A Mathematical Explanation of UNet

Xue-Cheng Tai*, Hao Liu†, Raymond H. Chan‡, Lingfeng Li§

Abstract

The UNet architecture has transformed image segmentation. UNet's versatility and accuracy have driven its widespread adoption, significantly advancing fields reliant on machine learning problems with images. In this work, we give a clear and concise mathematical explanation of UNet. We explain what is the meaning and function of each of the components of UNet. We will show that UNet is solving a control problem. We decompose the control variables using multigrid methods. Then, operator-splitting techniques is used to solve the problem, whose architecture exactly recovers the UNet architecture. Connections between the proposed algorithm and general networks are also discussed. Our result shows that UNet is a one-step operator-splitting algorithm for the control problem.

Key words: UNet, operator splitting, deep neural network, image segmentation

1 Introduction

Deep neural networks have made remarkable successes in many tasks, including image segmentation [31–33, 48], image denoising [1, 40, 46], image classification, natural language processing [21], etc. Among these works, UNet [33] stands out as a renowned network and inspired a lot of following works [2, 8, 42, 48].

UNet was originally proposed for medical image segmentation. It consists of four components: encoder, decoder, bottleneck an skip-connections. Given an input image, the encoder part conducts dimension reduction and convert the image to a low-dimensional tensor. The bottleneck performs some operations on the tensor, after which the tensor is converted to the segmented image by the decoder. Skip-connections are used to directly pass information from encoder to decoder. UNet does a great job in medical image segmentation, and has garnered significant attention. Its encoder-decoder architecture inspired a lot of subsequent works, including DeepLab [8], SegNet [2], UNet++ [48] for image segmentation, SUNet [15], RDUNet [22] for image denoising.

A series of works have been aimed at elucidating the empirical successes of deep neural networks [4, 9, 43, 47] and establishing connections between deep learning and mathematical models [13, 31, 34, 39]. The current work is inspired by a series of earlier researches. In [13, 14], the authors initiated the idea to treat networks as discretized representations of continuous

*Norwegian Research Centre (NORCE), Nygårdsgaten 112, 5008 Bergen, Norway. Email: xtai@norceresearch.no, xuechengtai@gmail.com. The work of Xue-Cheng Tai is partially supported by HKRGC-NSFC Grant N-CityU214/19, HKRGC CRF Grant C1013-21GF and NORCE Kompetanseoppbygging program.

†Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. Email: haoliu@hkbu.edu.hk. The work of Hao Liu is partially supported by NSFC 12201530 and HKRGC ECS 22302123.

‡Lingnan University, Tuen Mun, Hong Kong SAR. Email: raymond.chan@ln.edu.hk. The work of Raymond H. Chan is partially supported by HKRGC GRF grants CityU1101120, CityU11309922, CRF grant C1013-21GF, and HKRGC-NSFC Grant N-CityU214/19.

§Hong Kong Center for Cerebro-Cardiovascular Health Engineering, Hong Kong SAR. The work of Lingfeng Li is supported by the InnoHK project at Hong Kong Centre for Cerebro-Cardiovascular Health Engineering (COCHE). Email: lfli@hkcoche.org

dynamical systems. The authors of [5] studied the connections between networks and control problems. PDE and ODE-motivated stable network architectures are proposed in [23, 34]. Inspired by the weak formulation of PDEs, [44] proposed weak adversarial networks for solving PDEs. This idea was further applied in [3] to solve constrained optimization problems. Many networks are designed with an encoder-decoder architecture, in which the encoder and decoder are expected to extract and reconstruct features of data, respectively. Analogies between this architecture and multiscale methods are pointed out in [23, 24]. In [25], the authors proposed to use operators with multigrid methods to extract and reconstruct features. In [26], the authors used networks based on the operator-splitting method to solve PDEs. For image processing, the regularizers of prior information are incorporated with networks to design new networks. Networks with volume-preserving properties and star-shape priors are proposed in [31] for image segmentation. Compactness priors are used in [45]. In [39], a multi-task deep variational model is proposed which variational models are incorporated into the loss functions. Based on the Chan-Vese model [7] and fields of experts regularizer, a novel deep neural network is proposed in [10] for image segmentation.

Based on the Potts model and operator-splitting methods, networks with mathematical explanations are proposed in [27, 28, 36]. In [36], the authors proposed PottsMGNet by integrating the Potts model, operator-splitting method, control problem, and multigrid method, which provides a mathematical explanation of the encoder-decoder-based networks. PottsMGNet demonstrated great performances in segmenting images with various noise levels using a single network. It was shown in [36] that most of the encoder-decoder-based neural networks are essentially operator-splitting algorithms solving certain control problems. The double-well net proposed in [27] utilizes the Potts model, operator-splitting methods, the double-well potential, and network representation theories. In double-well nets, a network is used to represent the region force term in the Potts model, providing a data-driven way to learn the region force term. The works mentioned above make connections among mathematical models, algorithms, and deep neural networks. However, the resulting networks are more or less different from UNet and cannot be directly applied to provide an explanation of UNet. In this paper, we aim to provide a clear and concise mathematical explanation of UNet. Building on the key concepts from [36], we rigorously formulate the problem to show that the network derived from the splitting-multigrid algorithms for the control problem corresponds exactly to UNet when only a single iteration of the algorithm is applied. In fact, UNet emerges as a special case of the more general algorithm described in [36]. The central ideas for multigrid methods we use in this work for solving minimization problems come from [35, 37, 38, 41]. The general explanations and convergence proofs provided in these works for multigrid methods present the method in a more general form, encompassing linear elliptic solvers as special cases and suits our proposed control problem well. Operator-splitting methods decompose complicated problems into multiple easy-to-solve sub-problems and are widely used in solving PDEs [17], inverse problems [16] and image processing [11, 12, 30]. We suggest readers to [18–20] for a comprehensive discussion on operator-splitting methods. Traditional splitting methods decompose the original problem into a small number of sub-problems. In the context of this work, the number of the decomposed sub-problems are rather large and thus need to introduce some hybrid splitting schemes as in Section 2.2 proposed in [36]

In this work, starting from a control problem, we will first derive its equivalent problem by introducing an indicator function. We then use the multigrid idea to decompose the control variables into different scales and utilize the hybrid splitting strategy to propose an operator-splitting method for the new problem. The algorithm consists of several sub-steps, each of which contains an explicit linear convolution step and an implicit step, where the implicit step has a closed-form solution which turns out to be the ReLU function. We show that the resulting algorithm exactly recovers the UNet architecture. Our results show that UNet is a one-step operator-splitting algorithm solving a control problem.

This paper is organized as follows: In Section 2, we present the control problem, derive its equivalent form using an indicator function, and introduce basic ideas of hybrid operator-splitting methods and multigrid methods. We discuss in Section 3 the decomposition of control variables and present our proposed operator-splitting method to solve the control problem. Solutions to subproblems in the proposed algorithm are presented in Section 4. We discuss connections between the proposed algorithm and general networks and how the proposed algorithm recovers UNet in Section 5, and conclude this paper in Section 6.

2 Proposed formulation

In this section, we present our control problem and briefly introduce hybrid operator-splitting methods and multigrid methods.

2.1 The control problem

Given an input image f, we consider the following initial value problem

$$\begin{cases} \frac{\partial u(\mathbf{x},t)}{\partial t} = W(\mathbf{x},t) * u(\mathbf{x},t) + d(t) - \ln \frac{u(\mathbf{x},t)}{1-u(\mathbf{x},t)}, \ (\mathbf{x},t) \in \Omega \times (0,T], \\ u(\mathbf{x},0) = H(f), \ \mathbf{x} \in \Omega, \end{cases} \tag{1}$$

where $W(\mathbf{x},t), d(t)$ are control variables that governs the dynamics of u, $*$ denotes convolution, $H(f)$ is some operation to generate initial condition from f, Ω is the domain where the images is defined and T is some fixed time. Due to the appearance of the term $\ln \frac{u}{1-u}$, the solution of the above equation is forced to be in $(0,1)$. For numerical consideration and to make the connection between operator-splitting methods and neural networks clearer, we introduce a constraint and consider the following constrained control problem

$$\begin{cases} \frac{\partial u}{\partial t} = W(\mathbf{x},t) * u(\mathbf{x},t) + d(t) - \ln \frac{u(\mathbf{x},t)}{1-u(\mathbf{x},t)}, \ (\mathbf{x},t) \in \Omega \times (0,T], \\ u(\mathbf{x},t) \geq 0, \\ u(\mathbf{x},0) = H(f), \ \mathbf{x} \in \Omega. \end{cases} \tag{2}$$

Due to the property of the term $\ln \frac{u}{1-u}$, the introduced constraint does not change the solution.

Next, we incorporate the constraint into the equation by introducing an indicator function. This technique has been used in designing fast operator-splitting methods for image processing [11, 12, 29, 30]. Define the set

$$\Sigma = \{u : u(\mathbf{x},t) \geq 0 \text{ for } (\mathbf{x},t) \in \Omega \times (0,T]\}$$

and its indicator function

$$\mathcal{I}_\Sigma(u) = \begin{cases} 0 & \text{if } u \in \Sigma, \\ \infty & \text{otherwise.} \end{cases}$$

Problem (2) is equivalent to the following unconstrained control problem

$$\begin{cases} \frac{\partial u}{\partial t} - W(\mathbf{x},t) * u - d(t) + \ln \frac{u}{1-u} + \partial \mathcal{I}_\Sigma(u) \ni 0, \ (\mathbf{x},t) \in \Omega \times (0,T], \\ u(\mathbf{x},0) = H(f), \ \mathbf{x} \in \Omega, \end{cases} \tag{3}$$

where $\partial \mathcal{I}_\Sigma$ denotes the subdifferential of $\mathcal{I}_\Sigma$.

By solving (3) for any input image f, We expect that $u(\mathbf{x},0)$ will evolve to $u(\mathbf{x},T)$ which is close to a binary function. For a given dataset $\{f_i, g_i\}_{i=1}^I$, we consider a control problem. Specifically, denote $\theta_1 = \{W(\mathbf{x},t), d(t)\}$ as the set of control variables, and $\mathcal{N}_1 : f \to u(\mathbf{x},T)$ as

the mapping from f to the solution of (3) at time T: $\mathcal{N}_1(f;\theta_1) = u(\mathbf{x}, T)$. We optimize θ_1 by solving

$$\min_{\theta_1} \sum_{i=1}^{I} \mathcal{L}(\mathcal{N}_1(f_i, \theta_1), g_i), \tag{4}$$

where $\mathcal{L}(\cdot,\cdot)$ is the loss function measuring the differences between its arguments. Common loss functions include logistic loss and hinge loss.

2.2 Hybrid splitting methods

We will use the hybrid splitting method proposed in [36] to solve (3). Refer to [18–20] for some general introduction to traditional splitting methods. In this subsection, we give a brief introduction to the hybrid splitting method. Consider a general initial value problem

$$\begin{cases} u_t + \sum_{m=1}^{M} \left(\sum_{k=1}^{c_m} \sum_{s=1}^{c_{m-1}} A_{k,s}^m(\mathbf{x}, t; u) + \sum_{k=1}^{c_m} S_k^m(\mathbf{x}, t; u) + \sum_{k=1}^{c_m} f_k^m(\mathbf{x}, t) \right) = 0 \text{ on } \Omega \times [0, T], \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}), \end{cases} \tag{5}$$

where $\{c_m\}_{m=0}^M$ are some given positive integers with $c_0 = c_M = 1$, $A_{k,s}^m, S_k^m$ are operators, f_k^m's are some given functions independent of u. The hybrid splitting method is a mixture of sequential splitting and parallel splitting. Briefly speaking, the hybrid splitting method arranges parallel splittings sequentially.

The algorithm of hybrid splitting is summarized in Algorithm 1. In the algorithm, all operators are distributed into M sequential sub-steps, each of which is a parallel splitting with c_m parallel pathways. The computation of each parallel pathway uses the c_{m-1} intermediate results from the previous sub-step. The structure of Algorithm 1 is illustrated in Figure 1.

Algorithm 1: A hybrid splitting scheme

Data: The solution u^n at time step t^n.
Result: The computed solution u^{n+1} at time step t^{n+1}.
Set $d_1 = 1, u_1^n = u^n$.
for $m = 1, ..., M$ **do**
  for $k = 1, ..., c_M$ **do**
    Compute $u_k^{n+m/M}$ by solving

$$\frac{u_k^{n+m/M} - u^{n+(m-1)/M}}{c_m \Delta t} = -\sum_{s=1}^{c_{m-1}} A_{k,s}^m(t^n; u_s^{n+(m-1)/M}) - S_k^m(t^{n+1}; u^{n+m/M}) - f_k^m(t^n). \tag{6}$$

  end for
  Compute $u^{n+m/M}$ as

$$u^{n+m/M} = \frac{1}{c_m} \sum_{k=1}^{c_m} u_k^{n+m/M}. \tag{7}$$

end for

The above scheme splits the original problem into M sequential steps with $m = 1, 2, \cdots .M$. Inside each sequential step, the problem is further split into c_m parallel steps for each m. For each of these parallel subproblems, we treat the operator S_k^m using implicit approximation and the operators $A_{k,s}^m$ using explicit approximations. It is shown in [36] that when all operators in (5) are linear, Algorithm 1 converges with first-order accuracy:

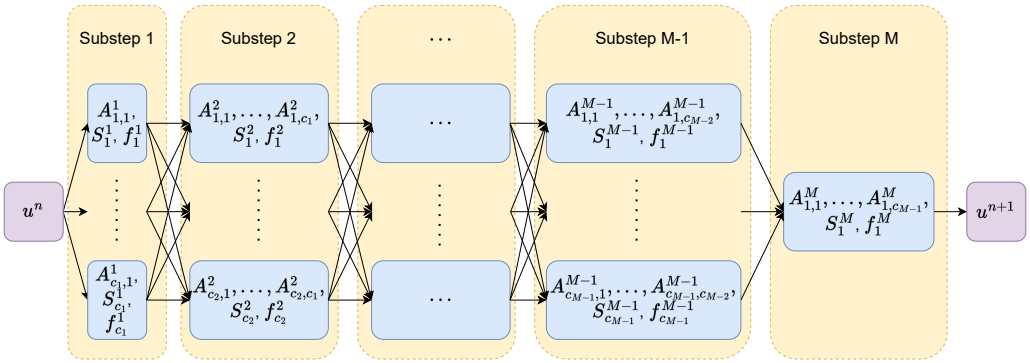

Figure 1: An illustration of Algorithm 1.

Theorem 1 (Theorem D.1 in [36])**.** For a fixed $T > 0$ and a positive integer N, set $\Delta t = T/N$. Let u^{n+1} be the numerical solution by Algorithm 1. Assume $A^m_{k,s}$'s and S^m_k's are Lipschitz with respect to $t, \mathbf{x}$, and are linear symmetric positive definite operators with respect to u. Assume Δt is small enough (i.e., N is large enough). We have

$$\|u^{n+1} - u(t^{n+1})\|_\infty = O(\Delta t) \tag{8}$$

for any $0 \le n \le N$.

In applying this algorithm to our control problem, the $A^m_{k,s}$ operators are coming from the decomposed control variables which are the convolutional kernels over the different levels of the multigrids explained in the next sections.

2.3 Multigrid discretizations

To demonstrate our splitting strategy, we will use the multigrid idea to decompose the control variables into components with different scales. in this subsection, we present the multigrid method for a general function f, which we will refer to as an image to remain consistent with the terminology.

Denote the original resolution (finest grid) of an image f by $\mathcal{T}$ with size $m \times n$, and grid step size h, with

$$m = 2^{s_1}, \quad n = 2^{s_2}$$

for some $h > 0$ and integers $s_1, s_2 > 0$. The image f is considered to have a constant value on each element (or called pixel) $[\alpha_1 h, (\alpha_1 + 1)h) \times [\alpha_2 h, (\alpha_2 + 1)h)$ for $\alpha_1 = 0, ..., m - 1$ and $\alpha_2 = 0, ..., n - 1$.

Set $\mathcal{T}^1 = \mathcal{T}$. Given grid $\mathcal{T}^j$, for the next level coarse grid $\mathcal{T}^{j+1}$, we downsample the number of grid points along each dimension by half. Following this process, we can generate a sequence of grids $\{\mathcal{T}^j\}_{j=1}^J$ with J denoting the coarsest level of grids and each $\mathcal{T}^j$ has grid size $m_j \times n_j$ and grid step size h_j with

$$m_j = 2^{s_1 - j + 1}, \quad n_j = 2^{s_2 - j + 1}, \quad h_j = 2^{j-1} h.$$

Denote $\mathcal{I}^j = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} = (\alpha_1, \alpha_2), \alpha_1 = 0, ..., m_j - 1, \ \alpha_2 = 0, ..., n_j - 1\}$. For a given grid $\mathcal{T}^j$, a set of piecewise-constant basis functions $\{\phi^j_\alpha\}_{\alpha \in I^j}$ is defined as

$$\phi^j_{\boldsymbol{\alpha}}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in [\alpha_1 h_j, (\alpha_1 + 1)h_j) \times [\alpha_2 h_j, (\alpha_2 + 1)h_j), \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

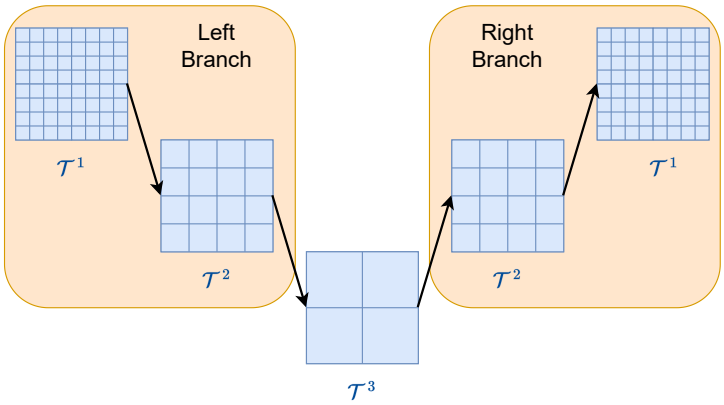

Figure 2: An illustration of a V-cycle of the multigrid method.

Let $\mathcal{V}^j = \text{span}(\{\phi_\alpha^j\}_{\alpha\in\mathcal{I}^j})$ be the linear space containing all the piecewise constant functions over grid $\mathcal{T}^j$, we have

$$\mathcal{V}^1 \supset \mathcal{V}^2 \supset \cdots \supset \mathcal{V}^J. \tag{10}$$

For each $f \in \mathcal{V}^j$, it can be expressed as $f(x,y) = \sum_{\boldsymbol{\alpha}\in\mathcal{I}^j} f_{\boldsymbol{\alpha}}^j \phi_{\boldsymbol{\alpha}}^j(x,y)$ with $f_{\boldsymbol{\alpha}}^j = f(\alpha_1 h_j, \alpha_2 h_j)$.

Next, we introduce the downsampling and upsampling operations that convert functions between different grids. Let $\mathcal{T}^j$ and $\mathcal{T}^{j+1}$ be two grids. Consider $f^{j+1} \in \mathcal{V}^{j+1}$. According to (10), there exists a function $f^j \in \mathcal{V}^j$ satisfying $f^j = f^{j+1}$. Denote the upsampling operator $\mathcal{U} : \mathcal{V}^{j+1} \to \mathcal{V}^j$ for any $j > 0$ so that

$$f^j = \mathcal{U}(f^{j+1}). \tag{11}$$

One can show that for $\boldsymbol{\alpha} \in \mathcal{I}^j$, it holds

$$(\mathcal{U}(f^j))_{\boldsymbol{\alpha}} = f_{\boldsymbol{\alpha}'}^{j+1} \text{ with } \alpha_1', \alpha_2' \text{ satisfying } 2\alpha_1' - 1 \le \alpha_1 \le 2\alpha_1',\ 2\alpha_2' - 1 \le \alpha_2 \le 2\alpha_2'. \tag{12}$$

The mapping discussed above is the simplest upsampling operator. One can also choose other upsampling operators that apply some operations while upsampling, such as interpolation or transpose convolution.

For the downsampling operator $\mathcal{D}^j : \mathcal{V}^j \to \mathcal{V}^{j+1}$, there are many ways to define it. For example, given a function $f^j \in \mathcal{V}^j$, we can define $\mathcal{D}^j$ as an averaging downsampling operator:

$$f^{j+1} = (\mathcal{U}^j(f^j))_{\boldsymbol{\alpha}} = \frac{1}{4} \sum_{\alpha_1'=2\alpha_1-1}^{2\alpha_1} \sum_{\alpha_2'=2\alpha_2-1}^{2\alpha_2} f_{\alpha_1',\alpha_2'}^j. \tag{13}$$

Another choice is the max pooling operator which is widely used in deep learning:

$$f^{j+1} = (\mathcal{U}^k(f^j))_{\boldsymbol{\alpha}} = \max_{\substack{\alpha_1'=2\alpha_1-1,2\alpha_1 \\ \alpha_2'=2\alpha_2-1,2\alpha_2}} f_{\alpha_1',\alpha_2'}^j. \tag{14}$$

3 The proposed algorithm

We decompose the control variables in (3) using the multigrid idea and then propose an algorithm based on the hybrid operator-splitting method to solve it. After these, it will then be shown that UNet is exactly one-step of the operator-splitting algorithm for the control problem.

3.1 Decomposition of control variables θ_1

In traditional multigrid methods, a popular framework is the "fine-grid $\rightarrow$ coarse grid $\rightarrow$ fine grid" strategy [6]. Such forms of V-cycle multigrid method can be interpreted as space decomposition and subspace correction [35, 37, 38, 41], see Figure 2 for an illustration. Traditional multigrid methods solve the decomposed subproblems by simple Gauss-Seidel or Jacobi iterations. In our approach shown here, we solve the subproblems by operating splitting sequentially or in parallel over the decomposed function subspaces.

We will decompose θ_1 into a sum of variables with different scales over the multigrids. Then, we use a hybrid splitting method to solve (3) so that all decomposed variables are distributed into several subproblems, which are solved sequentially or in parallel. Within one iteration of the splitting method, all decomposed variables are gone through. The general splitting idea is to split the operators based on a V-cycle according to the scale level. We assign several sub-steps to each scale level of each branch of the V-cycle. Each sub-step consists of several parallel splitting pathways.

We decompose all terms in the right-hand side of (3) via the following six steps:

(i) According to the idea of a V-cycle, we decompose $W(\mathbf{x},t)$ and $d(t)$ as

$$W(\mathbf{x},t) = A(\mathbf{x},t) + \widetilde{A}(\mathbf{x},t),\ d(t) = b(t) + \widetilde{b}(t). \tag{15}$$

These variables will be further decomposed next. Above, A, b are sums of control variables in the left branch of the V-cycle, and $\widetilde{A}, \widetilde{b}$ are sums of the control variables in the right branch. We also decompose the nonlinear operator as follows:

$$-\ln\frac{u}{1-u} - \partial\mathcal{I}(u) = S(u) + \widetilde{S}(u). \tag{16}$$

Here $S(u)$ contains nonlinear operations in the left branch and $\widetilde{S}(u)$ contains nonlinear operations in the right branch. In particular, we put $-\ln\frac{u}{1-u}$ in $\widetilde{S}(u)$ only, i.e., $S(u)$ only contains operator $\partial\mathcal{I}(u)$. Later, we will show our operator splitting method recovers UNet. The operation $-\ln\frac{u}{1-u}$ corresponds to the sigmoid layer at the end of UNet.

(ii) We further decompose the operators into components at different scales as:

$$A(\mathbf{x},t) = \sum_{j=1}^{J} A^j(\mathbf{x},t), \quad b(t) = \sum_{j=1}^{J-1} b^j(t), \quad S(u) = \sum_{j=1}^{J} S^j(u), \tag{17}$$

$$\widetilde{A}(\mathbf{x},t) = \sum_{j=1}^{J-1} \widetilde{A}^j(\mathbf{x},t) + A^*(\mathbf{x},t), \quad \widetilde{b}(t) = \sum_{j=1}^{J-1} \widetilde{b}^j(t) + b^*(t), \quad \widetilde{S}(u) = \sum_{j=1}^{J-1} \widetilde{S}^j(u) + S^*(u), \tag{18}$$

where $A^j, b^j, \widetilde{A}^j, \widetilde{b}^j$ contain control variables at grid level j, A^*, b^* are control variables that are applied to the output of the V-cycle at the finest mesh, i.e. $A^j, \widetilde{A}^j \in \mathcal{V}^j, A^* \in \mathcal{V}^1, b^j, \widetilde{b}^j, b^* \in \mathbb{R}$. Operators $S^j, \widetilde{S}^j$ are applied to the intermediate solution on grid level j. Operator S^* is applied to the output of the V-cycle at the finest mesh.

(iii) At grid level j, let L_j be the number of sub-steps to be solved at grid level j in the left and right branches of the V-cycle. We decompose

$$A^j(\mathbf{x},t) = \sum_{l=1}^{L_j} A^{j,l}(\mathbf{x},t), \quad b^j(t) = \sum_{l=1}^{L_j} b^{j,l}(t), \quad S^j(u) = \sum_{l=1}^{L_j} S^{j,l}(u), \tag{19}$$

$$\widetilde{A}^j(\mathbf{x},t) = \sum_{l=1}^{L_j} \widetilde{A}^{j,l}(\mathbf{x},t), \quad \widetilde{b}^j(t) = \sum_{l=1}^{L_j} \widetilde{b}^{j,l}(t), \quad \widetilde{S}^j(u) = \sum_{l=1}^{L_j} \widetilde{S}^{j,l}(u). \tag{20}$$

In our splitting scheme, we will use a sequential splitting techniques for the operators given above both for the left and right branch, where $A^{j,l}, b^{j,l}$, and $S^{j,l}$ are the operators at the l-th sequential sub-step of the left branch, $\widetilde{A}^{j,l}, \widetilde{b}^{j,l}$ and $\widetilde{S}^{j,l}$ are the operators at the l-th sequential sub-step of the right branch.

(iv) At grid level j, for each sequential sub-step l of each branch, we decompose

$$A^{j,l}(\mathbf{x},t) = \sum_{k=1}^{c_j} A_k^{j,l}(\mathbf{x},t), \quad b^{j,l}(t) = \sum_{k=1}^{c_j} b_k^{j,l}(t), \quad S^{j,l}(u) = \sum_{k=1}^{c_j} S_k^{j,l}(u), \tag{21}$$

$$\widetilde{A}^{j,l}(\mathbf{x},t) = \sum_{k=1}^{c_j} \widetilde{A}_k^{j,l}(\mathbf{x},t), \quad \widetilde{b}^{j,l}(t) = \sum_{k=1}^{c_j} \widetilde{b}_k^{j,l}(t), \quad \widetilde{S}^{j,l}(u) = \sum_{k=1}^{c_j} \widetilde{S}_k^{j,l}(u). \tag{22}$$

At grid level j, we split these operators into c_j parallel pathways, where $A_k^{j,l}, b_k^{j,l}$ and $S_k^{j,l}$ are used in the k-th parallel splitting pathway in the left branch, $\widetilde{A}_k^{j,l}, \widetilde{b}_k^{j,l}$ and $\widetilde{S}_k^{j,l}$ are used in the k-th parallel splitting pathway in the right branch.

(v) For the left branch, at grid level j, the l-th sequential step and the k-th parallel splitting pathway, we take all c_{j-1} outputs from the previous sequential step as inputs and use components from $A_k^{j,l}$ to convolve with them. We decompose $A_k^{j,l}$ into c_{j-1} kernels:

$$A_k^{j,l}(\mathbf{x},t) = \sum_{s=1}^{c_{j-1}} A_{k,s}^{j,l}(\mathbf{x},t) \quad \text{with} \quad c_{j,l} = \begin{cases} c_{j-1} & \text{if } l = 1, \\ c_j & \text{if } l > 1. \end{cases} \tag{23}$$

Similarly, for the right branch, the previous sub-step has c_{j+1} outputs. We decompose $\widetilde{A}_k^{j,l}$ into c_{j+1} kernels:

$$\widetilde{A}_k^{j,l}(\mathbf{x},t) = \sum_{s=1}^{\widetilde{c}_j} \widetilde{A}_{k,s}^{j,l}(\mathbf{x},t) \quad \text{with} \quad \widetilde{c}_{j,l} = \begin{cases} c_{j+1} & \text{if } l = 1, \\ c_j & \text{if } l > 1. \end{cases} \tag{24}$$

(vi) Similar to Step (v), we take all c_1 outputs from the V-cycle as inputs and use components from A^* to convolve with them. We decompose A^* as

$$A^*(\mathbf{x},t) = \sum_{s=1}^{c_1} A_s^*(\mathbf{x},t), \tag{25}$$

where A_s^* is used to convolve with the s-th output from level 1 of the right branch of the V-cycle.

After the decomposition, the control variables and operations are decomposed as:

$$A(\mathbf{x},t) = \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j}\sum_{s=1}^{c_{j,l}} A_{k,s}^{j,l}(\mathbf{x},t), \qquad \widetilde{A}(\mathbf{x},t) = \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j}\sum_{s=1}^{\widetilde{c}_{j,l}} \widetilde{A}_{k,s}^{j,l}(\mathbf{x},t) + \sum_{s=1}^{c_1} A_s^*(\mathbf{x},t), \tag{26}$$

$$b(t) = \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j} b_k^{j,l}(t), \qquad \widetilde{b}(t) = \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j} \widetilde{b}_k^{j,l}(t) + b^*(t), \tag{27}$$

$$S(u) = \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j} S_k^{j}(u), \qquad \widetilde{S}(u) = \sum_{j=1}^{J-1}\sum_{l=1}^{L_j}\sum_{k=1}^{c_j} \widetilde{S}_k^{j}(u) + S^*(u). \tag{28}$$

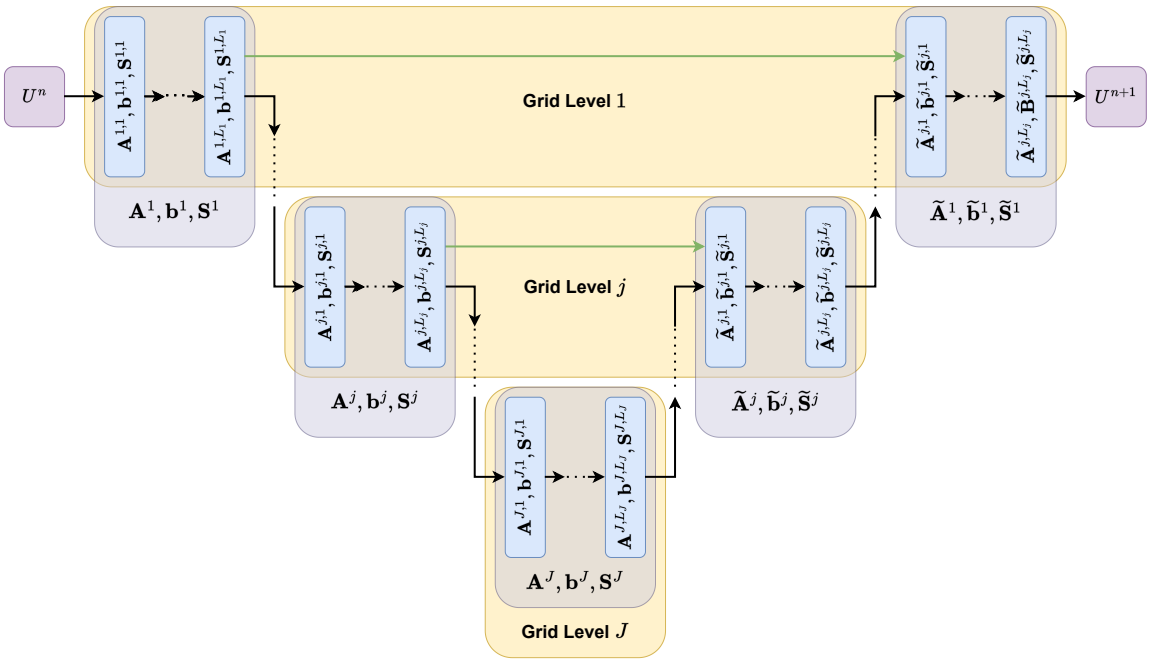

Figure 3: An illustration of Algorithm 2.

The original control problem is transferred minimize the loss (4 for u being the solution of the following equation:

$$\begin{cases} \frac{\partial u}{\partial t} = A * u + \widetilde{A} * u + b + \widetilde{b} + S(u) + \widetilde{S}(u), \ (\mathbf{x}, t) \in \Omega \times [0, T], \\ u(\mathbf{x}, 0) = H(f), \ \mathbf{x} \in \Omega. \end{cases} \tag{29}$$

From (26)-(27), we see that the control variables $\theta_1 = (W(x,t), b(t))$ is decomposed into a large sum and the items in these sums are the new control variables. The number of the control variables are large, but each of them is very small in number of unknowns.

To solve (29), we use the hybrid splitting method introduced in Section 2.2. Divide the time interval $[0, T]$ into N subintervals with time step $\Delta t = T/N$. Denote the computed solution at time $t^n = n\Delta t$ by U^n. The resulting algorithm that updates U^n to U^{n+1} is summarized in Algorithm 2. For simplicity, variable dependencies on $\mathbf{x}$ are omitted. In Algorithm 2, we use $u^{j,l}, v^{j,l}$ to denote intermediate variables in the left and right branches, respectively. The superscript j denotes the grid level at which the computation is conducted, and l denotes the index of the sequential sub-step at grid level j. The architecture of Algorithm 2 is illustrated in Figure 3. In Figure 3, a relaxation step is used for each grid level to pass information from the left branch to the right branch, as indicated by the green arrows. The explanations of all indices for operators and variables of the left branch are summarized in Table 1.

Denote $\theta_2 = \{\theta_2^n\}_{n=1}^N$ with

$$\begin{aligned} \theta_2^n = \big(& \{A_{k,s}^{j,l}(\mathbf{x}, t^n)\}_{j,l,k,s}, \{\widetilde{A}_{k,s}^{j,l}(\mathbf{x}, t^n)\}_{j,l,k,s}, \{A_s^*(\mathbf{x}, t^n)\}_s, \\ & \{b_k^{j,l}(t^n)\}_{j,l,k}, \{\widetilde{b}_k^{j,l}(t^n)\}_{j,l,k}, \widetilde{b}^*(t^n) \big). \end{aligned}$$

We also denote $\mathcal{N}_2$ as the mapping:

$$\mathcal{N}_2 : f \to H(f) \to U^1 \to \cdots \to U^N,$$

Algorithm 2: A hybrid splitting method to solve the control problem (29)

Data: The solution U^n at time t^n.
Result: The computed solution U^{n+1} at time step t^{n+1}.
Set $c_0 = 1, L_0 = 1, v^{1,0} = v_1^{1,0} = U^n$.
for $j = 1, \cdots, J$ **do**
If $j > 1$, set $v^{j,0} = \mathcal{D}(v^{j-1,L_{j-1}})$ and $v_k^{j,0} = \mathcal{D}(v_k^{j-1,L_{j-1}})$ for $k = 1, ..., c_{j-1}$.
for $l = 1, ..., L_j$ **do**
for $k = 1, ..., c_j$ **do**
Compute $v_k^{j,l}$ on $\mathcal{V}^j$ by solving

$$\frac{v_k^{j,l} - v^{j,l-1}}{2^{j-1} c_j \Delta t} - \sum_{s=1}^{c_{j,l}} A_{k,s}^{j,l}(t^n) * v_s^{j,l-1} - b_k^{j,l}(t^n) - S_k^{j,l}(v_k^{j,l}) \ni 0, \tag{30}$$

where $c_{j,l}$ is defined in (23).
end for
Compute $v^{j+1,l}$ as

$$v^{j,l} = \frac{1}{c_j} \sum_{k=1}^{c_j} v_k^{j,l}.$$

end for
end for
Set $u^{J,L_J} = v^{J,L_J}$ and $u_k^{J,L_J} = v_k^{J,L_J}$ for $k = 1, 2, \cdots c_J$.
for $j = J-1, \cdots, 1$ **do**
Set $u^{j,0} = \mathcal{U}(u^{j+1,L_{j+1}})$ and for $k = 1, ..., c_{j+1}$, compute

$$u_k^{j,0} = \frac{1}{2} u_k^{j+1,L_{j+1}} + \frac{1}{2} \mathcal{U}(u^{j+1,L_{j+1}})$$

for $l = 1, ..., L_j$ **do**
for $k = 1, 2, \cdots c_j$ **do**
Compute $u_k^{j,l}$ on $\mathcal{V}^j$ by solving

$$\frac{u_k^{j,l} - u^{j,l-1}}{2^{j-1} \widetilde{c}_j \Delta t} - \sum_{s=1}^{\widetilde{c}_{j,l}} \widetilde{A}_{k,s}^{j}(t^n) * u_s^{j,l-1} - \widetilde{b}_k^{j}(t^n) - \widetilde{S}_k^{j}(u_k^{j}) \ni 0, \tag{31}$$

where $\widetilde{c}_{j,l}$ is defined in (24).
end for
Compute $u^{j,l}$ as

$$u^{j,l} = \frac{1}{c_j} \sum_{k=1}^{c_j} u_k^{j,l}.$$

end for
end for
Compute U^{n+1} by solving

$$\frac{U^{n+1} - u^{1,L_1}}{\Delta t} - \sum_{s=1}^{c_1} A_s^*(t^n) * u_s^{1,L_1} - b^*(t^n) - S^*(U^{n+1}) \ni 0. \tag{32}$$

For $A^j_{k,s}, b^j_k, S^j_k$, $A^{j,l}_{k,s}, b^{j,l}_k, S^{j,l}_k$	j	l	k	s
Index meaning: index of	grid levels	sequential splittings	parallel splittings	output from the previous substep
For u^j_k, v^j_k, $u^{j,l}_k, v^{j,l}_k$	j	l	k	-
Index meaning: index of	grid levels	sequential splittings	parallel splittings	-

Table 1: Explanation of indices for kernels and variables in the left branch of 2.

which maps f to U^N by applying Algorithm 2 N times. Parameters in θ_2 are learned by solving

$$\min_{\theta_2} \sum_{i=1}^{I} \mathcal{L}(\mathcal{N}_2(f_i, \theta_2), g_i). \tag{33}$$

In (33), θ_2 is a space decomposition representation for a discretization of θ_1. The operation procedure $\mathcal{N}_2$ is a numerical scheme solving (1). We can see that problem (33) is a discretization of the optimization problem (4) with some proper decomposition of the control variables.

4 Algorithm details

In Algorithm 2, one needs to solve (30), (31) and (32), which includes components of $S, \widetilde{S}$. We discuss the choices of $S, \widetilde{S}$ and present how to solve (30), (31) and (32) in the following subsections.

4.1 On the choices of $S, \widetilde{S}$

According to (16), $S + \widetilde{S}$ consists of two terms: (i) The first term is $-\ln \frac{u}{1-u}$, which will be used in S^*. This term enforces u to be between 0 and 1 and provides the prediction results. (ii) The second term $-\partial \mathcal{I}_\Sigma(u)$ will be used at every sub-step except for S^*. We will show that this part corresponds to the ReLU activation function in a network. Specifically, we set

$$S^{j,l}_k(u) = \widetilde{S}^{j,l}_k(u) = \partial \mathcal{I}_\Sigma(u), S^*(u) = -\ln \frac{u}{1-u}. \tag{34}$$

4.2 On the solution to (30), (31) and (32)

Observe that (30) and (31) are in the form of

$$\frac{u - u^*}{\gamma \Delta t} - \sum_{s=1}^{c} \widehat{A}_s * u^*_s - \widehat{b} + \partial \mathcal{I}_\Sigma(u) \ni 0, \tag{35}$$

where γ is some constant, c is some integer, $u^* = \frac{1}{c}\sum_{s=1}^{c} u^*_s$ for some functions u^*_s's, $\widehat{A}_s$'s are some convolution kernels, $\widehat{b}$ is some bias function. The solution to (35) can be computed using the following two-sub-step splitting method:

$$\begin{cases} \bar{u} = u^* + \gamma \Delta t \left(\sum_{s=1}^{c} \widehat{A}_s * u^*_s + \widehat{b} \right), \\ \frac{u - \bar{u}}{\gamma \Delta t} + \partial \mathcal{I}_\Sigma(u) \ni 0. \end{cases} \tag{36}$$

In (36), there is no difficulty in solving for $\bar{u}$ in the first sub-step as it is an explicit step. For u in the second sub-step, it is, in fact, a projection. Its closed-form solution is given as

$$u = \max\{\bar{u}, 0\} = \mathrm{ReLU}(\bar{u}), \tag{37}$$

where $\mathrm{ReLU}(u) = \max\{\bar{u}, 0\}$ is the rectified linear unit.

Problem (32) can be written as

$$\frac{u - u^*}{\gamma \Delta t} = \sum_{s=1}^{c} \widehat{A}_s * u_s^* + \widehat{b} - \ln \frac{u}{1-u}. \tag{38}$$

Following the steps for solving (30) and (31) above, we solve (38) as

$$\begin{cases} \bar{u} = u^* + \gamma \Delta t \left(\sum_{s=1}^{c} \widehat{A}_s * u_s^* + \widehat{b} \right), \\ \frac{u - \bar{u}}{\Delta t} = -\ln \frac{u}{1-u}. \end{cases} \tag{39}$$

The first sub-step is an explicit step. We solve the second sub-step approximately by a fixed point iteration.

Initialize $p^0 = \bar{u}$. Given p^k, we update p^{k+1} by solving

$$\frac{p^k - \bar{u}}{\Delta t} = -\ln \frac{p^{k+1}}{1 - p^{k+1}}, \tag{40}$$

for which we have the closed-form solution

$$p^{k+1} = \mathrm{Sig}\left(-\frac{p^k - \bar{u}}{\Delta t} \right), \tag{41}$$

where $\mathrm{Sig}(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. By repeating (41) so that p^{k+1} converges to some function p^*, we set $u = p^*$. In particular, since $p^0 = \bar{u}$, the updating formula (41) always gives $p^1 = 0.5$. If we only consider a two-step fixed point iteration, we get

$$u = \mathrm{Sig}\left(-\frac{0.5 - \bar{u}}{\Delta t} \right) = \mathrm{Sig}\left(\frac{\bar{u} - 0.5}{\Delta t} \right). \tag{42}$$

4.3 Initial condition

Problem (3) requires an initial condition. A simple choice is to set it as some convolution of f:

$$u(\mathbf{x}, 0) = H(f) = \mathrm{Sig}\left(\sum_{k=1}^{3} A_k^0 * f^k \right) \tag{43}$$

for $f = (f^1, f^2, f^3)$. f^1, f^2, and f^3 denote the RGB channels of an image respectively.

4.4 Discretization

To discretize a continuous function u at grid level j, we compute the scaled inner product

$$a_{\boldsymbol{\alpha}}^j = \frac{1}{(2^{j-1}h)^2} \int_\Omega u \phi_{\boldsymbol{\alpha}}^j dxdy$$

for each basis function $\phi_{\boldsymbol{\alpha}}^j$ (defined in (9)) of the space $\mathcal{V}^j$. Note that each ϕ_α^j is an indicator function of a $(2^{j-1}h \times 2^{j-1}h)$ patch indexed by $\boldsymbol{\alpha}$. The inner product $a_{\boldsymbol{\alpha}}^j$ gives the pixel value of u at the $\boldsymbol{\alpha}$-th patch. We take the original image resolution as grid level 1 (the finest grid). Other grid levels and the basis functions can be defined according to the discussion in Section 2.3.

5 Algorithm 2 recovers UNet

In this section, we show that by properly setting the number of grid levels J, parallel splittings c_j's, and sequatial splittings L_j's, Algorithm 2 exactly recovers UNet.

5.1 Algorithm 2 building blocks recover UNet layers

We first show that a building block of Algorithm 2 is equivalent to a layer of UNet. Each layer of UNet is a convolution layer activated by ReLU. Given outputs from the previous layer $\{v_s^*\}_{s=1}^c$, a UNet layer outputs v by the following operations:

$$\begin{cases} \bar{v} = \sum_{s=1}^c W_s * v_s^* + b, \\ v = \mathrm{ReLU}(\bar{v}), \end{cases} \tag{44}$$

where W_s's are convolutional kernels and b is the bias. In Algorithm 2, the building block is (35) and (38), which is solved by (36) and (39). In fact, (44) (or problem (39)) and (36) have the same form.

Specifically, in the first equation of (36), substitute the expression of u^*, and we have

$$\bar{u} = \frac{1}{c}\sum_{s=1}^c u_s^* + \gamma\Delta t\left(\sum_{s=1}^c \widehat{A}_s * u_s^* + \widehat{b}\right) = \sum_{s=1}^c \left(\frac{1}{c}\mathbb{1} + \gamma\Delta t\widehat{A}_s\right) * u_s^* + \gamma\Delta t\widehat{b}, \tag{45}$$

where $\mathbb{1}$ denotes the identity kernel satisfying $\mathbb{1} * g = g$ for any function g. In (44), set

$$W_s = \frac{1}{c}\mathbb{1} + \gamma\Delta t\widehat{A}_s, \quad b = \widehat{b}. \tag{46}$$

We have $\bar{v} = \bar{u}$, and $v = u$. Essentially, Algorithm 2 and UNet have the same building block.

5.2 Algorithm 2 structure recovers UNet architecture

UNet architecture consists of four components: encoder, decoder, bottleneck and skip-connections, each of which has a corresponding component in the structure of Algorithm 2:

(i) **Encoder**: Encoder in UNet corresponds to the left branch of the V-cycle in Algorithm 2. The number of data resolution levels corresponds to the number of grid levels J. At each data resolution, the number of layers and the width of each layer correspond to the number of sequential splittings L_j and parallel splittings c_j at the corresponding grid level.

(ii) **Decoder**: Decoder in UNet corresponds to the right branch of the V-cycle in Algorithm 2. The number of data resolution levels corresponds to the number of grid levels J. At each data resolution, the number of layers and the width of each layer correspond to the number of sequential splittings L_j and parallel splittings c_j at the corresponding grid level.

(iii) **Bottleneck**: Bottleneck in UNet corresponds to the computations at the coarsest grid level (grid level J) in Algorithm 2. The number of layers and layer width in bottleneck correspond to the number of sequential splittings L_J and parallel splitting c_J at grid level J.

(iv) **Skip-layer connection**: Skip-layer connections in UNet correspond to the relaxation steps in Algorithm 2.

UNet has 5 data resolution levels. For each resolution level, there are two layers in the encoder, decoder and bottleneck. From the finest resolution to the coarsest resolution, the layers has width $64, 128, 256, 512, 1024$. Thus, set $J = 5$, $L_1 = L_2 = L_3 = L_4 = L_5 = 2$, $[c_1, c_2, c_3, c_4, c_5] = [64, 128, 256, 512, 1024]$, downsampling operator $\mathcal{D}$ as the max-pooling operator, and upsampling operator $\mathcal{U}$ as the transpose convolution, Algorithm 2 exactly recovers UNet. As a consequence, one can explain UNet as a one-step operator-splitting algorithm solving a control problem (1).

6 Conclusion

In this paper, we consider the control problem (1) and propose an operator-splitting method to solve it. The ingredients of our algorithm include the multigrid method and the hybrid operator splitting method. We show that the resulting algorithm has the same building block and architecture as UNet. Our result demonstrates that UNet is a one-step operator-splitting algorithm that solves some control problems; thus, it gives a mathematical explanation of the UNet architecture from an algorithmic perspective.

References

[1] S. Anwar and N. Barnes. Real image denoising with feature attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3155–3164, 2019.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] G. Bao, D. Wang, and B. Zou. Wanco: Weak adversarial networks for constrained optimization problems. *arXiv preprint arXiv:2407.03647*, 2024.

[4] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[5] M. Benning, E. Celledoni, M. Ehrhardt, B. Owren, and C. Schhönlieb. Deep learning as optimal control problems: models and numerical methods. *Journal of Computational Dynamics*, 2019.

[6] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta numerica*, 3:61–143, 1994.

[7] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[9] 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.

[10] Z. Cui, T. Y. Pan, G. Yang, J. Zhao, and W. Wei. A trainable variational chan-vese network based on algorithm unfolding for image segmentation. *Mathematical Foundations of Computing*, pages 0–0, 2024.

[11] 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(2):1190–1230, 2019.

[12] 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(3):1–28, 2022.

[13] W. E. A Proposal on Machine Learning via Dynamical Systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[14] W. E, C. Ma, and L. Wu. Machine learning from a continuous viewpoint, I. *Science China Mathematics*, 63(11):2233–2266, 2020.

[15] C.-M. Fan, T.-J. Liu, and K.-H. Liu. Sunet: Swin transformer unet for image denoising. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2333–2337. IEEE, 2022.

[16] R. Glowinski, S. Leung, and J. Qian. A penalization-regularization-operator splitting method for eikonal based traveltime tomography. *SIAM Journal on Imaging Sciences*, 8(2):1263–1292, 2015.

[17] 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(1):1–47, 2019.

[18] R. Glowinski, S. Luo, and X.-C. Tai. Fast operator-splitting algorithms for variational imaging models: Some recent developments. In *Handbook of Numerical Analysis*, volume 20, pages 191–232. Elsevier, 2019.

[19] R. Glowinski, S. J. Osher, and W. Yin. *Splitting methods in communication, imaging, science, and engineering*. Springer, 2017.

[20] R. Glowinski, T.-W. Pan, and X.-C. Tai. Some facts about operator-splitting and alternating direction methods. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 19–94. Springer, 2016.

[21] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[22] J. Gurrola-Ramos, O. Dalmau, and T. E. Alarcón. A residual dense u-net neural network for image denoising. *IEEE Access*, 9:31742–31754, 2021.

[23] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):1–23, 2018.

[24] E. Haber, L. Ruthotto, E. Holtham, and S. H. Jun. Learning across scales - Multiscale methods for convolution neural networks. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3142–3148, 2018.

[25] J. He and J. Xu. MgNet: A unified framework of multigrid and convolutional neural network. *Science China Mathematics*, 62(7):1331–1354, 2019.

[26] Y. Lan, Z. Li, J. Sun, and Y. Xiang. Dosnet as a non-black-box pde solver: When deep learning meets operator splitting. *arXiv preprint arXiv:2212.05571*, 2022.

[27] H. Liu, J. Liu, R. Chan, and X.-C. Tai. Double-well net for image segmentation. *arXiv preprint arXiv:2401.00456*, 2023.

[28] H. Liu, X.-C. Tai, and R. Chan. Connections between operator-splitting methods and deep neural networks with applications in image segmentation. *Ann. Appl. Math*, 39(4):406–428, 2023.

[29] H. Liu, X.-C. Tai, and R. Glowinski. An operator-splitting method for the gaussian curvature regularization model with applications to surface smoothing and imaging. *SIAM Journal on Scientific Computing*, 44(2):A935–A963, 2022.

[30] 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(2):717–748, 2021.

[31] 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*, pages 1–21, 2022.

[32] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[33] 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*, pages 234–241. Springer, 2015.

[34] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62:352–364, 2020.

[35] X.-C. Tai. Rate of convergence for some constraint decomposition methods for nonlinear variational inequalities. *Numerische Mathematik*, 93(4):755–786, 2003.

[36] X.-C. Tai, H. Liu, and R. Chan. Pottsmgnet: A mathematical explanation of encoder-decoder based neural networks. *SIAM Journal on Imaging Sciences*, 17(1):540–594, 2024.

[37] X.-C. Tai and J. Xu. Subspace correction methods for convex optimization problems. *Eleventh International Conference on Domain Decomposition Methods (London, 1998)*, pages 130–139, 1998.

[38] X.-C. Tai and J. Xu. Global and uniform convergence of subspace correction methods for some convex optimization problems. *Mathematics of Computation*, 71(237):105–124, 2002.

[39] L. Tan, L. Li, W.-Q. Liu, S.-J. An, and K. Munyard. Unsupervised learning of multi-task deep variational model. *Journal of Visual Communication and Image Representation*, 87:103588, 2022.

[40] T. Wu, C. Huang, S. Jia, W. Li, R. Chan, T. Zeng, and S. K. Zhou. Medical image reconstruction with multi-level deep learning denoiser and tight frame regularization. *Applied Mathematics and Computation*, 477:128795, 2024.

[41] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM review*, 34(4):581–613, 1992.

[42] M. Xu, Q. Ma, H. Zhang, D. Kong, and T. Zeng. MEF-UNet: An end-to-end ultrasound image segmentation algorithm based on multi-scale feature extraction and fusion. *Computerized Medical Imaging and Graphics*, 114:102370, 2024.

[43] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[44] Y. Zang, G. Bao, X. Ye, and H. Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

[45] K. Zhang, L. Li, H. Liu, J. Yuan, and X.-C. Tai. Deep convolutional neural networks meet variational shape compactness priors for image segmentation. *arXiv preprint arXiv:2406.19400*, 2024.

[46] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5743–5752. IEEE, 2017.

[47] D.-X. Zhou. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, 48(2):787–794, 2020.

[48] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, 2019.