JST: Smart Systems and Devices
Volume 34, Issue 3, September 2024, 017-026

HyperPolypDEQ: A Lightning Transformer-Based Deep Equilibrium Model for Colon Polyp Segmentation

Nguyen Minh Chau, Le Truong Giang, Dinh Viet Sang*
Hanoi University of Science and Technology, Ha Noi, Vietnam
**Corresponding author email: sangdv@soict.hust.edu.vn*

Abstract

Deep Equilibrium Models (DEQs) have recently emerged as a promising approach to building implicit deep learning models that can achieve on-par accuracy with traditional explicit models but with considerably smaller sizes. However, the significant downside of DEQs is their slow inference speed, primarily due to the time cost of the fixed-point solver. This paper proposes to overcome this issue by applying HyperSolver, a novel technique that replaces traditional fixed-point solvers with a lightweight neural network. This is an extension of our previous work on PolypDeq concerning DEQs for medical image segmentation as an attempt to accelerate our existing implicit models. Experimental results show that our new models using Hyper-Solver can achieve similar results to existing DEQ models on several benchmark medical image datasets while having a significant speedup in inference time (about 9 times). To the best of our knowledge, this is the first attempt to accelerate DEQs for medical image segmentation using HyperSolver, representing a significant step towards making implicit deep learning models more practical for real-world applications.

Keywords: Semantic segmentation, polyp segmentation, implicit deep learning, deep equilibrium models.

1. Introduction

Over the past few years, there has been a significant rise in the incidence of colorectal cancer. Detecting and diagnosing polyps with high precision is crucial for effective treatment. Hence, developing computer-assisted systems capable of performing these tasks can provide valuable support to medical professionals and doctors. Artificial intelligence with the capability of polyp segmentation [1, 2] has shown great potential for this job.

In computer vision, especially medical image analysis, semantic segmentation has become a popular area of research lately. With the advancements in deep learning methods, it is possible to treat semantic segmentation as a classification problem, where each pixel is classified into specific categories. Most of the works in semantic segmentation [3, 4] follow the encoder-decoder architecture, which consists of two primary parts: an encoder that processes the input image and generates feature maps, which contain essential information about the image in a tensor format, and a decoder that uses the feature maps generated by the encoder to produce a segmentation mask. Earlier studies [3, 5-7] utilized Fully Convolutional Networks (FCNs) [8] for both the encoders and decoders components. Recently, many new architectures (such as SegFormer [4]) have emerged that employ the Vision Transformer (ViT) [9] to achieve impressive results in semantic segmentation tasks. These new architectures using ViT have demonstrated superior performance to traditional FCN architectures.

However, in both of these approaches, it is necessary to create explicit networks, or architectures, that require a large number of parameters in order to achieve good results. To address this issue, [10] introduced a new class of implicit deep learning models for computer vision tasks called Multi-scale Deep Equilibrium models (MDEQ). These models have been shown to have better memory efficiency and representative power than the explicit models mentioned above while achieving comparable performance. In our previous work [11], we proposed a novel deep implicit model based on SegFormer [4], a powerful network architecture for semantic segmentation based on transformers. Experimental results on medical image datasets showed that our method outperformed MDEQ in terms of accuracy. However, PolypDEQ is still impeded by the relatively slow inference speed commonly associated with implicit deep learning models, and it requires a method to accelerate PolypDEQ to make it more applicable to the medical image analysis field, particularly polyp segmentation.

In this paper, we adapt HyperSolver [12], a novel method to accelerate DEQs, on top of our previous model, PolypDEQ [11], to create a novel method, called HyperPolypDEQ. The primary concept

ISSN: 2734-9373
https://doi.org/10.51316/jst.176.ssad.2024.34.3.3
Received: Feb 23, 2024; revised: Apr 19, 2024;
accepted: Apr 25, 2024.

underlying the HyperSolver method is to approximate the conventional iterative Newton solvers employed in DEQs through the use of a lightweight neural network, which leads to a substantial reduction in inference time without sacrificing the precision of the model. We experiment with our proposed model on several medical image benchmark datasets. According to the experimental results, HyperPolypDEQ can significantly improve inference speed, with a 9-fold acceleration, while maintaining on-par accuracy.

In summary, our main contributions are:

- We validate our previous work on implicit deep learning models on a new medical image benchmark dataset;
- We adapt HyperSolver to our model to accelerate its inference speed;
- We demonstrate that our new model, HyperPolypDEQ, achieves significant speedup in inference time while maintaining high accuracy, making it a promising approach for medical image segmentation tasks.

The rest of this paper is organized as follows: Section 2 briefly reviews some prior studies regarding traditional explicit deep learning and novel implicit deep learning methods. Then, we revise the backgrounds of deep equilibrium models, iterative solvers, and HyperSolver in Section 3. Our proposed method is described in Section 4. Section 5 outlines our experiment settings. The results and discussions are presented in Section 6. Finally, we conclude this study in Section 7.

2. Related Work

2.1. Explicit Deep Learning Methods

Colonoscopy analysis is required for clinicians to detect the location and severity of polyps in colorectal cancer treatment. However, polyps are various in shape, size, and location, causing difficulties in analyzing colonoscopy by human eyes.

Traditional deep learning-based semantic segmentation methods mainly follow the conventional explicit deep learning approach, meaning they focus on designing explicit computational graphs, or so-called "architecture", for forward and backward propagation. Based upon the success of UNet [3] for medical image segmentation, UNet++ [5], ResUNet++ [6], among other methods, following the same Fully Convolutional Network family, yielded promising results. Recently, along with the advent of Transformers and self-attention mechanisms for computer vision tasks, many studies have adapted Transformers to achieve state-of-the-art results in medical image segmentation tasks. NeoUnet [13] leveraged the lightweight HardNet backbone network, combined with a self-attention mechanism for polyp segmentation and neoplasm detection. TransFuse [14] combined both Convolutional Neural Networks (CNNs) and Transformers to create a lightweight, efficient network with a parallel structure. ColonFormer [15] achieved state-of-the-art results in the polyp segmentation task using a pure Transformers-Based architecture. However, while these models have achieved impressive results, they require careful engineering of the model architecture and tend to have a large number of parameters, making them expensive to deploy in practice.

2.2. Implicit Deep Learning

Unlike the aforementioned explicit deep learning methods, implicit deep learning takes another path with a novel idea. Instead of defining a computational graph or an explicit architecture, it provides a criterion the models must follow (e.g., the network output must satisfy an equation). Implicit models operate forward and backward propagation as root-finding problems (also referred to as finding equilibrium points), using Newton's and Quasi-Newton algorithms [16], such as Broyden [17] and Anderson [18], as equilibrium solvers. The main benefit of implicit deep learning compared to its explicit counterpart is memory efficiency: implicit deep learning models naturally do not require as many parameters as explicit models, leading to a considerable reduction in memory cost while maintaining similar accuracy. An example could be Neural ODEs (NODEs) [19], which use just one residual block in a recursive fashion and equilibrium solvers and is equivalent to an infinite-depth ResNet [20]. DEQs [21], which is another instance of implicit models, used Broyden and Anderson solvers to find the equilibrium points of a model for sequential tasks in Natural Language Processing. Soon after, Multi-Scale Deep Equilibrium Models (MDEQs) [10] were proposed and became the first implicit model for computer vision tasks, including image classification and semantic segmentation, and showed comparable results with state-of-the-art explicit models.

However, one drawback of these implicit deep learning models is their extremely slow training and inference speed compared to conventional explicit models, mainly due to the iterative equilibrium solvers. These solvers have to represent the internal state of the models as tensors and store them in memory during the fixed-point solving process, hence hindering the models' performance speed. In this work, we attempt to overcome this issue by using HyperSolver [12].

3. Background: Equilibrium Models and Solvers

3.1. Deep Equilibrium Models

Consider a neural block f_θ, such as a self-attention or residual block, and an input image x. A DEQ [21] solves for an equilibrium representation that is equivalent to the feature obtained from input x

through an infinite number of successive blocks f_θ. DEQ achieves this by solving an equation to find the fixed point z^*:

$$g_\theta(z^*, x) := f_\theta(z^*, x) - z^* \quad (1)$$

Here, one can apply Newton's or quasi-Newton methods [16], such as Broyden [17] and Anderson [18], to estimate the fixed point solution z^*. In the backward propagation, one can implicitly differentiate through the fixed point by the formula:

$$\frac{\partial l}{\partial \theta} = \frac{\partial l}{\partial z^*}(I - \frac{\partial f_\theta(z^*, x)}{\partial z^*})^{-1} \frac{\partial f_\theta(z^*, x)}{\partial \theta}$$
$$= -\frac{\partial l}{\partial z^*} J_g(z^*)^{-1} \frac{\partial f_\theta(z^*, x)}{\partial \theta} \quad (2)$$

where l is the loss, $J_g(z^*)$ is the Jacobian of $g_\theta = f_\theta(z^*) - z^*$ w.r.t z^*, which can be estimated via solving yet another equation, see [21] for more details. Note that this backward pass can be calculated without any knowledge of how z^* was computed. Since both the forward and backward passes of DEQ can be formulated as root-finding problems, the algorithms used for finding the solution to these equations play a decisive role in DEQ's performance.

3.2. Anderson Equilibrium Solver

Anderson [18] is one of the iterative equilibrium point solvers that can be used for DEQs. For a DEQ model f_θ , starting with one initial point $z^{[0]}$(e.g., the initial internal state of the model), the Anderson solver defines a maximum of $m + 1$ consecutive solver steps. At iteration k, it calculates the set of $m + 1$ past residuals (e.g., the solver error) $G^{[k]} = [g_\theta(z^{[k-m]}), g_\theta(z^{[k-m+1]}), \ldots, g_\theta(z^{[k]})]$ with $g_\theta(z^{[i]}) = f_\theta(z^{[i]}) - z^{[i]}$; then it computes a set of weights for the past $m + 1$ steps $\alpha^{[k]} = [\alpha_0^{[k]}, \alpha_1^{[k]}, \ldots, \alpha_m^{[k]}]$ in a greedy manner to minimize the following error:

$$\alpha^{[k]} = argmin_\alpha ||\alpha G^{[k]}||_2 \quad s.t\, \Sigma\alpha^{[k]} = 1 \quad (3)$$

The next internal state $z^{[k+1]}$ is then computed as the linear combination of the past $m + 1$ states weighted by $\alpha^{[k]}$, plus the linear combination of the past $m + 1$ residuals weighted by the same $\alpha^{[k]}$ and scaled by another smoothing factor β.

$$z^{[k+1]} = \beta \sum_{i=0}^{m} \alpha_i^{[k]} f_\theta\left(z^{[k-m+i]}\right) +$$
$$(1-\beta) \sum_{i=0}^{m} \alpha_i^{[k]} z^{[k-m+i]}$$
$$= \sum_{i=0}^{m} \alpha_i^{[k]} z^{[k-m+i]} + \beta \sum_{i=0}^{m} \alpha_i^{[k]} g_\theta(z^{[k-m+i]}) \quad (4)$$

These steps are repeated until the stopping conditions hold, such as $g_\theta(z^{[i]})$ at some iteration i is negligible or approaching the maximum iterations are met. Note that Anderson solver requires us to keep track of a set of past $m + 1$ (typically 5) internal states, as well as the residuals (same size as the states). It means that, in practice, for a DEQ model whose internal states z are tensors of millions of elements, the memory cost needed during the solving process is also scaled linearly by the size of those tensors and the choice of m. Another downside to this solver and other iterative solvers is that they typically initialize the first state $z^{[0]}$ to be zero or randomly sampled from a normal distribution, plus the greedy strategy to find the weights α during the solving. These factors, in practice, lead to a need for a large number of solving iterations to reach the fixed point solution, which is very time-consuming due to the lack of parallelism nature of these iterative algorithms. It is argued that with better initialization and a more accurate way of weight computing, we can achieve a much faster solving process without losing much quality in the final equilibrium point solution.

3.3. Hyper Solver.

HyperSolver [12] was proposed to address the issue of the slow speed and ineffective weight computing of traditional iterative solvers like Anderson. The idea is to make the parameters α and β of the Anderson solver learnable via a compact neural network. The core ideas of HyperSolver are stated as follows.

Given a DEQ layer f_θ with input x, a fixed-point solution z^*, HyperSolver uses a tiny neural network, which consists of two sub-modules: an initializer and HyperAnderson iterations, parameterized by $w = \{\phi, \xi\}$, respectively. This tiny neural network learns the initialization and the parameters α and β of Anderson's solving process.

According to the authors, the initializer is designed to "guess" the initial values of the solving process, which is modeled as a tiny network $h_\phi: z^{[0]} = h_\phi(x)$ consisting of just one convolutional layer with ReLU activation. The initializer takes in the injection of the input x of DEQs (e.g., a feature map(s) of the input image) and produces an initial guess for the equilibrium point.

The main module of the HyperSolver, HyperAnderson iterations, is yet modeled by another lightweight network to simulate the traditional Anderson iterative equilibrium point-solving process. It is a module that takes in a small set of $m+1$ consecutive residuals:

$$G^{[k]} = [g_\theta^{[k-m]}, g_\theta^{[k-m+1]}, \ldots, g_\theta^{[k]}],$$

where $g_\theta^{[i]} = f_\theta(z^{[i]}) - z^{[i]}$ and $G^{[0]} = [g_\theta^{[0]}, 0, 0, \ldots, 0]$.

The HyperAnderson iteration module first compresses these residuals tensors via a pooling layer,

then applies a temporal convolution, which consists of 1D convolutional, group normalization, and ReLU activation, to "mix" the information of these residuals together. This module then uses a linear layer to predict a set of $\alpha^{[k]}$and the smoothing factor $\beta^{[k]}$. Finally, it applies the same update rule in (4) to get the next internal state $z^{[k+1]}$ and repeat until the fixed point z^* is found.

Notably, the HyperSolver itself can be considered an iterative process. However, with a much smaller total number of iterations, typically only tens instead of hundreds of iterations, to achieve the same precision as its traditional Anderson counterpart. This is due to a better initial point produced by the initializer instead of using zero or Gaussian sampling initialization. The improvement also comes from a more effective neural-network-based weight (α and β) finding instead of the greedy optimization algorithm in the traditional Anderson solver. The overall pipeline of HyperSolver is summarized as follows:

- The input x is fed into the Initializer $h_\phi(x)$ to get the initial guess $z^{[0]}$ of the internal state;
- It is then input into the HyperAnderson iterations over M total steps (typically 12 as mentioned in [12]) to iteratively produce $z^{[1]}, z^{[2]}, \ldots, z^{[K]}$.

Bai *et al.* [12] showed that with the aid of HyperSolver, which only costs an extra 1-3% of the DEQ model size and 0.9-1.1% training time, these DEQs models can enjoy up to 2 times speedup in inference without any degradation in accuracy loss. In this work, we attempt to incorporate HyperSolver with some modifications to fit our existing PolypDEQ model.

4. Methodology

4.1. PolypDEQ

In our prior work [11], we presented an original neural network design called PolypDEQ. This architecture is founded on the principles of implicit deep learning, specifically the use of MDEQs, and SegFormer. A detailed illustration of this comprehensive architecture is delineated in Fig. 1.

Adhering to the implicit deep learning methodology, our core design was the iterative transformation f_θ. Initially, the input image was fed into the first ViT encoder block to get a feature map. Then, we set z, our internal state, as a tensor filled with zeros, and possessing the same resolution as the first input representation, following the completion of the first ViT encoder block.

Unlike MDEQs, which determine the equilibrium state for multiple spatial resolutions, we found only one equilibrium state, z^*, for z at a singular resolution scale. We merged z and the first feature map of the ViT encoder to generate another feature map possessing the same resolution. This process emulates the outcome of "input representation injection", as presented in [10].

Our previous work [11] discussed two approaches to combine two tensors: direct element-wise addition and utilizing a basic residual block. The output of this operation was fed into three additional ViT encoder blocks, resulting in four distinct feature maps, which include the original feature map with input injection.

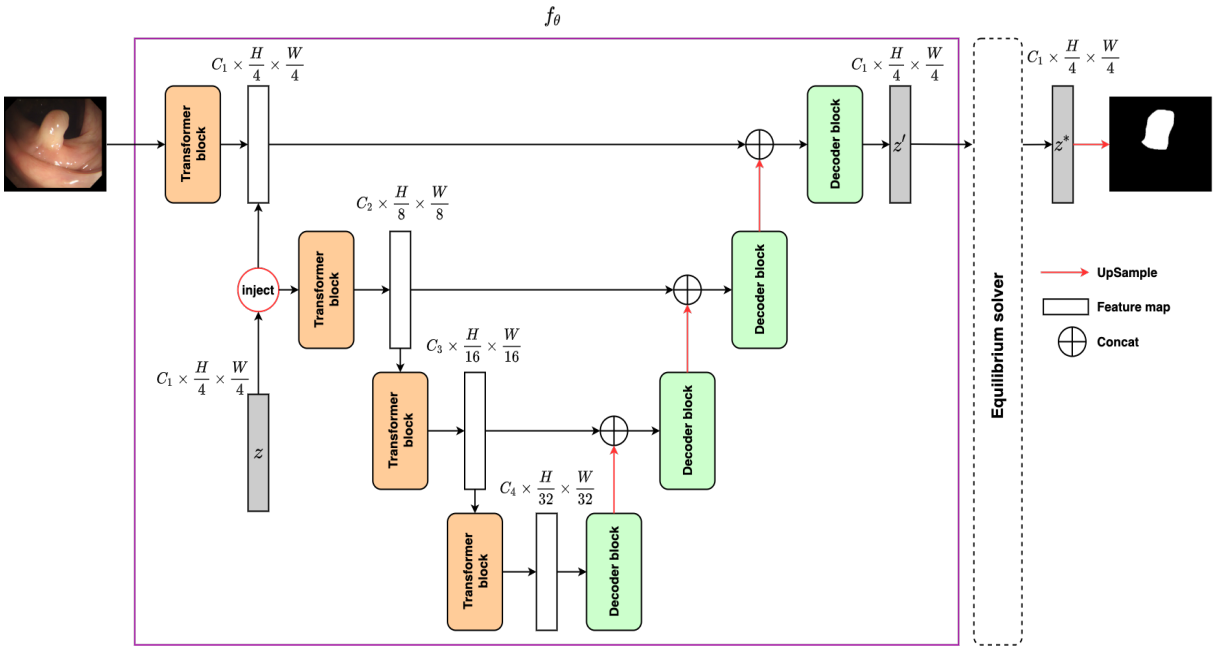

Fig. 1. Overview of our previous PolypDEQ.

PolypDEQ employed a straightforward hierarchical decoder module resembling the conventional UNet decoder module, which hierarchically decodes feature maps from coarse to fine. At every level of resolution, the feature map is interpolated to align with the resolution of the next larger feature map. Then, they are concatenated and go through a convolution layer. The resultant feature maps are fed into the encoder. The encoder produces four feature maps, which are given as input to the decoder. Instead of applying a convolution layer or a linear layer to predict the mask, an implicit deep learning approach is used. Broyden's equilibrium solver is utilized to find the equilibrium point z^*, which is equivalent to iterating f_θ infinitely, and then predictions are made on it. This model can be viewed as an infinite number of weight-tied U-Net one after another. We experimented with two approaches, explicit models that make predictions right after the first iteration of f_θ , and implicit models that use the Broyden equilibrium solver to find the equilibrium state z^* and make predictions on it.

4.2 HyperPolypDEQ

This section describes our modifications to incorporate HyperSolver into our existing work, PolypDEQ. We keep the original design of HyperSolver except for two notable changes: the use of single-scale internal state representation and our training strategy for HyperSolver.

Single-scale internal state representation: We integrate the HyperSolver used for Multi-scale Deep Equilibrium Models (MDEQs) into our previous PolypDEQ. The original HyperSolver in [12] was designed specifically for MDEQs, which use multi-scale internal states: MDEQs modeled their internal states as four tensors at different resolution scales (e.g., 1/4, 1/8, 1/16, and 1/32), and therefore their solvers have to solve for equilibrium state at all four scales. The initializer in [12] hence also produces multi-scale output states, and their HyperAnderson Iteration has to store these states in $m + 1$ consecutive steps, as well as their corresponding residuals. This results in a notable memory cost.

PolypDEQ, however, uses only one internal state at a single resolution of 1/4. Hence, the memory cost is greatly reduced, resulting in an even faster speed than the original MDEQ. Therefore, we design our initializer to produce only one internal state of the same resolution. Our HyperAnderson iterations also require less memory for storing the consecutive internal states and residuals. The core modules, such as convolutional layers, pooling layers, as well as the pipeline, are similar to those in the original HyperSolver Design. The overall design of our HyperSolver is summarized in Fig. 2 and Fig. 3.

HyperSolver training strategy: For the training strategy, in [12], Bai *et al*. proposed to train the HyperSolver jointly with the DEQ network in an alternative manner as follows:

1. First, the DEQ model is trained from scratch for a small number of steps;
2. Then, a snapshot of the current DEQ model is taken, and a HyperSolver is trained from scratch on top of it;
3. After that, the Anderson solver in the DEQ model is replaced with the current HyperSolver and the DEQ model is trained for the next *M* steps;
4. Finally, the DEQ model is frozen again, and the HyperSolver is fine-tuned on top of it. Steps 3 and 4 are repeated until convergence.

However, we argue that this training strategy is unstable: In the first epochs, the fixed point z^* provided by DEQs with traditional solvers is an unreliable ground truth to train the HyperSolver. Hence, we proposed to train the DEQ model separately with a Broyden or Anderson solver with a large number of solver iterations to reach convergence with high precision first. Then, we train a HyperSolver from scratch on top of it and replace the original iterative solver with the trained HyperSolver during inference.

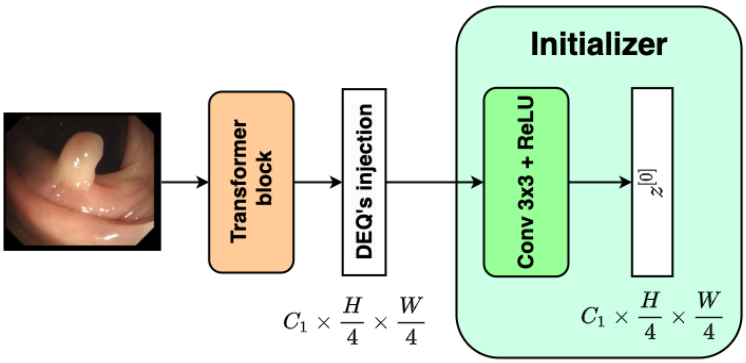

Fig. 2. Our modified initializer.

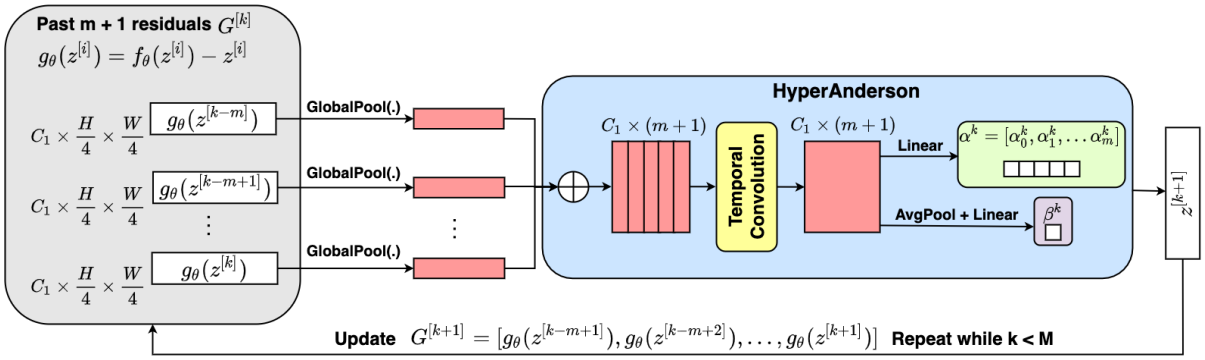

Fig. 3. Our modified HyperAnderson Iterations.

We found that this training strategy can greatly stabilize the convergence of the HyperSolver model. This might be due to the fact that HyperSolver is trained by the ground truth z^* obtained by solving the DEQ model's equilibrium points with Broyden or Anderson for hundreds of iterations, so an already converged DEQ model with high precision will provide more reliable ground truth, thus, stabilized the training of the HyperSolver.

5. Experiments

5.1. Datasets

We conducted experiments on six benchmark datasets for polyp segmentation, which are detailed in Table 1. The Kvasir dataset [22] was gathered from endoscopic equipment at Vestre Viken Health Trust (VV) in Norway, and the images were meticulously annotated and reviewed by experienced gastroenterologists from VV and the Cancer Registry of Norway. The dataset contains 1000 images with various resolutions ranging from 720 × 576 to 1920 × 1072 pixels. The CVC-ClinicDB dataset [23] is a collection of frames taken from colonoscopy videos, consisting of 612 images with a resolution of 384 × 288 pixels extracted from 31 colonoscopy sequences. It was used in the MICCAI 2015 Sub-Challenge on Automatic Polyp Detection Challenge in Colonoscopy Videos training stages. The CVC-ColonDB dataset [24] was provided by the Machine Vision Group (MVG) and includes 380 images with a resolution of 574 × 500 pixels taken from 15 short colonoscopy videos.

The EndoScene dataset [26] is the test set of a larger dataset called Endoscene, which contains 60 images from 44 video sequences acquired from 36 patients. The ETIS-Larib dataset [26] has 196 high-resolution (1226 × 996) colonoscopy images. The NeoPolyp-Small [13] is a public dataset available in a Kaggle competition with 1200 images. The training set comprises 1000 images, and the remaining 200 images form the test set.

Table 1. The properties of benchmark datasets.

Dataset	# Training images	# Test images	Resolution
Kvarsir-SEG [22]	900	100	Various
CVC-ClinicDB [23]	550	62	384 x 288
CVC-ColonDB [24]	0	380	574 x 500
ETIS-Larib PolypDB [25]	0	196	1225 x 966
EndoScene [26]	0	60	574 x 500
NeoPolyp-Small [13]	1000	200	Various

Note that the ground truth segmentation mask for the first five datasets contains two classes: polyp and background. Meanwhile, the NeoPolyp Small dataset has three classes: neoplastic polyp, non-neoplastic polyp, and background.

5.2. Implementation Details

As mentioned before, our training strategy for HyperPolypDEQ consists of two phases: First, we trained a PolypDEQ with Broyden solver till convergence with a large number of solver iterations of 100. Then, we trained a HyperSolver on top of the trained PolypDEQ for better stability. The training process of PolypDEQ comprised two distinct phases: explicit and implicit.

During the explicit phase, the model was trained without utilizing any equilibrium solvers, where the iterate function f_θ was only iterated once. This phase involved training the model as an explicit model, utilizing the AdamW optimizer with an initial learning rate of 10^{-3}, and a cosine annealing learning rate scheduler. The learning rate was reduced to 10^{-6} by the end of the training process. The explicit models were trained for 100 epochs with a batch size of 16. In addition, a multi-scale training strategy was employed to boost the model's generalization capability, whereby each training image was resized to 0.75, 1, and 1.25 times the original scale before being fed into the model. Following the explicit phase, the explicit version of the model was obtained.

Subsequently, for the implicit phase, the weight of the explicit model was copied, given that the structure of both explicit and implicit versions of the model was identical. The only difference between the two versions was the existence of an equilibrium solver. The model was then trained explicitly for five epochs before applying the equilibrium solver and continuing training as an implicit model for the remaining 95 epochs. The same training configurations as the first phase were utilized, except for the learning rate scheduler. For this phase, the learning rate was initially set to 10^{-3} and linearly decreased to 10^{-6} over the first five training epochs. The learning rate was then reset to 10^{-4} and decreased to 10^{-6} following a cosine annealing scheduler for the rest of the training. Following the implicit phase, the implicit model was obtained.

We set the maximum number of consecutive solver steps to be stored $m = 5$, and the total number of solver iterations $M = 12$ (much less than 100 for traditional Anderson). The training of HyperSolver after PolypDEQ convergence was rather simple. We deployed Gradient Descent with Adam optimizer, a batch size of 8. The learning rate was set to 10^{-3} and gradually decreased to 10^{-6} following a cosine annealing scheduler. We trained our HyperSolver for a total of 8 epochs using the three loss functions mentioned in the original paper [12]. This training specification was applied to all the datasets in our experiments.

Our approach to assessing the performance of models in image segmentation tasks involved the utilization of two widely recognized evaluation metrics, namely the mean *Dice* score and the mean *IoU*. In particular, these metrics are calculated as follows:

$$IoU = \frac{TP}{TP + FP + FN} = \frac{\sum_i^N TP_i}{\sum_i^N TP_i + FP_i + FN_i} \quad (5)$$

$$Dice = \frac{2TP}{2TP + FP + FN} = \frac{2\sum_i^N TP_i}{\sum_i^N 2TP_i + FP_i + FN_i} \quad (6)$$

These metrics were employed to measure the similarity and overlap between the predicted segmentation masks and the ground truth masks. We calculated *Dice*, and *IoU* for each image and subsequently derived their respective mean values as the average performance measures across all the images.

6. Results and Discussion

Table 2 and Table 3 show the values of the performance metrics on the benchmark datasets of our models, including PolypDEQ and HyperPolypDEQ, compared with MDEQ. We compare both implicit and explicit versions of each model, except for HyperPolypDEQ, with only the implicit version using HyperSolver. Meanwhile, Table 4 demonstrates the inference speed of these models. For the datasets in Table 2, the metrics were simply computed for the class Polyp; meanwhile, for the NeoPolyp-Small dataset, we recorded the metrics for two classes of Polyps: neoplastic and non-neoplastic, we also further performed the metric with the two classes being considered as one single class of polyp, as shown in Table 3. The time measurement recorded in Table 4 is the average inference time of the models over 100 images taken randomly from our benchmark datasets.

The experimental results show that the transformer architecture used in designing our models, PolypDEQ and HyperPolypDEQ, is more effective than the architectures used in the three baseline models, including implicit and explicit MDEQ, as well as Segformer-B0.

The implicit PolypDEQ models perform well on all datasets, surpassing the baseline MDEQ models. Additionally, the HyperPolypDEQ yields superior performance compared to the baselines on all datasets. Note that the addition of HyperSolver results in a slight accuracy loss for HyperPolypDEQ on some datasets, CVC-Clinic, CVC-Colon, and ETIS. However, it still outperforms the three baselines on those datasets. However, on the Kvasir and EndoScene datasets, HyperPolypDEQ surprisingly achieves more accuracy than PolypDEQ.

Table 2. Quantitative results on five benchmark datasets.

Method	Mode	Kvasir		ClinicDB		ColonDB		EndoScene		ETIS-Larib	
		mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
MDEQ [10]	expl	84.6	77.3	83.6	76.9	58.5	48.7	78.7	69.3	48.5	40.4
MDEQ [10]	impl	87.3	80.3	81.1	74.3	72.4	64.4	82.7	74.0	65.4	57.9
Segformer-B0 [4]	expl	89.7	83.9	86.2	80.7	73.5	65.7	88.2	80.3	65.7	58.3
PolypDEQ-add [11]	impl	90.4	84.6	**89.5**	**83.9**	**74.2**	66.4	87.3	79.3	**68.9**	**60.8**
PolypDEQ-res [11]	impl	90.5	84.7	88.8	83.2	74.2	**66.6**	87.6	79.3	68.3	60.5
HyperPolpDEQ (Ours)	impl	**90.7**	**84.9**	88.9	83.3	73.8	66.3	**88.3**	**80.4**	67.6	59.7

Table 3. Quantitative results on the NeoPolyp-Small dataset.

Method	**Mode**	**Neoplastic**		**Non-neoplastic**		**Polyp***	
		mDice	mIoU	mDice	mIoU	mDice	mIoU
MDEQ [10]	explicit	75.6	70.1	72.7	69.5	85.3	76.9
MDEQ [10]	implicit	81.5	76.6	78.4	75.9	87.7	80.2
Segformer-B0 [4]	explicit	**85.2**	**81.0**	79.3	77.1	89.2	82.4
PolypDEQ-add [11]	implicit	83.6	80.3	**81.6**	**79.4**	**90.8**	**84.8**
HyperPolpDEQ (Ours)	implicit	82.8	79.1	81.4	78.9	89.9	83.4

(*) We treat both Neoplastic and Non-neoplastic as a single class called Polyp and calculate the metrics

Table 4. Performance comparison in terms of inference time

Model	**Mode**	**Time (s)**	**Speed-up rate**
MDEQ [10]	explicit	0.011	-
PolypDEQ-add [11]	explicit	0.005	-
PolypDEQ-res [11]	explicit	0.005	-
MDEQ [10]	implicit	0.774	x 1
PolypDEQ-add [11]	implicit	0.549	x 1.41
PolypDEQ-res [11]	implicit	0.560	x 1.38
HyperPolypDEQ (Ours)	implicit	0.085	x 9.11

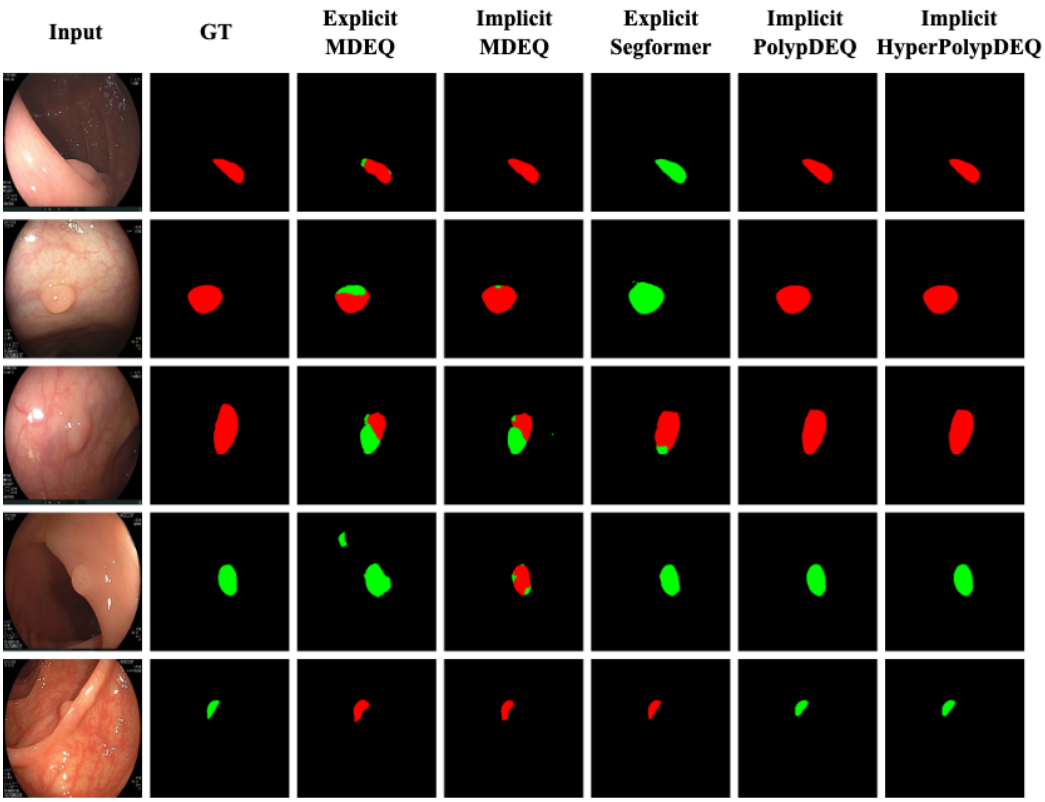

Fig. 4. Sample results from different models on the NeoPolyp-Small dataset.

On the NeoPolyp-Small dataset, while the baseline Segformer-B0 performs best for the neoplastic class, both the PolypDEQ and HyperPolypDEQ are marginally better for the more challenging non-neoplastic polyps. The slight accuracy loss observed for HyperPolypDEQ compared to PolypDEQ supports the findings in the original paper of HyperSolver [12], where the authors showed that HyperSolver might result in some negligible accuracy loss.

Fig. 4 visualizes some predictions of the available models made on images taken from the NeoPolyp-Small dataset. It is observed that the predictions made by explicit and implicit PolypDEQ are close to the ground truth label. Both MDEQ models are able to predict the location of polyps with some certain accuracy. However, they often misclassify the types of the detected polyps, PolypDEQ does not suffer from this issue. Note that the predictions of HyperPolypDEQ are very similar to those of PolypDEQ. This further proves that the accuracy loss when incorporating HyperSolver into PolypDEQ is negligible.

Table 4 provides a comparison of the inference speed of different models. We also compare the acceleration rate of our implicit models, PolypDEQ-add, PolypDEQ-res, and HyperPolypDEQ, to the baseline implicit MDEQ model. Overall, the implicit models had significantly slower inference speeds than their corresponding explicit models. For instance, the inference speed of Implicit MDEQ was approximately 70 times slower than its explicit version, with an inference time of 0.774s versus 0.011s, respectively. Similarly, PolypDEQ-add and PolypDEQ-res were approximately 100 times slower than their explicit counterparts. Compared to the baseline implicit MDEQ model, which used multi-scale internal state representation, our PolypDEQ can achieve a marginal speedup of 1.41 and 1.38 times for PolypDEQ-add and PolypDEQ-res, respectively. Including HyperSolver in our HyperPolypDEQ helped achieve a remarkable inference speed of 0.085s, which is about 6.5 times faster than the PolypDEQ-res and surpasses the implicit MDEQ by a large margin of 9.11 times acceleration. Note that this was achieved while sacrificing less than 1% accuracy, making it a desirable trade-off between inference time and accuracy. The results further demonstrate the novelty of HyperSolver, which can help implicit models achieve faster inference speeds while remaining accurate.

7. Conclusion

PolypDEQ has demonstrated superior efficiency compared to other DEQs and explicit models of comparable size. However, it suffers from low inference speed. This study focuses on enhancing inference speed by utilizing HyperSolver to accelerate finding fixed-point solutions. The proposed HyperPolypDEQ outperforms PolypDEQ in inference time while retaining a similar level of accuracy. We aspire that this research will play a pivotal role in enabling DEQs to be utilized in real-life applications that require real-time processing.

In future work, we plan to explore the use of more efficient and powerful backbone architectures to enhance accuracy and reduce inference time. Another challenge that arises during training DEQs is the issue of unstable convergence. We also intend to conduct further research and exploration to identify and implement effective strategies to overcome this issue.

Acknowledgments

This work was funded by Vingroup Innovation Foundation (VINIF) under project code VINIF.2020.DA17.

References

[1] Guanyu Zhou, Xiaogang Liu, Tyler M Berzin, Jeremy R Glissen Brown, Liangping Li, Chao Zhou, Zhenzhen Guo, Lei Lei, Fei Xiong, Yan Pan, *et al*., 951e-a real-time automatic deep learning polyp detection system increases polyp and adenoma detection during colonoscopy: a prospective double-blind randomized study, Gastroenterology, vol. 156, iss. 6, sup. 1, May. 2019 pp. S-1511. https://doi.org/10.1016/S0016-5085(19)40856-1

[2] Shin-ei Kudo, Yuichi Mori, Masashi Misawa, Kenichi Takeda, Toyoki Kudo, Hayato Itoh, Masahiro Oda, and Kensaku Mori, Artificial intelligence and colonoscopy: Current status and future perspectives, Digestive Endoscopy, vol. 31, iss. 4, Jan. 2019, pp. 363-371. https://doi.org/10.1111/den.13340

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-net: Convolutional networks for biomedical image segmentation, in International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, Nov. 2015, pp. 234-241. https://doi.org/10.1007/978-3-319-24574-4_28

[4] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo, Segformer, Simple and efficient design for semantic segmentation with transformers, Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 12077-12090.

[5] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming 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. https://doi.org/10.1007/978-3-030-00889-5_1

[6] Debesh Jha, Pia H. Smedsrud, Michael A. Riegler, Dag Johansen, Thomas De Lange, P˚al Halvorsen, and H˚avard D. Johansen, Resunet++: An advanced architecture for medical image segmentation, in 2019 IEEE International Symposium on Multimedia (ISM), 2019, pp. 225-232.

[7] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr, Res2net:

A new multi-scale backbone architecture, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell, Fully convolutional networks for semantic segmentation, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, Jun. 07-12, 2015, pp. 3431-3440. https://doi.org/10.1109/CVPR.2015.7298965

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May. 3-7, 2021.

[10] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter, Multiscale deep equilibrium models, in Advances in Neural Information Processing Systems (Neur IPS 2020), Vancouver, Canada, vol. 33, 2020, pp. 5238-5250.

[11] Nguyen Minh Chau, Le Truong Giang, and Dinh Viet Sang, Polypdeq: Towards effective transformer-based deep equilibrium models for colon polyp segmentation, in Advances in Visual Computing: 17th International Symposium, ISVC 2022, San Diego, CA, USA, Oct. 3-5, 2022, Proceedings, Part I, Springer, 2022, pp. 456-467. https://doi.org/10.1007/978-3-031-20713-6_35

[12] Shaojie Bai, Vladlen Koltun, and J Zico Kolter, Neural deep equilibrium solvers, in International Conference on Learning Representations (ICLR), 2022.

[13] Phan Ngoc Lan, Nguyen Sy An, Dao Viet Hang, Dao Van Long, Tran Quang Trung, Nguyen Thi Thuy, and Dinh Viet Sang, Neounet: Towards accurate colon polyp segmentation and neoplasm detection, in International Symposium on Visual Computing, Springer, 2021, pp. 15-28. https://doi.org/10.1007/978-3-030-90436-4_2

[14] Yundong Zhang, Huiye Liu, and Qiang Hu, Transfuse: Fusing transformers and cnns for medical image segmentation, in International Conference on Medical Image Computing and Computer-Assisted Intervention - MICCAI, Springer, 2021, pp. 14-24. https://doi.org/10.1007/978-3-030-87193-2_2

[15] Nguyen Thanh Duc, Nguyen Thi Oanh, Nguyen Thi Thuy, Tran Minh Triet, and Viet Sang Dinh, Colonformer: An efficient transformer based method for colon polyp segmentation. IEEE Access, vol. 10, Aug. 2022, pp. 80575-80586. https://doi.org/10.1109/ACCESS.2022.3195241

[16] John E Dennis, Jr and Jorge J Mor´e, Quasi-Newton methods, motivation and theory, SIAM Review, vol. 19, iss. 1, 1977, pp. 46-89. https://doi.org/10.1137/1019005

[17] Charles G Broyden. A class of methods for solving nonlinear simultaneous equations, Mathematics of Computation, vol. 19, 1965, pp. 577-593. https://doi.org/10.1090/S0025-5718-1965-0198670-6

[18] Homer F Walker and Peng Ni, Anderson acceleration for fixed-point iterations, SIAM Journal on Numerical Analysis, vol. 49, iss. 4, 2011, pp. 1715-1735. https://doi.org/10.1137/10078356X

[19] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud, Neural ordinary differential equations, Advances in Neural Information Processing Systems 31, 2018.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep residual learning for image recognition, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.

[21] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, Deep equilibrium models, Advances in Neural Information Processing Systems 32 (NeurIPS), 2019.

[22] Debesh Jha, Pia H Smedsrud, Michael A Riegler, P˚al Halvorsen, Thomas de Lange, Dag Johansen, and H˚avard D Johansen, Kvasir-seg: A segmented polyp dataset, in International Conference on Multimedia Modeling, Springer, 2020, pp. 451-462. https://doi.org/10.1007/978-3-030-37734-2_37

[23] Jorge Bernal, F Javier S´anchez, Gloria Fern´andez-Esparrach, Debora Gil, Cristina Rodr´ıguez, and Fernando Vilari˜no, Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians, Computerized Medical Imaging and Graphics, vol. 43, 2015, pp. 99-111. https://doi.org/10.1016/j.compmedimag.2015.02.007

[24] Nima Tajbakhsh, Suryakanth R Gurudu, and Jianming Liang. Automated polyp detection in colonoscopy videos using shape and context information. IEEE Transactions on Medical Imaging, vol. 35, iss. 2, Oct. 2015, pp. 630-644. https://doi.org/10.1109/TMI.2015.2487997

[25] Juan Silva, Aymeric Histace, Olivier Romain, Xavier Dray, and Bertrand Granado, Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer, International journal of Computer Assisted Radiology and Surgery, vol. 9, Sep. 2013, pp. 283-293. https://doi.org/10.1007/s11548-013-0926-3

[26] David V´azquez, Jorge Bernal, F Javier S´anchez, Gloria Fern´andez Esparrach, Antonio M L´opez, Adriana Romero, Michal Drozdzal, and Aaron Courville, A benchmark for endoluminal scene segmentation of colonoscopy images, Journal of Healthcare Engineering, vol. 2017, iss. 1, Jul. 2017. https://doi.org/10.1155/2017/4037190