Path-Adaptive Matting for Efficient Inference Under Various Computational Cost Constraints

Qinglin Liu [1], Zonglin Li[1], Xiaoqian Lv[1*], Xin Sun[1], Ru Li[1], Shengping Zhang[1]

[1] Harbin Institute of Technology, Weihai, China
qinglin.liu@outlook.com, zonglin.li@hit.edu.cn, xiaoqian.hit@gmail.com,
sunxintyc@hit.edu.cn, liru@hit.edu.cn, s.zhang@hit.edu.cn

Abstract

In this paper, we explore a novel image matting task aimed at achieving efficient inference under various computational cost constraints, specifically FLOP limitations, using a single matting network. Existing matting methods which have not explored scalable architectures or path-learning strategies, fail to tackle this challenge. To overcome these limitations, we introduce Path-Adaptive Matting (PAM), a framework that dynamically adjusts network paths based on image contexts and computational cost constraints. We formulate the training of the computational cost-constrained matting network as a bilevel optimization problem, jointly optimizing the matting network and the path estimator. Building on this formalization, we design a path-adaptive matting architecture by incorporating path selection layers and learnable connect layers to estimate optimal paths and perform efficient inference within a unified network. Furthermore, we propose a performance-aware path-learning strategy to generate path labels online by evaluating a few paths sampled from the prior distribution of optimal paths and network estimations, enabling robust and efficient online path learning. Experiments on five image matting datasets demonstrate that the proposed PAM framework achieves competitive performance across a range of computational cost constraints.

Introduction

Natural image matting is a classic computer vision task, aiming to estimate the alpha matte of the foreground in a given image. This technique serves as a key technology in many applications such as remote meetings, live streaming, and post-production in film. Hence, it has been extensively researched over the past few decades. Mathematically, the matting problem aims to estimate the alpha matte α given an image I and its foreground F and background B as

$$\alpha = \arg\min_{\alpha} |I - \alpha \cdot F + (1 - \alpha) \cdot B| \tag{1}$$

Since only I is known while F and B are both unknown, image matting is an ill-posed problem necessitating additional assumptions or knowledge for resolution. Traditional matting methods (Wang and Cohen 2007; Ranjbar, Abdelaziz, and Jauhar 2008; He et al. 2011) estimate the alpha matte by distinguishing foreground and background or propagating color information, which often struggle in scenarios with overlapping color distributions. Recently, deep matting methods (Qiao et al. 2020; Park et al. 2022; Yao et al. 2024) have employed neural networks to estimate the alpha matte, leveraging high-level semantics to distinguish foreground and background, thereby achieving significant progress.

Despite the remarkable performance achieved by deep image matting methods, real-world image editing applications prioritize achieving efficient inference across different hardware. Unfortunately, most of these methods rely on complex architectures, making them only suitable for GPU deployment. For instance, FBAMatting, a popular matting method (Forte and Pitié 2020), consists of 34.8 million parameters and requires 686 GFlops to process a 1024×1024 image, making it impractical for smartphones. Furthermore, the architectures of current matting methods are non-scalable and require extensive tuning and retraining to achieve optimal performance under different computational cost constraints (Hu et al. 2023; Yao et al. 2024), leading to substantial computational overhead and carbon dioxide emissions. While dynamic network methods based on the Gumbel-Softmax technique (Yu et al. 2021; Li et al. 2021; Han et al. 2022, 2024) hold promise for constructing scalable matting networks under computational cost constraints, their path-learning strategies largely suffer from the well-known performance collapse issue due to the aggregation of skip connections (Xu et al. 2019; Chu and Zhang 2021).

In this paper, we present Path-Adaptive Matting (PAM), a framework that learns to dynamically adjust the network path based on image contexts and computational cost constraints, specifically FLOP limitations, for efficient inference. Our approach first formulates computational cost-constrained matting network training as a bilevel optimization problem, which optimizes the matting network and the path estimator successively (Anandalingam and Friesz 1992; Colson, Marcotte, and Savard 2007). Based on this formulation, we then design a path-adaptive matting architecture by incorporating path selection layers and learnable connect layers to estimate optimal paths and perform efficient inference within a unified network. Then, we propose a performance-aware path learning strategy to generate network path labels online by evaluating only a few network paths sampled from the prior distribution of optimal paths

*Corresponding Author

and network estimations, thus enabling robust and efficient online path learning. Extensive experimental results on five image matting datasets demonstrate that the proposed PAM framework achieves competitive performance across a range of computational cost constraints.

The main contributions of this paper are as follows

- We present Path-Adaptive Matting (PAM), the first image matting framework that can adaptively adjust the network path based on the image contexts and computational cost constraints for efficient inference.
- We formalize computational cost-constrained matting network training as a bilevel optimization problem, optimizing both the matting network and the path estimator, and design a path-adaptive matting architecture to address it using a unified network.
- We introduce a performance-aware path learning strategy to generate path labels online by evaluating only a few network paths sampled from the prior distribution of optimal paths and network estimations, thus enabling robust and efficient online path learning.
- Extensive experiments on five popular image matting datasets demonstrate that the proposed PAM framework achieves competitive performance across a range of computational cost constraints.

Related Work

Researchers have delved into deep image matting methods and efficient neural network design. Here, we provide an overview of the significant related work in this field.

Deep image matting methods. Deep image matting methods (Xu et al. 2017; Yu et al. 2021; Dai, Lu, and Shen 2021; Dai et al. 2022; Liu et al. 2024) learn generalizable knowledge from the dataset to estimate alpha matte. DIM (Xu et al. 2017) provides the first image matting dataset and introduces an end-to-end matting network, which makes the method robust in a variety of complex scenes. GCAMatting (Li and Lu 2020) adopts a Guided Contextual Attention module to extract contextual affinity to estimate the alpha matte of semi-transparent objects. HDMatt (Yu et al. 2021) adopts a cross-patch contextual module to aggregate image contexts for robust patch-based matting. MGMatting (Yu et al. 2021) adopts a progressive refinement to estimate the alpha matte from coarse masks, which avoids the laborious trimap annotation. TIMI (Liu et al. 2021b) proposes a 3-branch encoder to mine the global and local neglected coordination, which helps improve predictions on rough trimaps. LFPNet (Liu et al. 2021a) adopts a center-surround pyramid pooling to propagate contextual information to help process high-resolution images. RMat (Dai et al. 2022) adopts the transformer to aggregate image contexts and explores data augmentation for improving method robustness. MODNet (Ke et al. 2022) employs a lightweight matting network and a self-supervised strategy to improve the efficiency and robustness. MatteFormer (Park et al. 2022) integrates CNN and transformer to extract detailed features and long-range features to achieve high performance. VitMatte (Yao et al. 2024) employs vision Transformers with a hybrid attention mechanism to improve performance.

Efficient neural network design. Early researchers proposed neural architecture search methods (Xie and Yuille 2017; Liu, Simonyan, and Yang 2019; Chen et al. 2019; Xu et al. 2019) to design efficient neural networks under various computational cost constraints. However, these methods are often time-consuming due to the need for training and evaluating independent structures. Therefore, recent studies have introduced dynamic neural networks (Wang et al. 2018; Liu et al. 2020b; Schwartz et al. 2020; Zhou et al. 2020) that adjust network structures during inference, significantly reducing the cost of network design by training only one supernet. Early research explored early-exit networks (Xin et al. 2020; Schwartz et al. 2020; Zhou et al. 2020), which reduce computational costs by leveraging predictions from early layers of the network. MSDNet (Huang et al. 2018) adopts a multi-scale architecture, stopping inference when shallow networks achieve high-confidence results to avoid redundant computation. RANet (Yang et al. 2020) employs intermediate classifiers at different scales within a single network, terminating inference when high-confidence predictions are obtained at lower scales. However, these networks exhibit significant differences in feature representations across scales and are primarily suitable for classification tasks. Recent research (Wang et al. 2018; Yu et al. 2021) focuses on enhancing feature continuity through dynamic depth or width adjustment. DRNet (Cai, Shu, and Wang 2021) employs an undirected graph design to predict the connections of current module nodes, altering network width. LASNet (Han et al. 2022) introduces a Gumbel-Softmax based mask estimator that guides the network bypass low-value regions, thereby accelerating the inference. However, these works focus on classification tasks, with limited attention to dense regression at high resolutions, making them less effective for addressing the computational cost constrained matting task explored in this work.

Method

In this section, we first formalize training a computational cost-constrained matting network as a bilevel optimization problem, where the upper level optimizes the network path estimator and the lower level optimizes the matting network. Based on this formalization, we introduce the Path-Adaptive Matting (PAM) framework to tackle this problem using a unified network. Subsequently, we delve into a comprehensive introduction of the PAM framework, elucidating its path-adaptive matting architecture and the performance-aware path learning strategy. Finally, we present the loss functions incorporated within the framework.

Problem Formulation

The objective of training a computational cost-constrained image matting network is to achieve the minimum matting error while adhering to a specified computational cost constraint C_c. This problem can be cast as a bilevel optimization problem: the lower level entails the optimization of weights across all conceivable matting networks, while the upper level optimizes the parameters of an estimator (using a neural network for approximation) to predict the optimal net-

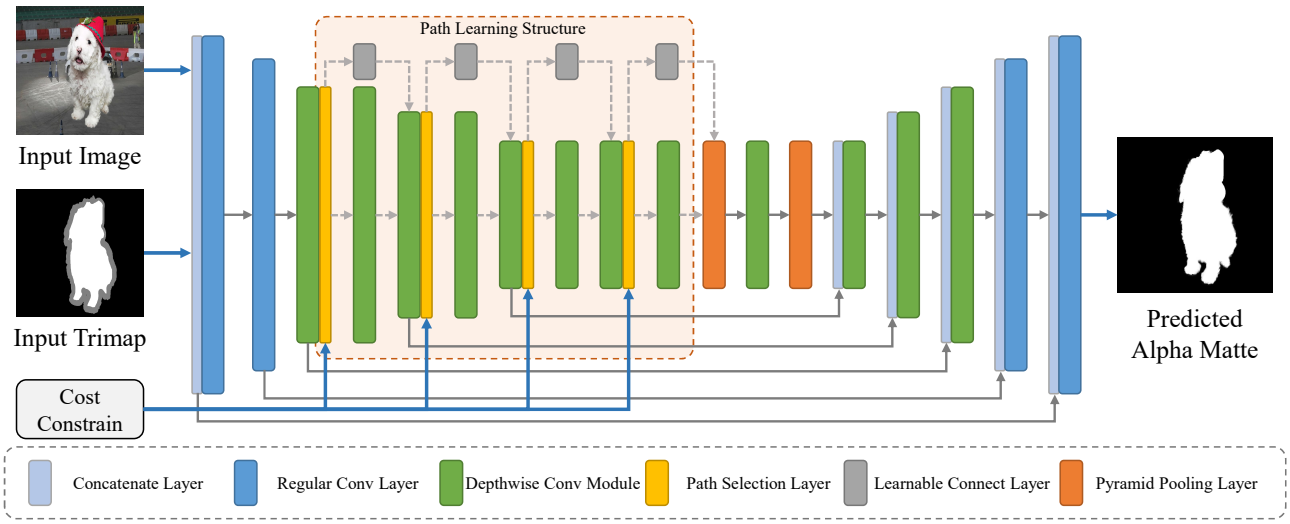

Figure 1: Overview of the path-adaptive matting architecture. We build a lightweight matting backbone using regular and depthwise convolution layers. To enable path-adaptive inference, a path-learning structure is introduced, which uses path selection layers to estimate network paths based on cost constraints and image context, and learnable connect layers for layer bypassing.

work path. Mathematically, it can be expressed as

$$\begin{aligned} &\omega_p^* = \arg\min_{\omega_p} \mathbb{E}_{I,C_c}\left[\mathcal{L}\left(N_m(I,\omega_n^*,N_p(I,C_c,\omega_p)),\alpha^{gt}\right)\right] \\ &\text{s.t.} \quad \omega_n^* = \arg\min_{\omega_n} \mathbb{E}_{I,P}\left[\mathcal{L}\left(N_m(I,\omega_n,P),\alpha^{gt}\right)\right] \\ &\qquad \mathrm{F_c}\left(N_p(I,C_c,\omega_p)\right) \le C_c \end{aligned} \tag{2}$$

Here, ω_p and ω_n denote the network weights, and $\mathcal{L}(\cdot,\cdot)$ denotes the loss function. α^{gt} represents the ground truth alpha matte. The function $N_m(I,\omega_n,P)$ represents a matting super network that can be inferred with a given path P and network weights ω_n. The function $N_p(C_c,\omega_p)$ represents a path estimation network that can be inferred with a given path C_c and network weights ω_p. The computational cost of the network is denoted by $\mathrm{F_c}(\cdot)$.

Based on Equation 2, it is evident that the problem involves training two networks. The lower-level task focuses on optimizing the weights ω_n of the matting network N_m to ensure high matting performance across all paths P. Conversely, the upper-level task involves optimizing the weights ω_p of the path estimation network N_p to predict network paths that adhere to the computational cost constraint C_c while achieving the best performance. However, employing two networks for prediction entails significant computational costs when inferring high-resolution images, thus substantially affecting computational efficiency. Therefore, we propose that network path estimation can be conducted locally based on the current image context and computational cost constraints. Building upon this, we employ a method of predicting the path of the subsequent network stage at each stage of the network, enabling the resolution of the bilevel optimization problem using a unified network. In particular, we design a path-adaptive matting architecture composed of efficient matting network layers and leverage a path learning structure to estimate paths with image contexts and computational cost constraints, thereby dynamically controlling the inference of the matting network.

Path-Adaptive Matting Architecture

As illustrated in Figure 1, the path-adaptive matting architecture first constructs a lightweight matting network. Regular convolution layers are adopted at the lower levels, while the higher levels are based on depthwise convolution modules that consist of large kernel depthwise convolution and point wise convolution, which ensures high computational efficiency and performance. In addition, we introduce stacked pyramid pooling modules (Zhao et al. 2017) to help aggregate image semantics. Based on the lightweight network, we incorporate cost constraint embedding to input the cost constraint. We then adopt a path learning structure, which includes path selection layers and learnable connect layers, to estimate optimal paths and perform efficient inference along the estimated paths.

Cost Constraint Embedding. To feed computational cost constraint information to the path learning structures, we embed the input computational cost constraint into fixed-length features. To match the constraints with the network architecture, we first evaluate the smallest and largest subnetworks and obtain their computational costs C_{min} and C_{max}, respectively. Then, we use an embedding layer to convert the integer computational cost constraint C_c between C_{min} and C_{max} into the computational cost constraint feature $\boldsymbol{F}^c$, which is subsequently fed to the network.

Path Learning Structure. The path-selective structure uses path selection layers and learnable connect layersto estimate the optimal path based on given computational cost constraints and image contexts, and to perform learnable layer bypassing, respectively. Below, we will provide a detailed explanation of these layers.

Path Selection Layer. To estimate the optimal network path under the given computational cost constraint and image contexts, we design a path selection layer to estimate the classification of the path, i.e., whether to bypass adjacent layers. Considering that the global semantics can aid in determining the optimal network path and be used to generate attention weights to refine features, we align the path selection layer with a channel attention design. We obtain global semantics by averaging the image features $\boldsymbol{F}_i^s$ from the i-th layer, and then generate channel attention to refine the network features and use multi-layer perceptron to estimate the distribution of the optimal network paths $\boldsymbol{Q}_i^e$ as

$$\begin{aligned} \boldsymbol{F}_i^{gs} &= \mathrm{Relu}(\mathrm{Conv}(\mathrm{Avgpool}(\boldsymbol{F}_i^s))) \\ \boldsymbol{F}_i^{rf} &= \boldsymbol{F}_i^s \otimes \mathrm{Sigmoid}(\mathrm{Conv}(\boldsymbol{F}_i^{gs})) \\ \boldsymbol{Q}_i^e &= \mathrm{MLP}(\mathrm{Concat}(\boldsymbol{F}_i^{gs}, \boldsymbol{F}^c)) \end{aligned} \tag{3}$$

Here, $\boldsymbol{F}_i^{rf}$ denotes the refined semantic features. $\mathrm{Conv}(\cdot)$ $\mathrm{MLP}(\cdot)$ and $\mathrm{Concat}(\cdot,\cdot)$ denote the convolution, multi-layer perceptron, and concatenation layers, respectively.

Learnable Connect Layer. To enhance network efficiency, we bypass unnecessary paths. However, naive skip connections often behave differently from depthwise convolution modules, leading to discrepancies between the features of bypassed and non-bypassed paths, which can impact performance. To address this challenge, we introduce learnable connect layers designed to approximate the transformation of features from bypassed layers. The learnable connect layer uses a residual block consisting of two 1×1 pointwise convolution layers to process the source feature $\boldsymbol{F}^{src}$ and output the destination feature $\boldsymbol{F}^{des}$ as

$$\boldsymbol{F}^{des} = \mathrm{Conv}(\mathrm{ReLU}(\mathrm{Conv}(\boldsymbol{F}^{src}))) + \boldsymbol{F}^{src} \tag{4}$$

Since our learnable connect layer only involves pointwise convolutions, it is more computationally efficient than the bypassed depthwise convolution modules.

Performance-Aware Path Learning

With the path-adaptive matting architecture, our objective is to train the network to estimate the optimal path and perform efficient inference. However, we have empirically found that using the Gumbel-Softmax trick (Han et al. 2022, 2024) often leads the path selection layers to favor the shortest path. This is because gradient-based NAS methods inherently suffer from operation co-adaptation issues, which often lead to an excessive aggregation of skip connections. Conversely, while training the network with the optimal path yields good results, obtaining the ground truth distribution requires evaluating all paths in each iteration, which is computationally expensive. To tackle these issues, we propose a performance-aware path learning strategy that generates path labels through online evaluation of paths generated by a prior distribution of the optimal path and the network predictions. Specifically, we use the Monte Carlo to estimate a prior distribution of the optimal path as a path generator. During training, we use an online path label generation method as described in Algorithm 1, evaluating both the paths generated from the prior distribution and those predicted by the network. The path with the highest performance is used as the label.

Algorithm 1: Performance-Aware Path Learning

Input: Paths $\mathbf{R} = \{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{N^e}\}$ generated by the prior distribution and the matting errors $\mathbf{E} = \{E_1, E_2, \ldots, E_{N^e}\}$, path $\boldsymbol{V}$ estimated by the network and the corresponding matting error E^v and computational cost C^v, the given computational cost constraint C^g.
Output: Optimal path $\boldsymbol{R}^o$.
Initialize the optimal path $\boldsymbol{R}^o \leftarrow \emptyset$, the lowest matting error $E^o \leftarrow +\infty$
if $C^v < C^g$ **then**
  $\boldsymbol{R}^o \leftarrow \boldsymbol{V}$
  $E^o \leftarrow E^v$
end if
for $i \leftarrow 1$ to N^e **do**
  if $E_i < E^o$ **then**
    $\boldsymbol{R}^o \leftarrow \boldsymbol{R}_i$
    $E^o \leftarrow E_i$
  end if
end for

Prior Distribution Estimation. Given a computational cost constraint C_c and an input image $\boldsymbol{I}$, the ground truth distribution $\boldsymbol{Q}^{gt} = \mathrm{P}(X_1, X_2 \ldots, X_{N^a} | \boldsymbol{I}, \boldsymbol{F}^c)$ of optimal paths can only be obtained by evaluating all paths in each iteration, which is computationally unaffordable. To solve this problem, we use a distribution that approximates $\boldsymbol{Q}^{gt}$ as a prior distribution to generate a few optimal candidate paths for efficient path evaluation. We compare a uniform distribution $\boldsymbol{Q}^u$ and a distribution $\boldsymbol{Q}^c = \mathrm{P}(X_1, X_2 \ldots, X_{N^p} | \boldsymbol{F}^c)$ of the optimal path under the given computational cost constraint. The expectation of $p(X_1^r, X_2^r \ldots, X_{N^a}^r | \boldsymbol{I}, \boldsymbol{F}^c)$ is

$$\mathbb{E}(p(X_1^r, X_2^r \ldots, X_{N^a}^r | \boldsymbol{I}, \boldsymbol{F}^c)) = \frac{1}{N^{ap}} \tag{5}$$

where $X_1^r, X_2^r \ldots, X_{N^a}^r$ is the path sampled from the uniform distribution $\boldsymbol{Q}^u$, N^{ap} is the number of all possible paths. The expectation of $p(X_1^c, X_2^c \ldots, X_{N^a}^c | \boldsymbol{I}, \boldsymbol{F}^c)$ is

$$\mathbb{E}(p(X_1^c, X_2^c \ldots, X_{N^a}^c | \boldsymbol{I}, \boldsymbol{F}^c)) = p(X_1^o, X_2^o \ldots, X_{N^a}^o | \boldsymbol{F}^c) \tag{6}$$

where $X_1^c, X_2^c \ldots, X_{N^a}^c$ is the path sampled from the distribution $\boldsymbol{Q}^c$ and $X_1^o, X_2^o \ldots, X_{N^a}^o$ is the optimal path. Since $p(X_1^o, X_2^o \ldots, X_{N^a}^o | \boldsymbol{F}^c)$ is usually much larger than $\frac{1}{N^{ap}}$, the path sampled from the distribution $\boldsymbol{Q}^c$ is more likely to be optimal. Therefore, we adopt the distribution $\boldsymbol{Q}^c$ as the prior distribution and use the Monte Carlo method to estimate the distribution via simulation.

To obtain the distribution $\boldsymbol{Q}^c$, we first follow SPOS (Guo et al. 2020) to train matting networks of random paths in the path-adaptive matting architecture. Then, we define a probability estimator that the path $X_1, X_2 \ldots, X_{N^a}$ is the optimal path under the computational cost constraint C_g as

$$\widehat{p}_{N^{val}}^{g}(X_1, X_2 \ldots, X_{N^a}) = \frac{1}{N^{val}} \sum_{i=1}^{N^{val}} B_i \tag{7}$$

where N^{val} denotes the number of input images. $B_i \in \{0, 1\}$ denotes whether the path $X_1, X_2 \ldots, X_{N^a}$ is opti-

mal or not for the i-th image and the computational cost constraint C_g. Next, we simulate N^{val} images using the foreground and background images in the dataset. Since the computational cost is positively correlated with the network performance, we simulate N^g paths whose computational costs are close to the given computational cost constraint C_g to reduce computational costs. Finally, we use a pretrained whole matting network to evaluate the simulated images and paths and obtain the estimate $\widehat{\boldsymbol{Q}}^c$ of the distribution as the prior distribution.

Online Path Label Generation. Although the prior distribution provides good path candidates, using it as a label limits the network's ability to search for better paths. Inspired by heuristic algorithms, we propose an online path label generation method, where we assess the paths generated by the prior distribution and those predicted by the network. The best-performing path among these is then used as the label for training. In each iteration of the training phase, we first generate candidate paths for the given computational constraint C^g with the prior distribution. Then, we use the distribution $\widehat{\boldsymbol{Q}}^c$ to randomly sample N^e candidate paths $\mathbf{R} = \{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{N^e}\}$ without repetition, and evaluate the sampled paths with the L1 loss function to obtain the errors $\mathbf{E} = \{E_1, E_2, \ldots, E_{N^e}\}$. Next, we use the network itself to sample the path $\boldsymbol{V}$ by independently estimating the path $\boldsymbol{V}_i$ for the i-th path selection layer with the unnormalized distribution $\boldsymbol{Q}_i^e$ as

$$V_{i,j} = \frac{\exp((Q_{i,j}^e + G_{i,j})/\tau)}{\sum_{k=1}^{2} \exp((Q_{i,k}^e + G_{i,k})/\tau)} \tag{8}$$

where $\boldsymbol{G} \in \mathbb{R}^{N^a \times 2}$ is a Gumbel noise that helps the network to explore low probability paths. $V_{i,j}$, $Q_{i,j}^e$, and $G_{i,j}$ are the j-th items of $\boldsymbol{V}_i$, $\boldsymbol{Q}_i^e$, and $\boldsymbol{G}_i$, respectively. τ is the temperature coefficient that controls the smoothness of the result. In addition, we use a gumbel-max reparameterization trick (Huijben et al. 2022) to convert the estimated path $\boldsymbol{V}_i$ to a one-hot tensor while keeping it differentiable. We evaluate the matting network with the path $\boldsymbol{V}$ to obtain the error E^v and the computational cost C^v. Finally, we use the evaluation algorithm described in Algorithm 1 to identify the path with the lowest error as the online path label $\boldsymbol{R}^o$, which is subsequently used to train the network.

Loss Functions

To train the PAM framework, we define the network loss as

$$\mathcal{L} = \lambda_\alpha \mathcal{L}_\alpha + \lambda_{ds} \mathcal{L}_{ds} + \lambda_{pt} \mathcal{L}_{pt} \tag{9}$$

where $\mathcal{L}_\alpha$ is the alpha matte loss defined as

$$\mathcal{L}_\alpha = \lambda_1 \mathcal{L}_1 + \lambda_{comp} \mathcal{L}_{comp} + \lambda_{lap} \mathcal{L}_{lap} \tag{10}$$

where $\mathcal{L}_1$, $\mathcal{L}_{comp}$, and $\mathcal{L}_{lap}$ are the L1 loss, compositional loss, and Laplacian loss, as defined in MatteFormer (Park et al. 2022). λ_1, λ_{comp}, and λ_{lap} are the weights for the three losses. The distillation loss $\mathcal{L}_{ds}$ is designed to help the learnable connect layers to learn the feature transformation of the bypassed layer with the labels generated by a trained complete PAM network, which is defined as

$$\mathcal{L}_{ds} = \frac{1}{|\mathcal{T}^U|} \sum_{i \in \mathcal{T}^U} \sqrt{(\alpha_i - \alpha_i^{sd})^2 + \epsilon^2} \tag{11}$$

where α_i^{sd} is the alpha matte of the i-th pixel estimated by a trained complete PAM network. $\mathcal{T}^U$ is a set of unknown pixels in the trimap. ϵ is the penalty coefficient. The path loss $\mathcal{L}_{pt}$ is designed to use the optimal path $\boldsymbol{R}^o$ to supervise the predictions of path selection layers $\boldsymbol{Q}^e$ as

$$\mathcal{L}_{pt} = \sum_{i=1}^{N^a} \mathrm{CrossEntroy}(\boldsymbol{Q}_i^e, \boldsymbol{R}_i^o) \tag{12}$$

where $\mathrm{CrossEntroy}(\cdot, \cdot)$ denotes cross entropy function.

Experiments

In this section, we first present the implementation details of the proposed PAM framework. Next, we evaluate PAM against existing methods using synthetic datasets, including Adobe Composition-1k (Xu et al. 2017), Distinctions-646 (Qiao et al. 2020), Transparent-460 (Cai et al. 2022), Semantic Image Matting (SIMD) (Sun, Tang, and Tai 2021), as well as the real-world Automatic Image Matting-500 (AIM-500) dataset (Li, Zhang, and Tao 2021).

Implementation details

The proposed method is implemented using the PyTorch framework. We train our PAM framework on an RTX 3090 GPU with a batch size of 4. All network weights are initialized using the Kaiming initializer (He et al. 2015). To avoid overfitting, we follow the data preprocessing methods of previous matting methods to process the train data (Forte and Pitié 2020). The training process is divided into three stages. The networks are trained using the Radam optimizer (Liu et al. 2020a) with a weight decay of 3×10^{-5} and betas of $(0.5, 0.999)$. The initial learning rate is set to 3×10^{-4} and decays to zero using a cosine annealing scheduler in each stage. In the first stage, we train the entire PAM network for 150 epochs. In the second stage, we perform warm-up training by randomly sampling sub-networks and training them for 20 epochs. In the third stage, we train PAM with the performance-aware path learning strategy for 150 epochs. The other coefficients used in this method are configured as follows: $N^a = 4$, $\lambda_\alpha = 1$, $\lambda_{ds} = 0.05$, $\lambda_{pt} = 0.05$, $\lambda_1 = 1$, $\lambda_{comp} = 0.25$, $\lambda_{lap} = 0.5$, $\epsilon = 10^{-6}$, $N^e = 4$, $N^{val} = 10^3$, $N^g = 10$, and $\tau = 1$.

Results on the Synthetic Datasets

To evaluate the performance of our PAM framework, we compare PAM with state-of-the-art methods, including DIM (Xu et al. 2017), IndexNet (Lu et al. 2019), GCAMatting (Li and Lu 2020), FBAMatting (Forte and Pitié 2020), SIM (Sun, Tang, and Tai 2021), TIMI (Liu et al. 2021b), LFPNet (Liu et al. 2021a), MatteFormer (Park et al. 2022), DiffusionMat (Xu et al. 2023), and VitMatte (Yao et al. 2024) on the Adobe Composition-1K dataset. Table 1 provides a

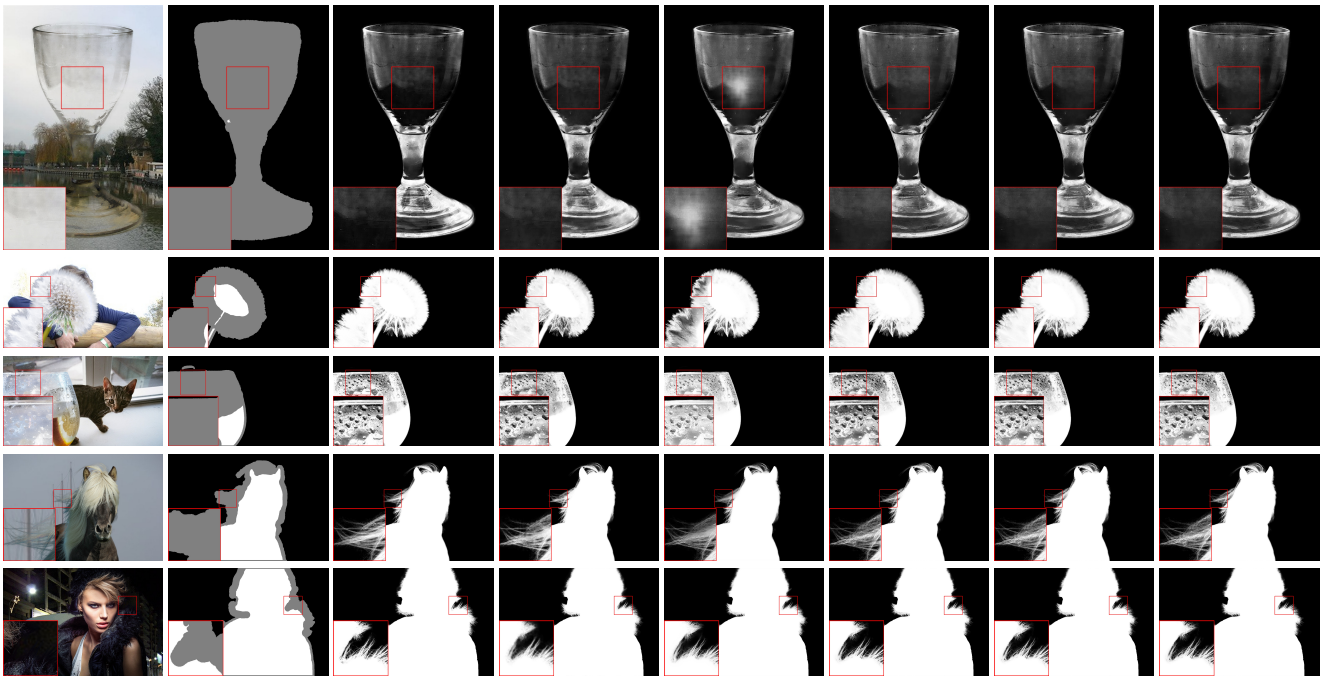

Figure 2: Qualitative results on the Adobe Composition-1K dataset. (a) Input Image. (b) Trimap. (c) Ground Truth. (d) GCA-Matting. (e) MatteFormer. (f) PAM (Aggressive). (g) PAM (Moderate). (h) PAM (Mild).

Method	SAD	MSE	GRAD	CONN	Flops	Param	Latency	Memory
DIM (Xu et al. 2017)	50.40	17.00	36.70	55.30	727.4G	130.5M	11.97s	18 GB
IndexNet (Lu et al. 2019)	45.80	13.00	25.90	43.70	116.6G	8.2M	7.67s	8 GB
GCAMatting (Li and Lu 2020)	35.28	9.00	16.90	32.50	257.3G	24.1M	37.76s	13 GB
FBAMatting (Forte and Pitié 2020)	26.40	5.40	10.60	21.50	686.0G	34.8M	15.44s	13 GB
SIM (Sun, Tang, and Tai 2021)	27.70	5.60	10.70	24.40	1001.9G	44.5M	20.17s	14 GB
TIMI (Liu et al. 2021b)	29.08	6.00	11.50	25.36	351.3G	33.5M	124.07s	72 GB
LFPNet (Liu et al. 2021a)	23.60	4.10	8.40	18.50	1539.4G	112.2M	45.93s	16 GB
MatteFormer (Park et al. 2022)	23.80	4.00	8.70	18.90	233.3G	44.9M	14.39s	11 GB
DiffusionMat (Xu et al. 2023)	22.80	4.00	6.80	18.40	29212.5G	32.8M	551.00s	33GB
VitMatte (Yao et al. 2024)	20.33	3.00	6.74	14.78	784.8G	89.2M	130.71s	30GB
PAM (Mild)	23.14	4.17	8.48	18.57	74.6G	7.1M	6.25s	8 GB
PAM (Moderate)	23.25	4.25	8.50	18.71	69.6G	7.1M	5.58s	8 GB
PAM (Aggressive)	24.05	4.52	8.97	19.63	57.8G	7.1M	5.23s	8 GB

Table 1: Quantitative results on Adobe Composition-1K. Our PAM method is evaluated under three computational cost constraints: mild (under 75 GFlops), moderate (under 70 GFlops), and aggressive (under 60 GFlops). Flops denotes the floating-point computations required for inferring a 1024×1024 image. Param denotes the network parameter number. Latency and memory refer to the latency and peak memory usage measured on an R9 3900X CPU for inferring a 2048×2048 image.

summary of the quantitative results for all compared matting methods. Our PAM is evaluated under three computational cost constraints: mild (under 75 GFlops), moderate (under 70 GFlops), and aggressive (under 60 GFlops). Flops denote the floating-point computations required for inferring a 1024×1024 image. Param represents the number of network parameters. Latency and memory refer to the inference latency and memory usage, respectively, measured on an AMD R9 3900X CPU for inferring a 2048×2048 image. Figure 2 illustrates the qualitative results of PAM compared with other methods. In comparison with existing methods, our demonstrates competitive performance with significantly lower computational costs and fewer parameters. PAM under the moderate constraint performs comparably to state-of-the-art image matting methods including LFPNet and MatteFormer. However, PAM under the moderate

Method	Distinctions-646				Transparent-460			
	SAD	MSE	GRAD	CONN	SAD	MSE	GRAD	CONN
DIM (Xu et al. 2017)	56.13	22.84	50.01	57.90	356.20	49.68	146.46	296.31
IndexNet (Lu et al. 2019)	40.31	8.23	37.60	39.92	434.14	74.73	124.98	368.48
GCAMatting (Li and Lu 2020)	31.32	6.64	28.69	30.45	219.38	23.17	130.46	224.65
TIMI (Liu et al. 2021b)	42.61	7.75	45.05	42.40	328.08	44.20	142.11	289.79
MGMatting (Yu et al. 2021)	33.24	4.51	20.31	25.49	344.65	57.25	74.54	282.79
TransMatting (Cai et al. 2022)	25.65	3.40	16.08	21.45	192.36	20.96	41.80	158.37
PAM (Mild)	21.97	3.45	15.80	20.65	194.09	20.27	34.26	197.35
PAM (Moderate)	22.02	3.44	15.73	20.72	199.24	21.60	34.30	203.65
PAM (Aggressive)	22.97	3.78	16.31	21.90	210.11	23.55	41.42	215.95

Table 2: Quantitative results on Distinctions-646 and Transparent-460.

Method	Semantic Image Matting				Automatic Image Matting-500			
	SAD	MSE	GRAD	CONN	SAD	MSE	GRAD	CONN
DIM (Xu et al. 2017)	98.95	61.34	32.19	103.86	47.81	79.47	38.16	49.07
IndexNet (Lu et al. 2019)	62.90	25.00	21.76	63.43	26.85	26.22	16.37	26.15
GCAMatting (Li and Lu 2020)	72.18	29.82	23.88	71.52	34.81	38.93	25.72	35.14
LFPNet (Liu et al. 2021a)	22.11	4.32	6.82	17.06	26.15	21.14	14.93	25.73
MatteFormer (Park et al. 2022)	23.59	4.88	7.67	18.69	26.87	29.00	23.00	26.63
PAM (Mild)	24.39	4.73	7.52	20.24	25.14	23.73	19.74	25.02
PAM (Moderate)	24.68	4.85	7.56	20.54	25.12	23.76	19.79	25.08
PAM (Aggressive)	25.37	5.25	7.95	21.42	24.35	23.47	19.82	24.33

Table 3: Quantitative results on Semantic Image Matting and Automatic Image Matting-500.

constraint consumes only 4.5% computation and 6.3% parameters of LFPNet, 29.8% computation and 15.8% parameters of MatteFormer.

To evaluate the generalization ability of PAM, we compare it with existing matting methods such as IndexNet, GCAMatting, TIMI, TransMatting (Cai et al. 2022), and MatteFormer on three synthetic datasets including Distinctions-646, Transparent-460, and Semantic Image Matting. All compared methods are trained on the Adobe Composition-1K dataset. Quantitative results are summarized in Tables 2 and 3. DIM, IndexNet, GCAMatting, LFPNet, MatteFormer, and our PAM exhibit strong performance across all three datasets. Notably, PAM achieves performance on par with leading methods like TransMatting and MatteFormer, indicating its robust generalization ability. Furthermore, as computational cost constraints relax, the performance of PAM also improves, underscoring the effectiveness of our proposed method.

Results on the Real-world Dataset

To evaluate the performance of the proposed PAM framework on real-world images, we use the Automatic Image Matting-500 (AIM-500) dataset to evaluate DIM, IndexNet, GCAMatting, LFPNet, MatteFormer, and our PAM, which are trained on the Adobe Composition-1K dataset. Table 3 summarizes the quantitative results of all compared methods. Compared with existing methods, PAM demonstrates strong generalization ability when applied to real-world images. PAM under the mild computational cost constraint has a similar performance to LFPNet and MatteFormer, which suggests that PAM has good generalization ability on real-world images. However, PAM under the aggressive constraint outperforms PAM under the mild constraint, which may be due to the domain shift when testing PAM on real data, as it is trained on synthetic data.

Conclusion

In this paper, we propose Path-Adaptive Matting (PAM), a framework that dynamically adjusts network paths based on image contexts and computational cost constraints for efficient inference. We first formulate the training of the computational cost-constrained matting network as a bilevel optimization problem, optimizing both the matting network and the path estimator. We introduce a path-adaptive matting architecture by incorporating path selection layers and learnable connect layers to estimate optimal paths and perform efficient inference within a unified network. Furthermore, we propose a performance-aware path learning strategy to generate path labels by evaluating a few paths sampled from the prior distribution of optimal paths and network estimations, thus enabling robust and efficient online path learning. Experimental results on popular image matting datasets demonstrate that PAM achieves competitive performance across various computational cost constraints.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 62272134, 62072141 and 62402136, in part by the National Natural Science Foundation of Shandong Province under Grants ZR2024QF136 and ZR2024QF064.

References

Anandalingam, G.; and Friesz, T. L. 1992. Hierarchical Optimization: An Introduction. *Annals of Operations Research*, 34: 1–11.

Cai, H.; Xue, F.; Xu, L.; and Guo, L. 2022. TransMatting: Enhancing Transparent Objects Matting with Transformers. In *ECCV*, 253–269.

Cai, S.; Shu, Y.; and Wang, W. 2021. Dynamic Routing Networks. In *WACV*, 3588–3597.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive Differentiable Architecture Search: Bridging the Depth Gap Between Search and Evaluation. In *ICCV*, 1294–1303.

Chu, X.; and Zhang, B. 2021. Noisy Differentiable Architecture Search. In *BMVC*, 217.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An Overview of Bilevel Optimization. *Annals of Operations Research*, 153: 235–256.

Dai, Y.; Lu, H.; and Shen, C. 2021. Learning Affinity-Aware Upsampling for Deep Image Matting. In *CVPR*, 6837–6846.

Dai, Y.; Price, B.; Zhang, H.; and Shen, C. 2022. Boosting Robustness of Image Matting with Context Assembling and Strong Data Augmentation. In *CVPR*, 11697–11706.

Forte, M.; and Pitié, F. 2020. F, B, Alpha Matting. *arXiv preprint arXiv:2003.07711*.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In *ECCV*, 544–560.

Han, Y.; Liu, Z.; Yuan, Z.; Pu, Y.; Wang, C.; Song, S.; and Huang, G. 2024. Latency-aware Unified Dynamic Networks for Efficient Image Recognition. *IEEE TPAMI*.

Han, Y.; Yuan, Z.; Pu, Y.; Xue, C.; Song, S.; Sun, G.; and Huang, G. 2022. Latency-aware Spatial-wise Dynamic Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *NeurIPS*, 36845 – 36857.

He, K.; Rhemann, C.; Rother, C.; Tang, X.; and Sun, J. 2011. A Global Sampling Method for Alpha Matting. In *CVPR*, 2049–2056.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*, 1026–1034.

Hu, Y.; Lin, Y.; Wang, W.; Zhao, Y.; Wei, Y.; and Shi, H. 2023. Diffusion for Natural Image Matting. *arXiv preprint arXiv:2312.05915*.

Huang, G.; Chen, D.; Li, T.; Wu, F.; Van Der Maaten, L.; and Weinberger, K. Q. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *ICLR*.

Huijben, I. A.; Kool, W.; Paulus, M. B.; and Van Sloun, R. J. 2022. A Review of the Gumbel-max Trick and its Extensions for Discrete Stochasticity in Machine Learning. *IEEE TPAMI*, 45: 1353–1371.

Ke, Z.; Sun, J.; Li, K.; Yan, Q.; and Lau, R. W. 2022. MODNet: Real-Time Trimap-Free Portrait Matting via Objective Decomposition. In *AAAI*, 1140–1147.

Li, C.; Wang, G.; Wang, B.; Liang, X.; Li, Z.; and Chang, X. 2021. Dynamic Slimmable Network. In *CVPR*, 8603–7613.

Li, J.; Zhang, J.; and Tao, D. 2021. Deep Automatic Natural Image Matting. In *IJCAI*, 800–806.

Li, Y.; and Lu, H. 2020. Natural Image Matting via Guided Contextual Attention. In *AAAI*, 11450–11457.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *ICLR*.

Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Han, J. 2020a. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR*.

Liu, Q.; Lv, X.; Meng, Q.; Li, Z.; Lan, X.; Yang, S.; Zhang, S.; and Nie, L. 2024. Revisiting Context Aggregation for Image Matting. In *ICML*.

Liu, Q.; Xie, H.; Zhang, S.; Zhong, B.; and Ji, R. 2021a. Long-Range Feature Propagating for Natural Image Matting. In *ACM MM*, 526–534.

Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Deng, H.; and Ju, Q. 2020b. FastBERT: a Self-distilling BERT with Adaptive Inference Time. In *ACL*, 6035–6044.

Liu, Y.; Xie, J.; Shi, X.; Qiao, Y.; Huang, Y.; Tang, Y.; and Yang, X. 2021b. Tripartite Information Mining and Integration for Image Matting. In *ICCV*, 7535–7544.

Lu, H.; Dai, Y.; Shen, C.; and Xu, S. 2019. Indices Matter: Learning to Index for Deep Image Matting. In *ICCV*, 3265–3274.

Park, G.; Son, S.; Yoo, J.; Kim, S.; and Kwak, N. 2022. MatteFormer: Transformer-Based Image Matting via Prior-Tokens. In *CVPR*, 11686–11696.

Qiao, Y.; Liu, Y.; Yang, X.; Zhou, D.; Xu, M.; Zhang, Q.; and Wei, X. 2020. Attention-Guided Hierarchical Structure Aggregation for Image Matting. In *CVPR*, 13676–13685.

Ranjbar, M.; Abdelaziz, A.; and Jauhar, M. A. 2008. A Closed-Form Solution to Natural Image Matting. *IEEE TPAMI*, 30(2): 228–242.

Schwartz, R.; Stanovsky, G.; Swayamdipta, S.; Dodge, J.; and Smith, N. A. 2020. The Right Tool for the Job: Matching Model and Instance Complexities. In *ACL*, 6640–6651.

Sun, Y.; Tang, C.-K.; and Tai, Y.-W. 2021. Semantic Image Matting. In *CVPR*, 11120–11129.

Wang, J.; and Cohen, M. 2007. Optimized Color Sampling for Robust Matting. In *CVPR*, 1–8.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. E. 2018. SkipNet: Learning Dynamic Routing in Convolutional Networks. In *ECCV*, 420–436.

Xie, L.; and Yuille, A. 2017. Genetic CNN. In *ICCV*, 1388–1397.

Xin, J.; Tang, R.; Lee, J.; Yu, Y.; and Lin, J. 2020. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In *ACL*.

Xu, N.; Price, B.; Cohen, S.; and Huang, T. 2017. Deep Image Matting. In *CVPR*, 311–320.

Xu, Y.; He, S.; Shao, W.; Wong, K.-Y. K.; Qiao, Y.; and Luo, P. 2023. DiffusionMat: Alpha Matting as Sequential Refinement Learning. *arXiv preprint arXiv:2311.13535*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. Partial Channel Connections for Memory-Efficient Differentiable Architecture Search. In *ICLR*.

Yang, L.; Han, Y.; Chen, X.; Song, S.; Dai, J.; and Huang, G. 2020. Resolution Adaptive Networks for Efficient Inference. In *CVPR*, 2366–2375.

Yao, J.; Wang, X.; Yang, S.; and Wang, B. 2024. ViTMatte: Boosting Image Matting with Pre-trained Plain Vision Transformers. *Information Fusion*, 103: 102091.

Yu, H.; Li, H.; Shi, H.; Huang, T. S.; and Hua, G. 2021. Any-Precision Deep Neural Networks. In *AAAI*, 10763–10771.

Yu, H.; Xu, N.; Huang, Z.; Zhou, Y.; and Shi, H. 2021. High-Resolution Deep Image Matting. In *AAAI*, 3217–3224.

Yu, Q.; Zhang, J.; Zhang, H.; Wang, Y.; Lin, Z.; Xu, N.; Bai, Y.; and Yuille, A. 2021. Mask Guided Matting via Progressive Refinement Network. In *CVPR*, 1154–1163.

Zhao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J. 2017. Pyramid Scene Parsing Network. In *CVPR*, 6230–6239.

Zhou, W.; Xu, C.; Ge, T.; McAuley, J.; Xu, K.; and Wei, F. 2020. BERT Loses Patience: Fast and Robust Inference with Early Exit. In *NeurIPS*, 18330–18341.