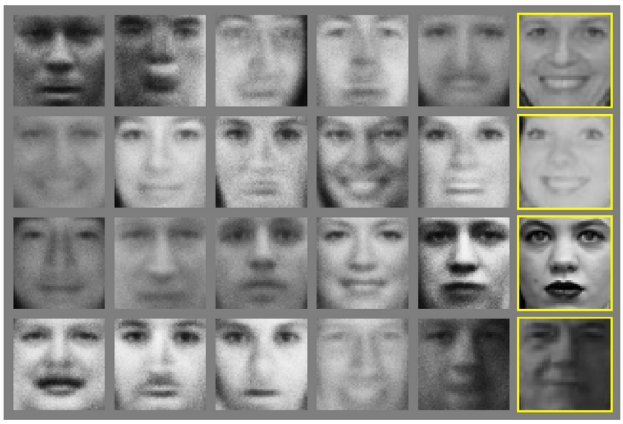
GANs: Generative Adversial Networks

You may try:
https://www.whichfaceisreal.com
https://thispersondoesnotexist.com
https://this-person-does-not-exist.com/en

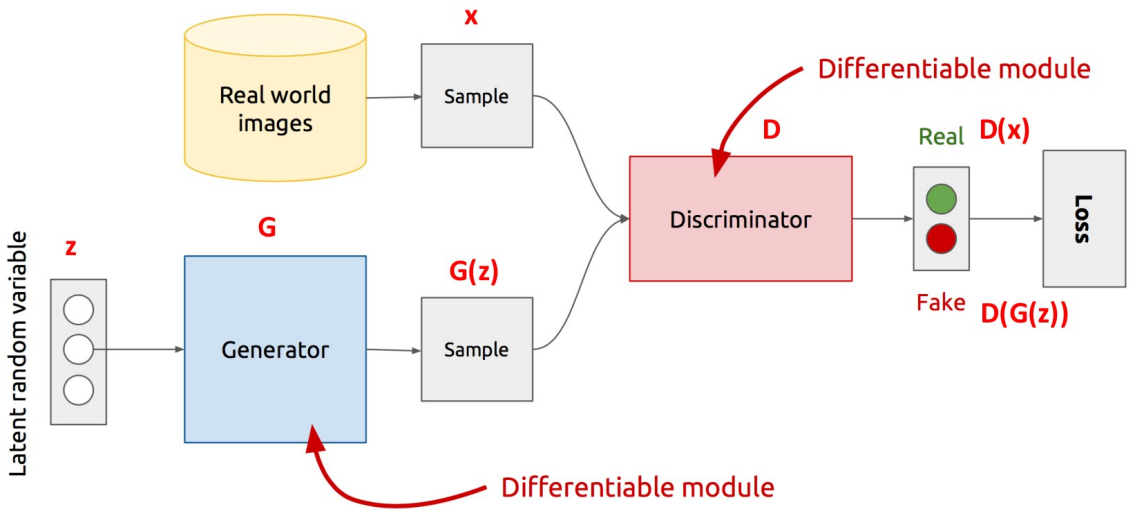
But we will start from GAN 2014: Ian J. Goodfellow Generative Adversarial Nets (2014)



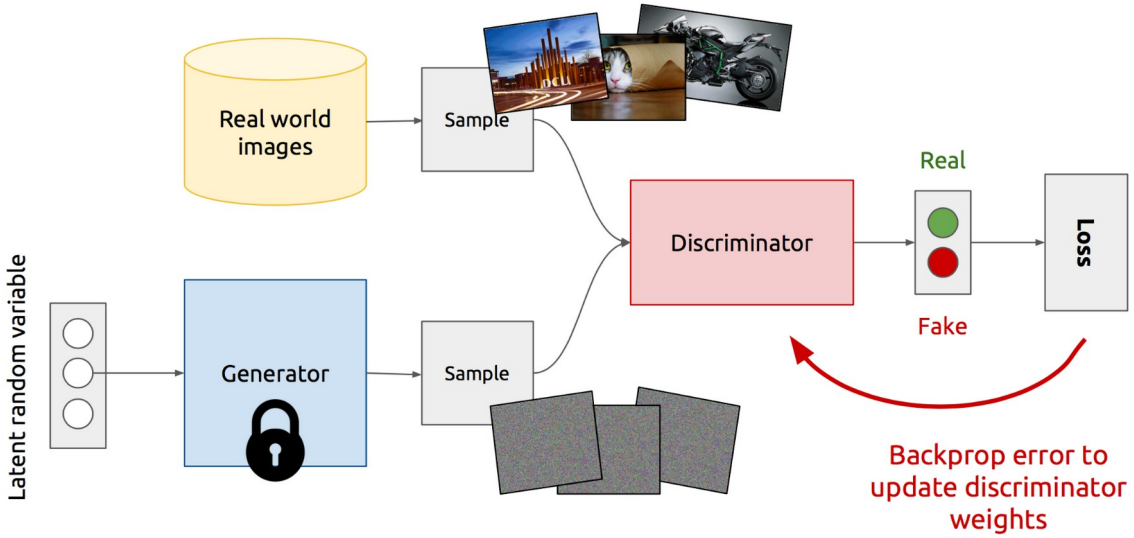
You may also try:

- DCGAN.py
- https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
- dcgan_faces_tutorial.ipynb
- https://github.com/nashory/pggan-pytorch

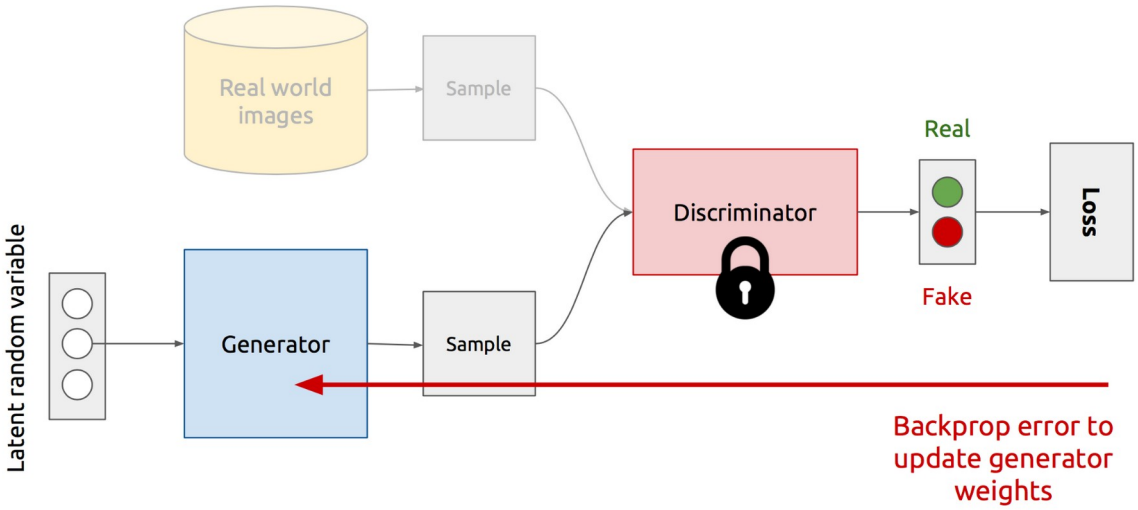
GAN Architecture



Training discriminator



Training generator



Generátor

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator updates

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

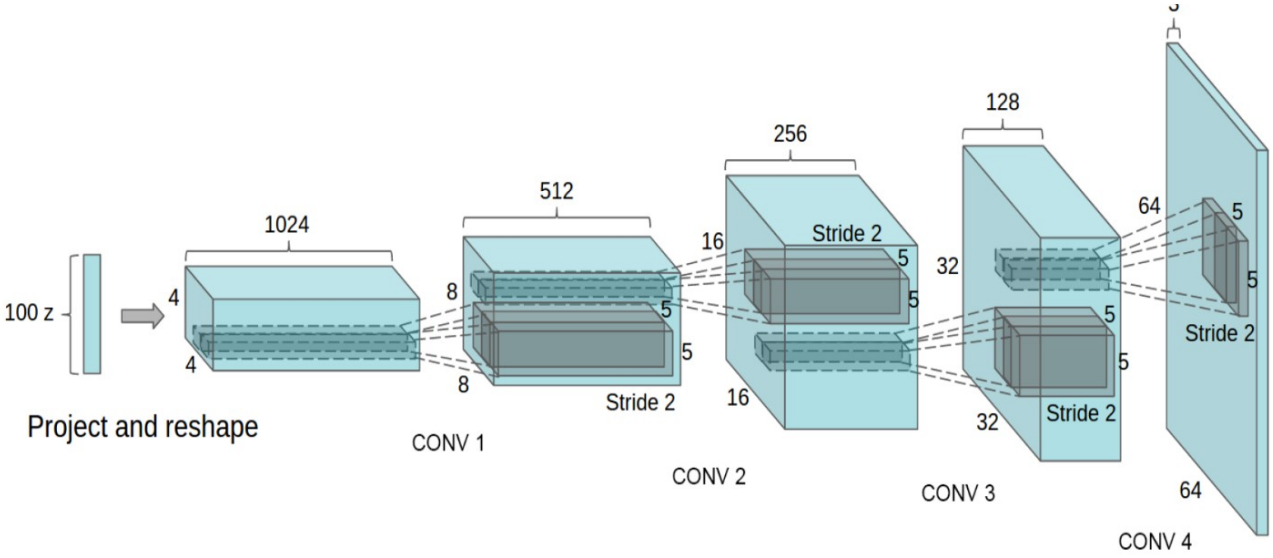
Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.




```python
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. ``(ngf*8) x 4 x 4``
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. ``(ngf*4) x 8 x 8``
            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. ``(ngf*2) x 16 x 16``
            nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. ``(ngf) x 32 x 32``
            nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. ``(nc) x 64 x 64``
        )
```

Možná architektura generátoru (DCGAN)
Možná architektura diskriminátoru (opět DCGAN)

```python
class Discriminator(nn.Module):
    def __init__(self, ngpu):
        super(Discriminator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is ``(nc) x 64 x 64``
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. ``(ndf) x 32 x 32``
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. ``(ndf*2) x 16 x 16``
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. ``(ndf*4) x 8 x 8``
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. ``(ndf*8) x 4 x 4``
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )
```

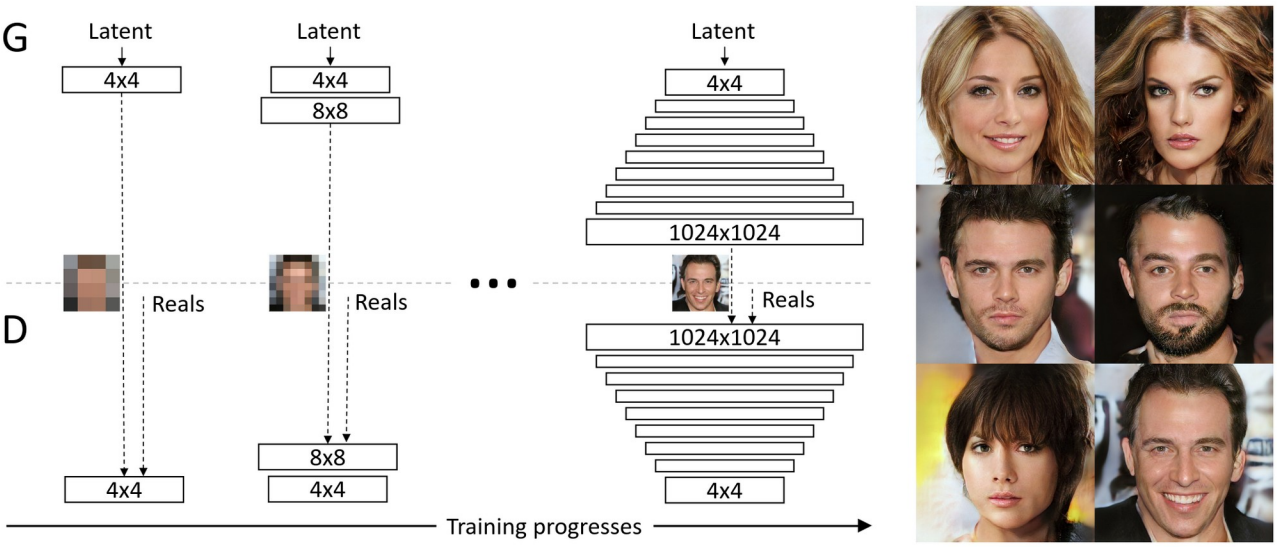
Problémy GANs:

- Učení: Generátor a diskriminátor mohou hned na začátku jeden druhého “přebít”, dalšími epochami už se to nenapraví.
- Generování: Nevíme, jaký obraz z toho při generování vypadne (možná chceme něco konkrétního).

Vylepšování GANs

Progresive GANs

Karras et al.: Progressive growing of GANs for improved quality, stability and variation (2018).



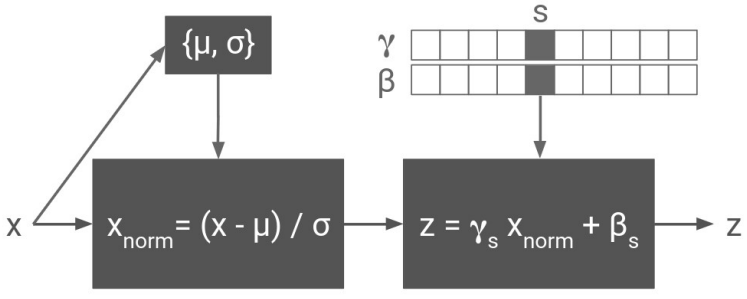
StyleGAN Prerekvizity

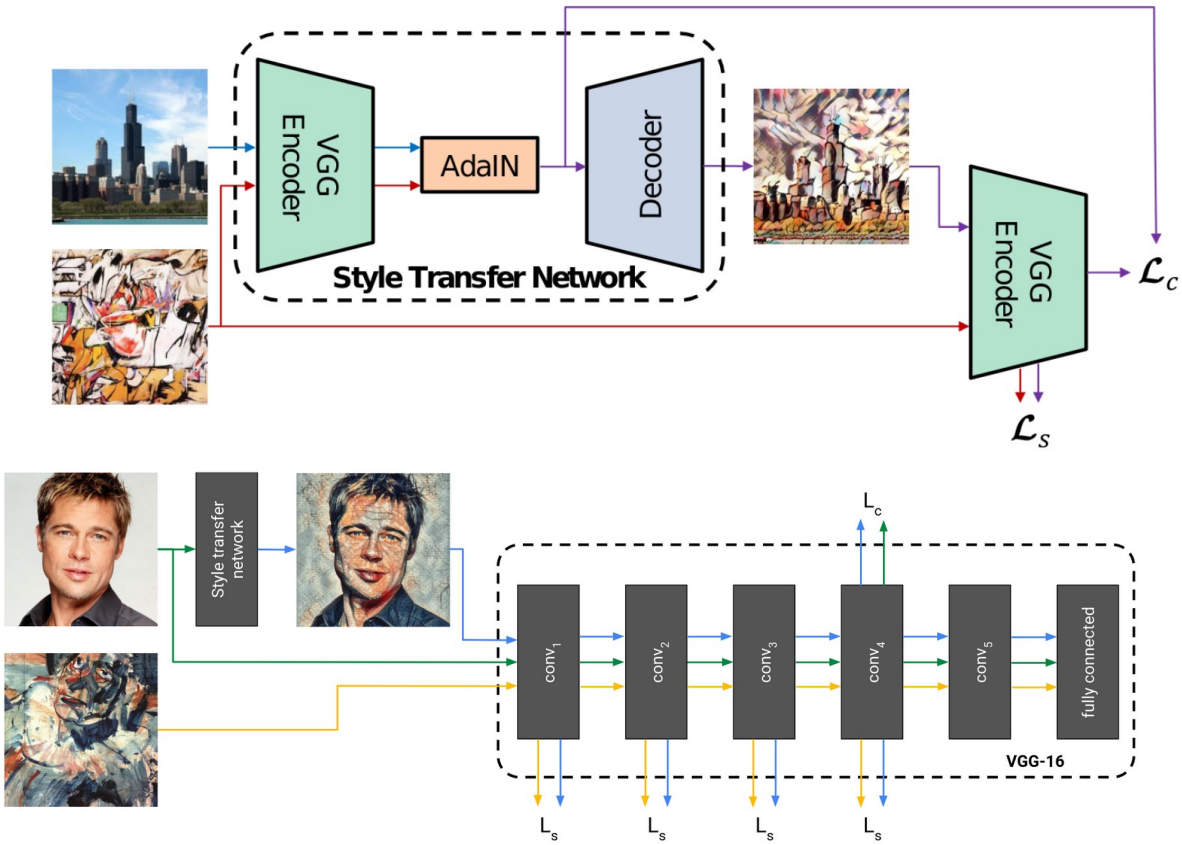
Využívají mimo jiné myšlenku adaptivní normalizace. Ta vznikla v souvislosti s generováním obrazů v různých stylech. Např.: Dumoulin et al.: A Learned Representation For Artistic Style (2017), Xun Huang and Serge Belongie: Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization (2017).

Intuition (Dumoulin) Many styles probably share some degree of computation. This sharing is thrown away by training N networks from scratch when building an N - styles style transfer system.

Idea: To to train a single conditional style transfer network T (c, s) for N styles. The conditional network is given both a content image and the identity of the style to be applied. Then it produces an image corresponding to that style. The open question remains of how that conditioning should be done. Very surprising fact about the role of normalization in style transfer networks: to model a style, it is sufficient to specialize scaling and shifting parameters after normalization to each specific style. In other words, all convolutional weights of a style transfer network can be shared across many styles, and it is sufficient to tune parameters for an affine transformation after normalization for each style.

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s$$





An overview of the style transfer algorithm. The first few layers of a fixed VGG-19 pretrained network encode the content and style images. A content loss Lc and a style loss Ls are determined using the outputs of particular layers.

Dumoulin:

$$\mathcal{L}(s,c,p) = \lambda_s \mathcal{L}_s(p) + \lambda_c \mathcal{L}_c(p)$$

$$\mathcal{L}_s(p) = \sum_{i \in \mathcal{S}} \frac{1}{U_i} \mid\mid G(\phi_i(p)) - G(\phi_i(s)) \mid\mid_F^2$$

$$\mathcal{L}_c(p) = \sum_{j \in \mathcal{C}} \frac{1}{U_j} \mid\mid \phi_j(p) - \phi_j(c) \mid\mid_2^2$$

where φl(x) are the classifier activations at layer l, Ul is the total number of units at layer l and G(φl(x)) is the Gram matrix associated with the layer l activations.

Xun Huang:

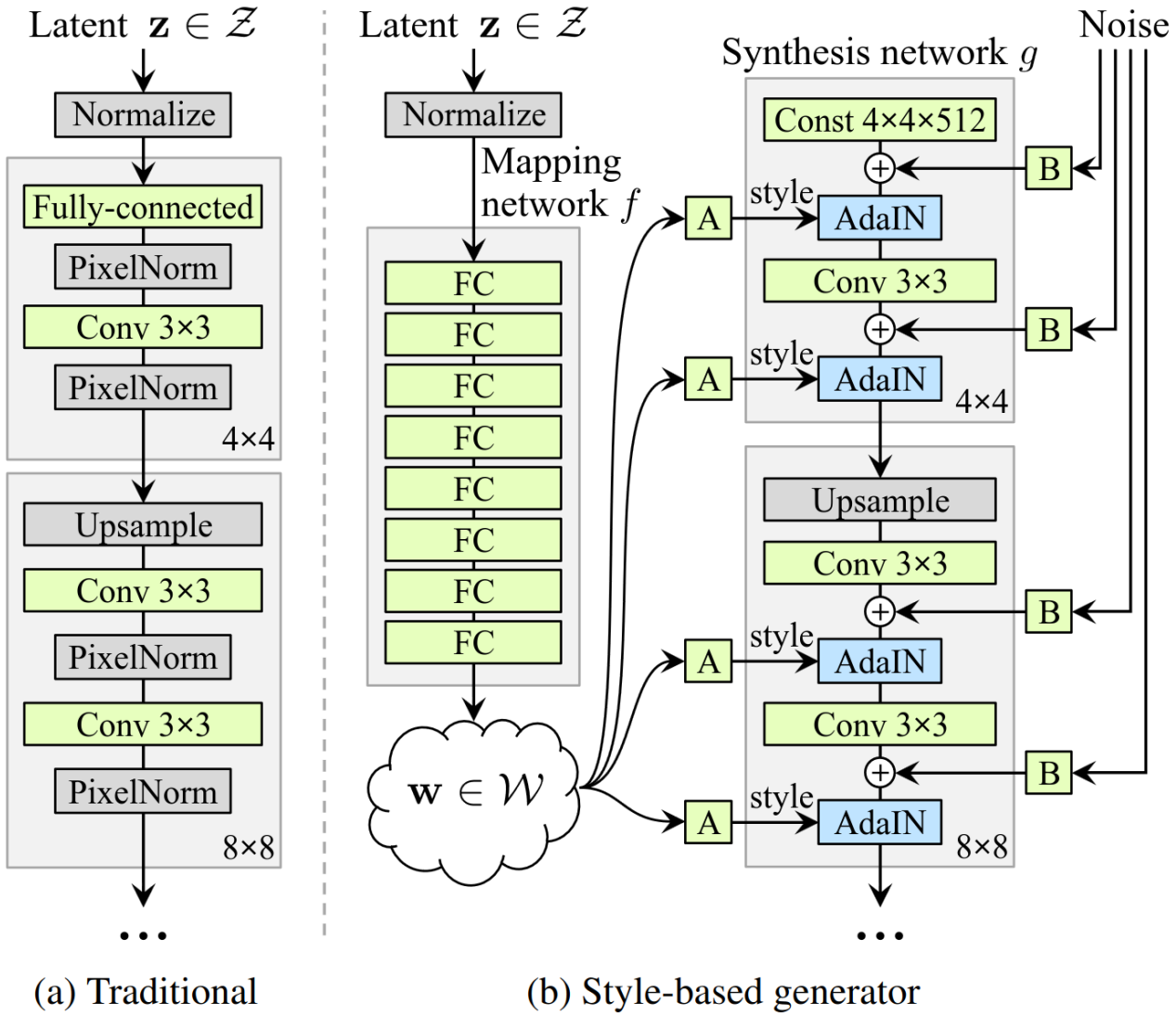
$$\mathcal{L}_c = \|f(g(t)) - t\|_2$$

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\|_2 \quad + \quad \sum_{i=1}^{L} \|\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))\|_2$$

StyleGAN, (StyleGAN1)

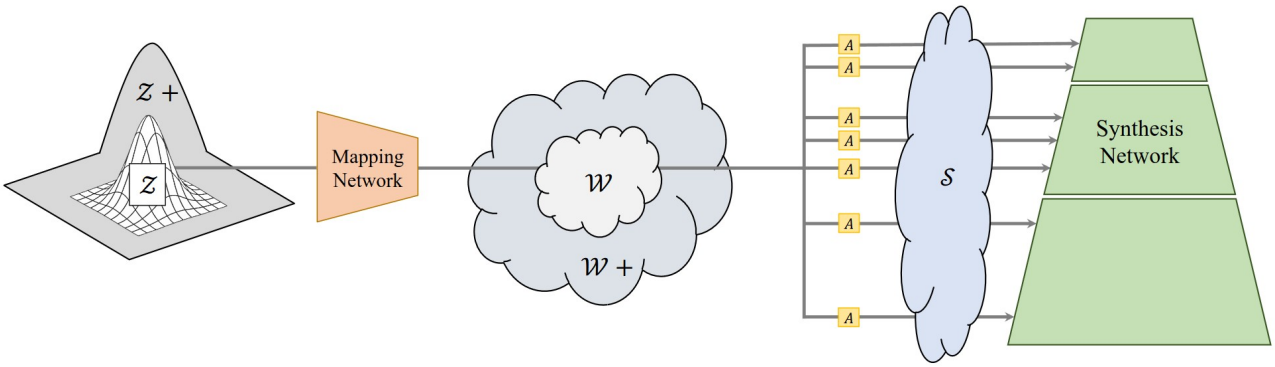
Tero Karras: A Style-Based Generator Architecture for Generative Adversarial Networks (2019).

Jinak je sestaven generátor:



$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Adaptive instance normalization: each feature map x_i is normalized separately, and then scaled and biased using the corresponding scalar components from style y. Thus the dimensionality of y is twice the number of feature maps on that layer.

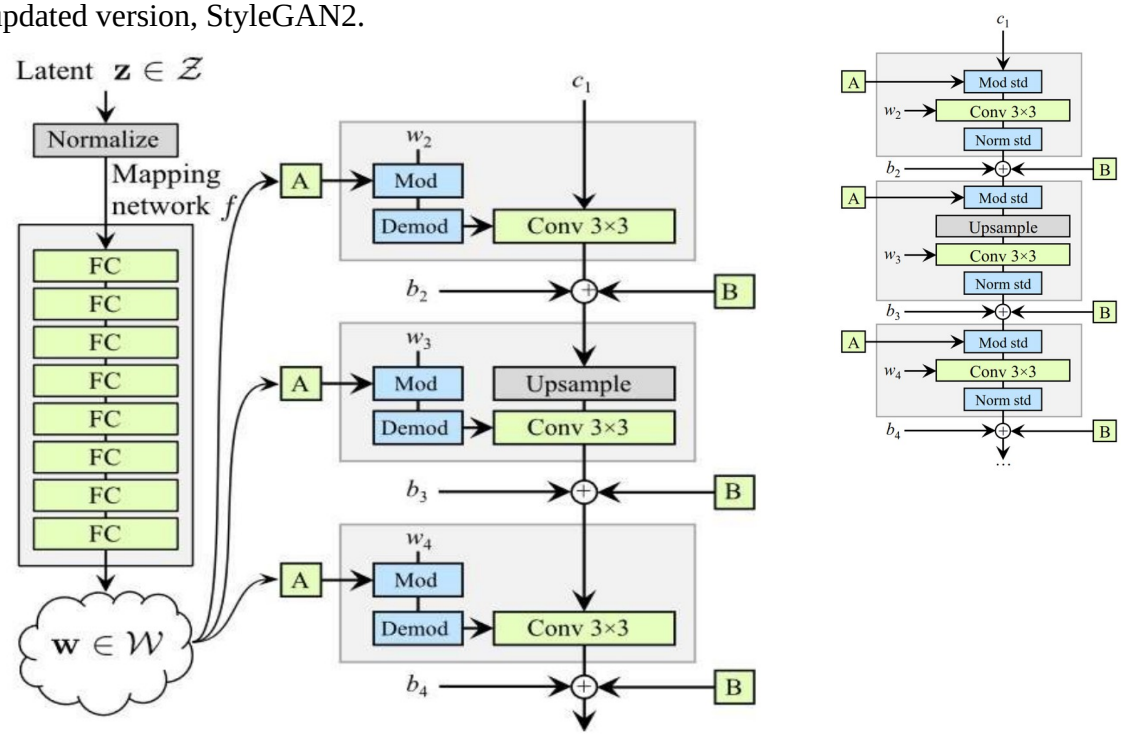


The StyleGAN architecture and its latent spaces. A random latent code z is sampled from the normally distributed latent space Z (on the left), then transformed to the learned latent space W through an MLP mapping, passed through a set of different learned affine transformations (denoted by A) to reach the S space, and finally inserted into the synthesis network. It is common to work in the extended spaces of Z and W, referred to as Z+ and W+, respectively.

StyleGAN2

Karras et al.: Analyzing and Improving the Image Quality of StyleGAN (2020)

StyleGAN was a major breakthrough towards the generation of high-resolution face images that looked very natural. Yet, the generated images tended to contain minor, but systematic artifacts, such as blobs or droplets. A careful analysis of this phenomenon enabled the development of an updated version, StyleGAN2.



Replacing the former AdaIN operations by a direct rescaling of the convolutional weights, again based on the the style parameter output associated with w ∈ W , followed by a normalization by the
standard deviation over the scaled weights.

Image editing of images from StyleGAN

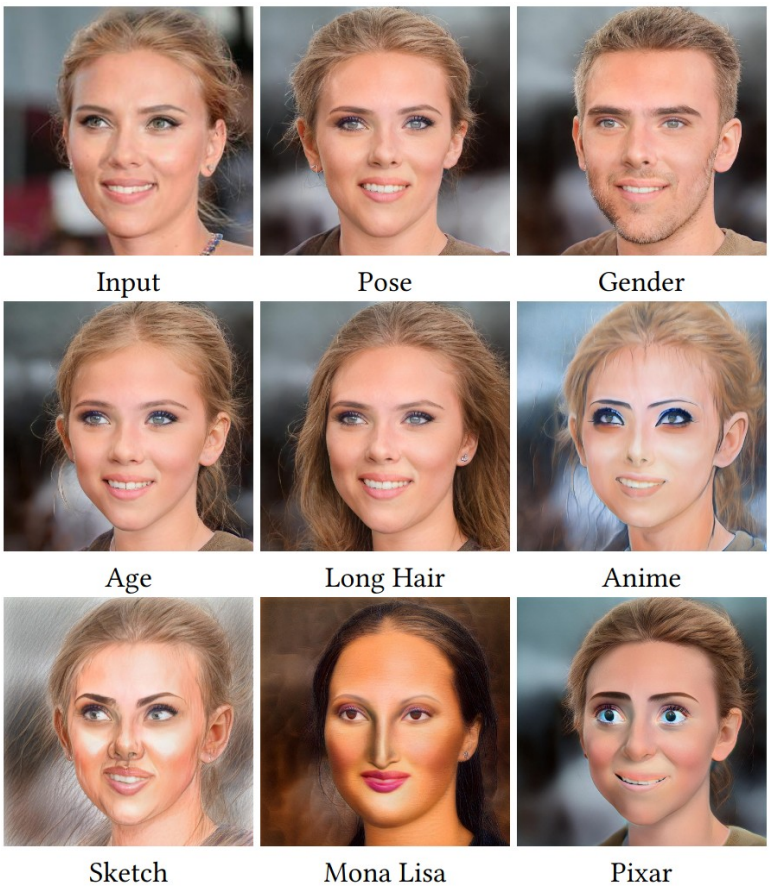
The latent space of the StyleGANs is known to be well disentangled, which means that one can manipulate semantic features without changing other characteristics so much. The process to obtain this latent vector is called GAN inversion. We can use the work *Abdal R. et al.: Image2stylegan: How to embed images into the stylegan latent space? a*s an example.

$$w^* = \min_w L_{percept}(G(w), I) + \frac{\lambda_{mse}}{N}\|G(w) - I\|_2^2$$

where I ∈ R n×n×3 is the input image, G(·) is the pretrained generator, N is the number of scalars in the image (i.e. N = n × n × 3), w is the latent code to optimize.

$$L_{percept}(I_1, I_2) = \sum_{j=1}^{4} \frac{\lambda_j}{N_j}\|F_j(I_1) - F_j(I_2)\|_2^2$$

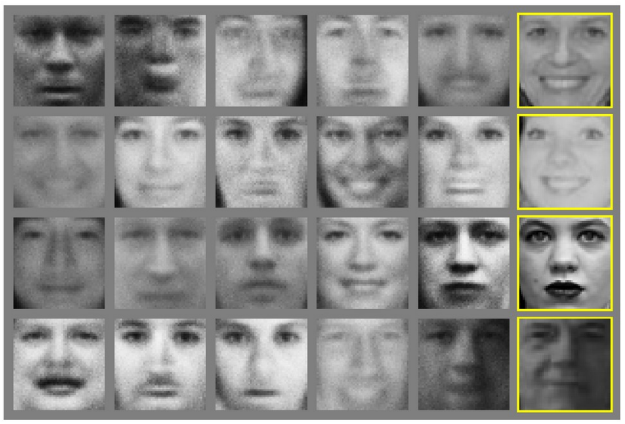
where I1 , I2 ∈ R n×n×3 are the input images, Fj is the feature output of VGG-16 layers conv1_1, conv1_2, conv3_2, and conv4_2 respectively, N j is the number of scalars in the jth layer output.



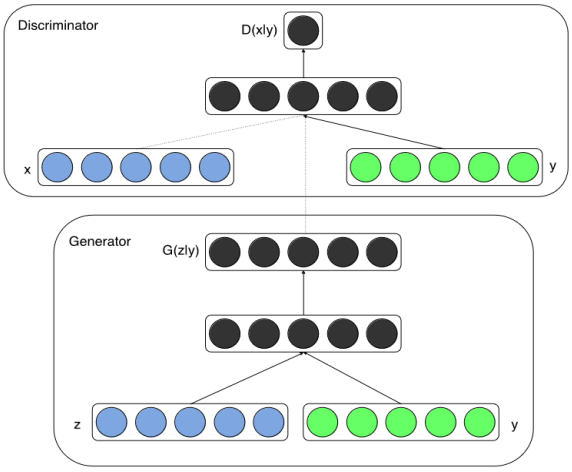
Editing a real image of Scarlett Johansson (on the top left) with StyleGAN. (AMIT H. BERMANO et al.:State-of-the-Art in the Architecture, Methods and Applications of StyleGAN)

Conditional Generative Adversarial Nets

Mehdi Mirza, Simon Osindero: Conditional Generative Adversarial Nets (2014).



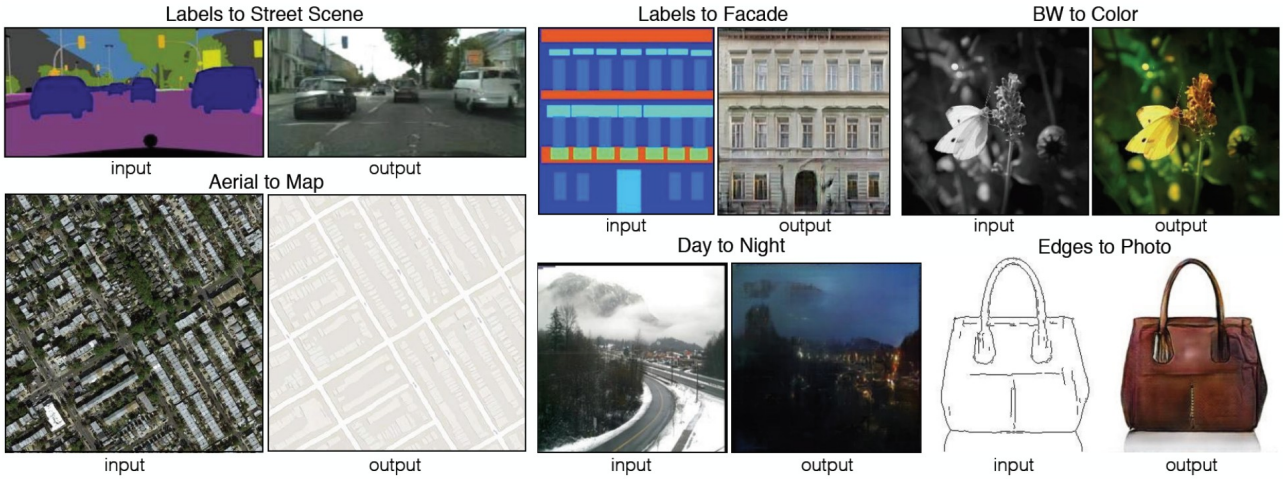
Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information, such as class labels or data from other modalities. We can perform the conditioning by feeding y into the both the discriminator and generator as additional input layer.

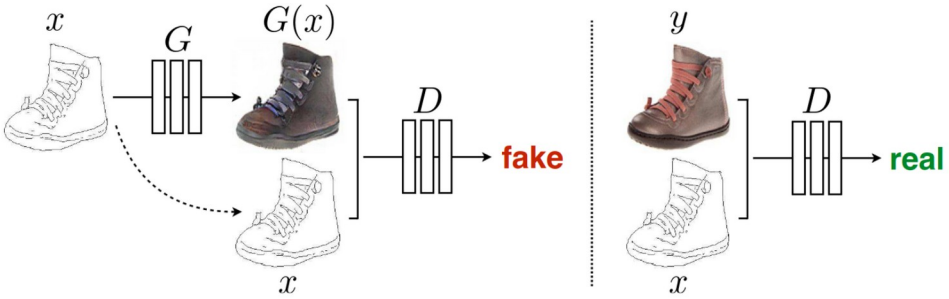


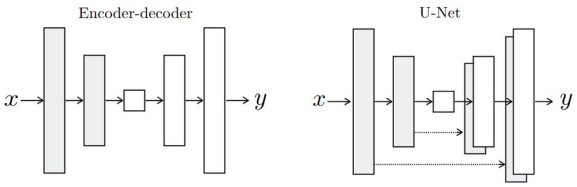
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

Image-to-Image Translation with Conditional Adversarial Nets

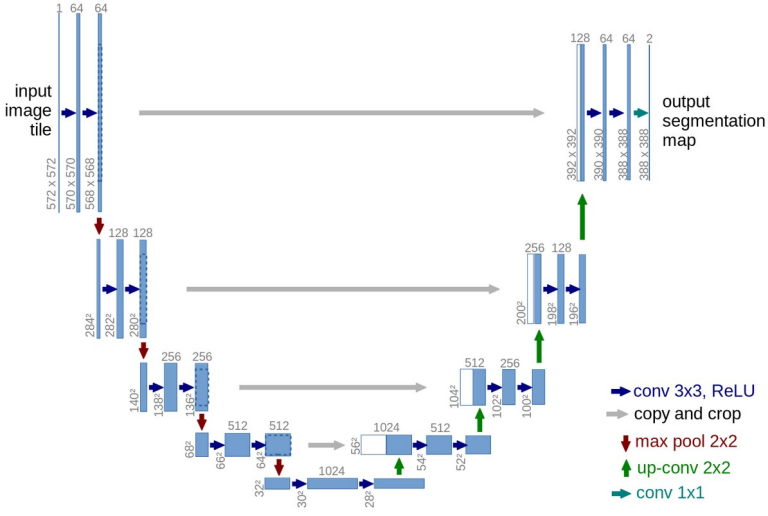
Isola P et al.: Image-to-Image Translation with Conditional Adversarial Networks (2017)

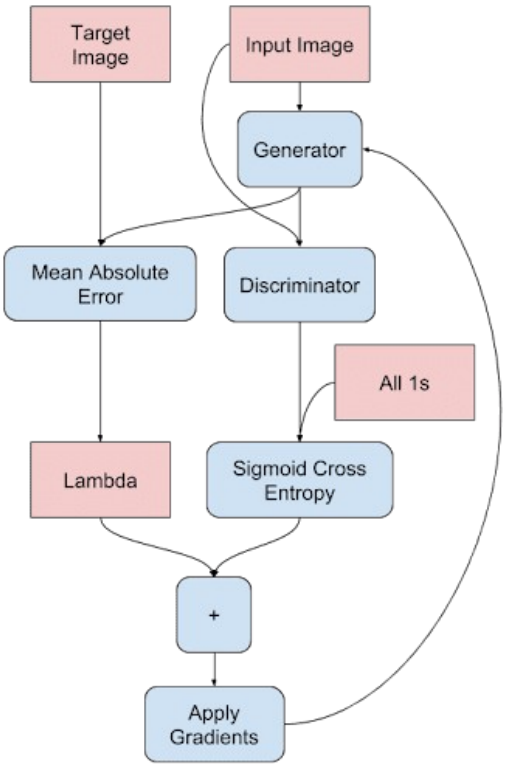






Two choices for the architecture of the generator. The “U-Net” [Ronneberger O. et al.: U-Net: Convolutional Networks for Biomedical Image Segmentation] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.





Conditional GANs learn a mapping from observed image x and random noise vector z, to y, G : {x, z} → y.

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned}$$

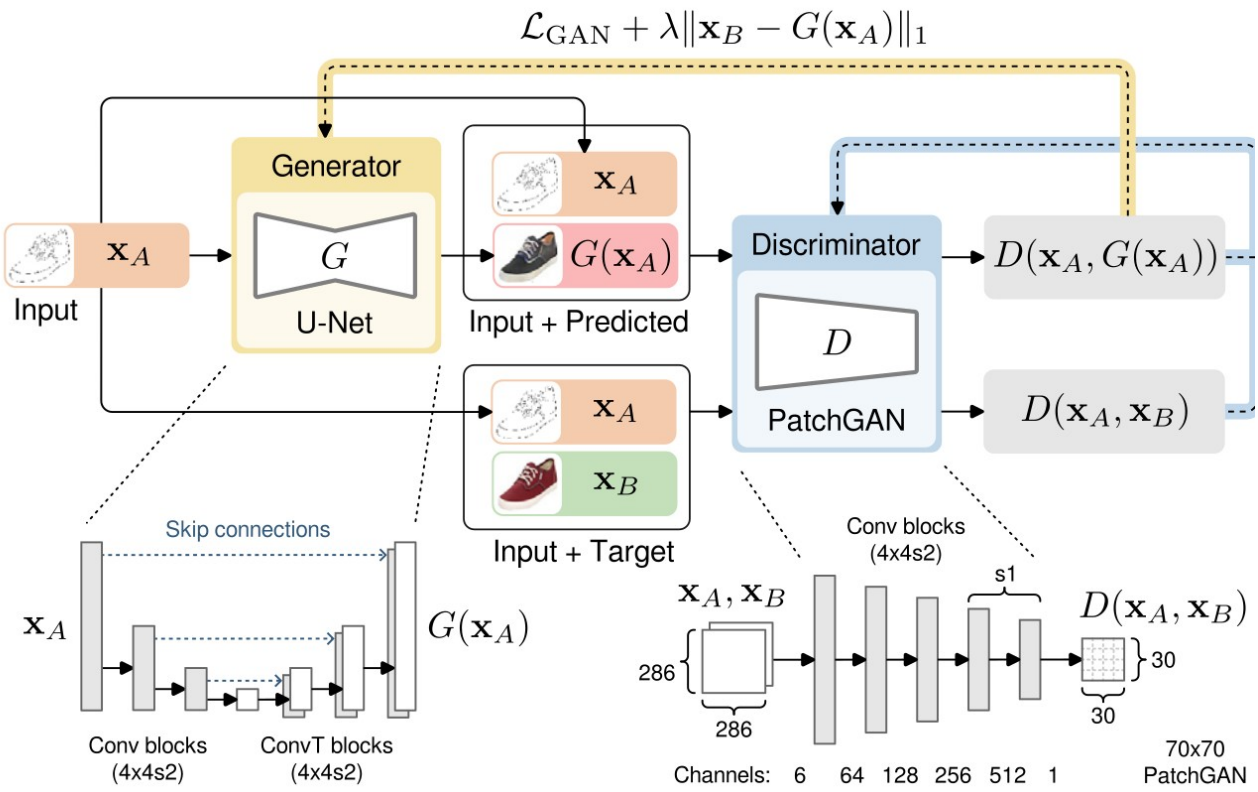
Moreover, the generator is tasked to not only to fool the discriminator but also to be near the ground truth output in an L2 or L1 sense.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

The final objective is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Trochu vylepšená architektura (PatchGAN)



The pix2pix architecture is based on the conditional GAN (CGAN). The image xA serves as the condition for the generator and discriminator, and no noise input is used on the generator. Instead of a single scalar, the PatchGAN discriminator outputs a matrix, in which each element represents a patch of the input image.

Text-to-Image Synthesis

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. “Generative adversarial text to image synthesis. ICML (2016).

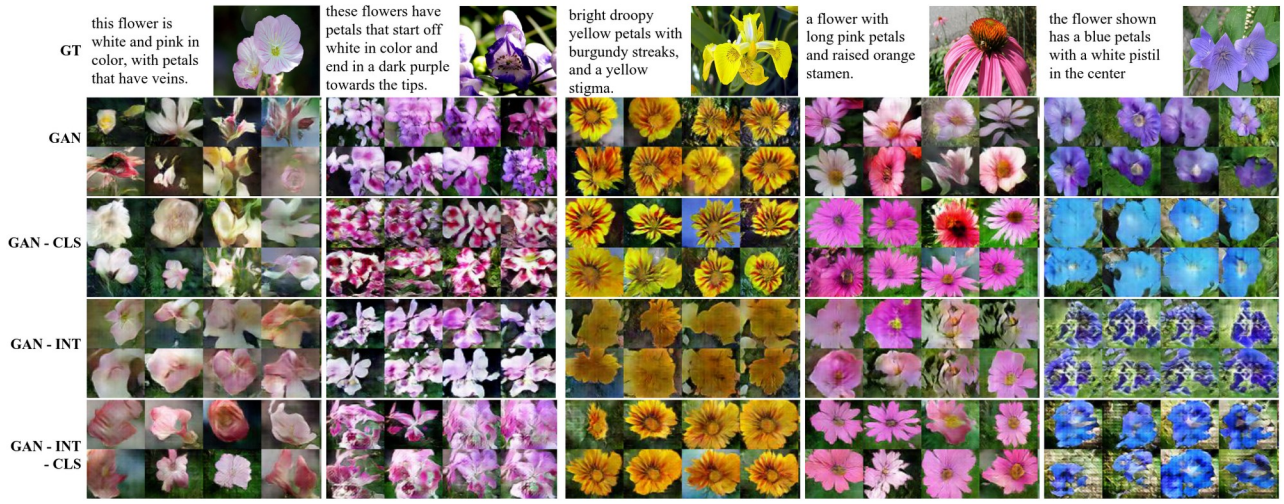
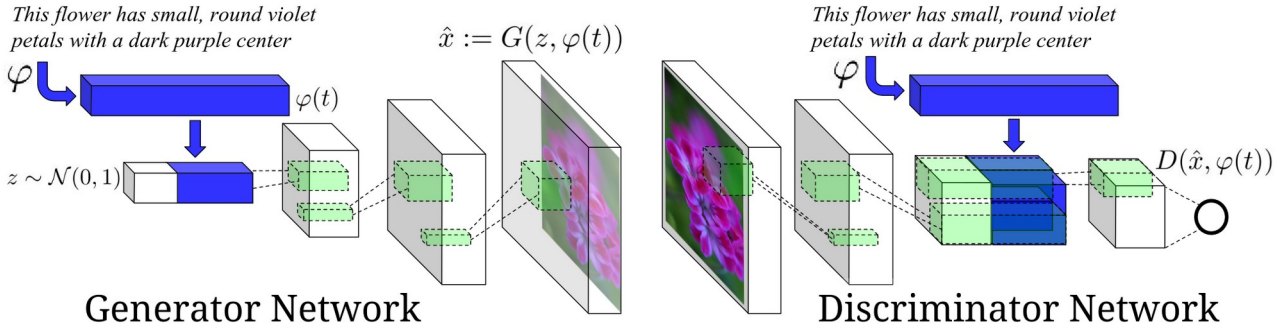
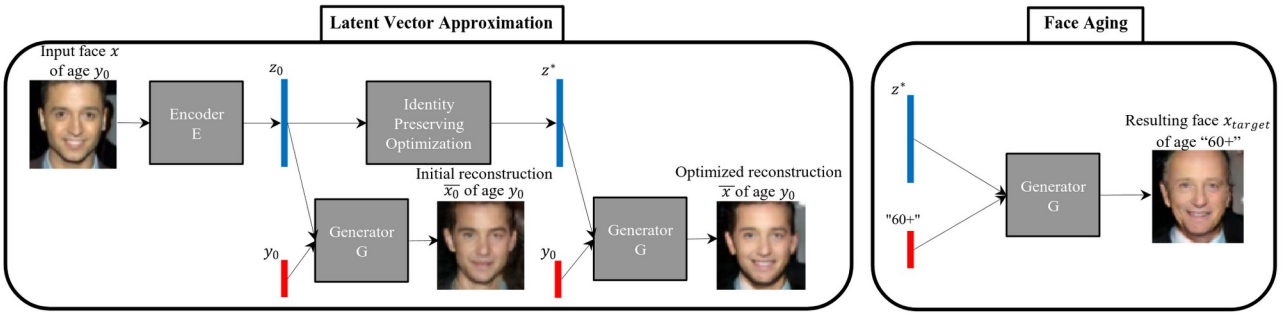


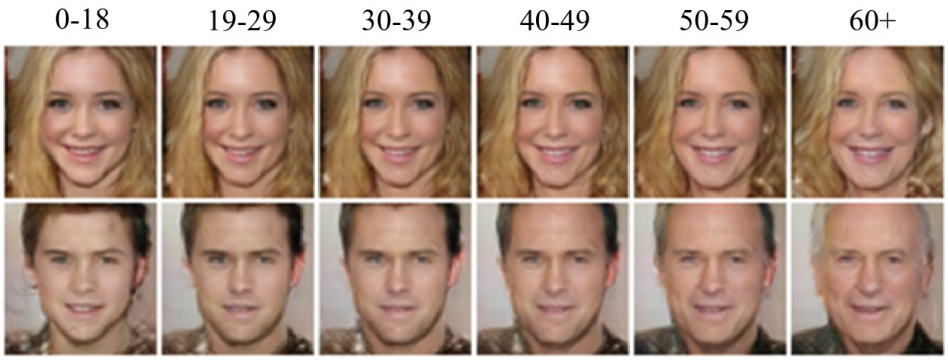
Figure 4. Zero-shot generated flower images using GAN, GAN-CLS, GAN-INT and GAN-INT-CLS. All variants generated plausible images. Although some shapes of test categories were not seen during training (e.g. columns 3 and 4), the color information is preserved.



Face Aging

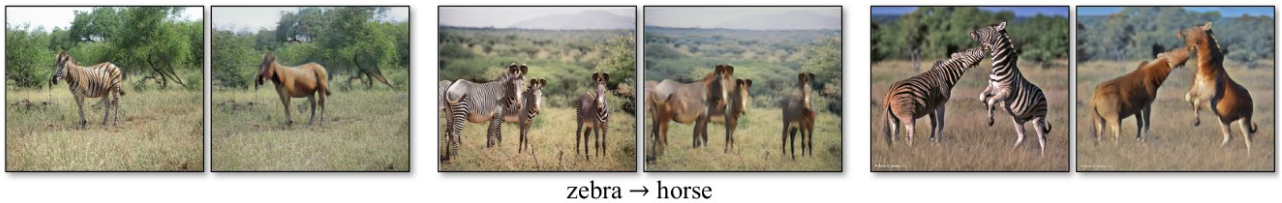
Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "Face Aging With Conditional Generative Adversarial Networks". arXiv preprint arXiv:1702.01983.



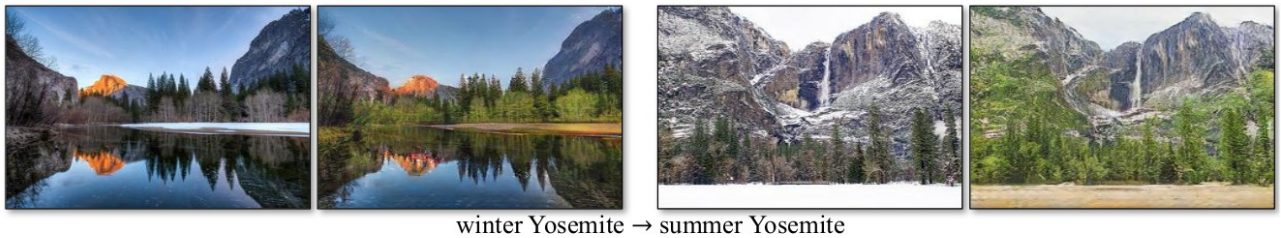


Vyměňování něco a něco (někoho za někoho)

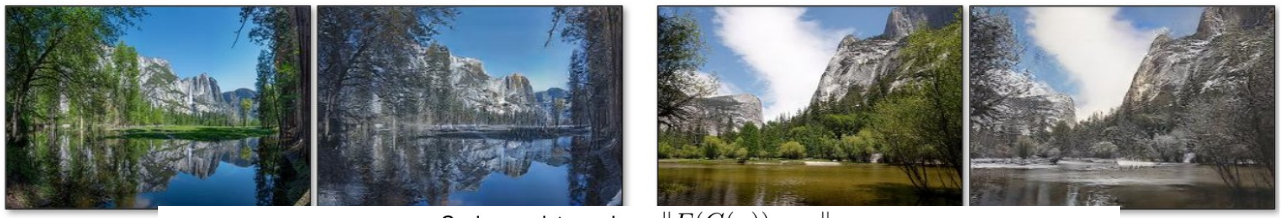
Jun-Yan Zhu et al.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2020)



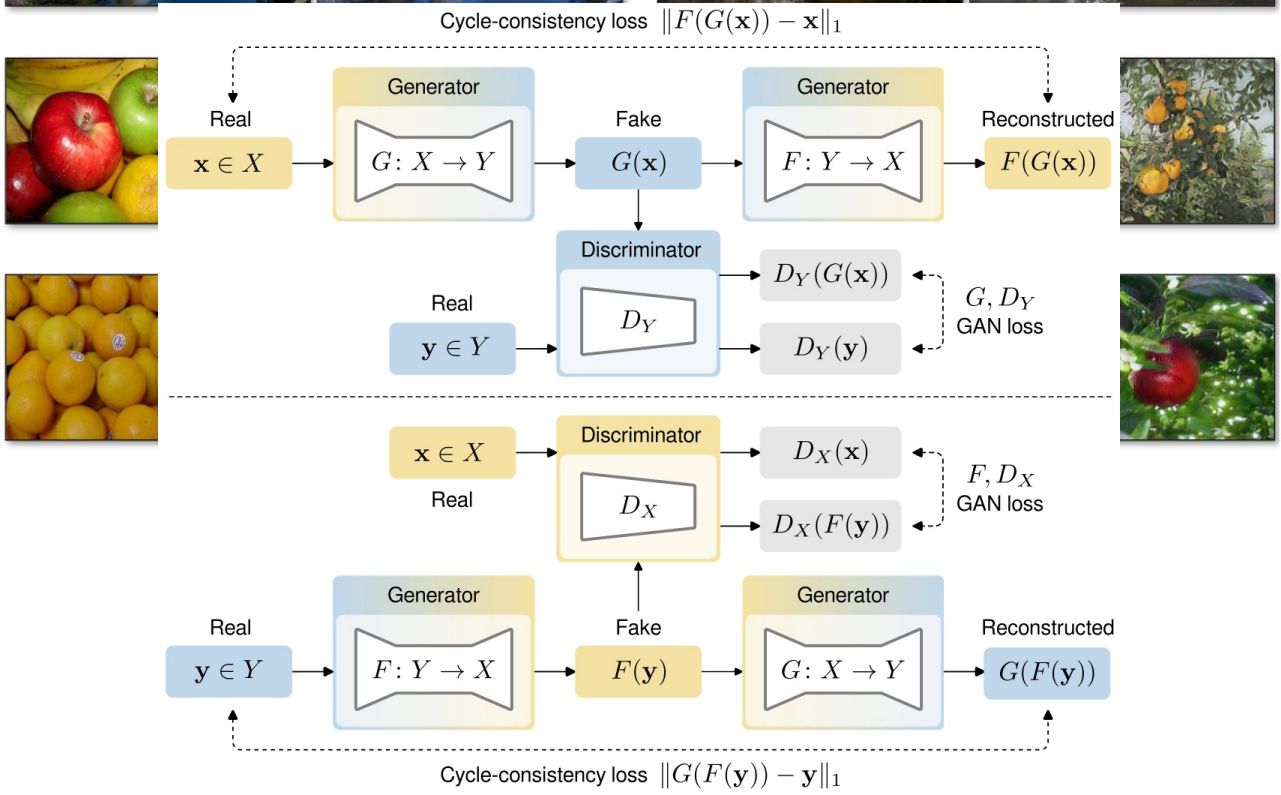
zebra → horse

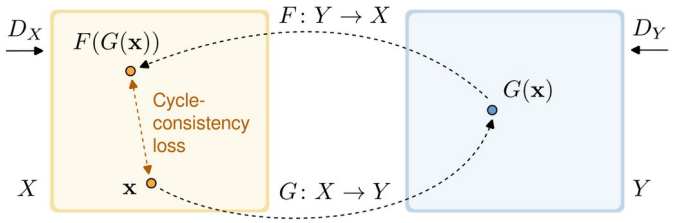


winter Yosemite → summer Yosemite



Cycle-consistency loss $\|F(G(\mathbf{x})) - \mathbf{x}\|_1$

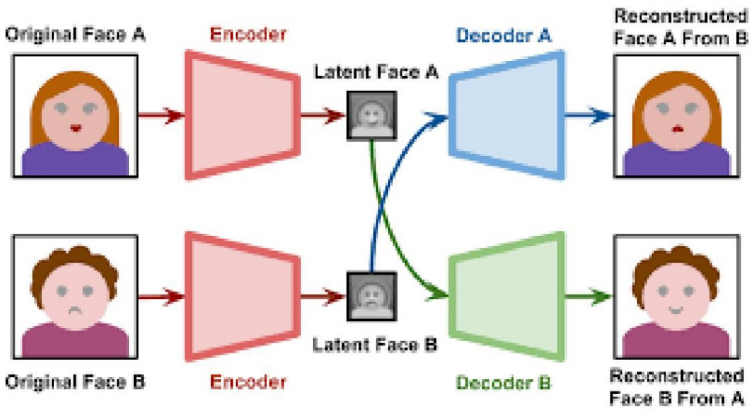




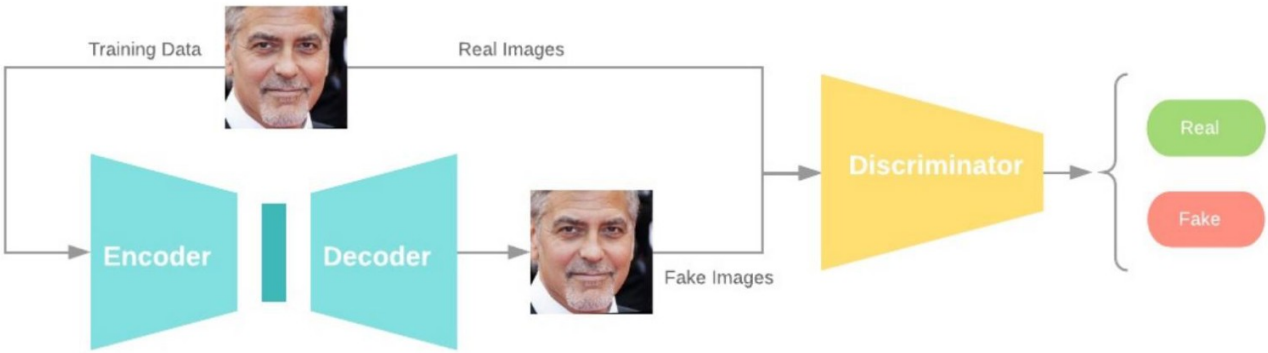
Face swapping with GANs

Toto je vcelku přímočará myšlenka převzatá z
http://cs230.stanford.edu/projects_spring_2019/reports/18681213.pdf
Současné přístupy bývají složitější

In order to swap person A and B's faces, we trained two autoencoder models (model A and model B), one for each person. During the training, model A and B share a same (common) encoder but have separate decoders.
In this way, both person A and B's face features are encoded at the same shared layers, and both decoder A and B knows how to decode person A and B's face from it.
When we want to make B's face on A's image, we use A's image as input data to perform prediction on model A, the key change is we swap model B's decoder with model A's decoder, so the output images have B's face on A's image. See the conceptual graph below:

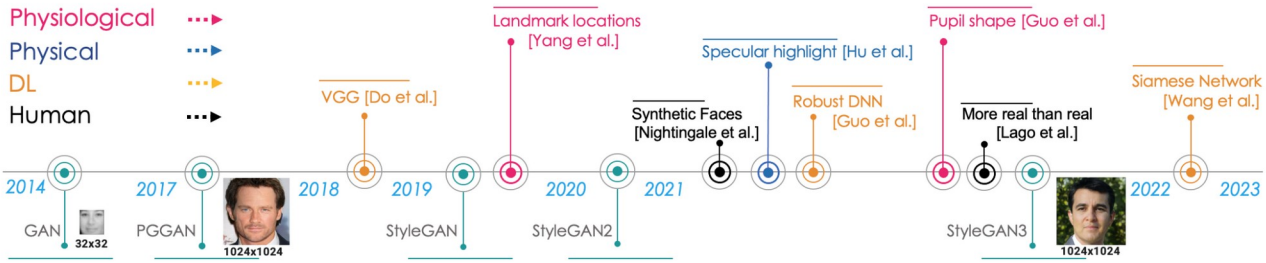


Beside exploring different hyper-parameters, we also introduce GAN algorithm to improve autoenoder's training performance. The structure is simple, we choose training data as real image and autoencoder output image as fake image, use them as intput to train a CNN based discriminator, see below graph for concept.



Detecting GAN Generated Face Images

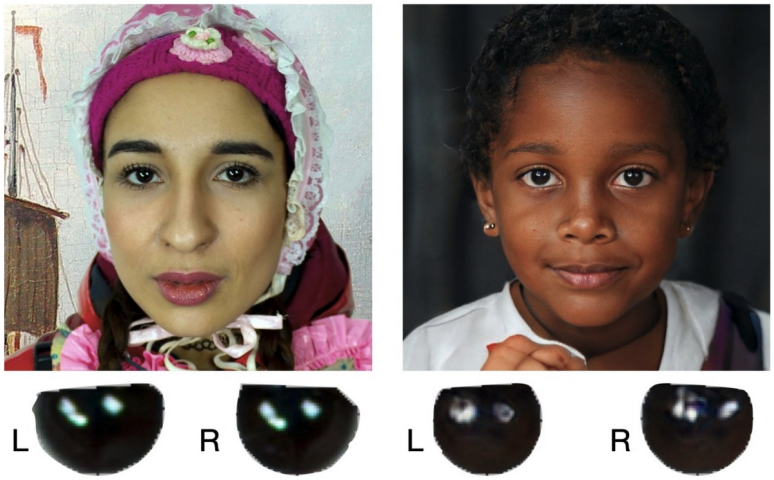
Xin Wang et al.: GAN-generated Faces Detection: A Survey and New Perspectives (2023)



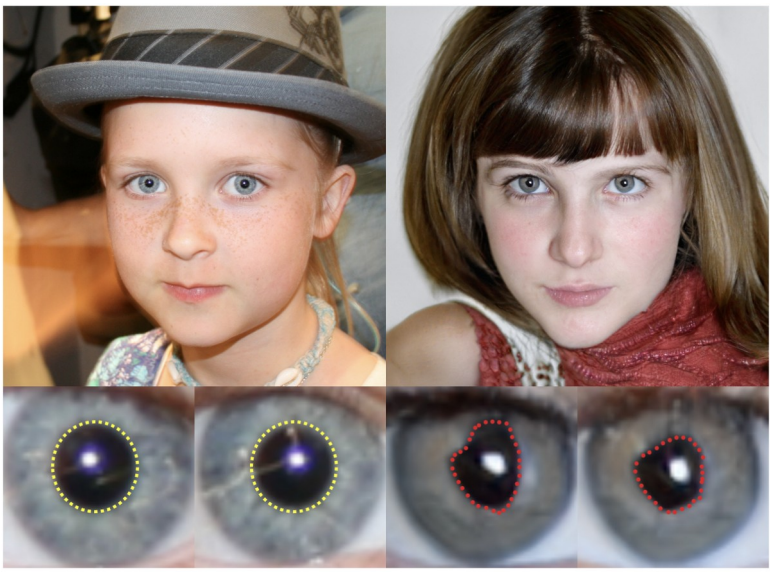
A brief chronology for GAN-faces generation and detection works.

Deep Learning-based Methods (budou uvedeny dále)

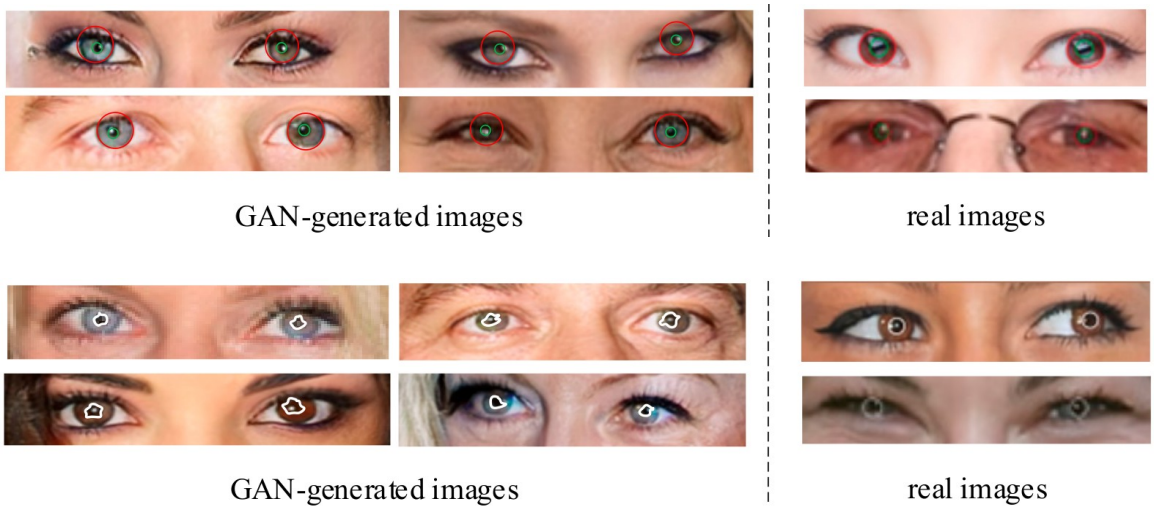
Physical-based Methods



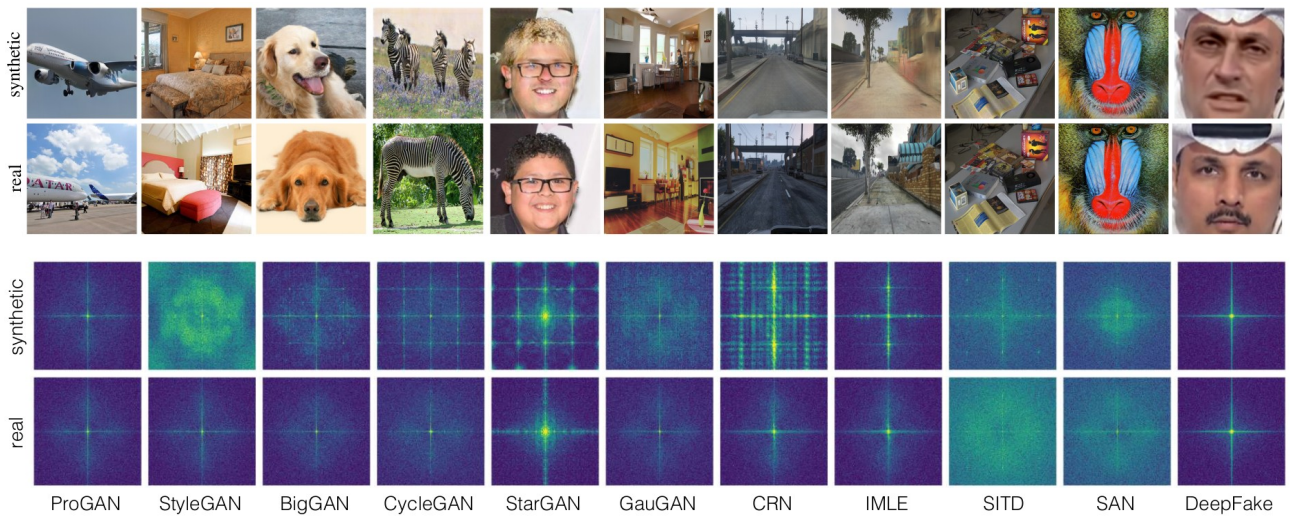
Top: Corneal specular highlights for a real human face (left) and a GAN-face (right) [42]. Bottom: The corneal regions are isolated and scaled for better visibility. Note that the corneal specular highlights for the real face have strong similarities while those for the GAN-faces are different.



Pupils of real (left) human face and GAN-face (right). Note that the pupils for the real eyes have strong circular shapes (yellow) while those for the GAN-generated pupils are with irregular shapes (red).



Sheng-Yu Wang et al.: CNN-generated images are surprisingly easy to spot... for now (2020)



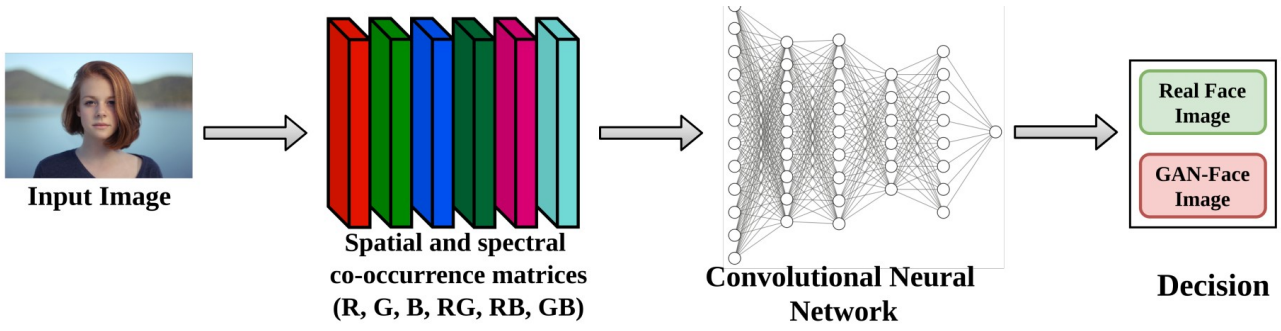
Frequency analysis on each dataset. The average spectra of each high-pass filtered image, for both the real and fake images. We observe periodic patterns (dots or lines) in most of the synthetic images, while BigGAN and ProGAN contains relatively few such artifacts.

The average power spectra of natural images tend to follow a $1/f^{\alpha}$ curve, where f is the frequency along a given axis and $\alpha \approx 2$. The images generate by StyleGAN, however, exhibit artifacts throughout the spectrum. In comparison to the spectra of natural images, StyleGAN-generated images contain strong high frequencies components, as well as generally higher magnitudes throughout the spectrum. Especially notably is the grid-like pattern. We hypothesize that the artifacts found for GAN-generated images in the frequency domain stem from their employed upsampling operations. We make this more concrete in the following: The generator of a GAN forms a mapping from a low-dimensional latent space to the higher-dimensional data space. In practice, the dimensionality of the latent space is much lower than the dimensionality of the data space, e.g., the generator of the StyleGAN-instance which generated the images presented in Figure 1 defines a mapping G: R100 $\rightarrow$ R1024×1024. In typical GAN architectures, the latent vector gets successively upsampled until it reaches the final output dimension.

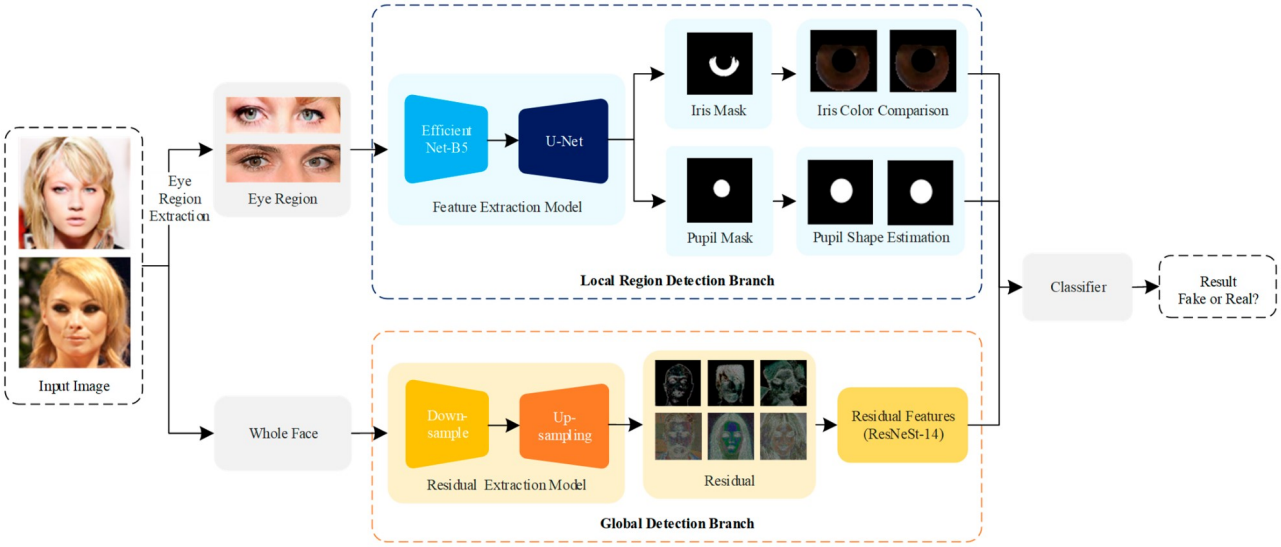
Ehsan Nowroozi et al.: Detecting High-Quality GAN-Generated Face Images using Neural Networks (2022)

For an image with p different pixel values, the $p \times p$ co-occurrence matrix **C** is defined over an . image $m \times n$, parameterized by an offset (Δx, Δy), as:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m} \begin{cases} 1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise} \end{cases}$$



Ziyu Xue et al.: Global–Local Facial Fusion Based GAN Generated Fake Face Detection (2023)


Input Image
Eye Region Extraction
Eye Region
Efficient Net-B5
U-Net
Feature Extraction Model
Iris Mask
Iris Color Comparison
Pupil Mask
Pupil Shape Estimation
Local Region Detection Branch
Whole Face
Down-sample
Up-sampling
Residual Extraction Model
Residual
Residual Features (ResNeSt-14)
Global Detection Branch
Classifier
Result Fake or Real?

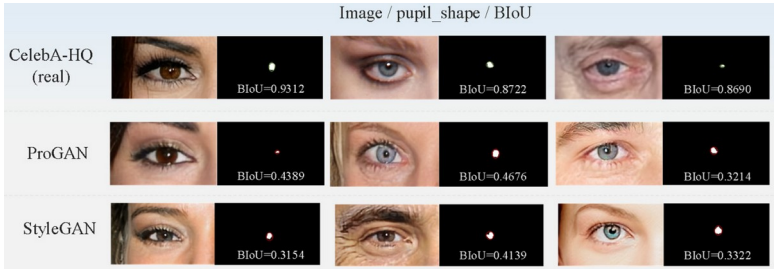


Image / pupil_shape / BIoU
CelebA-HQ (real)
BIoU=0.9312
BIoU=0.8722
BIoU=0.8690
ProGAN
BIoU=0.4389
BIoU=0.4676
BIoU=0.3214
StyleGAN
BIoU=0.3154
BIoU=0.4139
BIoU=0.3322

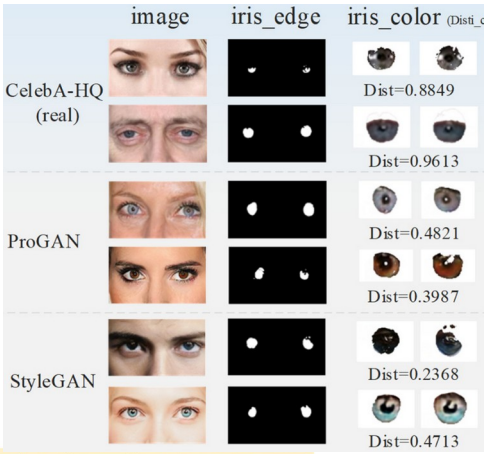
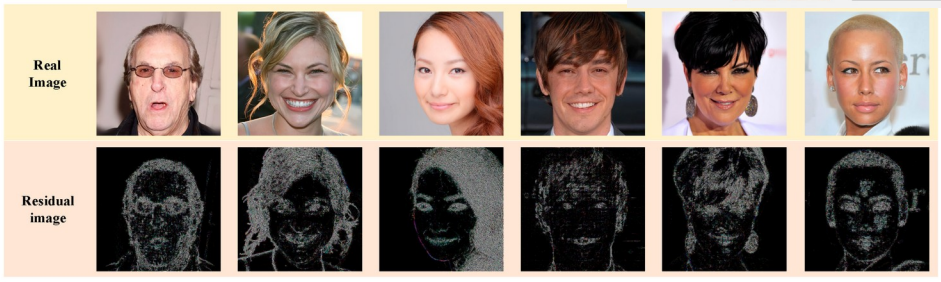
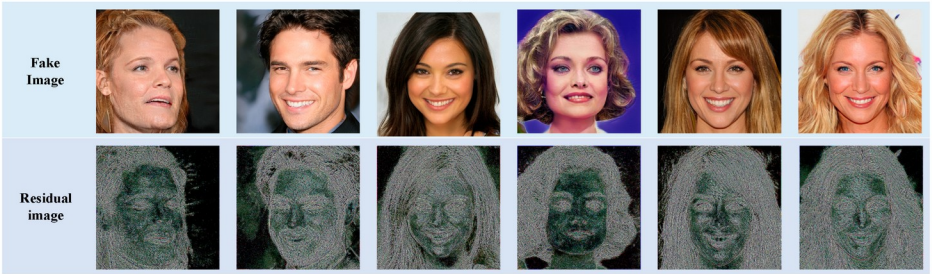


image
iris_edge
iris_color
CelebA-HQ (real)
Dist=0.8849
Dist=0.9613
ProGAN
Dist=0.4821
Dist=0.3987
StyleGAN
Dist=0.2368
Dist=0.4713

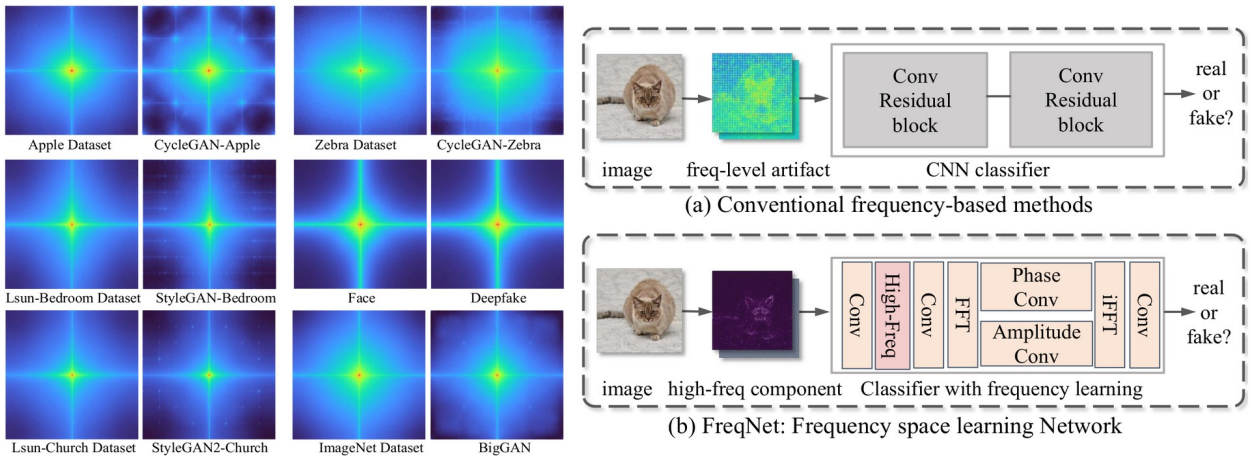

Real Image
Residual image

(a)

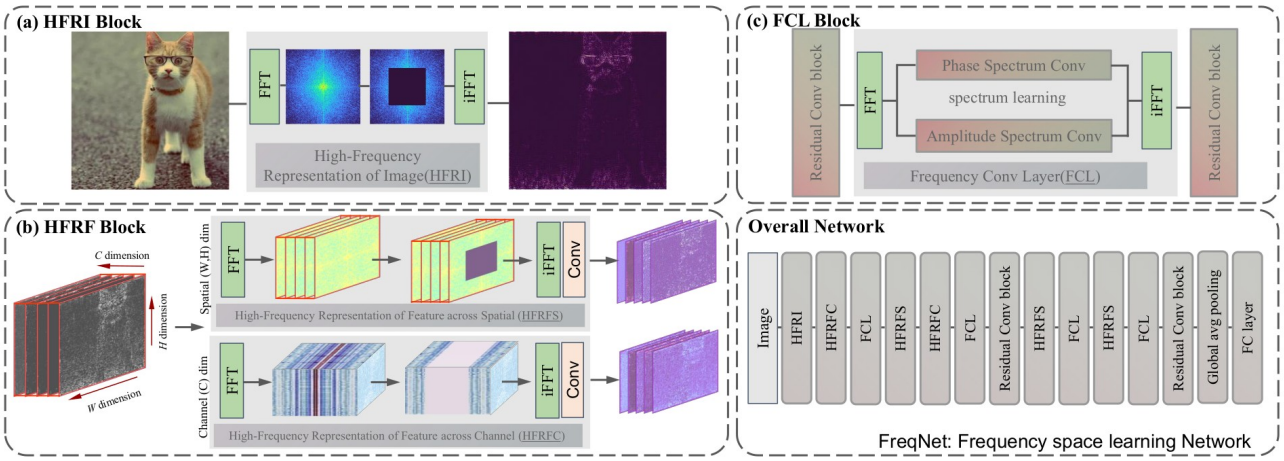

Fake Image
Residual image

(b)

Chuangchuang Tan et al.: Frequency-Aware Deepfake Detection: Improving Generalizability through Frequency Space Learning (2024)



Obrázek vlevo nahoře: Frequency analysis on various sources. This mean FFT spectrum computation involves averaging over 2000 images, following the methodology detailed in (Frank et al. 2020).



Architecture of FreqNet for generalizable deepfake detection. To augment the capacity for generalization, our FreqNet focuses on the enhancement of frequency spectrum information, prioritizing frequency domain learning within the classifier, consisting of (a) High-Frequency Representation of Image (HFRI), (b) High-Frequency Representation of Feature(HFRF), and (c) Frequency Conv Layer (FCL).

https://github.com/chuangchuangtan/freqnet-deepfakedetection

Tianbo Zhai et al.: Learning spatial-frequency interaction for generalizable deepfake detection (2024)

The authors combine combine spatial and frequency domain information for forgery detection using using attention mechanisms, significantly enhancing deepfake detection performance.

Ukázat článek.

Různé dodatky:

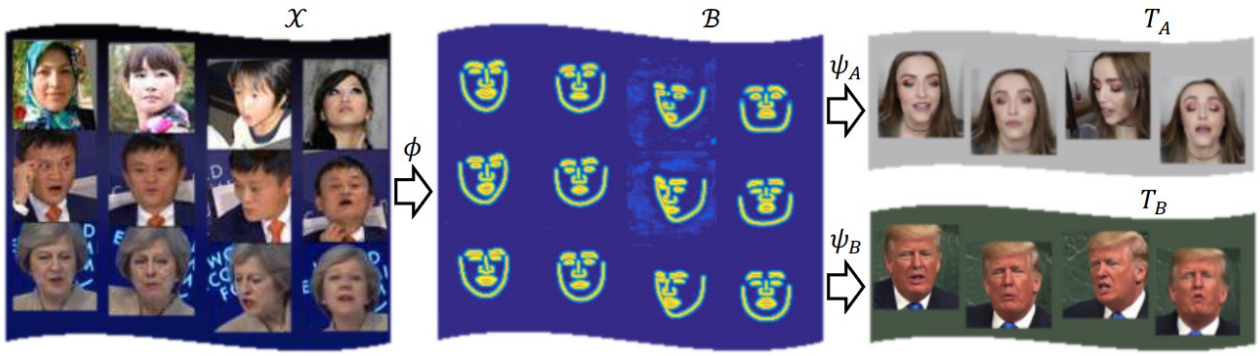
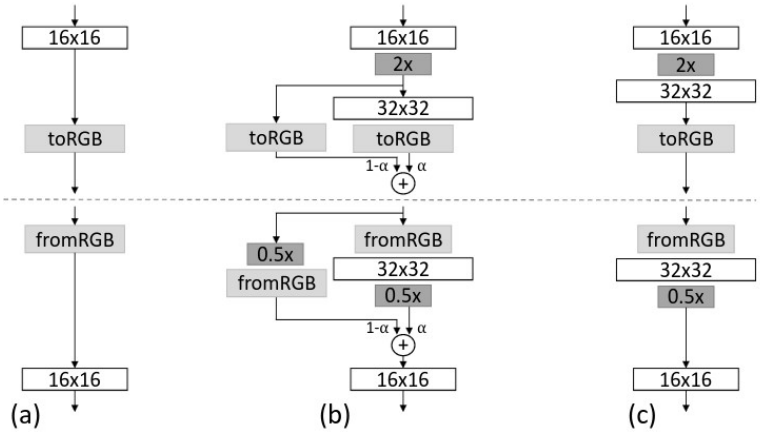


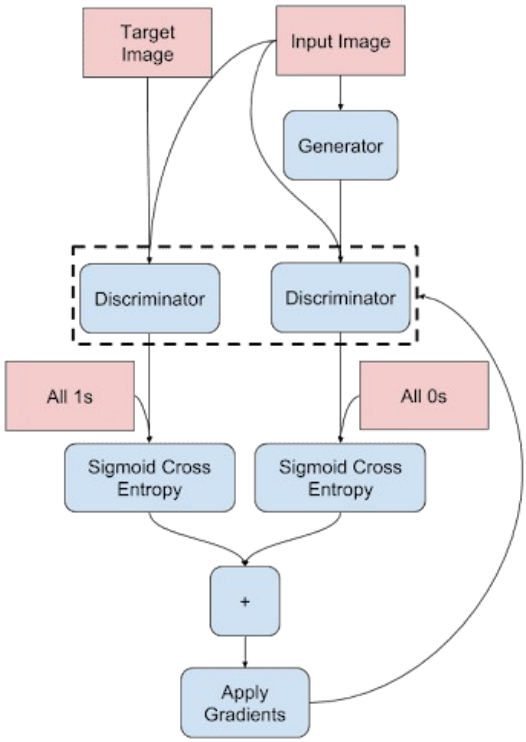
Table 6: Overview of representative forgery detection methods. Notations: ① FF++, ② DFDC, ③ Celeb-DF, ④ Deeperforensics, ⑤ Self-build, ⑥ UADFV, ⑦ Celeb-HQ, ⑧ DFDCp, ⑨ FFHQ, ⑩ DFD.

	Method	Venue	Train	Test	Highlight
Space Domain	Gram-Net [175]	CVPR'20	⑦⑨	⑦⑨	The method posits that genuine faces and fake faces exhibit inconsistencies in texture details.
	Face X-ray [153]	CVPR'20	①	①②③⑩	Focusing on boundary artifacts of face fusion for forgery detection.
	Zhao *et al.* [342]	CVPR'21	①	①②③	A texture enhancement module, an attention generation module, and a bilinear attention pooling mod -ule are proposed to focus on texture details.
	Nirkin *et al.* [206]	TPAMI'21	①	①②③	Detecting swapped faces by comparing the facial region with its context (non-facial area).
	SBIs [235]	CVPR'22	①	②③⑧⑩	The belief that the more difficult to detect forged faces typically contain more generalized traces of forg -ery can encourage the model to learn a feature representation with greater generalization ability.
	RECCE [21]	CVPR'22	①	①②③ [363]	Reconstruction learning on real samples to learn common compressed representations of real images.
	LGrad [249]	CVPR'23	⑤	⑤	The gradient is utilized to present generalized artifacts that are fed into the classifier to determine the truth of the image.
	NoiseDF [267]	AAAI'23	①	①②③④	Extracting noise traces and features from cropped faces and background squares in video frames.
	Ba *et al.* [8]	AAAI'24	①②③	①②③	Multiple non-overlapping local representations are extracted from the image for forgery detection. A local information loss function, based on information bottleneck theory, is proposed for constraint.
Time Domain	Yang *et al.* [302]	ICASSP'19	⑥	⑥	Focusing on the inconsistency in the head pose in videos by comparing the estimated head pose using all facial landmarks with the one estimated using only the landmarks in the central region.
	FTCN [347]	ICCV'21	①	①②③④ [152]	It is believed that most face video forgeries are generated frame by frame. As each altered face is inde -pendently generated, this inevitably leads to noticeable flickering and discontinuity.
	LipForensics [85]	CVPR'21	①	①②③	Concern about temporal inconsistency of mouth movements in videos.
	M2TR [260]	ICMR'22	①	①②③⑩	Capturing local inconsistencies at different scales for forgery detection using a multiscale transformer.
	Gu *et al.* [72]	AAAI'22	①	①②③ [363]	By densely sampling adjacent frames to pay attention to the inter-frame image inconsistency.
	Yang *et al.* [304]	TIFS'23	①②③	①②③	Treating detection as a graph classification problem and focusing on the relationship between the local image features across different frames.
	AVoiD-DF [301]	TIFS'23	②⑤ [130]	②⑤ [130]	Multimodal forgery detection using audiovisual inconsistency.
	Choi *et al.* [38]	CVPR'24	①③④	①③	Focus on the inconsistency of the style latent vectors between frames.
	Xu *et al.* [290]	IJCV'24	①②③④	①②③④	Forgery detection is conducted by converting video clips into thumbnails containing both spatial and temporal information.
	Peng *et al.* [212]	TIFS'24	①③⑧	①③⑧	Focuse on inter-frame gaze angles, extracting gaze informations and employing spatio-temporal feature aggregation to combine temporal, spatial, and texture features for detection and classification.
Frequency	F^3-Net [219]	ECCV'20	①	①	A two-branch frequency perception framework with a cross-attention module is proposed.
	FDFL [150]	CVPR'21	①	①	Propose an adaptive frequency feature generation module to extract differential features from different frequency bands in a learnable manner.
	HFI-Net [185]	TIFS'22	①	②③④⑥ [138]	Notice that the forgery flaws used to distinguish between real and fake faces are concentrated in the mid- and high-frequency spectrum.
	Guo *et al.* [80]	TIFS'23	①②	①②③	Designing a backbone network for Deepfake detection with space-frequency interaction convolution.
	Tan *et al.* [248]	AAAI'24	⑤	①⑤	A lightweight frequency-domain learning network is proposed to constrain classifier operation within the frequency domain.
Data Driven	Dang *et al.* [45]	CVPR'20	⑤	③⑥	Utilizing attention mechanisms to handle the feature maps of the detection model.
	Zhao *et al.* [344]	ICCV'21	①	①②③④⑧⑩	Proposes pairwise self-consistent learning for training CNN to extract these source features and detect deep vacation images.
	Finfer [99]	AAAI'22	①	①③⑧ [363]	Based on an autoregressive model, using the facial representation of the current frame to predict the facial representation of future frames.
	Huang *et al.* [102]	CVPR'23	①	①②③⑩ [152]	A new implicit identity-driven face exchange detection framework is proposed.
	HiFi-Net [75]	CVPR'23	⑤	⑤	Converting forgery detection and localization into a hierarchical fine-grained classification problem.
	Zhai *et al.* [323]	ICCV'23	[52]	[98] [276]	Weakly supervised image processing detection is proposed such that only binary image level labels (real or tampered) are required for training.

https://www.geeksforgeeks.org/stylegan-style-generative-adversarial-networks/

Přidat k progressive GAN





The latent space of the StyleGANs is known to be well disentangled, which means that one can manipulate semantic features without changing other characteristics so much. However, to be able to edit a specific image, it is first necessary to know the corresponding latent code that will recreate this image in a trained GAN. The pro cess to obtain this latent vector is called GAN inversion. Specifically, in the StyleGAN case, the survey by Bermano et al. [44] has a comprehensive study of different techniques used to invert and manipulate StyleGANs.