Improving Face Liveness Detection Robustness with Deep Convolutional Generative Adversarial Networks

Ruslan Padnevych†, David Semedo*, David Carmo†, João Magalhães*
Universidade NOVA de Lisboa, Caparica, Portugal
†{r.padnevych, dm.carmo}@campus.fct.unl.pt *{df.semedo, jm.magalhaes}@fct.unl.pt

***Abstract*—Non-intrusive face authentication and biometrics are becoming a commodity with a wide range of applications. This success increases their vulnerability to attacks that need to be addressed with more sophisticated methods. In this paper we propose to strengthen face liveness detection models, based on photoplethysmography (rPPG) estimated pulses, by learning to generate high-quality, yet fake pulse signals, using Deep Convolutional Generative Adversarial networks (DCGANs). The simulated liveness signals are then used to improve detectors by providing it with a better coverage of potential attack-originated signals, during the training stage. Thus, our DCGAN is trained to simulate real pulse signals, leading to sophisticated attacks based on high-quality fake pulses. The full liveness detection framework then leverages on these signals to assess the genuineness of pulse signals in a robust manner at test-time. Experiments confirm that this strategy leads to significant robustness improvements, with relative AUC gains $> 3.6\%$. We observed a consistent performance improvement not only in GAN-based, but also in more traditional attacks (e.g. video face replay). Both code and data will be made publicly available to foster research on the topic[1].**

***Index Terms*—Face liveness detection, Generative adversarial networks, Presentation attacks, EVM pulse signals**

I. Introduction

Due to the rise in popularity of face unlocking mechanisms on smartphones (and subsequently on other mediums), the risk of face spoofing - by using a copy of someone's face to bypass security measures - becomes higher [1]. One possible reason for this risk is the fact that face spoofing is a very low-cost attack, since basically anyone has access to a camera or a printer, and a picture of the person they want to target [2]. Hence, mechanisms like liveness biometrics can help mitigating this risk, since the presence of a real person in time and space implies that it is not a presentation attack (PA) [3]–[6].

Pulse estimation through photoplethysmography (rPPG) is an interesting approach that enables the development of non-intrusive and biometric-based liveness detectors [4], [5], [7]. In this setting, through an RGB camera, individual's pulse signals can be estimated with well-established approaches like Eulerian Video Magnification (EVM) [7]. However, given the sensitive of this particular domain due to a myriad of reasons, data scarcity is often an issue, making the gathering of representative data to learn robust detectors a major challenge. We propose to increase the robustness of such detectors by improving their generalization through robust training strategies. Namely, inspired by previous works in the generation of ECG signals [8], instead of just adding random noise to existing pulses to perform data augmentation, we propose a more sophisticated approach leveraged by a Discriminative Convolutional Generative Adversarial Network (DCGAN) [9], that learns the distribution of training data.

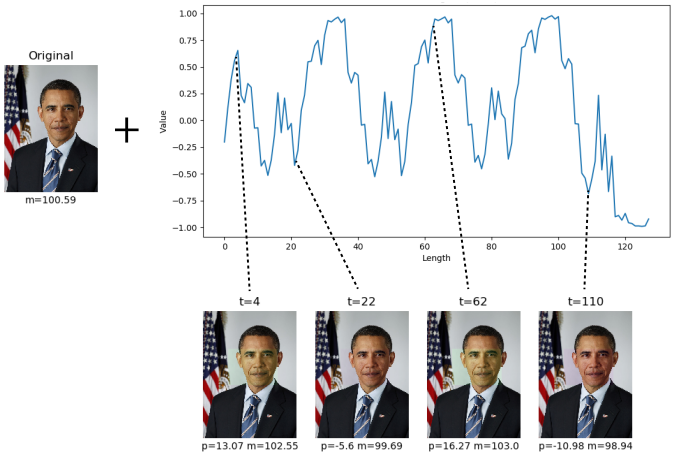


Fig. 1: Creation of a presentation attack using generated pulse signals. p - represents the pulse value added to each pixel of the face bounding-box (variation of green channel); m - represents the mean value of the green channel of the whole photograph. Link to injected GAN pulse face video.

These signals are obtained by placing a Generator in competition (min-max game) with a Discriminator, until both converge to a point of equilibrium. The goal of the Generator is to generate fake signals that are similar to the real signals used during the training. On the other hand, the Discriminator, which is a binary classifier with the task of discriminating between fake and real pulses, is jointly trained with both the samples produced by the Generator and the real samples from the training set. This process leads to two outcomes that we explore in this paper:

- **Presentation Attack Generator.** The DCGAN generated pulse signals can be used to create an attack. We show that the generated pulse signals are quite similar to the real ones. This makes it feasible to leverage the Generator, depicted in Figure 2, as a fake pulse signals source.
- **Roust TCN-based PA Detector.** The second outcome of this paper consists of a novel robustly trained TCN-based [10] PA detector, whose effectiveness further evi-

[1]ToBePublishedUponAcceptance.

dences the feasibility of our augmented training procedure, that includes both generated fake signals and pulse signals from conventional attacks.

II. Related Work

Through photoplethysmography (rPPG) with RGB cameras, face pulses can be estimated and used for liveness assessment [3]. For instance, Eulerian Video Magnification [7] takes a video sequence and performs spatial decomposition, followed by a temporal filtering of frames. This process results in an amplified signal revealing information that was originally invisible to the naked eye. Namely, is enables capturing blood circulation through skin color variation. Alternatively, pulse can be remotely estimated using CNN-based approaches [6]. Since the latter requires training, PA detectors often opt for statistic-based approaches. In [4] liveness is assessed by jointly considering rPPG features represented by multi-scale long-term spectral statistics, and global and local region information extracted from a patch-based CNN.

On a different vein, GANs [11] are generative models, composed by a discriminator and a generator, that are trained in a zero sum game. GANs have been used in the literature for generating ECG signals [8], hinting that they should also be adequate in a liveness detection setting, as a way to automatically generate pulse signals.

III. Robust Face Liveness Detection

The proposed framework implements a face liveness detector that is divided in two parts. First, the face video is processed and with the EVM technique, a pulse signal is extracted. Second, the estimated pulse signal is passed to a classifier that decides if it is an attack or not. This section describes how to augment the classifier training data through the creation of artificial pulse signals in an automated way. This is achieved with a DCGAN model, illustrated in Figure 2, that will be described in the following sections.

A. Generator of artificial Pulse signals

The Generator of pulse signals, detailed in Figure 2, will be responsible for the generation of pulse signals from a random seed: it is composed of transposed convolutional layers, batch normalization and ReLU activation functions. For the output, a $tanh$ function is applied to constrain the resulting values to a range of $[-1, 1]$, thus matching the scale of the training data.

As input data, for each sample, the generator receives a vector drawn from a normal distribution, and tries to produce a signal whose size and distribution is the same as the real signals of the training set. In practice, this approach is only possible due to the use of the stride in the transposed convolutional layers, followed by a batch normalization [12] layer and a ReLU activation function. During our experiments, we found that it is important to apply batch normalization after the transposed convolution layer, to help the flow of gradients during training. It acts as a regularizer, eliminating the need to use Dropout. As a consequence, we were able to use higher learning rates and make the model less sensitive to bad initializations. Formally:

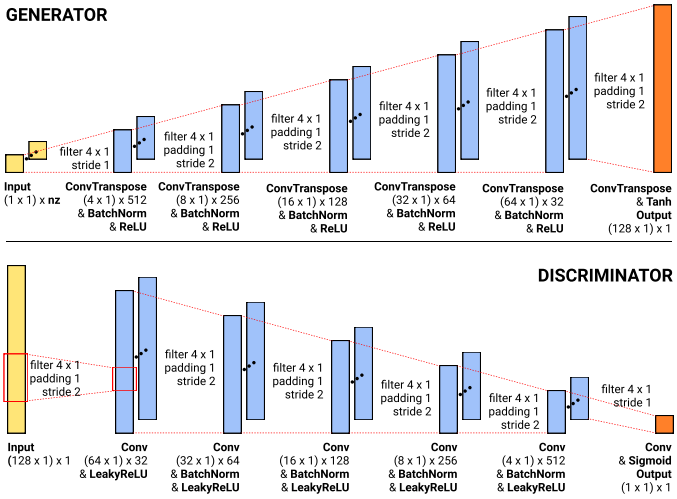


Fig. 2: DCGAN Generator model architecture used to produce artificial pulse signals (top) and DCGAN Non-linear Discriminator model architecture used to distinguish artificial from real pulse signals (bottom).

- z represents a vector extracted from a normal distribution.
- *G(z)* represents the Generator's network which tries to approximate the distribution of the signal z to the distribution of training data. Hence, the purpose of the Generator is to estimate the distribution of the training data (P_{data}) in order to generate artificial samples from the estimated distribution (P_z).

B. Non-linear Discriminator for liveness detection

The architecture of the non-linear Discriminator, presented in Figure 2, is quite similar to the Generator architecture, described in the previous subsection. The non-linear Discriminator model, which is a binary classifier, unlike the Generator, is formed by convolutional layers. These are followed by a batch normalization layer and finally a *LeakyReLU* activation function.

As input data, this model receives a 128-dimensional signal and returns as result 1 (artificial signal) or 0 (real signal) using a *sigmoid* activation function. It is worth mentioning that we use strided convolutional layers instead of using pooling to reduce the input sample. This enables the the network to learn its own pooling function [9]. Formally we have:

- x represents the pulse signal given to the Discriminator.
- *D(x)* represents the Discriminator's model which classifies the signal x, returning as a result 1 (artificial signal) or 0 (real signal).
- *D(G(z))* represents the result (0 or 1) of the classification produced by the Discriminator when evaluating a signal created by the Generator.

C. Cost function

The cost function of a DCGAN is defined as [11]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[log\ D(x)\,] + \mathbb{E}_{z\sim P_z(z)}\,[log(1-D(G(z)))\,]. \quad (1)$$

At the beginning of the training process, when the Generator is weak, the Discriminator manages to reject the samples with great confidence because they are different from the training data, which causes the saturation of $(log(1-D(G(z))))$. To solve this problem, we adopt the solution of [9], [11], which instead of letting the Generator minimize the $(log(1-D(G(z))))$ it is more efficient to maximize the $log\ D(G(z))$ which allows us to have much stronger gradients at the beginning of the training. For that reason, the previous equation was rewritten by applying this new approach, leading to:

$$\max_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}\,[log\ D(x)\,] + \mathbb{E}_{z\sim P_z(z)}\,[log(D(G(z)))\,]. \quad (2)$$

This Discriminator-Generator competition ends when both models reach a point where they stagnate , i.e. $P_z \approx P_{data}$. At this point, the Generator is expected to be able to fool the Discriminator half the time, i.e. $D(x) = \frac{1}{2}$.

D. Training Process of the DCGAN

The DCGAN training process used to obtain a strong generator of artificial pulse signals is carried in two alternating steps.

1) Discriminator Update: We recall that the objective of the Discriminator's training is to maximize the result of correctly discriminate between false and real signals. In practice, as stated in [11], the intention is to update the Discriminator in a way that maximizes each update gradient information. To achieve this, it is necessary to maximize the $log(D(x))+log(1-D(G(z)))$. Having this in mind and following [9], we created small sets of real and false pulses to perform estimate the gradients in two phases.

First phase: Initially, a batch of real pulses is formed from the training set. Then, the created batch is passed to the Discriminator to evaluate the loss $(log(D(x)))$ through the binary cross-entropy loss function.

Second phase: In the second phase of this process, a batch of artificial signals produced by the current Generator is created. As in the first phase, the created batch is used as input data for the Discriminator to evaluate the loss $(log(1-D(G(z))))$ using the binary cross-entropy loss function.

After these two phases, with gradients of both batches already accumulated, we backpropagate gradients using the Adam optimizer. Then, we proceed to the Generator's training part.

2) Generator Update: Following [11], we maximize the $log(D(G(z)))$, instead of minimizing the $log(1-D(G(z))$, to mitigate vanishing gradients. To achieve this, the following steps have been taken: (1) The data batch produced by the Generator in Part 1 is previously classified by the Discriminator; (2) The loss of the Generator is computed using a binary cross-entropy loss over ground truth labels; (3) The gradients of the Generator is evaluated using a backward pass; (4) Finally, the parameters of the Generator are updated using Adam's optimizer.

Using ground truth labels enables us to use $log(x)$ from the binary cross-entropy loss function instead of $log(1-x)$, which is exactly what was intended. The process of both parts is then repeated until the two models converge and plausible artificial heart signals are generated.

E. Injecting Fake Generated Pulse Signals in Face Videos

The generated pulse signals can be used to easily create a real-life attack using just a static photograph. Given a generated pulses, we can compose a video from the photo, where we vary the green color in an amount that is sampled from the generated signal, as shown in the Figure 1. This way, we can observe that the color of an individual's face is slightly changed (greener→blood flows) in some frames, in a way that it follows a close to genuine pulse signal.

This simple example illustrates the importance of this work in preventing presentation attacks.

IV. Experiments

A. Dataset

To evaluate the proposed models on our setting, i.e. face liveness detection on the fly, we collected a representative dataset. All videos were filmed in different environments such as good/low/frontal illumination and different stress conditions to simulate the real use case. For each volunteer, we decided to gather 9 videos of the face of a real individual (Real) and 12 videos of a presentation attack (Fake) in different formats of the same person. Furthermore, the real videos were captured under different stress conditions, such as rest, after walking upstairs and after brisk walking, to create diverse pulse variations. On the other hand, as it is not possible to vary the stress conditions in the presentation attack videos, we decided to use different sizes of photographs, for example, photo of the neutral face, moving the photo, face sized photo and mask as these represent potential basic attacks that can occur in a real setting. Therefore, we obtained a total of 21 videos, of roughly 2 minutes each, all of them with distinct characteristics. Subsequently, the recorded videos were cut into 5 sec videos To perform these cuts, a 1 sec sliding window was used, which gradually progresses until it reaches 2 minutes. As a result, more or less 2.436 videos were obtained from each of 6 volunteers, obtaining in total 7.424 videos of which 5.364 were used to train the models, 947 to validate the training and 1.113 to realise the final test.

In addition to the data collected (which will be made available), we generated as many artificial signals as needed using *DCGAN with different hyper-parameters*. We kept the balance between between Real and Fake + Generated samples used along the learning process of the models. During the data processing, we normalize each signal in order to guarantee that all of them are on the same scale, $[-1, 1]$.

B. Presentation Attack Detector Baselines

All models were trained in a supervised fashion under three attack conditions: (i) *Real* vs *Fake*, (ii) *Real* vs *Generated*, and

(iii) *Real* vs *Fake + Generated*, where this last type of attack combines the two previous ones. Using the user videos and the our generated pulse signals, we trained the GAN as described previously and used PA detectors sharing the same architecture as the GAN discriminator:

- **GAN:** Classifier with the same architecture as the Discriminator used during the DCGAN training, which is a 5 CNN layers classifier as represented in Figure 2.
- **GAN-2:** Same architecture as in GAN, but with 2 fewer CNN layers, to verify how it affects performance when pulse signals are generated by a less complex architecture.
- **GAN+2:** Same architecture as GAN, but with 2 more CNN layers, representing a more complex architecture.

The above baselines favour the cases where the attacker GAN discriminator has privileged information regarding the target PA detector. However, one would expect the PA detector architecture to be unknown, i.e. with a different architecture from the GAN discriminator, which is a more plausible scenario. Following this reasoning, several robust liveness detection algorithms were trained and evaluated:

- **Linear:** simple fully connected feed-forward neural network (FFNN), with a *sigmoid* activation function.
- **TCN:** a) a Temporal Convolutional Network model [10] (**TCN**) with depth $d = 5$, that uses dilated causal convolutions to process signals in a sequence-modeling setting.
- **CNN+Res:** a CNN-based classifier (**CNN+Res**), with standard convolutions, simulating a TCN single-block. This includes a 5 CNN layers classifier as represented in Figure 2

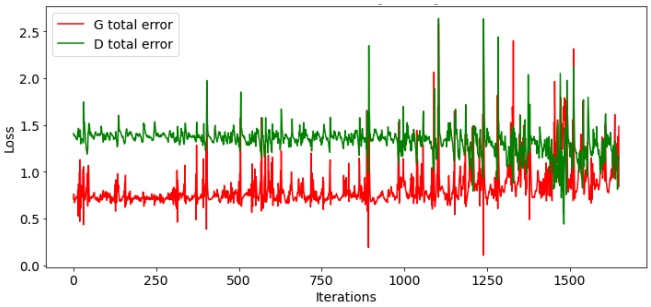
All PA detectors were trained on the indicated data sample.

C. Results and Discussion

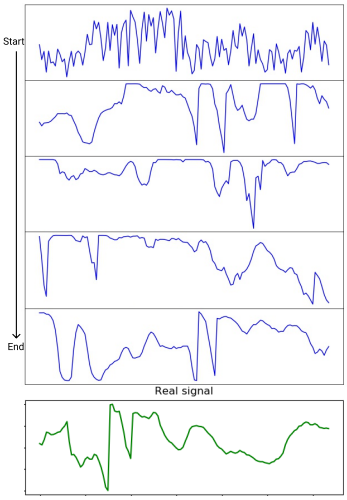
1) Training convergence: First, to confirm that the implemented DCGAN was converging, we analyzed the training loss of the Generator and the Discriminator. It is possible to observe in Figure 3a, that the losses of both models are converging as expected. This means that the Generator is creating more realistic pulse signals and due to this, the Generator's loss is decreasing. On the other hand, during the training process, the Discriminator is also improving and correctly discriminating the *Real* samples from the *Generated* samples. Thus, the Generator's loss is increasing.

2) Generation of Pulse signals: Figure 3b illustrates how the Generator is generating increasingly better signals as the training epochs advance. In the last epoch, the **Generated** signal (blue curve at the bottom of Figure 3b) is very similar to the **Real** signals (green line at the bottom of the figure). With this approach, we were able to produce good quality pulse signals, meaning that the data distribution is very similar to the real ones. Making use of these generated pulse signals, we now proceed to evaluate how different PA detectors perform.

3) Robustness to Presentation Attacks: In the most critical setting, presentation attack detectors should be able to detect both *Fake* signals and *Generated* signals (R vs F + G).



(a) Generator and Discriminator loss convergence.



(b) Pulse signals evolution.

Fig. 3: DCGAN training and pulse signals quality analysis.

Analyzing Table I, we see that performance is directly linked to the classifier complexity, i.e., the linear detector is the worst performing classifier, far from the performance achieved by the GAN based detector that share the same architecture as the DCGAN's Discriminator. It is clear that the TCN approach achieves the best AUC result (93.55), followed by the more complex DCGAN+2 model with an AUC of 92.02.

Another main observation is that in all models, adding generated signals helps improving the performance, based on columns R vs (F+G). When looking at the *Fake signal* type of attack (column R vs F), it is possible to see that both CNN+Res and TCN models demonstrate equivalent performance. In terms of detecting *Generated pulses*, while the more robust models CNN+Res and TCN obtain higher AUC, DCGAN achieves similar AUC given that it matches the signal generator architecture. The simpler CNN method with 2 fewer layers than the original DCGAN architecture, GAN-2, achieves the worst AUC (88.08). This compares poorly with the best GAN-based detector, GAN+2, that has an AUC of 92.02.

Finally, it is interesting to note that when the detector architecture matches the generator one, using the same architecture that was used to generate the pulse signals is never the best approach in the most critical setting: R vs (F + G).

TABLE I: Presentation attack detection results under different types of attacks: Fake photo (F), Generated pulse signal (G) and both attacks (F + G). Δ AUC column shows the robustness improvement by adding the generated pulse signals.

Detector	Attack Type	F1-score	AUC	Δ AUC
	PA detector architecture = Discriminator			
DCGAN-2	R vs F	73.37	81.13	+7.9%
	R vs G	93.06	97.73	
	R vs (F + G)	84.11	**88.08**	
DCGAN	R vs F	83.39	83.82	+6.8%
	R vs G	94.84	98.06	
	R vs (F + G)	85.13	**89.97**	
DCGAN+2	R vs F	84.99	87.37	+5.1%
	R vs G	93.33	97.80	
	R vs (F + G)	87.39	**92.02**	
	PA detector architecture $\neq$ Discriminator			
Linear	R vs F	79.69	67.73	+1.2%
	R vs G	80.99	89.07	
	R vs (F + G)	73.98	**68.55**	
CNN+Res	R vs F	87.59	90.21	+0.6%
	R vs G	95.71	98.23	
	R vs F + G	89.80	**90.73**	
TCN	R vs F	87.70	90.17	+3.6%
	R vs G	95.47	98.49	
	R vs (F + G)	88.70	**93.55**	

4) ROC curves: The presentation attack detection performance can be observed in Figure 4, which shows the real case scenario where the detector may be faced with a *Real* pulse (genuine user), *Fake* pulse (corresponding to a traditional attack) or *Generated* pulse signal samples.

For comparing the models in terms of the ROC AUC, we can observe in Figure 4 that all classifiers have the same monotonic behaviour. The best performing approach, TCN with 93.55 AUC, is consistently superior, followed by the GAN+2 model with 92.02 AUC. This demonstrates that models with more complex architectures benefit from having access to more training data, covering both pulse signals corresponding to traditional attacks (e.g. photo replay) and generated signals.

V. Conclusion

This paper demonstrated how pulse signals, generated by a DCGAN, can be used to increase the robustness of presentation attack detector models. Namely, the key contributions of this paper are as follows:

- Our DCGAN approach provides a principled approach for face pulse generation, that can effectively increase the robustness of presentation attack detectors.
- Complex detectors models, such as the TCN, benefit from the use generated pulse signals. We observed that by training detectors with both photo replay attacks and generated signals, provides better generalization and increased performance.

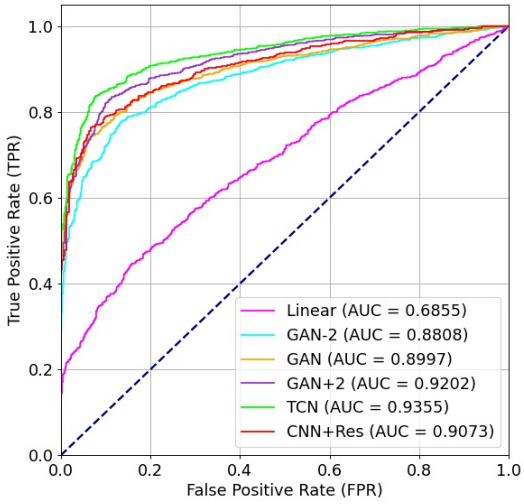


Fig. 4: ROC curves on the Real, Fake and Generated pulses scenario.

As future work, we plan to explore a multi-architecture pulse generation solution, making it harder for an attacker to exploit weaknesses of a specific architecture.

References

[1] L. Tarassenko, M. Villarroel, A. Guazzi, J. Jorge, D. Clifton, and C. Pugh, "Non-contact video-based vital sign monitoring using ambient light and auto-regressive models," *Physiological measurement*, vol. 35, no. 5, p. 807, 2014.

[2] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 746–761, 2015.

[3] R. Ramachandra and C. Busch, "Presentation attack detection methods for face recognition systems: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 1–37, 2017.

[4] B. Lin, X. Li, Z. Yu, and G. Zhao, "Face liveness detection by rppg features and contextual patch-based cnn," in *Proceedings of the 2019 3rd International Conference on Biometric Engineering and Applications*, ser. ICBEA 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 61–68.

[5] X. Li, J. Komulainen, G. Zhao, P.-C. Yuen, and M. Pietikäinen, "Generalized face anti-spoofing by detecting pulse from face videos," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 4244–4249.

[6] F. Bousefsaf, A. Pruski, and C. Maaoui, "3d convolutional neural networks for remote pulse rate measurement and mapping from facial video," *Applied Sciences*, vol. 9, p. 4364, 10 2019.

[7] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. T. Freeman, "Eulerian video magnification for revealing subtle changes in the world," *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*, vol. 31, no. 4, 2012.

[8] T. Golany and K. Radinsky, "Pgans: Personalized generative adversarial networks for ecg synthesis to improve patient-specific deep ecg classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 557–564.

[9] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.