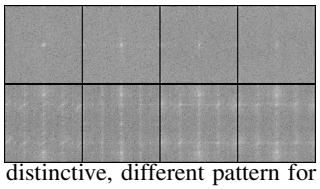
We thank all reviewers for their insightful comments. We will improve readability and structure as suggested by R1 and R3.

[R1]: Residual DDH (R-DDH). As suggested, R-DHH is explored and indeed outperforms DDH but only by a small margin (R-DHH 2.36/3.46 vs. DHH 2.68/3.50 for cAPD/sAPD). The corresponding results in Table 6 for R-DDH are 7.2/12.8/19.5/24.2/29.1 for a shift of 10/20/30/40/50. At first sight, the UDH robustness to constant-shift corruptions might be attributed to its "residual" property. We found that the shortcut helps R-DDH behave like UDH in the early stage of training, however, with both C and S as the input of network H, H eventually learns to adapt to encode S dependent on C. To further prove this, we add the residual S_e in the R-DDH to a random cover, no S can be revealed. Such encoding dependence is due to the C shortcut in R-DHH facilitates training instead of behaving like an independent "cover noise" as in UDH, highlighting the importance of universal property of UDH.

Corruption for watermarking. In Sec. 5, we compare with HiDDeN[29] which randomly adds one corruption type per mini-batch and evaluates all corruptions separately. Thus, we mimic HiDDeN but adopted our dividing strategy instead of their swap strategy, resulting in a significant performance boost as shown in Table 3. Note [23,25] are for the task of LFM not watermarking.

Concerns regarding LFM task. Different from StegaStamp[23] applying various corruptions and a delicate loss design, we only used perspective warp[1] in training with a simple loss described in lines 94-95. Despite the simplicity, UDH achieves impressive performance as pointed out by R1. We confirmed that DDH fails in the setting of Table7, which is consistent with the Baluja result in Figure 2 of LFM[25]. Training a CDTF network as in [25] requires a 1.9TB display-camera pair dataset and finally leads to over-fitting, resulting in a performance drop for unseen display-camera pairs. Excluding the need for such collected display-camera data, our approach is not over-fitting to any certain type of hardware devices, thus the comparison in Table7 is fair to a large extent. Additional new comparison with [23] shows a mean accuracy for our UDH and [23] is 98.8% & 97.6%, respectively for hiding 100 bits in 256×256 under our hardware setups. During evaluation, [23] is not flexible with changing #bits, while ours is flexible and also versatile for hiding image.

[R2]: JPEG compression. In essence, the JPEG compressed container image is simulated by only adding a "noise" to the original container image. Since this "noise" is JPEG$(C') - C'$, and has the "JPEG compression" pattern, the pipeline (H and R) would be naturally adapted to be robust to it even though it is only added as "noise" without back-propagation. Table3 supports the effectiveness of training with our JPEG with an APD of 23.6 vs. 60.1 (Mask[29]) and 58.5 (Drop[29]). Indeed, JPEG suppresses HF content and leads to a performance drop (9.6 vs. 23.6 see Table3) even with such adaptation in training. The Fourier analysis on the right reveals a distinctive, different pattern for training without JPEG augmentation (top) from that with it (bottom). **Architecture & training method influence.** We apply U-Net and ResNet of 3 sizes for H and R, experiment with SGD, ADAM, and decreasing and cyclic learning rate. All variants show a consistency with the results in Sec. 4.1. **Data efficiency.** We clarify that the data efficiency claim is made for the task of LFM (see line310-314). Compared to [25], UDH is data-efficient in the sense that arduous collection of a separate (1.9TB) dataset is not necessary, since UDH can achieve its robustness by training on ImageNet. **Image size.** For both DDH and UDH, the image size during evaluation is flexible. We confirmed the results in Table 1 are equivalent for image size ranging from 64, 128, 256 to 512.

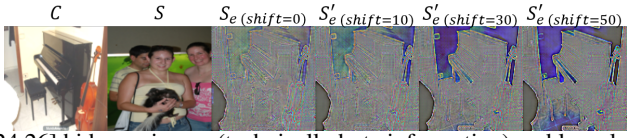


[R3]: Choice of perturbations in Table 5 & 6. Applying constant shift to the cover images C (Table Left) has limited influence on DDH and UDH. Adding random noise to container image C' (Table Right) degrades performance on DDH and UDH but more on DDH. Together With Table 5 & 6, we conclude: UDH is more robust to corruptions on C', while DDH is more robust to HF corruptions on C. As suggested, for DDH we visualize the difference between S_e and encoding S'_e with a constant shift on C (See image below). S_e and S'_e are different, naturally leading to $(C + S_e) + shift \neq (C + shift) + S'_e$ explaining why DDH is not robust to constant shift on container image C'.

Arch	10	30	50
DDH	3.6	3.8	4.1
UDH	3.5	3.6	3.7

Arch	10	30	50
DDH	30.1	71.9	96.0
UDH	10.9	33.0	51.7

[R4]: Traditional Steg vs. Deep Steg. Steganalysis is indeed a major concern for Steg (steganography), but hiding a large data capacity is also non-trivial. According to Baluja[1], traditional Steg methods, e.g. HUGO, have a small hiding capacity of <0.5 bpp, while their DDH hides a full image (24bpp). Different from traditional Steg targeting accurate bit information, deep steg[1,24,26] hides an image (technically byte information) and loosely recovers it with the goal of less distortion on C. Our work attempts understanding the success of *Deep Steg* hiding an image. We appreciate R4's suggested 7 papers related to understanding *traditional Steg* and will cite them to clearly differentiate our work.



Steganalysis on deep Steg/hiding. Like Baluja[1], we confirmed that UDH is robust to StegExpose steganalysis[2] but not to steganalysis DNNs. Baluja[1] and HiDDeN[25] showed a trade-off between capacity, secrecy (steganalysis), and/or robustness. This trade-off challenges DDH/UDH to hide a full image while deceiving steganalysis. The reviewer has the concern that deep Steg fails for steganalysis thus should not be called Steg, leading to the claim that "the core idea of the paper is incorrect". However, *Deep Steg* was widely used in prior works [1,24,25,26,29] and for consistency, we adopted the same term. However, "deep hiding" can be used to avoid confusion. Regardless of chosen terms, the success of deep Steg/hiding[1] is non-trivial, which inspires follow-up works, such as deep watermarking[29] and LFM[25], where steganalysis is **not** a major concern, instead robustness to corruptions[29] and light effect[25] are the major concerns. Despite their impressive performance, the mechanisms of deep hiding remain mostly unexplored and UDH, disentangling C and S, is the first work to provide a frequency explanation towards a better understanding.

UDH Description. Lines90-95 describe the UDH training along with mentioned lines97-109. Fig. 1 & 2 shows the core difference between UDH & DDH. More granular details are given in Sec. 2 of the supplementary along with the code. Performance-wise, UDH is comparable to DDH for Steg (arguably slightly better and worse than Baluja[1] and R4's [9], respectively).

Novelty. We proposed the first universal hiding meta-architecture UDH that (1) first explains the success of deep Steg (2) first achieves (DNN-based) universal watermarking and outperforms HiDDeN (3) first achieves hiding an image for LFM and outperforms [25] without collecting additional dataset. Overall, our proposed UDH is simple yet versatile. **Sec3.2.** Different from R4's [9], our work shows the possibility of hiding M in N images; R4's [9] did not explore different recipients getting different secret messages.

[1]We provide the link for the "perspective warp" function: https://pastebin.com/FsAC8EHu.