Source Generator Attribution via Inversion

Michael Albright and Scott McCloskey
Honeywell ACST
1985 Douglas Drive North, Golden Valley, MN, USA
`firstname.lastname@honeywell.com`

Abstract

With advances in Generative Adversarial Networks (GANs) leading to dramatically-improved synthetic images and video, there is an increased need for algorithms which extend traditional forensics to this new category of imagery. While GANs have been shown to be helpful in a number of computer vision applications, there are other problematic uses such as 'deep fakes' which necessitate such forensics. Source camera attribution algorithms using various cues have addressed this need for imagery captured by a camera, but there are fewer options for synthetic imagery. We address the problem of attributing a synthetic image to a specific generator in a white box setting, by inverting the process of generation. This enables us to simultaneously determine whether the generator produced the image and recover an input which produces a close match to the synthetic image.

1. Introduction

Because of its use in 'fake news' and 'revenge porn', the implications of fully- or partially-synthetic imagery has recently become a matter of broad social concern. Underlying both of these is the technology of deep networks used to generate imagery, often of faces, that are increasingly realistic. Whereas traditional image forensics already include powerful techniques applicable to images captured with a wide range of cameras [15, 11, 3, 4, 9], there are relatively fewer options available to forensic analysts operating on synthetic imagery.

We aim to close the capability gap between source *camera* attribution and source *generator* attribution, and to provide additional functionality for generator attribution. Source camera attribution methods vary considerably, but typically use low-level cues (such sensor non-uniformity) and statistical indicators; they are unable to re-create the image capture process, because the physical camera may be unattainable and the scene being photographed may be ephemeral. Neither of these restrictions apply to
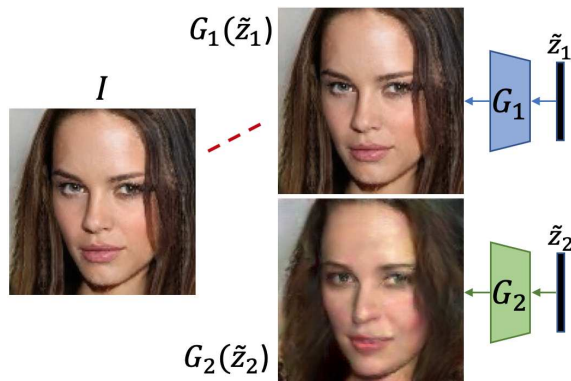

Figure 1. Image attribution by generator inversion: Given a synthetic probe image I and pre-trained generators G_i, the generators are inverted to find latent vectors $\tilde{z}_i$ such that generated outputs $G_i(\tilde{z})$ approximate the probe. Attribution is assigned to the generator with the best reconstruction (red dashed line).

synthetically-generated imagery, so we expand the attribution problem to encompass both the determination of whether the generator produced the image *and* the inputs necessary to re-create the generation process. Mathematically, we describe the generator as a function G which transforms a vector z into an image I_g as

$$I_g = G(z). \tag{1}$$

When there are multiple (known) generators G_1, G_2, etc. the **limited attribution problem** is to determine for a 'probe' image I the value of i that - for some value z - satisfies $G_i(z) = \tilde{I} \approx I$. The related **inversion problem**, given a specific generator G and probe image I, is to estimate a latent vector $\tilde{z}$ such that $G(\tilde{z}) = \tilde{I} \approx I$. In both, we use approximation to acknowledge that small differences should be expected due to quantization, dynamic range clipping, and perhaps compression applied to I.

In our **extended attribution problem** (Fig. 1), we consider the case where there are multiple (known) generators and, given a probe image, we determine *both* i and $\tilde{z}$ such that $G_i(\tilde{z}) = \tilde{I} \approx I$. The utility of attribution, in the context

of synthetically generated imagery, is that training GAN-based generators for high-resolution outputs is compute-intensive, data-intensive, and error-prone. As such, the majority of users are more likely to download and use publicly-available generators than they are to train their own from the ground up. We also believe that, beyond limited attribution, the ability to re-create the generation process (enabled by our estimation of z) is critical to improving the explainability of the attribution decision.

In our experiments, we demonstrate extended attribution on generators from two different domains, trained via two different methods. We first show that the generative layers of an auto-encoder, trained on MNIST [12] characters, can successfully be attributed in increasingly difficult scenarios, up to and including discrimination between generators trained on the same data, in the same order, but with different random seeds. We then show that generators trained to produce realistic facial images in a GAN can be attributed, and discuss some of the interesting similarities and differences between the probe and reconstructed image.

2. Related Work

Both the limited attribution and inversion problems described in the introduction have attracted attention in recent years. For attribution, Yu *et al*. [22] and Marra *et al*. [18] have recently presented methods which use noise-type signatures to attribute a given image to a specific generator, which is conceptually similar to how Photo Response Non-Uniformity (PRNU) [15] provides source *camera* identification. These methods are very successful, providing nearly perfect attribution performance, even being able to discriminate between two generators with the same architecture, having been trained on the same data. A related problem is the detection of whether a given image was generated by *any* GAN-type generator, which has successfully been demonstrated by several groups using various methods [17, 20, 7, 6, 13, 19].

The generator inversion problem has likewise attracted attention. Creswell and Bharath [5] develop a generator inversion method as a means to explore GAN performance as a function of attributes in the image space. Luo *et al*. [16] solve inversion by training an encoder coupled to a pre-trained generator in an auto-encoder framework. Lipton and Tripathi [14] introduce a method called stochastic clipping to recover arbitrarily precise approximations of z, even in the presence of simple types of noise. None of the prior inversion work has addressed attribution.

Another related work, by Kilcher *et al*. [10], demonstrates that generators instantiated with random weights can produce blurry versions of real images from various benchmark datasets. This demonstrates that the convergence of inversion, by itself, does not signal that the image can be attributed to a generator.

3. Method

Our method for attributing a probe image I to a generator G_i is predicated on the assumption that other generators G_j ($j \neq i$) cannot generate I as well as the true generator G_i, so the minimum reconstruction error will correspond to the true generator i. We offer a brief discussion on why one may reasonably expect that situation to often (though not always) hold: First, we note that generators map a low dimensional latent vector z into very high dimensional output space $G_j(z)$. For instance, in the case of ProGAN [8], dim(z) = 512 and dim(I) = 3,145,728 (3x1024x1024). Hence, the mapping from z to $G_j(z)$ parameterizes a union of low dimensional manifolds (of dimension $\leq$ dim(z)) in the high dimensional space [1]. Furthermore, there are a number of sources of inherent randomness in generator training, such as random initialization of network weights, ordering of training images, and absence or inclusion of particular images in training batches. Arjovksy and Bottou [1] have shown that it is highly improbable for low-dimensional manifolds in high-dimensional spaces to perfectly overlap everywhere if subjected to random perturbations; this is in fact part of the motivation for using a Wasserstein metric in Wasserstein GANs [1, 2]. Furthermore, in the case of GANs, training failures such as *mode dropping* may make the generator incapable of generating some regions of the manifold of natural images.

In the extended attribution problem, illustrated in Figure 1, we start from a probe image I which was generated by one of several generators G_1, G_2, ..., G_n , where we assume we know both the architecture and weights of all generators. We do not know a-priori which generator G_i made the probe image, nor do we know the latent vector z, but we seek to determine them. Note that if we could perfectly identify z and i, we could (in the ideal case) perfectly recreate the probe image $I = G_i(z)$. In practice, there may be some residual discrepancies due to post-processing of the generated image, e.g. quantization, clipping, image compression, etc., so we allow for for some small differences between the probe image and the generator output. Hence, we seek to estimate a latent vector $\tilde{z}$ such that $G_i(\tilde{z}) = \tilde{I}$ is as close as possible to the true probe I. For our experiments, we formalize generator inversion as an optimization problem, where we minimize the loss function

$$L_j(z) = \frac{1}{MN}||I - G_j(z)||^2, \qquad (2)$$

with N and M being the number of pixels and color channels in the image, respectively.

We separately minimize Eq. 2 for each generator j, using an optimization algorithm to find the best latent vector z which minimizes the loss function. Attribution is assigned to the generator i with the smallest residual error, and the

estimated latent vector $\tilde{z}$ is the point of minimum loss

$$\tilde{z} = \operatorname*{argmin}_{z} \frac{1}{MN} ||I - G_i(z)||^2 \,. \qquad (3)$$

Because generators used in GANs and autoencoders are neural networks, they are readily differentiable via back-propagation, so the loss function in Eq. 2 may be optimized efficiently using gradient-based methods; in this work, we found the Adam optimization algorithm to perform well for all tested generators. Because Eq. 2 is not convex, there is no guarantee that the optimization will converge to a global minimum. Hence, to ensure low reconstruction errors are obtained, we perform a multi-start optimization, where we perform each optimization multiple times, each starting from different random initial starting guesses for the latent vector z, and we chose the result with the lowest residual error.

We note that since the magnitude of the minimum residual error $L_i^{\min} = L_i(\tilde{z})$ quantifies how well the generated output $G_i(\tilde{z}) = \tilde{I}$ matches the probe image I, one may use the residual errors to assess confidence in the attribution assignment: for a correct assignment, one would expect the minimum loss $L_i^{\min}$ to be very small, and for high confidence it should be significantly smaller than reconstruction errors $L_j^{\min}$ from other generators $j \neq i$.

In the special case where there are only two generators i and j, we can summarize our attribution decision and confidence by a single numerical score:

$$S = \frac{L_j^{\min} - L_i^{\min}}{L_j^{\min} + L_i^{\min}} \,. \qquad (4)$$

Note that $S \to 1$ when the ith generator perfectly reconstructs the probe ($L_i^{\min} \to 0$), $S \to -1$ when the jth generator perfectly reconstructs the probe ($L_j^{\min} \to 0$), and $S \to 0$ when both generators reconstruct the probe equally well ($L_i^{\min} = L_j^{\min}$; no attribution possible). Hence, the score S provides a natural way to evaluate attribution performance using Receiver Operating Characteristic (ROC) curves, effectively treating binary attribution as binary classification; we do so in Section 4.1.

In the next section, we show that differences in generator training indeed contribute to detectable differences in generated images, even in the case of identical generator architectures.

4. Experiments

In this section, we describe experiments by which we demonstrate the utility of our generator attribution and inversion. They present a series of increasingly difficult attribution problems, distinguishing between pairs of generators with increasing similarity in how they are trained.

4.1. MNIST Experiments

The first set of experiments were carried out with the MNIST dataset, selected because the smaller image size allows for faster training and testing. In each MNIST experiment, we trained a pair of fully-connected auto-encoders with sigmoid activations and an L2 loss. The encoder and decoder parts are symmetric, each having 784 nodes at the input/output layers and hidden layers with 64 and 32 nodes each. Each auto-encoder was trained on approximately 30,000 MNIST digits for 20,000 - 60,000 steps (depending on the experiment) in batches of 256 images using the Adam optimization algorithm with a learning rate of 0.01.

Once the training was completed, the decoders were separated from their encoders and the weights were frozen, rendering them pre-trained generators which map 32 dimensional latent vectors into 784 dimensional outputs. We then performed optimization-based generator inversion for image attribution. Due to the ease and success of the optimization-based inversion, multi-start was not employed for the MNIST experiments. For testing, we used 1000 digits generated from each generator and saved as a PNG. We summarize the performance as a binary classification problem and use Receiver Operating Characteristic (ROC) curves and use Area Under the Curve (AUC) as a summary performance statistic. We arbitrarily designated one of the generators as the target generator and measure the True Positive Rate (TPR) at which outputs from the target generator are classified as such. The False Positive Rate (FPR) measures the frequency with which non-target generator outputs are classified as target outputs. These values are computed over a range of thresholds on S (as defined in Eq. 4).

4.1.1 Non-overlapping Training Data

In this experiment, we trained the auto-encoders on two non-overlapping subsets of the MNIST digits using 20,000 optimization steps. For simplicity, we train one using odd digits and the other with even digits, though we note that our attribution does *not* recognize the digit or its parity. Having thus trained even and odd digit generators, we attribute each of the test images by the method descried above. Figure 2 shows that the performance of our attribution on this experiment is nearly perfect, despite the fact that the generators do a surprisingly good job synthesizing digits that they've never seen before. The bottom row of Figure 2, for example, shows a '9' synthesized by the odd generator (left column) and reconstructions from inverting both even and odd generators (center and right columns, respectively). Despite never having been trained on '9's, the even number generator can produce a good approximation of this input, though less so for the odd generator and the '2' in the row above.
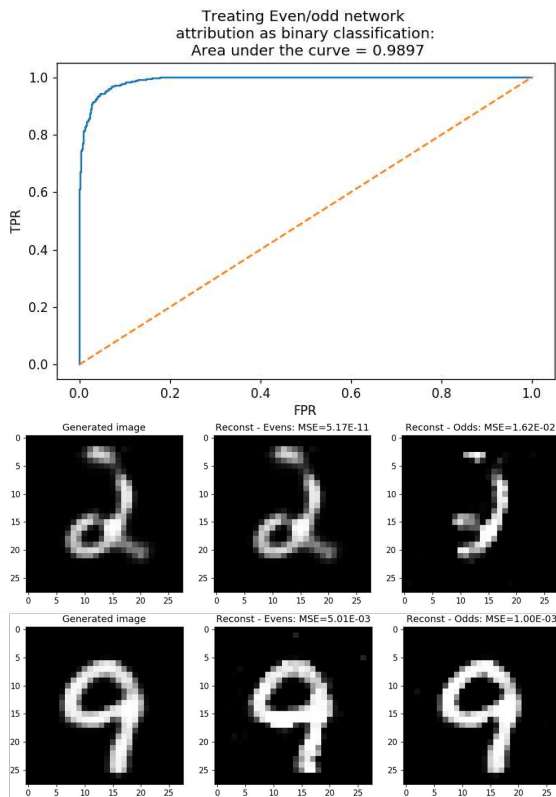
Figure 2. (Top) ROC curve on attribution between two generators with non-overlapping training data. One generator is trained on only even MNIST digits, and the other on odd digits. For reference, the diagonal line shows the performance of random attribution. (Bottom rows) Examples of two successful attributions. The probe image (left) was approximated with the even (middle) and odd (right) generators via inversion.
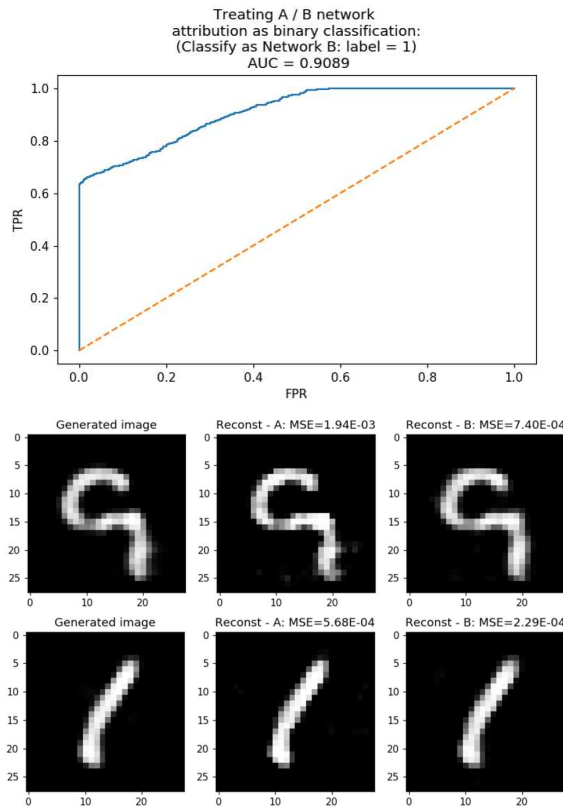

Figure 3. (Top) ROC curve on attribution between two generators with different permutations of the same training data. (Bottom rows) Examples of two *unsuccessful* attributions.

4.1.2 Same Training Data, Different Order

In this experiment, we trained the auto-encoders on the same set of MNIST digits, but shuffled the order in which they're used in training. Both generators have the same architecture, but they have different random initializations. We again use 20,000 optimization steps for training. Figure 3 shows that the performance is reduced when our algorithm is asked to differentiate between generators trained with the same training data, but is still quite good. The figure also shows two examples of *mis*-attribution, illustrating that the outputs of the two generators are quite similar, visually, and have reconstruction errors which are quite close to one another. This portends difficulty in successfully attributing images subject to the normal forensic challenges such as compression and re-encoding.

4.1.3 Same Data, Same Ordering

In the last of our MNIST experiments, the two auto-encoders are trained on the same subset of MNIST digits, and those training digits are presented in the same order. The only difference between the two generators is the initial (random) weights; we also increased the number of training steps to 60,000 to be sure the training converged. Despite this high level of similarity, Fig. 4 shows that our approach can unambiguously attribute about 55% of the inputs with extremely high confidence.

4.2. CelebA Experiments

In the second set of experiments, we performed attribution on generators that were trained as Generative Adversarial Networks (GANs). The two GANs tested were Pro-GAN [8] and SAGAN [23], both of which were modified from their original description to output 128x128 resolution images and were trained on face images from the CelebA dataset. We re-used pre-trained generator weights shared by Ning Yu [22]. The SAGAN generator proved easy to invert via optimization algorithms, so multi-start was not strictly necessary, but for good measure we employed 10
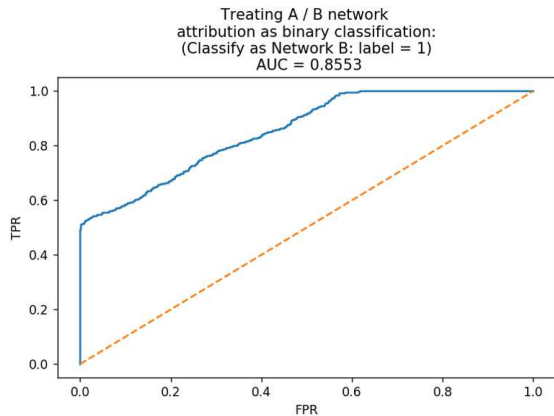
Figure 4. ROC curve on attribution between two generators with the same training data, in the same order. Only the random initialization of the network differs between the two generators, despite which we can attribute their outputs well.
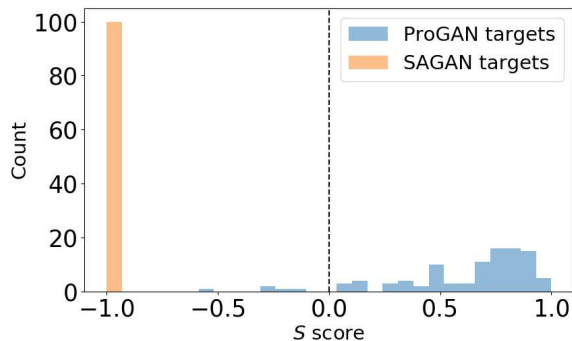

Figure 5. Histogram of S scores for probe images from ProGAN and SAGAN. Note that inversion produces a nearly exact reconstruction of the probe for SAGAN, but a rougher approximation for ProGAN.

Source	ProGAN	SAGAN
Accuracy	95%	100%

Table 1. Inversion-based attribution accuracy, by data source, measured on a dataset of 200 synthetic images—100 generated by ProGAN and 100 by SAGAN.
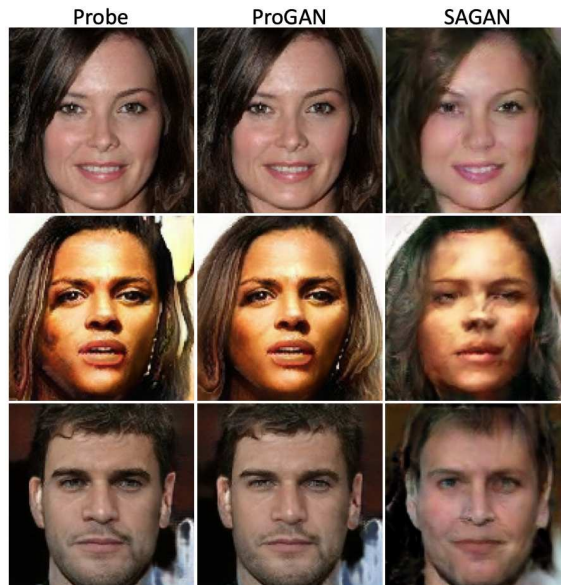

Figure 6. Examples of correct attribution of probe images generated by ProGAN. The left column shows probe images while the middle and right columns show the closest approximations found by inverting ProGAN and SAGAN generators. Note the subtle differences in the 2nd row between the probe and ProGAN reconstruction.

random initial optimization starts per image. For each random start, we used the Adam optimization algorithm with a learning rate of 0.1 and 1000 optimization steps per image, and chose the best reconstruction with the lowest loss encountered. We found it helpful to employ an explicit learning rate reduction on plateau, to reduce oscillation in the loss as the optimizer approached the minimum; we used a learning rate shrink factor of 0.5 and a patience of 30. The ProGAN generator proved slightly more challenging to invert reliably, so we employed 20 random initial optimization starts per image. We also used the Adam optimizer, but with a learning rate of 0.9, 300 optimization steps per image, and no explicit learning rate reduction (besides those built into Adam).

We tested inversion-based attribution on a dataset of 200 synthetic images, 100 of which were generated by ProGAN and 100 by SAGAN, which were saved as PNGs. As summarized in Table 1 and the histogram of scores in Fig.

5, attribution accuracy was 100% for images generated by SAGAN and 95% for images generated by ProGAN, with the attribution threshold corresponding to $S = 0$. Examples of successful attributions of images generated by ProGAN and SAGAN are shown in Figures 6 and 7. As can be seen in Figures 6 and 7, successful reconstruction of the probe image by a generator offers compelling, interpretable evidence that a probe image was created by the indicated generator. This ability to offer strong, interpretable evidence of a synthetic image's source is unique to our method of generator attribution, compared to other attribution methods which tend to operate as black boxes without strong interpretability or confidence measures. In Figure 8, we show three examples where attribution of images generated by ProGAN failed. From the middle column of the figure, it is clear that the cause of the failure is that the optimization algorithm failed to converge to the best reconstruction, causing mis-attribution. In each of the three cases, it is obvious that the quality of the best image reconstruction is poor, with relatively large residual error.
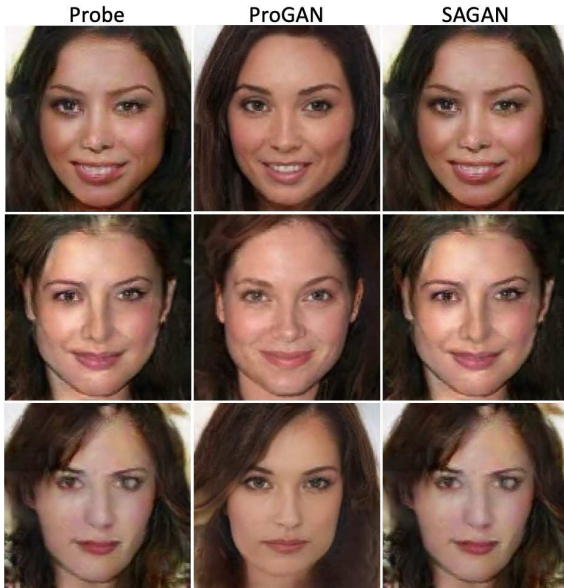
Figure 7. Examples of correct attribution of probe images generated by SAGAN. The left column shows probe images while the middle and right columns show the closest approximations found by inverting ProGAN and SAGAN generators.
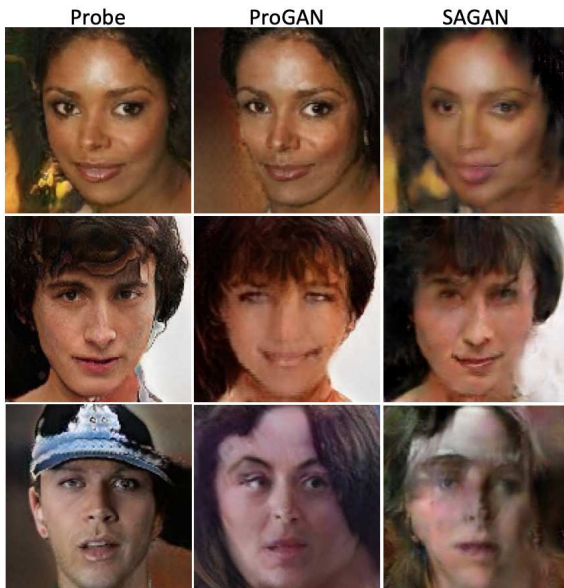

Figure 8. Examples of incorrect attribution of probe images generated by ProGAN. In each case, mis-attribution is caused by failures of the inversion, evident in the background in the first row, the face in the second row, and everything in the third.

In addition to the above work, we experimented with minimizing $L1$ loss functions, on the hope that it would encourage a sparser residual error vector $r_i = I - G_i(\tilde{z})$, but we saw no significant attribution benefit. We also tested

Figure 9. Example inversion with semantically meaningful differences. In this case, the reconstruction (right) has lips that are slightly more closed and a smaller reflection in the hair (upper left) compared to the probe image (left). Note that this inversion was performed against the full-resolution (1024x1024) version of the CelebA-HQ ProGAN, whereas inversions in Sec. 4.2 were done against a lower-resolution version.

the LBFGS optimization algorithm, which uses 2nd-order derivatives to accelerate convergence; similar to [21], we found that significantly fewer optimization steps were required to invert an image, but we saw no significant benefit in the minimum loss obtained by the optimizer.

5. Discussion

We have shown that generator inversion is a useful means by which to pursue synthetic image attribution, with the added advantage of being able to re-create the generation process by estimating the latent vector via inversion. Despite this increased functionality, we achieve similar results to black box systems such as Yu *et al.* [22]. Both our method and Yu's are able to distinguish between generators that are quite similar, up to and including generators trained from the same data in the same order.

While our attribution results correspond to a completely automated system, it is interesting to consider how our output image would be used and interpreted by a forensic analyst. As illustrated by Fig. 8, inversion failures are visually salient and so the assigned attributions could be rejected by the analyst for failing to meet quality thresholds. We note, however, that designing such a threshold is non-trivial in light of semantically meaningful differences between the probe and reconstruction in the case of successful inversion, an example of which is illustrated in Fig. 9.

6. Future Work

Though our experiments have so far been limited to attribution between two generators, it can easily be extended to N-way attribution with a modification of eq. 4; note that the dynamic range would need to be normalized for networks having different outputs, i.e. different than the $[-1, 1]$ range used by ProGAN and SAGAN. However, the computational complexity of inversion over N generators

can be quite high. Given the computational complexity of inverting the generator, a more efficient system could use an attribution-only method, i.e. one of [22, 18], to determine the source generator and then apply inversion on only that generator. Another extension would be to train neural network encoders to explicitly perform the inverse mapping from an image to a latent vector, which could also be further fine-tuned by optimization, as in [24].

One area of future work would be to extend our attribution capabilities into the black box domain, in order to handle generators which may be offered online as a service, but without published network weights. It has been shown that, for deep networks performing classification tasks, representative proxy networks can be trained based on a relatively small number of input/output pairs from the target network. It may be the case that we can develop proxies for target *generators* in order to support our white-box attribution method.

Acknowledgements

References

[1] Martin Arjovsky and Lon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862 [cs, stat]*, Jan. 2017. arXiv: 1701.04862. 2

[2] Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, Jan. 2017. arXiv: 1701.07875. 2

[3] M. Chen, J. Fridrich, M. Goljan, and J. Lukas. Source digital camcorder identification using sensor photo response non-uniformity. *Proceedings of the SPIE*, 2007. 1

[4] K. S. Choi, E. Y. Lam, and K Wong. Source camera identification using footprints from lens aberration. *Proceedings of the SPIE*, 2006. 1

[5] Antonia Creswell and Anil A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2018. 2

[6] D. Guera and E. J. Delp. Deepfake video detection using recurrent neural networks. In *IEEE International Conference on Advanced Video and Signal-based Surveillance (to appear)*, 2018. 2

[7] C.-C. Hsu, C.-Y. Lee, and Y.-X. Zhuang. Learning to Detect Fake Face Images in the Wild. *ArXiv e-prints*, Sept. 2018. 2

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2, 4

[9] M. Kharrazi, H. T. Sencar, and N. Memon. Blind source camera identification. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, 2004. 1

[10] Yannic Kilcher, Aurlien Lucchi, and Thomas Hofmann. Generator Reversal. *arXiv:1707.09241 [cs, stat]*, July 2017. arXiv: 1707.09241. 2

[11] K. Kurusowa, K. Kuroki, and N. Saitoh. Ccd fingerprint method. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, 1999. 1

[12] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998. 2

[13] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *International Workshop on Information Forensics and Security*, 2018. 2

[14] Zachary C. Lipton and Subarna Tripathi. Precise Recovery of Latent Vectors from Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR) Workshop Track*, 2017. 2

[15] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *IEEE Trans. on Information Forensics and Security*, 1(2), 2006. 1, 2

[16] Junyu Luo, Yong Xu, Chenwei Tang, and Jiancheng Lv. Learning inverse mapping by autoencoder based generative adversarial nets. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, pages 207–216, Cham, 2017. Springer International Publishing. 2

[17] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 384–389, April 2018. 2

[18] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? *CoRR*, abs/1812.11842, 2018. 2, 7

[19] Scott McCloskey and Michael Albright. Detecting gan-generated imagery using color cues. *CoRR*, abs/1812.08247, 2018. 2

[20] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, B. S. Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H. Bappy, and Amit K. Roy-Chowdhury. Detecting GAN generated fake images using co-occurrence matrices. *CoRR*, abs/1903.06836, 2019. 2

[21] Ryan Webster, Julien Rabin, Loic Simon, and Frederic Jurie. Detecting overfitting of deep generative networks via latent recovery. *arXiv preprint arXiv:1901.03396*, 2019. 6

[22] Ning Yu, Larry Davis, and Mario Fritz. Attributing Fake Images to GANs: Analyzing Fingerprints in Generated Images. *arXiv:1811.08180 [cs]*, Nov. 2018. arXiv: 1811.08180. 2, 4, 6, 7

[23] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 4

[24] Jun-Yan Zhu, Philipp Krhenbhl, Eli Shechtman, and Alexei A. Efros. Generative Visual Manipulation on the Natural Image Manifold. *arXiv:1609.03552 [cs]*, Sept. 2016. arXiv: 1609.03552. 7