Oracle Supervision Transfers for Hyperparameter Prediction in Model-Based Image Denoising

Jianmin Liao*
Department of Mathematics
Syracuse University
jliao21@syr.edu

Lixin Shen
Department of Mathematics
Syracuse University
lshen03@syr.edu

Yuesheng Xu
Department of Mathematics and Statistics
Old Dominion University
y1xu@odu.edu

Abstract

Hyperparameter prediction is a critical practical bottleneck for model-based image denoisers, ranging from classical TV/TGV variational solvers to modern diffusion-based models such as DiffPIR. While existing learned predictors can achieve near-oracle performance, this approach scales poorly: each new configuration conventionally requires its own oracle-labeled training set, and each label requires a hierarchical grid search evaluated against clean ground truth. We therefore ask whether oracle supervision collected on source configurations can transfer to target configurations with few or no target oracle labels. We propose HyperDn, a single configuration-conditioned predictor that pools oracle supervision across source configurations and predicts heterogeneous hyperparameters for new denoiser–noise configurations. In a cross-paradigm experiment, HyperDn transfers from relatively cheap TV/TGV variational sources to more expensive diffusion-based DiffPIR. With 2 target labels for predictor fitting after shared configuration calibration, it comes within 0.90 dB of the oracle and matches the 64-label Afkham baseline trained from scratch while using $1/32$ as many target labels. In addition, transfer remains effective when oracle labels are defined by SSIM or LPIPS. We also observe clear transfer gains from variational models to NLM and CBM3D, despite substantial differences in hyperparameter semantics. Without any target oracle labels, HyperDn also reaches near-oracle PSNR on two unseen mixtures of seen noise types, on transfer from mostly 96×96 source images to targets as large as roughly 1024×768, and under simultaneous shifts in image domain and diffusion prior, from WED natural images with an ImageNet-trained DiffPIR prior to FFHQ faces with an FFHQ-trained prior. Together, these results show that transferable oracle supervision can replace repeated per-configuration labeling with a shared upfront cost amortized across configurations, plus a small marginal labeling cost for the evaluated targets.

1 Introduction

Model-based denoisers include variational methods such as TV [Rudin et al., 1992] and TGV [Bredies et al., 2010], nonlocal patch-based methods such as NLM [Buades et al., 2005] and CBM3D [Dabov et al., 2007], and diffusion-based methods such as DiffPIR [Zhu et al., 2023]. Their reconstruction quality depends strongly on image-dependent hyperparameters. Learned predictors can approach oracle selection for one fixed inverse-problem configuration [Maboudi Afkham et al., 2021], but their supervision is costly: every label evaluates a hierarchical hyperparameter sweep against clean ground truth, and the process is repeated for each denoising configuration. For example, one DiffPIR call on a 256×256 image takes about 10 seconds in our setup.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

The number of configurations, N_{cfg}, is also large: each specifies a denoiser, a noise setting, and, where applicable, a data term. There are numerous existing denoisers, including variational, nonlocal, and diffusion-based ones, while noise types include Gaussian, Poisson, black-and-white impulse, RGB impulse, and their mixtures. The design space keeps expanding as new methods and noise mixtures appear. The total cost therefore scales as

$$\underbrace{t_{\text{call}}}_{\text{sec/call}} \times \underbrace{K_{\text{sweep}}}_{\text{candidates/image}} \times \underbrace{N_{\text{img}}}_{\text{training images}} \times \underbrace{N_{\text{cfg}}}_{\text{configurations}} .$$

We ask whether oracle labels collected on source configurations can reduce the label need of a new target configuration. Transfer is plausible for two reasons. First, image properties such as noise strength, texture, and edges affect hyperparameter selection across denoisers. Second, hyperparameters can have similar semantics while taking different numerical values. For example, TV and TGV explicitly fit $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{u}} D(\boldsymbol{u}, \boldsymbol{y}) + \lambda R(\boldsymbol{u})$, while DiffPIR analogously uses λ to balance data consistency against its diffusion prior. Supervision from relatively cheap TV/TGV configurations can therefore inform hyperparameter prediction for the more expensive DiffPIR. We introduce HyperDn, a single predictor trained on labels pooled across source configurations. A shared encoder extracts image properties, while a configuration-conditioned slot readout handles different parameter sets and denoiser-specific ranges.

We evaluate transfer across denoisers, evaluation metrics, noise types, image domains, and image resolutions. Transferring from TV/TGV to DiffPIR, HyperDn reaches 30.23 dB with two target labels, matching the 64-label Afkham per-configuration CNN predictor with $1/32$ as many labels. Transfer also holds under SSIM and LPIPS: one target label gives better mean scores than Afkham with 16 labels under both objectives. For NLM and CBM3D, HyperDn achieves the highest mean PSNR at every tested label budget despite their different hyperparameter semantics; with two target labels, HyperDn outperforms Afkham by 3.73 and 3.95 dB, respectively. Without any target oracle labels, HyperDn transfers zero-shot from WED natural images to FFHQ faces while DiffPIR simultaneously switches from an ImageNet-trained to an FFHQ-trained prior, achieving near-oracle PSNR with a gap of 0.273 dB. With no target labels, HyperDn also stays within ~ 0.2 dB of the oracle on two unseen mixtures of seen noise types and transfers from mostly 96×96 source images to targets up to roughly 1024×768. On 512×768 Kodak24, oracle labeling costs roughly $40\times$ more denoiser time than at 96×96, making it especially valuable to reuse oracle labels from low-resolution images.

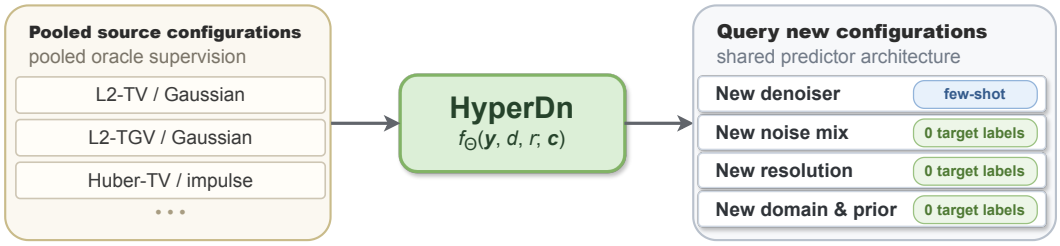


Figure 1: **HyperDn: one pool of oracle labels for many configurations.** Oracle labels from multiple source configurations are pooled to train a shared conditional predictor $f_\Theta(\boldsymbol{y}, d, r; \boldsymbol{c})$. HyperDn reuses this supervision for few-shot transfer to new denoisers (DiffPIR, NLM, and CBM3D) and zero-shot transfer to unseen mixtures of seen noise types, larger resolutions, and simultaneous shifts in image domain and diffusion prior. Here, $\boldsymbol{y}$ is the degraded image, d is the data-term tag, r is the prior identifier, and $\boldsymbol{c}$ encodes noise attributes and observation metadata only when provided.

Contributions.

- We formulate oracle hyperparameter labels for image denoising as reusable supervision pooled across configurations.
- We introduce HyperDn, a single configuration-conditioned predictor for denoisers with different hyperparameters.
- We show few-shot transfer from variational models to DiffPIR, NLM, and CBM3D, including denoisers with substantially different hyperparameter semantics. Two DiffPIR labels

for predictor fitting match the 64-label Afkham baseline, and we observe similar transfer gains with SSIM- or LPIPS-defined labels. Without any target oracle labels, HyperDn also transfers zero-shot to unseen mixtures of seen noise types, to larger resolutions, and from WED natural images to FFHQ faces while DiffPIR simultaneously switches from an ImageNet-trained to an FFHQ-trained diffusion prior.

2 HyperDn

Given a degraded image and a denoising configuration, HyperDn (Figure 2) predicts the hyperparameters needed to reconstruct that image. We represent configuration-specific hyperparameters in a padded vector. A single configuration-conditioned predictor then shares information across configurations while preserving each denoiser's numerical range. It is trained on oracle labels pooled across the source configurations.

2.1 Problem setting: oracle-supervised hyperparameter prediction

Configuration and hyperparameter vector. We describe each configuration by a prior R, represented by a regularizer or denoiser, and, where applicable, a data term D. Here, D ranges over L2 for Gaussian noise, Huber for Gaussian or impulse noise and their mixtures, and negative-log-likelihood-style (NLL) data terms for Poisson or Poisson–Gaussian observations; R ranges over TV, TGV, DiffPIR, NLM, and CBM3D; observation noise spans Gaussian, Poisson, impulse, and mixtures thereof. A denoising configuration is defined as a tuple $(d, r; \boldsymbol{c})$, where d is the data-term identifier, r is the prior identifier, and $\boldsymbol{c}$ encodes noise attributes and observation metadata, such as camera calibration parameters, if available; $\boldsymbol{c}$ is empty otherwise.

For each configuration, we pad its hyperparameter vector to $\boldsymbol{p} \in \mathbb{R}^{L_{\max}}$, where $L_{\max}$ is the largest supported vector length. The pair (d, r) determines the ordered active entries and the remaining padding. The Huber data term contributes the threshold δ; L2 and NLL contribute none. TV and DiffPIR use the prior weight λ; TGV uses λ together with the second-order to first-order weight ratio γ. Thus, for example, Huber-TV uses (λ, δ) and Huber-TGV uses $(\lambda, \gamma, \delta)$, followed by padding when needed.

Training and test data. The dataset consists of tuples $(\boldsymbol{y}, \boldsymbol{x}, d, r, \boldsymbol{c})$, where $\boldsymbol{x}$ is the clean image; each tuple is paired with an oracle hyperparameter target $\boldsymbol{p}^\star$. Let $\mathcal{R}_{d,r}(\boldsymbol{y}; \boldsymbol{p})$ denote the reconstruction for $\boldsymbol{y}$ under configuration (d, r) with hyperparameters $\boldsymbol{p}$ and required observation metadata implicit, $\mathcal{P}_{d,r}(\boldsymbol{y}, \boldsymbol{x})$ the finite, nonempty set of valid hyperparameter vectors evaluated by the oracle search for this image pair, and Q a quality criterion to be maximized. Q can be any scalar quality measure computed for a reconstruction–reference image pair, including PSNR, SSIM [Wang et al., 2004], and negative LPIPS [Zhang et al., 2018]; each choice defines its own oracle labels. The oracle target maximizes Q against the clean image:

$$\boldsymbol{p}^\star \in \operatorname*{arg\,max}_{\boldsymbol{p} \in \mathcal{P}_{d,r}(\boldsymbol{y}, \boldsymbol{x})} Q(\mathcal{R}_{d,r}(\boldsymbol{y}; \boldsymbol{p}),\, \boldsymbol{x})\,. \tag{1}$$

Predictor input/output. HyperDn is a learned predictor f_Θ with parameters Θ. Given configuration $(d, r; \boldsymbol{c})$ and degraded image $\boldsymbol{y}$, f_Θ produces a padded hyperparameter vector

$$\hat{\boldsymbol{p}} = f_\Theta(\boldsymbol{y},\, d,\, r;\, \boldsymbol{c}) \in \mathbb{R}^{L_{\max}}, \tag{2}$$

in solver-interface units, matching $\boldsymbol{p}^\star$ after any cache conversion. The head predicts encoded slots $\hat{\boldsymbol{q}}$ and trains against $\tilde{\boldsymbol{q}} = \mathrm{enc}_{d,r}(\boldsymbol{p}^\star)$. The configuration-specific decoder gives $\hat{\boldsymbol{p}} = \kappa_{d,r}(\hat{\boldsymbol{q}})$, applying 10^{q_j} to log-scale slots and leaving linear slots unchanged. The reconstruction is $\mathcal{R}_{d,r}(\boldsymbol{y}; \hat{\boldsymbol{p}})$.

2.2 Architecture: configuration-conditioned hyperparameter prediction

HyperDn has two parts: a shared encoder and an output head. The encoder uses one set of weights for all configurations, producing a feature $\boldsymbol{h}$ from the image and configuration $(d, r; \boldsymbol{c})$. The output head is designed to share hyperparameter roles across configurations while still distinguishing how each denoiser uses those roles. For example, λ plays a prior-weight role—the weight balancing the prior against data fidelity—in TV, TGV, and DiffPIR, allowing supervision of this role to be shared across

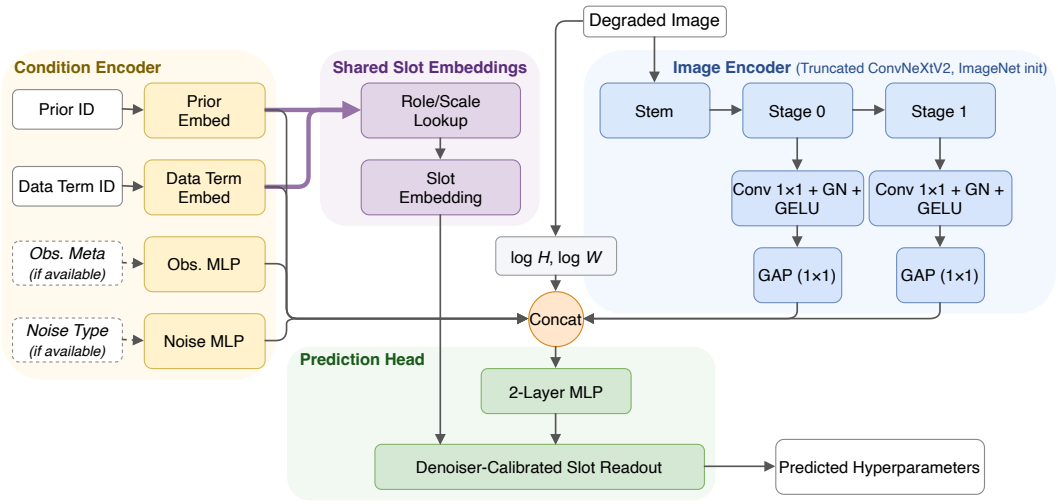


Figure 2: HyperDn architecture. A shared image encoder built on a truncated ConvNeXtV2 (stem and stages 0–1; FCMAE-pretrained, then fine-tuned on ImageNet-22K and ImageNet-1K) extracts features from the degraded image. A configuration-conditioned output head, comprising a condition encoder, shared slot embeddings, and a denoiser-calibrated slot readout, produces the padded hyperparameter vector for the queried configuration; padding slots are ignored in the loss.

denoisers. Each denoiser uses its prior weight λ in its own numerical range. Configurations also activate different sets of hyperparameters (e.g., L2-TV activates only λ, while Huber-TGV activates δ, λ, and γ), and the head predicts each configuration's active parameters. HyperDn therefore represents the output as padded *slots*: each coordinate of the encoded vector $\boldsymbol{q}$ is called a slot, and under (d, r) a slot is either active (it holds a hyperparameter) or padding. The output head is a denoiser-calibrated slot readout that maps $\boldsymbol{h}$ and per-slot metadata to $\hat{\boldsymbol{q}}$. Figure 2 shows the architecture of HyperDn.

Image and configuration encoder. The image encoder uses the ConvNeXtV2-base `fcmae_ft_in22k_in1k` checkpoint [Deng et al., 2009, Woo et al., 2023], truncated to its stem and stages 0–1. Our ablation shows lower PSNR with either fewer or more stages (Appendix C.2). For each retained stage, a 1×1 convolution projects the spatial features to 128 channels, adaptive average pooling forms a 12×12 grid, and averaging that grid produces a per-stage vector. These per-stage vectors are concatenated with $\log H$ and $\log W$ to form the image embedding, where H and W are the height and width of the input degraded image. The condition encoder produces four embeddings: data-term and prior embeddings of d and r, respectively, and two optional MLP encodings of $\boldsymbol{c}$—one for noise attributes if available (a multi-hot categorical code flagging which noise types are present, e.g., Gaussian alone or black-and-white impulse + Gaussian) and one for observation metadata if available. We provide *observation metadata* only for Poisson–Gaussian source configurations, using per-image calibrated acquisition parameters such as photon count and read-noise standard deviation. All other configurations receive no observation metadata. If observation metadata or the noise type is unavailable, the corresponding MLP receives a fixed default vector. The four condition embeddings are concatenated with the image embedding, which includes $\log H$ and $\log W$. A 2-layer MLP then maps this concatenation to a feature $\boldsymbol{h}$ used by the denoiser-calibrated slot readout described below.

Denoiser-calibrated slot readout. Our readout is designed to share supervision across configurations with different hyperparameter sets, roles, and numerical ranges. All slots use a shared scoring function that combines $\boldsymbol{h}$ with their role and scale embeddings. Slots with the same role, such as λ in TV and TGV, also share a role embedding even when their vector positions differ. To accommodate different numerical ranges for the same role across denoisers, we additionally transform the shared scores using denoiser-specific scales and offsets. For configuration (d, r), the role/scale lookup (Figure 2) assigns each slot j a role tag $\rho_j \in \{\delta, \lambda, \gamma, \texttt{UNKNOWN}, \text{pad}\}$ and a scale tag τ_j ($\log_{10}$ for δ, λ, and the two unseen principal scalars; linear for γ; none for padding). The `UNKNOWN` role is unused in source pretraining and activated for the unseen NLM/CBM3D targets. Learned embedding

matrices $\mathbf{E}_{\text{role}}$ and $\mathbf{E}_{\text{scale}}$ give the slot embedding

$$\boldsymbol{e}_j \;=\; \mathbf{E}_{\text{role}}[\rho_j] + \mathbf{E}_{\text{scale}}[\tau_j].$$

The readout first computes an intermediate scalar score s_j for predicting the hyperparameter in slot j. The matrix $\mathbf{U}$ and vectors $\boldsymbol{w}, \boldsymbol{v}$ are learned coefficients shared across all slots. The projection $\mathbf{U}\boldsymbol{h} + \boldsymbol{w}$ combines a linear map and a bias, and its inner product with $\boldsymbol{e}_j$ makes the score depend on the slot's role and scale. The term $\boldsymbol{v}^\top \boldsymbol{h}$ adds a shared linear contribution from $\boldsymbol{h}$. We then scale and shift s_j using a learned denoiser-specific scale $g_{d,r,j}$ and offset $b_{d,r,j}$, and add a linear shortcut with slot-specific weight vector $\boldsymbol{a}_j$:

$$s_j = (\mathbf{U}\boldsymbol{h} + \boldsymbol{w})^\top \boldsymbol{e}_j + \boldsymbol{v}^\top \boldsymbol{h}, \qquad \hat{q}_j = \underbrace{g_{d,r,j} s_j + b_{d,r,j}}_{\text{denoiser-calibrated shared score}} + \underbrace{\boldsymbol{a}_j^\top \boldsymbol{h}}_{\text{linear shortcut}}. \tag{3}$$

For log-scale slots, the decoder returns the raw parameter $\hat{p}_j = 10^{\hat{q}_j}$; for linear slots, $\hat{p}_j = \hat{q}_j$. Our ablation study in Appendix C.1 demonstrates the effectiveness of the proposed denoiser-calibrated slot readout.

2.3 Pooled supervision

HyperDn is trained on oracle-labeled samples pooled across all source configurations. For each sample, the network predicts the full padded encoded vector $\hat{\boldsymbol{q}} \in \mathbb{R}^{L_{\max}}$, but the loss penalizes only the slots active for that sample's (d, r)—for example, λ for TV; δ and λ for Huber-TV; λ and γ for TGV. The regression loss and training details are provided in Appendix A.

2.4 Theoretical Analysis of Transfer Error

We analyze the error between predicted encoded hyperparameters and oracle labels on the target task. We derive a three-term bound: source error and prediction-error discrepancies across data and tasks. Let f be the trained predictor mapping observations and supplied task information to encoded hyperparameters, with its weights and inference settings fixed. For source tasks $k = 1, \ldots, K$ ($1 \le K < \infty$) and target task T, let $f_k(x), f_T(x)$ be its outputs and $q_k^\star(x), q_T^\star(x)$ the oracle labels. Let P_k, P_T be the source and target data distributions, with image samples $X_S^{(k)} \sim P_k$ and $X_T \sim P_T$, respectively. The pooled source and target root mean squared prediction errors (RMSEs) are

$$\begin{aligned} \varepsilon_S(f) &= \left(\sum_k \alpha_k \mathbb{E}_{P_k} \| f_k(X_S^{(k)}) - q_k^\star(X_S^{(k)}) \|^2 \right)^{1/2}, \\ \varepsilon_T(f) &= \left(\mathbb{E}_{P_T} \| f_T(X_T) - q_T^\star(X_T) \|^2 \right)^{1/2}, \end{aligned} \tag{4}$$

with fixed source weights $\alpha_k > 0$ satisfying $\sum_k \alpha_k = 1$. Let $\Pi(P_k, P_T)$ be the set of joint distributions of the source and target samples with marginals P_k and P_T, respectively.

In our notation, $\Delta_{\text{data}}(f)$ measures changes in prediction error as the predictor generalizes to new image distributions under the source tasks:

$$\Delta_{\text{data}}(f) = \sqrt{\sum_k \alpha_k \inf_{\pi \in \Pi(P_k, P_T)} \mathbb{E}_\pi \left\| [f_k(X_T) - f_k(X_S^{(k)})] - [q_k^\star(X_T) - q_k^\star(X_S^{(k)})] \right\|^2}. \tag{5a}$$

$\Delta_{\text{task}}(f)$ measures changes in prediction error as the same predictor generalizes to the target task on the same target image:

$$\Delta_{\text{task}}(f) = \left(\sum_k \alpha_k \mathbb{E}_{P_T} \left\| [f_T(X_T) - f_k(X_T)] - [q_T^\star(X_T) - q_k^\star(X_T)] \right\|^2 \right)^{1/2}. \tag{5b}$$

Theorem 1 (Transfer error of a fixed predictor)**.** *Under the definitions above and the conditions in Appendix E.1,*

$$\varepsilon_T(f) \le \varepsilon_S(f) + \Delta_{\text{data}}(f) + \Delta_{\text{task}}(f). \tag{6}$$

The proof is given in Appendix E.

3 Experiments

3.1 Experimental setups

Goal. We evaluate whether HyperDn can handle configurations not seen during source training. We study few-shot transfer to DiffPIR, NLM, and CBM3D; transfer under SSIM and LPIPS objectives; and zero-shot transfer across noise mixtures and resolutions, and under simultaneous shifts in image domain and diffusion prior. **Training Setup.** For each image–configuration pair, the oracle label is the hyperparameter vector found by hierarchical grid search to maximize the chosen quality criterion. The PSNR-based DiffPIR, NLM/CBM3D, and cross-resolution studies start from the non-DiffPIR source checkpoint; FFHQ evaluation follows additional adaptation on WED DiffPIR labels. Mixed-noise transfer uses the two corresponding single-noise sources, while each SSIM/LPIPS study uses two WED TV/TGV sources labeled under its objective. **Evaluation Setup.** We report PSNR and its gap to the oracle, and SSIM or LPIPS for the metric-specific studies. Unless otherwise stated, learned-method results are reported as mean$\pm$std over 5 random seeds. Full experimental details are provided in the appendices.

3.2 Few-shot transfer to DiffPIR

We evaluate few-shot transfer from cheaper TV/TGV source labels to DiffPIR [Zhu et al., 2023] on 256×256 crops from the Waterloo Exploration Database (WED) [Ma et al., 2017].

We compare three predictors across $\{1, 2, 4, 16, 32, 64, 128\}$ DiffPIR target labels: **HyperDn** initialized from the source checkpoint, **ImageNet init**, and an **Afkham**-style per-configuration CNN [Maboudi Afkham et al., 2021]. ImageNet init shares HyperDn's architecture, ImageNet-pretrained backbone initialization, target-label subsets, and finetuning protocol, but omits source training. All methods share the same DiffPIR sampling schedule and oracle construction. Budgets count predictor-fitting labels after shared configuration calibration (Appendix B.1).

HyperDn has the highest mean at every label budget (Figure 3; full results in Appendix B.1). With 1 label, its gap to the 31.13 dB oracle is 1.76 dB, versus 8.07 dB for Afkham and 6.15 dB for ImageNet init. With 2 labels, HyperDn reaches 30.23 dB (0.90 dB below the oracle), matching Afkham at 64 labels (30.28 dB) with $1/32$ of its label budget. Its gap narrows to 0.51 and 0.10 dB with 4 and 32 labels, respectively. Source training adds 4.39 and 2.50 dB over ImageNet init with 1 and 2 labels, respectively. In Figure 4, both predictors use 16 target labels. Afkham leaves substantial residual noise and trails the oracle by 10.85 dB. HyperDn is just 0.07 dB below the oracle and nearly visually indistinguishable from it even in the $4\times$ enlarged crop.

This benefit also extends to 256×256 crops from CBSD68 and Urban100: with one target label, HyperDn exceeds Afkham trained with 16 labels by 1.68 and 0.99 dB in mean PSNR, respectively (Appendix B, Table 11).

Transfer under structural and perceptual objectives. To demonstrate that HyperDn is agnostic to the evaluation criterion, we evaluate transfer from WED TV/TGV sources to DiffPIR under SSIM and LPIPS objectives. HyperDn has the best mean at every tested budget under both objectives (Table 1); with one target label, it achieves better mean scores than Afkham with 16 labels for both SSIM ($\mathbf{.821}$ vs. $.789$) and LPIPS ($\mathbf{.227}$ vs. $.246$).

3.3 Zero-shot transfer to mixed noise

We test whether HyperDn handles unseen noise mixtures when trained only on single-noise labels. The number of possible noise mixtures grows combinatorially with the number of noise types, so labeling every mixture is expensive.

For each unseen mixed-noise target, all three methods use the same source data—the two single-noise source training sets for the target's noise types—and evaluate directly on the mixture. The two targets are black-and-white (bw) impulse + Gaussian with Huber-TV and R/G/B (rgb) impulse + Gaussian with Huber-TGV. **HyperDn** pools both training sets into one. **Afkham-mean** averages predicted hyperparameters of two Afkham predictors trained independently on the two source sets. **Mean-opt** computes one mean oracle hyperparameter vector per single-noise training set, then averages the two. These baselines test whether averaging single-noise predictions or optima is sufficient.

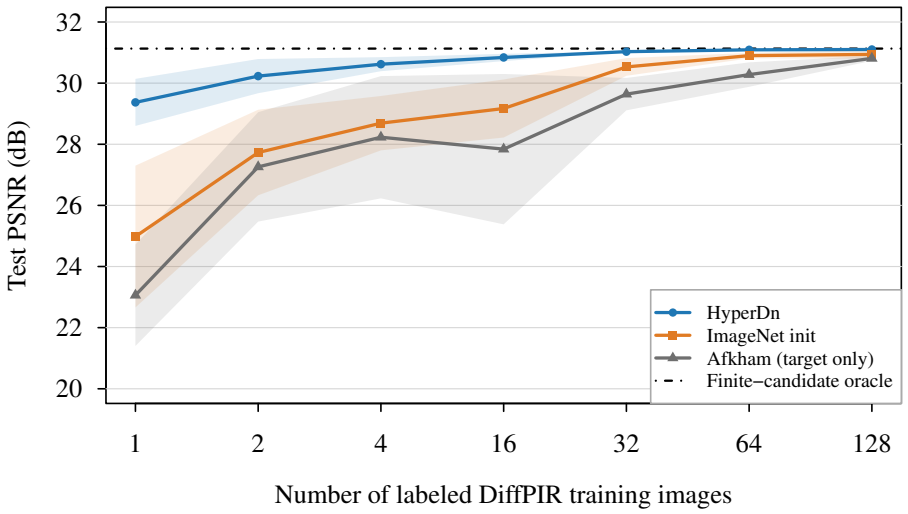


Figure 3: Few-shot DiffPIR transfer on WED 256×256 crops (100 iterations). Curves show mean test PSNR against target-label budget; shading shows ±std. Oracle: 31.13 dB (dash-dot).

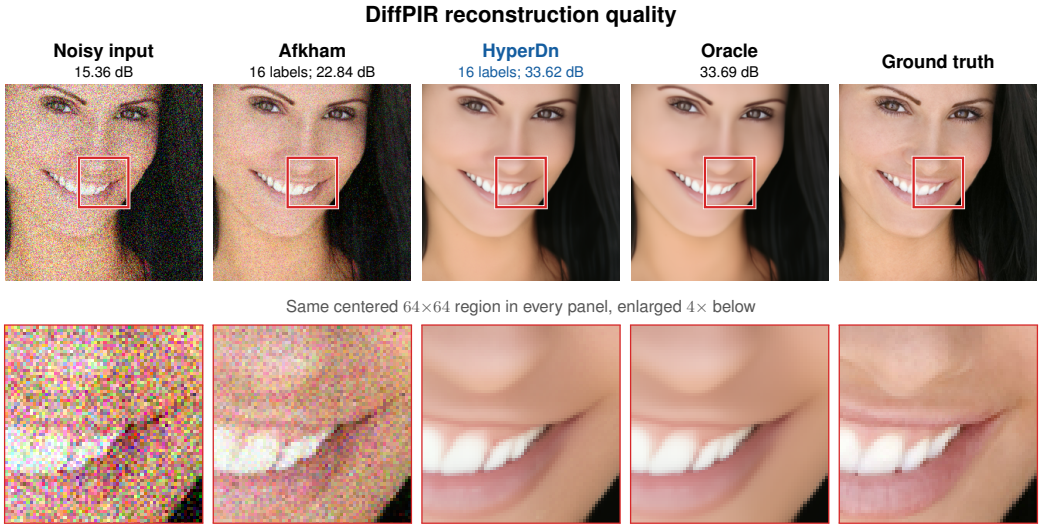


Figure 4: DiffPIR reconstructions with 16 target labels. Bottom: centered 64×64 crops at 4× magnification.

HyperDn stays within 0.21 / 0.09 dB of the oracle on the two targets (Table 2, Figure 5), while Afkham-mean and Mean-opt trail by 5–7 dB. This shows that HyperDn's zero-shot mixed-noise transfer is not a trivial consequence of averaging single-noise predictors or optima.

3.4 Zero-shot cross-resolution transfer

We test whether HyperDn, source-pretrained jointly on the small-image source training sets of §3.1 (STL10 [Coates et al., 2011] 96×96 images and WED 256×256 crops, spanning multiple denoiser/noise configurations), stays near-oracle on larger test images without any target labels. A 512×768 Kodak24 oracle label costs roughly 40× more denoiser time than a 96×96 STL10 label, so transferring from cheap small-image labels to expensive large-image targets is practically valuable. Figure 6 illustrates this cross-resolution transfer. We evaluate three configurations on Kodak24 [Eastman Kodak Company, 1999]; for L2-TV with Gaussian we additionally evaluate on

Table 1: Transfer from WED TV/TGV sources to DiffPIR under SSIM and LPIPS objectives, with N target training labels. Oracles: .8911 SSIM and .1490 LPIPS.

Method	N=1	2	4	16	32	64	128
				SSIM ↑			
HyperDn	**.821±.055**	**.838±.065**	**.877±.007**	**.881±.006**	**.885±.002**	**.886±.002**	**.888±.002**
ImageNet init	.552±.047	.639±.093	.674±.045	.791±.050	.866±.017	.881±.004	.883±.009
Afkham	.537±.115	.649±.095	.698±.079	.789±.018	.838±.024	.863±.013	.872±.003
				LPIPS ↓			
HyperDn	**.227±.024**	**.206±.005**	**.204±.008**	**.177±.006**	**.170±.006**	**.166±.005**	**.164±.005**
ImageNet init	.387±.021	.339±.041	.320±.021	.246±.030	.193±.013	.177±.006	.167±.007
Afkham	.407±.052	.332±.045	.301±.040	.246±.017	.219±.022	.199±.012	.182±.003

Table 2: Zero-shot transfer to mixed-noise targets with known categorical composition codes but no observation metadata. PSNR (dB) ↑; Mean-opt and Oracle are deterministic.

Noise type (Denoiser)	Afkham-mean	Mean-opt	HyperDn	Oracle
bw impulse + Gaussian (Huber-TV)	17.44 ± 0.92	18.95	$\mathbf{24.31 \pm 0.07}$	24.52
rgb impulse + Gaussian (Huber-TGV)	19.68 ± 0.86	17.91	$\mathbf{25.00 \pm 0.06}$	25.09

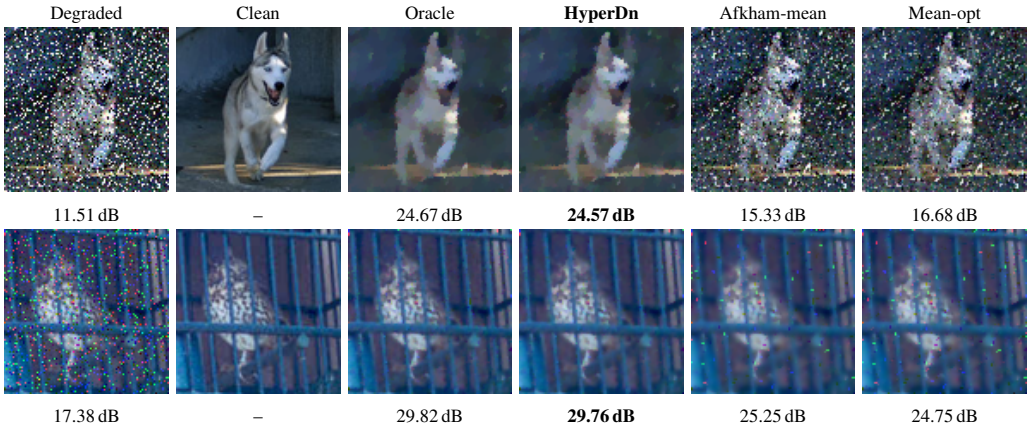


Figure 5: Reconstruction samples of zero-shot transfer to mixed noise. Top row: bw impulse + Gaussian with Huber-TV. Bottom row: rgb impulse + Gaussian with Huber-TGV. Both averaging baselines (Afkham-mean, Mean-opt) leave residual noise; HyperDn matches the oracle on both targets.

CBSD68 [Martin et al., 2001] and Urban100 [Huang et al., 2015]. HyperDn stays near-oracle in every evaluated cell (Table 3), showing zero-shot transfer from small source images to larger target images.

3.5 Unseen nonlocal denoisers and parameter roles

These experiments show that HyperDn transfers effectively even when source and target hyperparameters have substantially different semantics. We evaluate few-shot transfer from variational source configurations to NLM [Buades et al., 2005] and CBM3D [Dabov et al., 2007], neither seen during source training. The source hyperparameters control the balance or shape of explicit data-fidelity and regularization terms. In contrast, NLM's principal hyperparameter h controls the decay of nonlocal patch-similarity weights and thus each patch's contribution to averaging. CBM3D's principal hyperparameter σ_{psd} specifies the assumed noise standard deviation that governs its collaborative filtering.

Table 3: Zero-shot cross-resolution transfer. HyperDn is the jointly pretrained checkpoint of §3.1 (STL10 96×96 images and WED 256×256 crops), applied to each target dataset without finetuning or observation metadata. "Gap to oracle" is the mean over test images of (oracle PSNR − HyperDn PSNR).

Configuration	Noise	Dataset	Resolution	Gap to oracle ↓
Huber-TV	bw impulse	Kodak24	512×768	0.06 ± 0.06
L2-TGV	Gaussian	Kodak24	512×768	0.02 ± 0.01
L2-TV	Gaussian	CBSD68	321×481	0.01 ± 0.0029
		Kodak24	512×768	0.27 ± 0.0030
		Urban100	$\sim1024\times768$	0.02 ± 0.0045

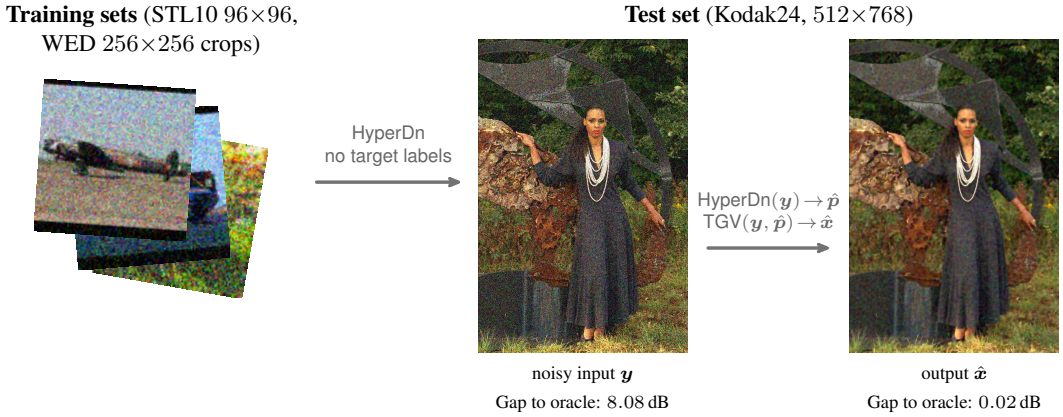


Figure 6: Zero-shot cross-resolution transfer. HyperDn is trained on small-image source training sets and applied directly to larger test images without any target labels. Given the noisy input $\boldsymbol{y}$, HyperDn predicts $\hat{\boldsymbol{p}}$, which the L2-TGV denoiser then uses to produce the reconstruction image $\hat{\boldsymbol{x}}$.

We adapt on N labels from a 128-image WED 256×256 pool and evaluate on 50 test images outside this pool. ImageNet init uses the same architecture without source supervision; Afkham is a per-configuration CNN trained from scratch. Table 4 reports all seven budgets, and Appendix B gives the oracle-search details.

Table 4: Few-shot transfer from variational sources to previously unseen nonlocal denoisers. PSNR (dB) ↑; oracle PSNR is 27.64 dB for NLM and 30.98 dB for CBM3D.

Method	$N=1$	2	4	16	32	64	128
			NLM				
HyperDn	**24.32±1.04**	**25.64±1.03**	**26.41±.48**	**27.18±.14**	**27.31±.14**	**27.47±.07**	**27.53±.02**
ImageNet init	20.94±.68	22.89±1.98	21.75±.62	24.23±1.42	26.89±.37	27.32±.09	27.42±.09
Afkham	21.04±.81	21.91±.93	23.00±1.14	25.57±.55	26.48±.22	27.07±.15	27.30±.03
			CBM3D				
HyperDn	**27.09±2.65**	**28.49±1.29**	**29.25±1.61**	**30.58±.16**	**30.81±.06**	**30.90±.02**	**30.93±.00**
ImageNet init	23.43±2.51	25.94±2.23	26.25±2.29	29.70±.53	30.65±.09	30.88±.01	30.90±.04
Afkham	22.42±2.03	24.54±2.41	26.16±1.95	28.83±.70	29.95±.29	30.49±.11	30.64±.08

Despite this substantial semantic shift, HyperDn outperforms both controls at matched budgets in the low-shot regime and achieves the highest mean PSNR at all seven budgets for both denoisers.

3.6 Zero-shot image-domain and diffusion-prior shift

After source pretraining and adaptation on 128 WED DiffPIR labels generated with the ImageNet prior, we directly evaluate on 50 FFHQ images [Karras et al., 2019] using an FFHQ-trained diffusion

prior unseen by the predictor. No FFHQ image or FFHQ-prior oracle label trains or adapts HyperDn. The fixed-λ control is computed from the same 128 WED training labels by geometric averaging, with no tuning on FFHQ. HyperDn reaches 32.553 dB, 0.273 dB below the oracle and 1.345 dB above the fixed control (Table 5).

Table 5: Zero-shot DiffPIR transfer from WED natural images to FFHQ faces using an FFHQ-trained prior in place of the ImageNet-trained prior.

Metric	Oracle	Fixed λ (WED)	HyperDn (zero-shot)
PSNR (dB) ↑	32.826	$31.208 \pm .169$	$\mathbf{32.553 \pm .010}$

We also test transfer from synthetic to real camera noise on SIDD sRGB using Huber–TV. With two target labels, HyperDn reaches 33.73 dB mean PSNR, compared with 31.80 dB for ImageNet init using the same labels (Appendix B).

4 Related work

Hyperparameter selection. Classical rules include the L-curve [Hansen, 1992] and generalized cross-validation [Golub et al., 1979]. Imaging methods predict fixed-model parameters [Maboudi Afkham et al., 2021], learn spatial TGV weights [Vu et al., 2025], tune plug-and-play methods [Wei et al., 2020], or use bilevel learning [De los Reyes et al., 2017]. HyperDn transfers oracle supervision across models. Our Gaussian L2-TV comparison gives Monte-Carlo SURE [Ramani et al., 2008] the true noise standard deviation (Appendix D.3).

Learned hyperparameter optimization. FSBO [Wistuba and Grabocka, 2021], OptFormer [Chen et al., 2022], and ranking-weighted Gaussian-process ensembles [Feurer et al., 2022] transfer past searches to propose target trials and use target-task responses during search. HyperDn predicts $\hat{\boldsymbol{p}}$ directly from the image and configuration, with no target oracle evaluations in zero-shot transfer.

Meta-learning. MAML [Finn et al., 2017] and Reptile [Nichol et al., 2018] learn initializations for rapid task adaptation. HyperDn instead trains a single configuration-conditioned predictor on pooled oracle labels across source configurations, without task-specific adaptation during pretraining.

Conditional parameter generation. Hypernetworks [Ha et al., 2017] and dynamic-filter networks [De Brabandere et al., 2016] generate network weights or input-conditioned filters; HyperRecon [Wang et al., 2022] maps hyperparameters to reconstruction-network weights. HyperDn instead uses its shared predictor to select denoiser hyperparameters in one forward pass, retaining the existing reconstruction procedure.

5 Conclusion and discussion

We demonstrate that oracle hyperparameter supervision transfers across denoising configurations. HyperDn, a single configuration-conditioned predictor trained on oracle labels pooled across source configurations, reduces expensive target labeling through few-shot and zero-shot transfer. The gains persist across different parameter semantics and evaluation criteria. Our analysis bounds the prediction RMSE of a fixed predictor by its source RMSE and mismatches between prediction and oracle-label changes across data and tasks. A shared source oracle bank can thus amortize its construction cost across compatible targets, reducing the marginal labeling cost of extending model-based denoising.

Limitations. Transfer to new denoisers and parameter roles requires few-shot adaptation. Mixed-noise tests use mixtures of seen types, and cross-resolution evaluation only covers the reported datasets and image sizes. Source-bank construction incurs an upfront cost.

Broader impacts. Reusing oracle supervision can reduce the repeated search cost of evaluating new regularizers or data terms. This may make model-based denoising research more accessible to groups with limited computing resources. Wider deployment of denoising may also aid privacy-sensitive surveillance or forensic enhancement.

Acknowledgments and Disclosure of Funding

This work was supported in part by National Science Foundation grants DMS-2208385 and DMS-2208386. The authors thank Syracuse University for providing computational resources through OrangeGrid.

The authors declare no competing interests.

References

Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1692–1700, 2018. doi: 10.1109/CVPR.2018.00182. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Abdelhamed_A_High-Quality_Denoising_CVPR_2018_paper.html.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. doi: 10.1561/2200000016.

Kristian Bredies, Karl Kunisch, and Thomas Pock. Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526, 2010. doi: 10.1137/090769521.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005. doi: 10.1109/CVPR.2005.38.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 32053–32068, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/cf6501108fced72ee5c47e2151c4e153-Abstract-Conference.html.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223. PMLR, 2011. URL https://proceedings.mlr.press/v15/coates11a.html.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance–chrominance space. In *2007 IEEE International Conference on Image Processing (ICIP)*, pages I–313–I–316, 2007. doi: 10.1109/ICIP.2007.4378954.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 667–675, 2016. URL https://papers.nips.cc/paper/2016/hash/8bf1211fd4b7b94528899de0a43b9fb3-Abstract.html.

J. C. De los Reyes, Carola-Bibiane Schönlieb, and Tuomo Valkonen. Bilevel parameter learning for higher-order total variation regularisation models. *Journal of Mathematical Imaging and Vision*, 57(1):1–25, 2017. doi: 10.1007/s10851-016-0662-8.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Eastman Kodak Company. Kodak Lossless True Color Image Suite (PhotoCD PCD0992), 1999. URL http://r0k.us/graphics/kodak/.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for Bayesian optimization. arXiv:1802.02219v4, 2022. URL https://arxiv.org/abs/1802.02219v4.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Tom Goldstein and Stanley Osher. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009. doi: 10.1137/080725891.

Gene H. Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979. doi: 10.1080/00401706.1979.10489751.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Per Christian Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Review*, 34(4): 561–580, 1992. doi: 10.1137/1034115.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015. doi: 10.1109/cvpr.2015.7299156.

Jakob Sauer Jørgensen, Evelina Ametova, Genoveva Burca, Gemma Fardell, Evangelos Papoutsellis, Edoardo Pasca, Kris Thielemans, Martin Turner, Ryan Warr, William R. B. Lionheart, and Philip J. Withers. Core Imaging Library—part I: a versatile Python framework for tomographic imaging. *Philosophical Transactions of the Royal Society A*, 379(2204):20200192, 2021. doi: 10.1098/rsta.2020.0192.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. doi: 10.1109/CVPR.2019.00453. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Karras_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks_CVPR_2019_paper.html.

Daniil Kazantsev, Edoardo Pasca, Martin J. Turner, and Philip J. Withers. CCPi-Regularisation toolkit for computed tomographic image reconstruction with proximal splitting algorithms. *SoftwareX*, 9:317–323, 2019. doi: 10.1016/j.softx.2019.04.003.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018. doi: 10.1109/CVPR.2018.00781.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. Waterloo Exploration Database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, 2017. doi: 10.1109/TIP.2016.2631888.

Babak Maboudi Afkham, Julianne Chung, and Matthias Chung. Learning regularization parameters of inverse problems via deep neural networks. *Inverse Problems*, 37(10):105017, 2021. doi: 10.1088/1361-6420/ac245d.

Ymir Mäkinen, Lucio Azzari, and Alessandro Foi. Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching. *IEEE Transactions on Image Processing*, 29: 8339–8354, 2020. doi: 10.1109/TIP.2020.3014721.

David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423, 2001. doi: 10.1109/ICCV.2001.937655.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. arXiv:1803.02999, 2018. URL https://arxiv.org/abs/1803.02999.

S. Ramani, T. Blu, and M. Unser. Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms. *IEEE Transactions on Image Processing*, 17(9):1540–1554, 2008. doi: 10.1109/TIP.2008.2001404.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, 1992. doi: 10.1016/0167-2789(92)90242-f.

Thanh Trung Vu, Andreas Kofler, and Kostas Papafitsoros. Deep unrolling for learning optimal spatially varying regularisation parameters for total generalised variation. In *Scale Space and Variational Methods in Computer Vision*, volume 15667 of *Lecture Notes in Computer Science*, pages 282–294. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-92366-1_22. URL https://arxiv.org/abs/2502.16532.

Alan Q. Wang, Adrian V. Dalca, and Mert R. Sabuncu. Computing multiple image reconstructions with a single hypernetwork. *Machine Learning for Biomedical Imaging*, 2022(June 2022 issue):1–25, 2022. doi: 10.59275/j.melba.2022-e5ec. URL https://melba-journal.org/2022:017.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Kaixuan Wei, Angelica I. Aviles-Rivero, Jingwei Liang, Ying Fu, Carola-Bibiane Schönlieb, and Hua Huang. Tuning-free plug-and-play proximal algorithm for inverse imaging problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 10158–10169. PMLR, 2020. URL https://proceedings.mlr.press/v119/wei20b.html.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=bJxgv5C3sYc.

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 (CVPR)*, pages 16133–16142, 2023. doi: 10.1109/cvpr52729.2023.01548.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. doi: 10.1109/CVPR.2018.00068. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1219–1229, 2023. doi: 10.1109/cvprw59228.2023.00129.

The appendices are organized as follows: implementation and experimental setup (Appendix A), transfer protocols and additional results (Appendix B), architecture ablations (Appendix C), computational cost and classical comparisons (Appendix D), and predictor transfer: definitions and proof (Appendix E).

A Implementation and experimental setup

A.1 Source configurations and noise generation

Table 6 lists the source settings: 12 STL10 96×96 configurations and 1 WED 256×256 configuration. These settings cover Gaussian, Poisson, Poisson–Gaussian, and black-and-white (bw) or channel-wise (rgb) impulse noise, alone or mixed with Gaussian noise.

Table 6: Source settings. "bw" and "rgb" denote black-and-white and channel-wise impulse noise; NLL denotes the likelihood-based data term used for Poisson or Poisson–Gaussian observations.

#	Dataset/res.	Noise	Data term	Prior
1	STL10 96×96	bw impulse + Gaussian	Huber	TV
2	STL10 96×96	bw impulse	Huber	TGV
3	STL10 96×96	bw impulse	Huber	TV
4	STL10 96×96	Gaussian	Huber	TGV
5	STL10 96×96	Gaussian	L2	TGV
6	STL10 96×96	Gaussian	L2	TV
7	WED 256×256 crops	Gaussian	L2	TV
8	STL10 96×96	Gaussian	Huber	TV
9	STL10 96×96	Poisson–Gaussian	NLL	TGV
10	STL10 96×96	Poisson–Gaussian	NLL	TV
11	STL10 96×96	Poisson	NLL	TV
12	STL10 96×96	rgb impulse + Gaussian	Huber	TGV
13	STL10 96×96	rgb impulse	Huber	TGV

In the synthetic-noise experiments, each noise realization uses a deterministic seed and is kept fixed across the oracle sweep, training, and evaluation. In Table 6, source rows 6, 8, 10, and 11 each contain four noise realizations per clean image, giving 20 000 records from 5 000 images. Row 9 contains 100 records, and the other STL10 sources contain 5 000 each. The WED source (row 7) contains 4 344 records. Records are oracle-labeled noisy observations; different noise realizations of one clean image count separately. Noise is sampled as follows:

- **Gaussian noise (DiffPIR few-shot target on WED 256×256 crops, Gaussian source training sets, and classical-comparison test sets):** per-observation standard deviation $\sigma \sim \mathrm{Uniform}(5/255, 50/255)$, applied additively per pixel.
- **Black-and-white impulse:** salt-and-pepper noise sharing locations and values across channels. Each observation uses a corruption probability drawn from $\mathrm{Uniform}(0.05, 0.30)$, with salt and pepper equally likely.
- **RGB impulse:** salt-and-pepper noise applied independently to each channel. Each observation uses one corruption probability drawn from $\mathrm{Uniform}(0.05, 0.30)$ for all channels, with salt and pepper equally likely.
- **Poisson and Poisson–Gaussian (NLL source training sets):** per-observation photon scale is sampled as $p_{\max} \sim \mathrm{Uniform}(2, 128)$ with $\alpha = 1/p_{\max}$ in the Poisson observation model. Pure Poisson uses read noise $\sigma_{\mathrm{read}} = 0$, while Poisson–Gaussian samples $\sigma_{\mathrm{read}} \sim \mathrm{Uniform}(0.5/255, 16/255)$; the background offset is fixed to 0.
- **Mixed-noise configurations** (Huber-TV bw impulse + Gaussian, Huber-TGV rgb impulse + Gaussian): the impulse corruption probability is sampled from the corresponding single-noise range above, Gaussian $\sigma \sim \mathrm{Uniform}(5/255, 30/255)$, and the two noises are applied in sequence (impulse, then Gaussian).

A.2 Denoising solvers

The following fixed denoisers are used for oracle-label generation and test-time reconstruction.

- **L2-TV denoising:** `skimage.restoration.denoise_tv_bregman` implements split Bregman TV [Goldstein and Osher, 2009] with isotropic TV, per-channel weight $1/(2\lambda)$, and the default stopping settings (`max_num_iter`$= 100$, `eps`$= 10^{-3}$; whichever is reached first).
- **Huber-TV denoising:** an alternating direction method of multipliers solver [Boyd et al., 2011] with a Huber proximal step, 100 iterations, ρ_{TV}=2.0, and ρ_{data}=5.0. It tunes TV weight λ and Huber threshold δ.
- **L2/NLL-TGV denoising:** second-order TGV with (λ, γ) passed as the CCPi [Kazantsev et al., 2019] weight `alpha` and ratio parameter `gamma`. The CIL wrapper [Jørgensen et al., 2021] uses the fast iterative shrinkage-thresholding algorithm (FISTA) for at most 200 outer iterations with relative-update tolerance 10^{-4} (after at least 5 iterations); each CCPi TGV proximal call uses at most 100 iterations.
- **NLL-TV denoising:** CIL/FISTA with TV weight λ and the CCPi `FGP_TV` proximal operator for isotropic TV: at most 200 outer iterations, objective-value stopping threshold 10^{-4}, and at most 100 iterations per proximal call.
- **Huber-TGV denoising:** the same CIL/FISTA and CCPi TGV settings with a Huber data term. It tunes $(\lambda, \gamma, \delta)$, where δ is the Huber threshold.

CIL solves each channel separately, clipping initial and final images to $[0, 1]$. FGP_TV enforces nonnegativity; TGV proximal outputs are clipped to $[0, 1]$. All five denoisers run deterministically on CPU. DiffPIR uses the public 100-iteration sampler (§3.2; Appendix B.1).

A.3 Oracle labels and training objective

All experiments use oracle labels produced by hierarchical grid searches. Except for the metric-specific SSIM and LPIPS study, the target hyperparameter vector maximizes PSNR for each image–configuration pair; that study instead optimizes its named metric. HyperDn fits encoded predictions $\hat{\boldsymbol{q}}$ to the encoded oracle labels $\widetilde{\boldsymbol{q}}$ over the active slots, as described in Section 2.3. For a configuration χ, let $\mathcal{A}_\chi$ be the set of slots that are real hyperparameters rather than padding. For a minibatch $\mathcal{B}$, let $\mathcal{B}_\chi$ contain its examples from configuration χ, and let $Z_\mathcal{B} = \sum_\chi |\mathcal{B}_\chi||\mathcal{A}_\chi|$ count its active scalar targets. The unweighted minibatch loss is

$$\mathcal{L}_{\mathrm{slot}}(\Theta; \mathcal{B}) = \frac{1}{Z_\mathcal{B}} \sum_\chi \sum_{u \in \mathcal{B}_\chi} \sum_{j \in \mathcal{A}_\chi} \left(\hat{q}_{\chi,u,j} - \widetilde{q}_{\chi,u,j}\right)^2. \tag{7}$$

The zero-shot mixed-noise runs use this plain active-slot MSE. The main joint pretraining run and HyperDn/ImageNet init finetuning on DiffPIR, NLM/CBM3D, and SIDD use learned per-role loss weights following Kendall et al. [2018]. Let $s_\rho = \log \sigma_\rho^2$ be the role log-variance, clipped to $[-10, 10]$ when computing the loss, and let $\rho_{\chi,j}$ be the role of active slot j under configuration χ. The weighted minibatch loss is

$$\mathcal{L}_{\mathrm{role}}(\Theta, \boldsymbol{s}; \mathcal{B}) = \frac{1}{Z_\mathcal{B}} \sum_\chi \sum_{u \in \mathcal{B}_\chi} \sum_{j \in \mathcal{A}_\chi} \left[\frac{1}{2} \exp(-s_{\rho_{\chi,j}}) \left(\hat{q}_{\chi,u,j} - \widetilde{q}_{\chi,u,j}\right)^2 + \frac{1}{2} s_{\rho_{\chi,j}}\right]. \tag{8}$$

This loss matches the Gaussian negative log-likelihood form up to constants independent of Θ and $\boldsymbol{s}$. For HyperDn, DiffPIR finetuning starts from the log-variance values learned during source pretraining.

A.4 Predictor architecture and source pretraining

The encoder uses the stem and stages 0–1 of the ConvNeXtV2-base `fcmae_ft_in22k_in1k` checkpoint [Woo et al., 2023]. The denoiser-calibrated readout has $L_{\max}$=3 slots to accommodate Huber-TGV's $(\lambda, \gamma, \delta)$. Role and scale embeddings have 32 dimensions; metadata embeddings have 16. The two-layer trunk MLP uses GELU, LayerNorm, dropout 0.1, and a 512-dimensional output $\boldsymbol{h}$. Each role has a learned log-variance $\log \sigma^2$ for loss weighting.

We use AdamW [Loshchilov and Hutter, 2019] for 80 epochs at batch size 128, head learning rate 2×10^{-4}, and layer-wise learning-rate decay (LLRD; Howard and Ruder, 2018) of 0.8. We use weight decay 10^{-5}, a cosine schedule to 10^{-6} after 10% linear warmup, gradient clipping at 1.0, and mixed precision. Augmentations are random horizontal/vertical flips and $90^\circ/270^\circ$ rotations; we apply no condition-embedding dropout. We randomly hold out 5% of the combined source records and select the checkpoint with the lowest weighted validation loss. Each batch contains one source so image sizes match; its loss averages active scalar targets. We shuffle batches across sources.

A.5 Statistical reporting

We compute standard deviations across seeds, after averaging each seed's metric over the test images. DiffPIR (including metric-specific transfer, ResNet50, and CBSD68/Urban100), SIDD, and FFHQ use sample standard deviation (divisor $n-1$ for n seeds). Mixed-noise, cross-resolution, NLM/CBM3D, architecture-ablation, and classical-comparison results use population standard deviation (divisor n). Fixed controls derived from training-label subsets use the same convention as their learned counterparts. Captions identify three-seed exceptions. Oracles use fixed evaluation images and deterministic searches over finite candidate sets.

A.6 Source and target settings

Table 7: Sources and targets for transfer experiments (continued in Table 8). HyperDn receives the degraded image and target denoiser identity. "Observation metadata" denotes numeric acquisition or noise parameters; noise-type information is represented by categorical codes. N counts target oracle-labeled images used for predictor fitting; DiffPIR's shared calibration uses additional labels (Appendix B.1). The architecture ablations reuse the Huber-TV mixed-noise protocol and change only the tested module.

Experiment	HyperDn source supervision	Target / evaluation	Target labels N	Test-time information beyond the degraded image
DiffPIR/ WED (PSNR)	TV/TGV source checkpoint (Table 6)	WED 256×256; DiffPIR with ImageNet prior and Gaussian noise	1, 2, 4, 16, 32, 64, 128	Gaussian category; $\zeta=1$; no observation metadata
DiffPIR/ WED (SSIM, LPIPS)	Fresh two-source WED TV/TGV checkpoint for each named objective	WED 256×256; DiffPIR with ImageNet prior and Gaussian noise	1, 2, 4, 16, 32, 64, 128	Gaussian category; $\zeta=1$; no observation metadata
Mixed noise, Huber-TV	Two single-noise STL10 sets: Huber-TV/bw impulse and Huber-TV/Gaussian	STL10 test; bw impulse + Gaussian	0	Known composition code; no observation metadata; all-zero-code ablation also tested
Mixed noise, Huber-TGV	Two single-noise STL10 sets: Huber-TGV/rgb impulse and Huber-TGV/Gaussian	STL10 test; rgb impulse + Gaussian	0	Known composition code; no observation metadata; all-zero-code ablation also tested
Cross resolution	TV/TGV source checkpoint (Table 6)	Kodak24: Huber-TV/bw impulse and L2-TGV/Gaussian; CBSD68, Kodak24, Urban100: L2-TV/Gaussian	0	Denoiser/noise code and input size; no observation metadata

Table 8: Source supervision and target evaluation for additional transfer experiments and classical comparisons. Definitions follow Table 7.

Experiment	HyperDn source supervision	Target / evaluation	Target labels N	Test-time information beyond the degraded image
NLM / CBM3D	TV/TGV source checkpoint (Table 6); neither target denoiser nor its scalar appears in source training	WED 256×256; NLM h or CBM3D σ_{psd}	1, 2, 4, 16, 32, 64, 128	Target denoiser and `UNKNOWN` parameter role; no observation metadata
DiffPIR/ CBSD68, Urban100	Seed-matched TV/TGV source checkpoints (Table 6); no WED DiffPIR labels for predictor fitting	Dataset-specific 256×256 crops; DiffPIR with ImageNet prior	1, 2, 4, 16	Gaussian category; $\zeta = 1$; no observation metadata
SIDD sRGB `UNKNOWN`	Fresh 10-source checkpoint: Table 6 excluding Poisson-family rows 9–11	SIDD Medium training crops; official Validation sRGB blocks; Huber-TV	2	`UNKNOWN` noise code; zero observation metadata
DiffPIR/ FFHQ prior	Source checkpoint, then 128 WED DiffPIR labels generated with the ImageNet prior	FFHQ faces; DiffPIR with an FFHQ-trained prior	0 FFHQ	Gaussian category; $\zeta = 1$; no observation metadata
Classical/ SURE comparison	TV/TGV source checkpoint (Table 6)	L2-TV/Gaussian on CBSD68, Kodak24, and Urban100	0	HyperDn: Gaussian category, no σ or observation metadata; SURE: true σ

B Transfer protocols and additional results

B.1 Few-shot transfer to DiffPIR

DiffPIR few-shot finetuning. HyperDn and ImageNet init use the same DiffPIR finetuning schedule: 150 epochs, head learning rate $\mathrm{LR}_{\mathrm{head}} = 10^{-4}$, LLRD of 0.1, 30% linear warmup, batch size at most 8 (or the target-label count when smaller), AdamW with weight decay 10^{-5} and gradient clipping at 1.0, no mixed precision, and the role-weighted loss in Eq. (8). We use all target labels for finetuning and select the last-epoch checkpoint.

HyperDn initializes from the *full* source-pretrained checkpoint (backbone, head, embeddings, and weighting parameters). The denoiser-dependent scale and offset tables in Eq. (3) are initialized at 1 and 0 before source pretraining. Their DiffPIR-specific rows receive no source-task gradients and are loaded from the source checkpoint for target finetuning. ImageNet init loads the ImageNet-pretrained ConvNeXtV2 backbone and uses a randomly initialized head (seeded per run).

Afkham baseline. Afkham trains from random initialization for each label budget and seed: 500 epochs for $N \leq 2$ and 100 otherwise. It uses AdamW with learning rate and weight decay both 10^{-3}, batch size $\min(N, 8)$, mixed precision, and a cosine schedule after one warmup epoch. It minimizes unweighted prediction MSE on all N labels and uses the final checkpoint. Following the convolutional design of Maboudi Afkham et al. [2021], adapted from the original single-channel deblurring setting to RGB DiffPIR denoising, this target-only neural network predicts regularization parameters for one inverse-problem configuration. It consists of three convolutional blocks (Conv 5×5, padding 2 → BatchNorm → ReLU → MaxPool 2×2, with channel widths 3→16→32→64), an adaptive average pool to 4×4, and an MLP head with input width 1024, hidden widths 256→128→64, and ReLU between hidden layers, followed by a linear output for the active DiffPIR target; this is approximately 369 K parameters in total.

ImageNet-pretrained ResNet50 control. We also fully finetune a standard ImageNet-V1-pretrained ResNet50 [He et al., 2016] on the same DiffPIR training subsets for each seed and evaluate it on the same 50 test images. Every trainable layer uses the same peak learning rate 10^{-5}, with no layer-wise decay; otherwise it follows the 150-epoch target schedule above. This compares HyperDn with ResNet50 at the same target-label budgets.

DiffPIR reconstruction and evaluation. We predict only $\log_{10} \lambda_{\mathrm{eff}}$, where $\lambda_{\mathrm{eff}} = \lambda_{\mathrm{cache}} \sigma^2$ converts the cached DiffPIR search parameter to the solver-interface coefficient. This conversion uses the training observation's noise variance; no noise-level input is needed for prediction. We reconstruct with 100 DiffPIR iterations using the ImageNet-trained guided-diffusion prior [Dhariwal and Nichol, 2021]; every method fixes $\zeta = 1$, selected because all 4,344 WED training images chose it in a preliminary joint $(\lambda_{\mathrm{cache}}, \zeta)$ sweep. The reported few-shot budgets count image-specific λ_{eff} labels used for predictor fitting, conditional on this shared calibration. The preliminary sweep uses additional target-side supervision and is excluded from those fitting budgets. Evaluation uses the same deterministic first 50 images of the 400-image validation split, identical degraded inputs, and float-$[0, 1]$ PSNR without border cropping; the oracle PSNR is 31.13 dB.

Complete budget results. Table 9 gives the exact values behind the few-shot transfer curve in Section 3.2 and includes the ResNet50 control.

Table 9: DiffPIR PSNR (dB) ↑ across all target-label budgets; oracle: 31.13 dB. Best mean per budget in **bold**.

N	Afkham	ImageNet init	ResNet50	HyperDn
1	23.06±1.66	24.98±2.32	27.31±.94	**29.37±.77**
2	27.26±1.79	27.73±1.40	27.96±.77	**30.23±.56**
4	28.23±2.00	28.69±.89	27.75±.99	**30.62±.23**
16	27.84±2.46	29.17±.95	28.70±.40	**30.84±.12**
32	29.64±.53	30.53±.29	29.63±.36	**31.03±.01**
64	30.28±.40	30.90±.09	30.44±.20	**31.09±.01**
128	30.81±.08	30.94±.14	30.81±.08	**31.10±.01**

B.2 Zero-shot mixed-noise transfer

For each mixed-noise target, we train a fresh predictor using the two corresponding single-noise source sets from the STL10 *train* split ($5\,000$ clean images at 96×96; source record counts are given in Appendix A): Huber–TV/bw impulse with Huber–TV/Gaussian, or Huber–TGV/rgb impulse with Huber–TGV/Gaussian. Mixed-noise target evaluation in Table 2 uses $1\,000$ images from the STL10 *test* split at 96×96, with no overlap with the training images; no mixture label enters training or model selection. We follow the source-pretraining optimizer, schedule, and augmentations in Appendix A, with 80 epochs, batch size 256, and equally weighted MSE over the active parameter dimensions.

The multi-hot noise-attribute code has six slots: (unknown, has-Gaussian, has-Poisson, has-impulse, impulse-is-bw, impulse-is-rgb). Thus bw impulse + Gaussian encodes to $[0, 1, 0, 1, 1, 0]$, and rgb impulse + Gaussian encodes to $[0, 1, 0, 1, 0, 1]$. Neither target receives observation metadata. Replacing all six categorical-code entries with zeros at inference changes the five-seed mean PSNR by -0.0074 and -0.0077 dB on the bw and rgb mixture targets, respectively, relative to using the mixture codes above.

B.3 Cross-resolution transfer

For cross-resolution evaluation on Kodak24 (512×768), L2-TGV uses the Gaussian noise range of its STL10 source, and Huber-TV uses the black-and-white impulse corruption-probability range of its STL10 source. For L2-TV, source supervision includes Gaussian-corrupted STL10 96×96 and WED 256×256 (Table 6). The predictor uses the joint source-pretrained checkpoint described in Appendix A; results are reported in Table 3 and Figure 6.

B.4 Previously unseen nonlocal denoisers and parameter roles

NLM, CBM3D, and their tuned scalars are unseen during source pretraining. Both targets use additive Gaussian noise with per-image $\sigma \sim \mathrm{Uniform}(5/255, 50/255)$ and data-term identifier $d = \mathrm{L2}$. We tune NLM's filtering strength h in `skimage.restoration.denoise_nl_means` and CBM3D's assumed noise standard deviation σ_{psd} in `bm3d.bm3d_rgb` [Mäkinen et al., 2020], using RGB input and package defaults for the remaining denoising settings. In both cases, we use a hierarchical grid search on a logarithmic scale over $[0.003, 0.30]$. We assign the scalar the `UNKNOWN` role, fine-tune on N images selected by seed from a 128-image WED 256×256 target-label pool, and evaluate on 50 test images outside this pool. HyperDn begins from the variational source checkpoint. ImageNet init uses the identical architecture with an ImageNet-pretrained backbone and a randomly initialized head; both follow the 150-epoch finetuning schedule in Appendix B.1. Afkham is trained from scratch for 500 epochs when $N \leq 2$ and 100 epochs otherwise, using AdamW with learning rate and weight decay both 10^{-3}, batch size $\min(N, 8)$, and a cosine schedule after one warmup epoch. It minimizes unweighted prediction MSE and uses the final checkpoint.

Table 4 reports the full results. HyperDn has the highest mean PSNR at all seven budgets despite the change in hyperparameter meaning.

B.5 Cross-dataset transfer and real camera noise

Table 10 summarizes results at selected label budgets for additional datasets and real camera noise.

Table 10: Additional dataset and real-noise targets: PSNR (dB) ↑. Controls use the listed N. For SIDD, the noise code is `UNKNOWN` and the observation metadata is zero. The oracle is found by per-image search over a finite candidate set. Detailed results appear below.

Shift / target	N	HyperDn	Strongest control	Oracle
New dataset: DiffPIR/CBSD68	16	**30.05**	27.30 (Afkham)	30.18
New dataset: DiffPIR/Urban100	16	**31.18**	27.92 (Afkham)	31.41
Real noise: SIDD sRGB `UNKNOWN`	2	**33.732**	31.893 (fixed)	35.640

B.5.1 DiffPIR on CBSD68 and Urban100

We repeat DiffPIR few-shot transfer on 256×256 center crops from CBSD68 and Urban100. The first 50 images form the evaluation set; the remaining 18 CBSD68 or 50 Urban100 images form the target-label pool. HyperDn starts from the non-DiffPIR source checkpoint with the same seed, while Afkham starts from scratch; no WED DiffPIR oracle label is used to fit either predictor. All methods use 100 DiffPIR iterations, the shared $\zeta = 1$ setting from Appendix B.1, and the same per-image Gaussian range as the main WED experiment.

Table 11: Cross-dataset DiffPIR PSNR (dB) ↑ (three training seeds), evaluated on the same 50 held-out images per dataset. Parenthetical values are oracle PSNR.

Dataset (oracle)	Method	N=1	N=2	N=4	N=16
CBSD68 (30.18)	HyperDn	$\mathbf{28.98\pm.52}$	$\mathbf{29.05\pm.56}$	$\mathbf{29.46\pm.40}$	$\mathbf{30.05\pm.02}$
	Afkham	23.35 ± 3.16	25.65 ± 1.40	$26.78\pm.21$	$27.30\pm.14$
Urban100 (31.41)	HyperDn	$\mathbf{28.91\pm2.23}$	$\mathbf{30.39\pm.45}$	$\mathbf{30.79\pm.08}$	$\mathbf{31.18\pm.05}$
	Afkham	$21.92\pm.45$	25.56 ± 2.18	$27.87\pm.39$	$27.92\pm.93$

B.5.2 Real SIDD sRGB noise

We test two-shot adaptation on SIDD sRGB [Abdelhamed et al., 2018] using Huber–TV with the categorical noise code set to `UNKNOWN` and the observation-metadata vector set to zero. For this experiment, HyperDn is pretrained on 10 source configurations, excluding all three Poisson and Poisson–Gaussian configurations. For each seed, it is then adapted using two target labels drawn from 160 deterministic SIDD Medium crops. HyperDn and ImageNet init follow the 150-epoch finetuning schedule in Appendix B.1, with the same initialization rules and final-checkpoint selection. Evaluation uses all $1{,}280$ official Validation sRGB blocks grouped into 40 captures. The fixed control uses the geometric mean of the same two target-label parameter vectors for each seed and applies it to every validation image.

Table 12: SIDD sRGB PSNR with two target training labels. The noisy input is deterministic.

Metric	Noisy input	Fixed (same 2 labels)	ImageNet init	HyperDn
PSNR (dB) ↑	23.662	31.893 ± 2.455	31.796 ± 3.401	$\mathbf{33.732\pm.490}$

B.6 Zero-shot image-domain and diffusion-prior shift

Starting from the source-pretrained checkpoint, we fine-tune HyperDn using 128 WED DiffPIR oracle labels generated with the ImageNet diffusion prior. We evaluate on the first 50 images in DiffPIR's FFHQ validation list, resized to 256×256 with Lanczos interpolation, using the official `diffusion_ffhq_10m` prior. This prior is unseen by the predictor. Each image receives additive Gaussian noise with $\sigma \sim \mathrm{Uniform}(5/255, 50/255)$. No FFHQ image or FFHQ-prior oracle label is used to train or adapt HyperDn. For each seed, the fixed-λ control uses the same 128 WED training labels as HyperDn. Each cached oracle λ_{cache} is first multiplied by its corresponding WED noise variance to obtain λ_{eff}, the coefficient passed to DiffPIR. We take the geometric mean of these converted values and apply this single coefficient unchanged to every FFHQ test image. This coefficient is fixed before FFHQ evaluation, with no tuning on FFHQ or direct optimization of mean WED PSNR. Every method uses $\zeta = 1$ and the FFHQ prior at evaluation.

The full results are reported in Table 5 in the main paper.

C Architecture ablations

Both ablations use the Huber-TV bw impulse + Gaussian zero-shot mixed-noise setting (§3.3) and the two corresponding single-noise STL10 source training sets. We vary either the prediction head or the ConvNeXtV2 stages, keeping the remaining setup fixed, to test the design choices in Section 2.

C.1 Prediction head design

The head ablation compares HyperDn against two alternatives with identical image, data-term, prior, and metadata embeddings: a plain MLP conditioned on the image and configuration that treats all padded output slots in the same way, and a slot-aware affine variant that scales and shifts the output using role and scale information but omits the bilinear interaction and the denoiser-dependent scale and offset. All variants share the same backbone, training schedule, and seed set; only the head differs.

Table 13 reports the comparison. Parameter relative error averages $|\hat{p}_j - p_j^\star| / \max(|p_j^\star|, 10^{-8})$ over active slots and then over images, using raw hyperparameter values after decoding log-scale outputs with 10^x.

Table 13: Prediction-head ablation on zero-shot mixed noise (3 seeds). PSNR and gap to oracle are in dB. All variants share the same encoder and training protocol. Parameter relative error is averaged over active slots after decoding log-scale outputs.

Head variant	PSNR ↑	Gap (dB) ↓	Param. rel. err. ↓
Conditioned MLP	23.72 ± 0.10	0.81 ± 0.10	0.255
Slot-aware affine	24.02 ± 0.14	0.50 ± 0.14	0.188
Denoiser-calibrated (**ours**)	$\mathbf{24.29 \pm 0.09}$	$\mathbf{0.23 \pm 0.09}$	$\mathbf{0.160}$

Role and scale information improves prediction even without the bilinear interaction: the slot-aware affine variant improves over the plain conditioned MLP ($23.72 \rightarrow 24.02$ dB). The full denoiser-calibrated slot readout adds another 0.27 dB over this intermediate variant and increases PSNR by 0.57 dB over the plain conditioned MLP ($23.72 \rightarrow 24.29$). It also reduces parameter relative error by 37% ($0.255 \rightarrow 0.160$) and narrows the gap to oracle from 0.81 dB to 0.23 dB. These gains support the combined contribution of role and scale information, the bilinear interaction, and denoiser-specific scaling and offsets in Eq. (3).

C.2 Backbone stage selection

The stage ablation tests whether hyperparameter prediction benefits from deeper semantic features. We compare the default stage choice (0–1) against a single shallow stage (*stage-0 only*) and the full backbone (*stages 0–3*), keeping all other settings identical (same training subset, schedule, head, and seeds).

Table 14: Backbone-stage ablation on the zero-shot mixed-noise setting (3 seeds). *Stages* $(0, 1)$ is our default.

Stage selection	Backbone params ↓	PSNR ↑	Gap to oracle (dB) ↓	Param. rel. err. ↓
Stage 0 only	0.42M	24.06 ± 0.26	0.46 ± 0.26	0.174
Stages $(0, 1)$ (**ours**)	2.18M	$\mathbf{24.29 \pm 0.09}$	$\mathbf{0.23 \pm 0.09}$	$\mathbf{0.160}$
Full backbone (stages 0–3)	87.69M	23.96 ± 0.06	0.56 ± 0.06	0.316

Stages 0 and 1 give the best result: removing stage 1 lowers PSNR by 0.23 dB, while adding the deeper semantic stages 2 and 3 lowers PSNR by 0.33 dB with more than 40 times as many backbone parameters. This pattern is consistent with hyperparameter prediction relying on low-level noise and edge statistics rather than semantic content. Relative parameter error also nearly doubles from 0.160 to 0.316 when deep stages are included, indicating that including deeper stages lowers performance under this training setup.

D Computational cost and classical comparisons

D.1 Hardware and oracle-labeling cost

Predictor training and evaluation run on 1–5 NVIDIA RTX 4090 GPUs (24 GB VRAM each) in parallel across seeds and configurations, with one card per run. Source pretraining takes ~ 2.5 hours per seed. DiffPIR few-shot finetuning takes ≤ 1 minute for ≤ 32 target labels and up to ~ 30 minutes for 128 target labels for each seed and method.

Oracle-label generation is the costly stage. TV/TGV source labels are generated on CPUs, with each image searched independently; in the paper run, the full 13-source search was distributed across a shared CPU cluster with approximately 600 CPU cores available and about 2 GB RAM per core, over roughly 10 days. DiffPIR target-label generation for the 4344 WED 256×256 training images is GPU-bound; in the paper run, these labels were generated on a heterogeneous, dynamically scheduled pool of 8–16 NVIDIA L40S and A100 GPUs over roughly 2 days.

The classical-method comparison (Table 16) times HyperDn and SURE on the same machine. HyperDn selects parameters with one GPU forward pass; the downstream TV solver runs on CPU. SURE and the oracle sweep both require many CPU TV solver calls per image.

After fixing $\zeta = 1$ using the shared training-set calibration, generating 2 image-specific λ labels for predictor fitting requires 40 DiffPIR reconstructions at 20 candidates per image, versus 1,280 for Afkham's 64 labels. At roughly 10 seconds per reconstruction, generating these labels would take 0.11 and 3.56 GPU-hours if run sequentially, respectively.

D.2 Batched predictor throughput

We time the GPU forward pass in single precision (FP32), processing each test set as one batch of equally sized images. The three resolutions are: WED 256×256 center crops, CBSD68 at 321×481, and Kodak24 at 512×768. For timing only, we rotate CBSD68 and Kodak24 portrait images 90° to landscape so that all images in each dataset have the same shape. Table 15 reports single-batch timing, throughput, and peak GPU memory for each set.

At 256×256, the forward pass takes 0.78 ms per image (1280 img/s). At 512×768, throughput falls to 212 img/s, approximately in inverse proportion to pixel count across these measurements. Per-batch wall-clock time for the entire test set is 0.11–0.31 s in all three cases. These times measure only the GPU forward pass. Table 16 instead measures parameter-selection time at batch size 1, including CPU→GPU transfer and Python overhead.

Table 15: HyperDn single-batch throughput (FP32; time and memory ↓, throughput ↑). Batch time is mean±population SD across 24 forward passes after 3 warmup passes, with CUDA synchronization. Each row uses one batch of size bs and equally sized images. "Mem/img" divides peak forward memory by bs, so fixed model weights are shared across the batch.

Dataset	Shape	bs	Batch time (ms)	Per-image (ms)	Throughput (img/s)	Mem/img
WED (center-cropped)	256×256	400	312.5 ± 6.8	0.78	1280	31 MB
CBSD68	321×481	68	124.4 ± 0.8	1.83	546	73 MB
Kodak24	512×768	24	113.1 ± 1.0	4.71	212	187 MB

D.3 Comparison with classical parameter selection

We compare HyperDn with Monte-Carlo Stein's unbiased risk estimate (SURE) [Ramani et al., 2008] for per-image selection of λ in L2-TV denoising under additive Gaussian noise.

Setup. All methods operate on the same three out-of-distribution test datasets (CBSD68 321×481, Kodak24 512×768, Urban100 $\sim1024\times768$) with identical per-image Gaussian noise levels σ:

- **Avg-oracle (fixed $\bar{\lambda}$):** applies the arithmetic mean of the $20\,000$ STL10 L2-TV Gaussian training labels to every test image;
- **SURE** [Ramani et al., 2008]: minimizes Monte-Carlo SURE over 15 logarithmically spaced λ values in $[10^{-5}, 10^{0.5}]$, using the true noise level σ. One standard-normal probe per image is fixed across candidates, with perturbation scale 10^{-3} and a seed derived from the observation's noise seed;
- **HyperDn:** a single forward pass of the source-pretrained HyperDn model;
- **Oracle:** per-image hierarchical grid search using the clean reference image.

Results. Table 16 reports PSNR, gap to oracle, and per-image wall-clock cost across three out-of-distribution test datasets spanning 321×481 (CBSD68) to $\sim1024\times768$ (Urban100). The non-adaptive avg-oracle baseline trails the per-image oracle by 1.19–1.58 dB on average, showing the limitations of this training-mean fixed-λ baseline on these datasets.

HyperDn matches or outperforms SURE in quality on every dataset: its gap to oracle is lower on all three datasets (CBSD68: 0.01 vs. 0.06, Kodak24: 0.27 vs. 0.32, Urban100: 0.02 vs. 0.08 dB). It is also much cheaper per image. SURE selects λ by evaluating a risk estimate over many candidate values; each candidate requires a TV solve on the observation and another on its perturbed copy. Its measured per-image parameter-selection time is 2.1 s on CBSD68, 5.7 s on Kodak24, and 11.2 s on Urban100. HyperDn selects λ with one forward pass (14 ms at 321×481, 19 ms at 512×768, 47 ms at $\sim1024\times768$) and then invokes the same TV solver once. Thus the learned predictor replaces SURE's per-image search with one learned prediction while retaining near-oracle reconstruction quality.

Table 16: Classical comparison on L2-TV Gaussian across three out-of-distribution test datasets. Per-image σ is sampled from the STL10 training range. "Mean gap to oracle" is oracle PSNR minus method PSNR. "Inference" reports the per-image λ-selection cost. Each method then uses the same TV solver for final reconstruction, whose cost is excluded. HyperDn is reported at batch 1 to match SURE's per-image λ search.

Dataset	Method	Mean PSNR ↑	Mean gap to oracle (dB) ↓	Inference ↓
CBSD68 (321×481)	Avg-oracle	25.73	1.19	—
	SURE	26.85	0.06	~2100 ms
	HyperDn	$\mathbf{26.91\pm0.0029}$	$\mathbf{0.01\pm0.0029}$	**14 ms**
Kodak24 (512×768)	Avg-oracle	26.44	1.58	—
	SURE	27.70	0.32	~5720 ms
	HyperDn	$\mathbf{27.75\pm0.0030}$	$\mathbf{0.27\pm0.0030}$	**19 ms**
Urban100 ($\sim1024\times768$)	Avg-oracle	24.49	1.37	—
	SURE	25.78	0.08	~11220 ms
	HyperDn	$\mathbf{25.84\pm0.0045}$	$\mathbf{0.02\pm0.0045}$	**47 ms**

E Predictor transfer: definitions and proof

E.1 Outputs and data

We use the same fixed predictor f to analyze target-task error between encoded hyperparameter predictions and oracle labels. Each task fixes the denoising configuration, the rules for supplying task information, and the oracle procedure, including search rules, solver settings, random seeds, and tie-breaking. A sample x includes the observed image, any per-image observation metadata, and reference information for its oracle label; the predictor reads only the image and supplied task information. The distributions $P_1, \dots, P_K, P_T$ share a measurable sample space allowing different image sizes. For source k, $X_S^{(k)} \sim P_k$ and $X_T \sim P_T$. Each source prediction and oracle procedure is defined on source and target samples; the target procedure is defined on target samples. Thus, $q_k^\star(X_T)$ applies source task k's oracle procedure to the target sample and reference information.

For a task with $m \geq 1$ active encoded coordinates, retain those coordinates in the padded slot order of Section 2, set inactive coordinates to zero, and divide both the prediction and label by $\sqrt{m}$. The vectors $f_k(x), f_T(x), q_k^\star(x), q_T^\star(x)$ denote these normalized outputs in the common space $\mathbb{R}^{L_{\max}}$. Their squared Euclidean distance equals the mean squared prediction error over the task's active encoded coordinates. Encoding, slot order, and normalization are fixed throughout. Both discrepancies compare prediction errors in these encoded coordinates; their interpretation follows the specified order and scales, including when tasks have different parameter meanings.

All prediction and oracle maps are measurable and finite-valued. Assume $\varepsilon_S(f), \varepsilon_T(f) < \infty$ and $\mathbb{E}_{P_T}\|f_k(X_T) - q_k^\star(X_T)\|^2 < \infty$ for every source k. For each k, $\Pi(P_k, P_T)$ is the set of joint distributions of $(X_S^{(k)}, X_T)$ with marginals P_k, P_T. For generalization across image distributions under the source tasks, the data term measures changes in the predictor's errors, taking the infimum over this set. For generalization to the target task, the task term measures error changes on the same target sample. The weights α_k specify the pooled evaluation average in Eq. (4).

E.2 Proof

Proof of Theorem 1. Choose $\pi_k \in \Pi(P_k, P_T)$ for each source k. Choose source k with probability α_k, then draw $(X_S^{(k)}, X_T)$ from π_k. For every selected source, adding and subtracting the intermediate predictions and labels gives

$$\begin{aligned} f_T(X_T) - q_T^\star(X_T) = f_k(X_S^{(k)}) - q_k^\star(X_S^{(k)}) \\ + \big([f_k(X_T) - f_k(X_S^{(k)})] - [q_k^\star(X_T) - q_k^\star(X_S^{(k)})]\big) \\ + \big([f_T(X_T) - f_k(X_T)] - [q_T^\star(X_T) - q_k^\star(X_T)]\big). \end{aligned}$$

All three terms have finite second moments. Take the L^2 norm, the square root of the expected squared Euclidean norm, under this sampling law. Each pairing has target marginal P_T and $\sum_k \alpha_k = 1$, so the left-hand side has norm $\varepsilon_T(f)$. The triangle inequality gives

$$\begin{aligned} \varepsilon_T(f) \leq \varepsilon_S(f) + \Delta_{\text{task}}(f) \\ + \left(\sum_k \alpha_k \mathbb{E}_{\pi_k} \big\| [f_k(X_T) - f_k(X_S^{(k)})] - [q_k^\star(X_T) - q_k^\star(X_S^{(k)})] \big\|^2 \right)^{1/2}. \end{aligned}$$

Taking the infimum over $\pi_1, \dots, \pi_K$ gives Eq. (6). □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately delimit the paper's contributions: transferable oracle-label supervision, one configuration-conditioned predictor trained across heterogeneous denoising configurations, and empirical transfer across denoisers, objectives, noise composition, image domains and diffusion priors, and resolution. Sections 2–3 support the main claims; Appendix B gives the additional experiments. Limitations are stated in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 states the main scope boundaries: DiffPIR transfer is evaluated with few target labels; zero-shot mixed-noise transfer is tested only when the constituent noise types are seen separately; cross-resolution evaluation covers the reported datasets and image sizes; and HyperDn amortizes the one-time source-bank construction cost across compatible targets.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 2.4 states the fixed-predictor transfer bound and defines its RMSE terms. Appendix E gives the output conventions, pairing and integrability conditions, and complete proof for pooled sources.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 defines source versus target supervision, the oracle-label construction, evaluation metrics, target-label budgets, and the transfer settings. Section 2 specifies the padded hyperparameter representation and denoiser-calibrated slot readout. Appendices A and B give the denoiser specifications, source/checkpoint provenance, data splits, noise ranges, optimization settings, complete budget curves, and generalization protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The supplementary package includes the model implementation, source training code, DiffPIR and CBM3D few-shot training and evaluation code with both controls, and parameter-prediction and denoising interfaces. It provides one source-pretrained checkpoint and two-shot DiffPIR and CBM3D checkpoints, all with seed 43, together with the corresponding oracle label banks and setup instructions. It does not include raw images, external diffusion-prior weights, or the code and checkpoints for every experiment in the paper. Our code is MIT-licensed; the included ConvNeXt V2-derived weights retain the upstream CC BY-NC 4.0 terms.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3 states each transfer axis, source and target supervision, oracle construction, metric, label budget, and evaluation set. Section 2 specifies the shared encoder, configuration-conditioned head, and denoiser-calibrated slot readout. Appendix A gives the backbone, optimizer, schedule, batch size, loss weighting, source composition, and validation; Appendix B gives DiffPIR finetuning details and the protocols for denoiser, dataset, real-noise, and diffusion-prior shifts. Architecture choices are analyzed in Appendices C.1 and C.2.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Section 3.1 states the default mean$\pm$std reporting convention. Appendix A specifies the sample/population standard-deviation definitions and aggregation, with three-seed exceptions identified in the corresponding captions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Predictor training and evaluation were conducted on 1–5 NVIDIA RTX 4090 GPUs (24 GB VRAM each) running in parallel across seeds and configurations, with one card per run. Joint source pretraining takes approximately 2.5 hours per seed; zero-shot mixed-noise training runs are substantially shorter; DiffPIR few-shot finetuning is $\leq$ 1 minute for $N \leq 32$ and up to $\sim$30 minutes for N=128 per (seed, method) combination (Appendix D). Oracle label generation depends on the solver: for the TV/TGV source training sets it is CPU-bound and embarrassingly parallel, with the full 13-source sweep parallelized on a shared CPU cluster (approximately 600 cores, about 2 GB RAM per core) for roughly 10 days of elapsed wall-clock time; for the DiffPIR target oracle sweep over the 4344 WED 256$\times$256 training images, labels were generated on a heterogeneous, dynamically scheduled pool of 8–16 NVIDIA L40S and A100 GPUs over roughly 2 days of elapsed wall-clock time. Preliminary runs for architecture selection and schedule tuning are not tabulated.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses established public benchmark datasets, with no participant interaction or private-data collection. Section 5 discusses the ethical considerations associated with denoising.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 5 discusses both. On the positive side, HyperDn reduces repeated oracle labeling by reusing a shared source bank across compatible targets. On the negative side, broader deployment of model-based denoising may aid privacy-sensitive surveillance or forensic enhancement.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: HyperDn produces hyperparameter vectors that configure existing denoising solvers.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Third-party assets are credited in the paper and below. The included HyperDn checkpoints derive from ConvNeXtV2 and retain its CC BY-NC 4.0 terms; raw datasets and diffusion-prior weights are not redistributed. DiffPIR [Zhu et al., 2023] code is used under the MIT license from its official repository. ConvNeXtV2 [Woo et al., 2023] is used from the official repository, whose terms apply MIT to the code and CC BY-NC 4.0 to the released pretrained weights. SIDD [Abdelhamed et al., 2018] is used under the MIT terms stated on its official project page. FFHQ [Karras et al., 2019] is used only for non-commercial evaluation; its official notice states that individual images retain the per-image Creative Commons, public-domain, or U.S.-Government-work terms recorded in the metadata, while the metadata, download script, and documentation are CC BY-NC-SA 4.0. The ImageNet diffusion checkpoint is OpenAI's guided-diffusion release [Dhariwal and Nichol, 2021], provided under the MIT license. The FFHQ diffusion checkpoint comes from the official DiffPIR model zoo; checkpoint-specific licensing remains unverified, and the checkpoint is not redistributed. The WED project page and the Berkeley resources underlying CBSD68 permit non-commercial research and educational use. STL10 and Urban100 provide academic download/citation instructions but no separate explicit dataset license; we use these datasets only for academic training or evaluation and do not redistribute the raw images. Kodak24 is used only for evaluation; its provenance is cited, and its images are not redistributed.

The implementation uses `timm` (Apache-2.0), `scikit-image` (BSD-3-Clause), CIL [Jørgensen et al., 2021] and the CCPi-Regularisation-Toolkit [Kazantsev et al., 2019] (both Apache-2.0), and LPIPS 0.1.4 [Zhang et al., 2018] from its BSD-2-Clause repository. The CBM3D experiment uses `bm3d` 4.0.3 [Mäkinen et al., 2020] under the Tampere University non-commercial terms; we use it only for academic evaluation, cite the method [Dabov et al., 2007], and do not redistribute the software. The ResNet50 control [He et al., 2016] uses torchvision's `IMAGENET1K_V1` weights; torchvision code is BSD-3-Clause, checkpoint-specific licensing remains unverified, and the weights are not redistributed.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplementary package documents source training, DiffPIR and CBM3D few-shot training and evaluation with both controls, parameter prediction, and denoising. Installation, data preparation, oracle label banks, input/output conventions, the provenance of all three seed-43 checkpoints, and the separate code and weight licenses are stated in the package. Appendices A and B document the complete experimental protocol, and Section 5 states the method limitations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Under human supervision, we used ChatGPT and Claude to assist with coding, developing and checking the theoretical analysis, checking the correctness of results, and manuscript revision. The authors take full responsibility for the content of the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.