NTIRE 2025 Efficient Burst HDR and Restoration FlowCNRDB

Simon J. Larsen
Esoft
Kochsgade 31D, 5000 Odense, Denmark
sjl@esoft.com

1. Team details

- Team name: SimonLarsen
- Team leader name: Simon J. Larsen
- Team leader contact:
  - Address: Kochsgade 31D, 5000 Odense, Denmark
  - Phone number: +45 30 51 59 27
  - Email: sjl@esoft.com
- Rest of the team members: None
- Affiliation: Esoft
- User names on Codalab:
  - Simon Larsen: `SimonLarsen`
- Best entries in Codalab:
  - Development phase: PSNR: 41.74. SSIM: 0.990.
  - Testing phase: PSNR: 41.38. SSIM: 0.989.
- Code and model weights: https://github.com/SimonLarsen/ntire2025_rawfusion_submission

2. Methodology

2.1. Network architecture

In recent years, state-of-the-art image restoration models have largely been built around efficient self-attention methods–most notably shifted window self-attention (Swin) [4]. However, although window-based methods avoid the quadratic complexity of global self-attention, Swin transformers are comparatively complex and lack the inductive biases associated with convolutional neural networks.

This approach instead aims to build a modern fusion and restoration model by applying the design principles outlined in ConvNeXt [5] and ConvNeXt V2 [7] to produce an efficient and fully-convolutional model. The overall network architecture consists of three stages: feature extraction, frame alignment, and finally a joint fusion and restoration model. In the first and second stages, each frame is processed in parallel, while the last stage processes all frames together to produce the final output. The overall architecture is shown in Figure 1.

All stages of the model use the basic block design of ConvNeXt V2 employing large kernel depth-wise convolutions followed by a point-wise MLP and global response normalization (GRN). To avoid overfitting, dropout is used throughout the model. The global feature aggregation in GRN will lead to train-test time inconsistency when trained on patched inputs. To mitigate this issue, a local aggregation method similar to test-time local conversion [1] is used during inference.

2.2. Feature extraction

The feature extraction stage encodes each frame $\mathbf{I}_i \in \mathbb{R}^{H\times W\times 1}$ into a latent representation $\mathbf{Z}_i \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times C}$. The frames are first patched using a 2×2 convolution with stride 2. The patch size is chosen to match the GRBG filter size of the raw Bayer inputs. Deep features are then extracted through a series of ConvNeXt (CN) blocks.

2.3. Frame alignment

The frame alignment model is a U-Net architecture [6] consisting of CN blocks. Given an encoded reference and target frame $(\mathbf{Z}_{\text{ref}}, \mathbf{Z}_i) \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 2C}$ (concatenated on the channel axis), the model outputs a dense flow field $\mathbf{F}_i \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 2}$ estimating the displacement between the reference and the target.

The predicted flow field is used to warp the target frame to match the reference frame using bilinear grid sampling. Tokens outside the target frame are padded with zeros, and a padding mask $\mathbf{M}_i \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 1}$ is computed to be passed on to the fusion model along with the warped features.

Sparse methods for frame alignment, such as predicting a homography matrix, were also explored, but the dense flow fields provided better validation results during early experimental testing and was ultimately chosen for this reason.

2.4. Fusion and restoration

The last stage takes in the full set of aligned frames and padding masks, concatenated on the channel axis, and predicts the final RGB output image $\hat{\mathbf{Y}} \in \mathbb{R}^{H\times W\times 3}$. If the

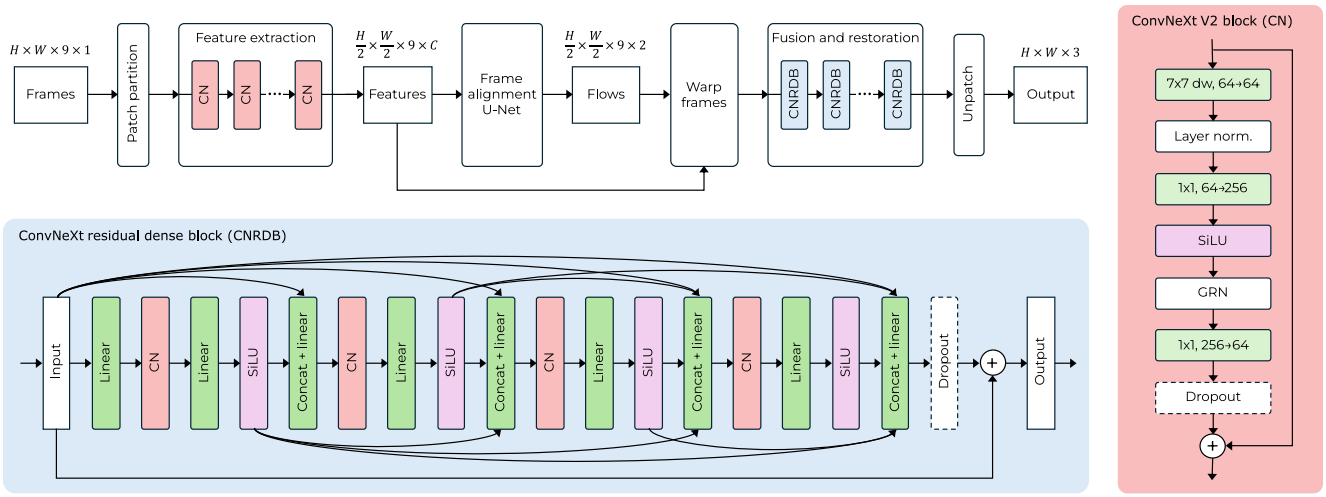

Figure 1. Overall architecture of the FlowCNRDB model.

frames were naïvely passed to the model in the input order during training, the model would converge to using only the reference frame while discarding the remaining frames due to the lack of supervision in the alignment stage. To mitigate this issue, the frames are randomly shuffled within the three exposure time groups during training. This effectively prevents the model from associating specific channels with the reference frame, forcing the model to utilize all frames and, in turn, learn how to align the frames in the second stage.

The fusion stage consists of a series of ConvNeXT residual dense blocks (CNRDB) using a macro design similar to DRCT [2], but replacing Swin transformer layers with CN blocks. In addition, a linear layer is used to project the number of channels down to a common size before each CN block to constrain the number of parameters. The output from the fusion stage is unpatched with a 2×2 pixel shuffle operation followed by a 3×3 convolution down to 3 channels to produce the final output image.

3. Training setup

3.1. Datasets

The model was trained solely on the training data set provided. The first 280 scenes (scenes 000 through 279) were used for training, while the remaining 20 scenes (scenes 280 through 299) were used for validation. The following Bayer-preserving augmentations [3] were used during training to improve generalization: horizontal/vertical flipping, rotation (in 90 degree increments), and random cropping. In addition, frame shuffling was used as detailed in Section 2.4).

3.2. Model training

The full model had 18.65 million parameters. The model was trained solely with MSE loss

$$\ell_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2 \tag{1}$$

No additional losses were used to supervise the frame alignment.

The model was trained at 256×256 resolution with batch size 10 for 224k iterations on a single NVIDIA RTX A6000 GPU using automatic mixed precision to reduce VRAM consumption. The initial learning rate was set at 3×10^{-4} and gradually reduced to 10^{-7} using a cosine annealing schedule. Throughout training, the AdamW optimizer was used with weight decay $= 0.01$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Training loss, learning rate schedule and validation metrics are shown in Figure 2. The model configuration details are outlined in Table 1.

4. Other details

NTIRE 2025 workshop submission

Not planned at the time of writing.

General comments and impressions of the NTIRE 2025 challenge

I want to thank the organizers for organizing a fun and well-organized challenge.

Other comments

The imposed limits on the number of model parameters and FLOPs were reasonable for a challenge on efficient on-device methods. However, the chosen method for estimating FLOP count, namely using *FlopCountAnalysis* from

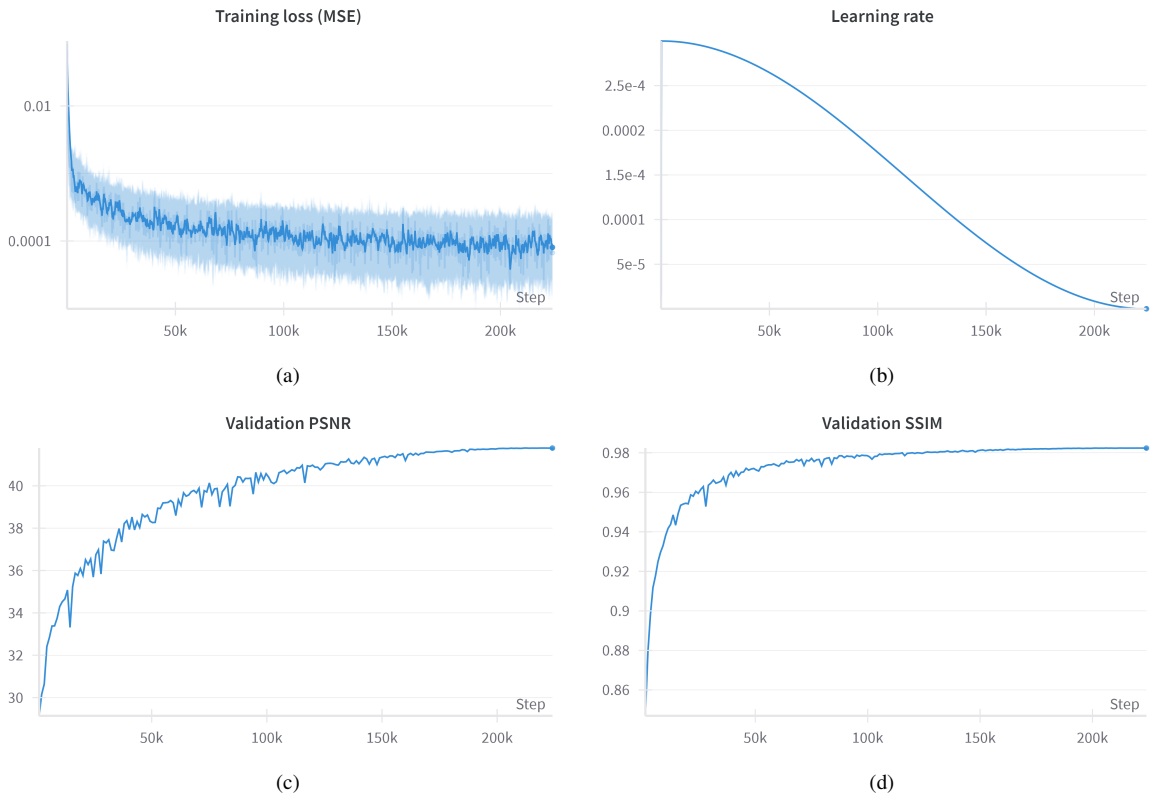

Figure 2. Model training loss (a) (log-scaled), learning rate schedule (b), and validation metrics (b-c). Metrics were computed for 256×256 patches from the reserved 20 scenes from the training dataset.

fvcore, was imprecise and a source of confusion for some participants. Future challenges should consider alternative methods for imposing computational constraints that are both objective and fair.

References

[1] Xiaojie Chu, Liangyu Chen, Chengpeng Chen, and Xin Lu. Improving image restoration by revisiting global information aggregation. In *European Conference on Computer Vision*, pages 53–71. Springer, 2022. 1

[2] Chih-Chung Hsu, Chia-Ming Lee, and Yi-Shiuan Chou. Drct: Saving image super-resolution away from information bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6133–6142, 2024. 2

[3] Jiaming Liu, Chi-Hao Wu, Yuzhi Wang, Qin Xu, Yuqian Zhou, Haibin Huang, Chuan Wang, Shaofan Cai, Yifan Ding, Haoqiang Fan, et al. Learning raw image denoising with bayer pattern unification and bayer preserving augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[4] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 1

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 1

[7] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16133–16142, 2023. 1

Stage 1: Feature extraction	
Kernel size	5
Width	32
Depth	5
Expand ratio	2.5
Dropout	0.05
Stage 2: Frame alignment	
Kernel size	7
Widths	32, 64, 128, 256, 512
Depths	2, 3, 4, 5, 6
Expand ratio	2.5
Dropout	0.1
Stage 3: Fusion and restoration	
Kernel size	7
Base channels	64
Grow channels	32
Depth	14
Expand ratio	2.0
Dropout	0.1

Table 1. Model configuration. Depth is the number of CN blocks in stage 1 and 2, and the number of CNRDB blocks in stage 3. Expand ratio is the