Fourier Domain Adaptation for Traffic Light Detection in Adverse Weather*

Supplementary Material

In this appendix, we provide supplementary material detailing the hyperparameters used in our experiments, along with an in-depth analysis of the extensive experimentation conducted to determine their optimal selection. Additionally, we present specifics of the benchmarking on clean images and tested on images affected by adverse weather conditions without applying FDA.

0.1. Datasets

The datasets LISA and S^2TLD were chosen since LISA supplies a rich night-time split, and S^2TLD adds high-resolution images of very small lights under a permissive MIT licence. Together, they cover our key corner-cases, low-light scenes, and tiny targets, without extra relabelling or preprocessing. Whereas DriveU and Bosch BSTLD were discarded since, DriveU encodes 344 composite classes that combine colour with arrows, pictograms, and relevance flags; only a handful map directly to the simple colour states we need, so most would have to be merged or discarded. Bosch BSTLD, meanwhile, is converted from 12-bit sensor data to 8-bit RGB, a step that can introduce colour artefacts and risk degrading FDA, which depends on accurate pixel intensities.

0.2. Hyperparameter Tuning

Table 1. Comparisons of different models when trained on D^s (clean, β=0) data but inferred on the D^t (foggy and rainy) data.

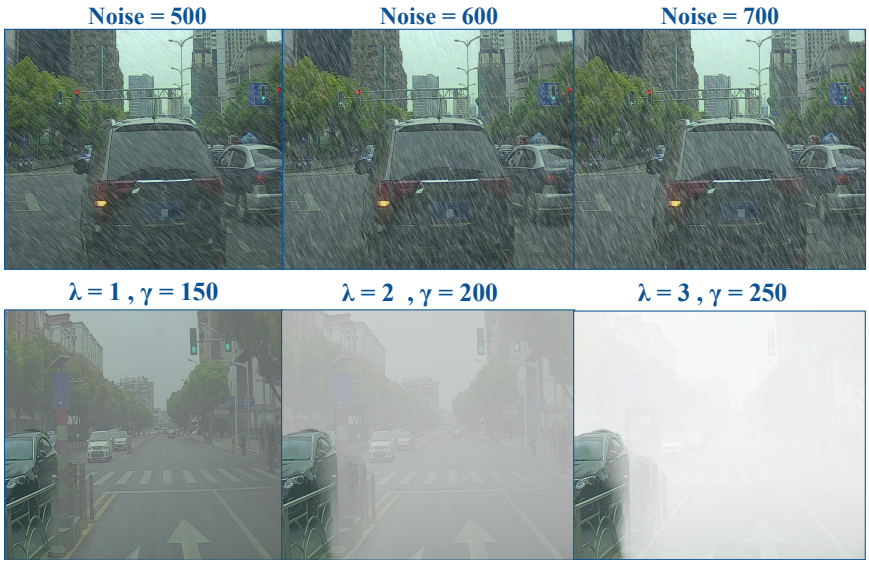
Model	Class	Precision	Recall	mAP50	mAP50-95
YOLOv10n	All	0.86	0.691	0.774	0.46
	Red	0.891	0.724	0.821	0.53
	Green	0.873	0.651	0.745	0.445
	Yellow	0.815	0.698	0.755	0.405
YOLOv8m	All	0.887	0.804	0.873	0.564
	Red	0.937	0.825	0.91	0.641
	Green	0.886	0.76	0.851	0.553
	Yellow	0.839	0.828	0.859	0.499
YOLOv6m	All	0.82	0.757	0.809	0.52
	Red	0.914	0.793	0.877	0.61
	Green	0.875	0.719	0.819	0.529
	Yellow	0.671	0.758	0.729	0.422
YOLOv5m	All	0.856	0.682	0.766	0.481
	Red	0.915	0.687	0.8147	0.549
	Green	0.87	0.658	0.76	0.484
	Yellow	0.784	0.7	0.722	0.41

In order to increase the proportion of yellow images in the merged dataset, several augmentations were applied; as shown in Fig. 1 and Table 2, for the random brightness contrast adjustment, brightness and contrast limits were set to 0.2 (brightness_limit=0.2, contrast_limit=0.2) to modulate the image intensity dynamically. The affine transformation encompasses scaling, translation, rotation, and shearing. The affine transformation included a shear range set between 0 and 20 degrees (shear=(0, 20)). The blur augmentation had a blur limit with a kernel size 7 (blur_limit=7).

As discussed in Section **??**, Fig. 3 illustrates findings which justify the choice of yellow lights as 13%. At 9%, YOLOv5 exhibits underfitting, as indicated by a noticeable gap between training and validation mAPs and overall lower performance compared to other models. This suggests that the limited data volume was insufficient for the model to capture the necessary distributional features, especially given its relatively smaller capacity compared to later versions like YOLOv8 and YOLOv10. Increasing to 13% allowed YOLOv5 to generalize better and close the performance gap, justifying its use in our final training pipeline. The slight performance dip at 13% is not necessarily indicative of model degradation. This behavior can arise due to increased data variance, introducing harder or noisier samples into the training set. The trade-off between marginal mAP fluctuations and improved real-world resilience is a well-documented phenomenon in deep learning. Therefore, selecting 13% strikes a practical balance — it compensates for underfitting in lighter models like YOLOv5 and introduces diversity that benefits overall generalization, even if it momentarily lowers validation mAP on a fixed subset. Such effects are acceptable in deployment-focused applications where robustness outweigh

As mentioned in Section **??**, the hyperparameters employed for addition of fog are λ and Airlight (γ). The amount of fog in an image is determined by λ, while γ controls the brightness of the foggy image. In the case of rain addition, the hyperparameters include noise, rain_len, rain_angle, rain_thickness, and α. Noise determines the density of rain in the image; rain_len specifies the length of each raindrop; rain_angle dictates the angle of the raindrops relative to the vertical axis; rain_thickness defines the thickness of each raindrop; and α sets the opacity of the raindrops. After extensive experimentation, we believe the suitable set of hyperparameters to be $\lambda = 1$, $\gamma = 150$, noise = 500, rain_len = [50,60], rain_angle = [-50,51], rain_thickness = 3, and α = 0.7. Examples of these hyperparameters with different values can be seen in Fig. 2.

Figure 1. A visualization of the various methods used to augment images. The first image from the left is unaltered, the second is horizontally flipped, the third's contrast and brightness have been altered, the fourth has an affine transformation, and the last has been blurred. The hyperparameters for these augmentations are mentioned in Table 2 .



i

Figure 2. The first row of images has rain added to them, and the second row has fog added to them. As evident from the images above, the set of hyperparameters, namely Noise=500, λ=1, and γ=150, result in a realistic and balanced effect.

Table 2. The list of augmentations applied to address the class imbalance and the hyperparameters employed in the process. All these hyperparameters were used with a probability of 0.5.

Augmentation	Hyperparameters
Horizontal Flip	p=0.5
Random Brightness Contrast	brightness_limit=0.2, contrast_limit=0.2
Affine	shear=(0,20)
Blur	blur_limit=7

0.3. Benchmarking

This section provides a summary of the different YOLO models used in the experiments, highlighting their architectural differences and unique characteristics.

YOLOv5 starts with a strided convolution layer to reduce memory and computational costs, and the SPPF (Spatial Pyramid Pooling Fast) layer accelerates computation by pooling multi-scale features into a fixed-size feature map.

YOLOv6 introduced a new backbone called EfficientRep [3], built on RepVGG [1], which offered greater parallelism than previous YOLO backbones. The network's neck used a PAN (Path Aggregation Network) structure, enhanced with RepBlocks or CSPStackRep Blocks for larger models. This redesigned backbone and neck significantly improved the model's efficiency and adaptability.

YOLOv8 retains a backbone architecture similar to YOLOv5 but introduces significant changes to the CSPLayer, now called the C2f module. This module, which stands for "cross-stage partial bottleneck with two convolutions," effectively merges high-level features with contextual information, resulting in improved detection accuracy.

YOLOv10 introduces a lightweight classification head

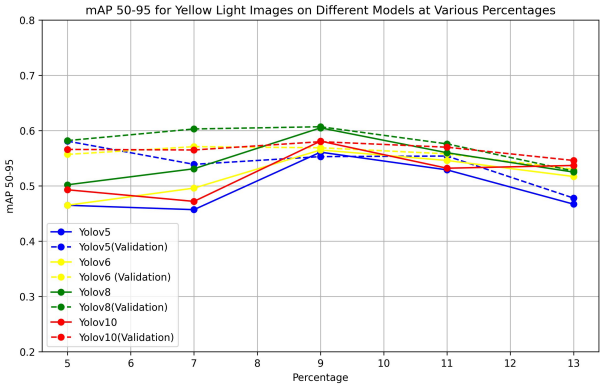


Figure 3. Benchmarked results of various models on various percentages of yellow lights. It is observed that all models converge at 13%, indicating that the class imbalance is abated. Here, the percentage on the x-axis refers to the percentage of yellow lights in the merged dataset.

with depth-wise separable convolutions to reduce computational overhead, Spatial-Channel Decoupled Downsampling to minimize information loss, and Rank-Guided Block Design for optimal parameter use. Accuracy is enhanced through Large-Kernel Convolution for better feature extraction and Partial Self-Attention (PSA) for improved global representation with minimal overhead. It also uses dual label assignments with a consistent matching metric, allowing for rich supervision and efficient deployment without the need for NMS, an integral component of previous YOLO versions.

Each of these YOLO versions differs in its architectural innovations and optimizations, catering to different aspects of object detection challenges. All aforementioned models were trained using the Adam optimizer [2] for 50 epochs on the NVIDIA RTX A6000 GPU with a batch size of 32.

The models are benchmarked by training them on the clean images without rain or fog and testing them on the images that have rain/fog in them. The results of this benchmarking process, which highlight the impact of training on clean images and testing on weather-degraded images, are presented in Table 1.

0.4. Additional Results

In Tables 1 and 3, two scenarios are presented: (1) when models are trained on D^{s} but tested on D^{t}, and (2) when models are trained on $D^{s \rightarrow t}$ and evaluated on D^{t}, for which the subplots are shown for representational purposes in the main paper in Fig. **??** and the exact values for which are mentioned in Table 3.

Table 3. Comparisons of results across models when trained on $\mathrm{D}^{s \rightarrow t}$ and inferred on D^{t} (rainy and foggy). All metrics are averaged across the 3 classes.

Model	Beta(β)	Precision	Recall	mAP50	mAP50-95
YOLOv10n	0.15	0.911	0.85	0.927	0.617
	0.1	0.903	0.83	0.903	0.58
	0.05	0.935	0.923	0.964	0.682
YOLOv8m	0.15	0.933	0.923	0.956	0.674
	0.1	0.927	0.877	0.927	0.633
	0.05	0.882	0.693	0.767	0.495
YOLOv6m	0.15	0.914	0.856	0.914	0.585
	0.1	0.890	0.772	0.843	0.516
	0.05	0.932	0.919	0.954	0.654
YOLOv5m	0.15	0.926	0.878	0.928	0.621
	0.1	0.887	0.765	0.826	0.532
	0.05	0.850	0.646	0.724	0.439

References

[1] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again, 2021. 2

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[3] Kaiheng Weng, Xiangxiang Chu, Xiaoming Xu, Junshi Huang, and Xiaoming Wei. Efficientrep: An efficient repvgg-style convnets with hardware-aware neural network design. *arXiv preprint arXiv:2302.00386*, 2023. 2