PRIVACY-PRESERVING IMAGE SHARING VIA SPARSIFYING LAYERS ON CONVOLUTIONAL GROUPS

Sohrab Ferdowsi[*]*, Behrooz Razeghi*[*‡]*, Taras Holotyak*[*]*, Flavio P. Calmon*[†]*, Slava Voloshynovskiy*[*]

[*] University of Geneva [†] Harvard University

ABSTRACT

We propose a practical framework to address the problem of privacy-aware image sharing in large-scale setups. We argue that, while compactness is always desired at scale, this need is more severe when trying to furthermore protect the privacy-sensitive content. We therefore encode images, such that, from one hand, representations are stored in the public domain without paying the huge cost of privacy protection, but ambiguated and hence leaking no discernible content from the images, unless a combinatorially-expensive guessing mechanism is available for the attacker. From the other hand, authorized users are provided with very compact keys that can easily be kept secure. This can be used to disambiguate and reconstruct faithfully the corresponding access-granted images. We achieve this with a convolutional autoencoder of our design, where feature maps are passed independently through sparsifying transformations, providing multiple compact codes, each responsible for reconstructing different attributes of the image. The framework is tested on a large-scale database of images with public implementation available.

Index Terms— privacy-preserving image sharing, convolutional autoencoders, sparse representation, learned compression, image obfuscation.

1. INTRODUCTION

In the era of big data, with recent advances in data driven machine learning frameworks coupled with growing concern about the privacy of individuals' identities when shared with third parties, it is desirable to release *useful representation* of data while simultaneously satisfying some *privacy constraints*. We consider scenarios where the data owner collects massive amounts of data to provide some utility for the authorized users (clients). For instance, governments, social media or fin-tech databases posses facial images of citizens or customers that should be efficiently communicated with several of their trusted partners, while strictly having to protect the privacy of the individuals. The engineering goal in such cases is to design a practical multi-party data-sharing mechanism with constraints on *data utility* and *data privacy*.

There is a rich literature of prior work on privacy-assuring mechanisms based, e.g., on cryptographic methods [1], differential private based techniques [2], generative adversarial models [3, 4] or embedding based schemes [5], just to name a few. The classical privacy-preserving image release mechanisms were mostly based on obfuscation techniques, such as pixelization and bluring [6,7], or partial encryption such as P3 [8], which are defeated using machine learning methods [9]. The framework of Sparse Coding with Ambiguation (SCA) [10, 11] shares a compressed, but ambiguated representation of data within the public domain, while the users can benefit some utility given an authorized query. This work is closely related to the SCA, however, rather than information-theoretic arguments, our focus here is mostly computational: Firstly, we formulate the requirements of a privacy-aware data-driven image sharing mechanism, where the data is split between public and trusted parties. Secondly, we provide an actual implementation of this framework in practice for the case of images using Convolutional Neural Networks (CNNs).

Concretely, consider a three-party image/visual information sharing or usage scenario that involves (a) a data owner, (b) data users, and (c) service provider(s). The data owner outsources some representations of the images that s/he owns to the 'honest but curious' server(s) for storage or further communication or sharing with data users. S/he attempts to: (1) protect the original data collection from server side analyses interested in knowing the data content provided by the data owner; (2) provide a pre-determined utility (e.g. reconstruction) for his/her authorized clients; (3) protect original data collection against the un-authorized parties. In order to follow Kerckhoffs's Principle in cryptography, we further assume that the data-sharing mechanism is publicly known, but a secret key used for the data protection is kept secret.

To avoid expensive solutions to provide security for privacy-sensitive data, e.g., through cryptographic encryption, we propose to ambiguate the data representation (as detailed next), and instead provide security only for the disambiguation key. This scheme, of course, is only useful when the key is much smaller in size than the original data[1] (before and after ambiguation), for which we provide a practical solution. A key justification for this double-splitting of the data

[‡] Work done while at Harvard University. B. Razeghi has been supported by the ERA-Net project ID_IoT No 20CH21_167534.

Implementation codes available at: `https://github.com/sssohrab/sparsifying_groups_imAmbiguation`

[1] So e.g., a naïve permutation of image pixels is not useful, since the permutation map is even larger than the original image.

is the fact that compactness, while always desired at large-scale setups, is much more of a cruicial need for security and privacy solutions, both storage and communication.

While there is an extensive literature on using traditional image compression codecs to provide security along with compression, (e.g., see [12–15]), an important limitation arises with these approaches when they are used in large-scales and possibly for domain-specific images. Since in such scenarios images have similar encoding (e.g., high concentration of activation of DCT coefficients at certain regions), the adversary can benefit from this to infer the statistics of encoded images.[2] Moreover, even the compression capability of standard codecs have been seriously challenged by learning-based solutions. This has led to an active area of research that uses deep learning (see e.g., [16, 17]) to learn optimal compression. This work follows the learning-based approach, however, instead of the usual binary representations used in these methods, we focus on sparsity of representations similarly to [18]. This allows us to benefit from the SCA framework [11, 19] for ambiguation.

This paper presents two main contributions: Firstly, we introduce a data sharing setup as detailed in section 3, where the cost of security is minimized in terms of the amount of bits required, thanks to our end-to-end solution for capturing data redundancies with representation learning. Secondly, we provide a practical and scalable solution with two particular architectural novelties for CNNs: 1) Multiple code-maps using fully-connected groups on convolutional filters. 2) The k-sparsity non-linearity in CNNs along with ReLUs without slowing down training. This is detailed in section 3. The experimental setup and concluding remarks are then presented in sections 4 and 5, respectively.

2. PRIVACY-PRESERVING IMAGE SHARING

We encode the data $\mathbf{x} \in \mathcal{X}$, as the pair $(\mathbf{u}_p, \mathbf{u}_s)$, where $\mathbf{u}_p \in \mathcal{U}_p$ is the part of the data stored in public, while $\mathbf{u}_s \in \mathcal{U}_s$ is secured and is kept private. We require that, firstly, the representation size of $\mathbf{u}_s$ in bits be much smaller than that of $\mathbf{u}_p$. Secondly, the guessing cost of the data only given the public portion, i.e., $\mathbf{x}|\mathbf{u}_s$ should be exponential, while its reconstruction provided both parts, i.e., $\mathbf{x}|\mathbf{u}_s, \mathbf{u}_p$ should be linear.

2.1. Proposed Scheme Overview

We achieve the above constraints with the following steps:

1) *Neural network training:* A network is first trained on a sub-collection of the data, such that it produces compact sparse codes. This is shared with the public domain.

2) *Data owner encoding and ambiguating:* The data owner uses the trained network to sparsely encode all images s/he possesses. The support of these codes are then kept secure (e.g., through encryption or secure communication), and shared with their corresponding authorized parties. The collection of all sparse codes are then ambiguated, as will be detailed below, and then shared with the public domain.

3) *Data users/parties disambiguating their content:* The individuals or trusted parties are provided with the indices of the images that they have access to, as well as the secured supports. This acts as the key to unlock their access-granted content from the public ambiguated database.

[2]However, if the image compression bases and quantizers are learned, the network tries to spread-out the activities of the representations, as this provides better rate-distortion trade-offs for the learned network.

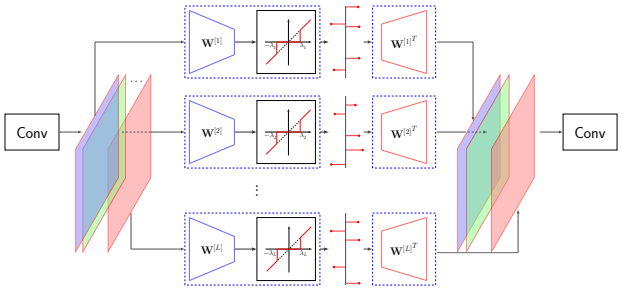

Fig. 1: The grouped linear block to produce sparse code-maps: In the encoder side, convolutional feature maps are independently fed to fully-connected linear layers and sparsified. Symmetrically, the decoder uses tied connections and reconstruct the convolutional feature maps. This block is put in the middle of the network.

2.2. Ambiguated Sparse Code Generation

Training Phase. Given the data samples $\mathbf{x} \in \mathbb{R}^n$, we train a bottlenecked auto-encoder structure consisting of L independent encoders, $\mathrm{Enc}^{[1]}(\cdot), \cdots, \mathrm{Enc}^{[L]}(\cdot)$, where input data variable $\mathbf{x}$ is encoded to L new representations as $\mathbf{z}^{[l]} = \mathrm{Enc}^{[l]}(\mathbf{x}), \forall l \in [L]$ with $\mathbf{z}^{[l]} \in \mathbb{R}^m$, and is (approximately) reconstructed as $\hat{\mathbf{x}}^{[l]} = \mathrm{Dec}^{[l]}[\mathbf{z}^{[l]}], \forall l \in [L]$. Furthermore, the encoding is performed such that the codes are k-sparse, i.e., $\mathrm{card}\left(\mathrm{supp}\left(\mathbf{z}^{[l]}\right)\right) = k, \forall\, l$, where $\mathrm{supp}\left(\mathbf{z}^{[l]}\right)$ is index set of nonzero entries of $\mathbf{z}^{[l]}$ and $\mathrm{card}\,(\cdot)$ denotes cardinality of the set. The sparsity level k controls the reconstruction fidelity of our auto-encoding mechanism.

Sharing Phase. The generated sparse representations are then ambiguated and shared to the public service provider. Given the sparse representation $\mathbf{z}^{[l]}$ with sparsity level k, the ambiguation mechanism $A\,(\cdot)$ adds $k_n \geq 0$ random noise components to the orthogonal complement of $\mathbf{z}^{[l]}$, with the same statistics as sparse code to guarantee the indistinguishably in the statistical properties. Therefore, we have:

$$\mathbf{u}_p^{[l]} = A\left(\mathbf{z}^{[l]}\right) = \mathbf{z}^{[l]} \oplus \mathbf{n}_{\mathrm{supp}}, \tag{1}$$

where $\mathbf{n}_{\mathrm{supp}}$ is random ambiguation noise added to the support complement of $\mathbf{z}^{[l]}$ and $\oplus$ denotes the direct sum. Note that $\|\mathbf{u}_p^{[l]}\|_0 = k + k_n = k'$. The ambiguated sparse representations $\mathbf{u}_p^{[l]}, \forall\, l$ are then shared to the public domain. The support of the latent representation $\mathbf{z}^{[l]}$, denoted by $\mathbf{u}_s^{[l]} = \mathrm{supp}\left(\mathbf{z}^{[l]}\right)$, is considered as the secure part of our data, which is shared with the private data users.

Reconstruction Phase. Given the support information $\mathbf{u}_s^{[l]}, \forall l \in [L]$, the service provider can reconstruct the data as: $\hat{\mathbf{x}}^{[l]} = \mathrm{Dec}^{[l]}(\mathbf{z}^{[l]} \mid \mathbf{u}_s^{[l]})$. Our encoding introduces a concept of *shared secrecy* based on support intersection of latent representations. We consider two hypotheses for support secrecy. $\mathcal{H}_1$: The authorized support secrecy $\mathbf{u}_s$, $\mathcal{H}_0$: The un-authorized support, generated and claimed by an adversary.

As a result of this encoding, the number of bits required to store an item of the secure part is:

$$H(\mathbf{u}_s) = \log_2 \binom{m}{k}^L \simeq mL \times H_2\Big(\frac{k}{m}\Big), \qquad (2)$$

where $H_2(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$ is the binary entropy, and the approximation follows the Stirling's. From the other hand, $\mathbf{u}_p$ is ambiguated and is k'-sparse, with $k' \geqslant k$ and for a typical 32-bit quantization of the non-zero values, requires approximately $H(\mathbf{u}_p) \simeq 32mL \times H_2\big(\frac{k'}{m}\big)$ bits of storage per item. The adversary should then make $\binom{k'}{k}^L$ guesses to reconstruct each item, i.e., $\mathcal{O}\big(\exp H(\mathbf{u}_s)\big)$.

3. AUTOENCODER ARCHITECTURE

We need a practical autoencoder architecture that has three properties: Firstly, it should be bottlenecked to provide compact codes. This excludes some of the famous neural regressors like the U-net [20], since they are not bottlenecked because of their skip-connections directly from the input to the output. Secondly, we need sparsified codes. The usage of sparsity-inducing non-linearities, however, is rare in the deep learning literature. The few examples available correspond to the line of work of "unrolling iterative algorithms as neural networks", e.g., as in LISTA [21,22], where sparsifying operators like the soft- or hard-thresholding functions are used as non-linearities within neural networks. This, however, is not common within the representation learning community.

Thirdly, we should be able to reconstruct images with arbitrarily chosen levels of fidelity, corresponding to a prescribed representation budget for $\mathbf{u}_s$ (and also a practical upper-bound for representation of $\mathbf{u}_p$).

To satisfy these requirements, we propose the "sparsifying linear layers on groups", as is schemed in Fig. 1. Note that, while the fully-connected linear layers in the literature are used in CNNs, mostly to bridge the convolutional features with the one-hot encoding of softmax for classification, we use them to diversify the activities of codes, since convolutional features are highly correlated and most activities are concentrated on small sub-sets of the feature space.

Note that it is not practical to simply reshape all convolutional filters and feed them directly to a linear layer, since this would require an extremely large matrix with intractable complexity and a very high risk of over-fitting. Therefore, we take each convolutional feature separately and pass it to a much smaller fully-connected linear layer, as if we have a large matrix with block-diagonal sparsity. As a byproduct, we notice that the network learns multiple codes, each describing different attributes of the image.

As far as the sparsifying operator is concerned, we craft a custom non-linearity (introduced in [18]) that only passes the k elements with largest magnitude, and zeros out other coefficients. Note that this, in fact, is an adaptive version of the hard-thresholding function, where the threshold is adapted to each input sample to choose only k elements out of m.

4. EXPERIMENTS

We conduct experiments on the large-scale CelebA [23] database of around $200,000$ images of size 128×128. We randomly split the dataset and picked 80% of the images only for training the network, and the rest only for testing. We used the PyTorch [24] framework to implement and train the above-explained architecture with 6 down-sampling convolutional blocks of ratios $[1,2,1,2,1,2]$, and a symmetric decoder. We trained the network for around 40 epochs using the standard Adam optimizer [25] and settings.

The network was designed to have $L = 20$ code-maps, each with length $m = 512$, and the sparsity of $k = 128$ per item and per code. The storage requirement to describe the support of each item is then $\frac{20 H_2(\frac{128}{512})}{8\times 1024} \simeq 1.01$KBytes, which is very practical for secure communication purposes.

After the training, images from the test set are first encoded and, in order to assess their quality, are then decoded with two sparsity values of $k = 64, 128$. These codes are then ambiguated with random noise to amount for a total sparsity value of $k' = 256$ (i.e., 128 fake components). In order to minimize the chances of the adversary for random guessing, we generate the ambiguation noise from the same distribution of the true non-zero informative components, i.e., the complementary truncated Gaussian distribution with adjusted threshold and variance. Note that even after ambiguation, the public database has a reasonably low storage cost, since the non-zero values can be further quantized without loss of quality. Since the adversary may have the knowledge that the true sparsity was $k = 128$, we then try to attack the system by picking k out of k' non-zero values. However, since the distribution of the non-zero activities is highly uniform (due to the network trying to maximize its rate-distortion performance), and the values were also added with the same distribution, this guess can only be random with uniform distribution.

Following the *shared secrecy* based on support intersection of data which is introduced by SCA privacy mechanism, the authorized data users can *purify* (unlock) the corresponding ambiguated representation. However, the un-authorized parties have no knowledge to 'unlock' the public stored representations. Fig. 3 compares the outcomes of different experiments visually and on 8 randomly-chosen images from the test set. We can clearly confirm the high fidelity of the encoded images for the authorized parties, as well as the non-distinguishable quality of reconstruction for the adversary.

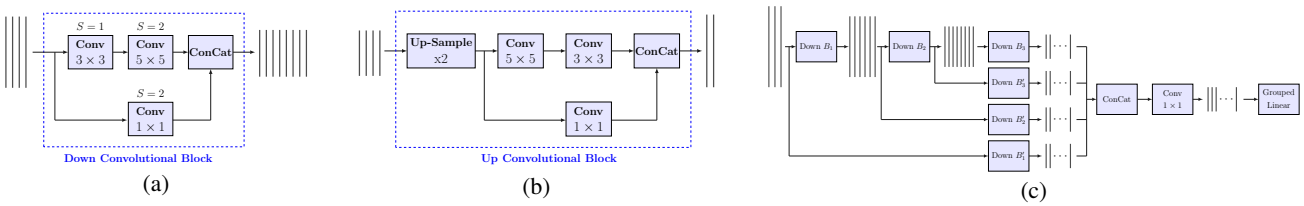

Fig. 2: (a) Convolutional down-sampling block. (b) Convolutional up-sampling block with bilinear interpolation. (c) Structure of the encoder of the network, where B_i blocks are according to (a), and B'_i blocks use addition instead of concatenation for skip-connections. The Grouped Linear block is sketched in Fig. 1. Note that the decoder network is symmetrical to the encoder.

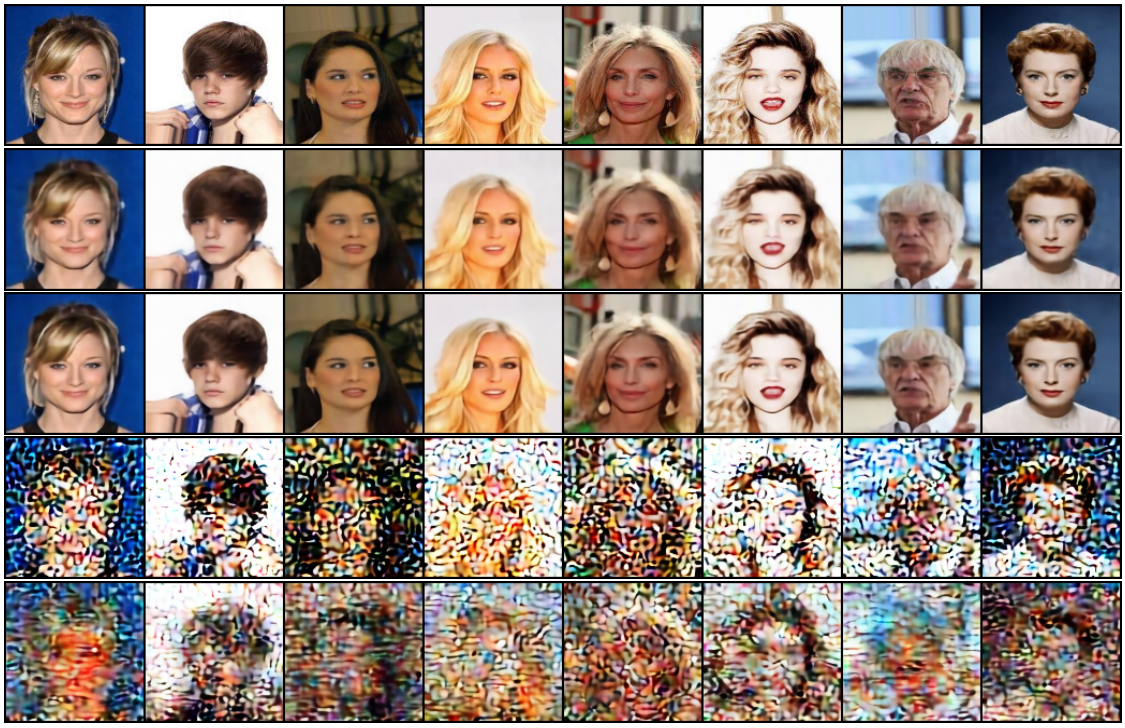

Fig. 3: Visual performance on images from the test database. First row: original images — second row: reconstructed ($k = 64$) — third row: reconstructed ($k = 128$) — forth row: reconstructed from ambiguated codes ($k' = 256$)— fifth row: reconstructed from random guessing of true codes (choosing $k = 128$ out of $k' = 256$).

As was expected, the random guessing does not improve the quality, since the chance of zeroing the noise is the same as that of the original content.

As a quantitative comparison and in order to measure the fidelity of the setup for both authorized and public domains, we calculate PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) of different experiments in comparison with the original image, as summarized in Table 1. As can be seen, the rate-distortion performance of the network is very high, even without lossless entropy coding, and is noticeably surpassing JPEG. This has two reasons, firstly, because of adapting to the content and hence better capturing redundancies than fixed codecs, secondly, the fact that we only encode the support and not the non-zero values. However, these values are not highly entropic and can be quantized, hence surpassing JPEG even by providing privacy utility.

	Recon. ($k = 64$) rate = 0.0845	Recon. ($k = 128$) rate = 0.1690	JPEG rate = 0.1830	ambiguated ($k' = 256$)	rand. guess ($k' = 256$)
PSNR	28.66	30.75	22.40	12.00	12.31
SSIM	0.92	0.95	0.76	0.24	0.25

Table 1: Rate vs. image quality measures (Results averaged over 200 randomly-selected images from the test set.)

5. CONCLUSIONS

We introduced a practical data sharing scheme for privacy-aware image sharing suitable for large-scale setups, where compactness of representations is of important concern for secure communication and storage. This was achieved by ambiguating code-maps of sparse representations, for which we designed a deep CNN as a bottlenecked autoencoder and learned it end-to-end on a large-scale image database.

REFERENCES

[1] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey, "Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain," *IEEE Signal Processing Magazine*, pp. 108–117, 2013.

[2] C. Dwork, "Differential privacy: A survey of results," in *Int. conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[3] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, "Context-aware generative adversarial privacy," *Entropy*, vol. 19, pp. 656, 2017.

[4] A. Tripathy, Y. Wang, and P. Ishwar, "Privacy-preserving adversarial networks," *arXiv preprint arXiv:1712.07008*, 2017.

[5] M. Gheisari, T. Furon, L. Amsaleg, B. Razeghi, and S. Voloshynovskiy, "Aggregation and embedding for group membership verification," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[6] S. Hill, Z. Zhou, L. Saul, and H. Shacham, "On the (in) effectiveness of mosaicing and blurring as tools for document redaction," *Proceedings on Privacy Enhancing Technologies*, , no. 4, pp. 403–417, 2016.

[7] YouTube Official Blog, "Face blurring: when footage requires anonymity," *Blog.(18 July 2012). Retrieved April*, vol. 13, 2012.

[8] M-R. Ra, R. Govindan, and A. Ortega, "P3: Toward privacy-preserving photo sharing," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 515–528.

[9] R. McPherson, R. Shokri, and V. Shmatikov, "Defeating image obfuscation with deep learning," *arXiv preprint arXiv:1609.00408*, 2016.

[10] B. Razeghi, S. Voloshynovskiy, D. Kostadinov, and O. Taran, "Privacy preserving identification using sparse approximation with ambiguization," in *IEEE Int. Workshop on Information Forensics and Security (WIFS)*, Rennes, France, 2017, pp. 1–6.

[11] S. Rezaeifar, B. Razeghi, O. Taran, T. Holotyak, and S. Voloshynovskiy, "Reconstruction of privacy-sensitive data from protected templates," in *IEEE Int. Conf. on Image Processing (ICIP)*, September 2019.

[12] T. Chuman, W. Sirichotedumrong, and H. Kiya, "Encryption-then-compression systems using grayscale-based image encryption for jpeg images," *IEEE Transactions on Information Forensics and Security*, pp. 1515–1525, June 2019.

[13] O. Watanabe, A. Uchida, T. Fukuhara, and H. Kiya, "An encryption-then-compression system for jpeg 2000 standard," in *2015 IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[14] P. Korshunov and T. Ebrahimi, "Scrambling-based tool for secure protection of jpeg images," in *IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 3423–3425.

[15] O. Watanabe, A. Nakazaki, and H. Kiya, "A fast image-scramble method using public-key encryption allowing backward compatibility with jpeg2000," in *Int. Conference on Image Processing, (ICIP)*, Oct 2004.

[16] J. Ballé, V.o Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[17] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, "Generative adversarial networks for extreme learned image compression," *arXiv preprint arXiv:1804.02958*, 2018.

[18] S. Ferdowsi, *Learning to compress and search visual data in large-scale systems*, Ph.D. thesis, 12/11 2018, ID: unige:114990.

[19] B. Razeghi and S. Voloshynovskiy, "Privacy-preserving outsourced media search using secure sparse ternary codes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Alberta, Canada, April 2018, pp. 1–5.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[21] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, 2010, pp. 399–406.

[22] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang, "Maximal sparsity with deep networks?," in *Advances in Neural Information Processing Systems*, 2016, pp. 4340–4348.

[23] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *International Conference on Computer Vision (ICCV)*, December 2015.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[25] D. P Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.