HAZEMT-DCR: DEMAND-AWARE COUNTERFACTUAL EXPERT ROUTING FOR INCREMENTAL MULTI-TASK PREDICTION IN HAZY SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing multi-task perception methods for hazy scenes usually adopt a two-stage "dehazing-then-prediction" paradigm. However, dehazing objectives for visual restoration or pixel reconstruction may not match downstream task requirements. A unified shared representation may also fail to capture the different semantic, structural, and geometric cues required by different tasks. To address these issues, we propose HazeMT-DCR, a demand-aware counterfactual expert routing framework for multi-task prediction in hazy scenes. We reformulate hazy multi-task perception as shared functional reuse and dynamic expert composition conditioned on task demands. Without explicit clear-image restoration, HazeMT-DCR calibrates multi-scale visual features using latent degradation states and conditions shared functional experts on task demands, degradation states, and image content. Each task learns an independent reinforcement learning routing policy to dynamically combine the shared experts. To improve routing reliability, we introduce a task-demand intervention counterfactual regret mechanism. Expert removal identifies redundant or negatively contributing experts, while task-demand replacement checks whether expert responses depend on the correct task demand. The resulting counterfactual signals are used to improve expert routing. HazeMT-DCR also decouples shared modules from task-specific parameters, allowing new tasks to be adapted by training only task-specific routing and output modules without retraining the full model. We evaluate HazeMT-DCR on four dense prediction tasks on hazy PASCAL-Context. Experiments show that HazeMT-DCR outperforms existing parameter-efficient multi-task adaptation methods while requiring only a small number of trainable parameters, and supports efficient task expansion.

1 INTRODUCTION

Multi-task dense prediction aims to perform multiple visual tasks within a unified model, such as semantic segmentation, human part parsing, saliency detection, and surface normal estimation. Sharing representations across tasks can reduce redundant computation and exploit complementary information. When haze is present, however, image contrast, textures, and structures are degraded, making reliable multi-task prediction more difficult. Meanwhile, different tasks rely on different semantic, structural, and geometric cues. Therefore, the key challenge in hazy scenes is to mitigate haze-induced degradation while preserving the task-specific information required by different tasks.

Existing studies mainly address this problem from two directions: image restoration and multi-task adaptation. Traditional approaches first restore a clear image and then perform downstream prediction, as shown in Fig. 1(a), but restoration objectives based on visual quality or pixel reconstruction may not match task-specific needs. Recent methods (Zhang et al., 2026a) use text instructions or task feedback to guide dehazing, yet still rely on the restored image as an intermediate representation rather than directly learning task-specific features. Parameter-efficient multi-task methods (Agiza et al., 2024; Mantri et al., 2025; Baek et al., 2025; Yao et al., 2026b) reduce adaptation costs through shared backbones and lightweight modules, but are mainly designed for clear scenes. In hazy scenes, multi-task prediction must jointly handle degradation, task differences, and negative transfer. Moreover, dynamic expert routing based only on the final task loss cannot easily identify harmful experts or task-requirement mismatches. Therefore, a more reliable routing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

mechanism is needed to adapt expert combinations to task requirements and diagnose undesirable routing behavior. As illustrated in Fig. 1(b), these limitations motivate a direct task-conditioned prediction paradigm that avoids explicit restoration and supports incremental task expansion.

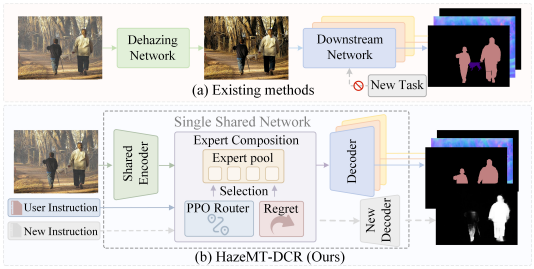


Figure 1: Comparison of traditional paradigms and our method in hazy scenes. Our method directly predicts multiple tasks without explicit restoration and supports incremental expansion.

To this end, we reformulate hazy multi-task learning as shared functional reuse and dynamic expert composition conditioned on task demands, and propose HazeMT-DCR, a task-demand-driven counterfactual expert routing framework. Instead of explicitly restoring clear images, HazeMT-DCR uses latent degradation states to calibrate multi-scale visual features. It further combines textual task instructions, degradation states, and image content to generate task-demand control representations, which modulate shared functional experts at both channel and spatial levels. In this way, the experts serve as reusable functional bases rather than being tied to fixed tasks. For each task, an independent reinforcement learning policy dynamically combines these experts according to the input and task requirements.

Since dynamic routing alone cannot ensure reliable expert selection, we further diagnose routing behavior from expert contribution and task-demand consistency. We introduce a task-demand intervention counterfactual regret mechanism, where expert-removal counterfactuals identify redundant or harmful experts, and task-demand-replacement counterfactuals test whether expert responses depend on the correct task demand. The resulting regret signals guide routing and reduce harmful expert combinations and task-demand mismatches. Furthermore, HazeMT-DCR also decouples shared modules from task-specific parameters. The haze-aware shared encoder, task-demand control module, and functional expert library are shared across tasks, while static routing parameters, actor–critic networks, and output adapters are task-specific. When a new task is added, the shared modules and existing task parameters remain frozen, and only the new task-specific modules are trained. This enables low-cost task expansion without retraining the full model while preserving existing task paths. We evaluate four dense prediction tasks on hazy PASCAL-Context. Results show that HazeMT-DCR achieves strong overall multi-task performance with low trainable-parameter overhead and good task scalability. The main contributions of this paper are as follows:

- We re-examine multi-task dense prediction in hazy scenes and reformulate it from "image restoration followed by task execution" into shared functional reuse and task-conditioned expert composition. To support this, we design a haze-aware shared encoder and task-conditioned functional experts that jointly model degradation, task demands, and image content, enabling task-specific feature adaptation without explicit clear-image restoration.
- We propose a task-demand-driven reinforcement learning routing framework with a counterfactual regret mechanism. To improve routing reliability, we design expert removal and task-demand replacement to diagnose harmful expert contributions and task-demand mismatches, and use the resulting feedback to correct unreasonable expert combinations.
- We design an incremental task adaptation scheme that separates reusable shared modules from task-specific parameters. Under this design, new tasks are added by training only their routing parameters, actor–critic policy network, and output adapter, while shared modules and existing task parameters remain frozen. This avoids full-model retraining and reduces the cost of task expansion.

2 Related Work

2.1 Image Dehazing for Downstream Visual Tasks

Single-image dehazing aims to recover a clear image from an image degraded by atmospheric scattering. Existing methods (He et al., 2011; Cai et al., 2016; Li et al., 2017; Liu et al., 2019; Dong et al., 2020; Guo et al., 2022; Wu et al., 2023) typically use physical priors, data-driven networks, or both. These methods mainly optimize restoration quality, such as image clarity and structural

fidelity. However, better restoration quality does not always lead to better downstream task performance because different tasks rely on different visual cues. To better connect dehazing with downstream tasks, some studies use task feedback to guide the dehazing process. AOD-Net (Li et al., 2017) is cascaded with Faster R-CNN to improve object detection in hazy scenes. UDnD (Zhang et al., 2020) further uses haze density information to jointly model dehazing and object detection. Adaptive Dynamic Dehazing (Zhang et al., 2026a) introduces task feedback and textual instructions so that dehazing can respond to different task requirements. Although these methods improve the connection between dehazing and downstream tasks, they still use the restored image as an intermediate representation for prediction. This introduces extra image reconstruction cost and may not directly meet the different semantic, boundary, and geometric needs of downstream tasks.

2.2 Parameter-Efficient Multi-Task Dense Prediction

Multi-task dense prediction aims to perform multiple pixel-level prediction tasks within a unified model. A key challenge is to exploit complementary information across tasks while avoiding task competition caused by excessive sharing. PAD-Net (Xu et al., 2018) promotes information transfer through intermediate task prediction and multi-modal distillation. MTI-Net (Vandenhende et al., 2020) models task interactions across multiple scales. InvPT (Ye & Xu, 2022) uses an inverted pyramid Transformer to capture global dependencies across spatial locations and tasks. PGT (Lu et al., 2024) modulates a shared Transformer with task prompts. MLoRE (Yang et al., 2024) introduces low-rank experts and shared paths to learn shared and task-specific representations with parameter overhead. Although these methods reduce task competition, they usually require joint training of large multi-task networks, resulting in high training and task-expansion costs.

Parameter-efficient multi-task fine-tuning reduces this cost by freezing most pre-trained parameters and updating only a small set of additional modules. Adapter (Houlsby et al., 2019), VPT (Jia et al., 2022), and LoRA (Hu et al., 2022) use lightweight adapters, learnable prompts, and low-rank updates, respectively, for efficient downstream adaptation. Polyhistor (Liu et al., 2022) uses a decomposed hypernetwork to generate task-specific adaptation parameters and reduce parameter growth. MTLoRA (Agiza et al., 2024) separates low-rank modules into task-agnostic and task-specific components to balance shared and task-specific information. DiTASK (Mantri et al., 2025) reduces task competition in shared low-rank spaces through task-dependent transformations. TADFormer (Baek et al., 2025) combines efficient prompting with dynamic task filters to extract task-relevant features from the input context. TDSS (Yao et al., 2026a) improves multi-task adaptation and task scalability using skill experts, dynamic task gating, and cross-task interactions. However, most of these methods are designed for clear scenes and do not explicitly consider how haze changes the feature requirements of different tasks.

3 Methodology

3.1 Overview

Figure 2 shows the overall framework of HazeMT-DCR. Rather than restoring a clear image, the model directly performs task-conditioned multi-task prediction from hazy inputs. It consists of a Haze-Aware Shared Encoder (HASE), an Instruction-Conditioned Functional Expert Library (ICFE-L), a Text-Instruction-Driven Reinforcement Learning Expert Router (TRLR), and a Task-Requirement-Intervention Counterfactual Regret (TICR) module. HASE calibrates and fuses multi-scale features using the latent degradation state. In ICFE-L, the Task-Requirement Control module(TCM) combines text instructions, degradation states, and shared visual features to guide functional experts. TRLR dynamically combines expert outputs for each activated task and sends the fused features to the corresponding output head. TICR uses expert removal and task-requirement replacement to detect harmful expert contributions and task-requirement mismatches, further improving expert learning and routing.

3.2 Haze-Aware Shared Encoder

The goal of HASE is to extract shared visual representations for downstream tasks without reconstructing clear images. Given a hazy image $\mathbf{P}$, the model obtains a degradation state vector $\mathbf{h}$ through a latent degradation state estimation network $\mathcal{H}_\eta(\cdot)$. Here, $\mathbf{h}$ does not represent physical haze density or atmospheric light. Instead, it is a data-driven latent descriptor of image degrada-

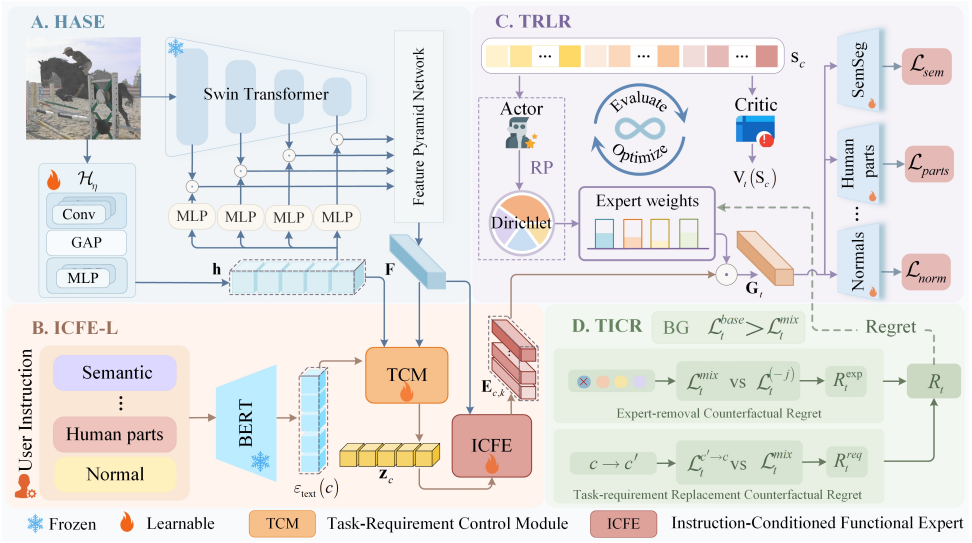


Figure 2: The overall framework of HazeMT-DCR. Haze-aware shared encoding and task-instruction-conditioned functional experts extract task-specific features, while task-level reinforcement learning routing dynamically combines the experts. Counterfactual regret refines routing during training, enabling multi-task dense prediction without explicit dehazing.

tion that provides global degradation information for multi-scale features. $\mathcal{H}_\eta(\cdot)$ consists of three convolutional blocks, global pooling, and a two-layer MLP. Each convolutional block contains a 3×3 convolution, a normalization layer, and a GELU activation. The hazy image $\mathbf{P}$ is also fed into a pre-trained Swin Transformer to obtain multi-level features $\{\mathbf{F}_l\}_{l=1}^{L}$. The degradation state $\mathbf{h}$ is used to generate channel modulation parameters for each feature level $[\boldsymbol{\gamma}_l^h, \boldsymbol{\beta}_l^h] = \mathrm{MLP}_l^h(\mathbf{h})$, where $\boldsymbol{\gamma}_l^h, \boldsymbol{\beta}_l^h \in \mathbb{R}^{C_l}$. The feature $\mathbf{F}_l$ is then calibrated as $\bar{\mathbf{F}}_l = (1 + \boldsymbol{\gamma}_l^h) \odot \mathbf{F}_l + \boldsymbol{\beta}_l^h$, where $\odot$ denotes channel-wise multiplication. The shared representation is obtained by $\mathbf{F} = \mathrm{FPN}(\{\bar{\mathbf{F}}_l\}_{l=1}^{L})$, where $\mathrm{FPN}(\cdot)$ denotes the Feature Pyramid Network.

3.3 Instruction-Conditioned Functional Expert Library

Relying only on $\mathbf{F}$ often leads to compromised representations across tasks, making it difficult to satisfy different requirements for semantics, structure, boundaries, and geometry. To address this, we construct an ICFE-L. Unlike standard MoE experts with relatively fixed transformations, we model the experts as a set of conditional functional transformation bases. The experts can therefore adapt their feature transformations to task instructions, degradation states, and image content.

Task-Requirement Control Module: As illustrated in Fig. 2, we first encode the textual task instruction c using a frozen BERT text encoder. A learnable task adapter $A_q(\cdot)$ in TCM maps the text embedding to a task requirement representation: $\mathbf{e}_c = A_q(\mathcal{E}_{\text{text}}(c))$, where $\mathbf{e}_c \in \mathbb{R}^{d_e}$. Unlike assigning a fixed learnable vector to each task, this design generates instruction-dependent task representations. We then construct the task requirement control vector $\mathbf{z}_c$ by combining $\mathbf{e}_c$, the degradation state $\mathbf{h}$, and the shared visual feature $\mathbf{F}$:

$$\mathbf{z}_c = \Psi\left(\mathrm{Concat}\left(\mathbf{e}_c, \mathbf{h}, \mathrm{GAP}(\mathbf{F}), \mathrm{GMP}(\mathbf{F})\right)\right), \tag{1}$$

where $\Psi(\cdot)$ consists of a two-layer MLP and a LayerNorm layer, and $\mathrm{GAP}(\cdot)$ and $\mathrm{GMP}(\cdot)$ denote global average pooling and global max pooling, respectively. The vector $\mathbf{z}_c$ serves as the shared control variable for expert modulation and routing. The detailed network architecture of TCM is shown in Fig. 3 (a) in the appendix.

Construction of the ICFE-L: ICFE-L contains K shared functional experts that generate diverse task-conditioned features from the shared feature $\mathbf{F}$. As illustrated in Fig. 3 (b), the k-th expert first applies an expert-specific transformation to $\mathbf{F}$:

$$\mathbf{U}_k = \mathrm{Conv}_{1\times1}^{up}\left(\sigma\left(\mathrm{DWConv}_{3\times3}\left(\mathrm{Conv}_{1\times1}^{down}(\mathbf{F})\right)\right)\right), \tag{2}$$

where $\sigma(\cdot)$ denotes the GELU activation. The task requirement control vector $\mathbf{z}_c$ is then used to generate channel modulation parameters: $[\boldsymbol{\gamma}_{c,k}, \boldsymbol{\beta}_{c,k}] = \mathrm{MLP}_k^m(\mathbf{z}_c)$, where $\boldsymbol{\gamma}_{c,k}$ and $\boldsymbol{\beta}_{c,k}$ denote the channel-wise scaling and shifting parameters, respectively. To further adapt the expert response to local image content, we generate a spatial mask from $\mathbf{U}_k$ and $\mathbf{z}_c$:

$$\mathbf{M}_{c,k} = \mathrm{Sigmoid}\left(\mathrm{Conv}_{1\times1}\left(\mathrm{Concat}\left(\mathbf{U}_k, \mathrm{Broadcast}(\mathbf{z}_c)\right)\right)\right), \tag{3}$$

where $\mathrm{Broadcast}(\mathbf{z}_c)$ expands $\mathbf{z}_c$ to the spatial resolution of $\mathbf{U}_k$, and $\mathbf{M}_{c,k}$ controls the spatial response of the k-th expert. The task-conditioned residual is defined as

$$\Delta\mathbf{E}_{c,k} = \mathbf{M}_{c,k} \odot \left[(1 + \boldsymbol{\gamma}_{c,k}) \odot \mathbf{U}_k + \boldsymbol{\beta}_{c,k}\right], \tag{4}$$

and the final expert output is $\mathbf{E}_{c,k} = \mathbf{F} + \Delta\mathbf{E}_{c,k}$. Channel modulation adjusts feature responses, while the spatial mask controls where they are applied. In this way, each expert adapts its response according to the task requirement, degradation state, and image content.

Expert Separation Constraint: To prevent different experts from learning highly similar transformations, we design an expert separation constraint. For an instruction c, each expert residual is first globally pooled and normalized:

$$\bar{\Delta\mathbf{E}}_{c,k} = \frac{\mathrm{GAP}\left(\Delta\mathbf{E}_{c,k}\right)}{\left\|\mathrm{GAP}\left(\Delta\mathbf{E}_{c,k}\right)\right\|_2 + \epsilon}. \tag{5}$$

Let $\mathcal{C}_B$ denote the set of textual task instructions in the current batch, and $|\mathcal{C}_B|$ its size. The expert residual separation loss is defined as

$$\mathcal{L}_{div} = \frac{1}{|\mathcal{C}_B|} \sum_{c\in\mathcal{C}_B} \frac{1}{K(K-1)} \sum_{\substack{i,j=1 \\ i\neq j}}^{K} \left|\bar{\Delta\mathbf{E}}_{c,i}^{\top} \bar{\Delta\mathbf{E}}_{c,j}\right|. \tag{6}$$

We also reduce the correlation between task requirement control vectors from different instructions:

$$\mathcal{L}_{ctrl} = \frac{1}{|\mathcal{C}_B|\left(|\mathcal{C}_B| - 1\right)} \sum_{\substack{c,c'\in\mathcal{C}_B \\ c'\neq c}} \left|\frac{\mathbf{z}_c^{\top}\mathbf{z}_{c'}}{\|\mathbf{z}_c\|_2 \|\mathbf{z}_{c'}\|_2 + \epsilon}\right|. \tag{7}$$

The final expert separation constraint is $\mathcal{L}_{sep} = \frac{1}{2}\left(\mathcal{L}_{div} + \mathcal{L}_{ctrl}\right)$.

3.4 Text-Instruction-Driven Reinforcement Learning-Based Expert Routing

For each task $t \in S(c)$ activated by the text instruction c, the model needs to select an appropriate combination of experts from the functional expert library. Different tasks rely on different visual cues in hazy scenes. We therefore adopt task-level expert routing, where all activated tasks share the same instruction-conditioned expert features, while each task learns its own fusion weights. To adapt the expert weights to both the input and task requirement, we formulate expert routing as a single-step reinforcement learning problem, which also enables policy exploration during training. The state is defined as

$$\mathbf{s}_c = \mathrm{StopGrad}\left(\mathrm{Concat}\left(\mathbf{z}_c, \mathrm{GAP}(\mathbf{F}), \mathrm{GMP}(\mathbf{F})\right)\right), \tag{8}$$

where $\mathrm{StopGrad}(\cdot)$ prevents gradients from propagating to $\mathbf{F}$ and $\mathbf{z}_c$ through the routing state. The action for task t is the expert weight vector

$$\mathbf{w}_t = [w_{t,1}, w_{t,2}, \ldots, w_{t,K}], \quad w_{t,k} \geq 0, \quad \sum_{k=1}^{K} w_{t,k} = 1, \tag{9}$$

where $w_{t,k}$ denotes the weight of the k-th expert for task t.

Each task has an independent actor–critic pair. The actor $f_\theta^t(\cdot)$ maps $\mathbf{s}_c$ to the concentration parameters of a Dirichlet distribution: $\boldsymbol{\alpha}_t = \mathrm{Softplus}\left(f_\theta^t(\mathbf{s}_c)\right) + \epsilon$, where $\boldsymbol{\alpha}_t = [\alpha_{t,1}, \ldots, \alpha_{t,K}]$ and $\alpha_{t,k} > 0$. The expert weights are sampled as $\mathbf{w}_t \sim \mathrm{Dirichlet}(\boldsymbol{\alpha}_t)$ using reparameterized sampling. This allows gradients from the task and counterfactual objectives to propagate through $\mathbf{w}_t$ to the actor network. The critic estimates the value of the current state. During training, sampling is used for policy exploration. During inference, we use the expectation of the Dirichlet distribution:

$w_{t,k}^{test} = \alpha_{t,k} / \sum_{j=1}^{K} \alpha_{t,j}$. The fused feature and prediction for task t are $\mathbf{G}_t = \sum_{k=1}^{K} w_{t,k} \mathbf{E}_{c,k}$ and $\mathbf{Y}_t = \mathcal{D}_t^{o(c,t)}(\mathbf{G}_t)$, respectively. Here, $\mathcal{D}_t^{o(c,t)}(\cdot)$ denotes the output adaptation head for task t, and $o(c,t)$ specifies its output format.

The task loss under dynamic routing is defined as $\mathcal{L}_t^{mix} = \mathcal{L}_t\left(\mathbf{Y}_t, \mathbf{Y}_t^{gt}\right)$, where $\mathbf{Y}_t^{gt}$ is the ground-truth annotation. To measure the gain of dynamic routing, we use the static routing learned during warm-up as the baseline. Each task has learnable static routing parameters $\mathbf{a}_t \in \mathbb{R}^K$, initialized to zero, with $\bar{\mathbf{w}}_t = \mathrm{Softmax}(\mathbf{a}_t)$. During warm-up, $\mathbf{a}_t$ is optimized by the task prediction loss. After warm-up, $\bar{\mathbf{w}}_t$ is frozen and used as the static baseline. The corresponding baseline feature is $\mathbf{G}_t^{base} = \sum_{k=1}^{K} \bar{w}_{t,k} \mathbf{E}_{c,k}$. The baseline and dynamic branches share the same expert features and output adaptation head, differing only in their fusion weights. The baseline prediction and loss are $\mathbf{Y}_t^{base} = \mathcal{D}_t^{o(c,t)}\left(\mathbf{G}_t^{base}\right)$ and $\mathcal{L}_t^{base} = \mathcal{L}_t\left(\mathbf{Y}_t^{base}, \mathbf{Y}_t^{gt}\right)$, respectively. The relative gain over the static baseline is defined as

$$\Delta_t^{base} = \frac{\mathcal{L}_t^{base} - \mathcal{L}_t^{mix}}{\mathcal{L}_t^{base} + \epsilon}. \tag{10}$$

A positive Δ_t^{base} indicates that dynamic routing achieves a lower task loss than the static routing learned during warm-up.

3.5 Task-Requirement Intervention Counterfactual Regret

Task-level performance feedback indicates whether an expert combination improves task performance, but cannot identify harmful experts or task-requirement mismatches. To address this, we propose TICR, which uses expert-removal and task-requirement-replacement counterfactuals to generate regret signals for refining expert learning and routing. Given the current expert combination $\mathbf{G}_t$, we construct an expert-removal counterfactual by removing expert j and normalizing the remaining weights:

$$w_{t,k}^{(-j)} = \frac{w_{t,k} \mathrm{I}(k \neq j)}{\sum_{m=1}^{K} w_{t,m} \mathrm{I}(m \neq j) + \epsilon}. \tag{11}$$

The corresponding counterfactual features, predictions, and losses are given by

$$\mathbf{G}_t^{(-j)} = \sum_{k=1}^{K} w_{t,k}^{(-j)} \mathbf{E}_{c,k}, \mathbf{Y}_t^{(-j)} = \mathcal{D}_t^{o(c,t)}\left(\mathbf{G}_t^{(-j)}\right), \mathcal{L}_t^{(-j)} = \mathcal{L}_t\left(\mathbf{Y}_t^{(-j)}, \mathbf{Y}_t^{gt}\right). \tag{12}$$

The expert removal regret is defined as

$$R_t^{exp} = \frac{1}{K} \sum_{j=1}^{K} \mathrm{I}(\Delta_t^{base} > 0) \max\left(0, \frac{\mathcal{L}_t^{mix} - \mathcal{L}_t^{(-j)}}{|\mathcal{L}_t^{mix}| + \epsilon}\right). \tag{13}$$

If removing an expert reduces the task loss, that expert may be redundant or harmful to the current expert combination. Minimizing R_t^{exp} encourages the full expert combination to perform no worse than its expert-removed counterfactuals. The gate $\mathrm{I}(\Delta_t^{base} > 0)$ activates this constraint only when dynamic routing outperforms the static baseline, avoiding unreliable counterfactual supervision before the dynamic routing establishes an improvement over the baseline.

Expert removal alone cannot determine whether the expert responses are properly conditioned on the current task requirement. We therefore construct a second counterfactual by replacing only the textual task representation. Let $\mathcal{C}_{tr}$ denote the set of textual task instructions available during training. For the current task t, we randomly sample another instruction $c' \neq c$ whose task requirement does not include t. We then replace $\mathbf{e}_c$ with $\mathbf{e}_{c'}$, while keeping the current degradation state $\mathbf{h}$ and shared visual feature $\mathbf{F}$ unchanged. The counterfactual task requirement control vector is constructed as

$$\mathbf{z}_{c' \to c} = \Psi\left(\mathrm{Concat}\left(\mathbf{e}_{c'}, \mathbf{h}, \mathrm{GAP}(\mathbf{F}), \mathrm{GMP}(\mathbf{F})\right)\right), \quad c' \neq c. \tag{14}$$

The corresponding counterfactual expert output is $\mathbf{E}_k^{c' \to c} = \mathcal{M}_k\left(\mathbf{F}, \mathbf{z}_{c' \to c}\right)$, where $\mathcal{M}_k(\cdot,\cdot)$ denotes the task-conditioned transformation implemented by the k-th functional expert defined above. Using the same expert fusion weights $\mathbf{w}_t$, the feature after task-requirement replacement becomes $\mathbf{G}_t^{c' \to c} = \sum_{k=1}^{K} w_{t,k} \mathbf{E}_k^{c' \to c}$. The corresponding prediction and loss are given by

$$\mathbf{Y}_t^{c' \to c} = \mathcal{D}_t^{o(c,t)}\left(\mathbf{G}_t^{c' \to c}\right), \mathcal{L}_t^{c' \to c} = \mathcal{L}_t\left(\mathbf{Y}_t^{c' \to c}, \mathbf{Y}_t^{\mathrm{gt}}\right). \tag{15}$$

This counterfactual changes only the textual task requirement used for expert modulation. The input image, degradation state, shared visual features, expert fusion weights, and output adaptation head remain the same. This isolates the effect of task-requirement replacement. If an incorrect task requirement produces a lower loss than the correct one, the expert responses are not sufficiently aligned with the current task requirement. We therefore define the task-requirement regret as

$$R_t^{req} = \mathrm{I}(\Delta_t^{base} > 0) \max\left(0, \frac{\mathcal{L}_t^{mix} - \mathcal{L}_t^{c' \to c}}{|\mathcal{L}_t^{mix}| + \epsilon}\right). \tag{16}$$

R_t^{req} penalizes cases where a mismatched task requirement yields a lower loss than the correct one. Minimizing it encourages expert responses under the correct requirement to perform better. The counterfactual regret for task t is $R_t = \frac{1}{2}\left(R_t^{exp} + R_t^{req}\right)$, and the counterfactual regret loss is $\mathcal{L}_{cf} = \frac{1}{|S(c)|}\sum_{t \in S(c)} R_t$. $\mathcal{L}_{cf}$ is optimized with the task loss and can backpropagate to the actor through the reparameterized expert weights. It encourages the expert combination to outperform expert-removed alternatives and correct task requirements to outperform mismatched ones.

Regret-Guided Routing Optimization: The reinforcement learning reward for task t is defined as $r_t = \Delta_t^{base} - \mathrm{StopGrad}(R_t)$. Here, R_t is detached in the reinforcement learning reward to prevent gradients from propagating through the reward construction. Meanwhile, $\mathcal{L}_{cf}$ remains differentiable and can directly optimize the expert modules and actor network through the reparameterized expert weights. For task t, the critic network V_t takes the state $\mathbf{s}_c$ as input and estimates its value $V_t(\mathbf{s}_c)$. The advantage is defined as $A_t = \mathrm{StopGrad}\left(r_t - V_t(\mathbf{s}_c)\right)$. The actor network $f_\theta^t(\cdot)$ generates the concentration parameters of the Dirichlet policy: $\boldsymbol{\alpha}_t = \mathrm{Softplus}\left(f_\theta^t(\mathbf{s}_c)\right) + \epsilon$, where $\boldsymbol{\alpha}_t = [\alpha_{t,1}, \ldots, \alpha_{t,K}]$ and $\alpha_{t,k} > 0$. The corresponding policy density is denoted by $\pi_\theta^t(\mathbf{w}_t|\mathbf{s}_c)$. Proximal Policy Optimization (PPO) (Schulman et al., 2017) then measures the change from the old policy using the probability ratio:

$$\rho_t(\theta) = \frac{\pi_\theta^t(\mathbf{w}_t|\mathbf{s}_c)}{\pi_{\theta_{\mathrm{old}}}^t(\mathbf{w}_t|\mathbf{s}_c)}. \tag{17}$$

The PPO policy loss is

$$\mathcal{L}_{ppo}^t = -\mathbb{E}\left[\min\left(\rho_t(\theta)A_t, \mathrm{clip}\left(\rho_t(\theta), 1-\epsilon_p, 1+\epsilon_p\right)A_t\right)\right]. \tag{18}$$

Here, $\pi_{\theta_{\mathrm{old}}}^t$ denotes the behavior policy used to generate the expert-weight samples for the PPO update, and ϵ_p is the PPO clipping coefficient. The function $\mathrm{clip}(\cdot)$ restricts the probability ratio to $[1-\epsilon_p, 1+\epsilon_p]$, and $\mathbb{E}[\cdot]$ is taken over the state-action samples in the batch. In the PPO likelihood ratio, the sampled $\mathbf{w}_t$ is treated as the action generated by the behavior policy, while reparameterized gradients are retained for the task and counterfactual objectives. The critic is optimized using $\mathcal{L}_{value}^t = \left(V_t(\mathbf{s}_c) - \mathrm{StopGrad}(r_t)\right)^2$. The overall routing loss is $\mathcal{L}_{route} = \sum_{t \in S(c)}\left(\mathcal{L}_{ppo}^t + \mathcal{L}_{value}^t\right)$. No entropy regularization is used. Policy exploration is achieved through reparameterized sampling of $\mathbf{w}_t$ from the Dirichlet distribution during training.

3.6 Task Prediction Loss

To avoid manually setting task-specific loss weights, we normalize each task loss using an exponential moving average (EMA). Let n denote the current training iteration. When task t is activated, its EMA loss is updated as $\bar{\mathcal{L}}_t^{(n)} = \mu\bar{\mathcal{L}}_t^{(n-1)} + (1-\mu)\mathcal{L}_t^{(n)}$, where $\bar{\mathcal{L}}_t^{(n)}$ denotes the EMA loss of task t at iteration n, $\mathcal{L}_t^{(n)}$ denotes its current supervised loss, and μ is the smoothing coefficient, which is set to 0.99. For simplicity, we use $\bar{\mathcal{L}}_t$ to denote the EMA value at the current iteration. For the task set $S(c)$ activated by instruction c, the task prediction loss is defined as

$$\mathcal{L}_{task} = \sum_{t \in S(c)} \frac{\mathcal{L}_t\left(\mathbf{Y}_t, \mathbf{Y}_t^{gt}\right)}{\mathrm{StopGrad}\left(\bar{\mathcal{L}}_t\right) + \epsilon}. \tag{19}$$

During the warm-up stage, $\mathbf{Y}_t$ in $\mathcal{L}_{task}$ is replaced by the static-routing prediction $\mathbf{Y}_t^{base}$. During joint routing optimization, the dynamically routed prediction $\mathbf{Y}_t$ is used. Here, ϵ is a small constant for numerical stability. This normalization reduces the effect of different loss scales and convergence rates across tasks, avoiding manual task-specific loss weighting. The overall training objective is

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda_r\mathcal{L}_{route} + \lambda_{cf}\mathcal{L}_{cf} + \lambda_s\mathcal{L}_{sep}, \tag{20}$$

where λ_r, λ_{cf}, and λ_s are the corresponding loss weights. For new-task expansion, details of the incremental adaptation procedure are provided in the appendix.

Table 1: Comparison of multi-task performance and efficiency with existing methods. The best and second-best results are highlighted in **pink** and **blue**, respectively.

Methods	Comparison Group	SemSeg mIoU↑	Parts mIoU↑	Saliency mIoU↑	Normal mErr↓	Δ_m (%)↑	#TP (M)↓	GFLOPs↓
STL	Setting 1	63.89	57.99	**64.58**	18.02	0.00	114.0	176.39
Hard Sharing		63.02	52.11	56.82	19.86	-8.43	31.12	**57.90**
Hazy+TGLoRA	Setting 2	65.60	58.58	63.67	18.73	-0.41	7.53	113.40
Restormer+TGLoRA		64.53	57.60	63.15	18.61	-1.29	7.64	161.92
BiLaLoRA+TGLoRA		61.07	52.73	59.24	19.42	-7.38	14.21	175.69
MTI-Net	Setting 3	69.93	58.56	64.01	18.72	+1.41	14.24	60.37
MTLoRA		67.87	57.48	62.91	18.64	-0.17	8.34	62.28
DiTASK		67.82	55.82	61.87	19.13	-1.99	**1.61**	325.03
TADFormer		69.15	58.45	63.39	17.90	+1.96	7.38	77.14
TGLoRA		68.43	58.11	63.79	18.35	+1.06	7.53	113.40
DeMT		58.91	52.10	60.52	18.47	-6.69	5.50	207.18
DeGMix		69.89	56.84	61.50	17.79	+0.98	11.57	96.36
FAAR		66.36	56.14	62.64	18.17	-0.79	6.70	127.22
Ours	–	**70.64**	**58.86**	64.53	**17.32**	**+3.96**	5.08	58.79

4 Experiments

4.1 Experimental Setup

We build a hazy multi-task dataset based on PASCAL-Context (Everingham et al., 2010), with 4,998 training and 5,105 validation images annotated for semantic segmentation, human part parsing, saliency detection, and surface normal estimation. Hazy images are synthesized from clear ones via the atmospheric scattering model. SemSeg, Parts, and Saliency are evaluated using mean Intersection over Union (mIoU), while Normal uses Mean Angular Error (mErr). We further use the average multi-task performance change Δ_m relative to independently trained single-task learning (STL) models to evaluate overall performance. Let M_t and M_t^{STL} denote the metrics of the multi-task and corresponding single-task models for task t, respectively. Δ_m is defined as

$$\Delta_m = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (-1)^{l_t} \frac{M_t^{STL} - M_t}{M_t^{STL}} \times 100 \tag{21}$$

Here, $|\mathcal{T}| = 4$, and $l_t = 1$ for higher-is-better metrics and $l_t = 0$ otherwise. A positive Δ_m indicates an overall improvement over STL, while a negative value indicates negative transfer.

4.2 Comparison with Existing Methods

We compare HazeMT-DCR with existing methods under three settings: single- and multi-task baselines (Setting 1), dehazing-then-prediction baselines (Setting 2), and general multi-task methods (Setting 3). Table 1 summarizes the results. For a fair comparison, we prioritize methods with public code that can be retrained under our four-task setting. All reproducible methods use the same data split, hazy inputs, annotations, input resolution, and evaluation metrics. Swin-Tiny (Liu et al., 2021) is adopted as the visual backbone whenever applicable. In Setting 1, we compare with STL and Hard Sharing to evaluate task sharing. Hard Sharing suffers from negative transfer with $\Delta_m = -8.43\%$, while HazeMT-DCR improves it to $+3.96\%$ with only 5.08M trainable parameters. In Setting 2, we compare with dehazing-then-prediction baselines, including Restormer (Zamir et al., 2022)+TGLoRA (Gangwar et al., 2026) and BiLaLoRA (Zhang et al., 2026b)+TGLoRA, as well as Hazy+TGLoRA. The results show that explicit restoration does not necessarily benefit downstream prediction, whereas direct task-conditioned modeling achieves better overall performance with lower computational cost. In Setting 3, we compare with general multi-task methods, including MTI-Net (Vandenhende et al., 2020), MTLoRA (Agiza et al., 2024), DiTASK (Mantri et al., 2025), TADFormer (Baek et al., 2025), TGLoRA, DeMT (Xu et al., 2023), DeGMix (Xu et al., 2026), and FAAR (Fontana et al., 2026). Among these multi-task methods, HazeMT-DCR achieves the best performance on all four tasks and the highest Δ_m of $+3.96\%$, using 5.08M trainable parameters and 58.79 GFLOPs.

4.3 Comparison of Task Scalability

To further evaluate task scalability, we compare HazeMT-DCR with existing multi-task methods under two incremental expansion settings. In the first setting, SemSeg and Saliency are treated as old tasks, while Parts and Normal are added as new tasks. In the second setting, SemSeg, Parts,

Table 2: Comparison of task expansion performance across different models.

Method	Task Split	Expansion Strategy	SemSeg mIoU↑	Parts mIoU↑	Saliency mIoU↑	Normal mErr↓	$R_{avg}^{retain}(\%)\uparrow$	$G_{new}(\%)\uparrow$	$\Delta N_{param}\downarrow$
MTI-Net	SemSeg, Saliency → Parts, Normal	Old-task Only	69.21	–	61.35	–	–	–	–
		Incremental Adaptation	64.83	53.16	57.12	19.92	93.39	-7.82	4.70
MTLoRA		Old-task Only	69.74	–	62.82	–	–	–	–
		Incremental Adaptation	63.21	54.03	58.48	19.56	91.86	-5.47	3.15
DiTASK		Old-task Only	68.13	–	62.21	–	–	–	–
		Incremental Adaptation	65.65	53.67	59.93	20.31	96.35	-5.01	0.92
TADFormer		Old-task Only	70.48	–	63.76	–	–	–	–
		Incremental Adaptation	64.97	54.24	60.37	19.12	93.43	-7.01	3.98
TGLoRA		Old-task Only	69.91	–	63.12	–	–	–	–
		Incremental Adaptation	64.42	54.83	60.85	19.98	94.28	-7.26	3.75
DeMT		Old-task Only	61.06	–	62.12	–	–	–	–
		Incremental Adaptation	56.42	49.83	58.85	19.91	93.57	-6.08	2.92
DeGMix		Old-task Only	69.38	–	63.79	–	–	–	–
		Incremental Adaptation	65.83	53.81	60.42	19.72	94.80	-8.09	3.94
FAAR		Old-task Only	67.36	–	63.03	–	–	–	–
		Incremental Adaptation	64.01	54.05	61.76	19.65	96.51	-5.93	3.10
Ours		Old-task Only	70.52	–	63.76	–	–	–	–
		Incremental Adaptation	70.11	58.19	63.43	17.86	99.45	-2.13	2.55
MTI-Net	SemSeg, Parts, Saliency → Normal	Old-task Only	69.35	58.92	64.38	–	–	–	–
		Incremental Adaptation	65.42	55.10	60.56	19.36	93.97	-3.42	2.85
MTLoRA		Old-task Only	68.31	57.86	63.24	–	–	–	–
		Incremental Adaptation	66.12	55.02	60.31	19.69	95.75	-5.63	2.55
DiTASK		Old-task Only	68.24	56.17	62.20	–	–	–	–
		Incremental Adaptation	65.01	54.31	58.28	21.78	95.22	-13.85	0.48
TADFormer		Old-task Only	69.64	58.83	63.74	–	–	–	–
		Incremental Adaptation	65.47	54.92	60.81	20.55	94.26	-14.80	2.49
TGLoRA		Old-task Only	68.89	58.47	64.12	–	–	–	–
		Incremental Adaptation	66.81	54.63	61.21	20.98	95.29	-14.33	2.17
DeMT		Old-task Only	59.42	52.61	60.96	–	–	–	–
		Incremental Adaptation	57.12	49.62	58.86	19.11	95.67	-3.47	1.96
DeGMix		Old-task Only	69.28	57.26	61.86	–	–	–	–
		Incremental Adaptation	66.21	55.31	60.02	20.46	96.40	-15.01	2.18
FAAR		Old-task Only	66.84	56.58	63.01	–	–	–	–
		Incremental Adaptation	64.72	54.69	61.18	20.81	96.86	-14.53	2.15
Ours		Old-task Only	71.08	58.31	64.14	–	–	–	–
		Incremental Adaptation	70.71	58.60	64.02	17.57	99.93	-1.42	1.14

and Saliency are used as old tasks, and Normal is added as a new task. For each method, we first train the Old-task Only model and then perform incremental adaptation under the same task split. Table 2 reports the old-task retention rate, new-task performance gain, and additional parameter cost. HazeMT-DCR achieves the highest R_{avg}^{retain} in both settings, reaching 99.45% and 99.93%, which indicates better preservation of old-task performance during task expansion. It also achieves the best G_{new} values of -2.13% and -1.42%, showing a smaller gap to full joint training on the newly added tasks. At the same time, HazeMT-DCR introduces only 2.55M and 1.14M additional parameters. These results show that HazeMT-DCR provides a better balance between old-task retention, new-task adaptation, and parameter efficiency than the compared methods.

4.4 Ablation Studies

We conduct ablation studies on HASE, ICFE-L, TRLR, and TICR, as shown in Table 3. The full model achieves the highest Δ_m of 3.96%. Removing $\mathcal{H}_\eta$, TICR, TRLR, and ICFE progressively reduces Δ_m to 3.14%, 2.36%, 1.35%, and 0.24%, respectively. These results demonstrate the benefits of degradation state modeling, counterfactual routing refinement, dynamic expert weighting, and task-conditioned functional experts. Notably, removing TICR does not change the 5.08M trainable parameters, indicating that it improves routing without additional trainable parameters. More detailed ablations on TCM and ICFE within ICFE-L are provided in the appendix.

Table 3: Ablation studies on HASE, TRLR, and TICR.

$\mathcal{H}_\eta$	ICFE	TRLR	TICR	#TP (M)	Δ_m (%)
✓				4.31	+0.24
✓	✓			4.77	+1.35
✓	✓	✓		5.08	+2.36
	✓	✓	✓	4.58	+3.14
✓	✓	✓	✓	5.08	+3.96

5 Conclusion

We propose HazeMT-DCR for incremental multi-task prediction in hazy scenes. It uses latent degradation states to enhance shared features and text instructions to modulate functional experts. Task-specific reinforcement learning routing dynamically adjusts expert weights according to image content and task requirements, while task-requirement-intervention counterfactual regret improves routing reliability and reduces negative transfer. By separating shared modules from task-specific parameters, HazeMT-DCR supports new-task expansion without changing old-task prediction paths. Experiments on hazy PASCAL-Context show improved overall performance. New tasks can also be added with few additional parameters while largely preserving old-task performance.

AI USE STATEMENT

In this work, we used generative AI tools for assist with translation. Additionally, we used generative AI tools for edit software code and edit a research paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work.

REFERENCES

Ahmed Agiza, Marina Neseem, and Sherief Reda. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16196–16205, 2024.

Seungmin Baek, Soyul Lee, Hayeon Jo, Hyesong Choi, and Dongbo Min. Tadformer: Task-adaptive dynamic transformer for efficient multi-task learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14858–14868, 2025.

Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.

Hang Dong, Jinshan Pan, Lei Xiang, Zhe Hu, Xinyi Zhang, Fei Wang, and Ming-Hsuan Yang. Multi-scale boosted dehazing network with dense feature fusion. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2154–2164, 2020.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88: 303–338, 2010.

Maxime Fontana, Michael Spratling, and Miaojing Shi. Faar: Efficient frequency-aware multi-task fine-tuning via automatic rank selection. *arXiv preprint arXiv:2603.20403*, 2026.

Neeraj Gangwar, Anshuka Rangi, Rishabh Deshmukh, Holakou Rahmanian, Yesh Dattatreya, and Nickvash Kani. Parameter-efficient multi-task learning via progressive task-specific adaptation. In *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, pp. 3916–3924, 2026.

Chun-Le Guo, Qixin Yan, Saeed Anwar, Runmin Cong, Wenqi Ren, and Chongyi Li. Image dehazing transformer with transmission-aware 3d position embedding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5802–5810, 2022.

Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2341–2353, 2011.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, pp. 709–727, 2022.

Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4780–4788, 2017.

Xiaohong Liu, Yongrui Ma, Zhihao Shi, and Jun Chen. Griddehazenet: Attention-based multi-scale network for image dehazing. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7313–7322, 2019.

Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 36889–36901, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

Yuxiang Lu, Shalayiding Sirejiding, Yue Ding, Chunlin Wang, and Hongtao Lu. Prompt guided transformer for multi-task dense prediction. *IEEE Transactions on Multimedia*, 26:6375–6385, 2024.

Krishna Sri Ipsit Mantri, Carola-Bibiane Schönlieb, Bruno Ribeiro, Chaim Baskin, and Moshe Eliasof. Ditask: Multi-task fine-tuning with diffeomorphic transformations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25218–25229, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *European Conference on Computer Vision (ECCV)*, pp. 527–543, 2020.

Rui-Qi Wu, Zheng-Peng Duan, Chun-Le Guo, Zhi Chai, and Chongyi Li. Ridcp: Revitalizing real image dehazing via high-quality codebook priors. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22282–22291, 2023.

Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018.

Yangyang Xu, Yibo Yang, and Lefei Zhang. Demt: Deformable mixer transformer for multi-task learning of dense prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 3072–3080, 2023.

Yangyang Xu, Yibo Yang, Bernard Ghanem, Lefei Zhang, Bo Du, and Jun Zhu. Degmix: Efficient multi-task dense prediction with deformable and gating mixer. *IEEE Transactions on Circuits and Systems for Video Technology*, 2026.

Yuqi Yang, Peng-Tao Jiang, Qibin Hou, Hao Zhang, Jinwei Chen, and Bo Li. Multi-task dense prediction via mixture of low-rank experts. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27927–27937. IEEE, 2024.

Haiming Yao, Qiyu Chen, Wei Luo, Zheng Zhang, Jianxing Liao, and Wei You. Tdss: Task dynamic-synergistic skill adaptation for boosting efficient and scalable multi-task learning in dense visual prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11857–11865, 2026a.

Haiming Yao, Wei Luo, Qiyu Chen, Jianxing Liao, and Wei You. Parameter-, memory-, time-efficient multi-task dense vision adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11848–11856, 2026b.

Hanrong Ye and Dan Xu. Inverted pyramid multi-task transformer for dense scene understanding. In *European Conference on Computer Vision*, pp. 514–530, 2022.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming–Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5718–5729, 2022.

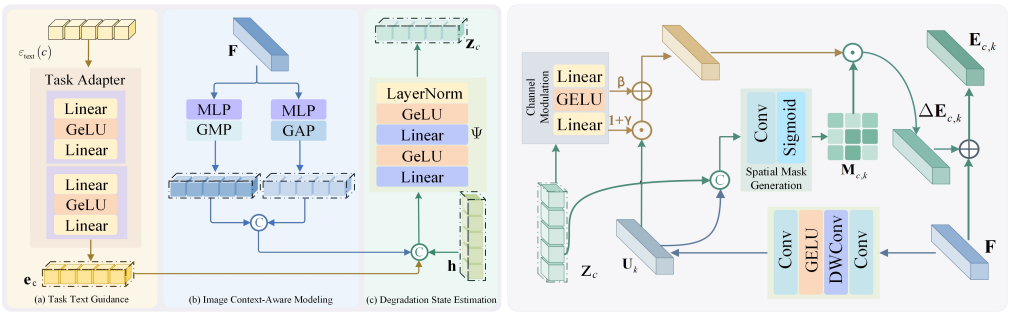
Yafei Zhang, Shuaitian Song, Huafeng Li, Shujuan Wang, and Yu Liu. Adaptive dynamic dehazing via instruction-driven and task-feedback closed-loop optimization for diverse downstream task adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12888–12896, 2026a.

Yan Zhang, Long Ma, Yuxin Feng, Zhe Huang, Fan Zhou, and Zhuo Su. Bilevel layer-positioning lora for real image dehazing. In *2026 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29969–29978, 2026b.

Zhengxi Zhang, Liang Zhao, Yunan Liu, Shanshan Zhang, and Jian Yang. Unified density-aware image dehazing and object detection in real-world hazy scenes. In *Asian Conference on Computer Vision*, pp. 119–135. Springer, 2020.

A Appendix

A.1 Network Architecture of the TCM and ICFE



(a) (b)

Figure 3: Detailed architectures of the TCM and ICFE. (a)Network architecture of the TCM. (b)Network architecture of the ICFE.

Fig. 3(a) illustrates the detailed architectures of the TCM and ICFE. As shown in Fig. 3(a), the TCM consists of three branches, namely TTG, DSE, and ICA. Their outputs are fused to generate the task requirement control vector $\mathbf{z}_c$. As shown in Fig. 3(b), the ICFE uses $\mathbf{z}_c$ to condition the expert features through CM and SMG, producing the functional expert feature $\mathbf{E}_{c,k}$ for subsequent expert routing.

A.2 Optimization Strategy

Training consists of two stages. During the expert warm-up stage, $\mathcal{L}_{task}$ and $\mathcal{L}_{sep}$ are used to train the haze-aware shared encoder, task requirement control module, functional expert library, static routing parameters, and task-specific output adaptation heads. This stage learns stable shared representations, expert transformations, and static routing priors. During the joint routing optimization stage, the actor–critic networks are introduced, while the static routing parameters remain frozen. The full objective $\mathcal{L}_{total}$ is then used for joint optimization. Through reparameterized Dirichlet sampling, the actor receives gradients from $\mathcal{L}_{task}$ and $\mathcal{L}_{cf}$, in addition to the PPO policy gradient from $\mathcal{L}_{route}$. During inference, only the haze-aware shared encoder, task requirement control module, functional expert library, actor networks, and task-specific output adaptation heads are retained. The critic networks and counterfactual branches are used only during training and are removed at inference. Therefore, they introduce no additional computational cost during testing.

A.3 Implementation Details

The proposed method is implemented using PyTorch 2.7.1 and CUDA 11.8. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU with 24 GB memory. The shared encoder uses an ImageNet-pretrained Swin-Tiny model. The model is trained for 400 epochs with a batch size of 16, including 200 epochs of expert warm-up and 200 epochs of joint routing optimization. We use AdamW for optimization. During the warm-up stage, the learning rate is set to 5×10^{-4} with a cosine annealing schedule. During joint routing optimization, the learning rate is reduced to

5×10^{-5} without further scheduling. The PPO clipping coefficient is set to $\epsilon_p = 0.2$. The loss weights are set to $\lambda_r = 1$, $\lambda_{cf} = 0.05$, and $\lambda_s = 0.1$.

Moreover, we use task-specific supervised losses for different prediction targets. Pixel-wise cross-entropy loss is used for semantic segmentation and human part parsing, balanced cross-entropy loss for saliency detection, and L_1 loss for surface normal estimation. We do not manually assign weights to individual task losses. Instead, each task loss is normalized using its exponential moving average (EMA), as defined in Eq. (19). During joint routing optimization, the normalized task loss is combined with the routing loss, counterfactual regret loss, and expert separation loss to form the overall objective in Eq. (20). For the four downstream tasks, we construct compositional text instructions for all non-empty task combinations, resulting in 15 instructions. Each instruction corresponds to a four-dimensional binary task mask that determines the active task set $S(c)$. The text instruction is also fed into the frozen BERT encoder to generate the task requirement representation.

A.4 Analysis of Incremental Adaptation for New Tasks

The proposed framework supports adding new downstream tasks without retraining the entire model. This is enabled by separating shared modules from task-specific parameters. The haze-aware shared encoder, task requirement control module, and functional expert library learn reusable visual representations and expert transformations. In contrast, the static routing parameters, actor–critic networks, and output adaptation heads are task-specific. For a new task t_{new}, we freeze all shared modules and the task-specific parameters of existing tasks. We only add the static routing parameters $\mathbf{a}_{t_{new}}$, actor network $f_\theta^{t_{new}}$, critic network $V_{t_{new}}$, and output adaptation head $\mathcal{D}_{t_{new}}^{o(c,t_{new})}$. These parameters are trained using samples from the new task following the same warm-up and routing optimization procedure. The new task obtains its task-requirement representation from the text instruction and learns how to combine the existing functional experts. In this way, it reuses the shared representations and expert capabilities learned from previous tasks. Since the shared modules and existing task-specific parameters remain frozen, the parameters and prediction paths of existing tasks remain unchanged during incremental adaptation. Therefore, new tasks can be added with only a small set of additional parameters and without full-model retraining.

A.4.1 Evaluation Metrics for Task Expansion

In the new-task expansion experiments, we evaluate the model from three aspects: additional parameter cost, old-task performance retention, and new-task performance. Accordingly, we use three metrics: the number of added parameters, the old-task performance retention rate, and the new-task performance gain.

Number of added parameters. Let N_{param}^{base} denote the total number of parameters before adding new tasks, and N_{param}^{exp} denote the total number after expansion. The number of added parameters is defined as

$$\Delta N_{param} = N_{param}^{exp} - N_{param}^{base}. \tag{22}$$

A smaller ΔN_{param} indicates lower parameter overhead for task expansion.

Performance retention rate for old tasks. Let M_t^{base} denote the performance on an old task t before adding new tasks, and M_t^{exp} denote the performance on the same task after expansion. To account for different metric directions, we set $l_t = 1$ for higher-is-better metrics and $l_t = 0$ for lower-is-better metrics. The average old-task performance retention rate is defined as

$$R_{avg}^{retain} = \frac{1}{|\mathcal{T}_{old}|} \sum_{t \in \mathcal{T}_{old}} \left(\frac{M_t^{base}}{M_t^{exp}} \right)^{(-1)^{l_t}} \times 100\%. \tag{23}$$

For the two-new-task setting, $|\mathcal{T}_{old}| = 2$; for the one-new-task setting, $|\mathcal{T}_{old}| = 3$. A value close to 100% means that old-task performance is well preserved. A value below 100% indicates performance degradation, while a value above 100% indicates improvement.

Performance gain for new tasks. For a new task t, let M_t^{exp} denote the performance of the expanded model and M_t^{joint} denote the performance of Four-Task Joint on the same task. Using the

Table 4: Results of incremental task expansion under different settings. $\mathcal{T}_{old}$ and $\mathcal{T}_{new}$ denote the old-task and new-task sets, respectively.

$\mathcal{T}_{old}$	$\mathcal{T}_{new}$	New Tasks	Expansion Strategy	SemSeg mIoU↑	Parts mIoU↑	Saliency mIoU↑	Normal mErr↓	R_{avg}^{retain}(%) ↑	G_{new}(%) ↑	ΔN_{param}↓
SemSeg, Parts	Saliency, Normal	2 Tasks	Old-task Only	70.79	58.19	–	–	–	–	–
			Incremental Adaptation	69.82	57.91	64.22	17.10	99.07	+0.39	1.45
SemSeg, Saliency	Parts, Normal		Old-task Only	70.52	–	63.76	–	–	–	–
			Incremental Adaptation	70.11	58.19	63.43	17.86	99.45	-2.13	2.55
SemSeg, Normal	Parts, Saliency		Old-task Only	70.66	–	–	17.69	–	–	–
			Incremental Adaptation	70.62	58.11	63.51	17.65	100.09	-1.43	1.72
Parts, Saliency	SemSeg, Normal		Old-task Only	–	58.89	63.88	–	–	–	–
			Incremental Adaptation	68.60	58.49	63.08	18.00	99.03	-3.40	1.67
Parts, Normal	SemSeg, Saliency		Old-task Only	–	59.28	–	17.54	–	–	–
			Incremental Adaptation	68.13	58.26	63.21	18.22	97.27	-2.80	0.84
Saliency, Normal	SemSeg, Parts		Old-task Only	–	–	64.12	17.34	–	–	–
			Incremental Adaptation	68.98	58.25	63.84	17.28	99.96	-1.69	1.93
Parts, Saliency, Normal	SemSeg	1 Task	Old-task Only	–	58.14	64.15	17.72	–	–	–
			Incremental Adaptation	68.52	58.17	64.19	17.63	100.21	-3.00	0.53
SemSeg, Saliency, Normal	Parts		Old-task Only	70.00	–	63.99	17.81	–	–	–
			Incremental Adaptation	70.21	57.77	63.92	17.84	100.01	-1.86	1.41
SemSeg, Parts, Normal	Saliency		Old-task Only	70.74	58.45	–	17.94	–	–	–
			Incremental Adaptation	70.64	58.34	63.73	17.88	100.00	-1.24	0.31
SemSeg, Parts, Saliency	Normal		Old-task Only	71.08	58.31	64.14	–	–	–	–
			Incremental Adaptation	70.71	58.60	64.02	17.57	99.93	-1.42	1.14

same l_t to account for metric direction, the average new-task performance gain is defined as

$$G_{new} = \frac{1}{|\mathcal{T}_{new}|} \sum_{t \in \mathcal{T}_{new}} (-1)^{l_t} \frac{M_t^{joint} - M_t^{exp}}{M_t^{joint}} \times 100\%. \tag{24}$$

When only one new task is added, $|\mathcal{T}_{new}| = 1$, and G_{new} directly measures the performance difference for that task. A positive G_{new} means that the expanded model outperforms Four-Task Joint, while a negative value means that it performs worse. Four-Task Joint is used as the reference and therefore has $G_{new} = 0$.

A.4.2 Task Scalability Analysis

In this section, we further evaluate incremental task scalability under two settings. In the first, two tasks are treated as old tasks and the remaining two are added simultaneously. In the second, three tasks are used as old tasks and the remaining task is added. For each setting, we first train an Old-task Only model using the old tasks. New tasks are then introduced with their text instructions, static routing parameters, actor–critic networks, and output adaptation heads, while all shared modules and old-task parameters remain frozen.

We compare three training settings: Four-Task Joint, Old-task Only, and Incremental Adaptation. Four-Task Joint trains all four tasks from scratch and serves as the full joint-training reference. Old-task Only provides the model before task expansion, while Incremental Adaptation adds new tasks by training only their task-specific modules. We evaluate task expansion using the number of added parameters, the old-task performance retention rate, and the new-task performance difference relative to Four-Task Joint. Their definitions are provided in the appendix.

Table 4 summarizes the results. When two tasks are added, R_{avg}^{retain} ranges from 97.27% to 100.09%, while G_{new} ranges from -3.40% to +0.39%. Only 0.84M–2.55M additional parameters are required. When one task is added, R_{avg}^{retain} ranges from 99.93% to 100.21%, and G_{new} ranges from -3.00% to -1.24%, with only 0.31M–1.41M additional parameters. These results show that HazeMT-DCR can add new tasks with low parameter overhead while largely preserving old-task performance.

A.4.3 Scalability to Object Detection Task in Real Hazy Scenes

To further verify the model's scalability, we add object detection in real hazy scenes as a new task in additional to the original four dense prediction tasks. Experiments are conducted on the RTTS real hazy object detection dataset, which consists of $4,322$ images covering five classes (person, bicycle, car, motorbike, bus), with 80% of the images for training and the remaining 20% for testing. For each method, we initialize from its pre-trained four-task model, freeze the shared modules and all parameters of existing tasks, and only train the task-specific modules and detection head for object

Table 5: Comparison of performance and computational complexity for different methods extended to object detection task.

Method	Person AP	Bicycle AP	Car AP	Motorbike AP	Bus AP	mAP ↑	ΔN_{param} ↓	ΔGFLOPs ↓
DeGMix	70.46	**70.01**	73.02	63.45	**52.41**	65.87	18.25	91.01
DeMT	76.08	62.03	76.46	**66.44**	48.04	65.81	18.03	138.64
DiTASK	74.13	65.12	71.42	62.84	43.08	63.32	19.13	86.94
FAAR	73.96	63.99	**76.52**	60.27	46.22	64.19	17.80	49.92
MTI-Net	65.43	58.86	67.37	55.23	46.97	58.77	29.28	82.44
MTLoRA	64.62	63.58	69.38	57.77	44.82	60.03	18.20	39.87
TADFormer	63.36	58.14	69.44	58.78	47.40	59.43	18.14	43.59
TGLoRA	64.90	60.48	68.93	54.54	47.26	59.22	18.22	81.30
Ours	**79.06**	67.53	75.94	64.05	48.75	**67.06**	**17.27**	**36.33**

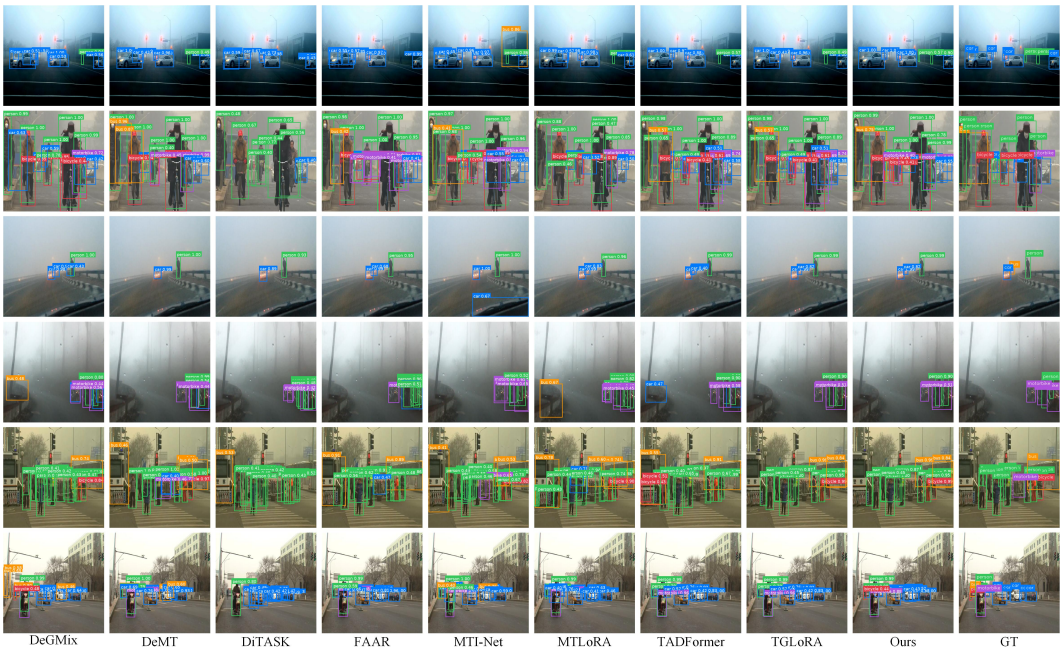


Figure 4: Qualitative comparison of different methods extended to the object detection task in real hazy scenes.

detection. All methods use identical data splits and evaluation protocols. We report per-class AP and mAP at IoU=0.5, together with the increments in parameters and GFLOPs introduced by object detection task.

Table 5 lists the quantitative results of different methods extended to the object detection task in real-world hazy scenes. Our method achieves the highest mAP, exceeding the second-best value by 1.19%. It reaches an AP of 79.06% for the person category, ranking first among all competitors. Our method also obtains the second-best AP on the bicycle, motorbike, and bus categories. Meanwhile, the increments in parameters and GFLOPs introduced by our method are the lowest across all compared approaches. Figure 4 further presents qualitative results for the object detection task in real-world hazy scenes. As can be seen from Figure 4, our method can localize primary targets more completely and accurately than competing methods for low-contrast, distant targets and scenes with densely packed pedestrians and vehicles. In summary, the quantitative and qualitative results demonstrate that our method possesses superior cross-task scalability to new tasks.

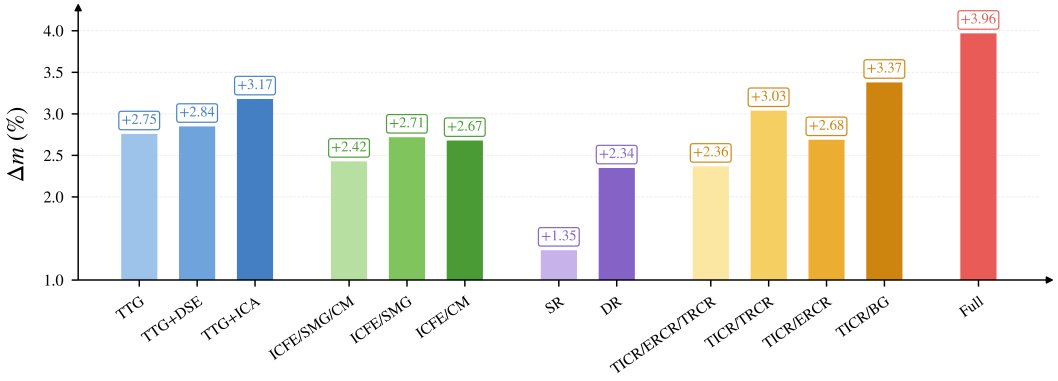


Figure 5: Detailed ablation studies.

A.5 More Ablation Studies

To further analyze the contribution of the internal components of key modules, we conduct fine-grained ablation studies on ICFE-L, TRLR, and TICR, as shown in Figure 5. Specifically, we examine the components of TCM and ICFE in ICFE-L, the Routing Policy (RP) in TRLR, and the Baseline Gate (BG), Expert-removal Counterfactual Regret (ERCR), and Task-requirement Replacement Counterfactual Regret (TRCR) in TICR.

TCM consists of Task Text Guidance (TTG), Degradation State Estimation (DSE), and Image Context-Aware Modeling (ICA). To analyze the contribution of different information sources, we replace the complete TCM with TTG, TTG+DSE, and TTG+ICA. Their corresponding Δ_m values are 2.75%, 2.84%, and 3.17%, respectively. Compared with TTG alone, adding either DSE or ICA improves performance. This shows that degradation state and image context information provide useful complements to task requirement modeling.

ICFE consists of Channel Modulation (CM) and Spatial Mask Generation (SMG). Removing both CM and SMG (ICFE\CM\SMG) gives a Δ_m of only 2.42%. Retaining only CM (ICFE\SMG) increases Δ_m to 2.71%, while retaining only SMG (ICFE\CM) achieves 2.67%. When both CM and SM are used, the complete ICFE configuration reaches 3.96%. These results show that both channel modulation and spatial masking improve expert adaptation to different task requirements, and their combination gives the best performance.

To evaluate the routing policy, we compare PPO routing with Static Routing (SR) and Dynamic Routing (DR). SR assigns uniform weights to all experts and achieves a Δ_m of 1.35%. DR learns dynamic expert weights through Softmax and improves Δ_m to 2.34%. PPO routing further increases it to 3.96%. These results show that dynamic expert weighting is more effective than static weighting, while PPO-based routing provides further gains.

Finally, we ablate the BG, ERCR, and TRCR components of TICR. Retaining only BG (TICR\ERCR\TRCR) gives a Δ_m of 2.36%. Removing ERCR (TICR\ERCR) results in 2.68%, while removing TRCR (TICR\TRCR) gives 3.03%. When ERCR and TRCR are retained but BG is removed (TICR\BG), Δ_m reaches 3.37%. The complete TICR achieves 3.96%. These results show that BG, ERCR, and TRCR all contribute to performance, and their combination performs best.

A.6 Hyperparameter Analysis

To analyze the model's sensitivity to hyperparameters, we examine the effects of the routing loss weight λ_r, counterfactual regret loss weight λ_{cf}, expert separation loss weight λ_s, and the number of experts K on the average multi-task performance change Δ_m. As shown in Figure 6, increasing λ_r from 0.2 to 1 steadily improves Δ_m from 3.07% to 3.96%, indicating that a larger routing loss weight is beneficial within the tested range. As λ_{cf} and λ_s increase, Δ_m first rises and then declines, reaching the highest value of 3.96% at 0.05 and 0.1, respectively. This suggests that moderate counterfactual regret and expert separation constraints are more effective for multi-task learning.

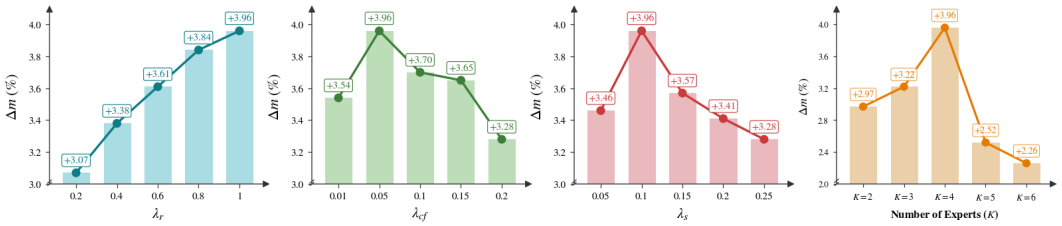


Figure 6: Hyperparameter analysis of loss weights and the number of experts. From left to right, we evaluate the routing loss weight λ_r, counterfactual regret loss weight λ_{cf}, expert separation loss weight λ_s, and the number of experts K. The vertical axis shows the average multi-task performance change Δ_m (%), where a larger value indicates better overall performance. The annotated values correspond to the results of each tested configuration.

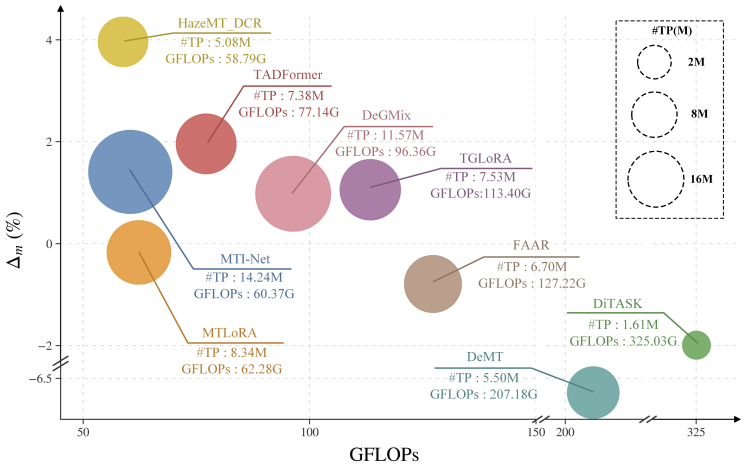


Figure 7: Comparison of multi-task performance and computational efficiency. The horizontal axis shows GFLOPs, the vertical axis shows the average multi-task performance change Δ_m, and the bubble size indicates the number of trainable parameters.

When K increases from 2 to 4, Δ_m improves from 2.97% to 3.96%. Further increasing K to 5 and 6 reduces Δ_m to 2.52% and 2.26%, respectively, showing that more experts do not necessarily lead to better performance. Based on these results, we set λ_r, λ_{cf}, λ_s, and K to 1, 0.05, 0.1, and 4, respectively.

A.7 Computational Complexity Analysis

Figure 7 compares the overall multi-task performance, computational cost (GFLOPs), and trainable parameters (#TP) of different methods on semantic segmentation, human part parsing, saliency detection, and surface normal estimation. Our method achieves $\Delta_m = 3.96\%$ with 58.79 GFLOPs and 5.08M trainable parameters, giving the best overall performance and the lowest computational cost among the compared methods. Compared with TADFormer, our method improves Δ_m by 2.00 percentage points while using fewer trainable parameters and lower computation. This shows a better balance between multi-task performance and efficiency.

A.8 Additional Visual Analysis

Figure 8 shows the performance of different methods under varying haze levels. Our method achieves Δ_m values of 7.21%, 3.96%, and 1.32% under light, medium, and heavy haze, respectively, outperforming all compared methods. As the haze level increases, the gains of all methods decrease. However, our method still maintains a positive gain under heavy haze, while the others

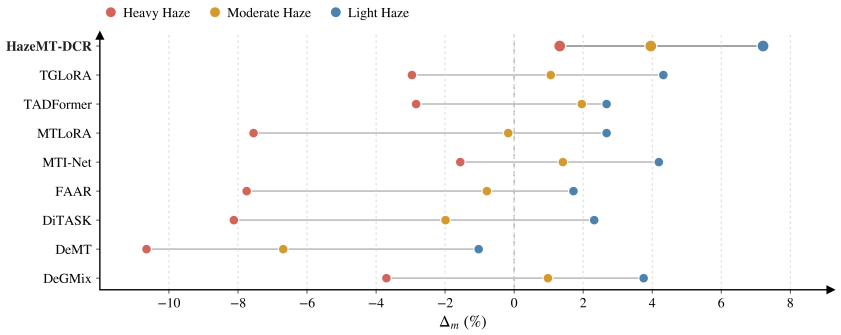


Figure 8: Average multi-task performance under different haze levels. Blue, orange, and red denote light, medium, and heavy haze, respectively. Gray lines connect the results of the same method, and the vertical dashed line indicates zero gain.

fall below zero. This indicates that our method can better preserve multi-task learning benefits under stronger degradation.

Figure 9 shows the predictions of four tasks for the same scene under different haze levels. Under light and medium haze, our method better preserves local semantic responses in the red-box regions, produces clearer pedestrian contours in surface normal estimation, and gives saliency responses closer to the GT for the smaller pedestrian on the left. The human part parsing results also preserve the main body regions and their boundaries. Under heavy haze, some local semantic details are still lost, showing that strong haze degradation remains challenging for small-region recognition.

Figure 10 further shows the predictions of four tasks for the different scenes under same haze levels. Existing methods generally suffer from issues such as semantic boundary ambiguity, lack of response to small targets, incomplete prediction of human body parts, and scattered saliency responses in the red-box regions. In contrast, our method generates more complete and clear predictions that better align with GT across all four tasks. Particularly for distant targets and low-contrast images, our method maintains good target integrity and cross-task prediction consistency, indicating its robust multi-task prediction capability under different foggy scenarios.

A.9 Limitations and Future Work

HazeMT-DCR still has several limitations. The reinforcement learning router and counterfactual interventions increase training complexity, although they do not affect inference cost. The framework also relies on predefined textual task instructions, whose quality may influence task conditioning. In addition, each new task still requires task-specific routing parameters and an output head, despite reusing the shared backbone and expert library. Future work will focus on simplifying routing optimization, improving task representations, and reducing the additional parameters required for task expansion. Evaluating the method on more diverse and real-world haze distributions is also an important direction.

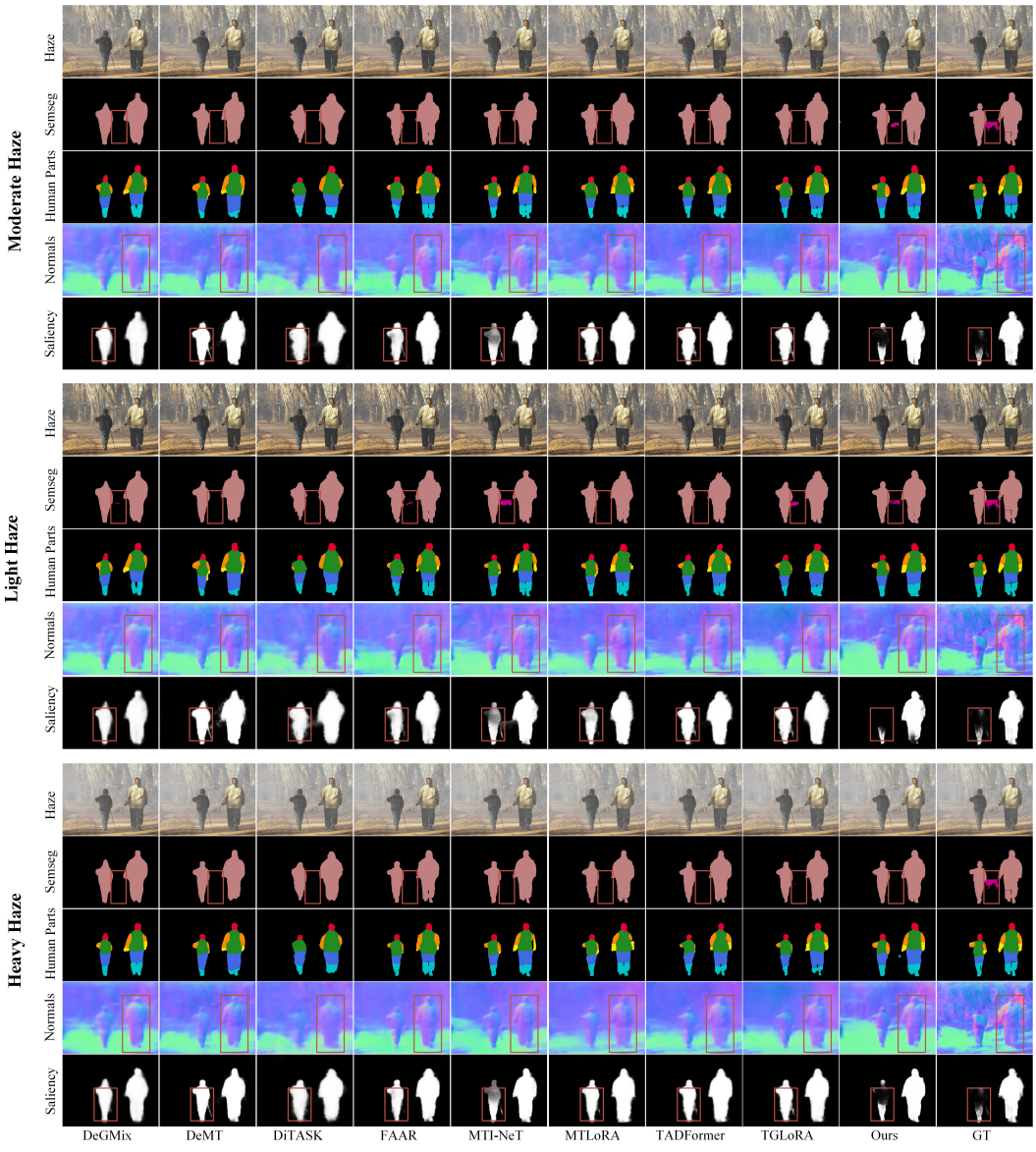


Figure 9: Multi-task prediction results under different haze levels. From top to bottom, the rows correspond to medium, light, and heavy haze. Each group shows the input image and predictions for the four tasks. Different columns correspond to different methods, and the last column shows the ground truth. Red boxes highlight local regions for comparison.

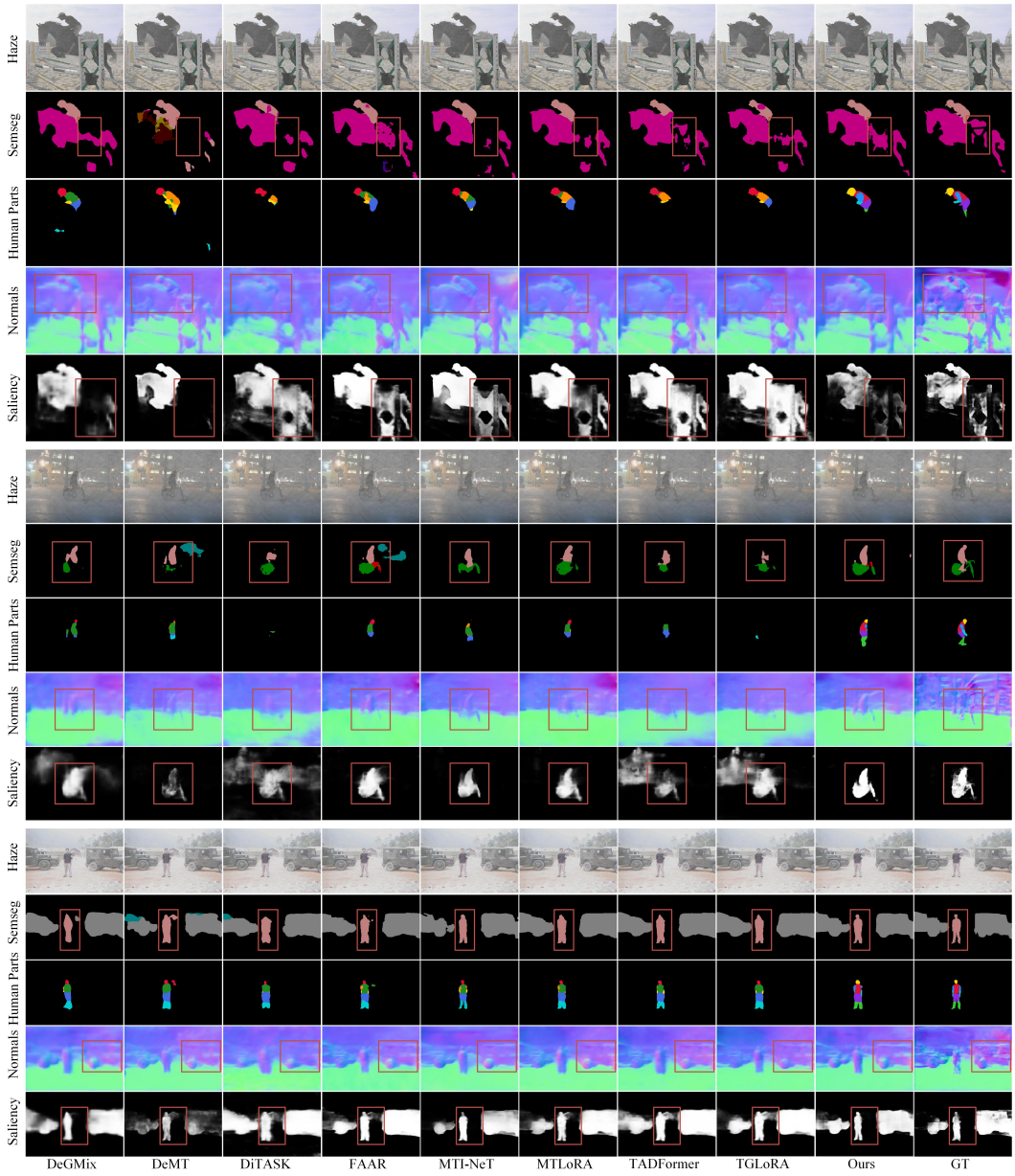


Figure 10: Different scenarios prediction results under same haze levels. Each group shows the input image and predictions for the four tasks. Different columns correspond to different methods, and the last column shows the ground truth. Red boxes highlight local regions for comparison.