Masked Pre-training Enables Universal Zero-shot Denoiser

Xiaoxiao Ma[1], Zhixiang Wei[1], Yi Jin[1], Pengyang Ling[1,2], Tianle Liu[1],
Ben Wang1, Junkang Dai[1], Huaian Chen[1]

[1]University of Science & Technology of China, [2]Shanghai AI Laboratory

NEURAL INFORMATION
PROCESSING SYSTEMS

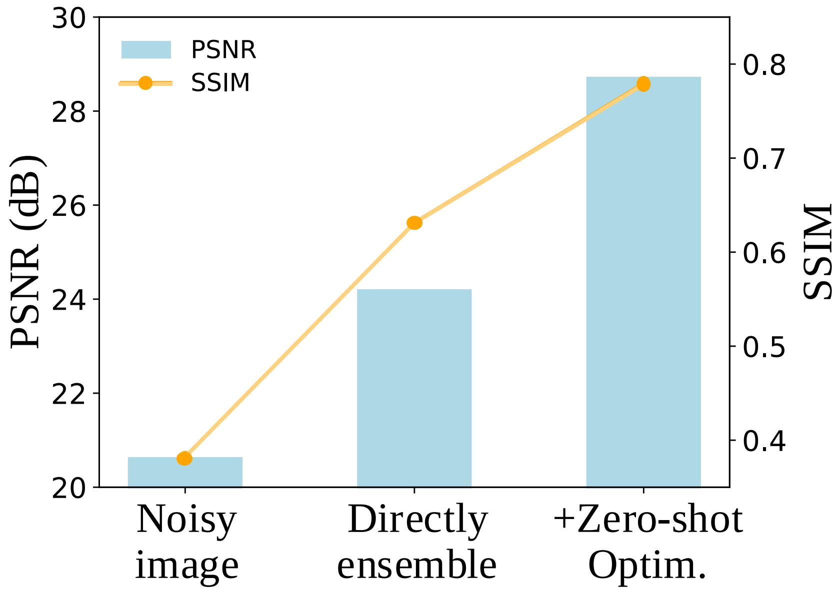
上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

NeurIPS 2024

Observations

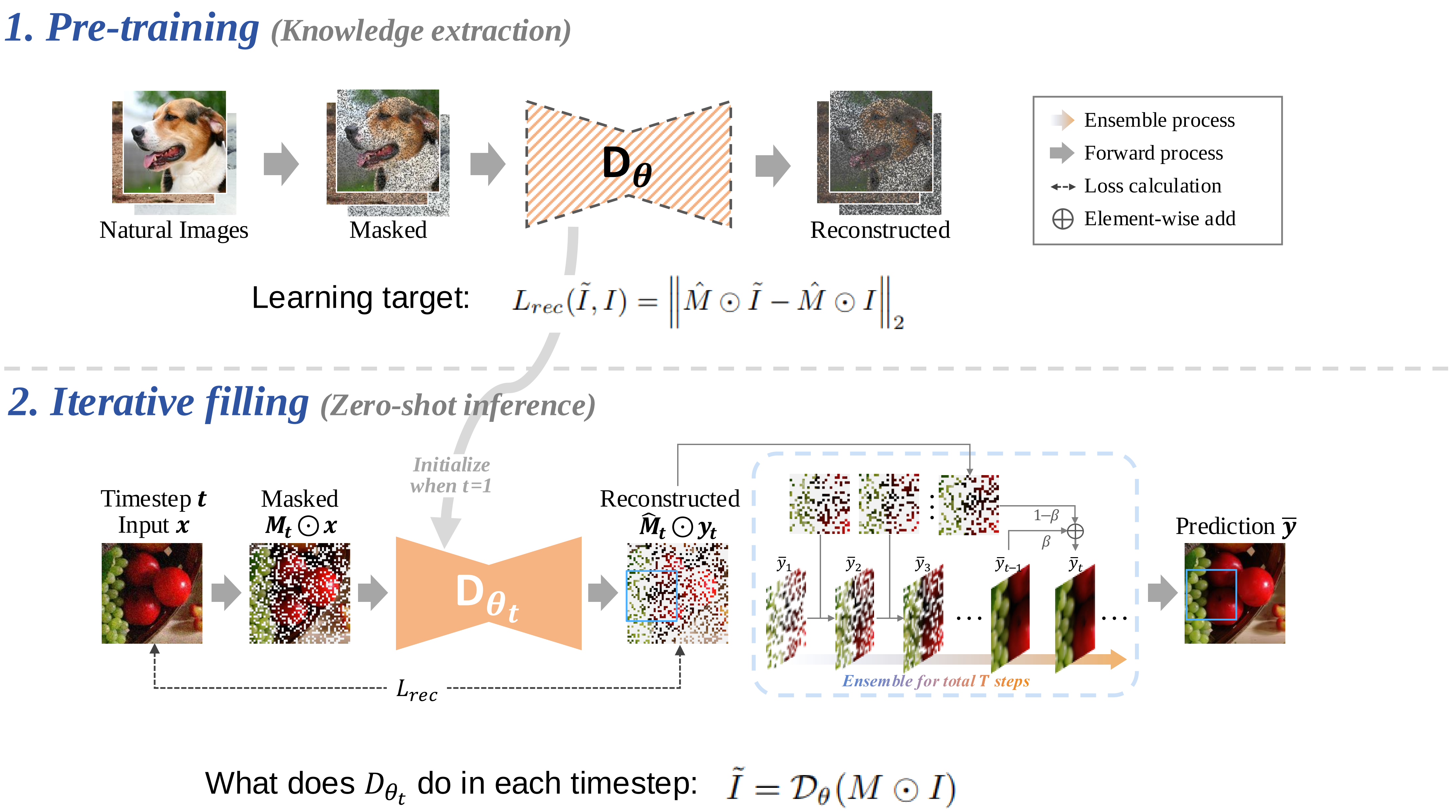
- ◆Masked Image Modeling enhances computer vision by learning from large natural image datasets.
- ◆A simple average of predictions from reconstruction of **pre-trained model with masks** on a noisy image can naturally denoise!

Noisy | Reference　Directly ensemble　+ Zero-shot Optim.



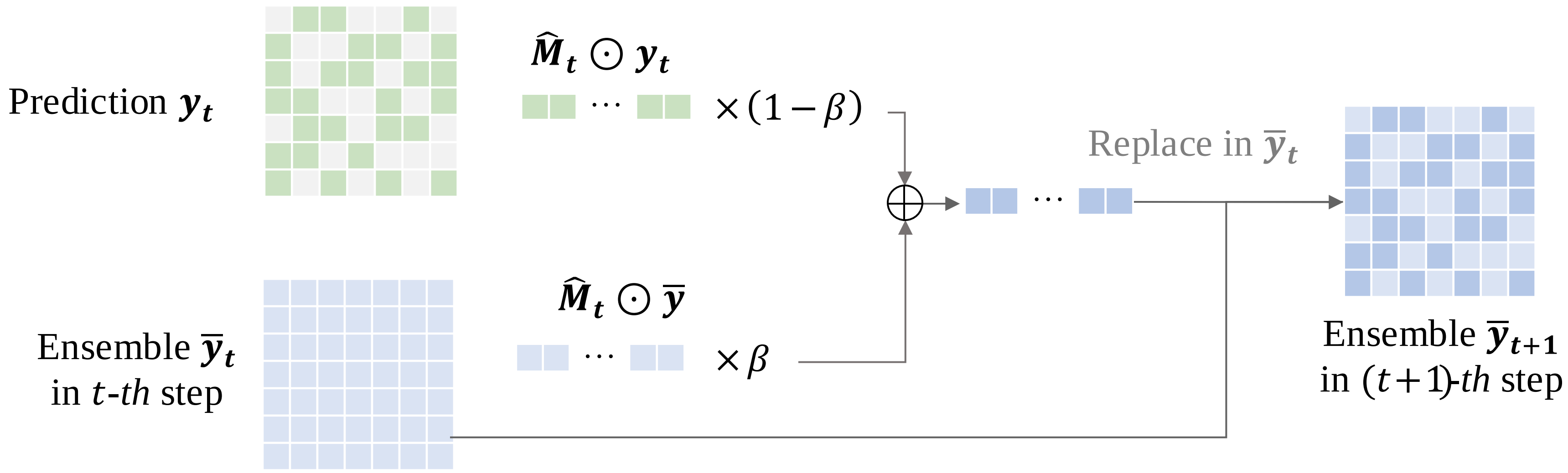
So we propose…

- A pretrained model on **large-scale datasets** (e.g. ImageNet)
- Pre-trained with **pixel-wise masking** to extract low-level correlations
- A **zero-shot inference** paradigm based on pre-trained model…


1. Pre-training (Knowledge extraction)
Natural Images
Masked
D_θ
Reconstructed
Ensemble process
Forward process
Loss calculation
Element-wise add
Learning target: $L_{rec}(\tilde{I}, I) = \left\| \hat{M} \odot \tilde{I} - \hat{M} \odot I \right\|_2$
2. Iterative filling (Zero-shot inference)
Initialize when t=1
Timestep t
Input x
Masked
$M_t \odot x$
D_{θ_t}
Reconstructed
$\widehat{M}_t \odot y_t$
$1-\beta$
β
$\bar{y}_1$
$\bar{y}_2$
$\bar{y}_3$
$\bar{y}_{t-1}$
$\bar{y}_t$
Ensemble for total T steps
Prediction $\bar{y}$
L_{rec}
What does D_{θ_t} do in each timestep: $\tilde{I} = \mathcal{D}_\theta(M \odot I)$

Iterative Filling

- Zero-shot paradigm designed to leverage pre-trained knowledge
- Total T timesteps, for each $t \in T$, subset of image is predicted via mask-recon scheme
- The final denoised result comes from ensemble:



Iterative Filling

◆Zero-shot paradigm designed to leverage pre-trained knowledge:

Algorithm 1: Iterative filling. Pipeline designed to leverage pre-trained representation θ for zero-shot denoising.

Input: Noisy image x, pre-trained parameter θ, network $\mathcal{D}(\cdot)$, exponential weight β, masking ratio p.

Output: denoised ensemble $\overline{y}$ from predictions of iteration $\{y_t\}$.

load pre-trained parameter θ for $\mathcal{D}(\cdot)$ as θ_1

initialize $\overline{y}$

for *t from* 1 *to* T **do**

  generate random mask M_t with mask ratio p

  $y_t = \mathcal{D}_{\theta_t}(M_t \odot x)$

  $\hat{M}_t = \neg M_t$

  $\theta_{t+1} = \theta_t - \nabla_\theta \left\| \hat{M}_t \odot y_t - \hat{M}_t \odot x \right\|_2$

  $\overline{y} \leftarrow \hat{M}_t \odot (\beta \cdot \overline{y} + (1 - \beta) \cdot y_t) + M_t \odot \overline{y}$

return $\overline{y}$

Results - Gaussian & Poisson Noise

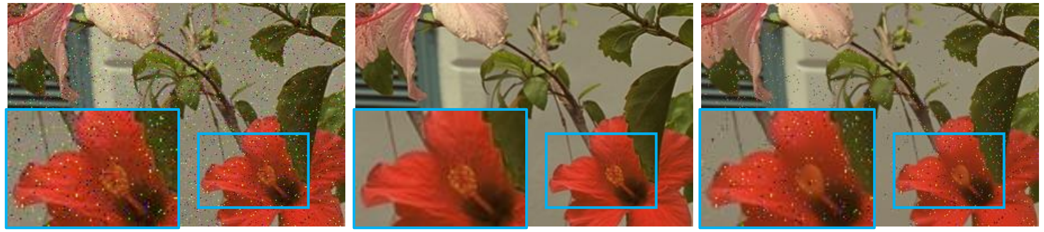
	σ	DIP	N2V*	N2S*	ZS-N2N	FasterDIP	Ours (faster)	Ours
CSet	10	32.05/0.829	31.55/0.885	28.04/0.819	33.87/0.883	31.59/0.815	33.82/0.889	**34.91/0.909**
	25	30.42/0.795	29.39/0.814	28.19/0.777	29.55/0.765	30.19/0.766	30.83/0.824	**31.61/0.841**
	50	24.73/0.533	27.35/0.694	26.62/0.699	26.10/0.624	26.09/0.669	28.14/**0.715**	**28.26**/0.710
McMaster	10	32.48/0.878	30.98/0.877	28.61/0.839	34.19/0.908	31.48/0.842	34.35/0.921	**35.46/0.937**
	25	31.07/0.856	29.11/0.833	27.59/0.776	29.37/0.786	29.47/0.794	30.99/0.862	**31.90/0.879**
	50	25.72/0.639	24.65/0.676	24.89/0.673	25.82/0.634	24.75/0.663	28.15/**0.779**	**28.37**/0.770
CBSD	10	31.18/0.865	31.18/0.918	28.17/0.853	33.73/0.923	30.89/0.857	34.20/0.935	**35.16/0.947**
	25	29.29/0.828	27.51/0.812	26.93/0.796	29.01/0.815	28.57/0.806	30.00/0.854	**30.58/0.865**
	50	23.06/0.540	25.74/0.700	24.78/0.695	25.37/0.657	24.75/0.669	**27.05/0.712**	26.85/0.703
Avg. Infer. time (s)		451.9	153.9	147.9	16.8	149.2	**10.1**	51.6

Gaussian σ=25	Reference	DIP	N2V	N2S	ZS-N2N	FasterDIP	Ours (faster)	Ours
20.23/0.334	PSNR/SSIM	29.75/0.776	29.52/0.786	27.31/0.731	29.16/0.772	28.86/0.765	29.63/0.804	30.20/0.820

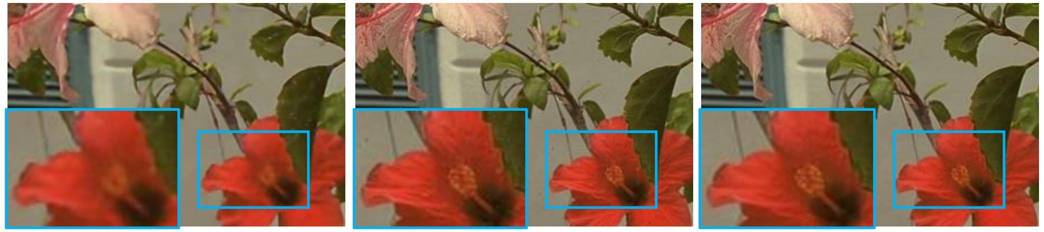
Poisson λ=25	Reference	DIP	N2V	N2S	ZS-N2N	FasterDIP	Ours (faster)	Ours
19.27/0.384	PSNR/SSIM	30.47/0.828	29.45/0.795	27.85/0.772	29.34/0.775	29.30/0.805	30.59/0.845	31.55/0.859

Results - Generalization

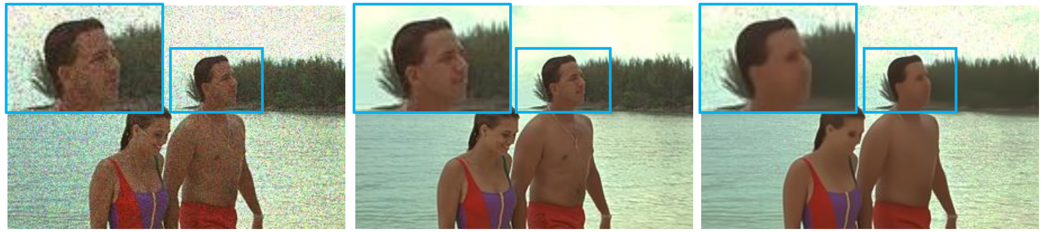
Test Noise	Supervised		Unsupervised		Zero-shot			
	SwinIR	Restormer	Nb2Nb	B2U	DIP	ZS-N2N	Ours (faster)	Ours
Gaussian $\sigma = 25$	32.89/0.895	**33.04/0.897**	32.06/0.880	32.26/0.880	30.05/0.806	29.46/0.775	30.94/0.848	31.78/0.865
Gaussian $\sigma \in [10,50]$	27.29/0.628	30.00/0.729	28.68/0.713	29.24/0.726	29.56/0.783	29.36/0.753	30.89/0.837	**31.66/0.846**
Poisson $\lambda \in [10,50]$	25.06/0.622	26.52/0.683	27.31/0.703	28.22/0.718	28.67/0.758	28.17/0.732	29.94/0.826	**30.57/0.832**
NLF	32.52/0.862	31.71/0.857	31.88/0.859	31.98/0.859	29.71/0.821	31.02/0.834	32.26/0.886	**33.15/0.901**
Speckle $v \in [10,50]$	31.97/0.841	33.52/0.884	31.31/0.837	31.65/0.847	30.73/0.818	33.78/0.891	34.79/0.924	**35.79/0.933**
S&P $d \in [0.02,0.05]$	23.96/0.614	23.63/0.613	27.04/0.686	29.44/0.796	29.54/0.800	35.25/0.952	35.05/0.953	**36.87/0.964**
Average	28.94/0.744	29.73/0.777	29.71/0.800	30.47/0.804	29.71/0.798	31.17/0.823	32.31/0.879	**33.30/0.890**



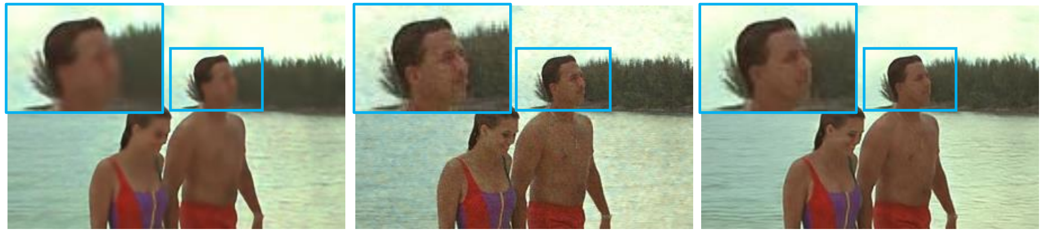
S&P d=0.025 | Reference | Restormer



DIP 31.77/0.889 | ZS-N2N 37.55/0.964 | Ours 37.78/0.968



Speckle v=41 | Reference | Restormer



DIP 30.82/0.792 | ZS-N2N 29.87/0.732 | Ours 32.61/0.853

Results – Real-world Noise

Methods	SIDD validation	SIDD benchmark	PolyU	FMD	Avg. Infer. Time (s)
DIP	33.68/0.802	33.67/0.863	37.91/0.952	32.85/0.840	333.2
N2V*	26.74/0.627	25.34/0.595	35.04/0.921	29.79/0.817	98.1
N2S*	26.78/0.573	26.93/0.658	32.82/0.930	31.61/0.759	114.4
ZS-N2N	25.59/0.422	25.61/0.559	36.04/0.915	31.65/0.768	15.1
FasterDIP	33.55/0.795	33.55/0.859	37.99/0.957	32.07/0.821	138.2
Ours (faster)	33.68/0.828	33.60/0.896	37.62/0.957	32.68/0.846	**7.9**
Ours	**34.43/0.844**	**34.32/0.903**	**38.11/0.962**	**32.97/0.847**	37.2

Noisy 26.55/0.457 | Reference PSNR/SSIM | DIP 37.35/0.948 | N2V 29.24/0.895 | N2S 29.92/0.835 | ZS-N2N 28.21/0.553 | FasterDIP 37.28/0.950 | Ours (faster) 36.86/0.937 | Ours 37.55/0.950

Noisy 36.61/0.938 | Reference PSNR/SSIM | DIP 36.23/0.942 | N2V 35.63/0.940 | N2S 30.28/0.926 | ZS-N2N 36.65/0.941 | FasterDIP 36.89/0.950 | Ours (faster) 37.29/0.961 | Ours 38.12/0.965

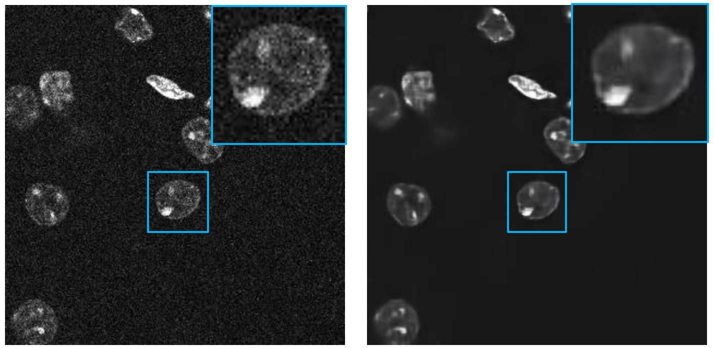
Results – What else can we do?

◆ Can even generalize to other types of images!

Extremely low-light images

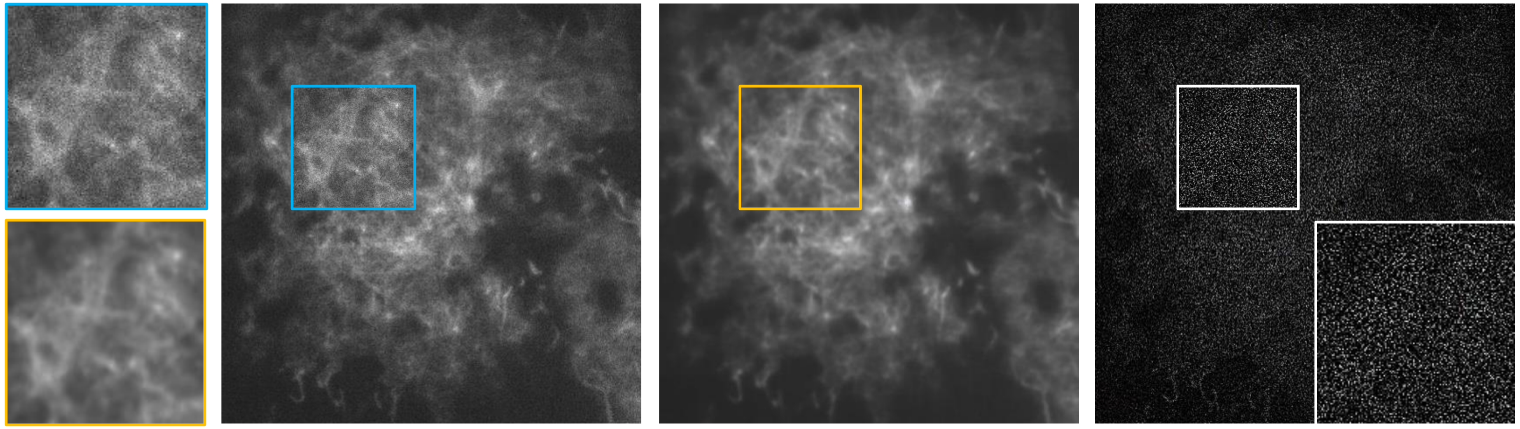
Noisy Image — Denoised

Fluorescence images



Noisy — Denoised

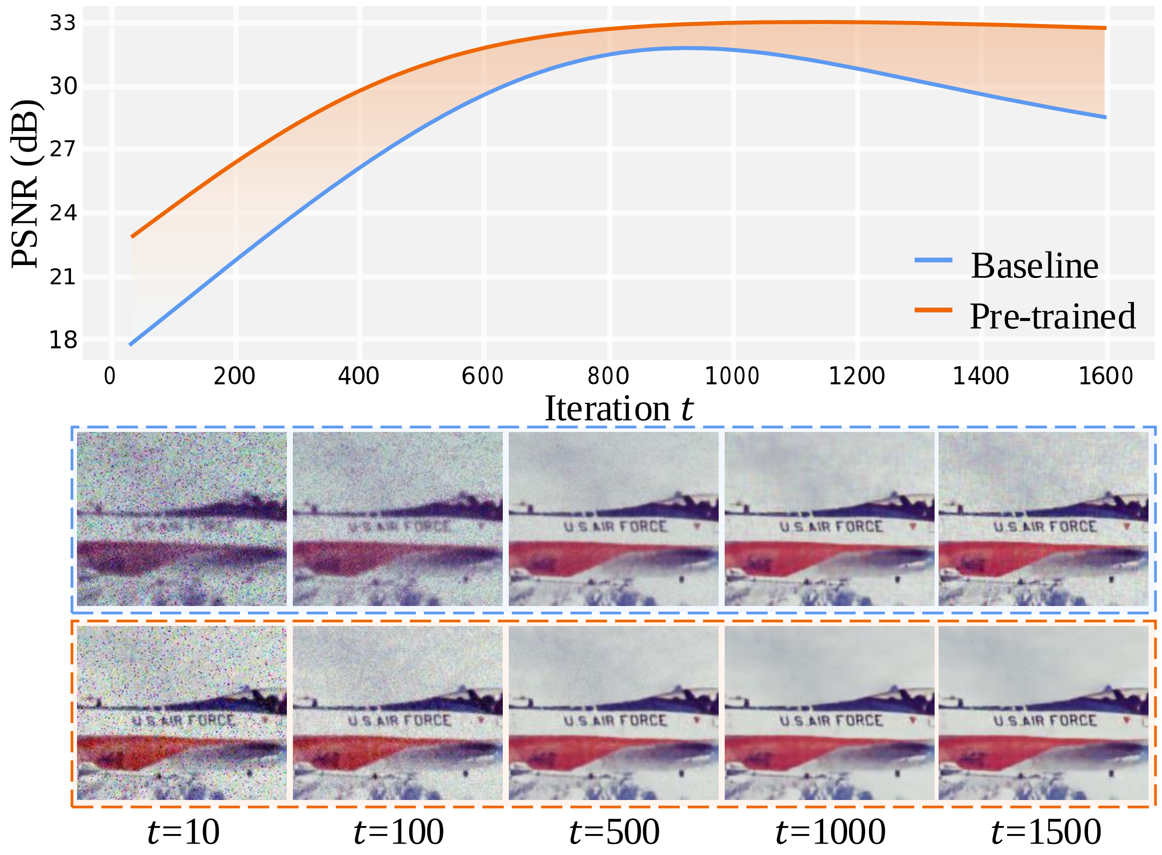
Microscopy images



Noisy Image — Denoised by Ours — Estimated Noise Map

Ablation – How much does pre-train contribute

- Pre-training not only provides faster inference speed, but also avoids the model from over-fitting

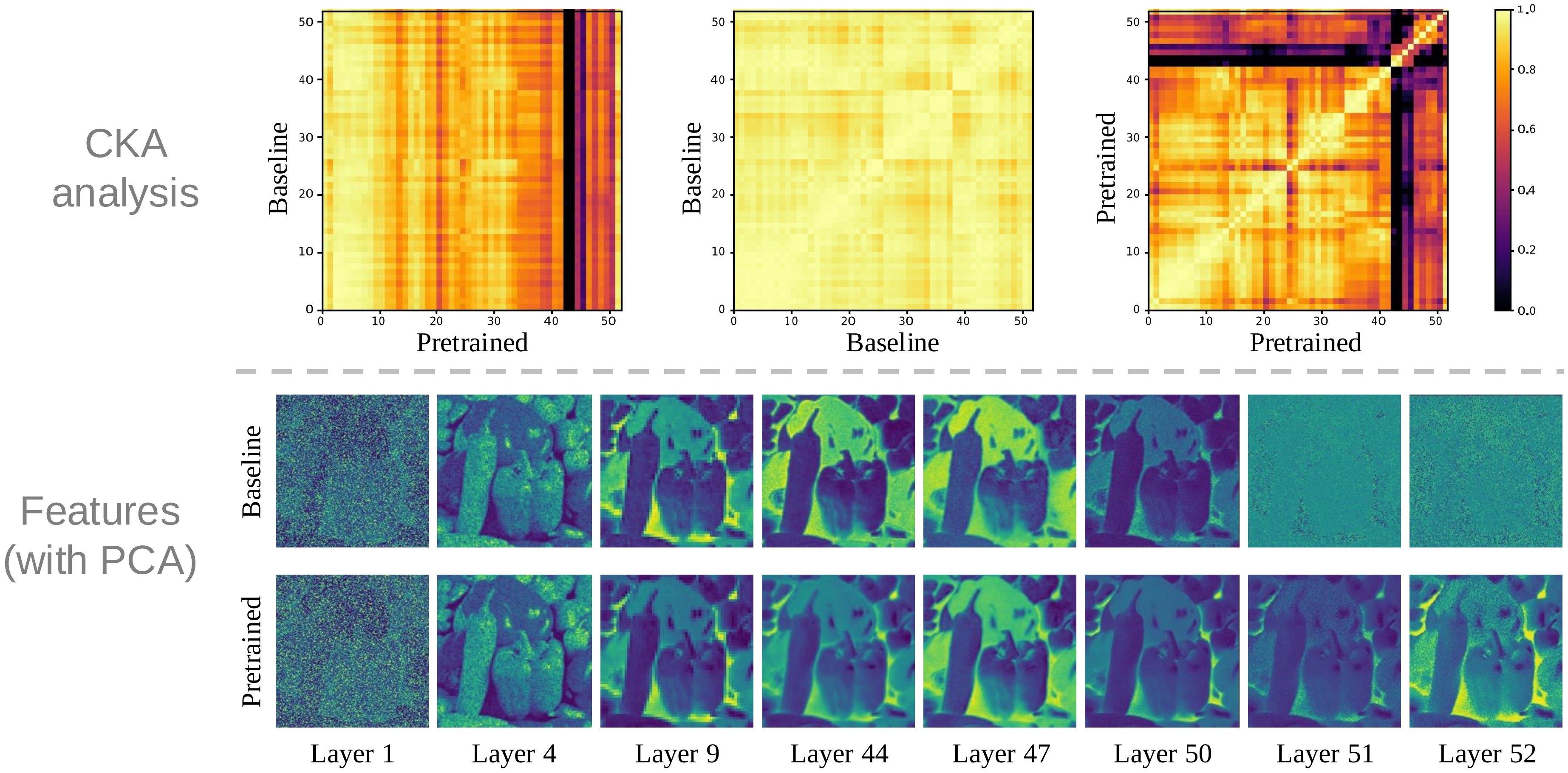


β	Pretrain	CSet	SIDD	FMD
0.99	√	31.61/0.841	34.43/0.844	32.97/0.847
	×	30.90/0.811	32.31/0.746	31.44/0.786
0.90	√	30.83/0.824	33.68/0.828	32.68/0.846
	×	30.10/0.806	33.42/0.824	32.31/0.833

Without pre-trained weights, model learns slowly and begins to overfit too early!

Analysis – Why did it work?

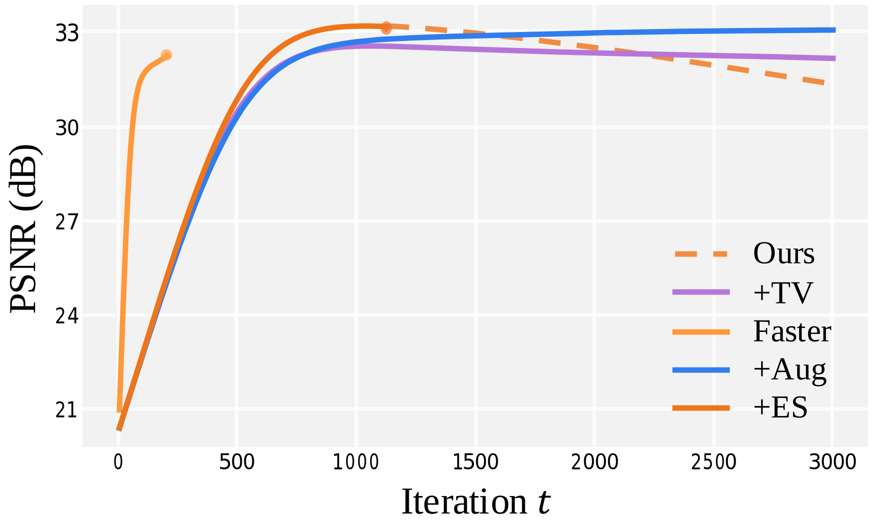
- Pre-trained features restore the full image with clearer layer distinctions
- Without pre-trained model, different parts of network tends to learn the same feature, i.e. learn to reconstruct only masked region



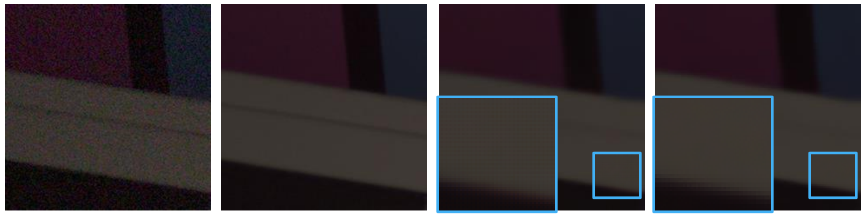
💡Moreover – How far can we go?

- With advanced strategy:

Over-fitting



Down-sampling strategy



Noisy 27.69/0.410 | Reference PSNR/SSIM | +PD 40.69/0.948 | +RSG 41.25/0.958

β	Pre-trained	Infer.time (s)
+PD	34.42/0.843	29.6
+RSG	34.75/0.852	38.3

- More network types:

Methods	Params (M)	β	Pre-trained	Baseline	Infer.time (s)
DnCNN	0.56	0.9	30.49/0.812	26.76/0.720	15.1
		0.99	31.69/0.845	30.68/0.825	75.0
ResNet	0.26	0.9	30.46/0.812	29.20/0.778	8.0
		0.99	31.43/0.838	31.16/0.836	39.4

Thanks!