TransWeather: Transformer-based Restoration of Images Degraded by Adverse Weather Conditions

Jeya Maria Jose Valanarasu, Rajeev Yasarla, and Vishal M. Patel
Johns Hopkins University
{jvalana1,ryasarl1,vpatel36}@jhu.edu

Abstract

Removing adverse weather conditions like rain, fog, and snow from images is an important problem in many applications. Most methods proposed in the literature have been designed to deal with just removing one type of degradation. Recently, a CNN-based method using neural architecture search (All-in-One) was proposed to remove all the weather conditions at once. However, it has a large number of parameters as it uses multiple encoders to cater to each weather removal task and still has scope for improvement in its performance. In this work, we focus on developing an efficient solution for the all adverse weather removal problem. To this end, we propose TransWeather, a transformer-based end-to-end model with just a single encoder and a decoder that can restore an image degraded by any weather condition. Specifically, we utilize a novel transformer encoder using intra-patch transformer blocks to enhance attention inside the patches to effectively remove smaller weather degradations. We also introduce a transformer decoder with learnable weather type embeddings to adjust to the weather degradation at hand. TransWeather achieves significant improvements across multiple test datasets over both All-in-One network as well as methods fine-tuned for specific tasks. TransWeather is also validated on real world test images and found to be more effective than previous methods. Implementation code can be found in the supplementary document. Code is available at https://github.com/jeya-maria-jose/TransWeather.

1. Introduction

Weather conditions like rain, fog, and snow reduce the visibility and corrupt the information captured by an image. This drastically affects the performance of many computer vision algorithms like detection, segmentation and depth estimation [3,5,41,52,59] which are important parts of autonomous navigation and surveillance systems [28,34–36]. Hence, it is essential to remove adverse weather effects from images in order to make these vision systems more reliable. Also, a clean image without any weather degrada-
have also been explored for weather removal tasks achieving better performance than CNNs [38, 48, 74]. Most of these methods just focus on one task at a time or fine-tune the model separately for each task. Although they achieve excellent performance, these are not generic solutions for all adverse weather removal problems as the networks have to be trained separately for each task. This makes it difficult to adopt them for real-time systems as they have to be multiple models making it computationally complex. Also, the system would have to decide and switch between a series of weather removal algorithms (Figure 1 (a)) making the pipeline more complicated.

Recently, Li et al. [23] proposed an All-in-One bad weather removal network which was the first work to propose an algorithm that takes in an image degraded by any weather condition as input and predicts the clean image. All-in-One network was tested across 3 datasets of rain, fog, and snow removal and achieved better or comparable performance than the previous methods which were tuned individually on separate datasets. All-in-One network is CNN-based and uses multiple encoders. In particular, it uses separate encoders for the different weather degradation at hand and uses neural architecture search to find the best network to address the problem (Figure 1 (b)). This network is still computationally complex as there are multiple encoders. To the best of our knowledge, no other methods apart from All-in-One network [23] have been proposed for a generic adverse weather removal in the literature. Although recent methods like MPR-Net [67], U-former [55], Swin-IR [27] have been proposed as generic restoration networks validated on multiple datasets, they are still fine-tuned on the individual datasets and do not use a single model for all the weather removal tasks.

In this work, we propose a single encoder-single decoder transformer network, called TransWeather, to tackle all adverse weather removal problems at once. Instead of using multiple encoders, we introduce weather type queries in the transformer decoder to learn the task (Figure 1 (c)). Here, the multi-head self attention mechanisms take weather type queries as input and match it with keys and values taken from features extracted from the transformer encoder. These weather type embeddings are learned along with the network to understand and adapt to the weather degradation type present in the image. The decoded features and the hierarchical features obtained form the encoder are fused and projected to the image space using a convolutional block. Thus, TransWeather just has one encoder and one decoder to learn the weather type as well as produce the clean image. Transformers are good at extracting rich global information when compared to CNNs [9]. However, we argue that when the patches are large like in ViT [9], we fail to attend much to the information within the patch. Weather degradations like rain streak, rain drop and snow are usually small in size and so multiple artifacts can occur within a single patch.

To this end, we propose a novel transformer encoder with intra-patch transformer (Intra-PT) blocks. Intra-PT works on sub-patches created from the original patches and excavates features and details of smaller patches. Intra-PT thus focuses on attention inside the main patches to remove weather degradations effectively. We use efficient self-attention mechanisms to calculate the attention between sub-patches to keep the computational complexity low. From our experiments, we find that introducing Intra-PT blocks enhances the performance of transformer and helps it adapt better to weather removal tasks. We train our network on a similar configuration as All-in-One and obtain superior performance across multiple test datasets for rain removal, snow removal, fog removal and even a combination of these weather degradations. We also outperform the methods designed specifically for these individual tasks which are finetuned on those datasets. We also show that TransWeather is fast during inference. Finally, we also test TransWeather on real-world weather degraded images, achieving excellent performance compared to the previous methods. TransWeather can act as an efficient backbone in the future for generic weather removal frameworks.

The key contributions of this work are as follows:

• We propose TransWeather - an efficient solution for all adverse weather removal problem with just a single encoder and a single decoder using transformers. We propose using weather type queries to efficiently handle the All-in-One problem.
• We propose a novel transformer encoder using intra-patch transformer (Intra-PT) blocks to cater to fine detail feature extraction for low-level vision tasks like weather removal.
• We achieve state-of-the-art performance on multiple datasets. We also validate the effectiveness of the proposed method on real-world images.

2. Related Works

Adverse weather removal problems like deraining [16, 21, 25, 51, 61, 65, 76], dehazing [1, 2, 10, 22, 42, 69], desnowing [29, 44, 44, 73] and rain drop removal [37, 39, 40, 66] have been extensively explored in the literature.

Rain Streak Removal: Yang et al. [61] used a recurrent network to decompose rain layers to different layers of various streak types to remove the rain. Zhang et al. [70] proposed using a conditional GAN for image deraining. Yasarla et al. [64] explored using Gaussian processes to perform transfer learning from synthetic rain data to real-world rain data. Quan et al. [39] used a complementary cascaded network to remove rain streaks and raindrops in a unified framework. A more detailed survey of various rain removal methods can be found in [62].

Fog Removal: Li et al. [18] proposed a CNN network considering both atmospheric light and transmission map to

**Rain drop Removal:** You et al. [66] proposed using temporal information to perform video-based raindrop removal. Qian et al. [37] used an attention GAN to remove raindrop and also introduced a new dataset. Quan et al. [40] used a dual attention mechanism to remove effects of raindrops.

**Snow Removal:** Desnow-Net [29] was one of the first CNN-based methods proposed to remove snow from an image. Li et al. [20] proposed a stacked dense network for snow removal. Chen et al. [6] proposed JSTASR in which a size and transparency aware mechanism was proposed to remove snow. Recently, DDMSNet [72] proposed a deep dense multiscale network using semantic and geometric priors for snow removal.

**All-in-One Weather Removal:** All-in-One Network [23] was proposed to handle multiple weather degradations using a single network. All-in-One uses a generator with multiple task-specific encoders and a common decoder. It uses a discriminator to classify the degradation type and only backpropagates the loss to specific encoders. It also uses neural architecture search to optimize the feature extraction from the encoder.

**Transformers in low-level vision:** Since the introduction of Vision Transformer (ViT) [9] for visual recognition, transformers have been widely adopted for various computer vision tasks [12, 31, 49, 60, 75]. Especially for low-level vision, Image processing transformer [4] shows how pretraining a transformer on large-scale datasets can help in obtaining a better performance for low-level applications. U-former [55] proposed a U-Net based transformer architecture for restoration problems. Swin-IR [27] adopted Swin Transformer [30] for image restoration. Zhao et al. [74] proposed a local-global transformer specifically for image dehazing. A multi-branch network [48] for dehazing was also proposed based on swin transformer. In ETDNet [38], an efficient transformer block to extract features in a coarse to fine way for image deraining was proposed.

Unlike the above methods, we propose a transformer-based single-encoder single-decoder network to solve all adverse weather removal tasks using a single model instance. Our Transformer encoder is also modified to cater to low-level tasks with the introduction of intra-patch transformer block. Our transformer decoder is trained with weather type queries to learn the task and uses that information to restore the clean image.

### 3. Proposed Method - TransWeather

In the literature, different weather phenomenons have been modelled differently with regards to the underlying physics involved. Rain drop [37] is modelled as

$$ I = (1 - M) \odot B + R, $$

(1)

where $I$ is the degraded image, $M$ is the mask, $B$ is the background and $R$ is the raindrop residual. Heavy rain with rain streaks and fog effect [21] is modelled as

$$ I = T \odot (B + \sum_{i} R_i) + (1 - T) \odot A, $$

(2)

where $T$ is the transmission map produced by scattering effect, and $A$ is the atmospheric light in the scene. According to [29], snow is generally modeled as

$$ I = M \odot S + M \odot (1 - z), $$

(3)

where $z$ is a mask indicating snow and $S$ corresponds to snow flakes. All-in-One method [23] generalizes the adverse weather removal problem as

$$ B = D(E_{ip}(I_{ip})), $$

(4)

where $E$ corresponds to the encoder and $D$ corresponds to the decoder. $p$ represents the weather type present in the image. Note that for each weather type a different encoder is used. In this work, we follow a similar formulation of all adverse weather removal as

$$ B = T(I_{ip}), $$

(5)

where $T$ corresponds to TransWeather which consists of a weather agnostic encoder and decoder network unlike All-in-One Network. The weather type queries are learnt along with the parameters of $T()$ thus making the problem setup more generic. We motivate this setup because a problem as generic as weather removal cannot be addressed by merely seeking for perfection on solving individual tasks. This formulation not only makes the process computationally efficient, but also helps in using complimentary information between the tasks to further improve the performance. Furthermore, it is also grounded with regards to how human vision works as our visual cortex can perform multiple tasks without any difficulty. This view is widely agreed in neurobiology as the visual cortex does not have different modules for different perception tasks [24, 32].

#### 3.1. Network Architecture

Given a degraded image $I$ of size $H \times W \times 3$, we first divide it into patches. We then feed forward the patches to a transformer encoder containing transformer blocks at different stages. Across each stage, the resolution is reduced to make sure the transformer learns both coarse and fine information. We then use a transformer decoder block that uses the encoded features as keys and values while using learnable weather type query embeddings as queries. The extracted features are then passed through a convolutional projection block to get the clean image of dimensions $H \times W \times 3$. An overview of the network architecture of
TransWeather can be found in Figure 2. In the following sections, we describe these components in detail.

### 3.1.1 Transformer Encoder

We generate a hierarchical feature representation of the input image by extracting multi-level features in the transformer encoder. The features are extracted at different stages in the encoder thus facilitating extraction of both high-level and low-level features. Across each stage, we perform overlapped patch merging [59]. Using this we combine overlapping feature patches to get features of the same size as that of non-overlapped patches before passing the features to the next stage.

**Transformer Block:** In each transformer block, we use multi-head self-attention layers and feed forward networks to calculate the self-attention features. The computation can be summarized as:

$$T_i(I_i) = FFN(MSA(I_i) + I_i),$$

where $T()$ represents the transformer block, $FFN()$ represents the feed forward network block, $MSA()$ represents multi head self attention, $I$ is the input and $i$ represents the stage in the encoder. Similar to the original self-attention network, the heads of queries ($Q$), keys ($K$) and values ($V$) have same dimensions and are calculated as:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V,$$

where $d$ represents the dimensionality. Note that we have multiple attention heads in each transformer block and that number is a hyper-parameter which we vary across each stage in the transformer encoder. More details regarding the hyper-parameter settings can be found in the supplementary document. We reduce the complexity of the original self-attention from $O(N^2)$ to $O(N^2/R)$ by introducing a reduction ratio $R$ [54]. We reshape the keys into a dimension from a dimension of $(N,C)$ to a dimension of $(N/R,C,R)$. We then use a linear layer to get the second dimension back to $C$ from $C.R$. Hence, the keys get a dimension of $N/R \times C$ thus reducing the complexity while calculating the self attention. The self-attention features are then passed to a FFN block. The FFN block used here has a slight variation from ViT as we introduce using depth-wise convolution to MLP inspired from [26, 58, 59]. Using depth-wise convolution here helps bring locality information and provide positional information for transformers as shown in [59]. The computation in the FFN block can be summarized as follows:

$$FFN_i(X_i) = MLP(\text{GELU}(\text{DWC}(\text{MLP}(X_i)))) + X_i,$$

where $X$ refers to self-attention features, $DWC$ is depth-wise convolution [7], $GELU$ is Gaussian error linear units [14], $MLP$ is multi-layer perceptron, $i$ indicates the stage.

**Intra-Patch Transformer Block:** The intra-patch transformer blocks are present in between each stage in the transformer encoder. These blocks take in the sub-patches created from the original patches as the input. These sub-patches are fixed in dimensions half of height and width of the original patch. Intra-PT also utilizes a similar transformer block as explained above. We use a high $R$ value to make the computation very efficient in the Intra-PT block. Intra-PT block helps in extracting fine details helpful in removing smaller degradation as we operate on smaller patches. Note that the Intra-PT block creates patches at the feature level except at the first stage where it is done at the image level. The output self-attention features from the Intra-PT block are added to the self-attention features from the main block across the same stage. The formulation of feed forward process in our transformer encoder can be summarized as follows:

$$Y_i = MT_i(I_i) + IntraPT_r(P(X_i)),$$

where $I$ is input to the transformer across each stage, $Y$ is the output across each stage, $MT()$ is the main transformer.
block, IntraPT is the intra-patch transformer block, $P()$ corresponds to the process of creating sub-patches from the input patches and $i$ denotes the stage.

### 3.1.2 Transformer Decoder

In the original transformer decoder [50], an autoregressive decoder is used to predict the output sequence one element at a time. Detection transformer (DETR) [3] uses object queries to decode the box coordinates and class labels to produce the final predictions. Inspired from them, we define weather type queries to decode the task, predict a task feature vector and use it to restore the clean image. These weather type queries are learnable embeddings which are learnt along with the other parameters of our network. These queries attend to the feature outputs from the transformer encoder. The transformer decoder here operates at a single stage but has multiple blocks. We illustrate the transformer decoder block in Figure 3. These transformer blocks are similar to encoder-decoder transformer blocks [50]. Unlike self-attention transformer block where $Q$, $K$ and $V$ are taken from the same input, here $Q$ is weather type learnable embedding while $K$ and $V$ are the features taken from the last stage of the transformer encoder. The output decoded features represent the task feature vector and are fused with the features extracted across the transformer encoder at each stage. All of these features are forwarded to the convolutional tail to reconstruct the clean image.

![Figure 3](image)

**Figure 3. Configuration of the transformer block in the decoder.** The queries here are learnable embeddings representing the weather type while the keys and values are features taken from the last stage of the transformer encoder.

### 3.1.3 Convolutional Projection Block

The set of hierarchical transformer encoder features and task features from the transformer decoder are passed through a set of 4 convolutional layers to output the clean image. We also use an upsampling layer before every convolutional layer to get back to the original image size. We also have skip connections across each stage in the convolutional tail from the transformer encoder. We also use a tanh activation function in the final layer.

### 3.2. Loss

Our network is trained in an end-to-end fashion using a smooth L1-loss between the prediction ($\hat{I}$) and the ground truth ($G$) defined as follows:

$$L_{\text{smoothL1}} = \begin{cases} 0.5E^2 & \text{if } |E| < 1 \\ |E| - 0.5 & \text{otherwise} \end{cases},$$

where $E = \hat{I} - G$. We also add a perceptual loss that measures the discrepancy between the features of prediction and the ground truth. We extract these features using a VGG16 network [47] pretrained on ImageNet. We extract features from the 3rd, 8th and 15th layers of VGG16 to calculate the perceptual loss. The perceptual loss is formulated as follows

$$L_{\text{perceptual}} = L_{\text{MSE}}(VGG_{3,8,15}(\hat{I}), VGG_{3,8,15}(G)).$$

The total loss can be summarized as follows

$$L_{\text{total}} = L_{\text{smoothL1}} + \lambda L_{\text{perceptual}},$$

where $\lambda$ is a weight that controls the contribution of $L_{\text{perceptual}}$ and L1-loss on the overall loss.

### 4. Experiments

We conduct extensive experiments to show the effectiveness of our proposed method. In what follows, we explain the datasets, implementation details, experimental settings, results and comparison with state-of-the-art methods.

#### 4.1. Datasets

We train our network on a combination of images degraded from a variety of adverse weather conditions similar to All-in-One Network [23]. We follow the same training set distribution used in All-in-One for fair comparison. The training data consists of 9,000 images sampled from Snow100K [29], 1,069 images from Raindrop [37] and 9,000 images of Outdoor-Rain [21]. Snow100K has synthetic images degraded by snow, raindrop has real raindrop images and Outdoor-Rain has synthetic images degraded by both fog and rain streaks. We term this combination of training data as “All-Weather” for better representation.

We test our methods on both synthetic and real-world datasets. We use the Test1 dataset [21, 23], the RainDrop test dataset [37] and the Snow100k-L test set [29] for testing our method. In addition, we also evaluate on real-world images degraded by rain streaks and rain drops.

#### 4.2. Implementation Details

We implement our method using Pytorch framework [33] and train it using an NVIDIA RTX 8000 GPU. We use an Adam optimizer [17] and a learning rate of 0.0002. We use a learning rate scheduler that anneals the learning rate by 2 after 100 and 150 epochs. The network is trained for a total of 200 epochs with a batch size of 32. Other hyperparameters regarding the TransWeather architecture can be found in the supplementary document.

#### 4.3. Comparison with state-of-the-art methods

First, we compare our method with state-of-the-art methods which are designed specifically for each task: rain drop
We also compare the performance of our method with All-in-One network [23] which is trained to perform all the above tasks with a single model instance. Our method TransWeather is also trained to perform all these tasks using a single model instance.

4.3.1 Referenced Quality Metrics

We use PSNR and SSIM to evaluate the performance of different models. We tabulate the quantitative results in terms of PSNR and SSIM in Tables 1, 2, and 3 while evaluating on the Test1 (fog+rain removal), Snow100K-L (snow removal) and RainDrop (rain drop removal) test datasets, respectively. As Test1 has both fog and rain, we sequentially apply deraining and dehazing methods for fair comparison on this dataset. For example, while using DetailsNet and RESCAN for deraining, we apply Multi-scale boosted dehazing network (MSBDN) [8] for dehazing. Note that from our experiments we found MSBDN to be the best performing network for dehazing. We compare the performance while applying deraining first, then dehazing and also vice-versa. We train Swin-IR and MPRNet directly on “Outdoor-Rain” (training split of Test1) and test it on Test1 for fair comparison. Similarly, Swin-IR was trained on Snow100K dataset, RainDrop and tested on SnowTest100k-L, RainDrop test datasets respectively. It can be noted that some recent methods like CCN and DDMSNet when fine-tuned on the individual datasets outperform All-in-One. TransWeather outperforms All-in-One as well as all the task-specific methods by a significant margin as we cater to low-level weather details as well as use weather queries to efficiently handle the All-in-One problem.

4.3.2 Visual Quality Comparison

Synthetic Images We illustrate the predictions from synthetic test datasets like Test1 and Snow100k-L in Figures 4 and 5. It can be seen that Transweather achieves visually pleasing results compared to the previous methods. It works very well in removing both fog and rain streaks as can be seen in Figure 4 while other methods including All-in-One fail to remove at least one of the degradations. It can be seen from Figure 5 that our method removes even the snow particles which are very small in structure while All-in-One has hard time removing them.

Real-World Images We illustrate the predictions from real test datasets like RainDrop and Real-World images in Figures 6 and 7. It can be seen in both the figures that Tran-
Figure 4. Sample qualitative results on the Test1 dataset. Red box corresponds to the zoomed-in patch for better comparison.

Figure 5. Sample qualitative results on the Snow100k-L dataset. Red box corresponds to the zoomed-in patches.

sweather removes even the finest rain streaks or drops when compared to the previous methods.

5. Discussions

Ablation Study: We conduct an ablation study to understand the contributions of individual components proposed in the TransWeather architecture. We start with a base transformer encoder architecture and a conv tail. We call this configuration Transformer Base. We then convert the transformer encoder to hierarchical transformer (HE) encoder to extract both high-level and low-level features where we perform patch merging between each stage in the transformer encoder. We then add the intra patch transformer block (Intra-PT) in the encoder. Then we add learnable weather type queries and a transformer decoder block to learn the task embeddings. This configuration corresponds to the TransWeather architecture. All of these models are trained on All-Weather and tested on the Raindrop test dataset. The results of ablation study can be found in Table 4. It can be observed that each individual contribution of this work helps in improving the performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (↑)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer Base</td>
<td>30.12</td>
<td>0.8512</td>
</tr>
<tr>
<td>+ HE</td>
<td>31.62</td>
<td>0.8671</td>
</tr>
<tr>
<td>+ HE + Intra-PT</td>
<td>32.37</td>
<td>0.9463</td>
</tr>
<tr>
<td>+ HE + Intra-PT +Weather Queries</td>
<td>34.55</td>
<td>0.9502</td>
</tr>
</tbody>
</table>

Table 4. Ablation Study on the RainDrop test dataset. HE denotes converting to hierarchical transformer encoder and Intra-PT represents intra-patch transformer blocks.

What do the weather queries learn? The weather queries are embeddings which learn what type of degradation is present in the image. These queries help in predicting the corresponding task vector which is helpful to inject the task information to get a better prediction. To show this, we visualize the attention maps of eight random queries (out of 512) for three images corresponding to different weather degradations in Figure 8. It is interesting to observe that queries \(Q_1, Q_3,\) and \(Q_6\) activate highly for foggy image. They attend throughout the image to all the places afflicted by the fog. Queries \(Q_2, Q_4\) and \(Q_8\) are observed to activate highly for rainy images and the attention maps are sparse corresponding to the rain details. Similarly, queries \(Q_5\) and \(Q_7\) activate to snow images more when compared to images with rain and fog. This shows that different queries activate for different weather degradations helping TransWeather learn the underlying weather condition and give better predictions. It can also be noted that when an image is degraded by multiple weather conditions, multiple task type queries activate to encode specific tasks. This can be observed from the middle row of Figure 8 where queries that attend to both fog and rain activate as the image is degraded by both of these conditions.

Inference Time: In Figure 1 (bottom row), we compare the inference speed in terms of seconds. The time reported in the table corresponds to the time taken by each model to feed forward an image of dimensions \(256 \times 256\) during the inference stage. We note that our method is faster (with just 0.14 seconds per image) during inference when compared to the previous weather removal methods. TransWeather has 31 M parameters which are less than that of All-in-One Network which has 44 M parameters.

Differences from All-in-One: As the All-in-One network [23] is the first method to look into using a single model instance for all weather removal problems, we present clear differences of our method from All-in-One. First, All-in-One is a CNN-based method while TransWeather uses a transformer backbone built specifically for low-level vision tasks with an extra focus on operating on smaller patches. All-in-One uses multiple encoders while TransWeather utilizes a single encoder. All-in-One uses adversarial training and neural architecture search while TransWeather just uses a combination of L1 and perceptual loss making the train-
The document contains the following sections:

**6. Conclusion**

In this work, we proposed TransWeather - an efficient transformer-based solution for the all adverse weather removal problem. We focus on building a single model instance which can remove any weather degradation present in the image. We build a single encoder-decoder network for restoration while using learnable weather type queries in the decoder to learn the type of weather degradation and use that information for the weather removal process. We also propose a novel transformer encoder architecture base which work on sub-patches thus aiding transformers to remove small weather degradations more efficiently. We extensively experiment on multiple synthetic and real-world datasets where we push the current state-of-the-art by a significant amount using a single model instance while also obtaining a faster inference speed. We also obtain better visual results when tested on real-world adverse weather images.

**Acknowledgement:** This work was supported by an ARO grant W911NF-21-1-0135.
References


[53] Tianyu Wang, Xin Yang, Ke Xu, Shaozhe Chen, Qiang Zhang, and Rynson WH Lau. Spatial attentive single-image deraining with a high quality real rain dataset. In Proceeding-


