TransforMark - Watermarking with Vision Transformers

Oree Leibowitz Roey Magen

Abstract

Digital watermarking is the process of embedding information into an image (or other types of multimedia) such that it can survive under distortions, while requiring the encoded image to have little or no perceptual difference from the original image. Recent works has shown that neural networks can learn to use invisible perturbations to encode a rich amount of useful information. In fact, one can exploit this capability for the task of embedding data into an image. State-of-the-art works are based on Convolutional Neural Networks (CNN). We examined new encoder-decoder architectures which use also Vision Transformers (ViT). ViT refers to a transformer architecture which is applied directly on a sequence of image patches. ViTs have been shown to perform very well on image classification and detection tasks. We show that ViTs can assist for the watermarking task as well.

1 Problem Statement and Motivation

A *Digital Watermark* is a kind of marker, covertly embedded inside a digital media (e.g. audio, video or an image) which enables to know the source or owner of the copyright. Digital watermarking is being used for identifying image ownership. Sample applications of it are tracing copyright infringement in social media and knowing the genuineness of the notes in the banking system. This work is focused on digital watermarking of images.

We can model the problem using "cryptographic" terms as three parties: Alice, Bob and an eavesdropper Eve. Alice encodes a fingerprint in an image. Then Eve distorts the image (for example by cropping, blurring, etc). Eventually, Bob should be able to detect the fingerprint in the distorted image.

One motivation to consider neural networks for this task is the phenomena of adversarial examples (can be shown as a "bright side of adversarial examples"). While the existence of adversarial examples is usually seen as a disadvantage of neural networks, it can be desirable for watermarking: if a network can be fooled with small perturbations into making incorrect class predictions, it should be possible to extract meaningful information from similar perturbations. Furthermore, the adversarial nature of these generated examples is preserved under a variety of image transformations (See [1]).

Nowadays we know that self-attention mechanisms can draw global dependencies between pixels in the image. Thus the motivation for our work is that finding global dependencies between pixels in the image can help to generate a better target image without perturbation of the spotlight and improve the robustness of the data to distortions (by finding a global relation between pixels that enhances robustness for distortions).

Moreover, state-of-the-art works for image watermarking made by deep learning are based on CNN and adversarial training. To the best of our knowledge, this is the first work that explores a usage of ViT for deep watermarking.

2 Related work

A wide variety of watermarking settings and methods have been proposed in the literature. Some watermarking methods encode information in the least significant bits of image pixels (See for example [3]). Other, more robust, methods encode information in the frequency domain (See for example [4]).

Steganography. *Steganography* is another common setting for hiding information in images. In steganography, the goal is secret communication: a sender encodes a message into an image such that the recipient can decode the message, but an adversary cannot tell whether any given image contains a message or not. In contrast to watermarking, steganography prefers secrecy over robustness. Most of the done works try to tackle both the settings using the same architecture (by adapting some constants in the loss that represent the relative weights of secrecy and robustness). In this work we chose to focus on Watermarking.

Neural networks have also been used for watermarking. Old works used neural networks for one stage of a larger pipeline, such as determining watermarking strength per image region, or as a part of the encoder or the decoder. Later works modeled the entire data hiding pipeline with neural networks and train them end-to-end. The networks are trained such that: 1) Given an input message and a cover image, the encoder produces a visually indistinguishable encoded image. 2) The encoded image go over noise layers and the decoder succeeds to decode the noisy image to recover the original message (with small error rate).

Zhu at el. ([2]) proposed HiDDeN, a unified framework for digital watermarking and image steganography that uses a network based on convolutional layers. The network is trained to be robust against noise using different noise layers, such as: Gaussian blurring, pixel-wise dropout, cropping, and JPEG. It is important to note that HiDDeN is a very influential paper in the field of data hiding, and most of the papers that were published in the last two years compare themselves to this work. A later work influenced by HiDDeN has been done by Xiyang at el. ([5]). Their work uses adversarial training to generate learnable noise layers in order to generate a watermarking mechanism that is more distortion agnostic. Yu ([7]) proposed to incorporate to the CNN input an attention mask of the cover image, in addition to the cover image itself and the message. The used attention model is the feature extraction backbone of ResNet50. This work inspired us to use the attention of a ViT.

3 Method

Our goal is to develop a learnable, end-to-end model for image watermarking that can be made robust to arbitrary types of image distortion. Our model is based on HiDDeN ([2]) model. To this end, our system comprises three main components: an encoder E_θ, a parameter-less noise layer N, and a decoder D_γ. θ and γ are trainable parameters. The encoder E_θ receives a cover image I_{co} of shape $C \times H \times W$ and a binary secret message $M_{in} \in \{0,1\}^L$ of length L and produces an encoded image I_{en} of the same shape as I_{co}. The noise layer N receives I_{co} and I_{en} as inputs and distorts I_{en} to produce a noised image I_{no}. The decoder D recovers a message M_{out} out of I_{no}.

The encoded image should be visually similar to the cover image. We characterize the "similarity" with an image distortion loss - the ℓ_2 distance between I_{co} and I_{en}:

$$L_I(I_{co}, I_{en}) := ||I_{co} - I_{en}||_2^2/(CHW)$$

The decoded message should be the same as the encoded message. We impose a message distortion loss using the ℓ_2 distance between the original and the decoded messages:

$$L_M(M_{in} - M_{out}) := ||M_{in} - M_{out}||_2^2/L$$

We perform batch stochastic gradient descent on θ, γ to minimize the following loss over the distribution of input messages and images:

$$\mathbb{E}_{I_{co}, M_{in}}[\eta_M L_M(M_{in}, M_{out}) + \eta_I L_I(I_{co}, I_{en})]$$

where η_M and η_I control the relative weights of the losses.

3.1 Network Architecture

A diagram for the baseline model is shown in Figure 1. The encoder first applies a network based transformers to input I_{co} in order to create some intermediate representation. In our first approach (see Section 3.2) the transformer mechanism is based on a learnable ViT (the training process trains the ViT). In the second and third attempts the transformer mechanism is based on a pretrained DINO model (see Sections 3.3 and 3.4 for additional details). Next, we aim to incorporate the message input (of length L) in such a way that the encoder can easily learn to embed parts of it at any spatial location of the output. For this reason, we replicate the message spatially, and concatenate this "message volume" to the encoder's intermediary representation. This ensures that each convolutional filter in the next layer has access to the entire message as it convolves across each spatial location. After more convolutional layers, the encoder produces I_{en} - the encoded image. From this point on, our architecture is the same as HiDDeN except of the adversarial training. The noise layer, given I_{co} and I_{en}, applies a noise to produce I_{no}. We do not require I_{no} to have the same dimension as I_{en}, so that we can model dimension-altering noise like cropping. The decoder first applies several convolutional layers to produce L feature channels in the intermediate representation. Then it applies a global spatial average pooling to produce a vector L of the same size as the message. The average pooling ensures that it can handle inputs of different spatial dimensions. Finally, it applies a single linear layer to produce the predicted message.

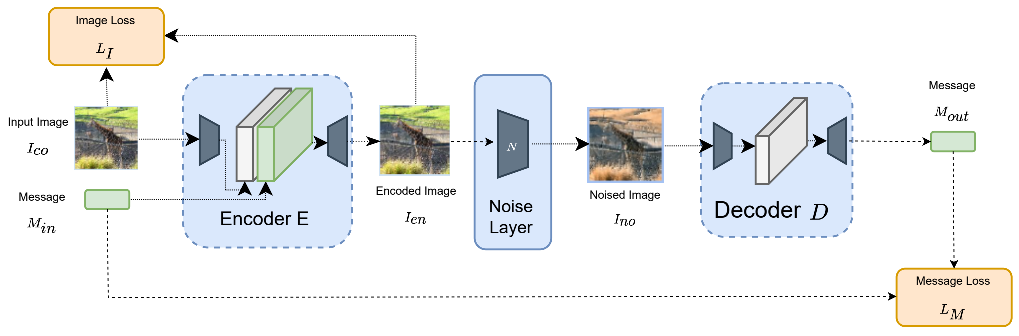

Figure 1: Our baseline model, influenced by HiDDeN

Noise Layer. Our model should be able to hide messages robustly against a variety of image distortions; we consider 4 different types of noise layers which apply different types of image distortion. The **Dropout** and **Cropout** layers undo some of the changes made by the encoder, producing the noisy image I_{no} by combining pixels from the cover image I_{co} and the encoded image I_{en}. Both types of noise keep a percentage p of pixels from I_{en} and use the rest from I_{co}, where **Dropout** makes this choice independently per pixel while **Cropout** keeps a random square crop from I_{en}. The **Crop** layer produces a random square $H' \times W'$ crop of the encoded image. The **Resize** layer produces a resized image of the encoded image of size $H' \times W'$.

Our approach to combine a transformer mechanism is reflected in the encoder architecture. In all the attempts, the goal is to take the input image I_{co} and extract some intermediate representation with shape $C' \times H \times W$. After that we apply the same architecture described in section 3.1.

Next we present three different attempts to tackle this challenge.

3.2 First Attempt: ViT Based Encoder

Our system setup is shown in Figure 2. In order to create some intermediate representation, the encoder first applies a ViT block to the input I_{co}. Then, the [CLS] token output is connected to a linear layer and the output is reshaped to the size $C' \times H \times W$.

Technical details. The ViT is composed of 3 layers, namely 3 transformer blocks. The number of heads in each multi-head attention layer is 16. The first convolutional layers transfer the input channel to 32 channels (in comprasion to 64 channels in HiDDeN), the intermediate layers preserve the number of channels and the last convolutional layer transfers to 3 channels (in order to generate an image). We used Adam optimizer with learning rate 0.001.

A drawback of ViTs is that they structurally lack locality inductive bias (in contrast to CNNs). Therefore they require a very large amount of training data (see [9]). COCO is a relatively small data-set and we trained our model on only 10,000 images. All of this leads us to our next approach.

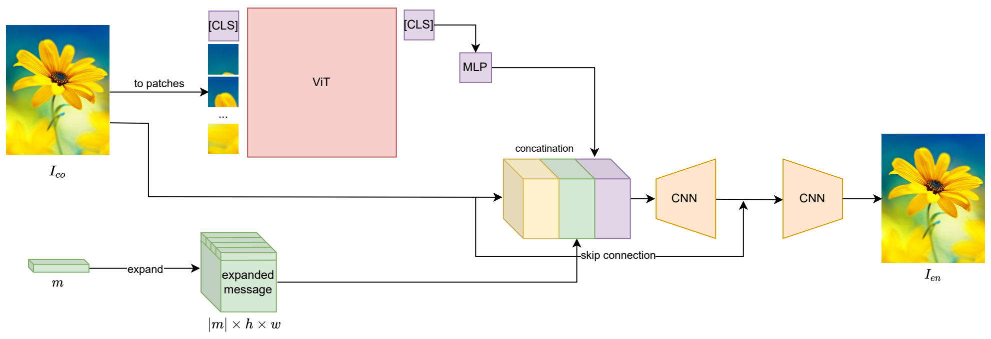

Figure 2: Our first approach, Encoder based ViT

3.3 Second Attempt: Pre-trained DINO Based Encoder Using the One-Dimensional Embedding

DINO ([8]) is a self-supervised system by Facebook AI. It is able to learn incredible representations from unlabeled data. The network learns through a process called *self-distillation*. There are a teacher and a student networks, both having the same architecture of a Vision Transformer (ViT). Some attention maps shows that DINO is able to focus exclusively on objects of interest in the image. This means that DINO is able to understand the object semantics so well that it's attention map looks like a segmentation mask. The teacher and student each predict a one-dimensional embedding. A softmax along with cross entropy loss are applied to make the student's distribution match the teacher's.

Under this approach we use the same idea of section 3.2, but instead of a ViT we used a pre-trained DINO (ViT-S/8 architecture) model as our backbone. First we apply DINO to the input I_{co}. Then we take the output of DINO, namely a one-dimensional embedding of the student, connect it to a linear layer and reshape the output to the form of $C' \times H \times W$.

3.4 Third Attempt: DINO Based Encoder Using Attention Map

Under this approach we use DINO again as our backbone. But, instead of taking the output of DINO, we take the attentions of the last layer of DINO with shape $heads \times patches \times patches$. We extract the patches that adopt to the CLS token and reshape it to a square in order to get a tensor at size $heads \times \sqrt{patches} \times \sqrt{patches}$. Actually this tensor is the attention map of DINO and thus it represents objects of interest in the image. Our hope was that this tensor has a correlation with the locations of sets of pixel in the image that are more robust for data distortions. We resize the tensor to the shape of $C' \times H \times W$, when here C' is the number of heads in the multi-head attention layer.

4 Data

Microsoft COCO: Common objects in context [6]. A large-scale object detection, segmentation, and captioning dataset. Almost all the papers we saw in the context of digital watermarking used this data set. All our models are trained on 10,000 cover images from the COCO training set, resized to experiment-specific dimensions. The messages are sampled with each bit drawn uniformly at random. We use 1,000 images for evaluation.

5 Experiments and Analysis of the Results

Similarly to other done works, we evaluate our model on two axes:

- **Robustness**. The extent of robustness to image distortions. Can be evaluated using bit accuracy: The hamming distance between the encoded message and the decoded message.
- **Perceptibility**. The distortion between the cover image and the watermarked image.

Note that there is a trade-off between these metrics. Models which have higher robustness typically sacrifice perceptibility.

We train our model on color images of size $C \times H \times W = 3 \times 128 \times 128$ with message length $L = 30$ and weight factors $\eta_I = 0.7$ and $\eta_M = 1$.

By training our models on the different noise layers, we show that all our three models can learn robustness to a variety of different image distortions.

Qualitative Results. Table 1 shows a comparison between our three models implementation (presented in sections 3.2-3.4) to HiDDeN published implementation. The comparison is conducted on 4 parameters: bitwise-error is the hamming distance between the encoded message and the decoded message, decoder-loss is the ℓ_2 distance between the input message and the output message before rounding, encoder-loss is the ℓ_2 distance between the input image I_{co} and the encoder image I_{en} and duration is the average time for one epoch in seconds. We compare HiDDeN to our model by the same number of epochs (between 180 to 300 epochs, depend on the type of noise). Examples of encoded 128 $\times$ 128 images are shown in Figure 3.

Analysis. In our first attempt we got comparable results to HiDDeN, and even got better results in some of the metrics. Under most of the noises (Crop, Cropout and Resize), HiDDeN achieves better results in the encoder loss, namely the cover image and the encoder image are more similar then ours. A possible reason is that HiDDeN also uses adversarial training, meaning they have a discriminator that gets the cover image and the encoded image and needs to decide if the image is a cover or an encoded image. HiDDeN's paper also pointed out that "adversarial training significantly improves the visual quality of the encoded images". On the other hand, under most of the

Metric/Noise	Crop (0.2, 0.25)	Dropout (0.55-0.6)	Cropout (0.55-0.6)	resize (0.7, 0.8)
ViT based encoder				
bitwise error	**0.01**/0.07	0.022/0.01	**0.074**/0.111	**0.004**/0.008
decoder loss (ℓ_2)	**0.03**/0.05	0.087/0.034	0.122/0.0942	**0.025**/0.029
encoder loss (ℓ_2)	0.005/0.003	**0.0027**/0.0031	**0.001**/0.002	0.006/0.003
epoch duration (sec)	71/66	**95**/96	**94**/96	**71**/81
DINO one-dimentional embedding based encoder				
bitwise error	**0.0613**/0.07	0.1078/0.01	0.1556/0.111	-
decoder loss (ℓ_2)	0.0699/0.05	0.1067/0.034	0.1474/0.0942	-
encoder loss (ℓ_2)	0.0065/0.003	**0.0029**/0.0031	0.0049/0.002	-
epoch duration (sec)	280/66	297/96	290/96	-
DINO with attention map based encoder				
bitwise error	0.0806/0.07	0.0709/0.01	0.1907/0.111	-
decoder loss (ℓ_2)	0.0669/0.05	0.0659/0.034	0.1614/0.0942	-
encoder loss (ℓ_2)	0.0083/0.003	0.0055/0.0031	0.0042/0.002	-
epoch duration (sec)	344/66	350/96	360/96	-

Table 1: Transformark (all our three modeld) vs HiDDeN. The left value in each cell match to our implementation, where the right values is HiDDeN implementation. The highlighted values are the ones that we got better results then HiDDeN

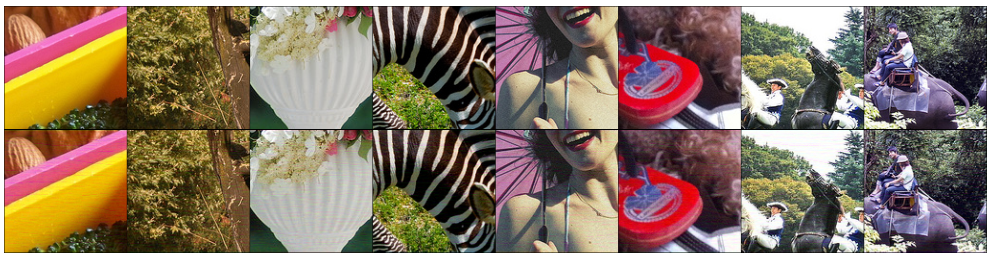

Figure 3: Results of our first model trained on Dropout noise after 193 epochs of training. The first row is the cover images and the second row is the corresponding encoded images. The reader can see that the images are almost indistinguishable, while the decoder succeeds to recover almost 98% of the original messages (after applying the Dropout noise to the encoded image).

noises we got better bit-wise error, meaning that we are able to recover the original message with higher accuracy. We got nice results under the Crop noise layer. With Crop, we succeed to recover the message with accuracy of 99% while HiDDeN architecture was able to restore the message with 93% of accuracy.

The results we got in the second approach, DINO based encoder with one-dimensional embedding, were worse, but are still comparable to HiDDeN. We reason that this is because the semantic representation that DINO learns is not very helpful for our mission, since the network should learn where is best to encode the message, and apparently there is no large correlation between this and the semantics of the cover image. On the other hand, the first approach succeeds to learn some useful representation. One option to try out is to train the pre-trained DINO model. In our implementation we freezes DINO weights, but it might be that the pre-trained DINO model can learn some beneficial representation faster than our first ViT model.

The results of the third approach appeared to be worse even than the second approach. We were a little bit surprised as we guessed that the model can benefit more from the attention map than the one-dimensional embedding (see [7]). A possible reason is that the second approach includes also a MLP that learns to convert the semantic representation to a beneficial information, whereas the third approach doesn't have any learnable parts at all in the ViT mechanism.

6 Conclusion

We proposed a deep network for digital watermarking based on ViTs. Through empirical evaluations, we have shown that our model reaches comparable performance to HiDDeN, and even get better results in some of the metrics. This leads us to think that there is a potential to the usage of ViT in watermarking tasks.

Further Work. Similarly to HiDDeN's model, our model training demands a lot of time and resources. It took us between 4 to 12 hours to train our model per each noise. Due to the time constraints, we didn't have the opportunity to play enough with the parameters of the ViT and the weights of the losses. We believe that paying extra attention to these parameters can lead to a significant improvement in the results. Moreover, it is interesting to combine the same ideas in the decoder architecture as well. A second improvement that can be done is to combine the adversarial loss of HiDDeN into our model. We believe it may improve the visual difference of the encoded images significantly. One more interesting approach is to check out another loss functions (other than ℓ_2), that can capture similarity between images more accurately. Another improvement to our model could be an addition of a regularization. We know that regularization can change the results of a network dramatically. One option for a regularization policy here is to "validate" that the decoder gives the same result on two images that encodes the same message.

Acknowledgments. We want to thank our advisor Shay Bagon and Shir Amir for the useful discussions, Narek Tumanyan for helping us to understand vision transformers better and Irad Zehavi for unraveling us the mysteries of adversarial examples.

References

[1] Kurakin, A., Goodfellow, I., Bengio, S. *Adversarial examples in the physical world* In: ICLR Workshop. (2017)

[2] Jiren Zhu, Russell Kaplan, Justin Johnson and Li Fei-Fei. *HiDDeN: Hiding Data With Deep Networks.* In: ECVV (2018)

[3] Van Schyndel, R.G., Tirkel, A.Z., Osborne, C.F. *A digital watermark. In: IEEE Converence on Image Processing. In: IEEE (1994)*

[4] Bi, N., Sun, Q., Huang, D., Yang, Z., Huang, J. *Robust image watermarking based on multiband wavelets and empirical mode decomposition.* In: IEEE Transactions on Image Processing (2007)

[5] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. *Distortion Agnostic Deep Watermarking.* Google Research, Stanford University (2020).

[6] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll ́ar, P., Zitnick, C.L.: *Microsoft COCO: Common objects in context.* In: ECCV. (2014)

[7] Yu, C. *Attention Based Data Hiding with Generative Adversarial Networks.* In: AAAI Conference on Artificial Intelligence, 34(01), 1120-1128. (2020).

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, Armand Joulin. *Emerging Properties in Self-Supervised Vision Transformers.* DINO. In: ICCV (2021).

[9] Behnam Neyshabur. *Towards learning convolutions from scratch.* arXiv preprint arXiv:2007.13657, 2020.