Abstract—Deep-learning-based watermarking technique is being extensively studied. Most existing approaches adopt a similar encoder-driven scheme which we name END (Encoder-NoiseLayer-Decoder) architecture. In this paper, we revamp the architecture and creatively design a decoder-driven watermarking network dubbed De-END which greatly outperforms the existing END-based methods. The motivation for designing De-END originated from the potential drawback we discovered in END architecture: The encoder may embed redundant features that are not necessary for decoding, limiting the performance of the whole network. We conducted a detailed analysis and found that such limitations are caused by unsatisfactory coupling between the encoder and decoder in END. De-END addresses such drawbacks by adopting a Decoder-Encoder-Noiselayer-Decoder architecture. In De-END, the host image is firstly processed by the decoder to generate a latent feature map instead of being directly fed into the encoder. This latent feature map is concatenated to the original watermark message and then processed by the encoder. This change in design is crucial as it makes the feature of encoder and decoder directly shared thus the encoder and decoder are better coupled. We conducted extensive experiments and the results show that this framework outperforms the existing state-of-the-art (SOTA) END-based deep learning watermarking both in visual quality and robustness. On the premise of the same decoder structure, the visual quality (measured by PSNR) of De-END improves by 1.6dB (45.16dB to 46.84dB), and extraction accuracy after JPEG compression (QF=50) distortion outperforms more than 4% (94.9% to 99.1%).

Index Terms—Deep-learning Watermarking, decoder-driven.

I. INTRODUCTION

Deep neural network (DNN) has powerful feature extraction capability. Such capability makes it a good replacement of traditional algorithms in many applications, with better performance or efficiency.

Watermarking technique is an important mechanism for copyright protection and leak source tracing [1], [2], [3]. A watermarking scheme has to guarantee two properties: robustness and transparency. To satisfy these properties, image features have to be first extracted and then embedded into corresponding coefficients. Traditionally, the feature extraction is achieved by handcrafted operations (e.g., discrete Fourier transformation [1], [4], discrete cosine transformation [5] and discrete wavelet transformation [6]).

Recently, many deep-learning-based watermarking schemes [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] proposed to replace the feature extractions process with DNN which effectively serves as a better embedding and extracting mechanism when training with the proper loss function and architecture.

**END architecture:** The existing DNN-based watermarking network adopts an auto-encoder-like architecture as the main backbone which contains an encoder, a noise layer and a decoder, as shown in Fig. 1(a). In this paper, we name such architecture as **END**. In **END** architecture, encoder, noise layer and decoder are cascaded serially. The encoder tries to embed the watermark into the host image and generate the watermarked image. The noise layer distorts the watermarked image. And the decoder aims to extract the watermark from the distorted image.

**Limitations of END:** Although **END** can be jointly trained, it is not the optimal architecture for watermarking system. In fact, we discovered a potential drawback in this architecture, that is: the encoder might embed some redundant features into the host image and such redundant features could limit the performance of the architecture.

We believe that such drawback comes from the imperfect coupling of encoder and decoder. Because in **END** architecture, the input of the encoder is the host image and watermark signal, which is independent to the decoder. But the input of the decoder is directly determined by the encoder, which makes **END** follows the encoder-driven rule that encoder embeds a feature and decoder extracts the encoded feature adaptively, and the features of the decoder cannot be well shared with the...
encoder. So training in this way will lead to differences in feature expression.

**De-END architecture:** To address drawbacks of END, in this paper, we propose De-END, as shown in Fig. 1(b), a decoder-driven watermarking network. Specifically, De-END adopts a **Decoder-Encoder-Noiselaye-Decoder** architecture, where the two decoder share the same parameters. In De-END, the host image is first fed into the decoder and outputs a latent feature that represents the decoder’s analysis of the host image. Then the latent feature is concatenated with the watermarked signal, and the concatenated feature will be sent to the encoder. The following noise layer and decoder are same as **END** settings. Based on this structure, the output of the decoder could determine the input of the encoder, and the input of the decoder is also determined by the output of the encoder, which ensures that the feature of encoder and decoder can be effectively shared, and encoder could be better coupled with decoder.

**Contributions:** The main contributions of the proposed scheme are summarized as follows:

1. We discover the potential drawback of the existing **END** architecture and analyze the key reason for this drawback. We hope the proposed analysis will benefit the follow-up watermarking scheme.

2. We propose **De-END**, a novel decoder-driven watermarking network architecture which could effectively couple the encoder and decoder. This architecture leads a potential way in designing high-performance watermark networks.

3. Various experiments indicate the superior performance of the proposed **De-END** architecture compared with the state-of-the-art DNN-based watermarking schemes both on visual quality and robustness.

**II. RELATED WORK**

**A. Traditional Watermarking Scheme**

Traditional watermarking scheme [17], [18] follows the encoder-driven rules. In order to meet the robustness requirement, the watermark is often embedded into the coefficients of transformed domain. And the choice of transformed domain depends on the qualitative analysis of the distortion. For example, to ensure the robustness of JPEG compression, traditional watermarking often embedded the watermark into the coefficients of discrete cosine transformation (DCT) domain [19], since JPEG compression is carried out on DCT coefficients. The most commonly used transformation includes discrete cosine transformation [19], [20], discrete wavelet transformation (DWT) [21], [22] and discrete Fourier transformation (DFT) [23], [24]. Based on the different characteristics of these transformations, the target robustness of watermarking schemes can be effectively achieved. But since traditional methods only apply handcrafted features to perform the embedding and extraction process, they cannot well balance the visual quality and the robustness.

**B. DNN-Based Watermarking Scheme**

Recently, the framework with DNN-based encoder and DNN-based decoder was proposed [11], [12], [13], [14], [25], [26], [27]. Zhu et al. [25] first proposed this framework and realize the common image processing robustness such as JPEG compression, Gaussian noise and so on. Tancik et al. [11] proposed a print-shooting noise layer to simulate the print-shooting process and further set it as the noise layer to train the whole network. As a result, the print-shooting robustness is guaranteed. Wengrowski et al. [12] proposed to use a CDTF network to simulate the screen-shooting process and regarded the well-trained CDTF network as the noise layer to guarantee the screen-shooting robustness. Liu et al. [26] proposed a two-stage method to improve the robustness, they believed the decoder can be further fine-tuned to acquire stronger robustness against non-differentiable distortions. So they fix the pre-trained encoder and further trained the decoder only to improve the robustness. Jia et al. [27] proposed a mini-batch-based JPEG noise layer to improve the JPEG robustness, by alternatively training the network with “real JPEG” and “simulated JPEG” noise, the JPEG robustness can be greatly guaranteed. Wen et al. [15] proposed an adversarial training-based method to train the watermarking network for unknown distortion. Based on the adversarial training idea, Luo et al. [16] proposed a GAN as the noise layer to generate the worst distorted image for better robustness. However, these methods all follow the END architecture which although build a connection with the encoder and the decoder, they cannot well couple the encoder and the decoder.

**III. ANALYSIS OF END ARCHITECTURE**

Fig. 1(a) shows the typical framework with END architecture, which contains an encoder, a noise layer and a decoder. The whole network is optimized by the image loss and the message loss. In END architecture, the input of the encoder is host image and watermark message, and the output of encoder is the watermarked image. For decoder, the input is the distorted image and the output of decoder is the extracted message. Since the output of the encoder will directly determine the input of the decoder, the features of encoder can be directly shared to the decoder. But the features of decoder can only be shared indirectly through message loss to the encoder. So in END, feature sharing will be conducted in an unbalanced way. As a result, there will be a potential drawback, that is, encoder might embed redundant features which are useless for decoding into the host image. And the existence of redundant features will influence the visual quality of the watermarked image and the extraction accuracy of decoder.

For better illustration, we visually show two examples of the encoded features and the decoder-needed features based on the method HiDDeN [25] which utilizes END architecture. For the features that decoder needed, we generated it with the following operations: we feed the host image into the pre-trained decoder and calculate the MSE loss of the output and watermark messages. After that, we backpropagate the loss and take the gradient map on the host image as the features that decoder needed. The corresponding results are shown in Fig. 2. $\hat{R}$ and $\nabla \hat{I}$ indicate the normalized encoded features and decoder-needed features respectively. It can be seen that the encoded features do not maintain high consistency with the decoder-needed features,
there are redundant features being embedded, which indicates that the encoder is not well coupled with the decoder in END architecture.

In order to better couple the encoder and decoder, we should revamp the architecture to make sure that the features should be directly shared in both directions of the encoder and decoder. In this way, not only can the decoder adapt to the encoded features, but also the decoder-needed features can guide the encoder. Therefore, in this paper, we propose De-END, a decoder-driven watermarking network that can effectively couple the encoder and decoder.

IV. PROPOSED ARCHITECTURE

The framework of the proposed De-END is shown in Fig. 3, which consist of four main parts: (1) The decoder $D$ with parameters $\theta_D$; (2) the encoder $E$ with parameters $\theta_E$; (3) the noise layer $N$ and (4) the discriminator $\text{Ad}$ with parameters $\theta_{\text{Ad}}$. The working flow of whole framework can be described as: The host image $I_o \in \mathbb{R}^{C \times H \times W}$ is first fed into $D$. $D$ will output a latent feature $F \in \mathbb{R}^L$ which has the same size of watermark message $M \in \{0, 1\}^L$ ($L$ indicates the length of $W$). Then $F$ and $M$ will be concatenated and further sent into $E$. The output of $E$ is a residual signal $R \in \mathbb{R}^{C \times H \times W}$ which will be weighted by a strength factor $\alpha$ and further superimposed on $I_o$ to generate the watermarked image $I_{em} \in \mathbb{R}^{C \times H \times W}$. After that, $I_{em}$ is distorted by the noise layer $N$ to generate the distorted image $I_{no} \in \{0, 1\}^L$. Among them, $\text{Ad}$ is used for optimizing the visual quality of the watermarked image and will not participate in the encoding and decoding.

A. Decoder

There are two main purposes of the decoder in this framework. The first one is guiding the encoder, which aims to generate a latent image feature $F$ with the input of $I_o$. $F$ represents the expression of $I_o$ for $D$. After that, $F$ will be concatenated with $M$ and further sent to $E$. In this way, the input of $E$ is closely connected to the output of $D$, so that the features required by $D$ can be effectively and directly shared with $E$. The second purpose is being guided by the encoder, which tries to extract the encoded feature from $I_{no}$ and realize the accurate decoding. Since $I_{no}$ is determined by the features of $E$, achieving accurate decoding is equivalent to making $D$ adapt to the features of $E$.

The structure of the decoder $D$ is shown in Fig. 3. Specifically, one “Conv-BN-ReLU” block is first processed on $I_o$ to generate $64 \times H \times W$ feature maps, then three “SE-block” [28] are applied to downsample the processed feature into a $64 \times H/8 \times W/8$ feature maps. After that, two “Conv-BN-ReLU” block, one “SE-block” and one “Linear” block are adopted to generate $F$ or $M_{de}$ with length $L$.

One object of $D$ is to minimize the difference between $M_{de}$ and the original watermark $M$ by updating $\theta_D$, which can be formulated by:

$$\mathcal{L}_D = \text{MSE}(M, M_{de}) = \text{MSE}(M, D(\theta_D, I_{no}))$$  \hspace{1cm} (1)

B. Encoder

Different from the traditional END architecture, the encoder in De-END only realizing the up-sampling process. It only takes the input of concatenated feature which is partially determined by $D$ and then outputs a residual $R$ which is further superimposed on $I_o$ to generate the watermarked image $I_{em}$. The structure of $E$ is shown in Fig. 3. Specifically, one “Linear” block is first adopted to resize the concatenated feature into a feature with length $H/8 \times W/8$. Then the feature is reshaped and processed by a “Conv-BN-ReLU” block. After that, three “Up-Conv-BN-ReLU” blocks are taken to up-sample the feature into the size of $64 \times H \times W$. Finally, one single “Conv” block is applied to generate the residual $R$ with size $C \times H \times W$. It is worth noting that a strength factor $\alpha$ is adopted in generating the final watermarked image $I_{em}$. And $\alpha$ is used for balancing the visual quality and the robustness.

The object of $E$ is to generate a better residual $R$ to minimize the difference between $I_{em}$ and the host image $I_o$ by updating $\theta_E$, which can be formulated by:

$$\mathcal{L}_E = \text{MSE}(I_o, I_{em}) = \text{MSE}(I_o, E(\theta_E, F, M) \times \alpha + I_o)$$  \hspace{1cm} (2)

C. Noise Layer

The noise layer $N$ is the key to realizing robustness. By setting different differentiable image processing operations in $N$, the watermarked image $I_{em}$ will be distorted into different versions. And the distorted image $I_{no}$ will be further decoded by the decoder. So the distortion adopted in training will determine the final robustness. Commonly used distortion include “JPEG Compression,” “Gaussian Noising,” “Median Filtering” and so on.

D. Discriminator

One essential purpose of the framework is generating a watermarked image with a given host image and a watermark. Such process is similar to Conditional-GAN [29], which aims to generate an image with a given condition. So in order to generate the
high-quality watermarked image, we adopt the structure of discriminator. The structure of discriminator Ad is same as [25] proposed. Specifically, it consists of four “Conv-BN-ReLU” blocks and an “AveragePooling” block. The discriminator performs as an adversary of the generating process and tries to give a correct identification between \( I_{em} \) and \( I_o \), which is realized by updating \( \theta_{Ad} \):

\[
L_{Ad} = \log (1 - Ad(\theta_{Ad}, I_{em})) = \log (1 - Ad(\theta_{Ad}, E(F, M) \times \alpha + I_o))
\]  

(3)

On the other hand, \( \theta_E \) should be also updated to mislead the discriminator, which can be realized by minimizing:

\[
L_{Ad} = \log(Ad(I_{em})) = \log(Ad(E(\theta_E, F, M) \times \alpha + I_o))
\]  

(4)

E. Loss Function

The final loss of the whole network is consist of image loss, adversarial loss and decoding loss, which can be formulated by:

\[
\mathcal{L} = \lambda_1 \mathcal{L}_E + \lambda_2 \mathcal{L}_D + \lambda_3 \mathcal{L}_{Ad}
\]  

(5)

where \( \lambda_1, \lambda_2, \lambda_3 \) are set as 1,10,0.0001 in the first 20 epoch and 10,1,0.0001 in the rest epoch.

V. EXPERIMENTAL RESULTS

In this section, we will first introduce the implementation details and the parameter selection of the proposed scheme. Then extensive experiments will be conducted to verify the robustness and visual quality of our method. Finally, more analysis of the proposed architecture will be provided to justify our design.

A. Implementation Details

The whole network is trained on COCO dataset [30], and we randomly choose 10000 images as the training dataset. When loading the image, data enhancement operations such as random padding and random cropping are applied to enrich diversity of training data. The framework is implemented by PyTorch [31] and performed on NVIDIA RTX 2080ti. The size of the image \( H \) and \( W \) are both set as 128. The length \( L \) is set as 64. Strength factor \( \alpha \) is fixed as 1 in training. For neural network parameter optimization, we utilize Adam [32] with default hyperparameters. In testing experiments, we utilize the classical USC-SIPI image dataset [33] as our testing data.

For visual quality measurement, we use PSNR as the assessment metric. For robustness evaluation, we directly use the extraction accuracy of the watermark message as the assessment metric. We compare the proposed framework with three deep-learning-based framework HiDDeN [25], TSDL [26], MBRS [27] and DADW [16]. The distortion we selected include “Cropout,” “Dropout,” “Gaussian Noise,” “Salt&Pepper Noise,” “Gaussian Blur,” “Median Blur” and “JPEG Compression”. For each distortion, we train a specific network to better show the effectiveness of the architecture and the robustness of the framework. All the compared methods are all training with specified noise layers. Since the different noise layer will result in different encoder and decoder, the visual quality of the watermarked image and the extraction accuracy varies. So in this paper, we show the PSNR, SSIM and LPIPS [34] value of the embedded image and the extraction accuracy for each noise layer respectively.

B. Coupling Measurement

As aforementioned, END architecture has a potential drawback that the encoder and decoder may not be well coupled. So we propose De-END to effectively handle this problem. In this section, we will give two examples to visually compare the coupling of encoder and decoder with different schemes, as shown in Fig. 4.

The first column is the original host images, the second column represents the watermarked images, the third column indicates the normalized encoded features and the fourth column shows the normalized features that the decoder needed. The noise layer we choose is JPEG compression. As we can see in Fig. 4, for the traditional END architecture, the embedded residual features are not highly consistent with the features required by the decoder. The encoder embeds more features than similar images.
the decoder needed, which may reflect some image textures. This means that the encoder and decoder obtained by the END framework cannot be effectively coupled together, resulting in certain feature expression differences. However, for the proposed De-END architecture, the embedded features keep high consistency with the features decoder needed, which means that the encoder and decoder trained by this architecture can be better coupled.

C. Visual Quality and Robustness

In this section, we will show and discuss the visual quality and robustness of different methods for 7 types of distortions: “Cropout,” “Dropout,” “Gaussian Noise,” “Salt&Pepper Noise,” “Gaussian Blur,” “Median Blur” and “JPEG Compression”. We visually show each distortion and the corresponding encoded features of the proposed architecture in Fig. 6. The first row is the embedded images, the second row indicates the distorted images. The third row and fourth row represents the encoded features and the features that decoder needed respectively. We can see that for all the distortions, the encoder and decoder are coupling well which results in high consistency of the feature map. Besides, the encoded residual adaptively changed with the distortion, which indicates the great learning ability of the proposed architecture.

For the visual quality, the PSNR, SSIM and LPIPS value of the proposed method and the compared methods with different distortions are shown in Fig. 5.

1) **Cropout Distortion:** Cropout refers to the distortion that crops a block of image with a certain ratio out and replaces the cropped region with original images. In training stage, we fixed the parameter of cropout ratio as 40%, and we test the cropped ratio from 10% to 40%. The experimental results are shown in Fig. 5 and Table I.
As seen in Fig. 5, the proposed method will lead to the watermarked images with the highest PSNR and SSIM values and lowest LPIPS values. Under such visual quality, the proposed scheme still maintains the strongest robustness in different crop-out ratios. For the crop ratio of 10% to 40%, the extraction accuracy of the proposed scheme is all higher than 97%. As the crop-out ratio increasing, the advantage of the proposed algorithm is more and more obvious.

2) Dropout Distortion: Dropout distortion indicates the operation that randomly replaces a certain ratio of image pixels with the original image. But different from cropout, such image pixels are randomly sampled in the whole image. We fixed the ratio of 40%. And for testing, we change the ratio from 20% to 60%. The experimental results are shown in Fig. 5 and Table II.

We can see in Fig. 5, the proposed framework maintains the best visual quality. Besides, Table II indicates the superior performance of the proposed scheme compared with other frameworks. For all the dropout ratios, the proposed framework guarantees the best extraction accuracy, which is at least 2% higher than the other schemes.

3) Gaussian Noise: Gaussian noise is commonly appeared in message transmission. For training stage, we randomly distort the image with a variance from 0.001 to 0.04. The testing variance ranges from 0.01 to 0.05. The results are shown in Table III.

As seen in Fig. 5, the proposed scheme maintains the highest PSNR and SSIM value which is more than 40 dB and 0.99, respectively. From Table III we can see that, in all the testing variance, the proposed scheme maintains the best robustness against Gaussian noise distortion. The accuracy is significantly
higher than HiDeN, TSDL and DADW, but for MBRS, the advantage appears in the large variance.

4) Salt&Pepper Noise: Similar to Gaussian noise, Salt&Pepper Noise is commonly appeared in transmission too, which randomly sampled a certain ratio of image pixels into a noise. The training ratio is randomly selected from 0.001 to 0.04. And the testing ratio ranges from 0.01 to 0.05. The final results are shown Table IV.

We can see that the PSNR and SSIM of the proposed scheme is higher than other compared schemes. For robustness, the proposed framework still ensures the best performance. The extraction accuracy of the proposed scheme is bigger than 98% in all the test distortions, which indicates the strong robustness against Salt&Pepper noise.

5) Gaussian Blur: For Gaussian blur distortion, we fixed the noise layer with variance of 2, and conduct the Gaussian blurring operation with variance from 0 to 2 in testing to show the robustness. The visual quality comparison and extraction accuracy are shown in Fig. 5 and Table V.

As seen in Fig. 5 and Table V, we ensure the best visual quality compared with other schemes. Besides, we maintain the highest extraction accuracy under the best visual quality. For all the testing median blur windows, the proposed scheme achieves more than 98% accuracy, which indicates superior robustness against median blur distortion.

D. Complexity Comparison

The proposed De-END architecture which cascades a decoder in front of the encoder is not similar as the original END architecture. In order to compare the number of parameters and the embedding and extraction times between different architectures, we have done the following experiments. First, we record the parameter numbers used for encoding, decoding and overall process with different algorithms to illustrate the complexity of different frameworks. After that we also tested the running time of embedding and extraction processes on one image for each algorithm in the same devices, as shown in Table VIII.

We could see in Table VIII that the parameter numbers of the proposed scheme is larger than that of HiDeN [25], TSDL [26] and DADW [16], but less than that of MBRS [27]. The reason is that the proposed method and MBRS utilized the architecture of “SE-block” which contains an attention module in each block. So compared with the simple “Conv-BN-ReLU” architecture which is utilized in HiDeN and TSDL, the parameter numbers

| TABLE IV |
| Extraction Accuracy With Different Ratio of Salt&Pepper Noise |
| Ratio | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
| HiDeN [25] | 95.12% | 93.79% | 94.35% | 92.92% | 90.42% |
| TSDL [26] | 97.29% | 95.63% | 93.54% | 92.71% | 91.46% |
| MBRS [27] | 98.05% | 98.74% | 98.34% | 97.56% | 96.68% |
| DADW [16] | 90.42% | 86.67% | 80.42% | 75.42% | 71.42% |
| Proposed | 99.41% | 99.51% | 99.22% | 99.12% | 98.73% |

| TABLE V |
| Extraction Accuracy With Different Variance of Gaussian Blur |
| Variance | 0.0001 | 0.5 | 1 | 2 |
| HiDeN [25] | 95.44% | 95.21% | 94.33% | 84.37% |
| TSDL [26] | 99.92% | 99.79% | 98.48% | 93.21% |
| MBRS [27] | 98.64% | 98.25% | 97.66% | 87.80% |
| DADW [16] | 91.70% | 89.80% | 76.25% | 66.25% |
| Proposed | 100% | 100% | 99.51% | 94.34% |

7) JPEG Compression: JPEG compression always appears in image saving and format conversion. In training stage, we choose the noise layer with QF=50 which is proposed by MBRS [27] as the default parameter. And we test the JPEG compression attack with QF from 40 to 90 to show the robustness. The experimental results are shown in Table VII.

As seen in Fig. 5, the proposed scheme ensures high quality watermarked images with PSNR value that is more than 46 dB and SSIM value that is higher than 0.99. For robustness, the proposed framework is significantly higher than the compared schemes, as shown in Table VII. Especially for QF=50, the proposed scheme reaches more than 99% accuracy, which implies that the algorithm can be effectively used in practice since in most cases, the quality factors will not be less than 50 in practical use.
TABLE VIII

| Complexity | Encoding Parameters | Decoding Parameters | Total Parameters | Times (s) |
|------------|---------------------|--------------------|-----------------|-----------|
| HiDDeN [25] | 188931 | 265472 | 454403 | 0.0014 | 0.0020 |
| TSDL [26] | 370051 | 300864 | 670915 | 0.0041 | 0.0064 |
| MBRS [27] | 561292 | 20244099 | 2080391 | 0.0107 | 0.0129 |
| DADW [16] | 188931 | 265472 | 454403 | 0.0014 | 0.0020 |
| Proposed | 5882118 | 5255235 | 5882118 | 0.0128 | 0.0099 |

will be larger. We also can see that for the compared method, the parameter numbers of the whole framework equal to the sum of encoder parameters and decoder parameters. But for the proposed method, the parameter numbers of the whole framework are same as the encoder parameters. This is because in De-END, the two decoders share the same parameters, so the parameter used for encoding contains the parameter used for decoding.

In the view of running time, the proposed algorithm does not have advantages in terms of embedding and extraction time. This is due to two reasons: 1) we use “SE-block” as the base unit, which has a more complex model structure and longer running time; 2) there are few “full-connection” layers in our backbone, which is more time consuming compared to single “Conv” layers.

Although the complexity of our proposed network is higher, our algorithm significantly outperforms the comparison scheme in terms of robustness and visual quality, which results in a trade-off of “performance and complexity”. Despite this, the embedding and extraction time required in this paper is very short, taking only 0.01 s to embedding or extracting, which means that the proposed scheme can ensure message embedding and extraction efficiently in real applications.

E. More Analysis

In this section, we will conduct more analysis of the proposed De-END architecture to verify the design.

1) Architecture Improvements: The key contribution of this paper is designing the De-END architecture, which achieves better robustness and visual quality compared with other schemes. The most important thing we should confirm is whether the performance improvement comes from the architecture or the encoder/decoder backbone. In order to verify the effectiveness of De-END architecture, we conduct the following experiments. We choose the same decoder as well as the noise layer proposed by HiDDeN, TSDL and MBRS, and set them as the decoder in De-END architecture. As for encoder, we use the proposed encoder structure. Then we train the network with JPEG compression distortion. The experimental results are shown in Table IX and Fig. 7.

As shown in Table IX, the proposed De-END architecture has significantly improved the visual quality of methods HiDDeN [25] and TSDL [26]. For MBRS [27], it achieves a little improvement. But for robustness, it can be seen that applying De-END architecture will greatly improve the extraction accuracy. As seen in Fig. 7, the blue line indicates the extraction accuracy with the original END architecture, the orange line represents the extraction accuracy with De-END architecture. For all the methods, simply utilizing De-END architecture without any backbone changing will bring performance improvement. This indicates that the improvements mainly come from the architecture instead of the backbone design.

2) Architecture Comparison: As aforementioned, in De-END, two decoders share the same parameters. And we believe sharing parameter is the key to realizing feature coupling of encoder and decoder. In this section, we will test whether it is necessary to share the same parameter. We change the De-END architecture from “Decoder-Encoder-NoiseLayer-Decoder” (denoted as De-END) to “Decoder A-Encoder-NoiseLayer-Decoder_B” (denoted as De_A END_B). Decoder_A has the same structure as Decoder_B, but we do not force them to share parameters. If De_A END_B can achieve similar performance as De-END, it means that sharing the same parameters will not be helpful in feature coupling, only applying a “Decoder-Encoder” like structure in embedding stage is enough to improve the performance. In addition to “Decoder_A-Encoder-NoiseLayer-Decoder_B,” we also test the structure of END where the encoder only adopts watermark as input, denote as “E_W ND”. If “E_W ND” performs better, it means setting decoder before encoder is useless, the main improvements just rely on the input changing of the encoder. So to verify our design, we train De-END, De_A END_B and E_W ND respectively. After training, the PSNR values of De_A END_B, E_W ND and De-END are 46.40 dB, 46.88 dB and 46.84 dB respectively. The extraction accuracy is shown in Table X.

It can be seen that under the same level of PSNR, De-END maintains the best extraction accuracy compared with De_A END_B and E_W ND, which means sharing the parameters with two decoders is a good way to couple encoder and decoder. Besides, setting the decoder before encoder is also necessary for getting better performance. It is worth noting that maybe training De_A END_B in some way may realize a similar performance.

Fig. 7. The robustness comparison with different architecture.
as De-END, but we believe sharing parameters in these two decoders is a fast and effective way to realize good performance.

3) Backbone of Decoder: As illustrated in Section V-E1, different structures of decoder will result in similar performance. Since the most commonly used union in HiDDeN [25], TSDL [26] and MBRS [27] is “Conv-BN-ReLU,” in this section, we will explore the relationship between the number of “Conv-BN-ReLU” union and the performance. We train the structure of the decoder with 3 to 7 “Conv-BN-ReLU” union and test the performance. The corresponding results are shown in Table XI and Table XII.

| Number | 3   | 4   | 5   | 6   | 7   |
|--------|-----|-----|-----|-----|-----|
| PSNR(dB) | 45.35 | 45.94 | 45.58 | 45.05 | 45.62 |

It can be seen from Table XI that the PSNR will not significantly change as the number of “Conv-BN-ReLU” union increases, and will maintain stability at a high level of 45 dB. But from Table XII we can conclude that the extraction accuracy grows as the number of “Conv-BN-ReLU” union increases from 3 to 5. But for 5 to 7 unions, the performance becomes similar. It indicates that deepening the network will not necessarily produce better performance. Applying 5 to 7 “Conv-BN-ReLU” union is enough to ensure great performance.

4) Backbone of Encoder: In the proposed De-END architecture, the encoder adopts a very simple structure that contains one three “Up-Conv-BN-ReLU” blocks. Such structure has already shown its powerful performance, and deepening the network may not result in better performance. So in this section, we will mainly investigate the influence of the up-sampling ways. We choose three commonly used upsampling ways: “Un-pooling,” “Transpose-conv” and “Nearest-interpolating” to test the performance. Specifically, we only change the up-sampling ways in “Up-Conv-BN-ReLU” union and further train each network respectively. For “Un-pooling,” since it is necessary to give the max-pooling location for each feature map in order to realize “Un-pooling” operation, we randomly sample the location and fixed it in training and testing. The final PSNR with “Un-pooling,” “Transpose-conv” and “Nearest-interpolating” backbone are 46.53 dB, 46.48 dB and 46.84 dB respectively. The extraction accuracy are shown in Table XIII.

 distintive PSNR values. As for robustness, “Transpose-conv” and “Nearest-interpolating” maintain similar high-level extraction accuracy, but for “Un-pooling,” the performance will be worse than the other two rules. We conclude the reason that when we conduct “Un-pooling” operation, we fixed the max-pooling location for each feature map, and the randomly sampled location may not be optimal, which will result in bad performance. In general, applying “Transpose-conv” and “Nearest-interpolating” all can ensure great performance.

5) Impact on Down-Sampling and Up-Sampling Operation: In previous DNN-based watermarking framework, each hidden layer of the encoder will maintain the same size as the original image, that is, the encoder does not contain the down-sampling and up-sampling operation. However, in this paper, since we want to ensure a tight couple of the encoder and the decoder, we cascade the decoder before the encoder. The decoder will down-sample the image to the size of the message. In order to keep the consistence of the image size, the image will be first down-sampled and then up-sampled. These “up down sampling” layers is not similar to the previous framework. Therefore, in order to illustrating the influence of such layers, we conduct the following experiments.

We construct two architectures with and without up-down-sampling. Each architecture follows the proposed De-END (“Decoder-Encoder-Noise layer-Decoder”) rules. For the architecture with up-down-sampling (denoted as “\(w \text{ ud}\)”), the decoder down-sampled three times, and the encoder up-sampled three times. For the architecture without up-down-sampling (denoted as “\(w/o \text{ ud}\)”), the encoder and the decoder will not change the size of the image. The size of the “Linear” layer contained in the decoder is adjusted according to the size of the encoder’s output. The basic backbone of decoder we choose is same as HiDDeN [25]. Then we train these two architectures with the noise layer of “zoom” and JPEG compression and test the corresponding robustness, the results are shown in Table XIV.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE XIV
PERFORMANCE WITH/WITHOUT SAMPLING ARCHITECTURE ON ZOOM ATTACK AND JPEG COMPRESSION

| Zoom   | PSNR       | 0.4 | 0.5 | 0.75 | 1.1 | 1.5 | 2 |
|--------|------------|-----|-----|------|-----|-----|---|
| w/o ud | 51.88 dB   | 82.81% | 98.34% | 100% | 100% | 100% | 100% |
| w ud   | 51.78 dB   | 87.79% | 99.51% | 100% | 100% | 100% | 100% |

TABLE XV
PERFORMANCE OF UNTRAINED DISTORTIONS

| Distortion | Hue | GIF |
|------------|-----|-----|
| Intensity  | -15 | 15  | QF=75 | QF=90 |
| Accuracy   | 100% | 100% | 93.75% | 100% |

![Original Image](image1.png)
![Hue (Red)](image2.png)
![Hue (Green)](image3.png)
![Hue (Blue)](image4.png)

Fig. 8. The tested untrained distortion.

VI. CONCLUSION

The existing DNN-based watermarking algorithms mainly adopt an END backbone which contains an encoder, a noise layer and a decoder. In this paper, we designed a novel decoder-driven DNN-based watermarking network dubbed De-END. The motivation comes from the potential drawbacks of the existing END framework we discovered, that is, encoder may embed redundant features which is not necessary for decoding into the image. And we deeply investigate the reason as the encoder and decoder cannot be well coupled under END framework. To the best of our knowledge, we are the first to give the analysis of such drawback and we hope it will benefit the follow-up work. As for mechanism, in order to address such problem, we propose the architecture of De-END, which cascades the decoder before the encoder. Based on this framework, the features of encoder and decoder can be better shared, so encoder and decoder can be better coupled. Various experiments show that the visual quality and watermark robustness of the proposed architecture is significantly better than the existing state-of-the-art algorithms, which greatly proves the superior performance of the De-END architecture.

ACKNOWLEDGMENT

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] T. K. Tsui, X.-P. Zhang, and D. Androutsos, “Color image watermarking using multidimensional fourier transforms,” IEEE Trans. Inf. Forensics Secur., vol. 3, no. 1, pp. 16–28, Mar. 2008.

[2] Y. Huang, B. Niu, H. Guan, and S. Zhang, “Enhancing image watermarking with adaptive embedding parameter and PSNR guarantee,” IEEE Trans. Multimedia, vol. 21, no. 10, pp. 2447–2460, Oct. 2019.

[3] Z. Chen, L. Li, H. Peng, Y. Liu, and Y. Yang, “A novel digital watermarking based on general non-negative matrix factorization,” IEEE Trans. Multimedia, vol. 20, no. 8, pp. 1973–1986, Aug. 2018.

[4] M. Hamidi, M. El Haziti, H. Cherifi, and D. Aboutajdine, “A blind robust image watermarking approach exploiting the DFT magnitude,” in Proc. IEEE/IACS 12th Int. Conf. Comput. Syst. Appl., 2015, pp. 1–6.

[5] H.-J. Ko, C.-T. Huang, G. Horng, and W. Shih-Jeng, “Robust and blind image watermarking in DCT domain using inter-block coefficient correlation,” Inf. Sci., vol. 517, pp. 128–147, 2020.

[6] H. Daren, L. Jiufen, H. Jiwu, and L. Hongmei, “A DWT-based image watermarking algorithm,” in Proc. IEEE Int. Conf. Multimedia Expo., 2001, pp. 313–316.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
[7] S.-M. Mun et al.,” “A robust blind watermarking using convolutional neural network,” 2017, arXiv:1704.03248.

[8] S.-M. Mun, S.-H. Ham, H. Jang, D. Kim, and H.-K. Lee, “Finding robust domain from attacks: A learning framework for blind watermarking,” Neurocomputing, vol. 337, pp. 191–202, 2019.

[9] M. Ahmadi, A. Norouzi, N. Karimi, S. Samavi, and A. Emami, “Redmark: Framework for residual diffusion watermarking based on deep networks,” Expert Syst. with Appl., vol. 146, 2020, Art. no. 113157.

[10] H. Kandi, D. Mishra, and S. R. S. Gorithi, “Exploring the learning capabilities of convolutional neural networks for robust image watermarking,” Comput. Secur., vol. 65, pp. 247–268, 2017.

[11] M. Tancik, B. Mildenhall, and R. Ng. “StegaStamp: Invisible hyperlinks in physical photographs,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2117–2126.

[12] E. Wengrowski and K. Dana, “Light field messaging with deep photostatic steganography,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1515–1524.

[13] C. Zhang, P. Benz, A. Karjuav, G. Sun, and I. S. Kweon, “UDHE: Universal deep hiding for steganography, watermarking, and light field messaging,” Adv. Neural Inf. Process. Syst., vol. 33, pp. 10223–10234, 2020.

[14] P. Zhang, C. Li, and C. Wang, “Viscode: Embedding information in visualization images using encoder-decoder network,” IEEE Trans. Vis. Comput. Graph., vol. 27, no. 2, pp. 326–336, Feb. 2021.

[15] B. Wei and S. Ajdore, “Romark: A robust watermarking system using adversarial training,” 2019, arXiv:1910.01221.

[16] X. Luo, R. Zhan, H. Chang, F. Yang, and P. Milanfar, “Distortion agnostic deep watermarking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 13548–13557.

[17] H. Fang et al., “Deep template-based watermarking,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 4, pp. 1436–1451, Apr. 2021.

[18] S. Mellimi, V. Rajput, I. A. Ansari, and C. W. Ahn, “A fast and efficient image watermarking scheme based on Deep Neural Network,” Pattern Recognit. Lett., vol. 151, pp. 222–228, 2021.

[19] H. Fang, W. Zhang, H. Zhou, H. Cui, and N. Yu, “Screen-shooting resilient watermarking,” IEEE Trans. Inf. Forensics Secur., vol. 14, no. 6, pp. 1403–1418, Jun. 2019.

[20] S.-M. Mun, S.-H. Nam, H. Jang, D. Kim, and H.-K. Lee, “Finding robust domain from attacks: A learning framework for blind watermarking,” Neurocomputing, vol. 337, pp. 191–202, 2019.

[21] J. Hu, L. Shen, and G. Sun, “Reasonable effectiveness of deep features as a perceptual metric,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 1509–1517.

[22] Z. Jia, H. Fang, and W. Zhang, “MBRS: Enhancing robustness of DNN-based watermarking by mini-batch of real and simulated JPEG compression,” in Proc. 29th ACM Int. Conf. Multimedia, 2021, pp. 41–49.

[23] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7132–7141.

[24] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014, arXiv:1411.1784.

[25] H.-T. Hu and T.-T. Lee, “Frame-synchronized blind speech watermarking via improved adaptive mean modulation and perceptual-based additive modulation in DWT domain,” Digit. Signal Process., vol. 87, pp. 75–85, 2019.

[26] Y. Gao, J. Wang, and Y.-Q. Shi, “Dynamic multi-watermarking and detecting in DWT domain,” J. Real-Time Image Process., vol. 16, no. 3, pp. 565–576, 2019.

[27] X. Kang, R. Yang, and J. Huang, “Geometric invariant audio watermarking based on an LCM feature,” IEEE Trans. Multimed., vol. 13, no. 2, pp. 181–190, Apr. 2011.

[28] X. Kang, J. Huang, and W. Zeng, “Efficient general print-scanning resilient data hiding based on uniform log-polar mapping,” IEEE Trans. Inf. Forensics Secur., vol. 5, no. 1, pp. 1–12, Mar. 2010.

[29] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “Hidden: Hiding data with deep networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 657–673.

[30] J. Jiyi Zhang, and A. Karjuav, G. Sun, and I. S. Kweon, “A fast and efficient image watermarking scheme based on Deep Neural Network,” 2020, pp. 13548–13557.

[31] Z. Jia, H. Fang, and W. Zhang, “MBRS: Enhancing robustness of DNN-based watermarking by mini-batch of real and simulated JPEG compression,” in Proc. 29th ACM Int. Conf. Multimedia, 2021, pp. 41–49.