

# DOUBLE-WELL NET FOR IMAGE SEGMENTATION\*

HAO LIU<sup>†</sup>, JUN LIU<sup>‡</sup>, RAYMOND CHAN<sup>§</sup>, AND XUE-CHENG TAI<sup>¶</sup>

**Abstract.** In this study, our goal is to integrate classical mathematical models with deep neural networks by introducing two novel deep neural network models for image segmentation known as Double-well Nets. Drawing inspirations from the Potts model, our models leverage neural networks to represent a region force functional. We extend the well-know MBO (Merriman-Bence-Osher) scheme to solve the Potts model. The widely recognized Potts model is approximated using a double-well potential and then solved by an operator-splitting method, which turns out to be an extension of the well-known MBO scheme. Subsequently, we replace the region force functional in the Potts model with a UNet-type network, which is data-driven and is designed to capture multiscale features of images, and also introduce control variables to enhance effectiveness. The resulting algorithm is a neural network activated by a function that minimizes the double-well potential. What sets our proposed Double-well Nets apart from many existing deep learning methods for image segmentation is their strong mathematical foundation. They are derived from the network approximation theory and employ the MBO scheme to approximately solve the Potts model. By incorporating mathematical principles, Double-well Nets bridge the MBO scheme and neural networks, and offer an alternative perspective for designing networks with mathematical backgrounds. Through comprehensive experiments, we demonstrate the performance of Double-well Nets, showcasing their superior accuracy and robustness compared to state-of-the-art neural networks. Overall, our work represents a valuable contribution to the field of image segmentation by combining the strengths of classical variational models and deep neural networks. The Double-well Nets introduce an innovative approach that leverages mathematical foundations to enhance segmentation performance.

**Key words.** Potts model, operator splitting, MBO scheme, double-well potential, deep neural network, image segmentation

**MSC codes.** 68U10, 94A08

**1. Introduction.** Image segmentation is an important problem in image processing, computer vision, and object recognition. How to segment objects accurately from a given image has been an active research topic for a long time [1, 9, 13, 38, 59]. In past decades, many mathematical models have been developed for image segmentation. One line of research focuses on contours and design forces to drive contours to objects' boundaries, such as the active contour model [38] and the geodesic active contour model [9]. Another class of methods, which was first introduced by Mumford and Shah in [59], aims to find piece-wise smooth functions to approximate the given image. In [59], the discontinuity set is the segmentation result, which is restricted to a smooth curve. The Mumford-Shah model inspired a lot of image processing models, among which a well-known one for image segmentation is the Chan-Vese model [13], in which the discontinuity set is restricted to closed curves. In [63], the authors use level set functions to represent the discontinuity set, which allows them to handle

---

\*Submitted to the editors DATE.

**Funding:** The work of Hao Liu was partially supported by National Natural Science Foundation of China 12201530, HKRGC ECS 22302123 and HKBU 179356. The work of Jun Liu was partially supported by NSFC 12371527. The work of Raymond Chan was partially supported by HKRGC GRF grants CityU1101120, CityU11309922, CRF grant C1013-21GF and HKITF MHKJFS Grant MHP/054/22. The work of Xue-Cheng Tai was partially supported by HKRGC-NSFC Grant N-CityU214/19, HKRGC CRF Grant C1013-21GF and NORCE Kompetanseoppbygging program.

<sup>†</sup>Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR. ([haoliu@hkbu.edu.hk](mailto:haoliu@hkbu.edu.hk)).

<sup>‡</sup>School of Mathematical Sciences, Laboratory of Mathematics and Complex Systems, Beijing Normal University, Beijing 100875, P.R. China. ([jliu@bnu.edu.cn](mailto:jliu@bnu.edu.cn)).

<sup>§</sup>Lingnan University, Hong Kong SAR. ([raymond.chan@ln.edu.hk](mailto:raymond.chan@ln.edu.hk)).

<sup>¶</sup>Corresponding author. NORCE Norwegian Research Centre, Nygårdstangen, NO-5838 Bergen, Norway. ([xtai@norceresearch.no](mailto:xtai@norceresearch.no)).

topology changes easily. Variants of the Chan-Vese model with various regularizers are studied in [1, 11, 83].

Another popular model for image segmentation is the Potts model [5, 10, 12, 75, 81], which originates from statistical mechanics [65] and can be taken as the generalization of the two-state Ising model for lattice [64]. The Potts model relates closely to graph cut algorithms. It is in fact a min-cut problem, which is equivalent to the max-flow and convex dual problems, see [75, 80, 88, 89] for some detailed explanations. Another interesting fact about the Potts model is that it can also be taken as a generalization of the Chan-Vese model [73]. Other mathematical models for image segmentation include a smooth and threshold method [7, 43], conformal mapping [90, 91], to name a few.

Most of the models mentioned above require solving some optimization problems. However, some models are complicated which are challenging to solve. Efficient and robust methods to solve these problems are also an active research field in image processing. A large class of numerical methods is based on the alternating direction method of multipliers (ADMM), which has been studied in [28], see also some recent expositions [1, 86, 88]. When the parameters are properly set, ADMM solves optimization problems very efficiently. However, parameters in ADMM need to be carefully tuned in order to get good results. Another class of methods that is not sensitive to parameters is the operator-splitting method, which decomposes a complicated problem into subproblems so that each subproblem either has a closed-form solution or can be solved efficiently. Recently, operator-splitting methods have been successively applied in image processing [20, 21, 49, 50], surface reconstruction [33], numerical PDEs [28, 47], inverse problems [27], obstacle problem [51], computational fluid dynamics [4, 6]. We suggest that readers refer to [29, 30] for a comprehensive discussion of operator-splitting methods. It has been shown in [20, 21] that compared to ADMM, operator-splitting methods give similar (or slightly better) results but are more stable and efficient. In fact, ADMM is a special type of operator-splitting method.

In the last few years, deep neural networks have demonstrated impressive performances in many tasks, including image denoising [42, 44, 84, 92] and image segmentation [14, 66, 94]. Many network architectures are designed for image segmentation, such as UNet [66], UNet++ [94] and DeepLabV3+ [14]. While deep neural networks provide very good results, and in many cases, they are better than traditional image segmentation methods, they are black box algorithms, and their mathematical understanding is unclear. Making connections between deep neural networks and traditional mathematical models, as well as developing mathematically interpretative models, remain open questions.

Recently, several attempts have been made to connect mathematical models and deep neural networks. In [81], neural networks are viewed as the discretization of continuous dynamical systems. PDE and ODE-motivated networks are proposed in [31, 69]. The connections between deep neural networks and control problems are studied in [3, 61, 68]. Theories on the relation between neural ordinary differential equations and the controllability problem are established in [22]. Specifically, they show that for any Lipschitz bounded vector field, neural ODEs can be used to approximate solutions of the continuity equation. Connections between deep neural networks with variational problems are pointed out in [25, 78]. The authors of [25] show that combining adversarial training and Lipschitz regularization improves adversarial robustness and is equivalent to total variation regularization. In [78], the authors show that the deep layer limit of residual neural network coincides with a parameter estima-

tion problem for a nonlinear ordinary differential equation. In [60], the authors replace the dot-product kernels in transformers with Fourier integral kernels, achieving higher accuracy in language modeling and image classification. MgNet, a network inspired by the multigrid method, is proposed in [32]. The connections between neural networks and operator-splitting methods are pointed out in [40, 48]. Regularizers in popular mathematical models are used as priors in [37, 41, 53] to design networks that enable the segmentation results to have special priors. Recently, based on the Potts model, multigrid method, and operator-splitting method, the authors proposed PottsMGNet [76], which gives a clear mathematical explanation for existing encoder-decoder type of neural networks.

In this paper, we integrate the Potts model with deep neural networks and propose two networks inspired by operator-splitting methods for image segmentation. Consider the two-phase Potts model. It contains a region force term (see the first term in (2.3)) that is manually designed for good performance. Whether this term is optimal so that the resulting model gives the best segmentation results is an open question. In this paper, we consider a data-driven approach to determine it. We propose representing this term by networks, whose parameters will be learned from data. Starting from the Potts model, we first formulate an initial value problem (in the sense of gradient flow) to minimize the functional. Then, the initial value problem is time-discretized by operator-splitting methods. In the resulting scheme, each time stepping consists of two substeps. The first substep computes the segmentation linearly or nonlinearly, depending on the scheme. The second substep is a nonlinear step that approximately minimizes a double-well potential with a proximal term. Such a splitting strategy is an extension of the well-known MBO scheme of [55, 56] as explained in [23, 74]. Our proposed approach is equivalent to a neural network consisting of several blocks: each time step corresponds to a block that is activated by approximately minimizing a double-well potential. The resulting scheme has an architecture that is similar to a convolutional neural network. We call our scheme Double-well Net (DN) and propose two variants of it: DN-I and DN-II. In general, a Double-well Net is an approximate solver for minimizing the Potts model using an extension of the MBO scheme, where the region force term is represented by networks. It is also a bridge between classical numerical algorithms and neural networks as it unveils the connections between the well-known MBO scheme and networks.

Compared to existing networks for image segmentation, such as UNet, the proposed Double-well Nets have several novelties: (1) DN-I uses a subnetwork as a bias term, while in existing networks, the bias term is a scalar. (2) DN-II uses the input image in each block instead of only at the input of the network, which is commonly done in existing networks. (3) Both DNs use an activation function derived from the double-well potential. (4) Both DNs have a mathematical background: they are operator-splitting algorithms that approximately solve the Potts model. PottsMGNet [76] is another network for image segmentation which is based on the Potts model, operator splitting methods and has mathematical explanations. PottsMGNet uses operator-splitting methods together with multigrid methods to give an explanation of encoder-decoder based neural networks, and uses sigmoid function as activation. In this paper, the DNs are derived using the Potts model, the MBO scheme and network approximation theory, which have a mathematical explanation from another perspective. Furthermore, DNs use a double-well potential related operator as activation function. Our numerical results show that Double-well Nets give better results than the state-of-the-art segmentation networks on several datasets.

This paper is structured as follows: We introduce the Potts model and derive the

corresponding initial value problem in Section 2. Operator-splitting methods which are extensions of the well-known MBO scheme for solving these initial value problems and numerical discretization are discussed in Section 3. We propose Double-well Nets in Section 4, and demonstrate their effectiveness by comprehensive numerical experiments in Section 5. This paper is concluded in Section 6.

**2. Potts model.** Let  $\Omega \subset \mathbb{R}^2$  be a rectangular image domain. The continuous two-phase Potts model is in the form of [10, 65, 75, 80, 88]

$$(2.1) \quad \left\{ \begin{array}{l} \min_{\Sigma_0, \Sigma_1} \left\{ \frac{1}{2} \sum_{k=0}^1 \text{Per}(\Sigma_k) + \sum_{k=0}^1 \int_{\Sigma_k} h_k(\mathbf{x}) d\mathbf{x} \right\}, \\ \Sigma_0, \Sigma_1 \subset \Omega, \quad \Sigma_0 \cup \Sigma_1 = \Omega, \quad \Sigma_0 \cap \Sigma_1 = \emptyset, \end{array} \right.$$

where  $\Sigma_k$  is regular enough so that its perimeter exists, denoted by  $\text{Per}(\Sigma_k)$ , and  $f_k(\mathbf{x})$ 's are non-negative weight functions. A popular choice of  $h_k$  (as in [13]) is

$$(2.2) \quad h_k(\mathbf{x}) = (f(\mathbf{x}) - r_k)^2 / \alpha,$$

in which  $\alpha$  is a scaling parameter, and  $r_k$  is the mean density of  $f(\mathbf{x})$  on  $\Sigma_k^*$ . Here, we use  $\Sigma_k^*$  to denote the ‘optimal’ segmentation region, the ground truth segmentation of  $f$  and is independent to any model. In practice, one has no information of  $\Sigma_k^*$  and has to estimate  $r_k$ . How to estimate it is not the focus of this paper and is omitted. Here we want to emphasis that such a  $r_k$  exists and only depends on  $f$ . As one can take  $r_k$  and  $\Sigma_k^*$  as functions depending on  $f$ , in this case,  $h_k$ 's are functions of the input image  $f$  only. We do not require  $\Sigma_k^*$  to be a minimizer of (2.1). But we hope there exists some  $h_k$  so that the minimizer is close to  $\Sigma_k^*$ . Later, we will represent  $h_k$  by a neural network.

The Potts model for two-phase problems can be solved using binary representations as the following min-cut problem:

$$(2.3) \quad \min_{v \in \{0,1\}} \int_{\Omega} F(f)v d\mathbf{x} + \lambda \int_{\Omega} |\nabla v| d\mathbf{x},$$

where  $F(f) = f_1 - f_0$  is called the region force depending on the input image  $f$ , and  $\Omega$  is the domain on which the image is defined. Model (2.3) requires the function  $v$  to be binary. The above nonconvex min-cut problem is equivalent to the following convex dual problem of a max-flow problem as explained in [80, Section 2.1] and [75, the section of (84) and (85)]:

$$(2.4) \quad \min_{v \in [0,1]} \int_{\Omega} F(f)v d\mathbf{x} + \lambda \int_{\Omega} |\nabla v| d\mathbf{x}.$$

This is often called the convex relaxation of (2.3). This model also recovers the well-known CEN model of Chan-Esedoglu-Nikolova [12]. If we use the Ginzburg-Landau functional  $\mathcal{L}_\varepsilon$  to approximate the second term in (2.3), we then need to solve:

$$(2.5) \quad \min_v \int_{\Omega} F(f)v d\mathbf{x} + \lambda \mathcal{L}_\varepsilon(v) d\mathbf{x},$$

with

$$\mathcal{L}_\varepsilon(v) = \int_{\Omega} \left[ \frac{\varepsilon}{2} |\nabla v|^2 + \frac{1}{\varepsilon} v^2 (1-v)^2 \right] d\mathbf{x},$$

where the second term in  $\mathcal{L}_\varepsilon$  is the double-well potential. It is shown that the minimizer of (2.5) converges to that of (2.3) in the sense of Gamma-convergence as  $\varepsilon \rightarrow 0$  [57, 58]. The Gamma-convergence of the graph-based Ginzburg-Landau functional was proved in [79], and its applications in image processing, and data segmentation on graphs have been studied in [26, 55]. Global minimization for graph data using min-cut/max-flow approaches, c.f. [1, 88, 89], has also been studied in [54].

Denote the minimizer of (2.5) by  $u$ . It satisfies the optimality condition

$$(2.6) \quad F(f) - \lambda\varepsilon\nabla^2 u + \frac{2\lambda}{\varepsilon}(2u^3 - 3u^2 + u) = 0,$$

where  $\nabla^2$  is the Laplacian operator. One way to solve (2.6) for  $u$  is to associate it with the following initial value problem (in the sense of gradient flow):

$$(2.7) \quad \begin{cases} \frac{\partial u}{\partial t} = -F(f) + \lambda\varepsilon\nabla^2 u - \frac{2\lambda}{\varepsilon}(2u^3 - 3u^2 + u) & \text{in } \Omega \times (0, T], \\ \frac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega, \\ u(0) = u_0 & \text{in } \Omega, \end{cases}$$

for some initial condition  $u_0$  and fixed time  $T$ . Then solving (2.6) is equivalent to finding the steady state solution of (2.7).

**3. The proposed models.** In classical mathematical models for image segmentation, the operator  $F$  is carefully designed, as (2.2) in the Chan-Vese model. These models have demonstrated great performances in segmenting general images. For a specific type of images, certain choices of  $F$  may give improved performances. In this paper, we consider a data-driven method to learn  $F$ .

Suppose we are given a training set of images  $\{f_i\}_{i=1}^I$  with their foreground-background segmentation masks  $\{g_i\}_{i=1}^I$  so that there exists some  $F$  such that for each  $f_i$ , the minimizer of (2.5) is close to  $g_i$ . We will learn a data-driven operator  $F$  so that for any given image  $f$  with similar properties as the training set, the steady state of (2.7) is close to its segmentation  $g$ . Note that (2.7) is an initial value problem of  $u$ . In practice, it might be difficult to solve (2.7) until the steady state. Instead, a more practical way is to solve it until a finite time  $t = T$ , and use the solution (denoted by  $u(\mathbf{x}, T)$ ) as the segmentation. However,  $u(\mathbf{x}, T)$  may be far away from the ground truth segmentation. To better control the evolutionary behavior of  $u$  and borrow some of the ideas from [76], we introduce control variables  $W(\mathbf{x}, t), b(t)$  into (2.7) and consider:

$$(3.1) \quad \begin{cases} \frac{\partial u}{\partial t} = -F(f) + \lambda\varepsilon\nabla^2 u - \frac{2\lambda}{\varepsilon}(2u^3 - 3u^2 + u) + W(\mathbf{x}, t) * u + b(t) & \text{in } \Omega \times (0, T], \\ \frac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega, \\ u(0) = u_0 & \text{in } \Omega, \end{cases}$$

where  $*$  stands for the convolution operator and  $W(\mathbf{x}, t)$  is a convolution kernel depending on  $\mathbf{x}$  and  $t$ , and  $b(t)$  is some function of  $t$ . The control variable will adjust the evolution of  $u$  and steer  $u(\mathbf{x}, T)$  to be close to the groundtruth segmentation. The control variables will be learned from the given dataset. We remark that here  $W(\mathbf{x}, t)$  is a convolution kernel instead of the double-well potential in many works related to the Ginzburg-Landau functional.

We are going to present two models to solve (3.1). The first model is an operator-splitting scheme for (3.1), see Section 3.1. The second model generalize the first one

in which we allow more complicated interactions among  $F(f)$ ,  $W(\mathbf{x}, t)$ ,  $b(t)$  and  $u$  and represent these interactions by one operator, see Section 3.2. In our methods, we represent  $F$  in the first model and the new operator in the second model as a neural network. Denote  $\mathcal{W}$  as the collection of all parameters to be determined from the data, i.e., the parameters in  $F$  and the control variables  $W(\mathbf{x}, t)$ ,  $b(t)$ . The initial condition  $u_0 = H(f)$  is taken as a function of the input image  $f$  with a properly chosen  $H$ , c.f. (4.5). Then the solution of (3.1) at time  $t = T$  only depends on  $f$  and  $\mathcal{W}$ . For each  $f, \mathcal{W}$ , denote the solution for (3.1) at time  $T$  by  $R(\mathcal{W}; f)$ , which is a function map from  $f$  to  $R(\mathcal{W}; f)$  (the solution at  $T$ ). We will determine  $\mathcal{W}$  by solving

$$(3.2) \quad \min_{\mathcal{W}} \frac{1}{K} \sum_{k=1}^K \ell(R(\mathcal{W}; f_k), g_k),$$

where  $\ell(\cdot, \cdot)$  is a loss function measuring the differences between its arguments. Popular choices of the loss functional include the hinge loss, logistic loss, and  $L^2$  norm, see [67] for a discussion of different loss functions.

In this section, we focus on the solvers for the control problem (3.1). Specifically, we consider model (3.1) and a variant model and propose operator-splitting methods to solve them. We will show that the resulting scheme has a similar architecture as deep neural networks.

*Remark 3.1.* There is no problem for our proposed methods if we replace  $W(\mathbf{x}, t) * u + b(t)$  with some other general linear operator on  $u$  in (3.1). The reason for choosing the convolution operator and the bias correction here is due to their success in applications related to Deep Convolutional Neural Networks.

**3.1. Model I.** The first model we study is (3.1). Note that it is well suited to be numerically solved by operator-splitting methods. For a comprehensive introduction to operator-splitting methods, we refer readers to [29, 30]. In this paper, we adopt the Lie scheme. The terms on the right-hand side of (3.1) can be classified into linear terms and nonlinear terms of  $u$ , based on how we split them into two substeps. Let  $\tau$  be the time step. For  $n \geq 0$ , we denote  $t^n = n\tau$  and our numerical solution at  $t^n$  by  $u^n$ . Set  $u_0 = H(f)$  with a properly chosen  $H$ , c.f. (4.5). We compute  $u^{n+1}$  from  $u^n$  via two substeps:  $u^n \rightarrow u^{n+1/2} \rightarrow u^{n+1}$ . The first substep focuses on linear terms; the second substep focuses on nonlinear terms. The details are as follows:

**Substep 1:** Solve

$$(3.3) \quad \begin{cases} \frac{\partial u}{\partial t} = -F(f) + \lambda \varepsilon \nabla^2 u + W(\mathbf{x}, t) * u + b(t) & \text{in } \Omega \times (t^n, t^{n+1}], \\ \frac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega, \\ u(t^n) = u^n & \text{in } \Omega, \end{cases}$$

and set  $u^{n+1/2} = u(t^{n+1})$ .

**Substep 2:** Solve

$$(3.4) \quad \begin{cases} \frac{\partial u}{\partial t} = -\frac{2\lambda}{\varepsilon}(2u^3 - 3u^2 + u) & \text{in } \Omega \times (t^n, t^{n+1}], \\ u(t^n) = u^{n+1/2} & \text{in } \Omega, \end{cases}$$

and set  $u^{n+1} = u(t^{n+1})$ .

Scheme (3.3)–(3.4) is semi-constructive since we still need to solve the two initial value problems. In this paper, we use a one-step forward Euler scheme to time

discretize (3.3) and a one-step backward Euler scheme to time discretize (3.4):

$$(3.5) \quad \begin{cases} \frac{u^{n+1/2} - u^n}{\tau} = -F(f) + \lambda\varepsilon\nabla^2 u^n + W^n * u^n + b^n \text{ in } \Omega, \\ \frac{\partial u^{n+1/2}}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \end{cases}$$

$$(3.6) \quad \frac{u^{n+1} - u^{n+1/2}}{\tau} = -\frac{2\lambda}{\varepsilon}(2(u^{n+1})^3 - 3(u^{n+1})^2 + u^{n+1}) \text{ in } \Omega,$$

where the notations  $W^n = W(\mathbf{x}, t^n)$ ,  $b^n = b(t^n)$  are used.

*Remark 3.2.* In (3.5),  $\lambda\varepsilon\nabla^2$  is a penalty term on the smoothness of the segmentation's boundary, and  $W^n$  is a control variable that is learnable. The term  $\lambda\varepsilon\nabla^2$  can be absorbed by  $W^n$  since both of them are linear in  $u^n$ . However, in Model I, we write  $\lambda\varepsilon\nabla^2 u$  explicitly to explicitly drive the segmentation boundary to be smooth.

**3.2. Model II.** In model (3.1), the functional  $F(f)$  is fixed through all iterations. In our implementation, we will use a UNet class (see Section 4 for details) to represent  $F$ , and learn  $F$  and the control variables  $W$  and  $b$  from data. In the second model, we learn a more general functional  $G(u, f, \mathbf{x}, t)$  as shown below in (3.7). In our implementation, we will use a UNet class to represent  $G$ , and learn it from data, without any assumption on the specific form of  $G$ . Our second model is as follows

$$(3.7) \quad \begin{cases} \frac{\partial u}{\partial t} = \lambda\varepsilon\nabla^2 u - \frac{2\lambda}{\varepsilon}(2u^3 - 3u^2 + u) + G(u, f, \mathbf{x}, t) \text{ in } \Omega \times (t^n, t^{n+1}], \\ \frac{\partial u}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \\ u(0) = u_0 \text{ in } \Omega. \end{cases}$$

Note that  $-F(f) + W(\mathbf{x}, t) * u + b(t)$  in (3.1) is a special case of  $G(u, f, \mathbf{x}, t)$  in (3.7). A linear relationship among  $F(f)$ ,  $W(\mathbf{x}, t)$ ,  $b(t)$  and  $u$  is enforced in (3.1). By replacing  $-F(f, t) + W(\mathbf{x}, t) * u + b(t)$  by the functional  $G(u, f, \mathbf{x}, t)$ , we allow more complicated interactions among them. Thus (3.7) is a more general model.

Similar to (3.1), we choose  $u_0 = H(f)$  as a function of the input image  $f$  with a properly chosen  $H$ , c.f. (4.5), and use a Lie scheme to time discretize (3.7). The splitting strategy is the same as (3.3)–(3.4) except the first substep becomes

$$(3.8) \quad \begin{cases} \frac{\partial u}{\partial t} = \lambda\varepsilon\nabla^2 u + G(u, f, \mathbf{x}, t) \text{ in } \Omega \times (t^n, t^{n+1}], \\ \frac{\partial u}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \\ u(t^n) = u^n \text{ in } \Omega. \end{cases}$$

We time discretize Substep 1 by a forward Euler method and Substep 2 by a backward Euler method. The time-discretized scheme reads as

$$(3.9) \quad \begin{cases} \frac{u^{n+1/2} - u^n}{\tau} = \lambda\varepsilon\nabla^2 u^n + G^n(u^n, f) \text{ in } \Omega, \\ \frac{\partial u^{n+1/2}}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \end{cases}$$

$$(3.10) \quad \frac{u^{n+1} - u^{n+1/2}}{\tau} = -\frac{2\lambda}{\varepsilon}(2(u^{n+1})^3 - 3(u^{n+1})^2 + u^{n+1}) \text{ in } \Omega,$$

where  $G^n(u, f) = G(u, f, \mathbf{x}, t^n)$ .

The main difference between Model I and Model II is how we treat  $F(f)$  and the control variables. Model I is directly derived from (3.1). It is linear in the control variables, and the so called region force  $F(f)$  is independent of  $u$  and  $t$ , i.e.,  $F(f)$  stays the same over time. Model II generalizes Model I, in which we allow more complicated

interactions among  $u, F(f)$  and the control variables. The interaction complexity is determined by the functional used to represent  $G(u, f, \mathbf{x}, t)$ . In later sections, we will use a network to represent  $F(f)$  for Model I and  $G(u, f, \mathbf{x}, t)$  for Model II.

*Remark 3.3.* There are many ways to combine and decompose terms in the right-hand side of (3.1), among which Model I and II are just two special ones that provide good results. As we will show in Section 4, both Model I and II have similar architectures as some neural networks. Based on Model I, we further explore Model II because Model II can provide similar results with fewer parameters.

*Remark 3.4.* The operator-splitting schemes for Model I and II are extensions of the Allen-Cahn-type MBO scheme [56], as explained in [23, 74]. The first substep focuses on linear operators and the second substep deals with nonlinear projection operators. In this work, only two-phase image segmentation is considered. Following the approaches proposed in [74], there is no problem to extend the method in this work to multiphase image segmentation problems using deep neural networks similar to Model I and II.

*Remark 3.5.* In this work, we present our method for two-phase image segmentation. For graph data, each vertex of the graph could be a high dimension vector. Using the ideas presented in this work, we can combine some existing graph neural networks, such as [19, 77], with the graph Ginzburg-Landau model of [26, 55] and the graph min-cut/max-flow approach of [54, 80, 87] to get some graph neural network similar to Model I and II for high-dimensional data analysis.

**3.3. Connections to neural networks.** In Model I and II, for each time step, our scheme has two substeps: the first substep is linear operation in  $u^n$ , and the second substep is a nonlinear operation. This structure is the same as the building block (layers) of neural networks (NN). An NN usually consists of several layers. Each layer conducts the computation

$$(3.11) \quad \sigma(Ax + b),$$

where  $A$  is a weight matrix,  $b$  is a bias term (a vector), and  $\sigma$  is a nonlinear activation function applied pointwisely. In (3.11), when the input to  $\sigma$  is a linear operation, it is similar to our first substep; and when the operation of  $\sigma$  is nonlinear, it is similar to our second substep.

**3.4. Accommodate periodic boundary conditions.** In image processing, it is common to use the periodic boundary conditions. Let  $\Omega$  be a rectangular domain  $[0, L_1] \times [0, L_2]$ . We denote the two spatial directions by  $x_1, x_2$ . To accommodate the periodic boundary conditions, we only need to replace (3.5) by

$$(3.12) \quad \begin{cases} \frac{u^{n+1/2} - u^n}{\tau} = -F(f) + \lambda\varepsilon\nabla^2 u^n + W^n * u^n + b^n & \text{in } \Omega, \\ u^{n+1/2}(0, x_2) = u^{n+1/2}(L_1, x_2), & 0 \leq x_2 \leq L_2, \\ u^{n+1/2}(x_1, 0) = u^{n+1/2}(x_1, L_2), & 0 \leq x_1 \leq L_1, \end{cases}$$

and replace (3.9) by

$$(3.13) \quad \begin{cases} \frac{u^{n+1/2} - u^n}{\tau} = \lambda\varepsilon\nabla^2 u^n + G^n(u^n, f) & \text{in } \Omega, \\ u^{n+1/2}(0, x_2) = u^{n+1/2}(L_1, x_2), & 0 \leq x_2 \leq L_2, \\ u^{n+1/2}(x_1, 0) = u^{n+1/2}(x_1, L_2), & 0 \leq x_1 \leq L_1. \end{cases}$$

In the rest of this paper, we always assume that  $u$  satisfies the periodic boundary condition.

**3.5. Spatial discretization.** In our discretization, we use spatial steps  $\Delta x_1 = \Delta x_2 = h$  for some  $h > 0$ . We denote the spatially-discretized  $u^n$  and  $W^n$  by  $\widehat{u}^n$  and  $\widehat{W}^n$ , respectively.

From (3.12), an updating formula of  $u^{n+1/2}$  is given as

$$u^{n+1/2} = u^n - \tau F(f) + \tau \lambda \varepsilon (\nabla^2 u^n) + \tau W^n * u^n + \tau b^n.$$

Note that  $\nabla^2 u^n$  is the Laplacian of  $u^n$ . We approximate it by central difference, which can be realized by convolution  $\widehat{W}_\Delta * \widehat{u}^n$  with

$$\widehat{W}_\Delta = \frac{1}{h^2} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Then the updating formula becomes

$$(3.14) \quad \widehat{u}^{n+1/2} = \widehat{u}^n - \tau F(f) + \tau \lambda \varepsilon (\widehat{W}_\Delta * \widehat{u}^n) + \tau \widehat{W}^n * \widehat{u}^n + \tau b^n.$$

Similarly, for  $u^{n+1/2}$  in model (3.13), the updating formula is

$$(3.15) \quad \widehat{u}^{n+1/2} = \widehat{u}^n + \tau \lambda \varepsilon (\widehat{W}_\Delta * \widehat{u}^n) + \tau G^n(\widehat{u}^n, f).$$

For updating  $\widehat{u}^{n+1}$ , note that in continuous case  $u^{n+1}$  is the minimizer of

$$(3.16) \quad \int_{\Omega} \frac{1}{2} (u - u^{n+1/2})^2 + \frac{\tau \lambda}{\varepsilon} u^2 (1 - u)^2 d\mathbf{x}$$

and (3.6) is the Euler-Lagrange equation of (3.16). Denote  $\alpha = 2\tau\lambda/\varepsilon$ . Then  $u^{n+1}$  satisfies

$$(3.17) \quad (1 + \alpha)u^{n+1} = u^{n+1/2} - \alpha(2(u^{n+1})^3 - 3(u^{n+1})^2).$$

After discretizing the equation above, we solve for  $\widehat{u}^{n+1}$  pointwisely using a fixed-point iteration: For  $i = 1, \dots, M_1, j = 1, \dots, M_2$ , set  $\widehat{v}^0 = \widehat{u}^{n+1/2}$ , and we update  $\widehat{v}^m \rightarrow \widehat{v}^{m+1}$  as

$$(3.18) \quad \widehat{v}^{m+1} = \frac{1}{1 + \alpha} (\widehat{u}^{n+1/2} - \alpha(2(\widehat{v}^m)^3 - 3(\widehat{v}^m)^2)).$$

Proving the convergence of (3.18) requires a dedicated analysis of the problem, for which we leave as our future work. Empirically, when the time step  $\tau$  (as well as  $\alpha$ ) is sufficiently small, we observe that iteration (3.18) converges. A discussion on how to numerically improve its convergence behavior is presented in Section 4.2. Let  $\widehat{v}^*$  denote the point  $\widehat{v}^m$  converges to. We then set  $\widehat{u}^{n+1} = \widehat{v}^*$ .

**4. Double-well net.** In our framework, we solve (3.2) to learn  $\mathcal{W}$ , which contains the parameters in  $F(f)$  and the control variables  $W(\mathbf{x}, t^n), b(t^n)$  for Model I, or the parameters in  $G(u, f, \mathbf{x}, t^n)$  for Model II. To learn  $F(f)$  and  $G(u, f, \mathbf{x}, t^n)$ , we need to assume a general form for them and then learn their weights by solving (3.2). In this paper, we will use neural networks to represent  $F(f)$  and  $G(u, f, \mathbf{x}, t^n)$ .

The representation theory for neural networks has been an active topic in past decades, see [2, 34, 45, 46, 52, 62, 72, 85, 93]. These works show that when the network size (depending on the width and depth) is sufficiently large, neural networks can approximate functions, functionals, or operators with certain regularity to arbitrary

accuracy. If the target function has low-dimensional structures, the network size can be reduced to maintain the same accuracy, i.e., the curse of dimensionality is mitigated, see [15, 16, 18, 45, 46, 70] for details.

For image segmentation, many neural networks have demonstrated great performances. In this paper, we adopt the well-known UNet [66] type architecture which has provided remarkable results in various segmentation tasks [8, 36]. In this section, we first introduce a class of UNet architecture. This class is then used to represent  $F(f)$  and  $G(u, f, \mathbf{x}, t^n)$  in the proposed models.

**4.1. UNet class.** UNet has an encoder-decoder architecture and is proposed in [66, 76] for medical image segmentation. UNet’s architecture consists of three parts: an encoding part, a decoding part, and a bottleneck part.

- The encoding part consists of several convolutional layers and pooling layers. Given an image  $f$ , this part gradually reduces the resolution of the image while increasing the number of channels. Specifically, for every two layers, the number of channels is doubled and a pooling layer is used to reduce the resolution. Here the number of channels corresponds to the third dimension of the output. Consider a convolution layer with output  $\mathbf{z} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ . The number of channels is  $d_3$ . Each  $\mathbf{z}(:, :, i)$  for  $i = 1, \dots, d_3$  is the result of computing the convolution of the input with a convolution kernel. Thus the number of channels also determines the number of convolution kernels a convolution layer contains.
- Following the encoding part, the bottleneck part downsamples the output of the encoding part again, doubles the number of channels, and conducts two convolutional layers. The result is then upsampled and passed to the decoding part.
- The decoding part consists of convolutional layers and upsampling layers. It upsamples the output of the bottleneck part from the coarsest resolution to the original image resolution and gradually reduces the number of channels. Specifically, for every two layers, the number of channels is halved and an upsampling layer is used to increase the resolution.
- Besides the three parts mentioned above, there are skip connections between the encoding and decoding parts. For each resolution, there are two convolutional layers in the encoding part and two in the decoding part. Skip connections pass the output of the layers in the encoding part directly to the corresponding layers of the decoding part.

In UNet, there are five levels of resolutions: four levels for the encoding and decoding part and one level for the bottleneck part. For each resolution, there are two convolutional layers in the encoding and decoding part (or the bottleneck part). Each downsampling operator reduces the resolution of each dimension by half, and each upsampling operator doubles the resolution of each dimension. Every time the intermediate result is downsampled, the number of channels doubles. Every time the intermediate result is upsampled, the number of channels is reduced by half.

Based on the architecture of UNet, we define a more general class in which we have the flexibility to control the number of resolution levels and number of channels. For each resolution level for the encoding and decoding parts, we manually set the number of channels. For the bottleneck part, we double the number of channels of the coarsest resolution level of the encoding and decoding parts. Considering the resolution levels for the encoding and decoding parts in the order from finest to the coarsest, we denote

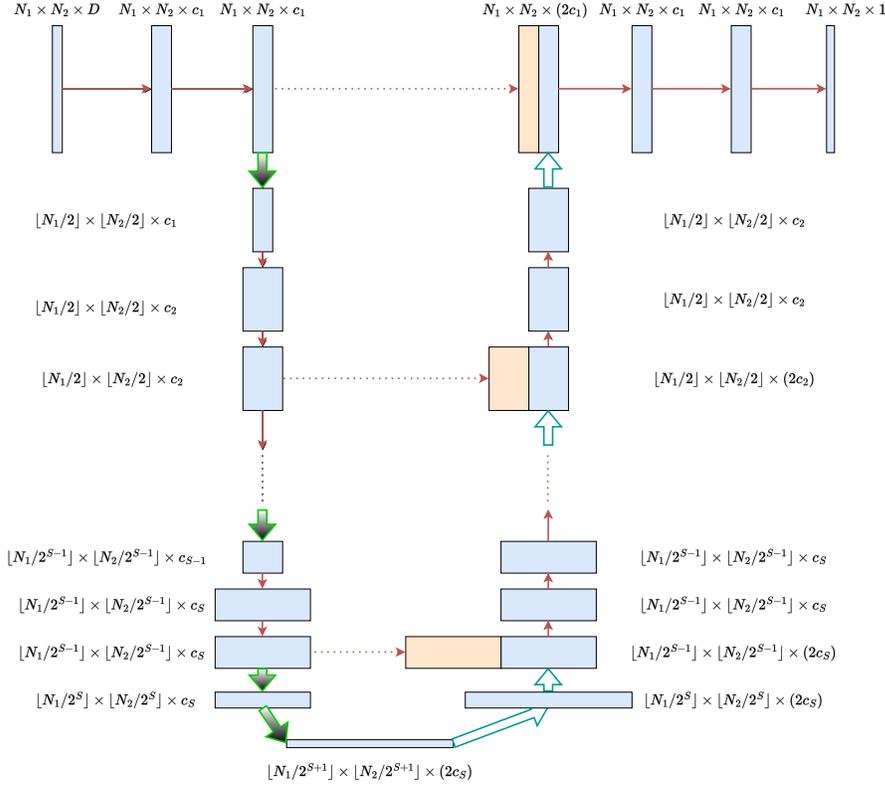


Fig. 1: Illustration of UNet type network with input of size  $N_1 \times N_2 \times D$ . The left branch is the encoding part, the right branch is the decoding part, and the bottom rectangle denotes the bottleneck.  $S$  denotes the number of resolution scales in the encoding part and decoding part.  $c_k$  denotes the number of channels at resolution scale  $k$ . Wide arrows with gradient shadow represent downsampling operations. Wide arrows without gradient shadow represent upsampling operations. Horizontal dashed arrows represent skip connections. The orange rectangles denote the outputs of the encoding part that are passed to the decoding part via the skip connections. The length and width of the rectangle represent the output resolution and number of channels, respectively.

their corresponding number of channels as a vector  $\mathbf{c} = [c_1, \dots, c_S]$  for some positive integers  $\{c_s\}_{s=1}^S$ , where  $S$  denotes the number of resolution levels. Every network in this class can be fully characterized by the channel vector  $\mathbf{c}$ : given a  $\mathbf{c} \in \mathbb{R}^S$ , the corresponding network has  $S + 1$  resolution levels,  $c_s$  channels at resolution level  $s$  for  $1 \leq s \leq S$ , and  $2c_S$  channels at resolution level  $S + 1$ . For an input with dimension  $N_1 \times N_2 \times D$ , the structure of such a class is illustrated in Figure 1. The architecture is designed to capture multiscale features of images: each resolution level corresponds to features of one scale. For the original UNet, it has  $\mathbf{c} = [64, 128, 256, 512]$ . In our model, we will choose  $F(f)$  and  $G(u, f, \mathbf{x}, t^n)$ 's in the UNet class with different  $\mathbf{c}$ 's.

**4.2. Double-well net for model (3.1).** Assume  $f$  is an image of size  $N_1 \times N_2 \times D$ . We approximate  $F(f)$  by a UNet class specified by channel vector  $\mathbf{c}$  (to be specified in Section 5).

**A double-well block I.** We next define a double-well block I (DB-I), which represents a step that updates  $\hat{u}^n$  to  $\hat{u}^{n+1}$  using (3.14) and (3.6). We first compute  $\hat{u}^{n+1/2}$  according to (3.14) as

$$(4.1) \quad \hat{u}^{n+1/2} = \hat{u}^n - \tau F(f) + \tau \lambda \varepsilon (\widehat{W}_\Delta * \hat{u}^n) + \tau (\widehat{W}^n * \hat{u}^n + b^n),$$

where  $\widehat{W}^n$  and  $b^n$  are the convolution kernel and bias in the  $n$ -th residual block, respectively. To compute  $\hat{u}^{n+1}$ , one can solve the fixed-point updating (3.18) with fixed number of iterations.

For efficiency, deep neural networks are trained using stochastic gradient descent with back-propagation. Since all we want is an approximate minimizer of (2.5), intermediate solutions ( $\hat{u}^n$ 's) are not so important. If the scheme converges, it is not necessary to solve the problem (3.16) exactly at every step. For practical consideration, to make the training more efficient, we solve for  $\hat{u}^{n+1}$  by using only few steps of the fixed-point iteration (3.18). In our experiments, three iterations of the fixed-point iteration already make the segmentation result good.

For robustness, note that the second fractional step (3.6) solves (3.16). Pointwisely, the solver of (3.16) drives  $\hat{u}^{n+1/2}$  towards 1 for  $\hat{u}^{n+1/2} > 0.5$ , and drives  $\hat{u}^{n+1/2}$  towards 0 for  $\hat{u}^{n+1/2} < 0.5$ . Our numerical experiments show that the fixed-point iteration (3.18) is robust (convergent) for large time steps when  $0 \leq \hat{u}^{n+1/2} \leq 1$ . However, if  $\hat{u}^{n+1/2}$  is larger than 1 or smaller than 0, the fixed-point iteration may diverge. To make the algorithm robust (convergent for large time step), especially when  $\hat{u}^{n+1/2} > 1$  or  $\hat{u}^{n+1/2} < 0$ , one has to use a very small  $\alpha$  (recall that  $\alpha = 2\tau\lambda/\varepsilon$  is defined in Section 3.5), implying a small time step  $\tau$ . With few fixed point iterations and a small  $\tau$ , the updated  $\hat{u}^{n+1}$  and  $\hat{u}^{n+1/2}$  are very close to each other, making the effect of this substep negligible.

To use a relatively large  $\tau$  while keeping the algorithm stable, we need to restrict  $\hat{u}^{n+1/2}$  in  $[0, 1]$ . In this paper, we consider two methods to achieve that. In the first method, we use a projection operator to project  $\hat{u}^{n+1/2}$  to  $[0, 1]$  before it is passed to the fixed point iteration. For any value  $a$ , we project it to  $[0, 1]$  by solving

$$\min_{z \in [0,1]} (z - a)^2.$$

The explicit solution is given by the following projection operator:

$$\text{Proj}_{[0,1]}(a) = \begin{cases} 0 & \text{if } a \leq 0, \\ a & \text{if } 0 < a < 1, \\ 1 & \text{if } a \geq 1. \end{cases}$$

Then  $\text{Proj}_{[0,1]}(\hat{u}^{n+1/2})$  is applied pointwisely and projects (truncates)  $\hat{u}^{n+1/2}$  at each pixel to  $[0, 1]$ . Denote the fixed-point iteration (3.18) with  $\gamma$  iterations by  $Q_\gamma$ . We update  $\hat{u}^{n+1}$  as

$$(4.2) \quad \hat{u}^{n+1} = Q_\gamma \circ \text{Proj}_{[0,1]}(\hat{u}^{n+1/2}).$$

The projection operator outputs a real number in  $[0, 1]$  and pointwisely preserves the relation between  $|\hat{u}^{n+1/2} - 0|$  and  $|\hat{u}^{n+1/2} - 1|$ : if  $|\hat{u}^{n+1/2} - 0| < |\hat{u}^{n+1/2} - 1|$ , then

$|\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) - 0| < |\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) - 1|$ . In other words, if  $\widehat{u}^{n+1/2}$  is closer to 0 than 1, then  $\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2})$  is closer to 0 than 1. Applying  $\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2})$  before the fixed-point iteration has two benefits: (i) Since the output of  $\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2})$  is in  $[0, 1]$ , we can use a larger time step  $\tau$  while having a convergent fixed-point iteration. (ii) The relation among  $\widehat{u}^{n+1/2}$ , 0 and 1 is preserved. When  $\tau$  is sufficiently small, we have  $Q_\infty(\widehat{u}^{n+1/2}) = Q_\infty \circ \text{Proj}_{[0,1]}(\widehat{u}^{n+1/2})$  pointwisely: When  $\widehat{u}^{n+1/2} < 0.5$ , we have  $Q_\infty(\widehat{u}^{n+1/2}) = 0$  and  $\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) < 0.5$  since the relation among  $\widehat{u}^{n+1/2}$ , 0 and 1 is preserved by the projection. Because  $\text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) < 0.5$ , we have  $Q_\infty \circ \text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) = 0 = Q_\infty(\widehat{u}^{n+1/2})$ . The same argument can be used to show  $Q_\infty \circ \text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) = 1 = Q_\infty(\widehat{u}^{n+1/2})$  when  $\widehat{u}^{n+1/2} > 0.5$  and  $Q_\infty \circ \text{Proj}_{[0,1]}(\widehat{u}^{n+1/2}) = 0.5 = Q_\infty(\widehat{u}^{n+1/2})$  when  $\widehat{u}^{n+1/2} = 0.5$ . Note that the projection operator can be realized by a two-layer ReLU network:

$$\text{Proj}_{[0,1]}(a) = \max\{\min\{a, 1\}, 0\} = \text{ReLU}(1 - \text{ReLU}(1 - a)),$$

which does not introduce extra complexity to the network.

The second method considered in this paper is to utilize the sigmoid function defined by:

$$\text{Sig}(a) = \frac{1}{1 + \exp(-a)}.$$

Compared to  $\text{Proj}$  which is the composition of two ReLU functions,  $\text{Sig}$  is a more common activation function in the deep learning community. We first compute  $\text{Sig}(\widehat{u}^{n+1/2} - 0.5)$ . Such an operation outputs a real number in  $(0, 1)$  and pointwisely preserves the relation between  $|\widehat{u}^{n+1/2} - 0|$  and  $|\widehat{u}^{n+1/2} - 1|$ : if  $|\widehat{u}^{n+1/2} - 0| < |\widehat{u}^{n+1/2} - 1|$ , then  $|\text{Sig}(\widehat{u}^{n+1/2} - 0.5) - 0| < |\text{Sig}(\widehat{u}^{n+1/2} - 0.5) - 1|$ . In other words, if  $\widehat{u}^{n+1/2}$  is closer to 0 than 1, then  $\text{Sig}(\widehat{u}^{n+1/2} - 0.5)$  is closer to 0 than 1. Thus using  $\text{Sig}$  has the same two benefits as using  $\text{Proj}_{[0,1]}$  discussed above.

Note that the formula of  $\widehat{u}^{n+1/2}$  has a bias term  $\tau b^n$ . In the second method, we can combine the shifting scalar 0.5 with the bias by shifting the bias by 0.5. Therefore, our updating formula for  $\widehat{u}^{n+1}$  with  $\text{Sig}$  is

$$(4.3) \quad \widehat{u}^{n+1} = Q_\gamma \circ \text{Sig}(\widehat{u}^{n+1/2}).$$

In our numerical experiments, the proposed methods with activation  $Q_\gamma \circ \text{Sig}$  work slightly better than that with  $Q_\gamma \circ \text{Proj}_{[0,1]}$ . In the rest of this section, we present our methods using  $\text{Sig}$  and activation function  $Q_\gamma \circ \text{Sig}$ . We remark that  $\text{Sig}$  and  $Q_\gamma \circ \text{Sig}$  can be replaced by  $\text{Proj}_{[0,1]}$  and  $Q_\gamma \circ \text{Proj}_{[0,1]}$ , respectively.

We call the procedure  $\widehat{u}^n \rightarrow \widehat{u}^{n+1/2} \rightarrow \widehat{u}^{n+1}$  a DB-I, denoted by  $B_1^{n+1}$ , see Figure 2(a) for an illustration with activation  $Q_\gamma \circ \text{Sig}$ . The  $B_1^{n+1}$  contains trainable parameters  $\widetilde{W}^n, b^n$ . The input for  $B_1^n$  includes the output of the previous block  $u^n$ , and  $F(f)$ .

In a DB-I, we have

$$(4.4) \quad \widehat{u}^{n+1/2} = \widetilde{W}^n * \widehat{u}^n + \widetilde{b}^n$$

with

$$\widetilde{W}^n = I_{\text{id}} + \tau \lambda \epsilon \widehat{W}_\Delta + \tau \widehat{W}^n, \quad \widetilde{b}^n = \tau(b^n - F(f)),$$

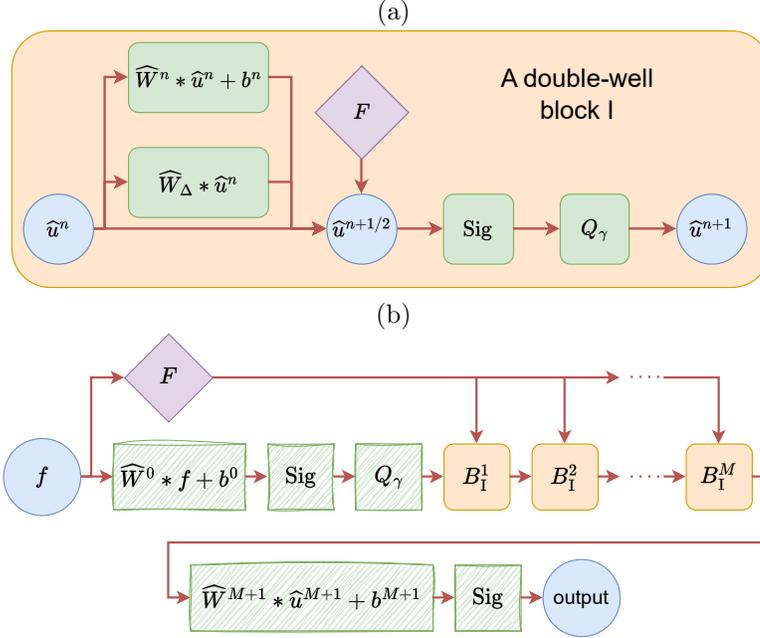


Fig. 2: For model (3.1): (a) An illustration of a double-well block I (DB-I) with activation  $Q_\gamma \circ \text{Sig}$ . (b) An illustration of the double-well net I (DN-I) with activation  $Q_\gamma \circ \text{Sig}$ . The architecture in (a) is the detailed representation of the block  $B_I^k$ 's in (b). In both figures, the disk represents input, output and intermediate variables. The diamond represents the functional  $F$ . In (a), rectangles represent operations applied in a DB-I. In (b), sketched rectangles represent operations in the input layer and final layer, normal rectangles represent DB-I's. To better present the architecture, some scalar factors are omitted.

where  $I_{\text{id}}$  denotes the identity kernel so that  $I_{\text{id}} * \hat{u} = \hat{u}$  for any  $\hat{u}$ . Thus a DB-I is a convolutional layer with weight kernel  $\widehat{W}^n$ , a heavy bias term  $\widehat{b}^n$ , and activation  $Q_\gamma \circ \text{Proj}_{[0,1]}$  or  $Q_\gamma \circ \text{Sig}$ .

**Double-well net I.** Given an image  $f$ , in order to use DB-I's to solve (3.1), we need an initial condition  $u^0$ . We propose to use a convolutional layer followed by a sigmoid function and fixed point iterations to generate  $\hat{u}^0$ :

$$(4.5) \quad \hat{u}^0 = H(f) = Q_\gamma \circ \text{Sig}(\widehat{W}^0 * f + b^0).$$

Since  $\hat{u}^n \in \mathbb{R}^{N_1 \times N_2}$  for any  $n$ , we set  $\widehat{W}^0$  as a convolution kernel with one output channel. The input image  $f$  is also passed to  $F$  to compute  $F(f)$ , where  $F$  will be chosen as a UNet class demonstrated in Figure 1. Then  $\hat{u}^0$  is passed through every DB-I sequentially and  $F(f)$  is passed to all DB-I's. Assume there are  $M$  blocks (DB-I's), and denote the  $m$ -th DB-I by  $B_I^m$ . After  $B_I^M$ , we add a convolution layer followed by a sigmoid function. Denote the kernel and bias in the last convolution layer by  $\widehat{W}^M$  and  $b^M$ , respectively. The double-well net I (DN-I) is formulated as

$$(4.6) \quad P_I(f) = \text{Sig}(\widehat{W}^M * (B_I^M(\dots B_I^2(B_I^1(\hat{u}^0, F(f)), F(f)) \dots, F(f)) + b^M),$$

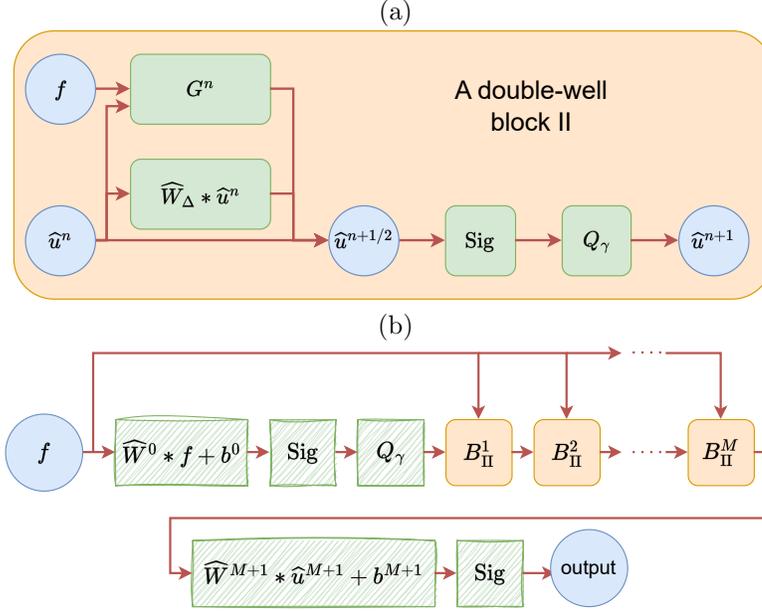


Fig. 3: For model (3.7): (a) An illustration of a double-well block II (DB-II) with activation  $Q_\gamma \circ \text{Sig}$ . (b) An illustration of the double-well net II (DN-II) with activation  $Q_\gamma \circ \text{Sig}$ . The architecture in (a) is the detailed representation of the block  $B_{\text{II}}^k$ 's in (b). In both figures, the disk represents input, output and intermediate variables. In (a), rectangles represent operations applied in a DB-II. In (b), sketched rectangles represent operations in the input layer and final layer, normal rectangles represent DB-II's. To better present the architecture, some scalar factors are omitted.

where  $u^0$  is defined in (4.5). We illustrate the architecture of DN-I in Figure 2(b).

In general, DN-I is a convolutional neural network (CNN) with  $M + 1$  layers. The main differences between DN-I from classical CNN are: (1) DN-I uses a subnetwork  $F(f) + b^n$  as the bias term in each layer, and the subnetwork  $F(f)$  is the same across all layers, whereas existing CNNs only use  $b^n$  as bias; (2) for the activation function, DN-I uses  $Q_\gamma \circ \text{Sig}$  or  $Q_\gamma \circ \text{Proj}_{[0,1]}$ , which are derived from the double-well potential, while classical CNN uses ReLU or Sig, which are designed empirically; (3) unlike existing networks, DN-I has a mathematical background: it is an operator-splitting scheme that approximately solves the Potts model.

PottsMGNet [76] is another neural network derived from the Potts model and operator-splitting method which gives a clear explanation for encode-decoder type of neural networks like UNet. However, the mathematical treatments and splitting strategy of DN-I are different from those of PottsMGNet. For the mathematical treatment of the Potts model, to drive the segmentation function  $u$  to be binary, a regularized softmax operator [53] is used in PottsMGNet, leading to the activation function being a sigmoid function. DN-I uses the double-well potential which leads to the activation function being a fixed-point iteration that minimize the potential. In the splitting strategy, inspired by the multigrid method, PottsMGNet uses control variables to represent the region force  $F(f)$  at several scales. Thus  $F(f)$  evolves

over time and is linear in  $f$  and  $u$ . In DN-I,  $F(f)$  is assumed to be a complicated function of  $f$ . Inspired by the approximation theory of deep neural networks, it is approximated by a subnetwork. Such an approximation is fixed and independent of time. This subnetwork is used as a heavy bias in each layer of DN-I.

**4.3. Double-well net for model (3.7).** For  $G(u, f, \mathbf{x}, t)$  in Model II in (3.7) at each time step  $t = t^n$ , we use a UNet specified by channel vector  $\mathbf{c}$  to represent it, see Section 5 for details.

**A Double-well block II.** In a Double-well Block II (DB-II), we first compute  $\widehat{u}^{n+1/2}$  according to (3.15), and then apply (4.2) or (4.3) to get  $\widehat{u}^{n+1}$ . A DB-II with activation  $Q_\gamma \circ \text{Sig}$  is visualized in Figure 3(a).

DB-II is different from DB-I in terms of how we treat the UNet class and its dependency on time. In DB-I, the UNet class is contained in the bias term and does not apply any convolution to  $\widehat{u}^n$ . This class is independent of  $t$  in the whole network, i.e., all blocks share the same UNet class and weight parameters. In DB-II, the UNet class is an operator that is directly applied to  $\widehat{u}^n$ . In DB-II, the UNet class depends on  $t$ , and its weight parameters can be different from block to block.

**Double-well net II.** For double-well net II (DN-II), we use (4.5) as an initial condition. Denote the  $m$ -th DB-II by  $B_{\text{II}}^m$  and assume we use  $M$  blocks. After  $B_{\text{II}}^M$ , we add a convolutional layer followed by a sigmoid function. The DN-II is formulated as

$$(4.7) \quad P_{\text{II}}(f) = \text{Sig}(\widehat{W}^M * (B_{\text{II}}^M(\cdots (B_{\text{II}}^2 \circ B_{\text{II}}^1(\widehat{u}^0, f), f) \cdots), f) + b^M).$$

An illustration of DN-II is shown in Figure 3(b).

Compared to DN-I, DN-II is more like the classical CNNs, while it still has three differences from existing networks for image segmentation: (1) in DN-II, the input image  $f$  is used at each block, instead of only at the beginning of the whole network; (2) For the activation function, DN-II uses  $Q_\gamma \circ \text{Sig}$  or  $Q_\gamma \circ \text{Proj}_{[0,1]}$ , which are derived from the double-well potential, while classical CNN uses ReLU or Sig, which are designed empirically; (3) unlike existing networks, DN-II has a mathematical background: it is an operator-splitting scheme that approximately solves the Potts model. Unlike DN-I, which fixes  $F(f)$  over time and assumes  $F(f)$  is only a function of the input image  $f$ , DN-II combines  $F(f)$  and the control variables and represents the combination using a subnetwork which is applied to both  $f$  and  $u$  and whose parameters evolve over time.

**5. Numerical experiments.** In this section, we demonstrate the performance of the proposed models by systematic numerical experiments. Our PyTorch code is available at <https://github.com/liuhaozm/Double-well-Net>. We consider three datasets: (i) The MARA10K data set [17], which contains 10000 images with pixel-level salient labeling. We resize the images to the size  $192 \times 256$  and select 2500 images for training and 400 images for testing. (ii) The Extended Complex Scene Saliency Dataset (ECSSD) [71]. ECSSD is a semantic segmentation data set containing 1000 images with complex backgrounds and manually labeled masks. We resize all images to the size  $192 \times 256$ , using 800 images for training and 200 images for testing. (iii) The Retinal Images vessel Tree Extraction dataset (RITE) [35]. RITE is a dataset for the segmentation and classification of arteries and veins on the retinal fundus. This dataset contains 40 images. We resize all images to the size  $256 \times 256$ , use 20 images for training and 20 images for testing. We train each model with Adam optimizer [39] and 400 epochs for MARA10K, and 600 epochs for ECSSD and RITE. Without further specification, for model DN-I, we use the UNet class with channels

$\mathbf{c} = [128, 128, 128, 128, 256]$ , and 10 blocks. We set  $\tau = 0.2$ . For DN-II, we use the UNet class with channels  $\mathbf{c} = [64, 64, 64, 128, 128]$  for each block and 3 blocks. We set  $\tau = 0.5$ . In both models, without specification, we set  $\gamma = 3, \lambda\varepsilon = 1$ , and  $\alpha = 2\tau\lambda/\varepsilon = 15$ . We use (4.5) to compute  $u^0$ , where  $W^0$  and  $b^0$  are to be learned from data. In all experiments, we use DN-I and DN-II to denote the networks with activation  $Q_\gamma \circ \text{Sig}$ , and use DN-I Proj and DN-II Proj to denote the networks with activation  $Q_\gamma \circ \text{Proj}_{[0,1]}$ .

We compare the proposed models with state-of-the-art image segmentation networks, including UNet [66], UNet++ [94], MANet [24], and DeepLabV3+ [14]. In our experiments, UNet++, MANet and DeepLabV3+ are implemented using the Segmentation Models PyTorch package [82]. For these models, the final layer is a sigmoid function. Thus, the output is a matrix with elements between 0 and 1. To obtain a binary segmentation result, we use a simple threshold  $T$  as

$$(5.1) \quad T \circ P(f) = \begin{cases} 1 & \text{if } P(f) \geq 0.5, \\ 0 & \text{if } P(f) < 0.5. \end{cases}$$

To measure the similarity between a model prediction and the given mask, we use accuracy and dice score. Given a set of test images  $\{f_k\}_{k=1}^K$  of size  $N_1 \times N_2$  and their segmentation masks  $\{v_k\}_{k=1}^K$ , for model  $P$ , we define the accuracy as

$$(5.2) \quad \text{accuracy} = \frac{1}{K} \sum_{k=1}^K \left[ \frac{|[T \circ P(f_k)] \cap v_k|}{N_1 N_2} \times 100\% \right]$$

and the dice score as

$$(5.3) \quad \text{dice} = \frac{1}{K} \sum_{k=1}^K \left[ \frac{2|[T \circ P(f_k)] \cap v_k|}{|[T \circ P(f_k)]| + |v_k|} \right],$$

where  $|v|$  denotes the number of nonzero elements of a binary function  $v$ , and  $\cap$  is the logic ‘and’ operation.

**5.1. Comparison with other networks.** We first consider MARA10K. The comparison of accuracy and dice score between the proposed models and other networks are presented in Table 1. In terms of performance, both activations  $Q_\gamma \circ \text{Sig}$  and  $Q_\gamma \circ \text{Proj}_{[0,1]}$  work similarly for DN-I and DN-II. Compared to UNet, the accuracy of the proposed models is 1% higher. Compared to other networks, the proposed models also provide higher accuracy and dice score.

We present some selected segmentation results in Figure 5. The selected results are not the best or worst ones. There are many test images with similar results. These images are selected to show the visual differences between the proposed methods and existing networks. In these examples, the proposed models accurately segment the given images. The predicted results are very similar to the given masks, while the other models more or less mistakenly segment some objects or miss some parts.

To measure the complexity of different models, we also present the number of parameters of each model in Table 1. The proposed models have the least number of parameters compared to other models. The number of parameters of the proposed models is less than one-third of that of UNet and MANet, and less than half of that of UNet++ and DeepLabV3+.

We present the evolution of loss, accuracy, and dice score of all models in Figure 4. They are plotted every 20 epochs. In Figure 4(a), we observe that the proposed

	Accuracy	Dice score	Number of Parameters
DN-I	<b>95.95%</b>	<b>0.9067</b>	$9.86 \times 10^6$
DN-I Proj	<b>95.95%</b>	0.9059	$9.86 \times 10^6$
DN-II	95.93%	<b>0.9067</b>	<b><math>9.21 \times 10^6</math></b>
DN-II Proj	95.68%	0.9003	<b><math>9.21 \times 10^6</math></b>
UNet	94.74%	0.8758	$31.04 \times 10^6$
UNet++	95.66%	0.8992	$26.08 \times 10^6$
MANet	95.33%	0.8936	$31.78 \times 10^6$
DeepLabV3+	95.41%	0.8937	$22.44 \times 10^6$

Table 1: Comparison of the accuracy, dice score, and number of parameters of DN-I and DN-II with UNet, UNet++, MANet, and DeepLabV3+ on MARA10K. Here DN-I and DN-II denote the proposed networks with activation  $Q_\gamma \circ \text{Sig}$ , and DN-I Proj and DN-II Proj denote the proposed networks with activation  $Q_\gamma \circ \text{Proj}_{[0,1]}$ .

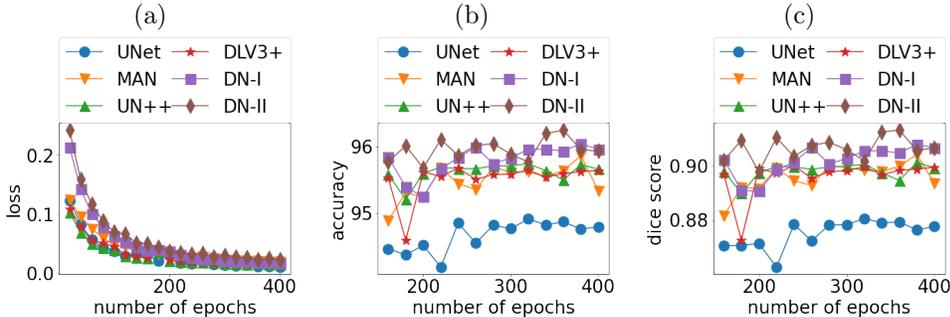


Fig. 4: Comparison of the histories of (a) training loss, (b) accuracy and (c) dice score of DN-I and DN-II with UNet, UNet++ (UN++), MANet (MAN), and DeepLabV3+ (DLV3+) on MARA10K.

models can reduce the loss as effectively as other models. Figures 4(b) and (c) show that for most of the time during training, the proposed models consistently give the highest accuracy and dice score.

We next consider ECSSD and RITE. The comparisons of DN-I and DN-II with other networks are shown in Table 2. For RITE, because the targets are vessels that are very thin, a smaller weight for the length penalty term should be used. We set  $\lambda\varepsilon = 0.25$  for both DN-I and DN-II,  $\lambda\varepsilon = 0.025$  for DN-I Proj, and  $\lambda\varepsilon = 0.05$  for DN-II Proj. For ECSSD and RITE, both DN-I and DN-II outperform other networks, with DN-I Proj providing the highest accuracy and dice score for ECSSD and DN-II providing the highest accuracy and dice score for RITE. The comparison of some selected segmentation results is shown in Figure 6. For ECSSD, DN-I and DN-II can successfully segment the target with smooth boundaries, while other networks either fail to segment the target or have noise in the results. For RITE, DN-I and DN-II provide clear main branches of vessels, while results by other networks contain scattered noise.

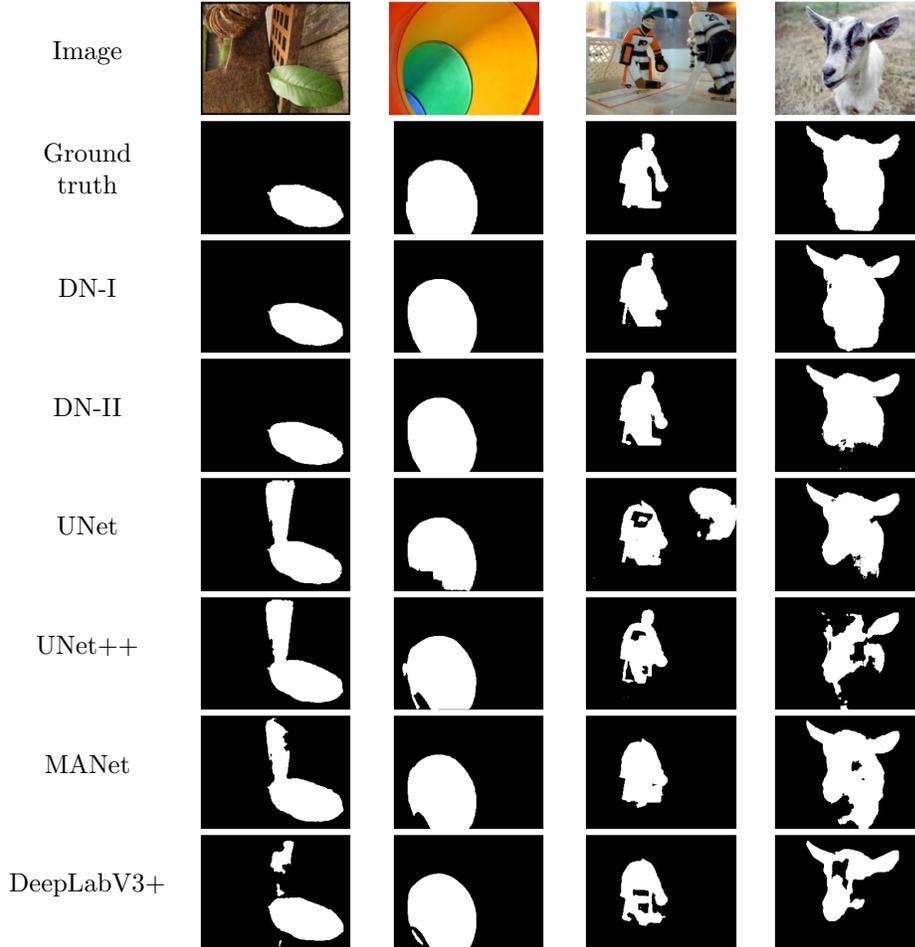


Fig. 5: Segmentation examples in the comparison of DN-I and DN-II with UNet, UNet++, MANet, and DeepLabV3+ on MARA10K.

**5.2. Effects of hyperparameters of DN-I.** We explore the effects of hyperparameters of DN-I using MARA10K. Except for the weight parameters in the neural network, DN-I has several hyperparameters: the number of blocks  $M$ , the network architecture in each block which is specified by  $\mathbf{c}$ , the time step  $\tau$ , the coefficient of diffusion  $\lambda$ , and the number of iterations of the fixed point iteration  $\gamma$ . In this subsection, we fix the network architecture as  $\mathbf{c} = [128, 128, 128, 128, 256]$  and explore the effects of other hyperparameters. For better visualization of the comparison, we present results from the 200th to 400th epoch. The accuracy, dice score, and loss are plotted every 20 epochs.

For the number of blocks, we test  $M = 1, 10, 40$  and fix  $\tau = 0.2, \lambda\varepsilon = 1, \gamma = 3$ . For a fixed  $F$ , the number of blocks corresponds to the number of iterations (time stepping) in the operator-splitting algorithm. The comparisons of the accuracy and dice score are shown in Figure 7. We observe that DN-I with  $M = 10$  provides the best results. Note that adding additional blocks only slightly increases the number

	ECSSD		RITE	
	Accuracy	Dice Score	Accuracy	Dice Score
DN-I	91.33%	0.8063	95.78%	0.7174
DN-I Proj	<b>91.47%</b>	<b>0.8084</b>	95.06%	0.6809
DN-II	91.17%	0.8056	<b>96.06%</b>	<b>0.7368</b>
DN-II Proj	90.74%	0.7945	95.89%	0.7145
UNet	90.94%	0.7941	95.00%	0.6760
UNet++	88.00%	0.7305	95.00%	0.6734
MANet	89.40%	0.7575	94.94%	0.6602
DeepLabV3+	89.17%	0.7571	93.19%	0.5209

Table 2: Comparison of the accuracy and dice score of DN-I and DN-II with UNet, UNet++, MANet, and DeepLabV3+ on ECSSD and RITE. Here DN-I and DN-II denote the proposed networks with activation  $Q_\gamma \circ \text{Sig}$ , and DN-I Proj and DN-II Proj denote the proposed networks with activation  $Q_\gamma \circ \text{Proj}_{[0,1]}$ .

of parameters of DN-I (the number of parameters in the models in Figure 7 are all around  $9.86 \times 10^6$ ), but significantly increases the model complexity. When  $M$  is too small, the model is too simple and cannot accomplish the segmentation task well. But a too-large  $M$  makes the model too complicated so that it can easily get stuck at a local minimizer during training. This test implies that as long as  $F$  is sufficiently good, a few operator-splitting iterations are enough to accomplish the segmentation task. In other words, we can find an effective  $F$  so that a few operator-splitting method iterations give desired segmentation.

We then set  $M = 10, \lambda\varepsilon = 1, \gamma = 3$ , and test  $\tau = 0.02, 0.2$  and  $2$ . The results are shown in Figure 8. In our model,  $\tau$  corresponds to the time step in the operator-splitting algorithm. In numerical methods, small  $\tau$  makes the solution evolve very slowly, and large  $\tau$  may make the algorithm unstable. This phenomenon is observed in Figure 8. Too small or too large  $\tau$  leads to slower decay of the loss and less accurate results.

The next test is for the number of fixed-point iterations. We fix  $M = 10, \tau = 0.2, \lambda\varepsilon = 1$  and test  $\gamma = 0, 3, 7$ . The results are shown in Figure 9. The number of fixed point iterations determines how well the result after each iteration approximates the binary function, i.e., the function value is either 0 or 1. Larger  $\gamma$  gives a better approximation and thus a better approximation of the Potts model. In Figure 9, too small  $\gamma$  makes the results worse, validating our argument. When  $\gamma$  is too large, like 7, the result also gets worse. That is because each fixed point iteration is not a simple function. If  $\gamma$  is too large, the operator, in general, gets very complicated, i.e., the loss functional has more local minimizers, making training the network more difficult.

In the last test, we study the effect of the diffusion coefficient  $\lambda$ . In this test, we set  $M = 10, \tau = 0.2, \gamma = 3$  and test  $\lambda\varepsilon = 0.05, 1$  and  $7.5$ . The results are shown in Figure 10. Although the diffusion operator is a linear operator which can be absorbed by  $W^n$  in each iteration, Figure 10 suggests that explicitly adding this diffusion operator with a proper coefficient improves the result.

**5.3. Effects of hyperparameters of DN-II.** Similar to DN-I, DN-II has hyperparameters: the number of blocks  $M$ , the network architecture in each block which is specified by  $\mathbf{c}$ , the time step  $\tau$ , the coefficient of diffusion  $\lambda$ , and the num-

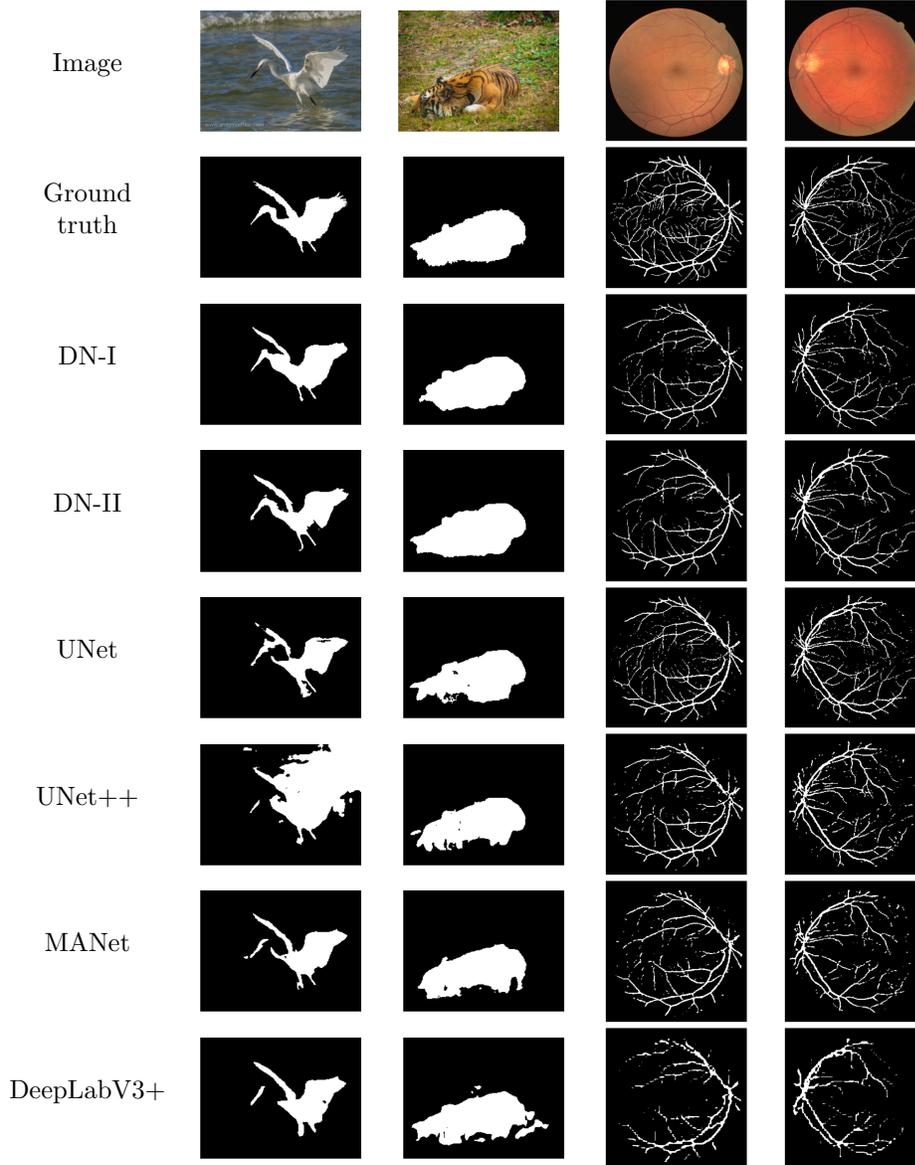
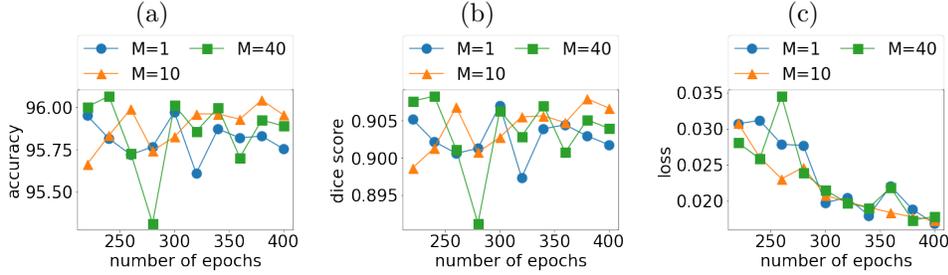
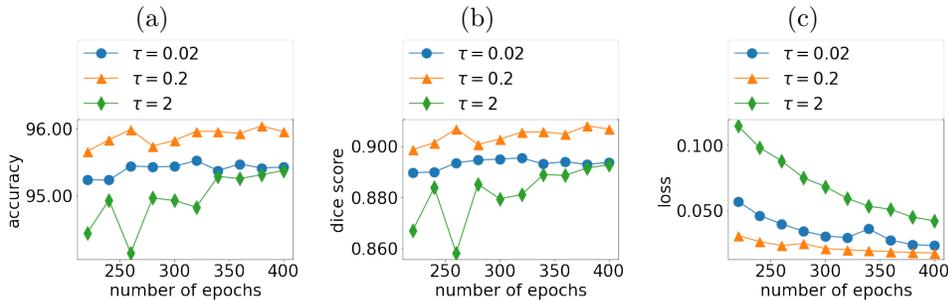
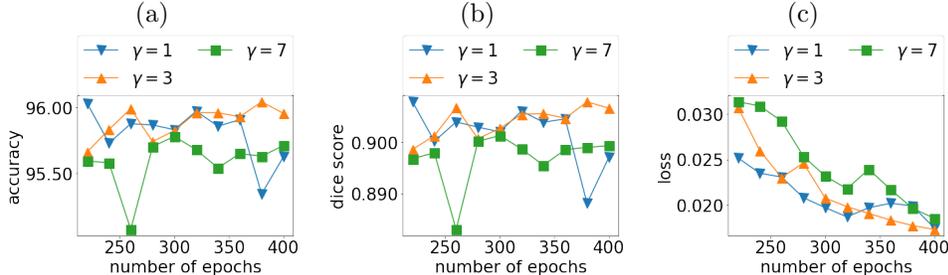


Fig. 6: Comparison between the proposed models and UNet, UNet++, MANet, and DeepLabV3+ on ECSSD (the first two columns) and RITE (the last two columns).

ber of iterations of the fixed point iteration  $\gamma$ . We fix the network architecture as  $\mathbf{c} = [64, 64, 64, 128, 128]$  and explore the effects of other hyperparameters. In this subsection, for better visualization of the comparisons, we present results from the 200th to 400th epoch. The accuracy, dice score, and loss are plotted every 20 epochs.

For the number of blocks, we set  $\tau = 0.5$ ,  $\lambda\varepsilon = 1$ ,  $\gamma = 3$ , and test  $M = 1, 3, 5$ . The number of blocks corresponds to the number of iterations in the operator-splitting algorithm. In DN-II, since every block is a UNet class but with different weight

Fig. 7: Effect of the number of blocks  $M$  for DN-I.Fig. 8: Effect of the time step  $\tau$  for DN-I.Fig. 9: Effect of the number of fixed point iteration  $\gamma$  for DN-I.

parameters, the number of weight parameters scales almost linearly with  $M$ . For  $M = 1, 3, 5$ , the number of parameters of the corresponding model is  $3.07M$ ,  $9.21M$ , and  $15.36M$ . The comparison of the accuracy, dice score, and training loss are shown in Figure 11. We observe that the model with  $M = 3$  gives the highest accuracy. We think it is because we need sufficiently many parameters so that the model can learn a good approximate solver of the Potts model. When  $M = 1$ , the number of parameters is too small, while when  $M = 5$ , the increased number of parameters makes the model more complicated and more difficult to train.

We then set  $M = 3, \lambda\varepsilon = 1, \gamma = 3$  and test  $\tau = 0.1, 0.5$  and  $1$ . The results are shown in Figure 12. Similar to our observations for DN-I, too small or too large  $\tau$

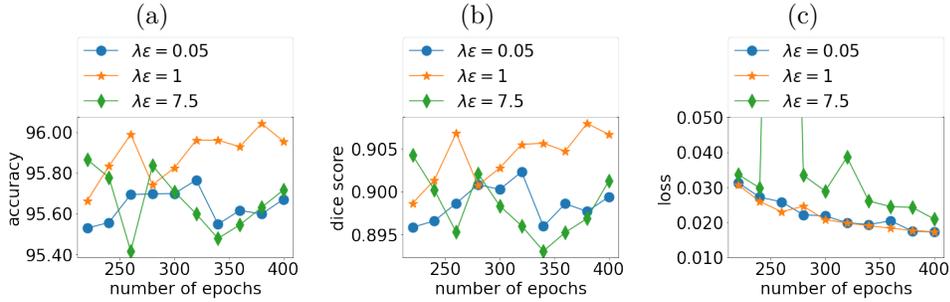


Fig. 10: Effect of the diffusion coefficient  $\lambda$  for DN-I. For the case  $\lambda\varepsilon = 7.5$  (the green curve with diamond) in (c), the loss has a large value of 0.2852 at the 260-th epoch. To better visualize the evolution of loss, the vertical axis is truncated to  $[0.01, 0.05]$ , making that point out of range and a sharp increase and decrease in the plot.

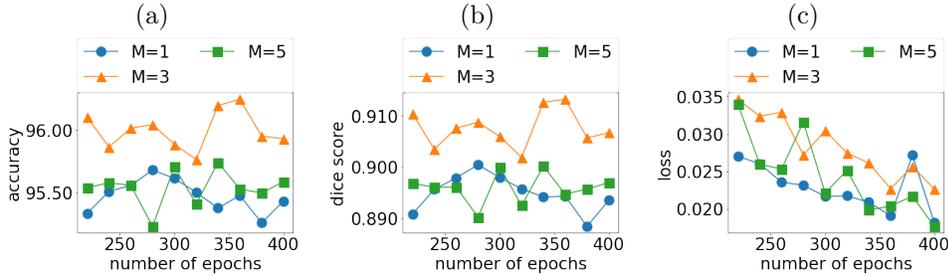


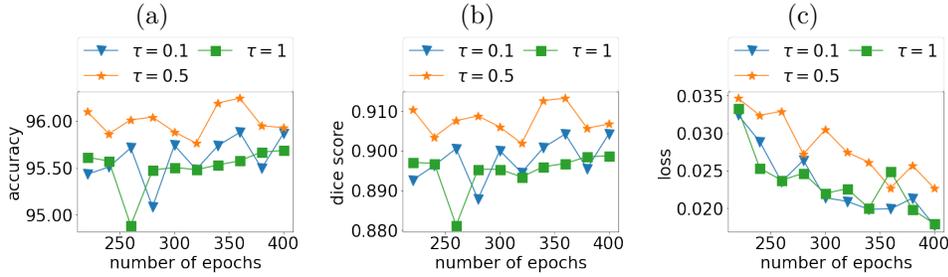
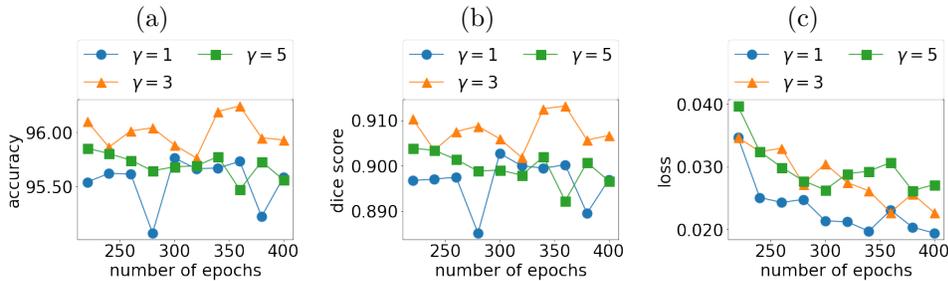
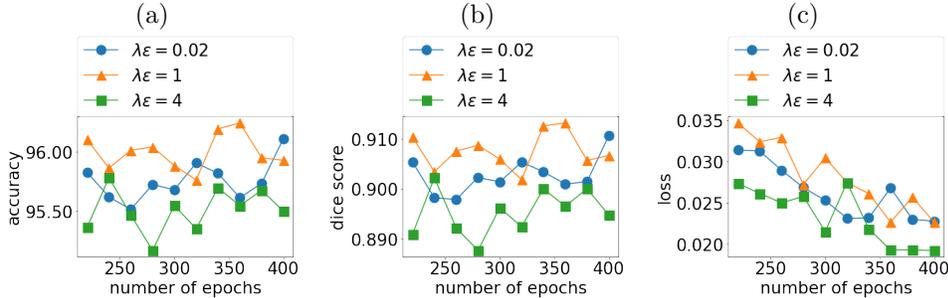
Fig. 11: Effect of the number of blocks  $M$  for DN-II.

leads to less accurate results.

The next test is for the number of fixed-point iterations. We fix  $M = 3, \tau = 0.5, \lambda\varepsilon = 1$  and test  $\gamma = 1, 3, 5$ . The results are shown in Figure 13. Our observation is similar to that of DN-I. The model with  $\gamma = 3$  gives the best result.

In the last test, we study the effect of the diffusion coefficient  $\lambda$ . In this test, we set  $M = 3, \tau = 0.5, \gamma = 3$  and test  $\lambda\varepsilon = 0.02, 1$  and 4. The results are shown in Figure 14. Again, Figure 14 suggests that explicitly adding this diffusion operator with a proper coefficient improves the result.

**5.4. The learned region force  $F(f)$  of DN-I.** Note that DN-I is an operator-splitting scheme solving (3.1), in which  $F(f)$  is the region force functional for the Potts model. By training DN-I, the region force is learned from the data. Under the same setting as in Section 5.1 for MARA10K, in Figure 15, we show the learned region force  $F(f)$  for two images. We observe the learned region force  $F(f)$  highlights the region to be segmented in white, while edges and textures information is also reflected in the learner region force  $F(f)$ . This clearly shows that DB-I provides a data-driven way to learn the region force functional in the Potts model. The evolution of the segmentation results are shown in Figure 16. The initial condition, output of the 3rd, 8th, 10th and the final layer are presented. Note that the final segmentation is generated by thresholding the output of the final layer. Starting from the initial condition, the

Fig. 12: Effect of the time step  $\tau$  for DN-II.Fig. 13: Effect of the number of fixed point iteration  $\gamma$  for DN-II.Fig. 14: Effect of the diffusion coefficient  $\lambda$  for DN-II.

segmentation is driven by region force  $F(f)$  and control variables towards the ground truth segmentation.

We then examine the learned region force  $F(f)$  on noisy data. For the network training, we adopt the progressive training strategy proposed in [76, Sec 7.3]. During training, we gradually increase the noise standard deviation (SD) as 0, 0.3, 0.5, 0.8, 1.0. For each noise level, we train the network for 400 epochs and then use the trained network parameters as the initial value for training the network for the next SD. The performance of the final trained network on training data with SD=1.0 is then tested on images with various noise levels. Two examples with SD=0.2 and SD=0.5 are shown in Figure 17. Even with substantial noise, DN-I still gives good

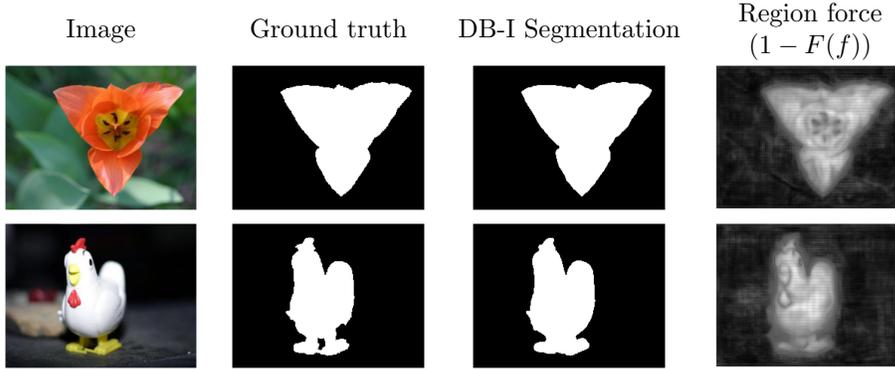


Fig. 15: The learned region force  $F(f)$  in DN-I on clean images. For better visualization, we normalized  $F(f)$  to be in  $[0, 1]$ .

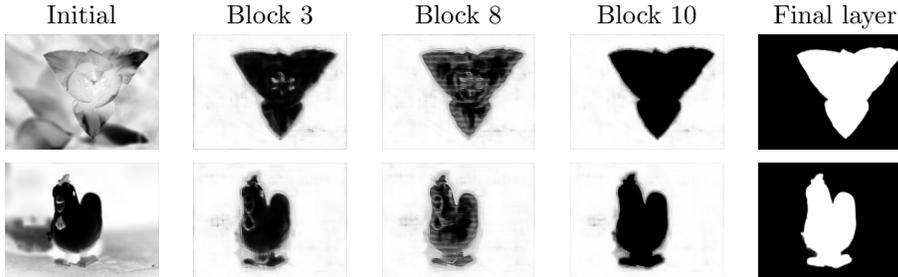


Fig. 16: Evolution of the segmentation across DB-I's in DN-I. Ten DB-I's are used in this experiment.

results. The learned region force  $F(f)$  highlights the region of the segmentation target. The evolution of the segmentation results are shown in Figure 18. Similar to our observation in Figure 16, the segmentation is driven by region force  $F(f)$  and control variables towards the ground truth segmentation.

**6. Conclusion.** In this paper, we propose two models for image segmentation, Double-well Net I and Double-well Net II, which are based on the Potts model, network approximation theory and operator-splitting methods. Starting from the Potts model, we approximate it using a double-well potential and then propose two control problems to find its minimizer. The control problems are time discretized by operator splitting methods whose structures are similar to the building block of neural networks. We then define a UNet class to represent some operators in the operator-splitting methods, which together with control variables are trained from data, leading to DN-I and DN-II. The UNet class is designed to capture multiscale features of images. DN-I and DN-II consist of several blocks, each of which corresponds to a one-time stepping of the operator-splitting schemes. All blocks use an activation that is a fixed-point iteration to minimize a double-well potential with a proximal term. Both networks have a mathematical explanation: they are operator-splitting methods to approximately solve the Potts model, which provide a new perspective

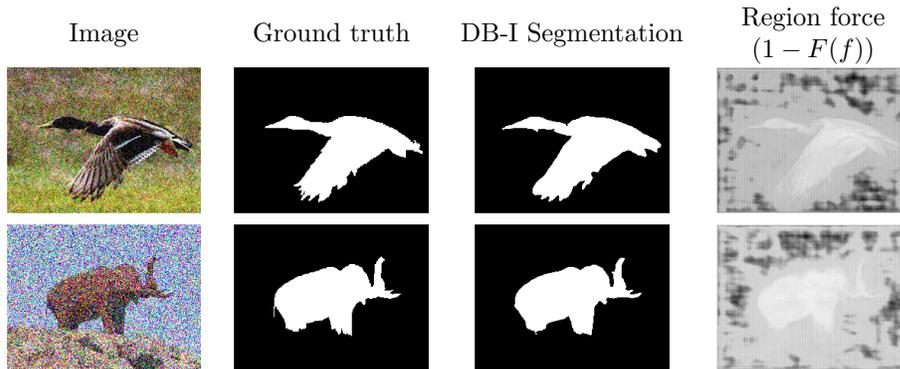


Fig. 17: The learned region force  $F(f)$  in DN-I on noisy images with (a) SD 0.2 and (b) SD 0.5. The DN-I is trained by progressive training. For better visualization, we normalized  $F(f)$  to be in  $[0, 1]$  and display  $1 - F(f)$ .

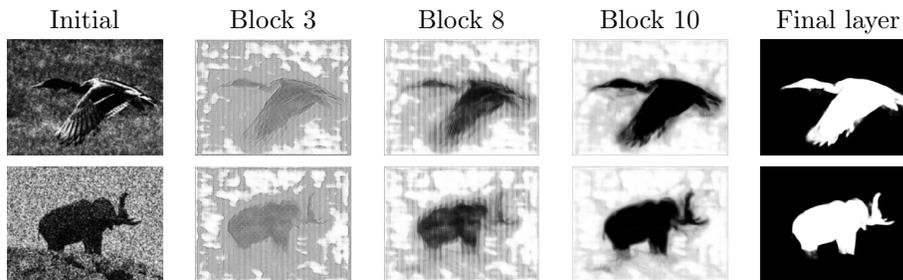


Fig. 18: Evolution of the segmentation across DB-I's in DN-I on noisy images with (a) SD 0.2 and (b) SD 0.5. The DN-I is trained by progressive training.. Ten DB-I's are used in this experiment.

for network design. The proposed models also bridge the MBO scheme with deep neural networks. In addition, since the region force functional in the Potts model is represented as a network in DN-I, DN-I provides a data-driven way to learn the region force functional. Systematic numerical experiments are conducted to study the performance of the proposed models. Compared to state-of-the-art networks, the proposed models provide higher accuracy and dice scores with a much smaller number of parameters.

### References.

- [1] E. BAE, X.-C. TAI, AND W. ZHU, *Augmented Lagrangian method for an Euler's elastica based segmentation model that promotes convex contours.*, Inverse Problems & Imaging, 11 (2017).
- [2] A. R. BARRON, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information Theory, 39 (1993), pp. 930–945.
- [3] M. BENNING, E. CELLEDONI, M. J. EHRHARDT, B. OWREN, AND C.-B. SCHÖNLIEB, *Deep learning as optimal control problems: Models and numerical methods*, Journal of Computational Dynamics, 6 (2019), pp. 171–198.

- [4] A. BONITO, A. CABOUSSAT, AND M. PICASSO, *Operator splitting algorithms for free surface flows: Application to extrusion processes*, in Splitting Methods in Communication, Imaging, Science, and Engineering, Springer, 2017, pp. 677–729.
- [5] Y. BOYKOV, O. VEKSLER, AND R. ZABIH, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239.
- [6] M. BUKAČ, S. ČANIĆ, R. GLOWINSKI, J. TAMBAČA, AND A. QUAINI, *Fluid–structure interaction in blood flow capturing non-zero longitudinal structure displacement*, Journal of Computational Physics, 235 (2013), pp. 515–541.
- [7] X. CAI, R. CHAN, AND T. ZENG, *A two-stage image segmentation method using a convex variant of the Mumford–Shah model and thresholding*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 368–390.
- [8] H. CAO, Y. WANG, J. CHEN, D. JIANG, X. ZHANG, Q. TIAN, AND M. WANG, *Swin-Unet: Unet-like pure transformer for medical image segmentation*, in European Conference on Computer Vision, Springer, 2022, pp. 205–218.
- [9] V. CASELLES, R. KIMMEL, AND G. SAPIRO, *Geodesic active contours*, International Journal of Computer Vision, 22 (1997), p. 61.
- [10] A. CHAMBOLLE AND T. POCK, *A first-order primal-dual algorithm for convex problems with applications to imaging*, Journal of Mathematical Imaging and Vision, 40 (2011), pp. 120–145.
- [11] T. CHAN AND W. ZHU, *Level set based shape prior segmentation*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 2, IEEE, 2005, pp. 1164–1170.
- [12] T. F. CHAN, S. ESEDOĞLU, AND M. NIKOLOVA, *Algorithms for finding global minimizers of image segmentation and denoising models*, SIAM Journal on Applied Mathematics, 66 (2006), pp. 1632–1648.
- [13] T. F. CHAN AND L. A. VESE, *Active contours without edges*, IEEE Transactions on Image Processing, 10 (2001), pp. 266–277.
- [14] L.-C. CHEN, Y. ZHU, G. PAPANDREOU, F. SCHROFF, AND H. ADAM, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 801–818.
- [15] M. CHEN, H. JIANG, W. LIAO, AND T. ZHAO, *Efficient approximation of deep ReLU networks for functions on low dimensional manifolds*, Advances in Neural Information Processing Systems, 32 (2019).
- [16] M. CHEN, H. JIANG, W. LIAO, AND T. ZHAO, *Nonparametric regression on low-dimensional manifolds using deep ReLU networks: Function approximation and statistical recovery*, Information and Inference: A Journal of the IMA, 11 (2022), pp. 1203–1253.
- [17] M.-M. CHENG, N. J. MITRA, X. HUANG, P. H. TORR, AND S.-M. HU, *Global contrast based salient region detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 37 (2014), pp. 569–582.
- [18] C. K. CHUI AND H. N. MHASKAR, *Deep nets for local manifold learning*, Frontiers in Applied Mathematics and Statistics, 4 (2018), p. 12.
- [19] M. DEFFERRARD, X. BRESSON, AND P. VANDERGHEYNST, *Convolutional neural networks on graphs with fast localized spectral filtering*, Advances in Neural Information Processing Systems, 29 (2016).
- [20] L.-J. DENG, R. GLOWINSKI, AND X.-C. TAI, *A new operator splitting method for the Euler elastica model for image smoothing*, SIAM Journal on Imaging Sciences, 12 (2019), pp. 1190–1230.

- [21] Y. DUAN, Q. ZHONG, X.-C. TAI, AND R. GLOWINSKI, *A fast operator-splitting method for Beltrami color image denoising*, Journal of Scientific Computing, 92 (2022), p. 89.
- [22] K. ELAMVAZHUTHI, B. GHARESIFARD, A. L. BERTOZZI, AND S. OSHER, *Neural ODE control for trajectory approximation of continuity equation*, IEEE Control Systems Letters, 6 (2022), pp. 3152–3157.
- [23] S. ESEDOĞLU AND Y.-H. R. TSAI, *Threshold dynamics for the piecewise constant Mumford–Shah functional*, Journal of Computational Physics, 211 (2006), pp. 367–384.
- [24] T. FAN, G. WANG, Y. LI, AND H. WANG, *MN-Net: A multi-scale attention network for liver and tumor segmentation*, IEEE Access, 8 (2020), pp. 179656–179665.
- [25] C. FINLAY, J. CALDER, B. ABBASI, AND A. OBERMAN, *Lipschitz regularized deep neural networks generalize and are adversarially robust*, arXiv preprint arXiv:1808.09540, (2018).
- [26] C. GARCIA-CARDONA, E. MERKURJEV, A. L. BERTOZZI, A. FLENNER, AND A. G. PERCUS, *Multiclass data segmentation using diffuse interface methods on graphs*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (2014), pp. 1600–1613.
- [27] R. GLOWINSKI, S. LEUNG, AND J. QIAN, *A penalization-regularization-operator splitting method for eikonal based traveltime tomography*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 1263–1292.
- [28] R. GLOWINSKI, H. LIU, S. LEUNG, AND J. QIAN, *A finite element/operator-splitting method for the numerical solution of the two dimensional elliptic Monge–Ampère equation*, Journal of Scientific Computing, 79 (2019), pp. 1–47.
- [29] R. GLOWINSKI, S. J. OSHER, AND W. YIN, *Splitting Methods in Communication, Imaging, Science, and Engineering*, Springer, 2017.
- [30] R. GLOWINSKI, T.-W. PAN, AND X.-C. TAI, *Some facts about operator-splitting and alternating direction methods*, Splitting Methods in Communication, Imaging, Science, and Engineering, (2016), pp. 19–94.
- [31] E. HABER AND L. RUTHOTTO, *Stable architectures for deep neural networks*, Inverse Problems, 34 (2017), p. 014004.
- [32] J. HE AND J. XU, *MgNet: A unified framework of multigrid and convolutional neural network*, Science China Mathematics, 62 (2019), pp. 1331–1354.
- [33] Y. HE, S. H. KANG, AND H. LIU, *Curvature regularized surface reconstruction from point clouds*, SIAM Journal on Imaging Sciences, 13 (2020), pp. 1834–1859.
- [34] S. HON AND H. YANG, *Simultaneous neural network approximation for smooth functions*, Neural Networks, 154 (2022), pp. 152–164.
- [35] Q. HU, M. D. ABRÀMOFF, AND M. K. GARVIN, *Automated separation of binary overlapping trees in low-contrast color retinal images*, in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013: 16th International Conference, Nagoya, Japan, September 22–26, 2013, Proceedings, Part II 16, Springer, 2013, pp. 436–443.
- [36] H. HUANG, L. LIN, R. TONG, H. HU, Q. ZHANG, Y. IWAMOTO, X. HAN, Y.-W. CHEN, AND J. WU, *UNet 3+: A full-scale connected UNet for medical image segmentation*, in ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 1055–1059.
- [37] F. JIA, J. LIU, AND X. C. TAI, *A regularized convolutional neural network for semantic image segmentation*, Analysis and Applications, (2020), pp. 1–19, <https://doi.org/10.1142/S0219530519410148>, <https://arxiv.org/abs/1907.05287>.

- [38] M. KASS, A. WITKIN, AND D. TERZOPOULOS, *Snakes: Active contour models*, International Journal of Computer Vision, 1 (1988), pp. 321–331.
- [39] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
- [40] Y. LAN, Z. LI, J. SUN, AND Y. XIANG, *DOSnet as a non-black-box PDE solver: When deep learning meets operator splitting*, Journal of Computational Physics, 491 (2023), p. 112343.
- [41] H. LI, J. LIU, L. CUI, H. HUANG, AND X. C. TAI, *Volume preserving image segmentation with entropy regularized optimal transport and its applications in deep learning*, Journal of Visual Communication and Image Representation, 71 (2020), p. 102845, <https://doi.org/10.1016/j.jvcir.2020.102845>, <https://doi.org/10.1016/j.jvcir.2020.102845>, <https://arxiv.org/abs/1909.09931>.
- [42] J. LI, B. CHENG, Y. CHEN, G. GAO, AND T. ZENG, *EWT: Efficient wavelet-transformer for single image denoising*, arXiv preprint arXiv:2304.06274, (2023).
- [43] X. LI, X. YANG, AND T. ZENG, *A three-stage variational image segmentation framework incorporating intensity inhomogeneity information*, SIAM Journal on Imaging Sciences, 13 (2020), pp. 1692–1715.
- [44] Y. LI, J. HU, G. NI, AND T. ZENG, *Deep CNN denoiser prior for blurred images restoration with multiplicative noise*, Inverse Problems and Imaging, 17 (2023), pp. 726–745.
- [45] H. LIU, M. CHEN, S. ER, W. LIAO, T. ZHANG, AND T. ZHAO, *Benefits of overparameterized convolutional residual networks: Function approximation under smoothness constraint*, in International Conference on Machine Learning, PMLR, 2022, pp. 13669–13703.
- [46] H. LIU, M. CHEN, T. ZHAO, AND W. LIAO, *Besov function approximation and binary classification on low-dimensional manifolds using convolutional residual networks*, in International Conference on Machine Learning, PMLR, 2021, pp. 6770–6780.
- [47] H. LIU, R. GLOWINSKI, S. LEUNG, AND J. QIAN, *A finite element/operator-splitting method for the numerical solution of the three dimensional Monge–Ampère equation*, Journal of Scientific Computing, 81 (2019), pp. 2271–2302.
- [48] H. LIU, X.-C. TAI, AND R. CHAN, *Connections between operator-splitting methods and deep neural networks with applications in image segmentation*, Annals of Applied Mathematics, 39 (2023), pp. 406–428.
- [49] H. LIU, X.-C. TAI, R. KIMMEL, AND R. GLOWINSKI, *A color elastica model for vector-valued image regularization*, SIAM Journal on Imaging Sciences, 14 (2021), pp. 717–748.
- [50] H. LIU, X.-C. TAI, R. KIMMEL, AND R. GLOWINSKI, *Elastica models for color image regularization*, SIAM Journal on Imaging Sciences, 16 (2023), pp. 461–500.
- [51] H. LIU AND D. WANG, *Fast operator splitting methods for obstacle problems*, Journal of Computational Physics, (2023), p. 111941.
- [52] H. LIU, H. YANG, M. CHEN, T. ZHAO, AND W. LIAO, *Deep nonparametric estimation of operators between infinite dimensional spaces*, Journal of Machine Learning Research, 25 (2024), pp. 1–67.
- [53] J. LIU, X. WANG, AND X.-C. TAI, *Deep convolutional neural networks with spatial regularization, volume and star-shape priors for image segmentation*, Journal of Mathematical Imaging and Vision, 64 (2022), pp. 625–645.
- [54] E. MERKURJEV, E. BAE, A. L. BERTOZZI, AND X.-C. TAI, *Global binary optimization on graphs for classification of high-dimensional data*, Journal of Mathematical Imaging and Vision, 52 (2015), pp. 414–435.

- [55] E. MERKURJEV, T. KOSTIC, AND A. L. BERTOZZI, *An MBO scheme on graphs for classification and image processing*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1903–1930.
- [56] B. MERRIMAN, J. K. BENICE, AND S. J. OSHER, *Motion of multiple junctions: A level set approach*, Journal of Computational Physics, 112 (1994), pp. 334–363.
- [57] L. MODICA, *The gradient theory of phase transitions and the minimal interface criterion*, Archive for Rational Mechanics and Analysis, 98 (1987), pp. 123–142.
- [58] L. MODICA AND S. MORTOLA, *Un esempio di  $\Gamma^-$ -convergenza*, Boll. Un. Mat. Ital. B, 14 (1977), pp. 285–299.
- [59] D. B. MUMFORD AND J. SHAH, *Optimal approximations by piecewise smooth functions and associated variational problems*, Communications on Pure and Applied Mathematics, (1989).
- [60] T. NGUYEN, M. PHAM, T. NGUYEN, K. NGUYEN, S. OSHER, AND N. HO, *Fourierformer: Transformer meets generalized Fourier integral theorem*, Advances in Neural Information Processing Systems, 35 (2022), pp. 29319–29335.
- [61] D. ONKEN, L. NURBEKYAN, X. LI, S. W. FUNG, S. OSHER, AND L. RUTHOTTO, *A neural network approach for high-dimensional optimal control applied to multiagent path finding*, IEEE Transactions on Control Systems Technology, 31 (2022), pp. 235–251.
- [62] K. OONO AND T. SUZUKI, *Approximation and non-parametric estimation of ResNet-type convolutional neural networks*, in International Conference on Machine Learning, PMLR, 2019, pp. 4922–4931.
- [63] S. OSHER AND J. A. SETHIAN, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.
- [64] T. POCK, A. CHAMBOLLE, D. CREMERS, AND H. BISCHOF, *A convex relaxation approach for computing minimal partitions*, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 810–817.
- [65] R. B. POTTS, *Some generalized order-disorder transformations*, in Mathematical Proceedings of the Cambridge Philosophical Society, vol. 48, Cambridge University Press, 1952, pp. 106–109.
- [66] O. RONNEBERGER, P. FISCHER, AND T. BROX, *U-Net: Convolutional networks for biomedical image segmentation*, in International Conference on Medical Image Computing and Computer-assisted Intervention, Springer, 2015, pp. 234–241.
- [67] L. ROSASCO, E. DE VITO, A. CAPONNETTO, M. PIANA, AND A. VERRI, *Are loss functions all the same?*, Neural computation, 16 (2004), pp. 1063–1076.
- [68] D. RUIZ-BALET AND E. ZUAZUA, *Neural ODE control for classification, approximation, and transport*, SIAM Review, 65 (2023), pp. 735–773.
- [69] L. RUTHOTTO AND E. HABER, *Deep neural networks motivated by partial differential equations*, Journal of Mathematical Imaging and Vision, 62 (2020), pp. 352–364.
- [70] Z. SHEN, H. YANG, AND S. ZHANG, *Deep network approximation characterized by number of neurons*, arXiv preprint arXiv:1906.05497, (2019).
- [71] J. SHI, Q. YAN, L. XU, AND J. JIA, *Hierarchical image saliency detection on extended CSSD*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 38 (2015), pp. 717–729.
- [72] L. SONG, Y. LIU, J. FAN, AND D.-X. ZHOU, *Approximation of smooth functionals using deep ReLU networks*, Neural Networks, 166 (2023), pp. 424–436.
- [73] H. SUN, X.-C. TAI, AND J. YUAN, *Efficient and convergent preconditioned ADMM for the Potts models*, SIAM Journal on Scientific Computing, 43 (2021),

- pp. B455–B478.
- [74] X.-C. TAI, O. CHRISTIANSEN, P. LIN, AND I. SKJÆLAAEN, *Image segmentation using some piecewise constant level set methods with MBO type of projection*, International Journal of Computer Vision, 73 (2007), pp. 61–76.
  - [75] X.-C. TAI, L. LI, AND E. BAE, *The Potts model with different piecewise constant representations and fast algorithms: a survey*, Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging: Mathematical Imaging and Vision, (2021), pp. 1–41.
  - [76] X.-C. TAI, H. LIU, AND R. CHAN, *PottsMGNet: A mathematical explanation of encoder-decoder based neural networks*, SIAM Journal on Imaging Sciences, 17 (2024), pp. 540–594.
  - [77] G. TE, W. HU, A. ZHENG, AND Z. GUO, *RGCNN: Regularized graph CNN for point cloud segmentation*, in Proceedings of the 26th ACM International Conference on Multimedia, 2018, pp. 746–754.
  - [78] M. THORPE AND Y. VAN GENNIP, *Deep limits of residual neural networks*, Research in the Mathematical Sciences, 10 (2023), p. 6.
  - [79] Y. VAN GENNIP AND A. L. BERTOZZI,  $\Gamma$ -convergence of graph Ginzburg-Landau functionals, Adv. Differential Equations, 17 (2012), pp. 1115–1180.
  - [80] K. WEI, K. YIN, X.-C. TAI, AND T. F. CHAN, *New region force for variational models in image segmentation and high dimensional data clustering*, Annals of Mathematical Sciences and Applications, 3 (2018), pp. 255–286.
  - [81] E. WEINAN, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics, 1 (2017), pp. 1–11.
  - [82] P. YAKUBOVSKIY, *Segmentation Models PyTorch*. [https://github.com/qubvel/segmentation\\_models.pytorch](https://github.com/qubvel/segmentation_models.pytorch), 2020.
  - [83] S. YAN, X.-C. TAI, J. LIU, AND H.-Y. HUANG, *Convexity shape prior for level set-based image segmentation method*, IEEE Transactions on Image Processing, 29 (2020), pp. 7141–7152.
  - [84] D. YANG AND J. SUN, *BM3D-Net: A convolutional neural network for transform-domain collaborative filtering*, IEEE Signal Processing Letters, 25 (2017), pp. 55–59.
  - [85] D. YAROTSKY, *Error bounds for approximations with deep ReLU networks*, Neural Networks, 94 (2017), pp. 103–114.
  - [86] M. YASHTINI AND S. H. KANG, *A fast relaxed normal two split method and an effective weighted TV approach for Euler’s elastica image inpainting*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 1552–1581.
  - [87] K. YIN AND X.-C. TAI, *An effective region force for some variational models for learning and clustering*, Journal of Scientific Computing, 74 (2018), pp. 175–196.
  - [88] J. YUAN, E. BAE, AND X.-C. TAI, *A study on continuous max-flow and min-cut approaches*, in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2217–2224.
  - [89] J. YUAN, E. BAE, X.-C. TAI, AND Y. BOYKOV, *A spatially continuous max-flow and min-cut framework for binary labeling problems*, Numerische Mathematik, 126 (2014), pp. 559–587.
  - [90] D. ZHANG AND L. M. LUI, *Topology-preserving 3D image segmentation based on hyperelastic regularization*, Journal of Scientific Computing, 87 (2021), p. 74.
  - [91] D. ZHANG, X.-C. TAI, AND L. M. LUI, *Topology-and convexity-preserving image segmentation based on image registration*, Applied Mathematical Modelling, 100 (2021), pp. 218–239.
  - [92] K. ZHANG, W. ZUO, Y. CHEN, D. MENG, AND L. ZHANG, *Beyond a Gaussian*

- denoiser: Residual learning of deep CNN for image denoising*, IEEE Transactions on Image Processing, 26 (2017), pp. 3142–3155.
- [93] D.-X. ZHOU, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis, 48 (2020), pp. 787–794.
- [94] Z. ZHOU, M. M. RAHMAN SIDDIQUEE, N. TAJBAKHSI, AND J. LIANG, *UNet++: A nested U-Net architecture for medical image segmentation*, in Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, Springer, 2018, pp. 3–11.