PhiNet-GAN: Bringing real-time face swapping to embedded devices

Alberto Ancilotto, Francesco Paissan, Elisabetta Farella
aancilotto@fbk.eu, fpaissan@fbk.eu, efarella@fbk.eu
E3DA Unit, Digital Society center, *Fondazione Bruno Kessler (FBK)*, Trento, Italy

***Abstract*—Recent years have seen an unprecedented development of deep learning-based techniques for processing live video from CCTV cameras, causing growing privacy concerns. A possible solution is to ensure that a subject's personal information never leaves the device in which it was collected, thus implementing a Privacy-by-Design (PbD) approach. In live video processing tasks, PbD can be guaranteed through anonymisation techniques, such as face-swapping, performed directly on the end device. This paper, therefore, presents PhiNet-GAN, an extension of the PhiNet family of embedded neural networks applied to generative networks. PhiNet-GAN targets resource-constrained platforms based on low-power microcontrollers. An example is the Kendryte K210, a RISC V dual-core processing unit working at 400MHz on which we tested our network. Overall we achieved a power consumption of less than 300mW, working at more than 15fps with an FID score lower than 150.**

***Index Terms*—Faceswap, Neural Networks, Edge AI, Tiny ML, Hardware-Aware Scaling, Generative Networks**

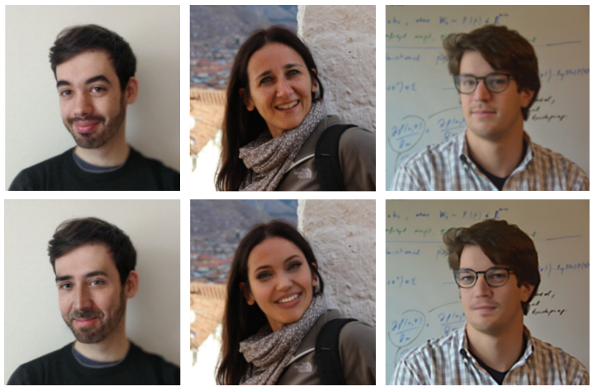
Fig. 1. These results are generated with an energy of $17.1mJ$ per frame. Comparable state-of-the-art methods running on GPUs can require $115J$ per frame, a difference of 4 orders of magnitude.

I. INTRODUCTION

In smart city scenarios, the primary users of data collected by distributed devices are public administrations (PAs) and law enforcement agencies (LEAs). The principal goal of PAs and LEAs regarding the data collected by urban ambient sensors, such as CCTV cameras and microphones, is to support decision-making and real-time intervention. In the last decades, the fast development of AI analytics created space for innovation in the described scenarios of interest. As most deep learning pipelines are designed to work on clusters of GPU-equipped workstations, streaming data from sensors to these high-end devices is crucial. However, data recorded in public spaces contain sensitive personal information - such as citizens' biometric data - that might be stolen in case of man-in-the-middle attacks. TinyML is a growing field focusing on bringing high-performance processing to low-power, low-resource end devices. One of the main benefits is that TinyML enables privacy-by-design (PbD). Source data, potentially including sensitive information, are not sent or stored but consumed close to where they are collected to produce an event, signature or feature. Even in the case of cyber-attacks, the original data cannot be retrieved. Only higher-level information, usually with a less sensitive nature, is maintained and/or sent (e.g. in the case of people counting, only a number will be stored or transmitted, not the entire frame or sequence of frames). We consider the case study of the EU project "MARVEL: Multimodal Extreme Scale Data Analytics for Smart Cities Environments" [1]. In MARVEL's applications, video streaming is required for high-end processing by different components since high-level features extracted at the edge do not suffice. Therefore, to protect citizens' privacy without affecting the components' performance, we anonymise target features at the edge, i.e. by face swapping.

In the described scenario, the challenges are multiple and can be clustered into two categories. The first category is typical of TinyML, i.e. reducing algorithm complexity and energy consumption while preserving performance. The second is specific to executing Generative Adversarial Networks (GANs) on resource-constrained edge devices. For example, existing toolchains provide limited support for generative networks. Furthermore, working with low-resource devices sometimes involves low sensing resolutions, imposing higher challenges to preserving sufficient information of interest for PAs and LEAs.

This paper presents PhiNet-GAN, an innovative face-swapping approach explicitly developed for edge devices with low computational capabilities. Furthermore, we will analyse the effects of quantisation on the network to achieve an anonymisation pipeline that enables high performance on embedded devices.

This work was partially funded by the EU H2020 project MARVEL (project number: 957337).

II. Related works

Traditional video anonymization techniques are generally based on blurring or pixelization, making subsequent image processing (e.g. face detection, action recognition, emotion classification) impossible due to corruption of the information in the frame. Recently, there has been a rapid increase in the development of face-swapping methods, targeting the movies, gaming, and entertainment industries [2], and focusing on designing approaches that can be utilized on consumer hardware without the need for high-end equipment. These techniques have been adapted for privacy protection, using face-swapping as an anonymisation technique aimed at removing only personal information from the frame but not alternating the content significantly [3]–[8]. Among the first approaches used are inpainting techniques applied to people's faces [9] that maintain the possibility of face identification in the frame, although still destroying information about pose and expression. More recent approaches based on image-to-image translation [10] can maintain unchanged the pose and expression of a target face. One of the earliest examples of this application is DeepPrivacy [3], which relies on StyleGAN [11] to generate lifelike images. CIAGAN [4], more recently, exploits a discriminator network-based architecture to force the generation of images with a lower rate of target re-identifications. Approaches such as SimSwap [12], FaceShifter [13], Ghost [14], FaceController [15], and HiFiFace [16] aim at the generalization of high-fidelity face swapping approaches by allowing the injection of different identities at runtime, without them being encoded during the training process but requiring architectures that are more complex than what can be run on embedded devices. With recent developments in tinyML techniques for generating networks with reduced computational requirements [17]–[23], it is increasingly possible to bring processing pipelines to edge devices. In this work, we focus on designing and implementing a face-swapping network suitable for very low-power embedded devices and microcontrollers, leveraging the tinyML innovations introduced with PhiNets [17], i.e. a hardware-aware scaling approach, and extending it to implement a face-swapping pipeline.

III. Methodology

Existing face-swapping approaches rannge in complexity and performance, from very lightweight techniques such as Mobile Faceswap [24], to more complex, 3D based methods and frameworks employing multiple networks for face generation and seemingly merging source attributes on a target face like FaceShifter [13]. Most of these approaches are based on a many-to-many swapping paradigm, where the source face features are encoded in a latent vector and then used by a second network to generate the target facial features. In this section, we will analyse the limitations of current anonymisation approaches when applied on embedded devices and propose a methodology to tackle those challenges.

A. Challenges of a TinyML generative pipeline

Platform characteristics and runtime limitations may make it impractical, if not impossible, to execute state-of-the-art approaches on microcontrollers. Merging a low-resolution, high filter count latent representation of the source image during the inference pass of a generative network poses two challenges for tinyML:

- It requires a large amount of RAM to store both the generative network intermediate tensors and the latent vector containing the source image features
- It needs a tinyML runtime capable of receiving different kinds of network inputs at different times during the computation (i.e. the original three-dimensional image and the one-dimensional latent representation).

For this reason, instead, we chose to base our framework on a one-to-one face-swapping approach. This has the great advantage that a single input encoder-decoder architecture transforms the input face using features from the source identity, where these are already known by the network without the need for additional input vectors, thus significantly improving the compatibility of our approach to all runtimes that can execute basic convolutional operations and upsampling.

B. A strategy towards lightweight architectures

We base our solution on the original Deepfakes framework [25], extending and improving it to achieve a very lightweight, many-to-one face-swapping framework fit for tiny embedded platforms. For this purpose, we extend the method by modifying the training procedure, implementing improvements presented in later works (such as the addition of adversarial and perceptual loss [12], [26]), and, most importantly, replacing the original face-swapping network (pix2pix [27]) with one designed ad-hoc for tinyML applications. To design the network, we start from the Phinets architectures family [17], by adding an upsampling convolutional block. The newly introduced block respects the scaling principles needed for Hardware Aware Scaling (HAS) principle, a novel paradigm that enables the one-shot design of neural architectures for our target edge platform thanks to the PhiNets' advanced scalability features [17]. With this, we only have to know three hardware specifications, namely RAM, FLASH, and the number of operations per second of the target platform, to design, in one shot, an optimal neural architecture that is suitable for our task and platform and matches the requirements.

IV. From Deepfake to Phinets-GAN

As mentioned, we start our design from DeepFakes [25]. The network architecture used in the original framework consists of 2 image-to-image translation networks (adapted from CycleGAN [10]) sharing a common encoder stage, and two separate decoders, that are trained one on images of the source identity, and one on images of the target one. During training, warped images of the source and target faces are sent to the respective network, which has the task of unwarping the pictures into the original photo. During inference, images of the target face are sent to the source network, which interprets them as warped source images reconstructing source identity facial features on the target image.

A. *Many-to-one face swapping*

While simple, this approach has the obvious disadvantage of only allowing for single-target-to-single-source swapping. To overcome this limitation, we modify the training procedure to obtain a network capable of many-to-one swapping. Specifically, we keep the encoder-source decoder operation the same during training, while the encoder-target decoder architecture will be trained on different identity images (vggface dataset [28]). In this way, we get an encoder with a higher generalization capability in encoding a generic identity and a specific decoder for our source identity. This could lead to mode collapse when training the second decoder, but in our case, this is not a problem as that network will get discarded and not be used for inference.

B. *Fully convolutional network*

The original DeepFake implementation relies on downsampling blocks to reduce the input resolution down to a 2×2 feature map, then using a reshape operation and fully connected layers to generate the latent feature vector that will be sent to the decoder network. This allows for a global receptive field in the architecture, where each value in the latent feature map is determined by all the pixels in the input image. This, however, is the cause of the two main limitations of the approach: the high number of parameters and operations used by the CycleGAN network.

Moreover, bringing this type of processing to embedded devices proves to be very inefficient, especially in the case of hardware platforms that include Tensor Processing Units, such as the chosen target platform for the tests, the Kendryte K210. The efficiency in computing convolutional layers is proportional to the input size of the layer itself: while on large feature maps, the convolutional weights can be loaded once in memory and reused many times, for very small tensors, these weights are only used a couple of times then discarded. Coupled with the fact that the lower resolution feature maps are usually the ones containing the highest number of channels, this leads to high inefficiencies in executing these networks.

TABLE I
EFFECTS OF DIFFERENT RESOLUTIONS FOR LATENT REPRESENTATION ON NETWORK PERFORMANCE METRICS

Latent resolution	**Params [M]**	**Latency [ms]**	**FID $\downarrow$**
1×1	1.80	125	152
2×2	1.59	104	164
4×4	1.48	62	173

Table I shows a comparison between three similar networks, where the number of convolutional blocks has been increased in order to change the latent resolution at the smallest network tensor. At the same time, the number of filters in the networks was reduced, so that the networks required a similar amount of operations to run. This experiment highlights the relationship between latent resolution, network latency, and performance, showing how a smaller resolution of the latent tensor allows for a larger receptive field and this, in turn, provides higher performance evaluated in terms of Fréchet Inception Distance (FID) [29]. It can also be seen that this directly affects the number of network parameters, as layers with lower spatial resolution make use of more feature maps.

This shows how different network configurations can be developed, either using the smallest latent resolution to provide the best performance despite the decrease in efficiency, or more efficient options where we limit the number of downsampling and upsampling layers for better throughput.

C. *Increasing generative quality*

More recent works [12], [26], [30] have introduced some training modifications to the original framework, aimed at improving its performance. For our work, we extend the original approach by adding:

- An adversarial loss term, given by a patchGAN [31] network trained to discern between real and fake images, to allow generating more true-to-life images
- A perceptual loss term [32] given by the distance between feature vectors generated by VGG-Face between a real and a fake face, to increase the quality of generated faces
- An additional reconstruction loss term in the eyes area to generate more realistic eyes movements, introduced in [26]

These additions lead to a high increase in generated face quality, which can counteract the performance penalty due to the network capacity reduction and the quantisation needed to make the approach suitable for MCUs.

D. *Upsampling block design*

We propose an efficient upsampling block (Figure 2) that is based on a sequence of a 1×1 and a 3×3 convolution, with an interpolation operation between the two. In particular

- The first pointwise convolution brings down the number of channels from C_{in} to C_{in}/E, where E (expansion factor) is a network hyperparameter ≥ 1 (default 3). It reduces the complexity of the interpolation and second convolution and allows the upsampling block to require less RAM for intermediate tensors.
- For the interpolation operation, different schemes can be used, such as nearest neighbour or bilinear interpolation. In our case, for our target platform, bilinear interpolation provided a better trade-off between latency and quality of the generated images.
- The third operation is the main convolution of the block. We chose a standard 2D convolution as, for our target platform and application, we can afford the increased number of parameters and increased latency given the higher quality of the generated faces. Nonetheless, a simpler depthwise operation can be used for even lower operation and parameter count architectures.

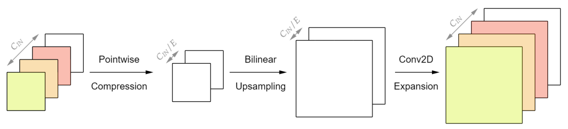

Fig. 2. Proposed upsampling block

1) Block hyperparameters: As in the case of the original Phinet convolutional block presented in [17], the proposed architecture presents 3 hyperparameters:

- Width scaling factor (α): it scales the number of filters linearly for all convolutional layers - e.g. going from $\alpha = 1$ to $\alpha = 0.5$ halves filter count for all layers, reducing parameter count and operations to one fourth of the original network.
- Base expansion factor (E_0): it is the ratio between input and output channels for the two block convolutions. Scaling E linearly changes the operation and parameter count for the network.
- Shape factor (β): it changes the block expansion factor linearly along the network, from $E = \beta E_0$ in the first block to $E = E_0$ on the last block. The expansion factor for the block at index B of a sequence of N blocks is computed as $E = E_0 \frac{(\beta-1)N+B}{B}$ to keep compatibility with the Hardware Aware Scaling approach [17].

As these hyperparameters control different aspects of the block topology, they can be tuned knowing the actual platform specifications using Hardware Aware Scaling [17].

2) E_0 vs α trade-offs: We analyzed, in particular, the trade-offs that can be achieved by varying α and E_0 at the same time since these affect the network similarly. It results in networks with the same (or very similar) number of parameters and operations. In particular, the block operations are given by the sum of the pointwise and 2D convolutions in the block (with $W \times H \times C$ feature map size, K kernel size):

$$\begin{aligned} MACC &= WH \times \alpha C_{in} \frac{\alpha C_{in}}{E} \\ &+ 2W2HK^2 \times \frac{\alpha C_{in}}{E} \times \alpha C_{out} \\ &\propto \frac{\alpha^2}{E} \end{aligned}$$

And similarly for the number of parameters. For this reason, we can vary alpha and E between two networks, A and B, so that $\frac{\alpha_A^2}{E_A} = \frac{\alpha_B^2}{E_B}$, achieving very similar operation and parameter count. Table II shows the result of these operations on network performance. Face generation performance of the different networks is given in terms of FID [29].

TABLE II
TRADE-OFFS BETWEEN DIFFERENT RATIOS OF E AND α ON NETWORK PERFORMANCE METRICS

E_0	α	**Params [M]**	**Latency [ms]** $\downarrow$	**FID** $\downarrow$
1	0.35	1.43	75.8	173
2	0.5	1.37	**51.3**	168
3	0.6	1.38	52.6	**157**
4	0.7	1.39	54.3	**157**

We can observe how, for $E > 1$, we gradually have a decrease in FID score by increasing E and increasing alpha, and an opposite behaviour regarding latency. As a trade-off, we will use by default a base expansion factor $E_0 = 3$ in the block. It should be noted that while parameters and operations remain unchanged, RAM usage decreases as E increases, so $E_0 = 4$ may prove to be a better choice for very constrained devices, at a small latency increase cost.

E. Network architecture

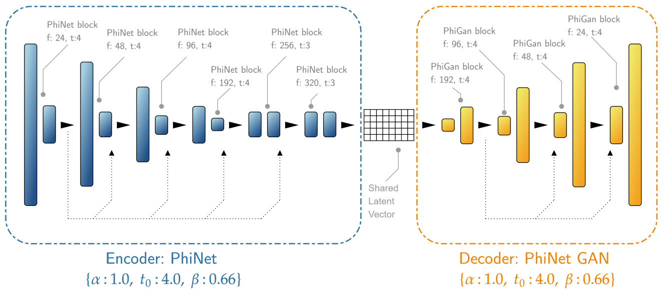

Fig. 3. The proposed network architecture

Figure 3 shows the proposed architecture. The network is an encoder-decoder structure where the encoder blocks are four standard *Phinet* [17] downsampling blocks, followed by a neck of two non-downsampling blocks that act on the low-resolution latent representation. The generative section of the network is composed of a sequence of four *Phinet-GAN* upsampling blocks described previously. The final architecture specifications are shown in Table III.

TABLE III
CHOSEN NETWORK HYPERPARAMETERS AND NECESSARY HARDWARE RESOURCES

E_0	α	**Params [M]**	**MACC [M]**	**RAM [KB]**
3	0.5	1.9	360	172

V. NETWORK QUANTISATION

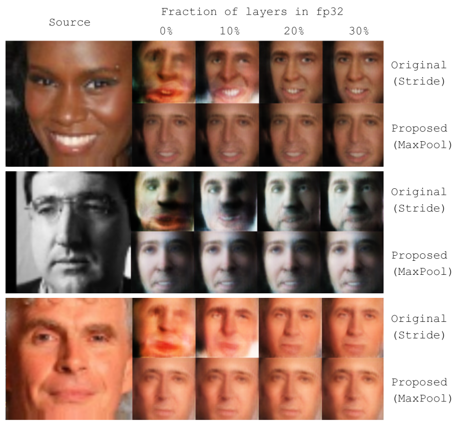

Fig. 4. Effects of partial quantisation on generated images

The quantisation of generative networks is a complex operation, as even small errors in the latent representation can

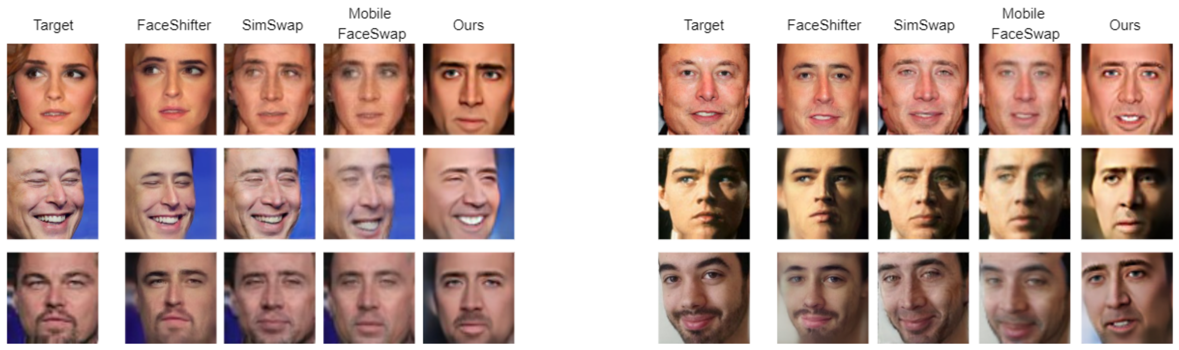

Fig. 5. Comparison between our proposed approach and other State-of-the-art face-swapping frameworks

generate large discrepancies with the unquantised model in the images generated by the decoder network.

In particular, naively applying post-training quantisation to the network causes a substantial degeneration in the quality of the produced images, as demonstrated in [33]. Such effects can be mitigated by employing a partial quantisation scheme, where a fraction of the layers of the network is left operating on floating point values. In particular, we can see from Figure 4 (top row of images for each source image) that to obtain images of acceptable quality, it is necessary to leave around 20% of the layers with lower output resolutions in floating point. While this allows for higher fidelity image generation, it impacts inference latency, as our target platform's TPU can only provide acceleration for integer operators while floating point operations need to be executed on the RISC V cores.

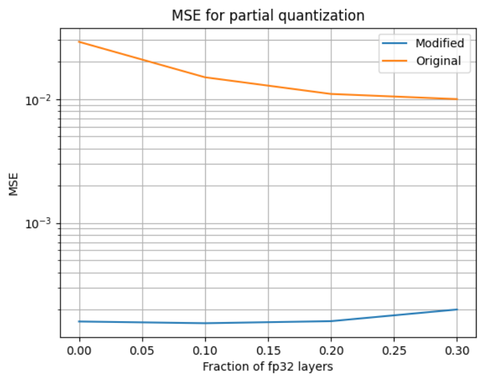

Fig. 6. MSE between generated images of full precision and quantised models varying the percentage of quantised layers.

To decrease the quantisation error, two changes were made in the convolutional blocks of the network:

- The activation function of all layers has been changed to the thresholded `ReLU6` activation to achieve more consistent ranges for quantisation
- The downsampling blocks have been changed slightly to include a max-pooling operation instead of relying on a striding operation for downsampling, increasing in this way the Signal to Quantisation Noise Ratio

Figure 4 shows the comparison between original network and network after these modifications for different percentages of partial quantisation. We can see how the two implemented changes allow for sufficient quality of generated images, even with a fully quantised network, without incurring in the latency penalty for partial quantisation. Figure 6 compares the Mean Squared Error, pixel per pixel of the generated image between the non-quantised floating-point model and the partially quantised model, with different thresholds. We observe a MSE orders of magnitude lower with the modified model. Some noise in the MSE measurement is to be expected.

VI. Results

Table IV compares our proposed approach with other SOTA techniques in terms of resource usage and performance. Latency tests were performed on a consumer GPGPU (RTX2060). Since our model relies on conv2d instead of depthwise operations, it can be seen that, in comparison with mobile-faceswap, the number of MACCs performed is higher. Still, the latency of the model is lower, given the higher efficiency of the operations and the lower fragmentation in our proposed architecture, thanks to the absence of an id-injection module that is instead required for many-to-many face swapping approaches. We also trained a second network, referred to as `Phinet GAN S` in Table IV, swapping the 2D convolutions for depthwise operators to compare the different approaches. Thanks to the even lower parameter and operation count, this network could prove a viable approach for even more resource-constrained devices.

TABLE IV
Requirements and performance for different SOTA approaches and our proposed networks

	Size	**MACC**	**Latency**	**FID ↓**
Simswap	182.4MB	24.1G	289ms	**130**
Faceshifter	1400MB	45.6G	491ms	134
Mobile faceswap	2MB	0.12G	18.3ms	132
Phinet GAN S	**0.46MB**	**0.088G**	**6ms**	136
Phinet GAN L	1.9MB	0.360G	11.2ms	**130**

We tested our approach on a Kendryte `K210` microcontroller, a tinyML-oriented device consisting of two 64-bit RISC-V cores and a convolutional accelerator (Tensor Processing Unit, TPU) capable of up to 500 GMACC per second. Power consumption of the device was measured at around $280mW$ with both the CPU and TPU operating at $400MHz$.

The network model was converted for usage on the K210 using `nncase 0.2.0`, and a modified version of micropython has been used to allow for generative networks to be run on the device. The final model required $1.37MB$ of FLASH storage while executing the network at a frame rate of $16.4fps$ and using $172KB$ of the device's RAM. The average power required by the platform running inference at the maximum speed has been measured at $280mW$ - an energy consumption of $17.1mJ$ per frame.

VII. Conclusion

This work showed the possibility of developing face-swapping frameworks with very low energy consumption using `Phinet` and `Phinet-GAN` convolutional blocks. Results comparable to state-of-the-art approaches can be obtained with only a fraction of the energy and using networks with such low hardware requirements that inference can be performed on microcontrollers. In particular, using a Kendryte K210 MCU in conjunction with the proposed approach, we verified that the energy required to perform face-swapping can be reduced by 3 to 4 orders of magnitude without significant performance loss. In future works, we will aim at adapting many-to-many face-swapping solutions to increase the generality of the proposed strategy. In conclusion, we would like to highlight the ethical concerns of the proposed pipeline, which are in line with the ones of the original approach, summarized in [34].

References

[1] D. Bajovic, A. Bakhtiarnia, G. Bravos, A. Brutti, F. Burkhardt, D. . Cauchi, and J. Zammit, "Marvel: Multimodal extreme scale data analytics for smart cities environments," in *2021 International Balkan Conference on Communications and Networking (BalkanCom)*, 2021, pp. 143–147.

[2] A. A. Ross and A. A. Othman, "Visual cryptography for biometric privacy," *IEEE Transactions on Information Forensics and Security*, vol. 6, pp. 70–81, 2011.

[3] H. Hukkelås, R. Mester, and F. Lindseth, "Deepprivacy: A generative adversarial network for face anonymization," *CoRR*, vol. abs/1909.04538, 2019. [Online]. Available: http://arxiv.org/abs/1909.04538

[4] M. Maximov, I. Elezi, and L. Leal-Taixé, "CIAGAN: conditional identity anonymization generative adversarial networks," *CoRR*, vol. abs/2005.09544, 2020. [Online]. Available: https://arxiv.org/abs/2005.09544

[5] N. Dall'Asen, Y. Wang, H. Tang, L. Zanella, and E. Ricci, "Graph-based generative face anonymisation with pose preservation," *CoRR*, vol. abs/2112.05496, 2021. [Online]. Available: https://arxiv.org/abs/2112.05496

[6] O. Gafni, L. Wolf, and Y. Taigman, "Live face de-identification in video," *CoRR*, vol. abs/1911.08348, 2019. [Online]. Available: http://arxiv.org/abs/1911.08348

[7] Q. Sun, A. Tewari, W. Xu, M. Fritz, C. Theobalt, and B. Schiele, "A hybrid model for identity obfuscation by face replacement," *ArXiv*, vol. abs/1804.04779, 2018.

[8] H. Tang, S. Bai, P. H. S. Torr, and N. Sebe, "Bipartite graph reasoning gans for person image generation," *ArXiv*, vol. abs/2008.04381, 2020.

[9] Z. Li, H. Zhu, L. Cao, L. Jiao, Y. Zhong, and A. Ma, "Face inpainting via nested generative adversarial networks," *IEEE Access*, vol. 7, pp. 155 462–155 471, 2019.

[10] H. Rai and N. Shukla, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2018.

[11] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019.

[12] R. Chen, X. Chen, B. Ni, and Y. Ge, "Simswap: An efficient framework for high fidelity face swapping," *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[13] L. Li, J. Bao, H. Yang, D. Chen, and F. Wen, "Faceshifter: Towards high fidelity and occlusion aware face swapping," *ArXiv*, vol. abs/1912.13457, 2019.

[14] D. Chesakov, A. Maltseva, A. Groshev, A. Kuznetsov, and D. Dimitrov, "A new face swap method for image and video domains: a technical report," *CoRR*, vol. abs/2202.03046, 2022. [Online]. Available: https://arxiv.org/abs/2202.03046

[15] Z. liang Xu, X. Yu, Z. Hong, Z. Zhu, J. Han, J. Liu, E. Ding, and X. Bai, "Facecontroller: Controllable attribute editing for face in the wild," in *AAAI Conference on Artificial Intelligence*, 2021.

[16] Y. Wang, X. Chen, J. Zhu, W. Chu, Y. Tai, C. Wang, J. Li, Y. Wu, F. Huang, and R. Ji, "Hififace: 3d shape and semantic prior guided high fidelity face swapping," *ArXiv*, vol. abs/2106.09965, 2021.

[17] F. Paissan, A. Ancilotto, and E. Farella, "Phinets: A scalable backbone for low-power ai at the edge," *ACM Trans. Embed. Comput. Syst.*, feb 2022. [Online]. Available: https://doi.org/10.1145/3510832

[18] A. Brutti, F. Paissan, A. Ancilotto, and E. Farella, "Optimizing phinet architectures for the detection of urban sounds on low-end devices," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1121–1125.

[19] A. Ancilotto, F. Paissan, and E. Farella, "On the role of smart vision sensors in energy-efficient computer vision at the edge," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, pp. 497–502.

[20] F. Paissan, A. Ancilotto, A. Brutti, and E. Farella, "Scalable neural architectures for end-to-end environmental sound classification," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 641–645.

[21] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "Mcunet: Tiny deep learning on iot devices," *ArXiv*, vol. abs/2007.10319, 2020.

[22] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "Mcunetv2: Memory-efficient patch-based inference for tiny deep learning," in *Neural Information Processing Systems*, 2021.

[23] F. Paissan, M. Gottardi, and E. Farella, "Enabling energy efficient machine learning on a ultra-low-power vision sensor for iot," *arXiv preprint arXiv:2102.01340*, 2021.

[24] Z. Xu, Z. Hong, C. Ding, Z. Zhu, J. Han, J. Liu, and E. Ding, "Mobilefaceswap: A lightweight framework for video face swapping," *CoRR*, vol. abs/2201.03808, 2022. [Online]. Available: https://arxiv.org/abs/2201.03808

[25] deepfakes, "faceswap," https://github.com/deepfakes/faceswap, 2020.

[26] shaoanlu, "faceswap-gan," https://github.com/shaoanlu/faceswap-GAN, 2017.

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017.

[28] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," *CoRR*, vol. abs/1710.08092, 2017. [Online]. Available: http://arxiv.org/abs/1710.08092

[29] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. [Online]. Available: http://arxiv.org/abs/1706.08500

[30] Y. Nirkin, Y. Keller, and T. Hassner, "Fsgan: Subject agnostic face swapping and reenactment," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7183–7192, 2019.

[31] U. Demir and G. B. Ünal, "Patch-based image inpainting with generative adversarial networks," *ArXiv*, vol. abs/1803.07422, 2018.

[32] M. Li, W. Zuo, and D. Zhang, "Convolutional network for attribute-driven and identity-preserving human face generation," *ArXiv*, vol. abs/1608.06434, 2016.

[33] P. Andreev, A. Fritzler, and D. P. Vetrov, "Quantization of generative adversarial networks for efficient inference: a methodological study," *ArXiv*, vol. abs/2108.13996, 2021.

[34] A. de Ruiter, "The distinct wrong of deepfakes," *Philosophy & Technology*, 2021.