**GAN Related Work**

* I. Goodfellow et al., 2014: Introduced GANs as a way to generate data
* Mirza and Osindero., 2014: Introduced cGANs, which use additional data that is fed to the discriminator and generator to condition the model. This way, the data that should be generated can be specified more precisely (e.g. generate images of a specific class). As an example, the authors trained a cGAN on MNIST images, to generate images of numbers.
* J. Gauthier, 2014: Builds upon Mirza and Osindero and goes into more detail. He further explains: The condition restricts the generator in it’s output and the discriminator in it’s input. He uses a cGAN for face generation.
* Denton et al., 2015: Introduced LAPGAN. This uses a Laplacian pyramid of networks (convnet). Each level operates conditioned on the output of the previous scale. (So each level generates the difference between the current and the next level of the corresponding gaussian pyramid. The different levels can be added to get the final result image). This allows for generation of larger images.
* Radford et al., 2016: Introduced DCGANs (Deep Convolutional GANs ; Unsupervised). Also they were concerned with exploring the latent space and visualizing the internals of their network. They showed that simple vector arithmetic can be used w.r.t the z-space. They used their network to generate images of faces.
* Creswell et al., 2017: Review of GANs. Good overview over the different improvements/challenges since the original paper in 2014.
* C. Wang et al., 2018: Introduced E-GANs for more stable training and improved generative performance. The training procedure is treated as an evolutionary problem. A population of generators is evolved, where only well-performing generators persist. They also use the CelebA dataset.
* T. Wang et al., 2018: Use a cGAN to generate high-resolution realistic images. They introduce a new loss function, as well as new coarse-to-fine generator and a multi-scale discriminator architecture. The generator is decomposed into two sub-networks, a global generator network who works on lower resolution label maps and a local enhancer network who is “wrapped around” the global generator. First the global generator is trained, then the local enhancer, then the two networks are trained together.  
  The discriminator consists of three discriminators, that operate at different image resolutions.
* Karras et al., 2019: Introduced StyleGAN, which incorporates a new style-based generator architecture that leads to separation of high-level attributes from stochastic variation in the generated images. It therefore enables intuitive mixing and interpolation. The input latent code is fed into a mapping network and transformed into a intermediate latent code, which is then transformed into styles that control the layers of a synthesis network.
* Karras et al., 2020: Further improvement of StyleGAN. (I find it quite hard to understand).
* Sagong et al., 2020: Introduces conditional convolutional layers, which use different weights, depending on the given class.

Interesting repositories:

<https://github.com/HighCWu/Condition-StyleGAN-PyTorch>

<https://github.com/jayleicn/animeGAN>