Watermarking Images in Self-Supervised Latent Spaces
Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, Matthijs Douze

To cite this version:

HAL Id: hal-03591396
https://inria.hal.science/hal-03591396
Submitted on 28 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
WATERMARKING IMAGES IN SELF-SUPERVISED LATENT SPACES

Pierre Fernandez1, Alexandre Sablayrolles1, Teddy Furon2*, Hervé Jégou1, Matthijs Douze1

1Meta AI, 2Univ. Rennes, Inria, CNRS, IRISA

ABSTRACT

We revisit watermarking techniques based on pre-trained deep networks, in the light of self-supervised approaches. We present a way to embed both marks and binary messages into their latent spaces, leveraging data augmentation at marking time. Our method can operate at any resolution and creates watermarks robust to a broad range of transformations (rotations, crops, JPEG, contrast, etc.). It significantly outperforms the previous zero-bit methods, and its performance on multi-bit watermarking is on par with state-of-the-art encoder-decoder architectures trained end-to-end for watermarking. The code is available at github.com/facebookresearch/ssl_watermarking.

Index Terms—deep watermarking, self-supervised learning

1. INTRODUCTION

Watermarking embeds a secret message into an image under the constraints of (1) imperceptibility — the distortion induced by the watermark must be invisible, (2) payload — the hidden message is a binary word of a given length, (3) robustness — the decoder retrieves the hidden message even if the image has been distorted to some extent. This paper deals with blind watermarking where the decoder does not have access to the original image. The classic approach, coined TEmIt (Transform, Embed, Inverse transform) by T. Kalker, embeds the watermark signal in the feature space of a transform (e.g. DFT, DCT, Wavelet). It provides perceptually significant coefficients (TEmIt) reliable for watermarking as conceptualized in [1, Sec. 8.1.3].

Watermarking is enjoying renewed interest from advancements in deep learning. New methods improve the robustness to a broad range of alterations thanks to neural networks offering a reliable latent space where to embed the information. Examples include directly marking into the semantic space resulting from a supervised training over a given set of classes like ImageNet [2], or explicitly training a watermarking network to be invariant to a set of image perturbations. In this case, networks are usually encoder-decoder architectures trained end-to-end for watermarking [3, 4, 5, 6].

Our key insight is to leverage the properties of self-supervised networks to watermark images. Ideally, according to [1], a perceptually significant coefficient does not change unless the visual content of the image is different. Similarly, some self-supervised methods aim to create representations invariant to augmentations, without explicit knowledge of the image semantics [7, 8]. These pre-trained networks offer us the desired embedding space "for free", saving us the heavy training of end-to-end architectures like HiDDeN [3].

In order to robustly embed in the latent spaces, gradient descent is performed over the pixels of the images. To further ensure both robustness and imperceptibility of the watermarks, we include data augmentation and image pre-processing at marking time.

Our contributions are the following:

• We provide a watermarking algorithm that can encode both marks and binary messages in the latent spaces of any pre-trained network;
• We leverage data augmentation at marking time;
• We experimentally show that networks trained with self-supervision provide excellent embedding spaces.

2. RELATED WORK

Image watermarking [1] approaches are often classified by their embedding space: few use the spatial domain [9, 10], most of them follow the TEmIt principle with a well known transformation like DFT [11], DCT [12] or DWT [13]. In zero-bit watermarking [14], the embedder only hides a mark and the detector checks for its presence in the content. For instance, the received content is deemed watermarked if its descriptor lies in a hypercone of the feature space. This detection strategy is near optimal [15, 16]. In multi-bit watermarking, the embedder encodes a binary word into a signal that is hidden in the image. At the other end, the decoder retrieves the hidden message bit by bit. This is the case for most deep-learning-based methods presented below.

Deep-learning-based watermarking has emerged as a viable alternative to traditional methods. The networks are often built as encoder-decoder architectures [3, 17], where an encoder embeds a message in the image and a decoder tries to extract it. For instance, HiDDeN [3] jointly trains encoder and decoder networks with noise layers that simulate image perturbations. It uses an adversarial discriminator to improve visual quality, and was extended to arbitrary image resolutions and message lengths by Lee et al. [18]. Distortion Agnostic [19] adds adversarial training that brings robustness to unknown transformations and [6, 20] embed the mark with an attention filter further improving imperceptibility. ReDMark [5] adds a circular convolutional layer that diffuses the watermark signal all over the image. Finally, ROMark [4] uses robust optimization with worst-case attack as if an adversary were trying to remove the mark. For more details, we refer to the review [21]. This type of methods has its drawbacks: e.g. it is hard to control the embedding distortion and it is made for decoding and does not translate so well to detection.

Vučić et al. [22] mark images with a neural network pre-trained on supervised classification instead of an encoder-decoder architecture. The network plays the role of the transform in a TEmIt approach. Since it has no explicit inverse, a gradient descent to the image pixels "pushes" the image feature vector into a hypercone. The follow-up work [2] increases the inherent robustness of the network by applying increasingly harder data augmentation at pre-training. It offers a guarantee on the false positive rate without requiring to train a network explicitly for watermarking, but no multi-bit version was proposed.

*Thanks to ANR and AID french agencies for funding Chaire SAIDA.
3. WATERMARKING WITH SSL NETWORKS

Our method adopts the framework of Vukotić et al. [2]. We improve it by showing that networks build better watermarking features when trained with self-supervision, and by introducing data-augmentation at marking time. We also extend it to multi-bit watermarking.

3.1. Using self-supervised networks as feature extractors

**Motivation.** We denote the image space by \(\mathcal{I}\) and the feature space by \(\mathcal{F} = \mathbb{R}^d\). The feature extractor \(\phi: \mathcal{I} \rightarrow \mathcal{F}\) must satisfy two properties: (1) geometric and valuemetric transformations on image \(I\) should have minimal impact on feature \(\phi(I)\), (2) it should be possible to alter \(I\) by applying invisible perturbations to the image in order to embed the watermark signal.

We choose \(\phi\) as a neural network pre-trained by self-supervised learning (SSL). Our assumption is that SSL produces excellent marking spaces because (1) it explicitly trains features to be invariant to data augmentation; and (2) it does not suffer from the semantic collapse of supervised classification, that gets rid off any information that is not necessary to assign classes [23]. From the recent SSL methods of the literature (contrastive learning [24, 25], statistical prior [26], teacher/student architecture [8]), we select DINO [7] for its training speed and good content-based image retrieval performance.

**DINO pre-training.** DINO [7] pertains to the teacher/student approach. The teacher and the student share the same architecture. Self distillation with no labels trains the student network to match the outputs of the teacher on different views of the same image. The student is updated by gradient descent while the teacher’s parameters are updated as an exponential moving average of the student’s parameters: \(\theta_t \leftarrow \lambda \theta_t + (1 - \lambda)\theta_s\), with \(\lambda \lesssim 1\).

The invariance of the features is ensured by random augmentation during the training: valuemetric (color jittering, Gaussian blur, solarization) and geometric (resized crops) transformations. Furthermore, DINO encourages local-to-global correspondence by feeding only global views to the teacher while the student sees smaller crops.

**Normalization layer.** The watermark embedding must drive the feature to an arbitrary space region (defined by a secret key and the message to hide). It is essential that the features are not concentrated onto a manifold far away from this region. Although Wang et al. [27] show that contrastive learning optimizes the uniformity of the features on the hypersphere, it does not occur in practice with DINO. To alleviate the issue, the output features are transformed by PCA-whitening (a.k.a. PCA-sphering). This learned linear transformation [28] outputs centered vectors with unit covariance of dimension \(d = 2048\).

3.2. Marking with back-propagation and augmentation

The marking takes an original image \(I_o\) and outputs a visually similar image \(I_w\). In the image space \(\mathcal{I}\), the distortion is measured by \(\mathcal{L}_i: \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}_+\). An example is the MSE: \(\mathcal{L}_i(I_w, I_o) = \|I_w - I_o\|^2\) / \(h/w\), but it could be replaced by perceptual image losses such as LPIPS [29].

In the feature space \(\mathcal{F}\), we define a region \(\mathcal{D}\) which depends on a secret key (zero-bit and multi-bit setups) and the message to be hidden (only in multi-bit setup). Its definition is deferred to Sect. 3.3 together with the loss \(\mathcal{L}_w: \mathcal{F} \rightarrow \mathbb{R}\) that captures how far away a feature \(x \in \mathcal{F}\) lies from \(\mathcal{D}\). We also define a set \(\mathcal{T}\) of augmentations, which include rotation, crops, blur, etc., each with a range of parameters. \(\text{Tr}(I, t) \in \mathcal{I}\) denotes the application of the transformation \(t \in \mathcal{T}\) to image \(I\).

If the feature extractor is perfectly invariant, \(\phi(\text{Tr}(I, t))\) lies inside \(\mathcal{D}\) if \(\phi(I)\) does. To ensure this, the watermarking uses data augmentation. The losses \(\mathcal{L}_w\) and \(\mathcal{L}_i\) are combined in:

\[
\mathcal{L}(I_w, I_o, t) := \lambda \mathcal{L}_w(\phi(\text{Tr}(I, t))) + \mathcal{L}_i(I, I_o).
\]

The term \(\mathcal{L}_w\) aims to push the feature of any transformation of \(I_w\) into \(\mathcal{D}\), while the term \(\mathcal{L}_i\) favors low distortion. The training approach is typical for the adversarial attacks literature [30, 31]:

\[
I_w := \arg \min_{I \in \mathcal{C}(I_o)} \mathbb{E}_{t \sim \tau}[\mathcal{L}(I, I_o, t)]
\]

where \(\mathcal{C}(I_o) \subset \mathcal{I}\) is the set of admissible images w.r.t. the original one. It is defined by two steps of normalization applied to the pixel-wise difference \(\delta = I - I_o\): (1) we apply a SSIM [32] heatmap attenuation, which scales \(\delta\) pixel-wise to hide the information in perceptually less visible areas of the image; (2) we set a minimum target PSNR and rescate \(\delta\) if this target is exceeded.

The minimization is performed by stochastic gradient descent since the quality constraints, \(\mathcal{L}\) and \(\mathcal{L}_i\) are differentiable w.r.t. the pixel values. Stochasticity comes from the fact that expectation \(\mathbb{E}_{t \sim \tau}\) is approximated by sampling according to a distribution over \(\tau\). The final image is the rounded version of the update after \(K\) iterations.

3.3. Detection and decoding

We consider two scenarios: zero-bit (detection only) and multi-bit watermarking (decoding the hidden message). Contrary to HiDDen [3] and followers, our decoding is mathematically sound.

**Zero-bit.** From a secret key \(a \in \mathcal{F}\) s.t. \(\|a\| = 1\), the detection region is the dual hypercone:

\[
\mathcal{D} := \left\{ x \in \mathbb{R}^d : |x^\top a| > \|x\| \cos(\theta) \right\}.
\]

It is well grounded because the False Positive Rate (FPR) is given by

\[
\text{FPR} := \mathbb{P}(\phi(I) \in \mathcal{D} | \text{"key} \ a \ \text{uniformly distributed")}
\]

\[
= 1 - I_{\cos^2(\theta)} \left( \frac{1}{2} - \frac{d - 1}{2} \right)
\]

where \(I_s(\alpha, \beta)\) is the regularized Beta incomplete function. Moreover, the best embedding is obtained by increasing the following function under the distortion constraint:

\[
-\mathcal{L}_w(x) = (x^\top a)^2 - \|x\|^2 \cos^2 \theta.
\]

This quantity is negative when \(x \notin \mathcal{D}\) and positive otherwise. I. Cox et al. originally called it the robustness estimate [1, Sec. 5.1.3]. This hypercone detector is optimal under the asymptotical Gaussian setup of [15, 16].

**Multi-bit.** We now assume that the message to be hidden is \(m = (m_1, \ldots, m_k) \in \{-1, 1\}^k\). The decoder retrieves \(m = D(I)\). Here, the secret key is a randomly sampled orthogonal family of carriers \(a_1, \ldots, a_k \in \mathbb{R}^d\). We modulate \(m\) into the signs of the projections of the feature \(\phi(I)\) against each of the carriers, so the decoder is: \(D(I) = \text{sign} (\phi(I)^\top a_1), \ldots, \text{sign} (\phi(I)^\top a_k))\).

At marking time, the functional is now defined as the hinge loss with margin \(\mu \geq 0\) on the projections:

\[
\mathcal{L}_w(x) = \frac{1}{k} \sum_{i=1}^k \max(0, \mu - (x^\top a_i) m_i).
\]
4. EXPERIMENTS & RESULTS

4.1. Experimental setup and implementation details

Data. We evaluate our method on: 1000 images of YFCC100M dataset [33] which is selected for the variety of its content, CLIC [34] composed of 118 professional high-resolution images when comparing to [2], and 1000 images of MS-COCO [35] composed of smaller images for comparison with [3, 19].

Backbone pre-training. We use the ResNet-50 [36] architecture as backbone model to extract features from its last convolutional layer (\(d = 2048\)). It is trained on ILSVRC2012 [37] without labels, using 200 epochs of DINO self-supervised learning with the default parameters [7] and with rotation augmentation. Models trained for classification come from the torchvision library [38]. The PCA-whitening is learned on 100k distinct images from YFCC (resp. COCO) when evaluating on YFCC and CLIC (resp. COCO).

Embedding. We first set a desired FPR (which defines the hypercone angle \(\theta\)) and a target PSNR. Watermarking (2) then uses the Adam optimizer [39] with learning rate 0.01 over 100 gradient descent iterations. The weight in (1) is set to \(\lambda = 1\) (zero-bit) or \(\lambda = 5 \cdot 10^4\) (multi-bit). The margin of (6) is set to \(\mu = 5\).

At each iteration, the preprocessing step performs the SSIM attenuation and clips the PSNR to the target. SSIM heatmaps are computed with \(C_1 = 0.01^2\), \(C_2 = 0.03^2\) and over 17×17 tiles of the image’s channels, then summed-up and clipped to be non-negative, which generates a single heatmap per image. Then, a transformation \(t\) is chosen randomly in \(T\) (identity, rotation, blur, crop or resize). The rotation angle \(\alpha\) is sampled from a Von Mises distribution with \(\mu = 0\), \(\kappa = 1\) and divided by \(2\). This generates angles in \(\pi/2 \times [-1, 1]\) with a higher probability for small rotations, that are more frequent in practice. The crop and resize scales are chosen uniformly in \([0.2, 1.0]\). The crop aspect ratio is also randomly chosen between \(3/4\) and \(4/3\). The blurring kernel size \(b\) is randomly drawn from the odd numbers between \(1\) and \(15\) and \(\sigma\) is set to \(0.15\cdot\sigma\). Finally, the image is flipped horizontally with probability 0.5.

Transformations. The following table presents the transformations used at pre-training, marking or evaluation stages. Parameter \(p\) represents the cropping ratio in terms of area, \(Q\) is the quality factor of the JPEG format and \(H\) and \(C\) are defined in [38]. "Meme format" and "Phone screenshot" come from the Augly library [40].

<table>
<thead>
<tr>
<th>Transformations</th>
<th>Parameter</th>
<th>Type</th>
<th>Geom</th>
<th>Value</th>
<th>Used for</th>
<th>Train</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation</td>
<td>angle (\alpha)</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td></td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Crop</td>
<td>ratio (p)</td>
<td>✓</td>
<td>x</td>
<td></td>
<td></td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Resize</td>
<td>scale (p)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Gaussian Blur</td>
<td>width (\sigma)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Brightness</td>
<td>(B)</td>
<td>x</td>
<td>x</td>
<td></td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Contrast</td>
<td>(C)</td>
<td>x</td>
<td>x</td>
<td></td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Hue</td>
<td>(H)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>JPEG</td>
<td>quality (Q)</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Meme format</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Phone screenshot</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

4.2. Zero-bit watermarking

Trade-offs. The hypercone angle \(\theta\) in (4) is given by the target FPR. A higher FPR implies a wider angle, making the method robust against more severe attacks, at the cost of detecting more false positives. The FPR is set to \(10^{-16}\) in further experiments. Large-scale applications usually operate at low FPR to avoid human verification. As a sanity check we run detection on 100k natural images from YFCC, none of which are found to be marked. Similarly, there is only one false positive out of the 1,281,167 images of ImageNet. Another trade-off lies in the imperceptibility, since allowing greater distortions (lower PSNR) improves the robustness. It is illustrated in Fig. 1.

Ablation studies. We showcase the influence of self supervision at pre-training and of augmentation at marking time. The performance measure is the True Positive Rate (TPR), at a target PSNR=40 dB, and FPR=\(10^{-6}\). Fig. 2 evaluates the robustness for the specific case of the rotation. Rotation augmentation is needed both at pre-training and marking stages to achieve high robustness against it. Comparison on a wider range of transformations is given in Tab. 1.

Comparision with the state of the art. Table 1 compares our method with [2] on CLIC. In their setup, the FPR=\(10^{-3}\) and PSNR must be \(\geq 42\text{dB}\). Overall, our method gives better results on CLIC than on YFCC because images have higher resolutions (hence more pixels can be used to convey the mark). We observe a strong improvement over [2], especially for large rotations, crops and Gaussian blur where our method yields almost perfect detection over the 118 images.

![Fig. 1: Robustness against image transformations. Top: PSNR set at 40 dB and FPR decreasing from \(10^{-2}\) (red) to \(10^{-12}\) (blue). Bottom: FPR set at \(10^{-6}\) and PSNR ranging from 52 dB to 32 dB.](image1)

![Fig. 2: Robustness against rotation. Each row (column) represents different ±amplitude of the rotation at training (resp. marking (2)).](image2)
Comparison with the state of the art. Table 3 compares against two deep data hiding methods [3, 19] using their setting: a payload of 30 bits, a target PSNR of 33 dB, over 1000 images from COCO resized to 128 × 128. Overall, our method gives comparable results to the state of the art, except for the center crop transform where it fails to achieve high bit accuracies. Note also that the resize operation is not used as noise layer in either of [3, 19], which means that our method should have the advantage. In contrast, while these methods are trained to be robust to JPEG compression with a differentiable approximation, our method achieves similar performance without specific training.

Furthermore, our method easily scales to higher resolution images, where it achieves lower BER for a fixed payload. We assume that [3, 19] also scales but at the cost of a specific training for a given resolution. This training is more computationally expensive since the message is repeated at each pixel [3]. It also needs a smaller batch-size to operate larger images, and new hyperparameters values.

Table 3: Comparison of BER. The first row uses original resolutions of COCO, while the others use a resized version (to 128 × 128).

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Identity</th>
<th>Jpeg (30)</th>
<th>Blur (1.0)</th>
<th>Brightness (0.1)</th>
<th>Meme</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>0.001</td>
<td>0.04</td>
<td>0.001</td>
<td>0.181</td>
<td>0.001</td>
<td>0.03</td>
</tr>
<tr>
<td>Ours, 128 × 128</td>
<td>0.001</td>
<td>0.16</td>
<td>0.011</td>
<td>0.451</td>
<td>0.181</td>
<td>0.06</td>
</tr>
<tr>
<td>HiDDEn [3]</td>
<td>0.001</td>
<td>0.23</td>
<td>0.011</td>
<td>0.001</td>
<td>0.15</td>
<td>0.29</td>
</tr>
<tr>
<td>Dist. Agnostic [19]</td>
<td>0.001</td>
<td>0.181</td>
<td>0.071</td>
<td>0.021</td>
<td>0.12</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Table 2: BER and WER (%) for 30-bits encoding at PSNR 40dB.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>BER</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone screenshot</td>
<td>0.1</td>
<td>3.3</td>
</tr>
<tr>
<td>Crop (0.1)</td>
<td>4.8</td>
<td>28.9</td>
</tr>
<tr>
<td>Res. (25)</td>
<td>2.1</td>
<td>0.5</td>
</tr>
<tr>
<td>Blur (2.0)</td>
<td>8.7</td>
<td>8.4</td>
</tr>
<tr>
<td>Jpeg (30)</td>
<td>2.5</td>
<td>6.4</td>
</tr>
<tr>
<td>Bright (2.0)</td>
<td>23.9</td>
<td></td>
</tr>
<tr>
<td>Meme</td>
<td>19.2</td>
<td></td>
</tr>
<tr>
<td>Screenshot</td>
<td>800</td>
<td></td>
</tr>
</tbody>
</table>

4.3. Multi-bit hiding

Quantitative results. We evaluate the method on YFCC, with a target PSNR of 40 dB and a payload k of 30 random bits as in [3, 19]. Tab. 2 presents the Bit and Word Error Rate (BER and WER) over various attacks. The decoding achieves low rates over a wide range of geometric (rotation, crops, resize, etc.) and valuemetric (brightness, hue, contrast, etc.) attacks. Rotation and Gaussian blur are particularly harmless since they are seen both at pre-training and at marking time. Some images are harder to mark, which can be observed statistically on the empirical WER reported in Tab. 2. If all images were as difficult to mark, then BER would be equal for all images, and WER = 1 − (1 − BER)k. Yet, the reported BER are significantly lower; e.g. for Brightness, WER = 0.607 < 1 − (1 − 0.087)30 = 0.935. Empirically, we see that images with little texture are harder to watermark than others, due to the SSIM normalization. In practice, ECC can be used to achieve lower WERs.

Qualitative results. We notice that the watermark is perceptually less visible for multi-bit than for zero-bit watermarking at a fixed PSNR. Our explanation is that the energy put into the image feature is more spread-out across carriers in the multi-bit setup than on the zero-bit one where the feature is pushed at much as possible towards a single carrier. Images are not displayed due to lack of space.

5. CONCLUSION & DISCUSSION

This paper proposes a way to robustly and invisibly embed information into digital images, by watermarking onto latent spaces of off-the-shelf self-supervised networks. By incorporating data augmentation and constraints into the marking process, our zero-bit watermarking method greatly improves performance over the baseline [2]. It is robust against a wide range of transformations while keeping high fidelity with regards to the original images, and ensuring a very low false positive rate for the detection. When we extend the method to multi-bit watermarking, we obtain promising results, comparable to the state-of-the-art in deep data hiding, and even better with regards to some transformations of the image (e.g. JPEG compression or blur).

Most interestingly, networks trained with self-supervision naturally generate excellent watermarking spaces, without being explicitly trained to do so. However, compared to encoder-decoder deep watermarking techniques, watermarking images with our method is expansive since it is not a single pass forward. In future works, we hope to show that further adapting the network for the specific task of watermarking would improve performance and efficiency.
6. REFERENCES


[33] Bart Thomée, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li, “Yfcc100m: The new data in multimedia research,” in *ECCV*, 2014.

[34] “Workshop and challenge on learned image compression.”


