψ defines the rest of the image. Again, channels one and channel two correspond to the 'true' and 'false' classes, respectively. The corresponding projection operators for an input vector $\mathbf{x}$ are given by

$$P_1(\mathbf{x}) = \begin{cases} \mathbf{x}_i = \text{median}(0, \mathbf{x}_i, 1) \text{ if } i \in \Omega, \\ \mathbf{x}_i = 0 \text{ if } i \in \Psi, \end{cases} \qquad P_2(\mathbf{x}) = \begin{cases} \text{median}(0, \mathbf{x}_i, 1) \text{ if } i \in \Omega, \\ \mathbf{x}_i = 1 \text{ if } i \in \Psi. \end{cases}.$$

6.3. **Structural constraints.** Rather than aiming to learn the shape and structure of objects and images in a purely data-driven fashion by using a large number of images and corresponding pixel-level segmentations, we can inject such knowledge directly into the optimization problem for selected applications. Consider the segmentation of street imagery to outline the vehicles. On average, vehicles will be a connected set of pixels (unless obscured by other objects) and have a relatively low total-variation. Utilizing such prior knowledge will enable training with fewer images and with fewer annotation (either pixels, bounding boxes, or fully annotated images), thus reducing the annotation time and cost.

Structural constraints, or 'shape' priors, come in many flavors, and the focus here is on sparse or compressible image descriptions in a suitable transform-domain. This class of image descriptions takes the form

$$\{g(\mathbf{K},\mathbf{d})_1 \,|\, \|\mathbf{A}g(\mathbf{K},\mathbf{d})_1\|_1 \leq \lambda_1\}$$
$$\{g(\mathbf{K},\mathbf{d})_2 \,|\, \|\mathbf{A}g(\mathbf{K},\mathbf{d})_2\|_1 \leq \lambda_2\},$$

and $\lambda_1 > 0$, $\lambda_2 > 0$ are different for each channel. The linear operator $\mathbf{A} \in \mathbb{R}^{M\times N}$ represents a transform-domain in terms of, for instance, Fourier or wavelet coefficients, or anisotropic total-variation using finite-difference matrices. Unlike the previously discussed bounds, cardinality and ℓ_1 constraints, the projection onto the above constraint sets is unavailable in closed form if $\mathbf{A}\mathbf{A}^\top \neq \beta I$, where βI is a scaled identity matrix [47, ch. 6 & 7]. Instead, another algorithm needs to compute the projection.

6.4. **Intersections of multiple constraints.** The constraint set of interests becomes an intersection $D = \bigcap_{i=1}^{p} C_i$ whenever more than one piece of prior knowledge is available, or when annotation is included as bound constraints together with at least one other piece of prior information. The projection operation inside the distance function then becomes the projection operator onto the intersection, $P_D = P_{C_1\bigcap C_2\bigcap\ldots}$. Generally, the projection onto an arbitrary intersection has no known closed-form solution.

However, we employ the specialized software [48] to (approximately) compute the projection onto an intersection. That approach formulates the projection of $\mathbf{m} \in \mathbb{R}^N$ onto the intersection,

$$P_D(\mathbf{m}) = \arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t.} \quad \mathbf{m} \in \bigcap_{i=1}^{p} C_i$$

as a sum of indicator functions ι_{C_i} that explicitly include linear operators $\mathbf{A}_i \in \mathbb{R}^{M_i\times N}$,

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p} \iota_{C_i}(\mathbf{A}_i\mathbf{x}),$$

followed by separating the linear operators from indicator functions via newly introduced variables $\mathbf{z}_i \in \mathbb{R}^{M_i}$ as

$$\min_{\mathbf{x},\{\mathbf{z}_i\}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p} \iota_{C_i}(\mathbf{z}_i) \quad \text{s.t.} \quad \mathbf{A}_i\mathbf{x} = \mathbf{z}_i \;\text{ for } i = 1,2,\cdots p. \tag{6.3}$$

Solving this equality constrained problem using the Alternating Direction Method of Multipliers ADMM [49, 50] via the augmented Lagrangian corresponding to (6.3) leads to the following algorithm with iteration counter k:

$$\mathbf{x}^{k+1} = \Big[\sum_{i=1}^{p}(\rho_i^k\mathbf{A}_i^\top\mathbf{A}_i) + \rho_{p+1}^k\mathbf{I}_N\Big]^{-1}\sum_{i=1}^{p+1}\Big[\mathbf{A}_i^\top(\rho_i^k\mathbf{z}_i^k + \mathbf{v}_i^k)\Big]$$
$$\text{for } i \in \{1,2,\cdots,p+1\} \text{ compute in parallel:}$$
$$\bar{\mathbf{x}}_i^{k+1} = \gamma_i^k\mathbf{A}_i\mathbf{x}_i^{k+1} + (1-\gamma_i^k)\mathbf{z}_i^k$$
$$\mathbf{z}_i^{k+1} = P_{C_i}(\bar{\mathbf{x}}_i^{k+1} - \frac{\mathbf{v}_i^k}{\rho_i^k})$$
$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \rho_i^k(\mathbf{z}_i^{k+1} - \bar{\mathbf{x}}_i^{k+1}).$$

This is the algorithm presented by [48], and without the relaxation step $\bar{\mathbf{x}}_i^{k+1} = \gamma_i^k \mathbf{A}_i \mathbf{x}_i^{k+1} + (1-\gamma_i^k)\mathbf{z}_i^k$, these iterations are a variation of the SALSA algorithm [51] or the simultaneous direction method of multipliers [SDMM, 52, 53]. Because the above algorithm is equivalent to ADMM, if all C_i are convex sets, relaxation constants assumed to satisfy $\gamma_i \in [1,2)$, see [54], and for any selection for the augmented Lagrangian penalty parameters $\rho_i > 0$, the above algorithm converges to a solution because the standard convergence conditions for ADMM are satisfied [49, 50]. The vectors $\mathbf{v}_i \in \mathbb{R}^{M_i}$ represent the Lagrange multipliers that originate from the augmented Lagrangian corresponding to (6.3). In the above algorithm, the computational cost concentrates on iteratively solving a single linear system inexactly, and projections in closed-form onto each constraint set individually. The linear operators were moved from the constraint set to the linear-system solve.

We prefer the above approach over another option that fits within our framework: tools similar to [41] describe how to add multiple distance functions $+\alpha_1/2\|P_{C_1}(\mathbf{x})-\mathbf{x}\|_2^2 + \alpha_2/2\|P_{C_2}(\mathbf{x})-\mathbf{x}\|_2^2 + \cdots$. This implementation is relatively straightforward, but introduces multiple penalty parameters that need tuning; a potential problem that becomes more involved/difficult when the number of penalties increases. We leave comparisions between this method and the ADMM-based approach in terms of practicality and computational cost for future work.

6.5. **Sums of multiple constraint sets.** The vector sum of constraint sets (the Minkowski set) (5.1) is another fundamental operation on sets that shows to be useful in the examples section. Minkowski sets can serve similar purposes to decomposition methods that are typically based on penalty functions such as cartoon-texture decomposition [29, 30, 31, 32, 33, 34] and variants of robust principal component analysis [35, 36, 37]. Projection onto the Minkowski set is generally not available in closed form. Instead, we use an algorithm dedicated to this task [55]. While notationally heavy, we briefly show this problem setup and its projection computation. The projection of a vector $\mathbf{m} \in \mathbb{R}^N$ onto a generalization of the Minkowski set that allows both the components $\mathbf{u}$ and $\mathbf{v}$, and their sum to be elements of different intersections of constraint sets can be written as

$$\min_{\mathbf{u},\mathbf{v},\mathbf{w}} \frac{1}{2}\|\mathbf{w}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p} \iota_{G_i}(\mathbf{A}_i\mathbf{u}) + \sum_{j=1}^{q} \iota_{E_j}(\mathbf{B}_j\mathbf{v}) + \sum_{k=1}^{r} \iota_{F_k}(\mathbf{C}_k\mathbf{w}) + \iota_{\mathbf{w}=\mathbf{u}+\mathbf{v}}(\mathbf{w},\mathbf{u},\mathbf{v}). \tag{6.4}$$

The projected result $\mathbf{w} \in \mathbb{R}^N$ is the sum of two components $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{v} \in \mathbb{R}^N$. The indicator functions ι correspond to constraint sets G_i and linear operators $\mathbf{A}_i$ that act on one component, constraint sets E_j and operators $\mathbf{B}_j$ that act on the second component, and constraint sets F_k with linear operators $\mathbf{C}_k$ that act on the sum of the two components. The indicator function $\iota_{\mathbf{w}=\mathbf{u}+\mathbf{v}}(\mathbf{w},\mathbf{u},\mathbf{v})$ enforces the equality constraint $\mathbf{w} = \mathbf{u}+\mathbf{v}$. As in the previous subsection, the strategy is to solve (6.4) using ADMM. To see why this is possible, first eliminate the equality constraint $\mathbf{w} = \mathbf{u}+\mathbf{v}$ by introducting the vector $\mathbf{x} \equiv \begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} \in \mathbb{R}^{2N}$, and substituting this in new definition into (6.4) yields

$$\min_{\mathbf{x}} \frac{1}{2}\|\begin{pmatrix}\mathbf{I}_N & \mathbf{I}_N\end{pmatrix}\mathbf{x}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p} \iota_{G_i}(\begin{pmatrix}\mathbf{A}_i & \mathbf{0}\end{pmatrix}\mathbf{x}) + \sum_{j=1}^{q} \iota_{E_j}(\begin{pmatrix}\mathbf{0} & \mathbf{B}_j\end{pmatrix}\mathbf{x}) + \sum_{k=1}^{r} \iota_{F_k}(\begin{pmatrix}\mathbf{C}_k & \mathbf{C}_k\end{pmatrix}\mathbf{x}),$$

where $\mathbf{0}$ is the matrix of all zeros and $\mathbf{I}_N$ is the identity matrix of size $N \times N$. As in the algorithm to project onto an intersection, introducing auxiliary variables $\mathbf{z}$ and adding equality constraints

separates indicator functions and linear operators as

$$\min_{\{\mathbf{z}_i\},\mathbf{x}} \frac{1}{2}\|\mathbf{z}_{p+q+r+1}-\mathbf{m}\|_2^2+\sum_{i=1}^{p}\iota_{G_i}(\mathbf{z}_i)+\sum_{j=1}^{q}\iota_{E_j}(\mathbf{z}_{j+p})+\sum_{k=1}^{r}\iota_{F_k}(\mathbf{z}_{k+p+q})$$
$$\text{s.t.}\begin{cases}(\mathbf{A}_i \quad \mathbf{0})\mathbf{x}=\mathbf{z}_i & \text{for } i=1,2,\cdots,p\\ (\mathbf{0} \quad \mathbf{B}_j)\mathbf{x}=\mathbf{z}_{j+p} & \text{for } j=1,2,\cdots,q\\ (\mathbf{C}_k \quad \mathbf{C}_k)\mathbf{x}=\mathbf{z}_{k+p+q} & \text{for } k=1,2,\cdots,r\\ (\mathbf{I}_N \quad \mathbf{I}_N)\mathbf{x}=\mathbf{z}_{p+q+r+1}\end{cases}. \tag{6.5}$$

In this form, one can proceed with formulating the ADMM iterations via the augmented Lagrangian of (6.5). This approach leads to the algorithm with iteration counter l,

$$\begin{aligned}&\mathbf{x}^{l+1}=\left[\mathbf{Q}^l\right]^{-1}\mathbf{b}^l,\\ &\text{for } t\in\{1,2,\cdots,p+q+r+1\} \text{ compute in parallel:}\\ &\bar{\mathbf{x}}_t^{l+1}=\text{relax}(\gamma_t^l,\mathbf{x}^l,\mathbf{z}_t^l),\\ &\mathbf{z}_t^{l+1}=\text{prox}_{t,\rho_t}(\bar{\mathbf{x}}_t^{l+1}-\frac{\mathbf{v}_t^l}{\rho_t^l}),\\ &\mathbf{v}_t^{l+1}=\mathbf{v}_t^l+\rho_t^l(\mathbf{z}_t^{l+1}-\bar{\mathbf{x}}_t^{l+1}),\end{aligned} \tag{6.6}$$

where γ_t are relaxation parameters, ρ_t are augmented Lagrangian penalty parameters, and $\mathbf{v}_t$ are the vectors of Lagrange multipliers. The relaxation for each vector involved takes the form

$$\text{relax}(\gamma_t^l,\mathbf{x}^{l+1},\mathbf{z}^l)=\begin{cases}\gamma_i^l(\mathbf{A}_i \quad \mathbf{0})\mathbf{x}^{l+1}+(1-\gamma_i^l)\mathbf{z}_i^l & \text{for } i=1,2,\cdots,p\\ \gamma_{j+p}^l(\mathbf{0} \quad \mathbf{B}_j)\mathbf{x}^{l+1}+(1-\gamma_{j+p}^l)\mathbf{z}_{j+p}^l & \text{for } j=1,2,\cdots,q\\ \gamma_{k+p+q}^l(\mathbf{C}_k \quad \mathbf{C}_k)\mathbf{x}^{l+1}+(1-\gamma_{k+p+q}^l)\mathbf{z}_{k+p+q}^l & \text{for } k=1,2,\cdots,r\\ \gamma_{p+q+r+1}^l(\mathbf{I}_N \quad \mathbf{I}_N)\mathbf{x}^{l+1}+(1-\gamma_{p+q+r+1}^l)\mathbf{z}_{p+q+r+1}^l\end{cases},$$

and the proximal mappings are given by

$$\text{prox}_{t,\rho_t}(\bar{\mathbf{x}}_t^{l+1}-\frac{\mathbf{v}_t^l}{\rho_t^l})=\begin{cases}P_{G_i}(\bar{\mathbf{x}}_i^{l+1}-\frac{\mathbf{v}_i^l}{\rho_i^l}) & \text{for } i=1,2,\cdots,p\\ P_{E_j}(\bar{\mathbf{x}}_{j+p}^{l+1}-\frac{\mathbf{v}_{j+p}^l}{\rho_{j+p}^l}) & \text{for } j=1,2,\cdots,q\\ P_{F_k}(\bar{\mathbf{x}}_{k+p+q}^{l+1}-\frac{\mathbf{v}_{k+p+q}^l}{\rho_{k+p+q}^l}) & \text{for } k=1,2,\cdots,r\\ \text{prox}_{\|\cdot\|_2^2,\rho_{p+q+r+1}}=\frac{\mathbf{m}+\rho_{p+q+r+1}\left(\bar{\mathbf{x}}_{p+q+r+1}^{l+1}-\frac{\mathbf{v}_{p+q+r+1}^l}{\rho_{p+q+r+1}^l}\right)}{1+\rho_{p+q+r+1}}.\end{cases}$$

The above proximal maps are thus projections onto 'simple' sets that we can compute in closed form, and one proximal map that corresponds to the squared 2-norm. All linear operators are now moved to a single linear system, that is given by the block-matrix

$$\mathbf{Q}=\begin{pmatrix}\sum_{i=1}^{p}\rho_i\mathbf{A}_i^\top\mathbf{A}_i+\sum_{k=1}^{r}\rho_k\mathbf{C}_k^\top\mathbf{C}_k+\rho_s\mathbf{I}_N & \sum_{k=1}^{r}\rho_k\mathbf{C}_k^\top\mathbf{C}_k+\rho_s\mathbf{I}_N\\ \sum_{k=1}^{r}\rho_k\mathbf{C}_k^\top\mathbf{C}_k+\rho_s\mathbf{I}_N & \sum_{j=1}^{q}\rho_j\mathbf{B}_j^\top\mathbf{B}_j+\sum_{k=1}^{r}\rho_k\mathbf{C}_k^\top\mathbf{C}_k+\rho_s\mathbf{I}_N\end{pmatrix},$$

and the right-hand-side $\mathbf{b}$ that is constructed at every iteration as

$$\begin{aligned}\mathbf{b}^l = &\sum_{i=1}^{p} (\mathbf{A}_i \quad \mathbf{0})^\top (\rho_i^l \mathbf{z}_i^l + \mathbf{v}_i^l) + \sum_{j=1}^{q} (\mathbf{0} \quad \mathbf{B}_j)^\top (\rho_{j+p}^l \mathbf{z}_{j+p}^l + \mathbf{v}_{j+p}^l) \\ &+ \sum_{k=1}^{r} (\mathbf{C}_{k+p+q} \quad \mathbf{C}_{k+p+q})^\top (\rho_{k+p+q}^l \mathbf{z}_{k+p+q}^l + \mathbf{v}_{k+p+q}^l) \\ &+ (\mathbf{I}_N \quad \mathbf{I}_N)^\top (\rho_{p+q+r+1}^l \mathbf{z}_{p+q+r+1}^l + \mathbf{v}_{p+q+r+1}^l).\end{aligned}$$

The final results is the projection of a vector $\mathbf{m}$ onto the the intersection of an intersection and a sum of constraint sets. Each of the components of the sum can also be an intersection itself. The projected vector is given by the left or right-hand side of $(\mathbf{I}_N \quad \mathbf{I}_N)\mathbf{x} = \mathbf{z}_{p+q+r+1}$, which are equal if the problem is solved to completion. The above iterations given by (6.6) summarize the approach from [55], and can be shown to be equivalent to ADMM, and its variant SDMM [52, 53].

7. Computational Considerations and Cost

The following is a discussion of the primary factors that determine the computational cost of using constraints on the output of a neural network.

Data type. Each iteration of Algorithm 1 requires one projection for each network output-channel that is constrained. Therefore, the constraints are more suitable for datasets with a small number of classes and few input data (examples).

Constraint type. The computation time for projections strongly depends on the type of constraints. The computational cost is the lowest for projections onto a single set for which we know the projection in closed form, such as bound or cardinality constraint sets that do not include a non-orthogonal linear operator. Sets without closed-form projections require computation via another iterative algorithm at a higher computational cost. Projections onto intersections and sums of constraint sets require the most computational work. Not much can be said in general about the computational cost of projecting onto the latter two sets because it heavily depends on the number and type of constraint sets that form the intersection or sum.

Initial point. The training of a neural network requires the projection of a vector that changes every gradient descent step. During the training phase, the cost of projecting drops dramatically when the network output becomes (almost) feasible. If the iterative algorithm tasked with computing the projection detects that input is feasible already, no further computational work is required.

While training a network using constraint sets is typically more time-consuming than training using just labels, the examples in the next section show that the constraints allow for training with fewer data and fewer annotation points. This greatly reduces the time and cost of obtaining data and annotation. Even more important than computational time is the ability to obtain a pixel-level segmentation using constraints when the provided annotation is insufficient. This is especially important when almost no way exists to obtain more annotation, e.g., ground truth in remote sensing and geophysical studies.

8. Examples

The examples are reproducible using Julia [56] software available at https://github.com/PetersBas/NNNCVXF. The emphasis of the examples is on data from the imaging sciences that often contain just a few or a single example, possibly with some partial annotation and some prior knowledge. While the notationally compact ResNet guided the algorithm details section, the examples use a fully invertible (or reversible) hyperbolic network [45, 57], which comes with longer and more involved expressions. Invertibility removes the requirement to store all network states for gradient computations. As a result, reversible networks can train on large-scale data that would otherwise not fit on a standard GPU. This is useful when working with applications where the high-level information is valid only on larger scales.

The reversible hyperbolic network follows the recursion $\mathbf{y}_j = 2\mathbf{y}_{j-1} - \mathbf{y}_{j-2} - h^2\mathbf{K}_j^\top \sigma(\mathbf{K}_j\mathbf{y}_{j-1})$, which shows that the network structure does not alter Algorithm 1 fundamentally because the final state appears in an explicit form. All experiments use the artificial timestep $h = 0.2$, convolutional kernels of size 3×3 or $3 \times 3 \times 3$, and use the implementation available via [58].

8.1. **Point annotations for one of the classes only.** For many remote sensing applications where it is challenging to obtain labels, there is a need to develop methods that can learn from few point annotations. This section shows that network-output constraints enable learning in the extreme case that not a single point annotation is available for one of the two classes. In this situation, we cannot learn by using standard loss functions to minimize the difference between prediction and labels because assigning one class to all predictions yields a perfect prediction of the known labels.

To 'compensate' for the lack of labels for one of the two classes, we assume some knowledge on the total area that a class should occupy in the prediction. For the other class, we assume a small number of point annotations. These pieces of information lead to the feasibility problem

$$\min_{\mathbf{K}} \frac{1}{2}\|P_{D_1}(g(\mathbf{K},\mathbf{d})_1) - g(\mathbf{K},\mathbf{d})_1\|_2^2 + \frac{1}{2}\|P_{D_2}(g(\mathbf{K},\mathbf{d})_2) - g(\mathbf{K},\mathbf{d})_2\|_2^2 \tag{8.1}$$

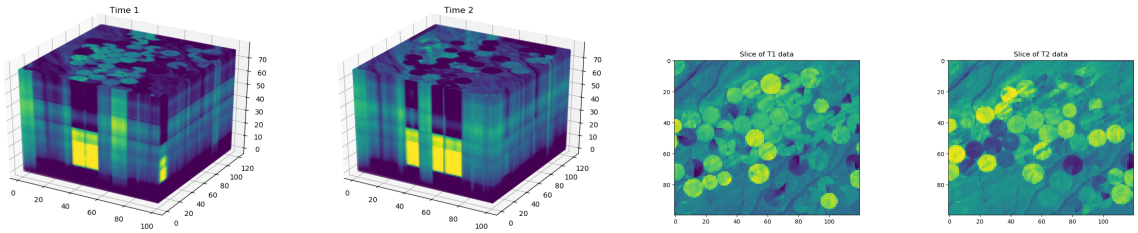
$$\text{where } D_1 = \{\mathbf{x} \,|\, \text{card}(\mathbf{x}) \leq a_2\} \bigcap \{\mathbf{x} \,|\, \mathbf{x}_\Omega = 1\},$$

$$D_2 = \{\mathbf{x} \,|\, \text{card}(\mathbf{x}) \leq (1-a_1)\} \bigcap \{\mathbf{x} \,|\, \mathbf{x}_\Omega = 0\},$$

where a_2 and $1 - a_1$ are the upper bounds on the 'size' of class numbers one and two, respectively. It is still assumed that the network output for each pixel sums to one over the channels $g(\mathbf{K},\mathbf{d})_1$ and $g(\mathbf{K},\mathbf{d})_2$. The set Ω contains the locations of the known point annotations for class number one. Two remote sensing applications demonstrate this formulation's applicability and the fact that we can solve this non-convex feasibility problem with sufficient accuracy.

8.1.1. *Time-lapse hyperspectral land-use change detection.* The goal is to create a 2D change map of a piece of the earth from two 3D hyperspectral datasets, collected at different times [59], see Figure 1. While it is relatively easy to spot the center-pivot farm fields, the challenge is that only a subset of those fields changed land-use. Domain-experts or ground truth can provide annotation.

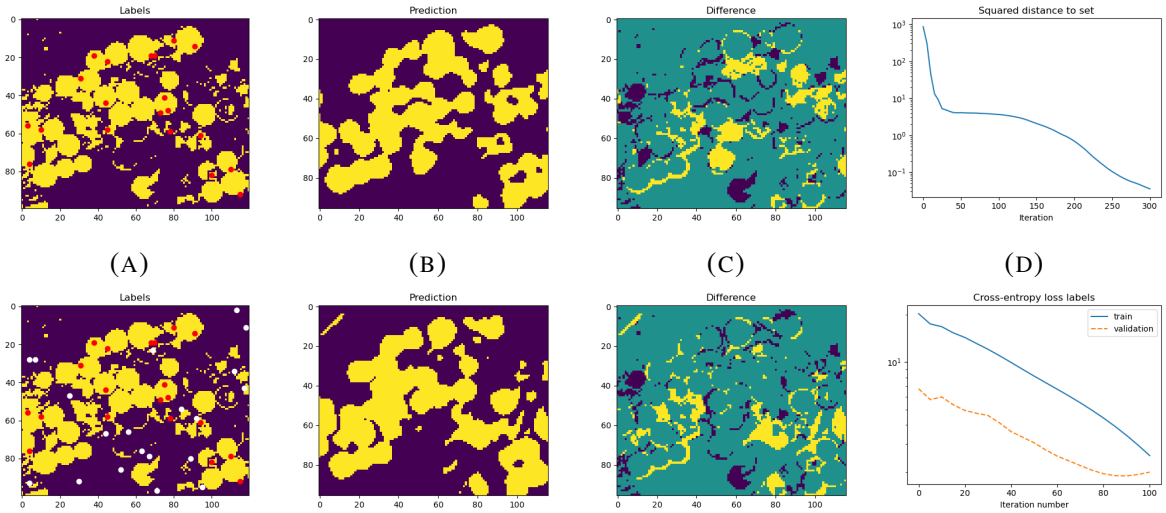
This dataset contains just one example with 20 point annotations for class one and no annotations for class two. Yet, it is still possible to obtain reasonably accurate predictions using knowledge on the surface area that is occupied by the two classes (no-change / change).



(A) (B) (C) (D)

FIGURE 1. (A) & (B): Two hyperspectral datasets recorded at different times for land-use change detection. (C) & (D): slices from (A,B) for a single frequency.

The true land-use change surface-area is $\approx 34\%$. We use the loose bounds $25\% \leq$ area(class 1) $\leq 45\%$. This translates to the cardinality constraints (6.2) with $a_1 = 0.25N$ and $a_2 = 0.45N$ in problem formulation (8.1), where N is the number of pixels. Gradient descent, as in Algorithm 1 finds a feasible point. The network is 24 layers deep, with 12 channels per layer and 144 convolutional kernels per layer.



(A) (B) (C) (D)

(E) (F) (G) (H)

FIGURE 2. Top row: (A) Provided annotations, experiment uses 20 labeled pixels from one of the two classes for training, indicated by red dots. (B) our prediction based on 20 point annotations and constraints on the area for the classes. (C) Difference. Most of the errors are boundary effects, as well as a few incorrectly identified fields. (D) the distance to the intersection of constraints sets during training. Bottom row: Similar experiment as the top row, except that no prior knowledge is used. Instead, there are 20 point annotations for each class; see red and white dots in (E), respectively.

The top row in Figure 2 shows the provided annotations, the prediction based on prior knowledge and 20 point annotations for one of the two classes, and the difference. Except for a few errors, our method was able to capture the subtle differences over time. To add some context to

this result, the bottom row of Figure 2 shows the result of using 20 point annotations for each class (40 in total) and no prior information on the area that each class can occupy. Using just point annotations makes the training is susceptible to overfitting. Therefore, each class can use 15 pixels for training and 5 for validation. The prediction quality with and without prior knowledge is similar. This observation indicates, for this problem, that high-level prior knowledge can compensate for not having a single point annotation for one of the two classes.

8.1.2. *Multi-modality subsurface sensing.* Figure 3 shows various types of remote sensing data and geological maps. Besides the four physical observable maps, the rock age and rock type maps are converted to 52 separate maps (one for each age/type present or not), totaling $52+4$ data input channels. The goal is to obtain a map of the aquifers (yes/no aquifer present in the subsurface) in an area that is roughly 40% of Arizona, see Figure 4. The labels from [60] were obtained via a combination of ground-truth observations, remote sensing, geophysical data, and expert interpretation. Therefore, these labels are not all ground-truth, and the goal is to investigate if a neural network can reproduce work by domain-experts. The neural network is provided with 100 point annotations for only one of the two classes collected in the set Ω, supplemented with an estimate of the total area of aquifers being between 50% and 65%, i.e., we find a feasible point via (8.1) with gradient computation as in Algorithm 1.

The prediction in Figure 4 shows a neural network can replicate domain expert's work using limited prior knowledge and few point annotations for just one of the classes. The differences are minor along the boundaries, as well as a few incorrectly classified small patches.

This example shows that prior knowledge helps in situations where annotating one of the classes is much more difficult than annotating the others.

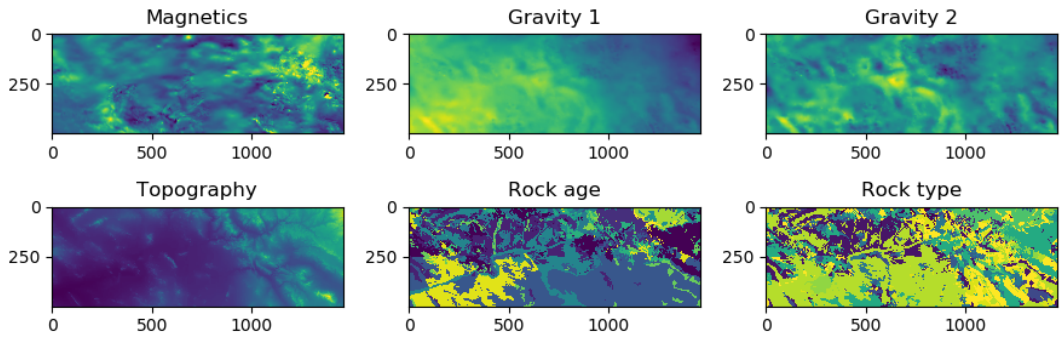


FIGURE 3. Data for the multi-modality remote sensing example. The rock age and rock type maps are converted to 52 separate maps that indicate if the rock age/type is present or not.

8.2. **Training with insufficient point-annotation and constraints.** This example is different from the previous ones because more than one image is available for training. The small dataset has 47 images from one of the CamVid videos [61] for training, and 15 for validation, see Figure 5. While the standard CamVid dataset comes with fully annotated images, we use eight annotated pixels per class per image to simulate limited annotation time/effort. The task is limited to segmenting the vehicles and background.

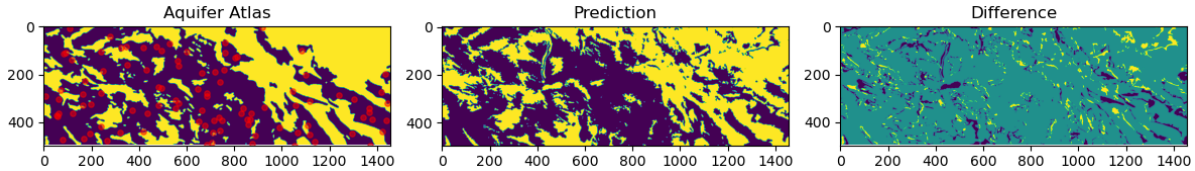
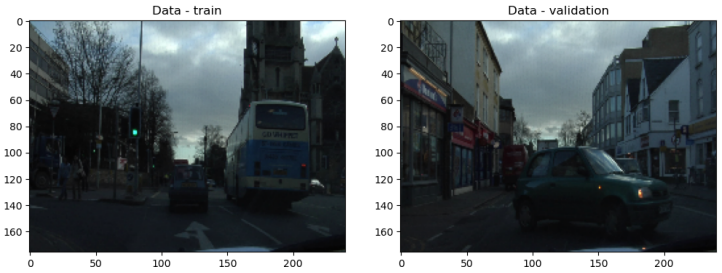


FIGURE 4. Left: aquifer map, based on data and manual interpretation. Red dots represent the annotation for training. Note that there is annotation for one of the two classes only. Middle: prediction. Right: difference between prediction from point annotations and the full label map.



(A) (B)

FIGURE 5. Examples of one training (A) and validation (B) image from the CamVid dataset.

To motivate why prior knowledge may be necessary when the annotation is insufficient, see Figure 6 for the results of training a network on the point annotations only. The Hyperbolic neural-network has 14 layers with 48 channels and 48^2 convolutional kernels per layer. The figures show that while fitting the training point annotations using the cross-entropy loss is easy, the validation loss starts increasing right after training begins. The predictions from each output channel reveal that the vehicles are detected, but many other parts of the image also contain incorrectly classified coherent and incoherent structures. The intersection over union (IoU) assessment of the classified pixels also verifies that training hardly increases the quality of the segmentations.

After concluding that training a network on just eight point-annotations per class per image is not sufficient (Figure 6), we add anisotropic total-variation constraints on just the first network output channel, i.e.,

$$\min_{\mathbf{K}} \sum_{i=1}^{n_{\text{examples}}} l(\mathbf{Q}^i g(\mathbf{K},\mathbf{d}^i),\mathbf{c}^i) + \frac{\alpha}{2}\|P_{C^i}(g(\mathbf{K},\mathbf{d}^i)_1) - g(\mathbf{K},\mathbf{d}^i)_1\|_2^2, \text{ where } C^i = \{\mathbf{x}\,|\,\|\mathbf{x}\|_{\text{TV}} \leq \lambda^i\},$$

where $\|\cdot\|_{\text{TV}}$ is the anisotropic total-variation 'norm' and the superscript indexes over the number of examples. This time we do not encode the point annotations as bound constraints but keep them as a separate cross-entropy loss function so we can better compare the results to training without constraints. It is not realistic to assume that the total variation of the true segmentation is known. Instead, each image in the datasets gets an independent estimate of the anisotropic total-variation drawn as a uniformly distributed number between 66% and 133% of the true total variation. More comprehensive estimates of the TV could, in principle, be obtained via

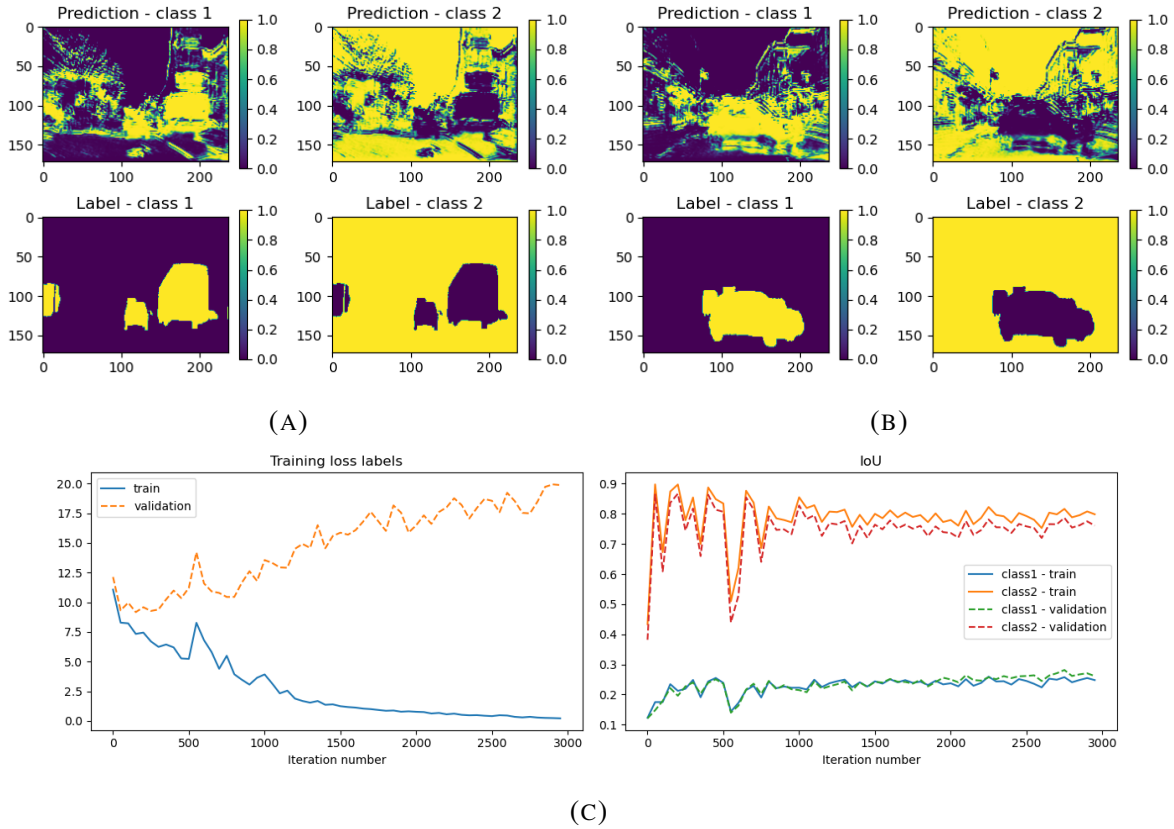


FIGURE 6. (A) and (B) are examples of the training and validation network outputs, respectively, when training on eight point-annotations per class per image for a subset of the CamVid dataset (no prior knowledge used). (C) Observe that the validation loss starts increasing soon after training begins, while the IoU of the segmentation still increases slightly. While the vehicles mostly get the correct class assigned, there is a lot of other coherent and incoherent structure classified as a vehicle. Compare to Figure 7.

cross-validation using the validation data. The optimization strategy is as follows: training starts using just point-annotations for the first 2000 iterations because it is computationally cheaper than using constraints as well. After iteration 2000, the constraints are activated using a small α, followed by $\alpha \leftarrow 5\alpha$ every 400 iterations. The distance to the intersection (as summed over all training images) starts to decrease after the constraints are activated, see Figure 7. At the same time, the validation loss starts to decrease as well. The IoU computed from the validation segmentations also indicates that constraints increase the accuracy of the prediction. The results using inexact TV prior knowledge (Figure 7) are a significant improvement compared to point annotations only (Figure 6).

8.3. **Single-image segmentation with coherent corruption.** For this final example, consider single-image segmentation problems. No point-annotations or pre-trained networks are available. Such a situation is of interest when manually creating pixel-level annotations is challenging or intractable, for instance, when the image is corrupted.

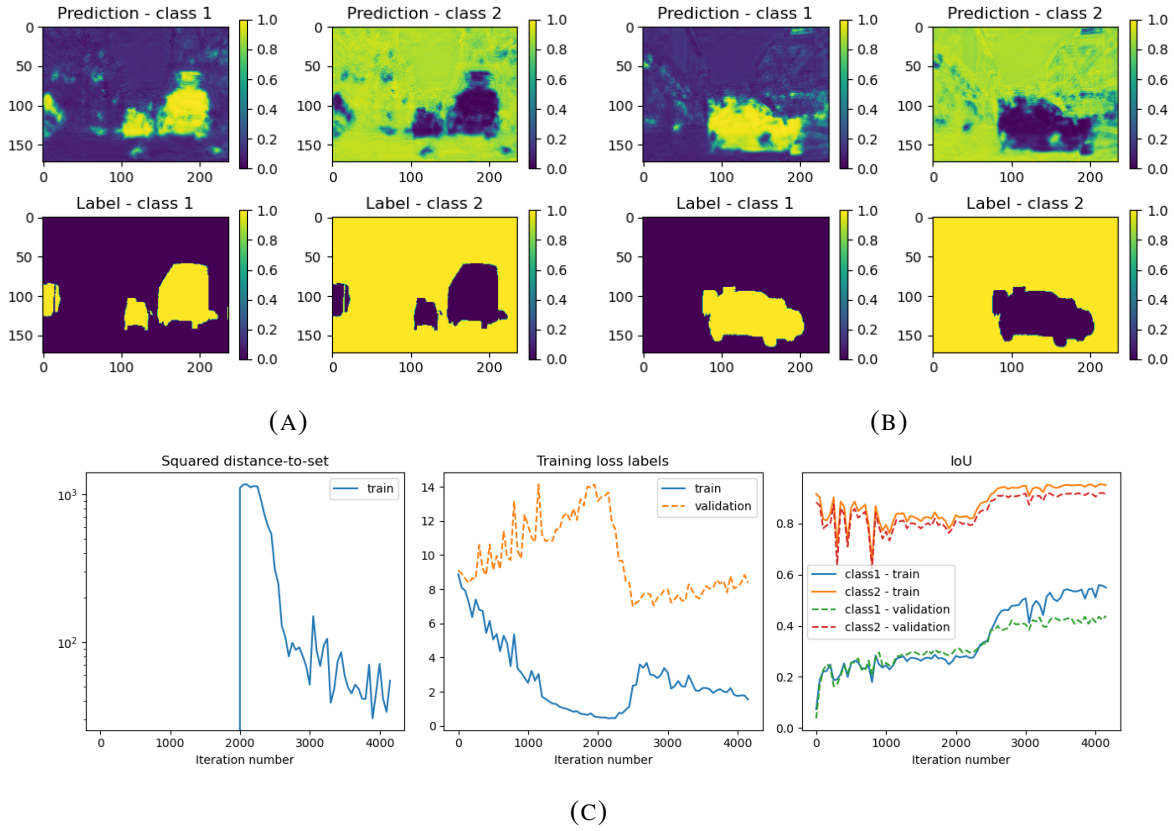


FIGURE 7. (A) and (B) are examples of the training and validation network outputs respectively, when training a neural network using point-annotations supplemented with approximate anisotropic total-variation information. (C) Achieved IoU is much higher compared to training on point-annotations only, compare to Figure 6.

To train a neural network using weak supervision, we can ask the network to highlight an object inside the bounding box that is different from anything outside of the bounding box, has a simple shape, and occupies between 20% and 50% of the pixels in the segmentation. This query translates to the following constraint sets: *1*) a loose bounding box as shown in Figure 8 that indicates an object is present in the box defined by the pixels in Ω: $F_{1,1} = \{\mathbf{x} \,|\, 0 \leq \mathbf{x}_\Omega \leq 1, \mathbf{x}_\Psi = 0\}$, and similarly for the other output channel $F_{2,1} = \{\mathbf{x} \,|\, 0 \leq \mathbf{x}_\Omega \leq 1, \mathbf{x}_\Psi = 1\}$; *2*) a rough estimate of the object's minimum and maximum size, implemented via cardinality constraints $F_{1,2} = \{\mathbf{x} \,|\, \text{card}(\mathbf{x}) \leq a_2\}$, and similarly for the second output channel $F_{2,2} = \{\mathbf{x} \,|\, \text{card}(\mathbf{x}) \leq (1 - a_1)\}$; *3*) the object has a relatively 'simple' shape, here we select a Minkowski set that describes the segmentation as the sum of monotonically increasing and decreasing functions, both vertically and laterally, i.e., the sets $G_1 = \{\mathbf{u} \,|\, 0 \leq (D_z \otimes I_x)\mathbf{u} \leq \infty\}$, $G_2 = \{\mathbf{u} \,|\, 0 \leq (I_z \otimes D_x)\mathbf{u} \leq \infty\}$, $E_1 = \{\mathbf{v} \,|\, -\infty \leq (D_z \otimes I_x)\mathbf{v} \leq 0\}$ and $E_2 = \{\mathbf{v} \,|\, -\infty \leq (I_z \otimes D_x)\mathbf{v} \leq 0\}$, where $\otimes$ denotes the Kronecker product and D_x a finite difference matrix in the 'x' coordinate. This constraint type applies to both network-output channels. This particular Minkowski set, heuristically, includes objects that roughly look like a 'blob', where the monotonic nature of the components does not allow many transitions between classes. In this description of constraint sets, sets F_k apply

to the network output, which is the sum of components $\mathbf{u}$ and $\mathbf{v}$. The sets G_i apply to component $\mathbf{u}$, and E_j apply to $\mathbf{v}$. To prevent the components from shifting up or down arbitrarily, additional bound constraints on each component are required: $G_3 = \{\mathbf{u} \mid -1 \leq \mathbf{u} \leq 1\}$ and $E_3 = \{\mathbf{v} \mid -1 \leq \mathbf{v} \leq 1\}$. Combining the constraints on the network output and its two components results in the sets $M_1 = \left(\bigcap_{k=1}^{2} F_{1,k}\right) \bigcap \left(\bigcap_{i=1}^{3} G_i + \bigcap_{j=1}^{3} E_j\right)$ for network output channel one, and $M_2 = \left(\bigcap_{k=1}^{2} F_{2,k}\right) \bigcap \left(\bigcap_{i=1}^{3} G_i + \bigcap_{j=1}^{3} E_j\right)$ for output channel two. To summarize, a feasible point corresponding to

$$\min_{\mathbf{K}} \frac{1}{2}\|P_{M_1}(g(\mathbf{K},\mathbf{d})_1) - g(\mathbf{K},\mathbf{d})_1\|_2^2 + \frac{1}{2}\|P_{M_2}(g(\mathbf{K},\mathbf{d})_2) - g(\mathbf{K},\mathbf{d})_2\|_2^2$$

$$\text{where } M_1 = \Big(\bigcap_{k=1}^{2} F_{1,k}\Big) \bigcap \Big(\bigcap_{i=1}^{3} G_i + \bigcap_{j=1}^{3} E_j\Big),\ M_2 = \Big(\bigcap_{k=1}^{2} F_{2,k}\Big) \bigcap \Big(\bigcap_{i=1}^{3} G_i + \bigcap_{j=1}^{3} E_j\Big)$$

should be interpreted as the neural network finding an object inside the bounding box, which is different from anything outside of the bounding box, has a simple shape, and occupies between 20% and 50% of the pixels in the segmentation.

The experiment employs a ten layer hyperbolic neural-network with nine channels and 81 convolutional 3×3 kernels per layer. This network is trained from scratch for each image. The final result of solving the above problem is a set of network parameters that output a segmentation for just the considered input image.

Figure 8 displays the images and results, including a comparison to the successful segmentation method GrabCut [62] via the OpenCV implementation [63]. GrabCut can also operate on a single image without learning from other images.

Figure 8 illustrates that using just constraints to train the neural network leads to similar or better performance on the original image and performs better on the images corrupted with random missing lines. No knowledge of the number of missing lines or their locations was used.

The advantage of training a network to find a feasible point is the adaptability of the proposed method: adding a constraint set when additional prior knowledge becomes available does not modify the problem formulation or the neural network. This feature is an advantage over methods not based on learning, which are often difficult to adapt to new information or experimental settings. For this particular example, the involved constraint sets do not allow for many changes between object and background classes when looking vertically or horizontally. This aims to mitigate the detrimental effects of missing pixels on the segmentation. Therefore the proposed method with the selected constraint sets acts as a joint inpainting-segmentation method.

9. CONCLUSIONS

Weak-supervision techniques inject quantitative prior-information about the expected image segmentation into the neural-network training problem to compensate for the lack of annotation. This work introduced a general framework that can include constraints on many image properties, and also work with various more complicated types of prior knowledge expressed through intersections and sums of constraint sets. The proposed framework includes constraints on the network output via a novel training algorithm based on point-to-set distance functions. This approach requires just the projection operator so that different constraints do not require translation into custom penalty functions and their derivatives. Moreover, multiple pieces of prior

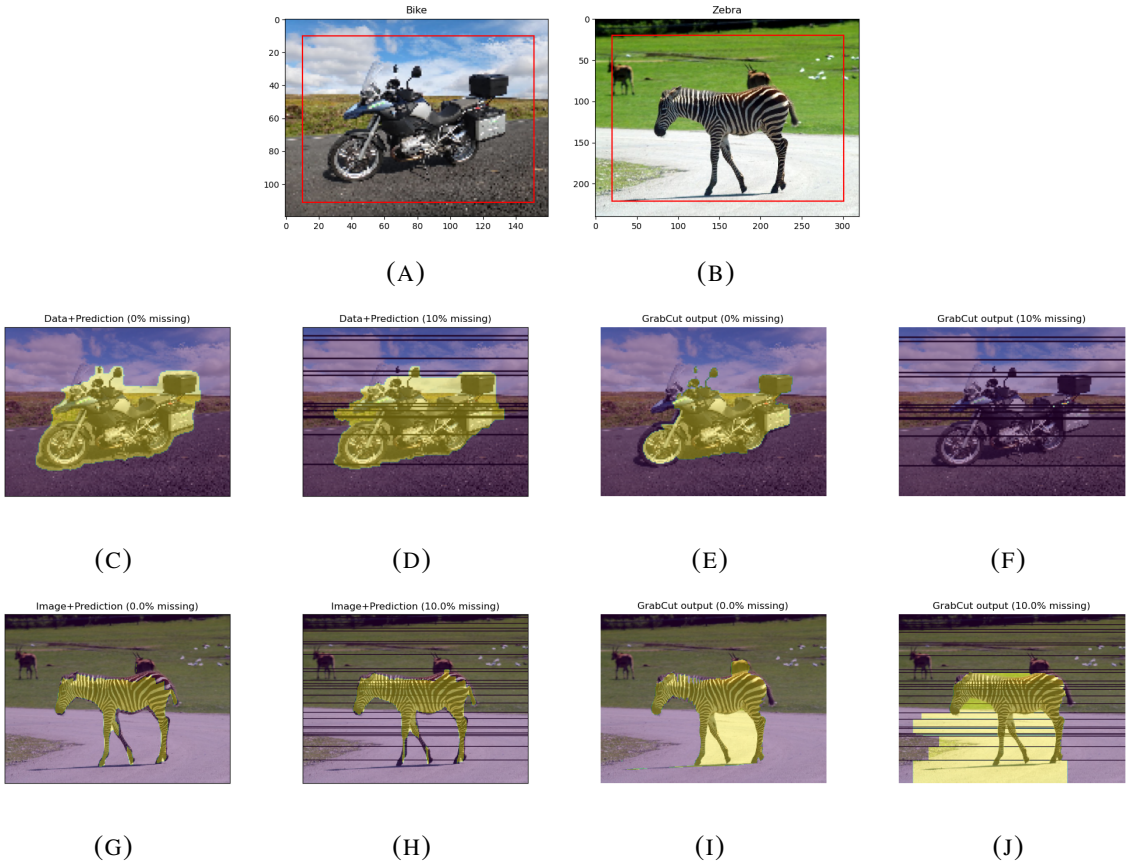


FIGURE 8. (A,B) Images with bounding boxes (red rectangles); (C,D,G,H) segmentations obtained using a neural network trained with constraints, overlaid on the data; (E,F,I,J) results from Grabcut (note that the GrabCut segmentation for 10% missing rows just highlights a tiny area near the rear of the bike).

knowledge do not introduce a multitude of difficult-to-tune penalty parameters. A quick look at the Lagrangian shows that the constraints implemented via a distance function fit seamlessly into training networks via backpropagation. Examples illustrate that constraints compensate for missing annotation for a range of segmentation problems in the imaging sciences: data where one or more classes have no annotation, corrupted images with bounding box information, object size, and shape information, and finally, constraints help the segmentation of a small dataset with very few point-annotations per class per image.

REFERENCES

[1] A. Bearman, O. Russakovsky, V. Ferrari, et al., What's the point: Semantic segmentation with point supervision, In: B. Leibe, J. Matas, N. Sebe and M. Welling, (eds.) Computer Vision – ECCV 2016, pp. 549–565, Springer, 2016.

[2] B. Peters, J. Granek, E. Haber, Multiresolution neural networks for tracking seismic horizons from few training images, Interpretation, 7 (2019), SE201-SE213.

[3] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, et al., 3d u-net: Learning dense volumetric segmentation from sparse annotation, In: S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal

and W. Wells, (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016, pp. 424-432, Springer, 2016.

[4] J. Dai, K. He, J. Sun, Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation, The IEEE International Conference on Computer Vision (ICCV), 2015.

[5] M. Rajchl, M. C. Lee, O. Oktay, et al., Deepcut: Object segmentation from bounding box annotations using convolutional neural networks, IEEE Trans. Medical imaging, 36 (2016), 674-683.

[6] A. Khoreva, R. Benenson, J. Hosang, et al., Simple does it: Weakly supervised instance and semantic segmentation, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[7] H. Kervadec, J. Dolz, S. Wang, et al., Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision, Medical Imaging with Deep Learning, 2020.

[8] G. Papandreou, L.-C. Chen, K. P. Murphy, et al., Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation, In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), IEEE Computer Society, ICCV 15, pp. 1742-1750, 2015.

[9] F. Saleh, M. S. Aliakbarian, M. Salzmann, et al., Built-in foreground/background prior for weakly-supervised semantic segmentation, In: B. Leibe, J. Matas, N. Sebe and M. Welling, (eds.) Computer Vision – ECCV 2016, pp. 413–432, Springer, 2016.

[10] T. Chan, L. Vese, Active contours without edges, IEEE Trans. Image Process. 10 (2001), 266-277.

[11] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, Commun. Pure Appl. Math. 42 (1989), 577-685.

[12] J. C. Platt, A. H. Barr, Constrained differential optimization, In: Proceedings of the 1987 International Conference on Neural Information Processing Systems, pp. 612–621, MIT Press, Cambridge, 1987.

[13] K. Ganchev, J. Graça, J. Gillenwater, et al., Posterior regularization for structured latent variable models, J. Mach. Learn. Res. 11 (2010), 2001-2049.

[14] K. Bellare, G. Druck, A. McCallum, Alternating projections for learning with expectation constraints, In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 09, pp. 43–50, AUAI Press, Arlington, Virginia, 2009.

[15] G. S. Mann, A. McCallum, Generalized expectation criteria for semi-supervised learning with weakly labeled data, J. Mach. Learn. Res. 11 (2010), 955-984.

[16] Y. Nandwani, A. Pathak, Mausam, et al., A primal dual formulation for deep learning with constraints, In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox and R. Garnett, (eds.) Advances in Neural Information Processing Systems 32, pp. 12157–12168, Curran Associates, 2019.

[17] D. Pathak, P. Krahenbuhl, T. Darrell, Constrained convolutional neural networks for weakly supervised segmentation, In: The IEEE International Conference on Computer Vision (ICCV), 2015.

[18] P. Márquez-Neila, M. Salzmann, P. Fua, Imposing hard constraints on deep networks: Promises and limitations, arXiv preprint arXiv: 1706.02025, 2017.

[19] H. Kervadec, J. Dolz, M. Tang, et al., Constrained-cnn losses for weakly supervised segmentation, Medical Image Anal. 54 (2019), 88-99.

[20] F. J. Herrmann, A. Siahkoohi, G. Rizzuti, Learned imaging with constraints and uncertainty quantification, arXiv preprint arXiv: 1909.06473, 2019.

[21] A. BenTaieb, G. Hamarneh, Topology aware fully convolutional networks for histology gland segmentation, In: S. Ourselin, L. Joskowicz, M. R. Sabuncu, et al. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016, pp. 460–468, Springer, 2016.

[22] B. Peters, E. Haber, J. Granek, Neural networks for geophysicists and their application to seismic data interpretation, The Leading Edge, 38 (2019), 534-540.

[23] Z. Mirikharaji, G. Hamarneh, Star shape prior in fully convolutional networks for skin lesion segmentation, In: A. F. Frangi, J. A. Schnabel, C. Davatzikos, et al. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2018, pp. 737-745, Springer, 2018.

[24] F. Jia, J. Liu, X.-C. Tai, A regularized convolutional neural network for semantic image segmentation, Anal. Appl. 19 (2021), 147-165.

[25] J. Liu, X. Wang, X.-C. Tai, Deep convolutional neural networks with spatial regularization, volume and star-shape priori for image segmentation, arXiv preprint arXiv: 2002.03989, 2020.

[26] S. Lunz, O. Öktem, C.-B. Schönlieb, Adversarial regularizers in inverse problems, In: S. Bengio, H. Wallach, H. Larochelle, et al. (eds.) Advances in Neural Information Processing Systems, vol. 31, Curran Associates, 2018.

[27] H. Heaton, S. W. Fung, A. T. Lin, et al., Projecting to manifolds via unsupervised learning, arXiv preprint arXiv: 2008.02200, 2020.

[28] J. Prost, A. Houdard, A. Almansa, et al., Learning local regularization for variational image restoration, arXiv preprint arXiv: 2102.06155, 2021.

[29] J.-F. Aujol, G. Gilboa, T. Chan, et al., Structure-texture image decomposition-modeling, algorithms, and parameter selection, Int. J. Comput. Vision 67 (2006), 111-136.

[30] S. Osher, A. Solé, L. Vese, Image decomposition and restoration using total variation minimization and the h1, Multiscale Modeling Simul. 1 (2003), 349-370.

[31] H. Schaeffer, S. Osher, A low patch-rank interpretation of texture, SIAM J. Imaging Sci. 6 (2013), 226-262.

[32] J. L. Starck, M. Elad, D. L. Donoho, Image decomposition via the combination of sparse representations and a variational approach, IEEE Trans. Image Process. 14 (2005), 1570-1582.

[33] S. Ono, T. Miyata, I. Yamada, Cartoon-texture image decomposition using blockwise low-rank texture characterization, IEEE Trans. Image Process. 23 (2014), 1128-1142.

[34] I. Daubechies, G. Teschke, Variational image restoration by means of wavelets: Simultaneous decomposition, deblurring, and denoising, Appl. Comput. Harmon. Anal. 19 (2005), 1-16.

[35] E. J. Candès, X. Li, Y. Ma, et al., Robust principal component analysis? J. ACM, 58 (2011), 11.

[36] H. Gao, J.-F. Cai, Z. Shen, et al., Robust principal component analysis-based four-dimensional computed tomography, Phys. Medicine Biolo. 56 (2011), 3181.

[37] H. Gao, H. Yu, S. Osher, et al., Multi-energy ct based on a prior rank, intensity and sparsity model (prism), Inverse Probl. 27 (2011), 115012.
[38] J.-B. Hiriart-Urruty, C. Lemaréchal, Fundamentals of Convex Analysis, Springer, 2012.
[39] J.-B. Hiriart-Urruty, C. Lemarechal, Convex Analysis and Minimization Algorithms I: Fundamentals, vol. 305, Springer, 1996.
[40] C. Byrne, Iterative oblique projection onto convex sets and the split feasibility problem, Inverse Probl. 18 (2002), 441-453.
[41] Y. Censor, T. Elfving, N. Kopf, et al., The multiple-sets split feasibility problem and its applications for inverse problems, Inverse Probl. 21 (2005), 2071-2084.
[42] M. Brooke, Y. Censor, A. Gibali, Dynamic string-averaging cq-methods for the split feasibility problem with percentage violation constraints arising in radiation therapy treatment planning, Intl. Trans. Oper. Res. DOI: 10.1111/itor.12929.
[43] T. Lu, L. Zhao, H. He, A multi-view on the cq algorithm for split feasibility problems: From optimization lens, J. Appl. Numer. Optim, 2 (2020), 387-399.
[44] K. He, X. Zhang, S. Ren, et al., Deep residual learning for image recognition, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 770–778.
[45] B. Chang, L. Meng, E. Haber, et al., Reversible architectures for arbitrarily deep residual neural networks, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.
[46] Y. Lecun, A theoretical framework for back-propagation, In: D. Touretzky, G. Hinton and T. Sejnowski, (eds.) Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA, Morgan Kaufmann, pp. 21–28, 1988.
[47] A. Beck, First-Order Methods in Optimization, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.
[48] B. Peters, F. J. Herrmann, Algorithms and software for projections onto intersections of convex and non-convex sets with applications to inverse problems, arXiv preprint arXiv: 1902.09699, 2019.
[49] S. Boyd, N. Parikh, E. Chu, et al., Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2011), 1-122.
[50] J. Eckstein, W. Yao, Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives, Pac. J. Optim. 11 (2015), 619-644.
[51] M. V. Afonso, J. M. Bioucas-Dias, M. A. T. Figueiredo, An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems, IEEE Trans. Image Process. 20 (2011), 681-695.
[52] P. L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, In: H. H. Bauschke, R. S. Burachik, et al. (eds.) Fixed-Point Algorithms for Inverse Problems in Science and Engineering, vol. 49 of Springer Optimization and Its Applications, pp. 185-212, Springer, New York, 2011.
[53] S. Kitic, L. Albera, N. Bertin, et al., Physics-driven inverse problems made tractable with cosparse regularization, IEEE Trans. Signal Process. 64 (2016), 335-348.
[54] Z. Xu, M. A. T. Figueiredo, X. Yuan, et al., Adaptive relaxed admm: Convergence theory and practical implementation, In: The IEEE Conference on Computer Vision and Pattern

Recognition (CVPR), 2017.
[55] B. Peters, F. J. Herrmann, Generalized minkowski sets for the regularization of inverse problems, arXiv preprint arXiv: 1903.03942, 2019.
[56] J. Bezanson, S. Karpinski, V. B. Shah, et al. Julia: A fast dynamic language for technical computing, arXiv.org, 2012.
[57] K. Lensink, E. Haber, B. Peters, Fully hyperbolic convolutional neural networks, arXiv preprint arXiv: 1905.10484, 2019.
[58] P. A. Witte, M. Louboutin, A. Siahkoohi, et al., slimgroup/InvertibleNetworks.jl: Jacobians and adjoint Jacobians of layers and networks, 2020.
[59] M. Hasanlou, S. T. Seydi, Hyperspectral change detection: an experimental comparative study, Int. J. Remote Sensing 39 (2018), 7029-7083.
[60] S. G. Robson, E. R. Banta, Ground water atlas of the united states: Segment 2, Arizona, Colorado, New mexico, Utah, Tech. rep., U.S. Geological Survey, 1995.
[61] J. Fauqueur, G. Brostow, R. Cipolla, Assisted video object labeling by joint tracking of regions and keypoints, 2007 IEEE 11th International Conference on Computer Vision, pp. 1-7, 2007.
[62] C. Rother, V. Kolmogorov, A. Blake, ”grabcut”: Interactive foreground extraction using iterated graph cuts, ACM Trans. Graph. 23 (2004), 309-314.
[63] G. Bradski, The OpenCV Library, Dr. Dobb’s Journal of Software Tools, 2000.