Deep Learning Final Project

Blind Motion Single Image Deblurring

Group 25
313510156 陳柏翔 313510217 康鈞睿

大綱 Outline

1. Introduction
   a. Blind Motion Image Deblurring
   b. Our Contribution
2. Background
   a. FFTformer
   b. MLWNet
3. Improvement
   a. HybridNet
   b. Channel-Efficient MLWNet
4. Experiments
   a. Performance Comparison
   b. Complexity Comparison
   c. Visualization
5. Conclusion

Introduction

Introduction – Blind Motion Image Deblurring

Deblur a blurry image caused by camera motion, using only a single blurry image and no additional information.

Sharp Image

Blurry Image

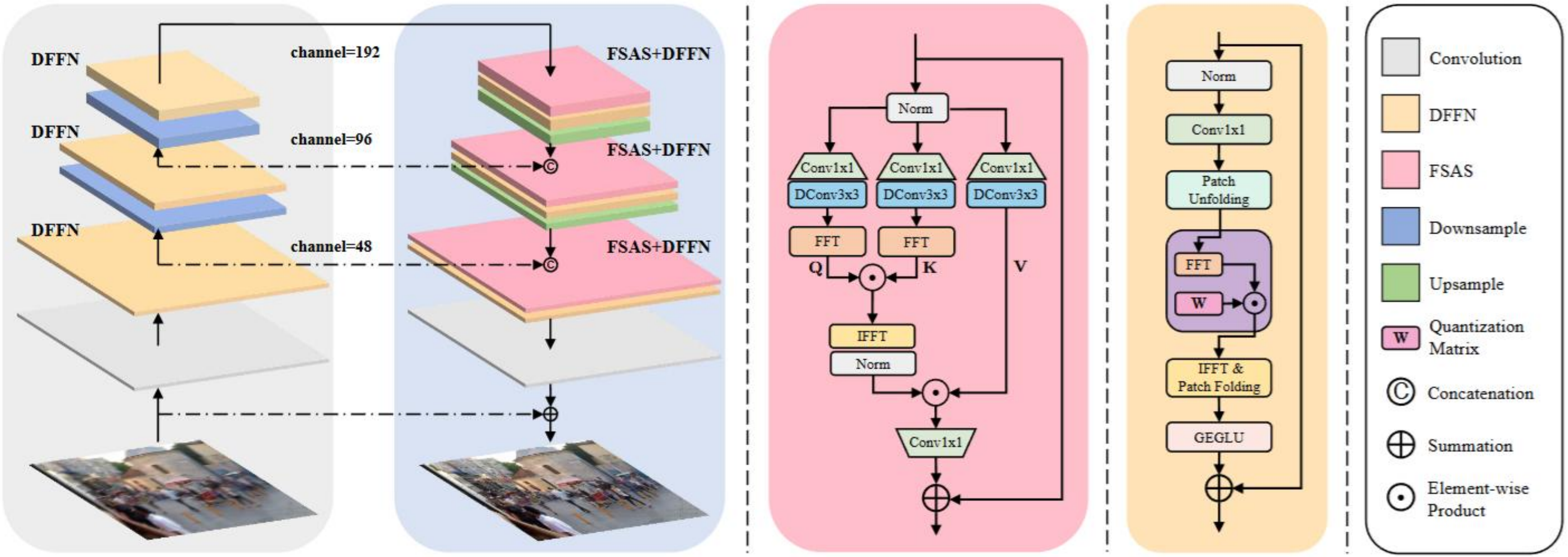
Estimated Sharp Image

Introduction – Our Contribution

1. [Topic 3] Implement and compare different papers in a specific domain.
   a. Reimplement the complete training flow and networks.
   b. Compare MLWNet and FFTformer.
2. [Topic 1] Implement a paper and improve it.
   a. Propose HybridNet (Simple Backbone with FFTformer's DFFN).
   b. Propose CE-MLWNet (Improved MLWNet).

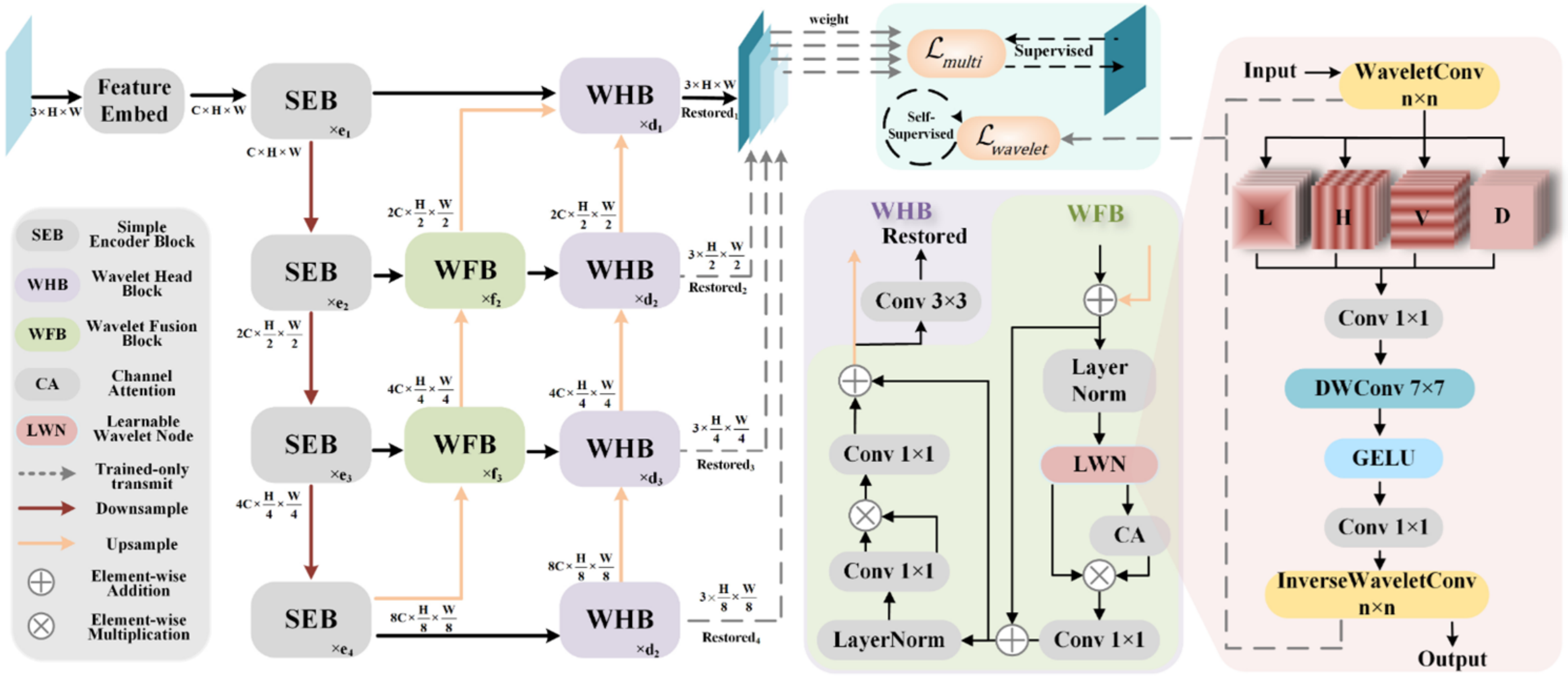
Background

Background – FFTformer



[1] Gao, Ning, et al. "Efficient frequency-domain image deraining with contrastive regularization.", ECCV, 2024.

Background – MLWNet

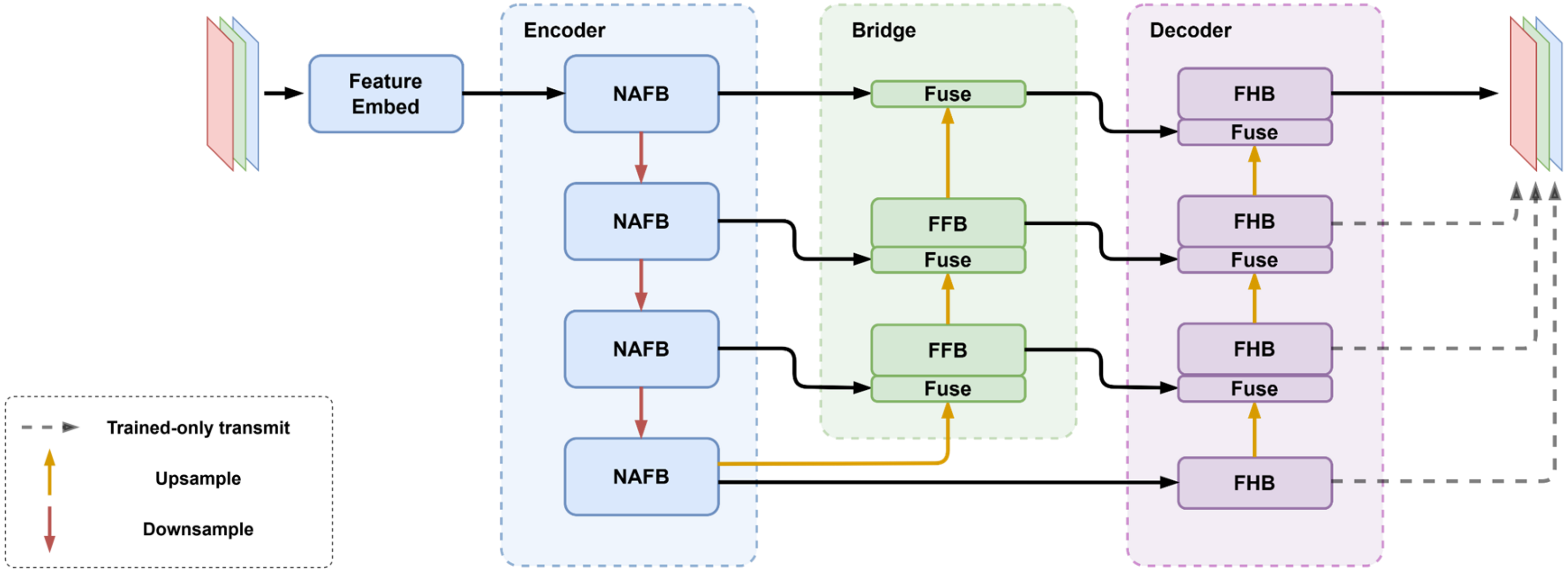


[1] Gao, Xin, et al. "Efficient multi-scale network with learnable discrete wavelet transform for blind motion deblurring.", CVPR, 2024.

Improvement

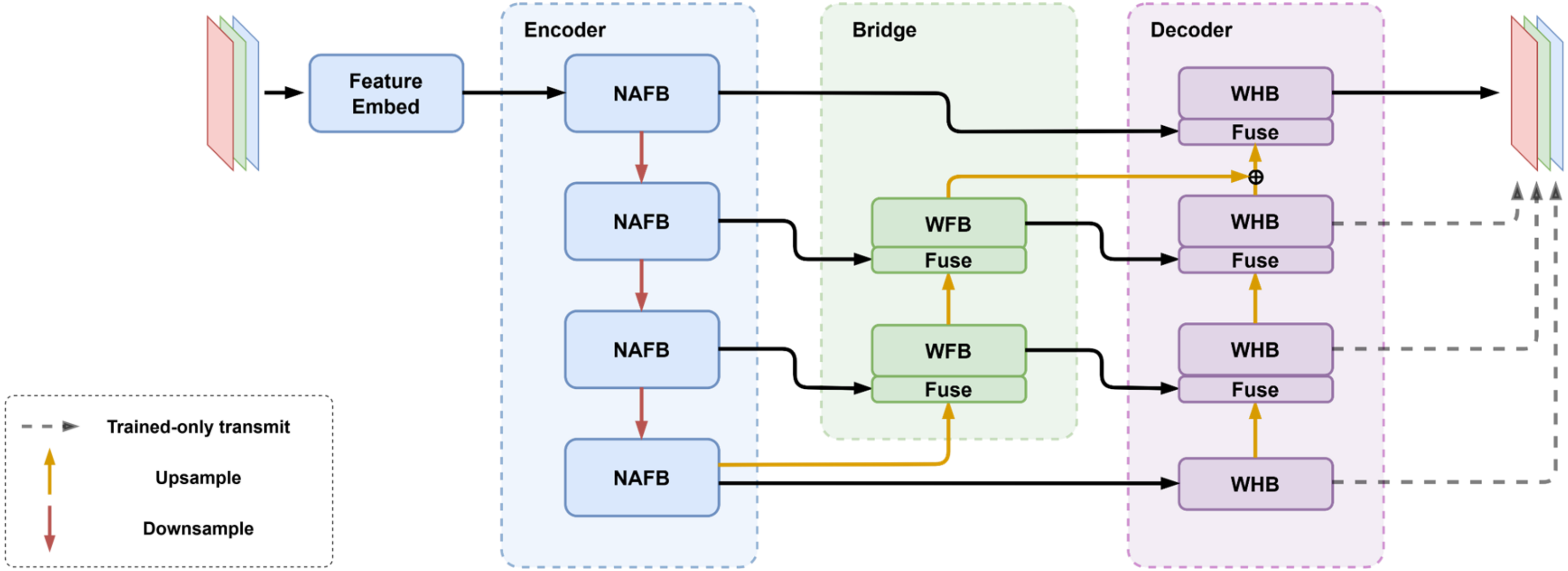
Improvement – HybridNet

- Architecture: Similar to MLWNet, consists of an encoder, a decoder, and a bridge module.
- Encoder: NAFNet-style Simple Block.
- Bridge / Decoder: NAFNet-style Simple Block enhanced with the DFFN in FFTformer.

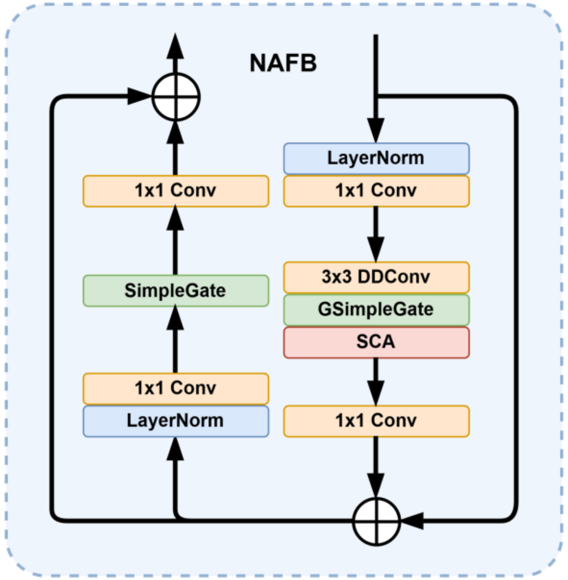


Improvement – Channel-Efficient MLWNet

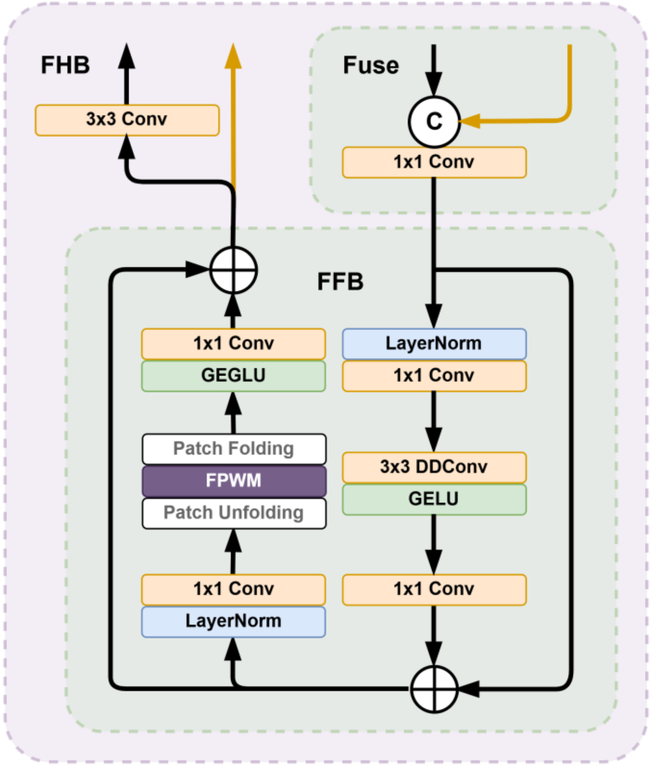
- Feature Fusion (Channel concatenation and reduction)
- Efficient Convolution Design (Diverse dilated convolution and seperable convolution)
- Encoder Simplification (Reduce the number of blocks in encoder module)



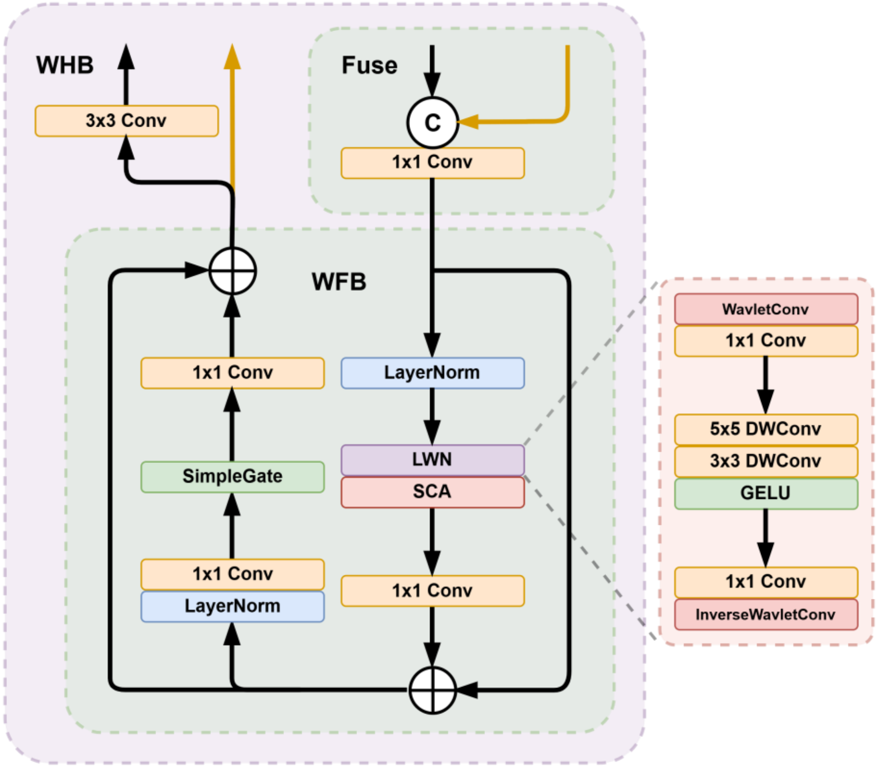
Improvement – NAFB, FF(H)B, WF(H)B



NAFNet-style Block
(Used in both networks, same as SEB in MLWNet)



Frequency Fusion (Head) Block
(Used in HybridNet)

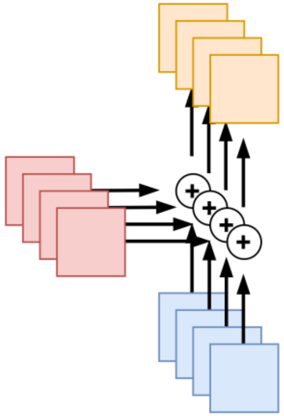


Wavelet Fusion (Head) Block
(Used in CE-MLWNet)

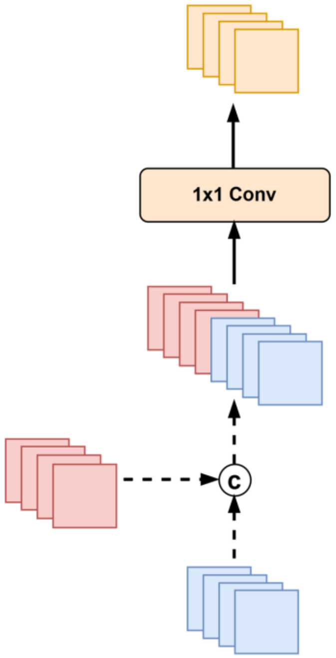
Improvement – Channel Concatenation + Reduction

- **Feature Fusion:** Replace channel addition with channel concatenation followed by point-wise convolution for channel reduction.

$$\mathbf{Y} = \mathbf{X}_1 + \mathbf{X}_2$$



Channel-wise Addition



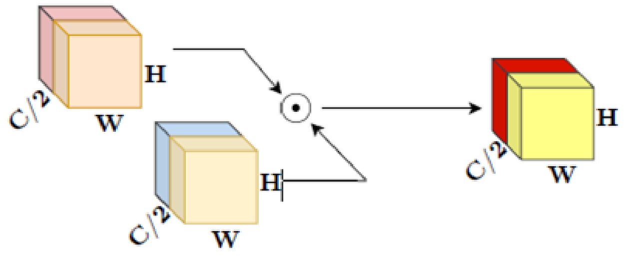
Channel Concatenation + Reduction

$$\mathbf{Y} = \mathbf{W}\begin{bmatrix}\mathbf{X}_1 \\ \mathbf{X}_2\end{bmatrix} + \mathbf{B}$$

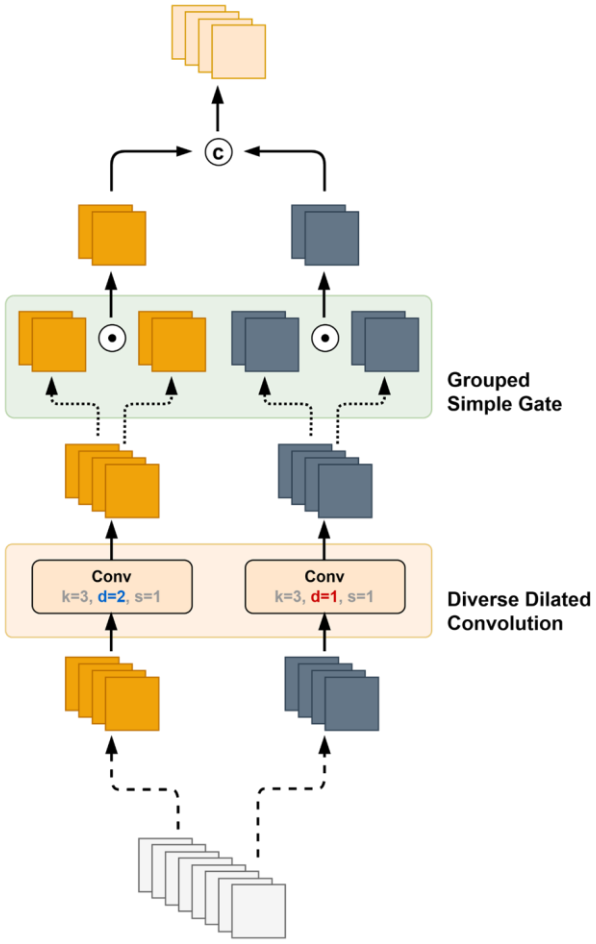
$\mathbf{Y}, \mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^{C \times HW}$ $\mathbf{W} \in \mathbb{R}^{C \times 2C}$ $\mathbf{B} \in \mathbb{R}^{C}$

Improvement – DDConv and GSimpleGate

- **Efficient Convolution Design:** Enhance the SEB using diverse dilated convolutions (d = 1, 2 in this work) combined with a grouped simple gate.

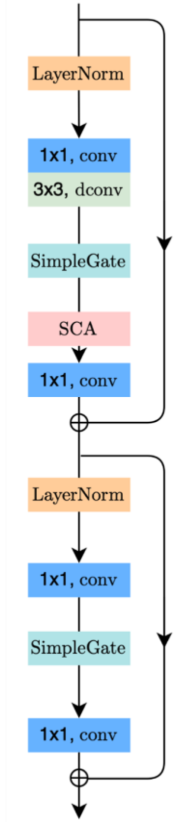


Simple Gate in [1]

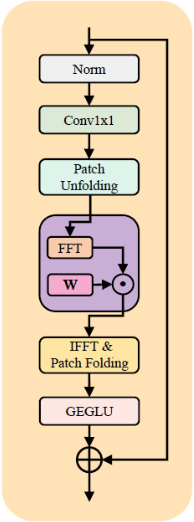


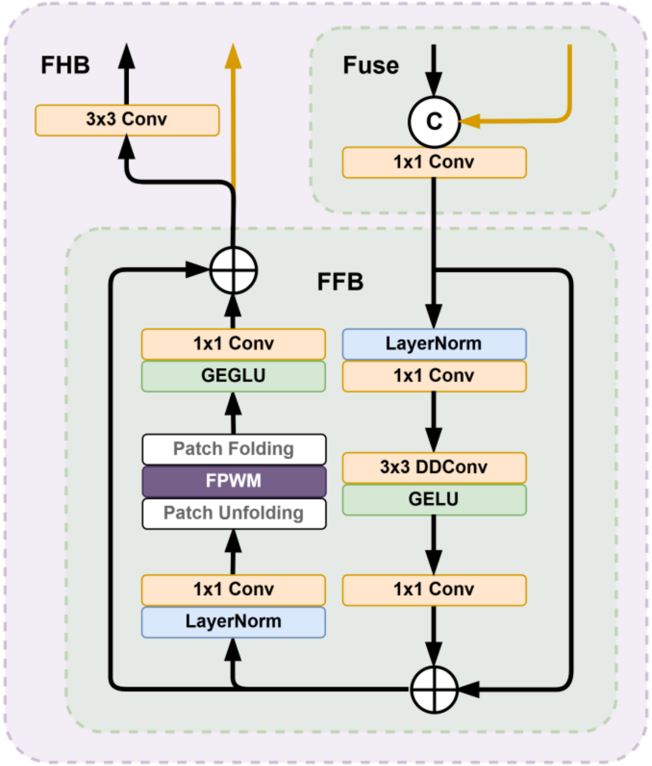
[1] Chen, Liangyu, et al. "Simple baselines for image restoration.", ECCV. Cham: Springer Nature Switzerland, 2022.

Improvement (HybridNet) – Simple Block with DFFN



+

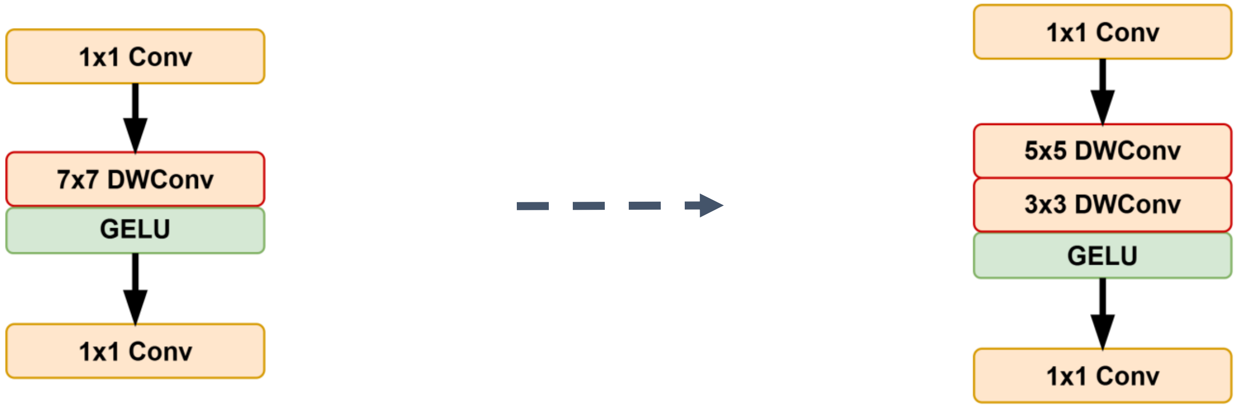




[1] Gao, Ning, et al. "Efficient frequency-domain image deraining with contrastive regularization.", ECCV, 2024.
[2] Chen, Liangyu, et al. "Simple baselines for image restoration.", ECCV. Cham: Springer Nature Switzerland, 2022.

Improvement (CE-MLWNet) – Seperable Convolution

- **Efficient Convolution Design:** Substitute the original 7×7 convolution with separable convolutions (5×5 + 3×3).



- **Params = 7×7= 49**
- **Computation = 49×HWC**
- **Receptive Field = 7**

- **Params = 5×5+3×3 = 34**
- **Computation = 34×HWC**
- **Receptive Field = 1+(5-1)+(3-1) = 7**

Improvement (CE-MLWNet) – Progressive Learning

As in the progressive learning strategy of [1], CE-MLWNet is trained on 256×256 patches and subsequently fine-tuned on 512×512 patches.

[1] Zamir, Syed Waqas, et al. "Restormer: Efficient transformer for high-resolution image restoration.", CVPR,. 2022.

Experiments

Experiments – Performance Comparison

Patch / Overlap Size	256 / 128		512 / 256		Full Image Size	
Metric	RSNR	SSIM	RSNR	SSIM	RSNR	SSIM
FFTformer [1]	31.75722	0.92060	**31.83560**	**0.92203**	31.74532	0.92087
MLWNet [2]	**32.21568**	0.91880	32.12404	**0.92046**	32.04214	0.91964
HybridNet	31.80117	0.92110	**31.91366**	**0.92299**	31.50054	0.91809
CE-MLWNet	31.84416	0.91699	**32.41885**	**0.92261**	31.88704	0.91902

Table. Testing Results on RealBlur-J Dataset.

[1] Gao, Ning, et al. "Efficient frequency-domain image deraining with contrastive regularization.", ECCV, 2024.
[2] Gao, Xin, et al. "Efficient multi-scale network with learnable discrete wavelet transform for blind motion deblurring.", CVPR, 2024.

Experiments – Complexity Comparison

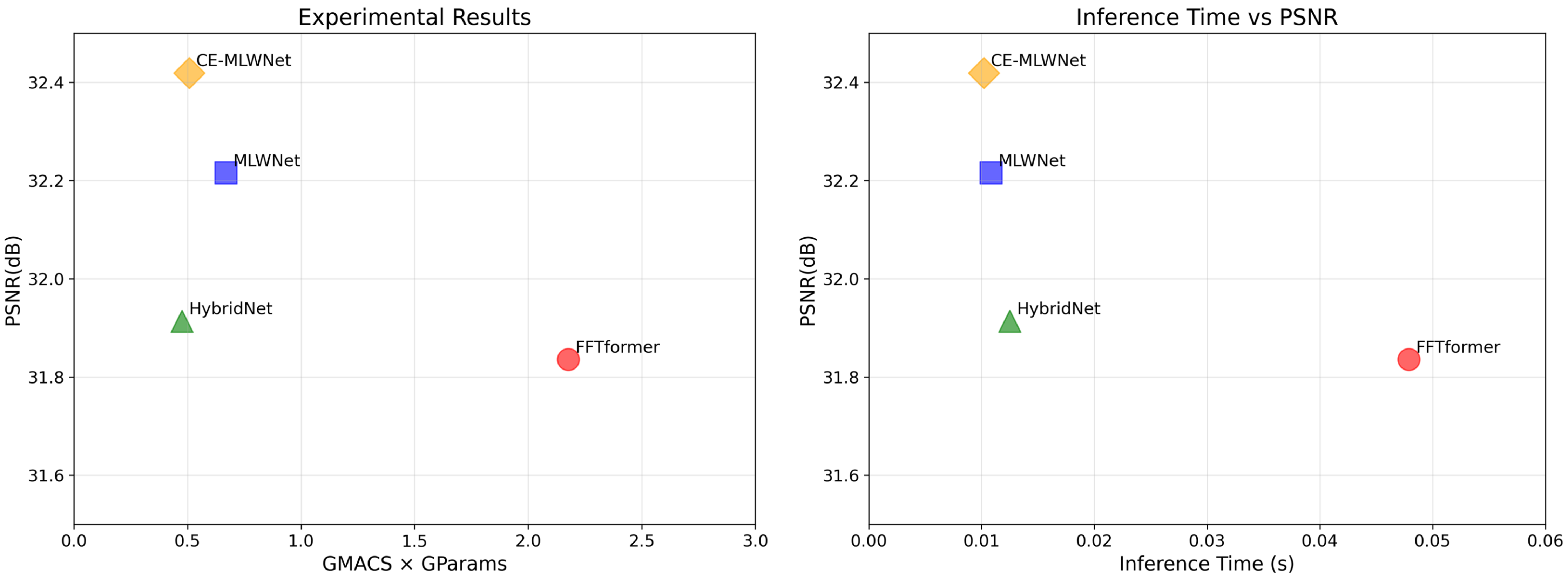
Property	GMACs	Model Size	Inference Time
FFTformer [1]	131.44	16,560,474	47.898 ms
MLWNet [2]	27.78	24,109,164	10.843 ms
HybridNet	30.40	**15,626,243**	12.490 ms
CE-MLWNet	**24.95**	20,330,531	**10.184 ms**

Table. Comparison of Model Complexity.

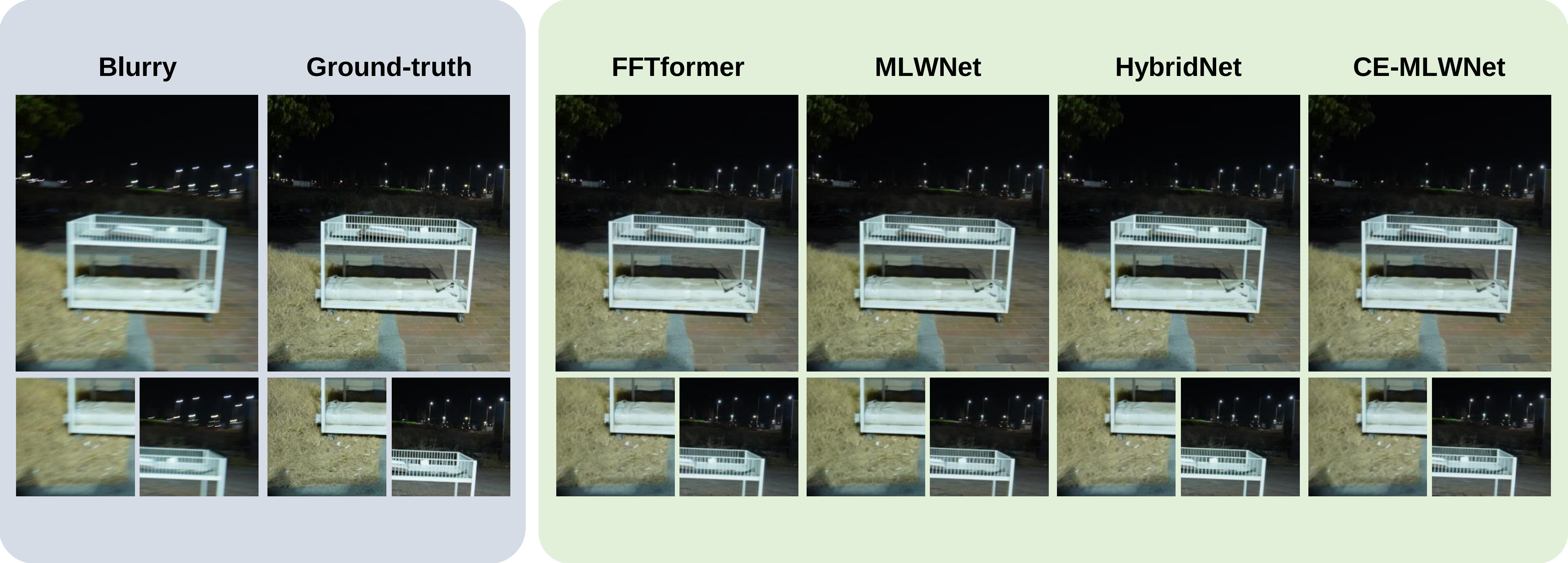
(MACs are measured on 256×256 input patches, and inference time is evaluated on an RTX 5090 GPU.)

[1] Gao, Ning, et al. "Efficient frequency-domain image deraining with contrastive regularization.", ECCV, 2024.
[2] Gao, Xin, et al. "Efficient multi-scale network with learnable discrete wavelet transform for blind motion deblurring.", CVPR, 2024.

Experiments – Performance Comparison



Experiments – Visualization (RealBlur-J Test Set)



Experiments – Visualization (RealBlur-J Test Set)

Blurry | Ground-truth | FFTformer | MLWNet | HybridNet | CE-MLWNet

Experiments – Visualization (Demo Results)

Blurry Image

Deblurred (CE-MLWNet)

Blurry-Deblurred GIF

Conclusion

Conclusion

1. [Topic 3] Implement and compare different papers in a specific domain.
    a. MLWNet has better PSNR than FFTformer, but more parameters.
    b. Although MLWNet has more parameters than FFTformer, its inference time is significantly lower due to the large difference in MACs.
2. [Topic 1] Implement a paper and improve it.
    a. Proposed HybridNet has best SSIM with lowest parameters
    b. Propose CE-MLWNet has best PSNR with lowest GMACs and inference time.

Thanks

Appendix

Links

- **Our Github Repository:**
  https://github.com/EEGuizhi/Single-Image-Blind-Motion-Deblurring.git
- **Pretrained Model Weights:**
  https://drive.google.com/drive/folders/1symu2hEiHB679yDPjsiQWi6pAvcd4_6w?usp=drive_link

Notes

- 由於 Final Project 時間不夠充足，我們雖然有準備 GoPro Dataset 相關的程式碼，但並沒有在 GoPro Dataset 上進行訓練或測試。
- 如果想要測試在自己拍攝的影像上，由於我們提供的 Pretrained Models 皆訓練在 RealBlur-J Dataset 上，影像全圖大小約為 700x700 左右 (並非指 Patch Size)。如果是日常手機拍攝之影像的話，通常解析度很高、影像邊長超過 1000，建議先將圖片縮放到長邊為 768 左右再進行還原較佳 (否則會因為模型沒有學過如此大的 Blur Kernel 而很難還原)。
- 訓練模型用 RTX 5090 約 26hr ~ 58hr 等，測試時間約需 15min ~ 20min。
- 此 Final Project 復現結果與原 MLWNet, FFTformer 差距約 0.7dB，是由諸多因素所構成的，例如：我們是完全從頭實作訓練環境，而非使用 BasicSR 作為框架；訓練時長相對不夠長；測試時的 ECC 疊代次數不夠多等。(但訓練時長的設置在每個 Networks 間皆為相同等級，以確保公平性。)

Notes

- 在程式碼中為 “Network” 名稱的東西皆對應此報告中所述之 “CE-MLWNet”。
- 之所以稱為 Channel-Efficient 是因為我們主要改動了 Channel 間特徵混和的方式，並且在不同 Channels 上採用不同 Dilation 可以得到更加多樣的特徵，而 Grouped Simple Gate 也是在 Channel 方向上進行 Group；綜上原因我們稱改良後的 MLWNet 為 CE-MLWNet。

Test Inference Time & Model Information

- 測試 Inference Time 時使用 CUDA Event 並搭配 Warm up 的方式來進行量測 (如右程式碼)。
- 測試 Model Complexity 所需的模型資料時，則是藉由 torchinfo 來進行量測 (如下程式碼)。

```
summary = torchinfo.summary(
     model,
     input_data=(torch.randn(1, 3, img_size[0], img_size[1]).to(DEVICE)),
     col_names=["input_size", "output_size", "num_params", "trainable"],
     depth=4,
 )
```

```
def test_inference_time(
    model: nn.Module,
    img_size: tuple[int, int],
    device: torch.device,
    iterations: int = 100,
    warmup: int = 20,
) -> float:
    """Measure the average inference time of the model (ms).
    Uses CUDA events if device is GPU, otherwise uses perf_counter for CPU.
    """
    # Initialization
    model.eval()
    model.to(device)
    times = []
    input_tensor = torch.randn(1, 3, img_size[0], img_size[1], device=device)

    # Warm-up
    with torch.no_grad():
        for _ in range(warmup):
            _ = model(input_tensor)

    # GPU timing
    if device.type == "cuda":
        starter = torch.cuda.Event(enable_timing=True)
        ender = torch.cuda.Event(enable_timing=True)
        with torch.no_grad():
            for _ in range(iterations):
                starter.record()
                _ = model(input_tensor)
                ender.record()

                torch.cuda.synchronize()
                times.append(starter.elapsed_time(ender))  # milliseconds
    # CPU timing
    else:
        with torch.no_grad():
            for _ in range(iterations):
                start = time.perf_counter()
                _ = model(input_tensor)
                end = time.perf_counter()
                times.append((end - start) * 1000)  # milliseconds
    return float(np.mean(times))
```

Image Alignment Before Testing

- 若直接將模型所預測的影像與 Ground-truth 影像逐像素進行比較，其評估結果通常會顯著偏低 (約為 30 dB)。其主要原因在於模糊影像本質上是由長時間曝光或多次取樣所造成，對應到的是一條 “連續” 的運動模糊軌跡。
- 而模型所還原的清晰影像，理論上可以對應於該模糊軌跡上的任意一個時刻。因此，在未進行對齊的情況下直接進行誤差計算，等同於將一張「雖然視覺上清晰，但在空間位置上未對齊」的影像與 Ground-truth 進行比較，進而導致評估指標被不合理地低估。
- 測試時我們採用 RealBlur Dataset 評估方法相同的方式進行 PSNR, SSIM 準確率量測，會先使用 OpenCV 的 cv2.findTransformECC, cv2.warpPerspective 等函式進行影像對齊。而我們為了加快測量時間，在 ECC 過程中的 iteration 數設置為 50 而非 100。