

Computer Vision 3

Ş.I. dr. ing. Mihai DOGARIU

www.mdogariu.aimultimedialab.ro

Structura cursului



- M1. Introducere
- M2. Fundamentele Învățării Adânci (Deep Learning Fundamentals)
- M3. Învățare Adâncă Supervizată (Supervised Deep Learning)
- M4. Învățare Adâncă Nesupervizată (Unsupervised Deep Learning)
- M5. Învățare Consolidată (Reinforcement Learning)

M4. Învățare Adâncă Nesupervizată (Unsupervised Deep Learning)

- 4.1. Concept Unsupervised Deep Learning
- 4.2. Autoencoders (AE)
- 4.3. Variational Autoencoders (VAE)
- 4.4. Generative Adversarial Networks (GAN)

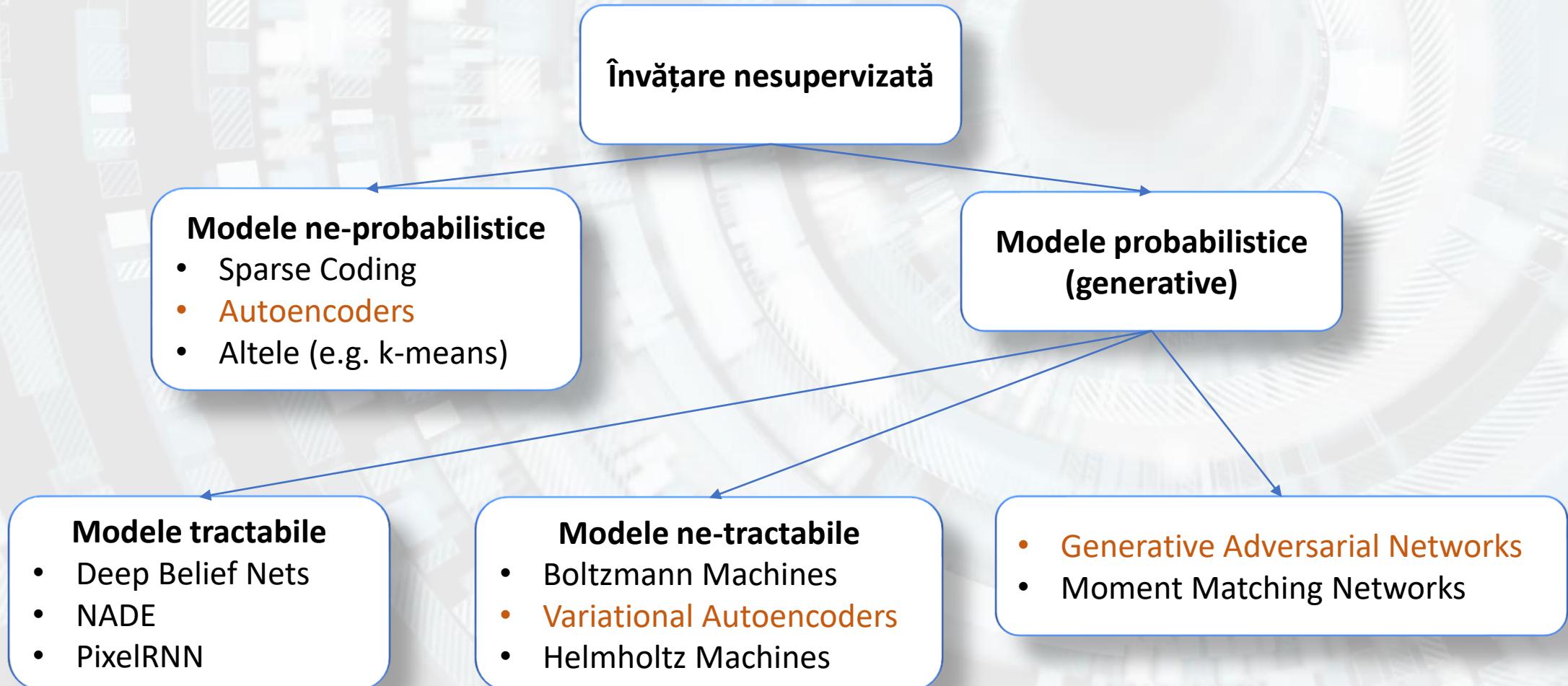
M4.1. Concept Unsupervised Deep Learning

Unsupervised Deep Learning

Învățarea supervizată = paradigmă de învățare a mașinilor în care datele de antrenare sunt etichetate. Fiecare exemplu de antrenare este format dintr-un descriptor de trăsături și o etichetă. Scopul învățării supervizate este de a învăța funcția de asociere dintre trăsăturile de intrare și eticheta corespondentă.

Învățarea nesupervizată = paradigmă de învățare a mașinilor în care datele de antrenare sunt neetichetate. Scopul învățării nesupervizate este de a învăța modelul inherent al exemplelor de antrenare prin imitare. Aceasta se face prin auto-organizare ce captează informații privitoare la comportamentul/distribuția de probabilitate a datelor de antrenare.

Unsupervised Deep Learning



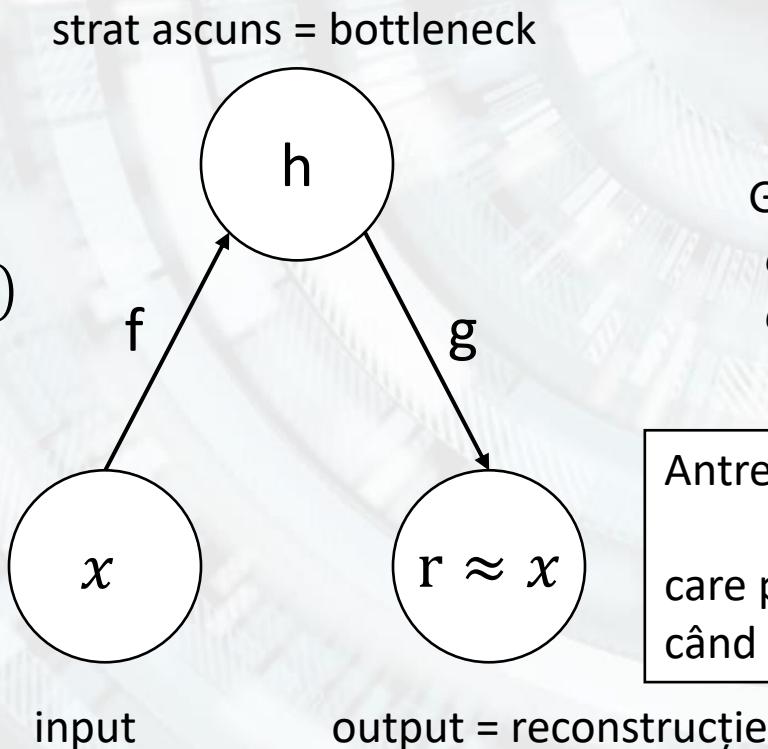
M4.2. Autoencoders

Autoencoders

Autoencoder = rețea neuronală antrenată pentru a încerca să reproducă intrarea la ieșire.

Funcții deterministe
encoder: $h = f(x)$
decoder: $r = g(h) = g(f(x))$

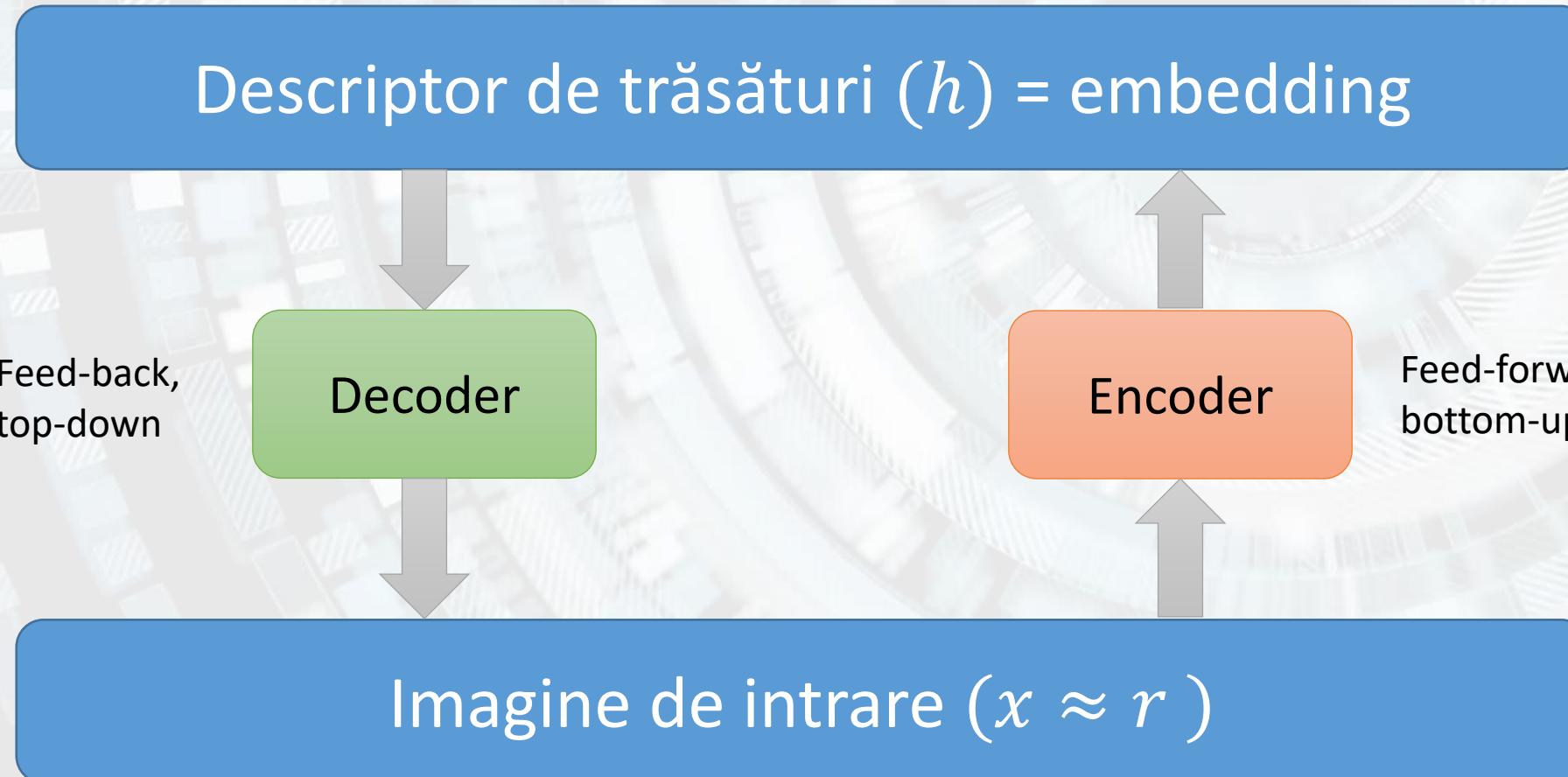
- f și g pot fi obținute cu ajutorul unor rețele neuronale;
- x și r reprezintă structuri informaționale (vectori/tensori);
- h reprezintă un descriptor de trăsături.



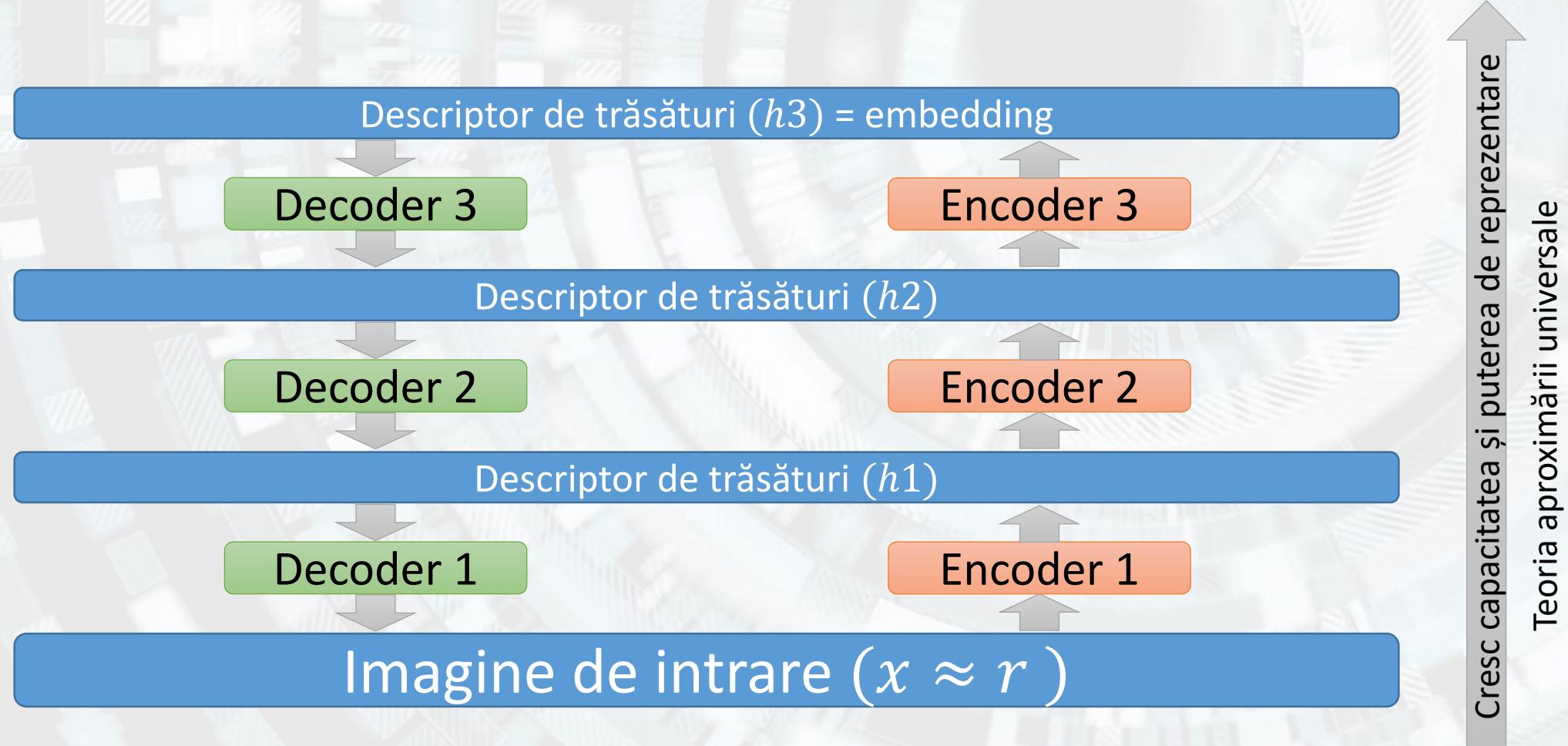
Generalizare stocastică
encoder: $p_{encoder}(h|x)$
decoder: $p_{decoder}(x|h)$

Antrenarea se face cu o pierdere $L(x, g(f(x)))$ care penalizează ieșirea $g(f(x))$ atunci când este diferită de x ; e.g. MSE.

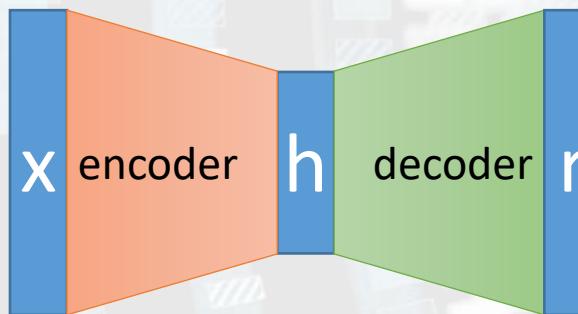
Autoencoders



Autoencoders – stacked



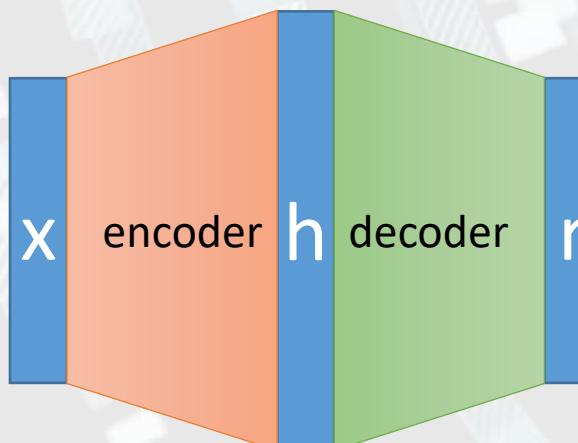
Autoencoders



Undercomplete autoencoder

$$\dim(h) < \dim(x)$$

Encoder-ul și decoder-ul învață o reprezentare contractată a datelor.

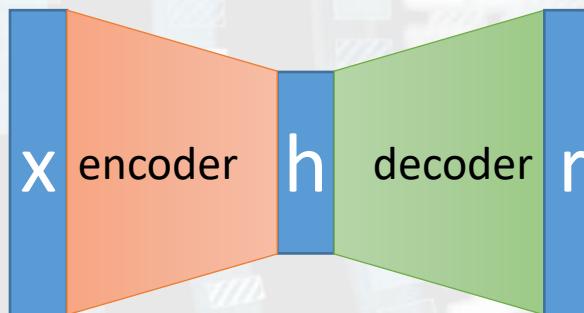


Overcomplete autoencoder

$$\dim(h) \geq \dim(x)$$

Encoder-ul și decoder-ul învață o reprezentare dilată a datelor.

Autoencoders



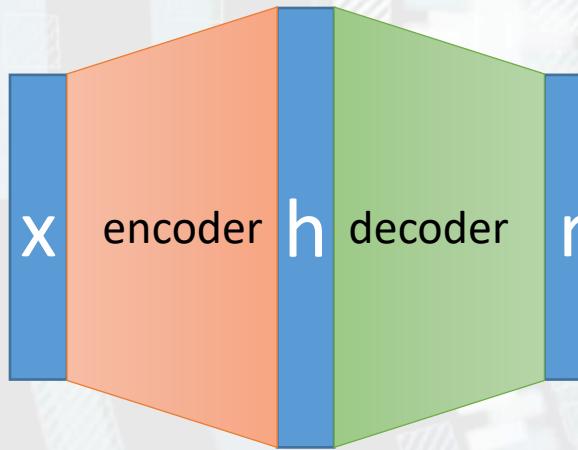
Undercomplete autoencoder

$$\dim(h) < \dim(x)$$

Encoder-ul și decoder-ul învață o reprezentare contractată a datelor.

- Descriptorul de trăsături este restricționat la o dimensiune mai mică decât intrarea => autoencoder-ul învață cele mai reprezentative trăsături;
- Decodor liniar + MSE => autoencoderul învață același subspațiu ca un PCA;
- Codor neliniar + decodor neliniar => reprezentare mai puternică decât PCA;
- Dacă oferim prea multă capacitate (numărul de parametri antrenabili) codorului și decodorului, acestea pot ajunge să învețe funcția identitate => autoencoder-ul devine inutil.

Autoencoders



Overcomplete autoencoder

$$\dim(h) \geq \dim(x)$$

Encoder-ul și decoder-ul învață o reprezentare dilată a datelor.

- Descriptorul de trăsături are dimensiunea cel puțin egală cu dimensiunea datelor;
- Autoencoder-ul poate reproduce intrarea la ieșire foarte simplu, prin copierea datelor de la un capăt la altul (encoder liniar + decoder liniar) – acest caz nu este dorit, deoarece autoencoder-ul nu învață nimic;
- Funcția identitate este evitată prin regularizare – o serie de constrângeri suplimentare ce împiedică copierea intrării la ieșire. Regularizarea se poate face prin:
 - Reprezentări rare (sparse autoencoder);
 - Introducerea robusteții la zgomot (denoising autoencoder);
 - Păstrarea derivatelor la niște valori mici.

Autoencoders – sparse autoencoder

Se introduce un termen de penalizare a rarității (sparsity penalty) în funcția de cost:

$$\mathcal{L} = L(x, g(f(x))) + \Omega(\mathbf{h})$$

Parametrul de penalizare a rarității este, de obicei, definit ca:

1. Regularizare L1:

$$\Omega(\mathbf{h}) = \lambda \sum_i |a_i^{(h)}|$$

Unde $a_i^{(h)}$ este vectorul activărilor din stratul h pentru eșantionul i.

2. Divergență Kullback-Leibler:

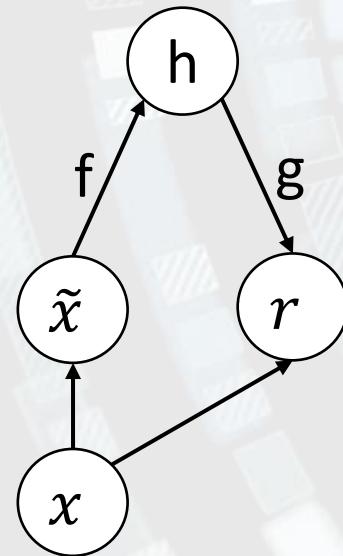
$$\Omega(\mathbf{h}) = \sum_j KL(\rho || \hat{\rho}_j)$$

Unde:

- $\hat{\rho}_j = \frac{1}{m} \sum_i |a_i^{(h)}(x)|$ = raritatea neuronului j din stratul h, calculată ca media activărilor a m eșantioane notate cu x;
- ρ este raritatea dorită și reprezintă activarea medie a unui neuron pentru un subset de date.

Autoencoders – denoising autoencoder

Dacă un autoencoder poate reconstrui intrarea, de ce nu ar putea să reconstruască și o variantă ușor modificată a acesteia?



Intrare: $\tilde{x} = x + n$

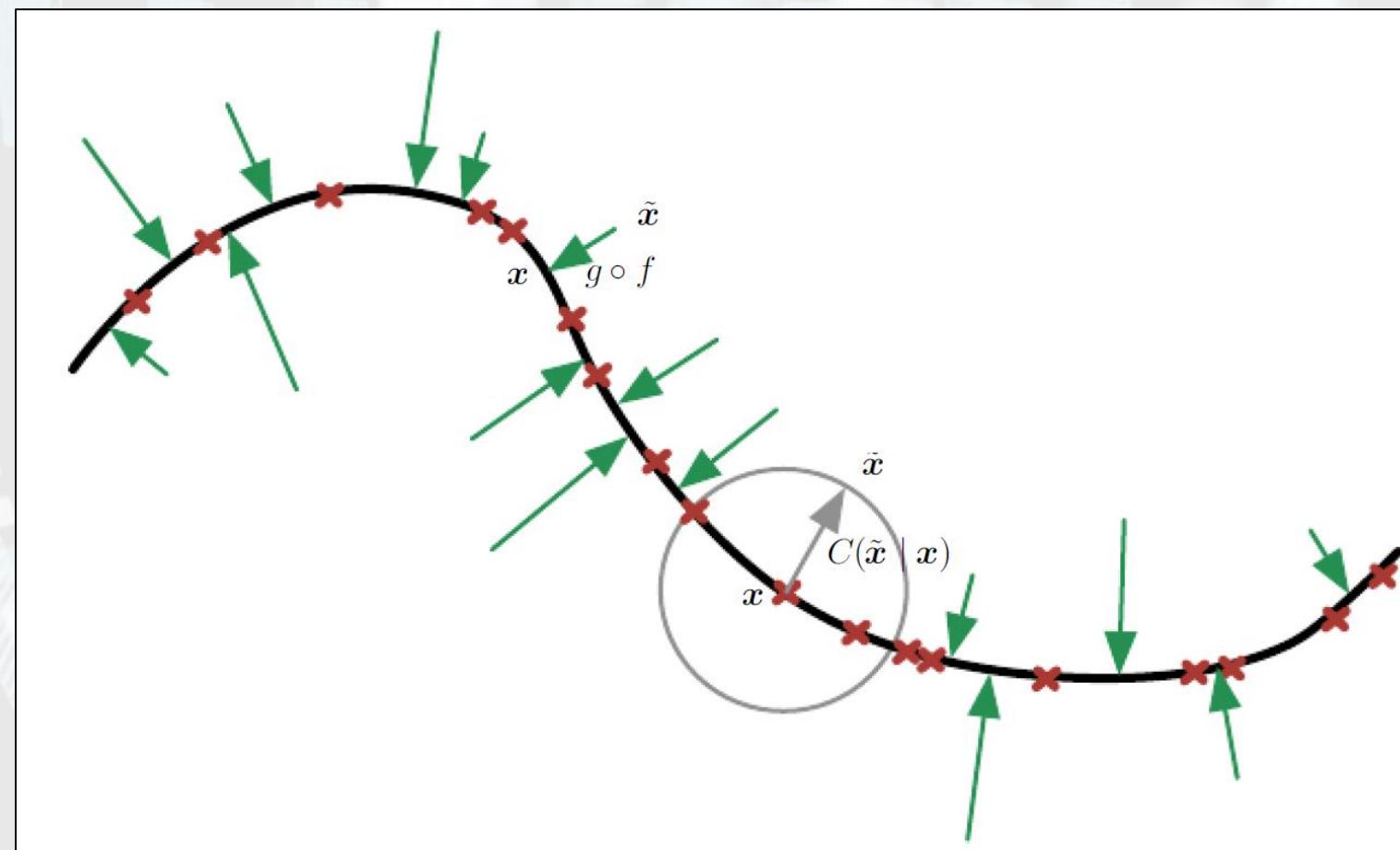
Ieșire (reconstrucție): x

Obiectiv: învățarea modului de eliminare a zgomotului n din intrare

Optimizare: minimizarea funcției de cost

$$\mathcal{L} = L(x, g(f(\tilde{x}))) = L(x, g(f(x + n)))$$

Autoencoders – denoising autoencoder



- Spațiul datelor de antrenare
- ✗ Eșantioane de antrenare
- Coruperea datelor
- Recuperarea datelor

Autoencoders – penalizarea derivatelor

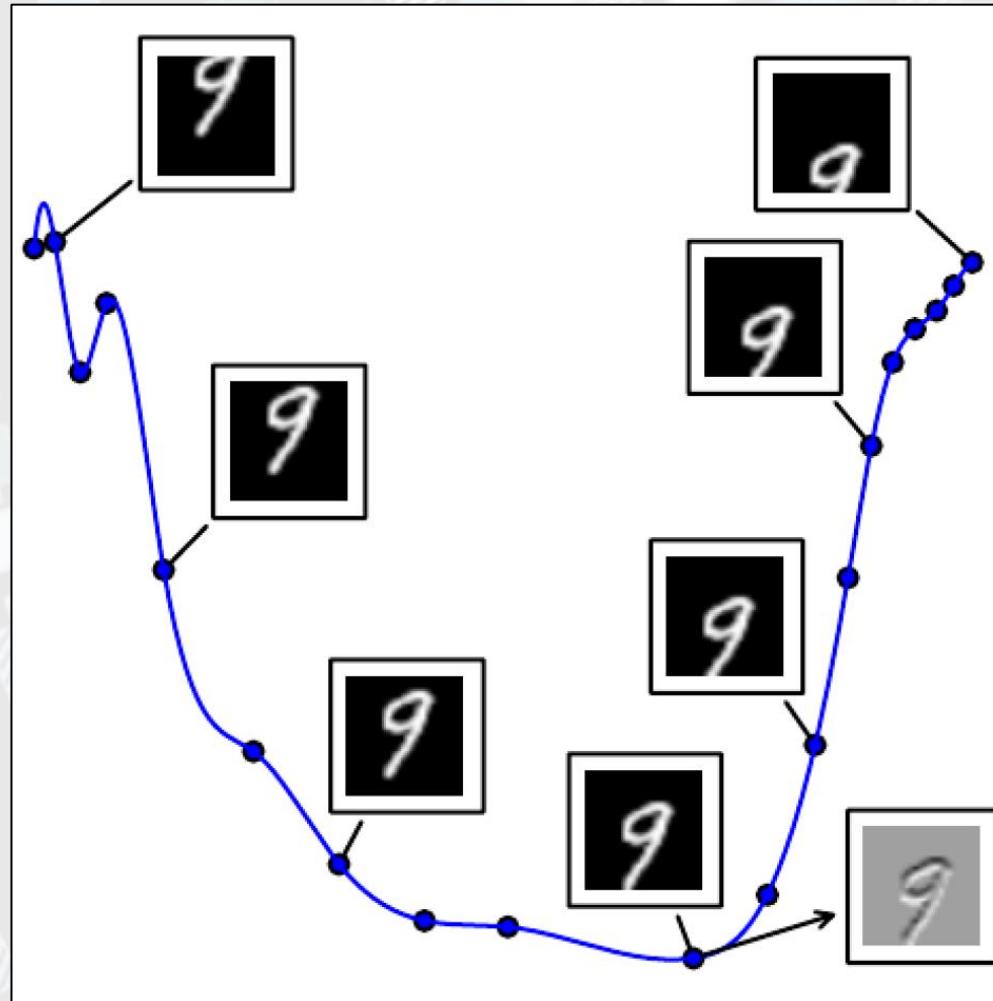
Se introduce un termen de penalizare a rarității (sparsity penalty) în funcția de cost:

$$\mathcal{L} = L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}, \mathbf{x})$$

$$\Omega(\mathbf{h}, \mathbf{x}) = \lambda \sum_i \|\nabla_{\mathbf{x}} h_i\|^2$$

- Forțează modelul să învețe funcții care nu suferă schimbări majore la mici modificări ale lui \mathbf{x}
- Se aplică doar în momentul antrenării, deci forțează modelul să învețe descriptori care captează informații legate de distribuția de probabilitate a datelor de antrenare.
- Se mai numește și autoencoder contractiv, deoarece asociază unei vecinătăți de intrări o vecinătate mai mică de ieșiri.

Autoencoders – învățarea spațiului de date



Asociere între planul imaginii (784-dimensional, imagini din baza MNIST) și curba spațiului de date (manifold) undimensională. În acest caz, s-a folosit reprezentare bidimensională cu ajutorul PCA.

Translația imaginii generează o curbă complexă a spațiului de date.

Hiperplan tangent = plan ce definește direcțiile în care ne putem deplasa cu un pas infinitezimal și să rămânem pe curba spațiului de date. În acest caz, există o singură linie tangentă.

Un spațiu n-dimensional are n linii tangente.

Imaginea gri reprezintă schimbarea suferită în urma avansării în direcția dată de tangentă:

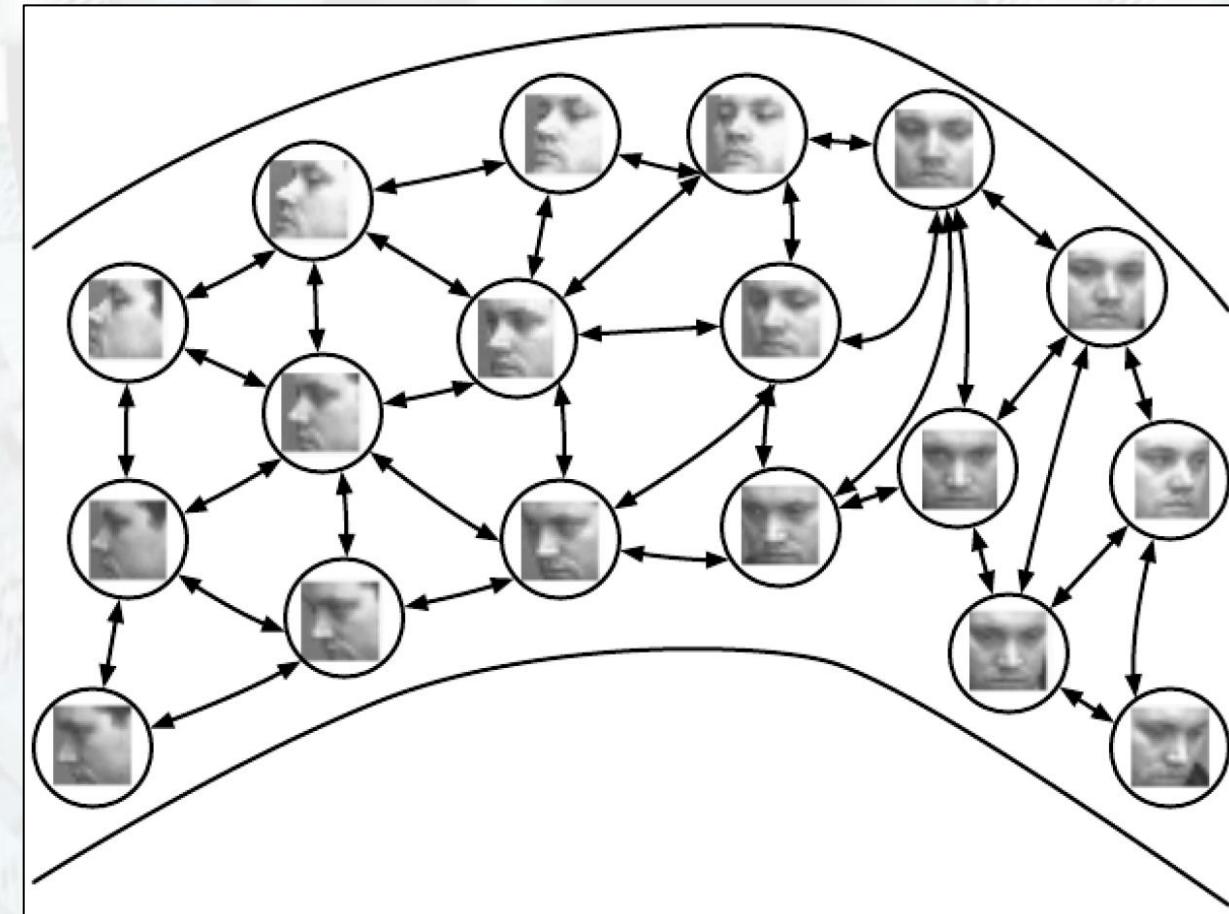
- pixeli gri => pixelii rămân neschimbați;
- pixeli albi => pixelii se deschid la culoare;
- pixeli negri => pixelii se închid la culoare.

Autoencoders – Învățarea spațiului de date

Învățarea spațiului de date duce la crearea unui graf cu cei mai apropiati vecini (nearest neighbor):

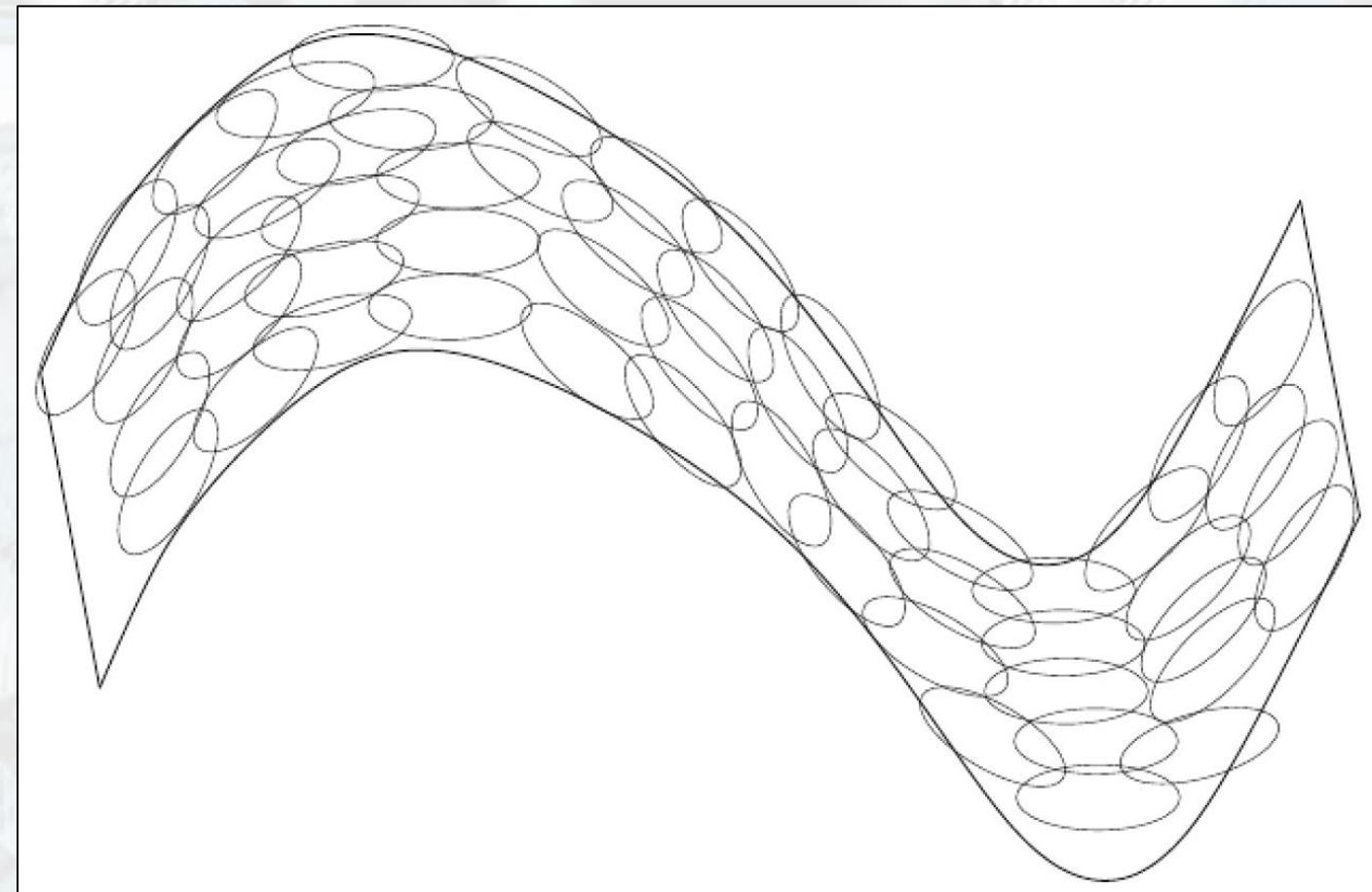
- nodurile = eșantioane de antrenare;
- muchiile = relațiile de tipul cel mai apropiat vecin.

Dacă există suficiente date de antrenare, se poate învăța întregul spațiu de date și, deci, se pot genera eșantioane noi prin interpolare.



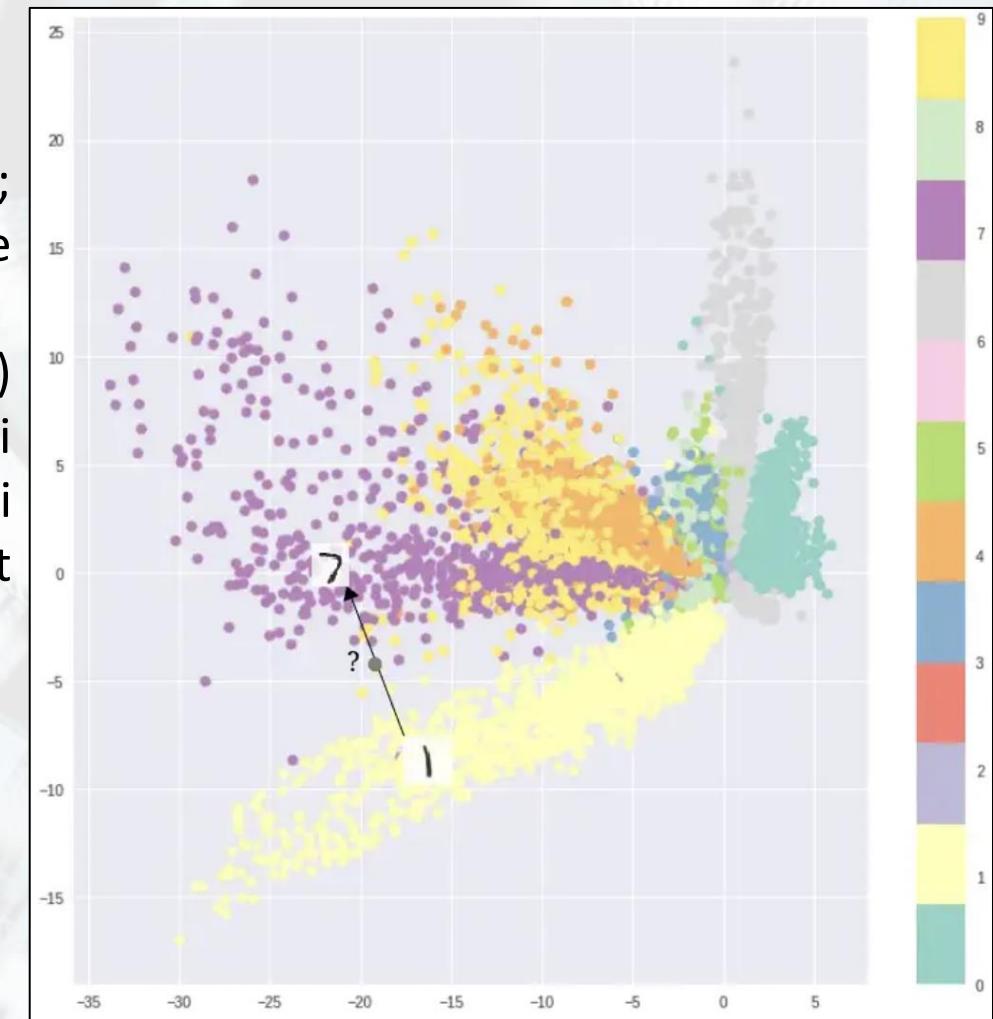
Autoencoders – Învățarea spațiului de date

- Cunoașterea planurilor tangente din fiecare locație conduce la combinarea lor și estimarea funcției de densitate a probabilității datelor, care modelează complet setul de date de antrenare.
- Fiecare plan tangent este aproximat cu o curbă Gaussiană.
- În practică, spațiul datelor este foarte complex și poate fi doar aproximat (imposibil de determinat precis).



Autoencoders – Învățarea spațiului de date

- Proiectarea descriptorilor într-un spațiu 2D;
- Antrenarea autoencoder-ului pe MNIST Digits;
- Fiecare cifră aparține unui cluster relativ bine definit;
- Alegem un punct ce nu se află pe grafic – ce imagine este decodată?
- **Limitare:** decodorul poate procesa (replica input) doar descriptori pe care i-a văzut în timpul antrenării – spațiu discret de valori. Decodarea altor valori duce la imagini fără sens => nu este adecvat generării de exemple noi.



Autoencoders – concluzii

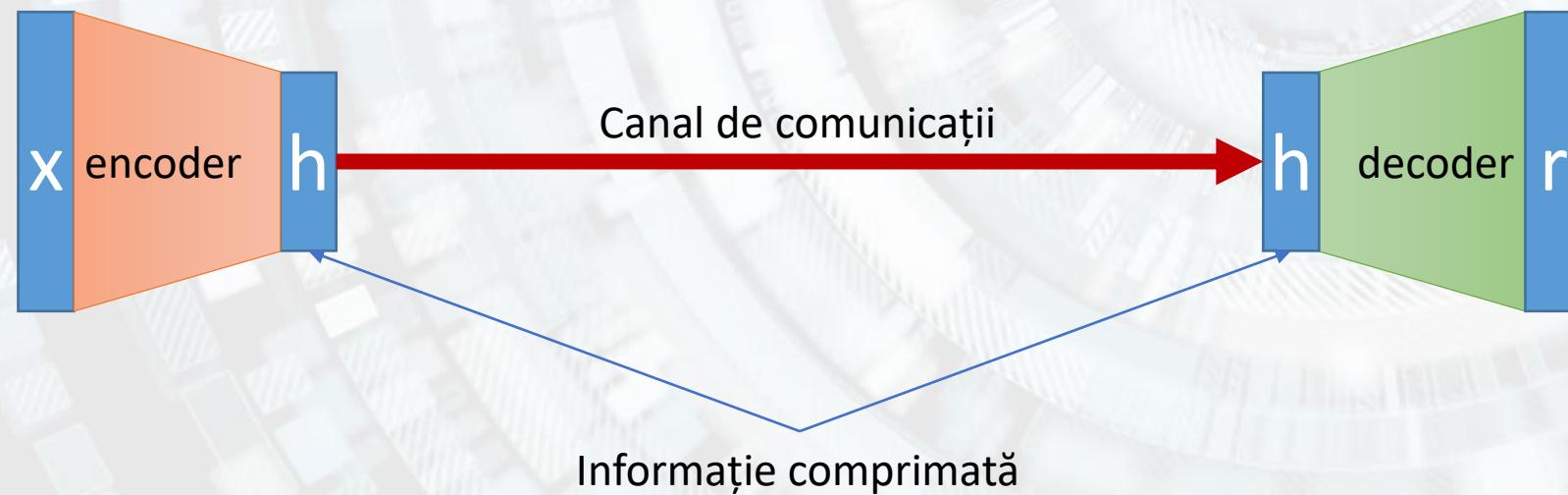
- Autoencoders sunt foarte bune pentru a recupera dintr-un spațiu de date informația folosită la antrenare;
- Implică antrenarea cu un număr mare de exemple pentru a acoperi întregul spațiu de date (manifold);
- Sunt robuste la o perturbație relativ mică a descriptorului de trăsături (a embedding-ului);
- Nu sunt adecvate generării de exemple noi.

Autoencoders – aplicații

1. **Reducerea dimensionalității datelor** – undercomplete autoencoders pot reduce dimensionalitatea datelor mai mult decât tehniciile clasice, e.g. PCA. O aplicație imediată este compresia datelor în comunicații.
2. **PCA** – în cazul folosirii activărilor liniare, autoencoderele obțin un spațiu latent similar cu PCA.
3. **Information retrieval** – descriptorii din stratul bottleneck pot fi utilizați pentru recuperarea informației (vezi M2, slide 63).
4. **Detectia anomalilor** – outliers vor fi evidențiați cu ușurință, deoarece vor avea o reconstrucție foarte slabă.
5. **Clustering** – spațiul latent va fi organizat sub forma unor clustere compuse din exemplele cel mai asemănătoare.
6. **Denoising** – autoencoder care primește la intrare imagini cu zgomot și este antrenat să reproducă imagini fără zgomot.
7. **Inpainting** – autoencoder care primește la intrare imagini cu zone ascunse și este antrenat să reproducă imagini complete.
8. **Colorizare** – autoencoder care primește la intrare imagini cu nuanțe de gri și este antrenat să reproducă imagini color.

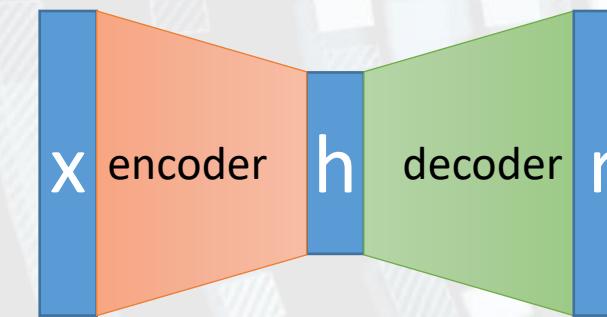
Autoencoders – aplicații

1. Reducerea dimensionalității datelor



Autoencoders – aplicații

2. PCA



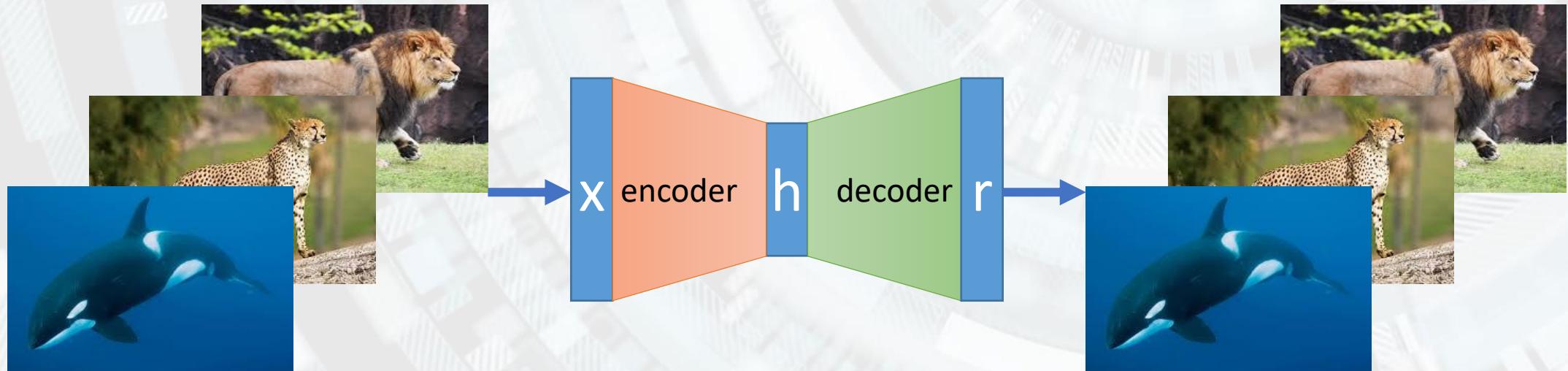
- un singur strat ascuns;
- activări liniare;
- straturi complet conectate;
- funcție de cost pătratică.

PCA (ponderi care acoperă același spațiu cu componentele principale)

Autoencoders – aplicații

3. Information retrieval

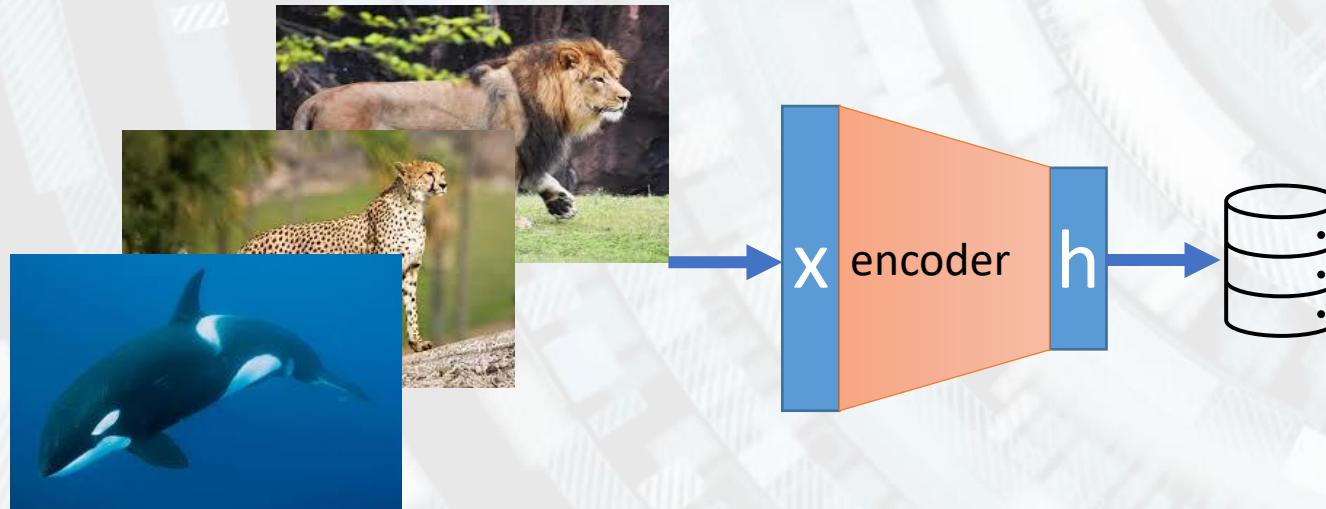
- Antrenarea autoencoderului cu baza de train



Autoencoders – aplicații

3. Information retrieval

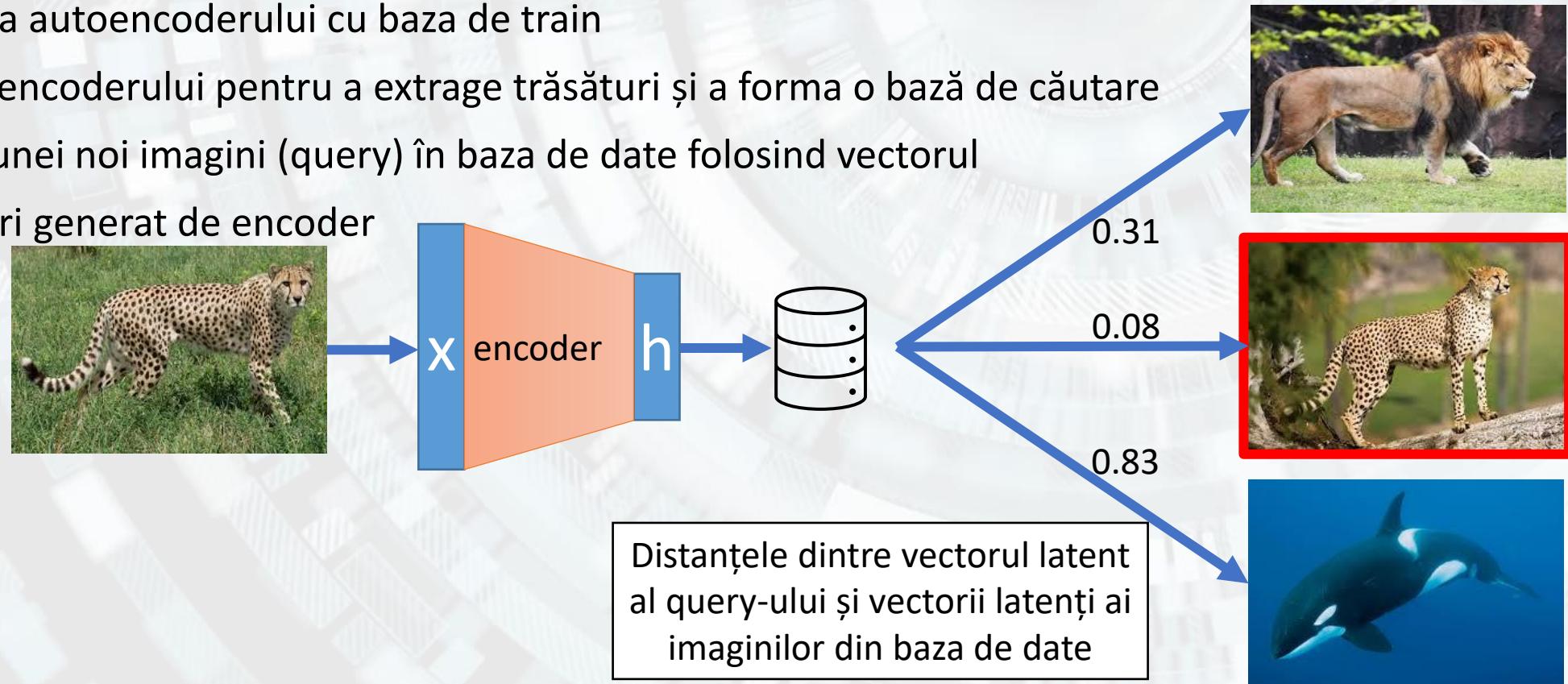
- Antrenarea autoencoderului cu baza de train
- Utilizarea encoderului pentru a extrage trăsături și a forma o bază de căutare



Autoencoders – aplicații

3. Information retrieval

- Antrenarea autoencoderului cu baza de train
- Utilizarea encoderului pentru a extrage trăsături și a forma o bază de căutare
- Căutarea unei noi imagini (query) în baza de date folosind vectorul de trăsături generat de encoder

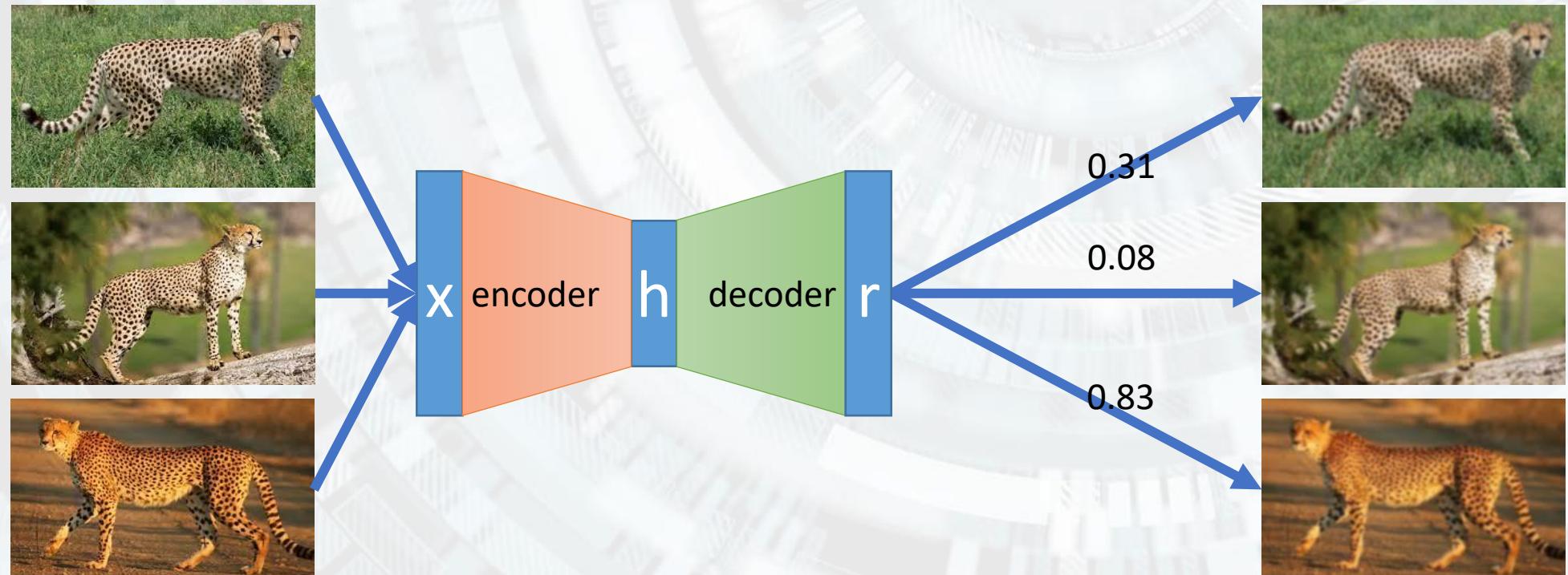


Autoencoders – aplicații

4. Detectia anomalilor

- Antrenarea autoencoderului cu baza de train fără anomalii

Ieșiri cu o eroare de
reconstrucție redusă
(posibile artefacte:blur)

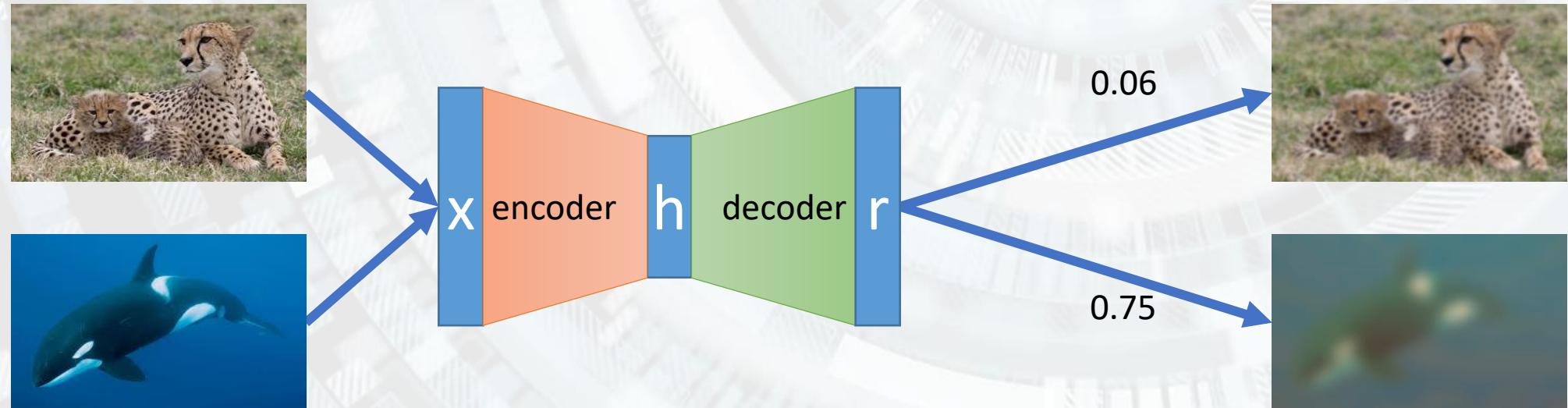


Autoencoders – aplicații

4. Detectia anomalilor

- Antrenarea autoencoderului cu baza de train fără anomalii
- Testarea autoencoderului cu imagini diverse (cu/fără anomalii)

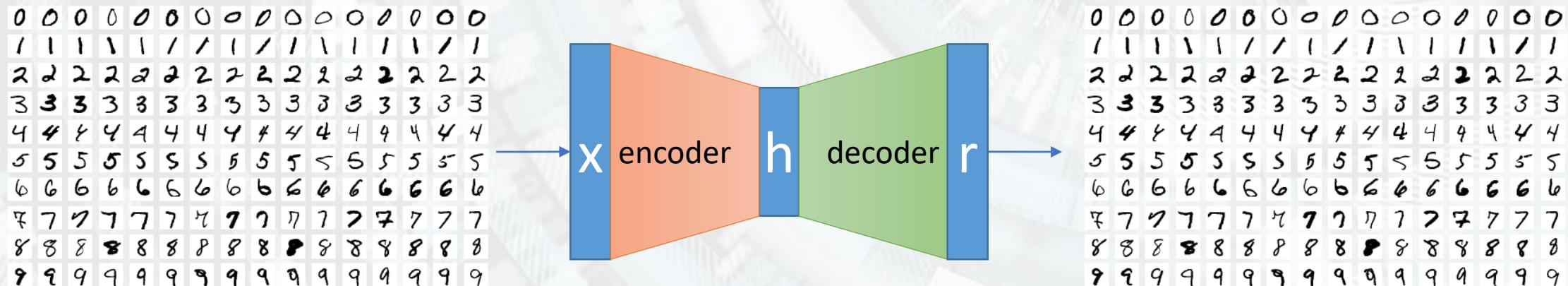
Eroarea mare de
reconstrucție
indică o anomalie



Autoencoders – aplicații

5. Clustering

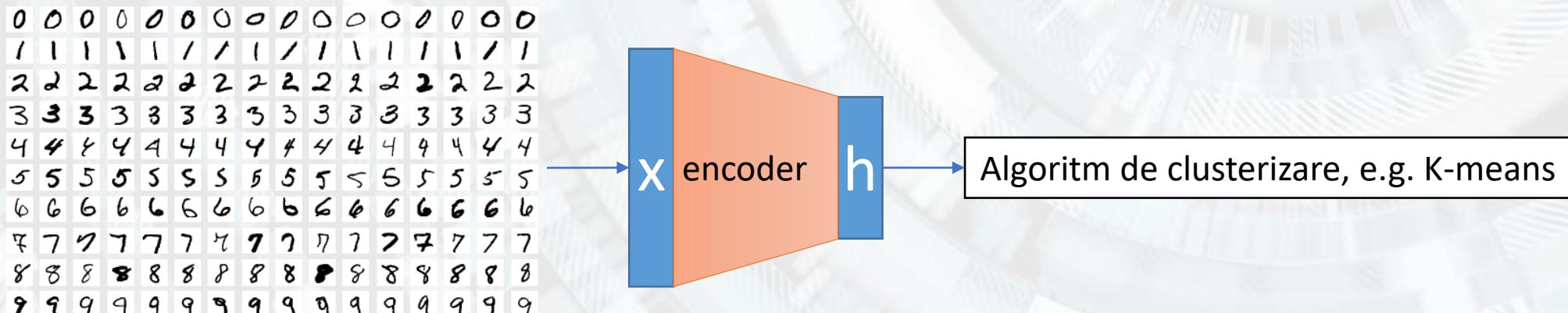
- Antrenarea autoencoderului cu baza de train



Autoencoders – aplicații

5. Clustering

- Antrenarea autoencoderului cu baza de train
 - Folosirea encoderului pentru a genera descriptori de trăsături utili pentru rularea unui algoritm de clusterizare, e.g. k-means.

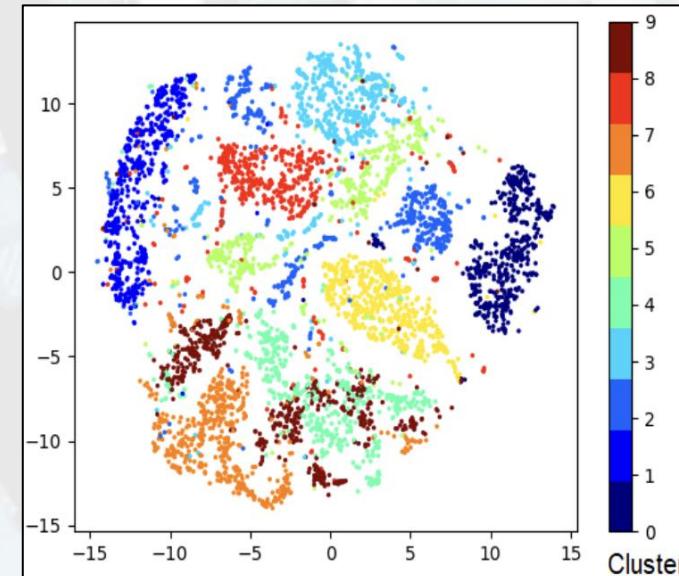


Autoencoders – aplicații

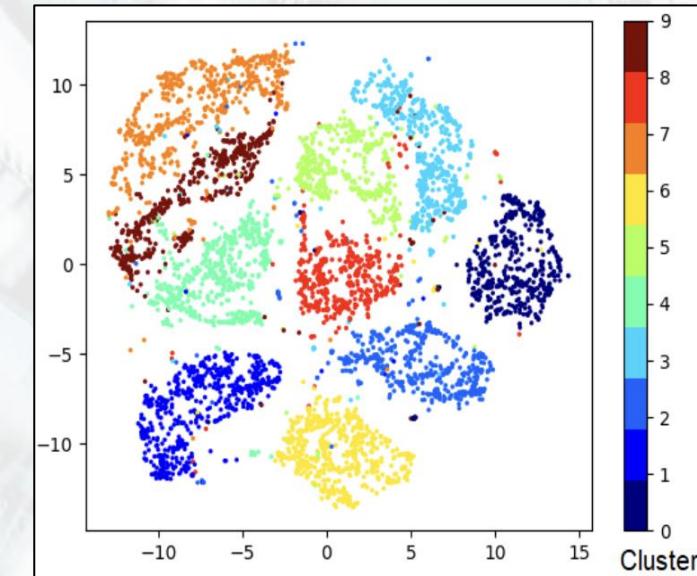
5. Clustering

- Antrenarea autoencoderului cu baza de train
- Folosirea encoderului pentru a genera descriptori de trăsături utili pentru rularea unui algoritm de clusterizare, e.g. k-means.

K-means rulat în spațiul pixelilor



vs

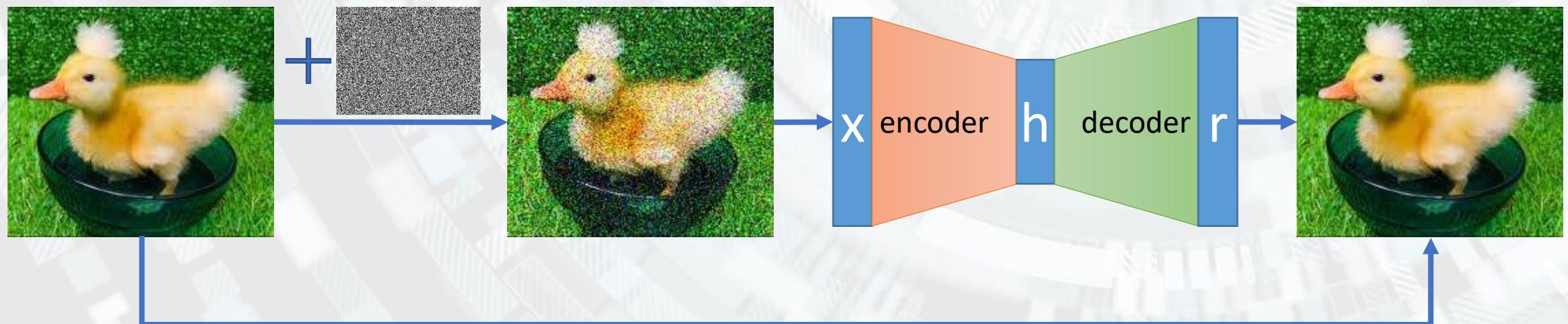


K-means rulat în spațiul latent

Autoencoders – aplicații

6. Denoising

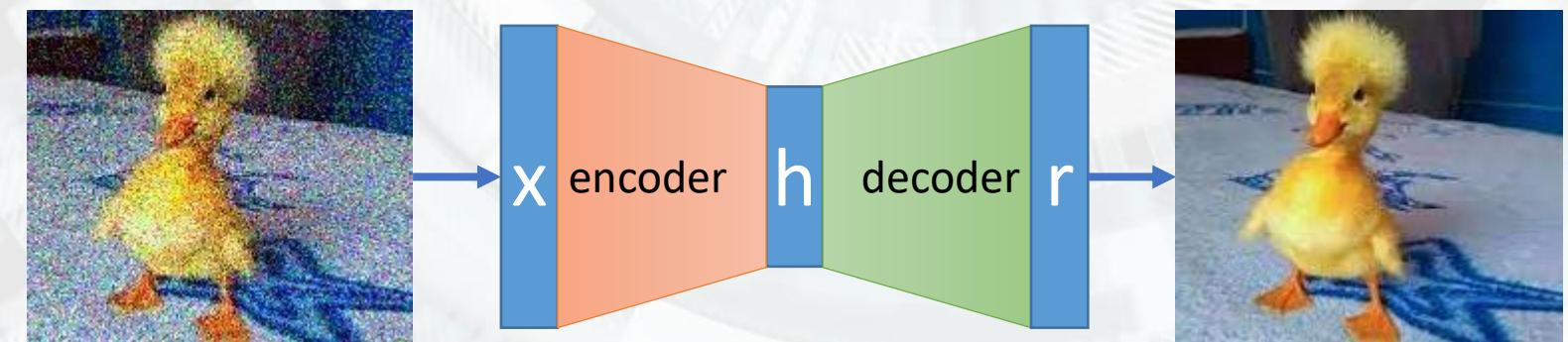
- Antrenarea autoencoderului cu baza de train în formatul input = original + zgomot, output = original.



Autoencoders – aplicații

6. Denoising

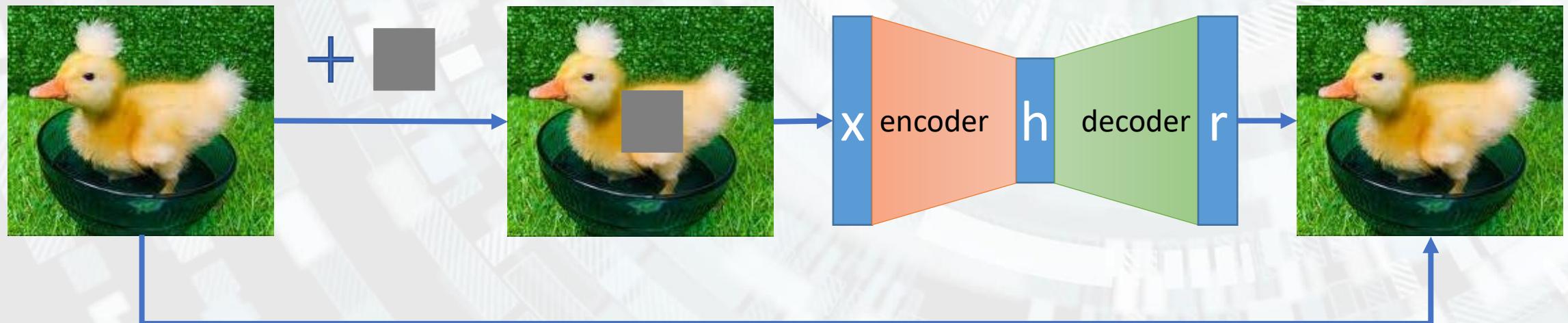
- Antrenarea autoencoderului cu baza de train în formatul input = original + zgomot, output = original.
- Testarea autoencoderului cu imagini cu zgomot. Autoencoderul învață să eliminate tipul de zgomot cu care a fost antrenat.



Autoencoders – aplicații

7. Inpainting

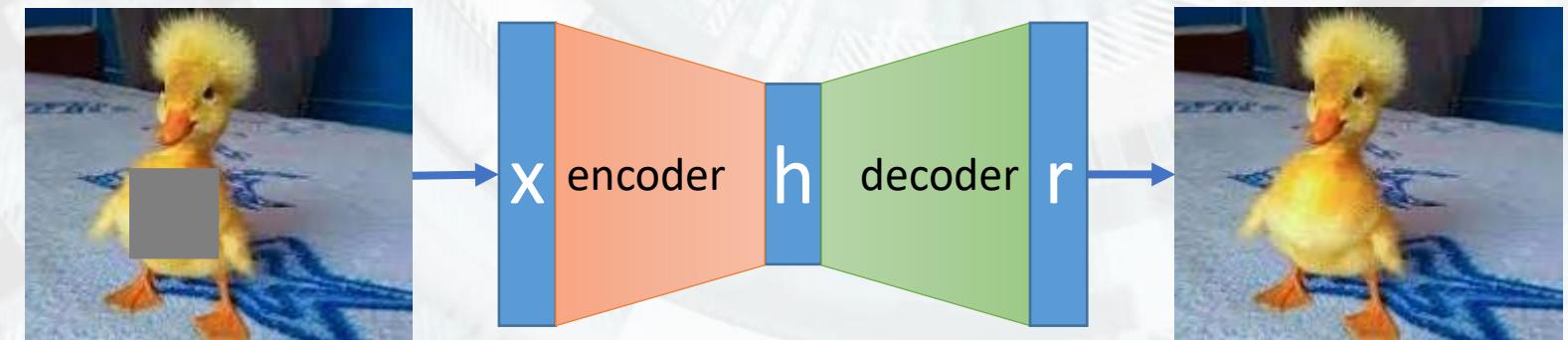
- Antrenarea autoencoderului cu baza de train în formatul input = original + ocluziune, output = original. Este o formă particulară de denoising.



Autoencoders – aplicații

7. Inpainting

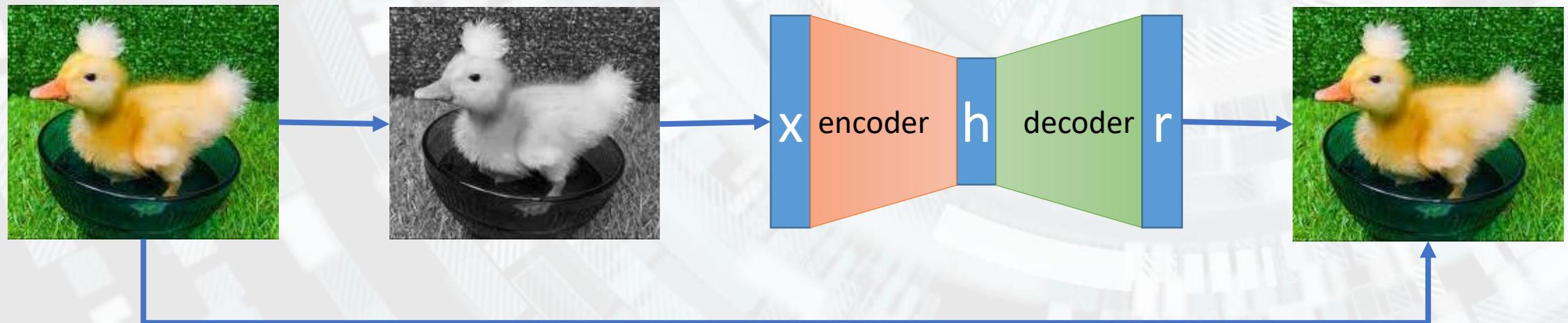
- Antrenarea autoencoderului cu baza de train în formatul input = original + ocluziune, output = original. Este o formă particulară de denoising.
- Testarea autoencoderului cu imagini cu ocluziuni. Autoencoderul învață să recupereze informația din spatele ocluziunii.



Autoencoders – aplicații

8. Colorizare

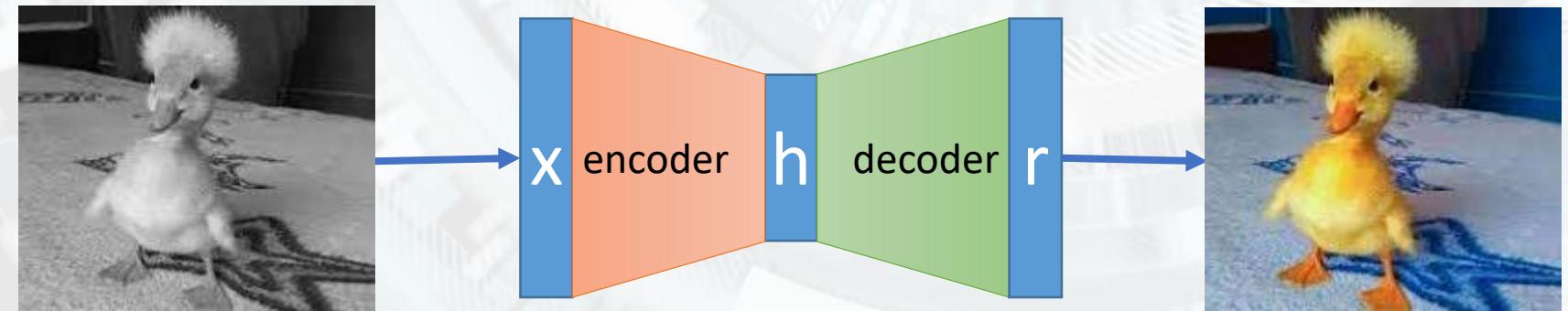
- Antrenarea autoencoderului cu baza de train în formatul input = original transformat în grayscale, output = original.



Autoencoders – aplicații

8. Colorizare

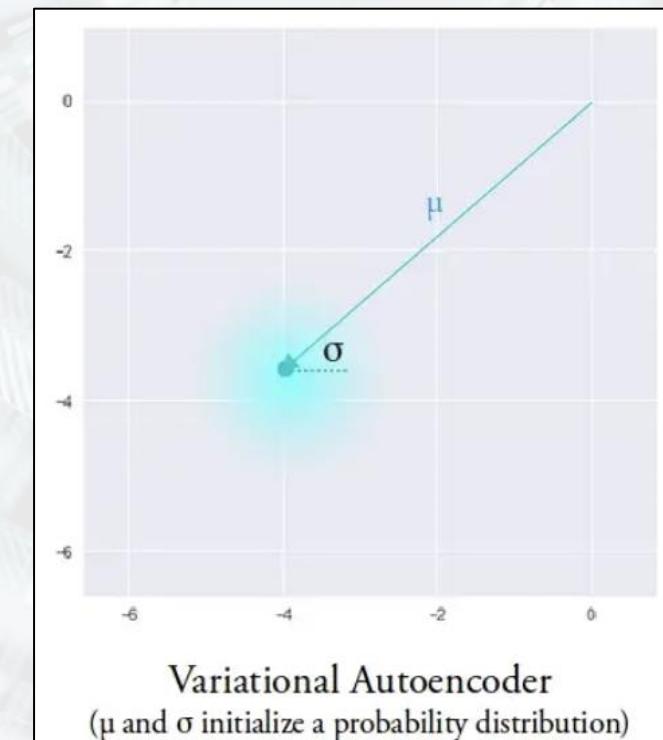
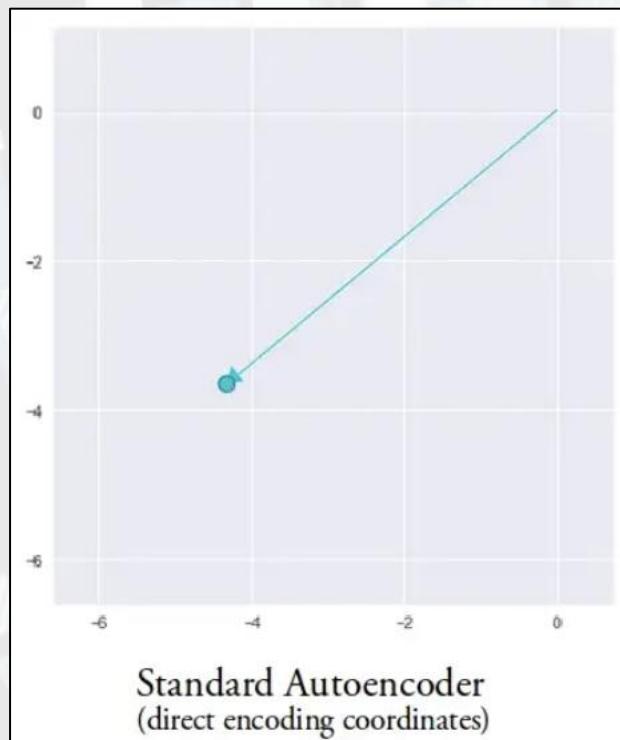
- Antrenarea autoencoderului cu baza de train în formatul input = original transformat în grayscale, output = original.
- Testarea autoencoderului cu imagini grayscale. Autoencoderul învață să coloreze imagini grayscale.



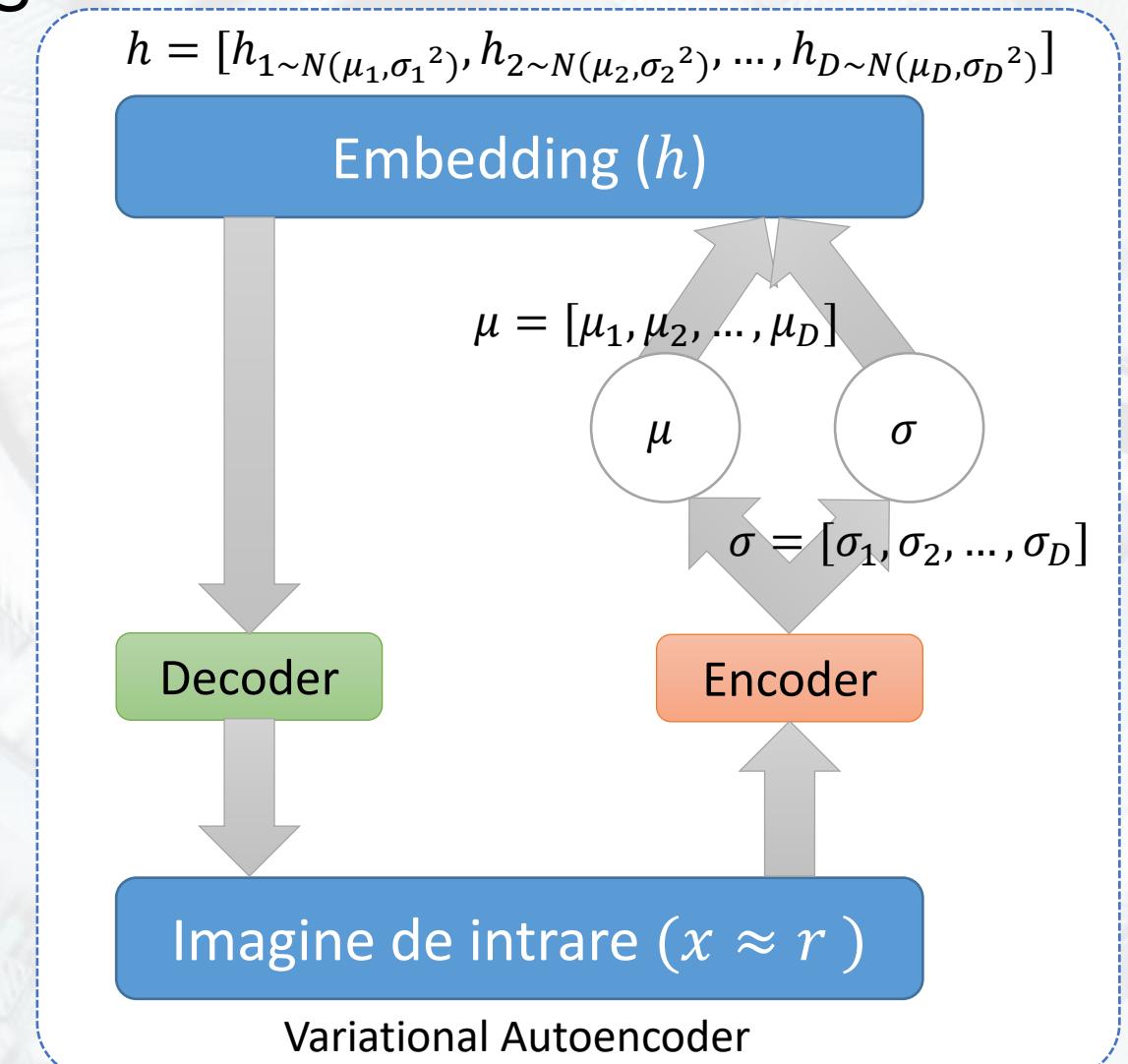
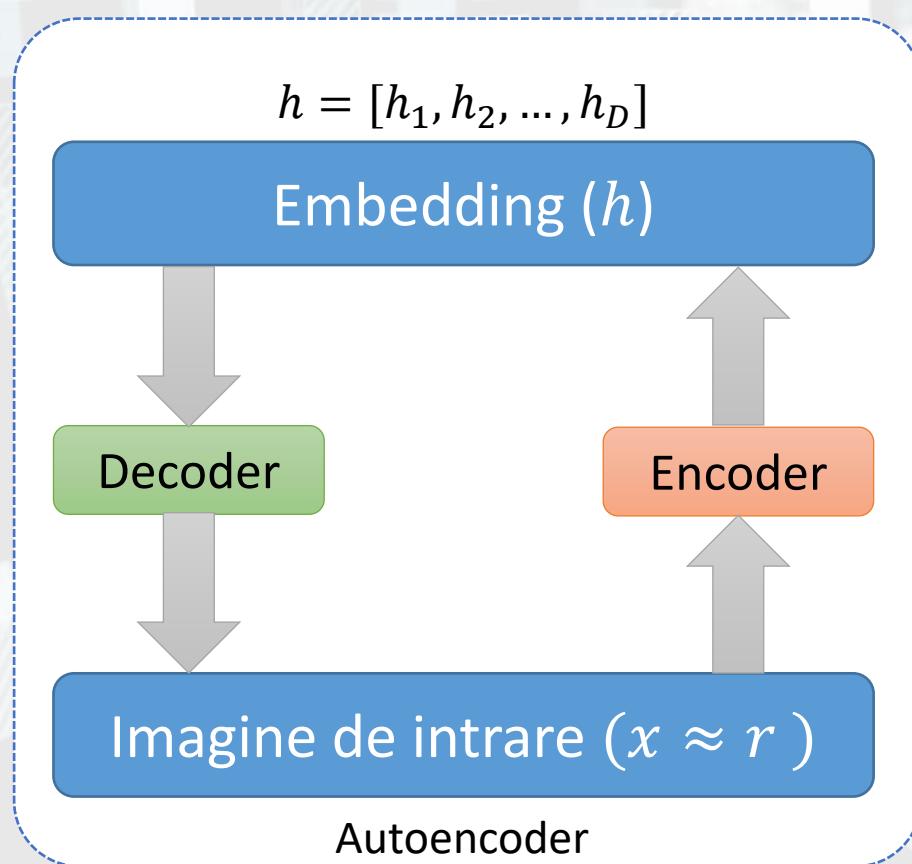
M4.3. Variational Autoencoders

Variational Autoencoders

VAE [2] preia ideea autoencoder-ului și o adaptează pentru **generare**. Cum face asta?



Variational Autoencoders



Variational Autoencoders

$$h = [h_{1 \sim N(\mu_1, \sigma_1^2)}, h_{2 \sim N(\mu_2, \sigma_2^2)}, \dots, h_{D \sim N(\mu_D, \sigma_D^2)}]$$

- Descriptorul este generat sub forma unui eşantion dintr-o distribuţie normală cu media μ şi deviaţia standard σ => acelaşi exemplu de antrenare va genera descriptori diferiţi, însă din aceeaşi distribuţie de probabilitate;
- μ controlează zona unde se va centra distribuţia;
- σ controlează aria pe care se va desfăşura distribuţia;
- Decodorul va asocia unei vecinătăţi (atribuite distribuţiei normale) o singură imagine de ieşire.
- Asociere continuă, nu discretă (cum este în cazul autoencoderului).

Variational Autoencoders

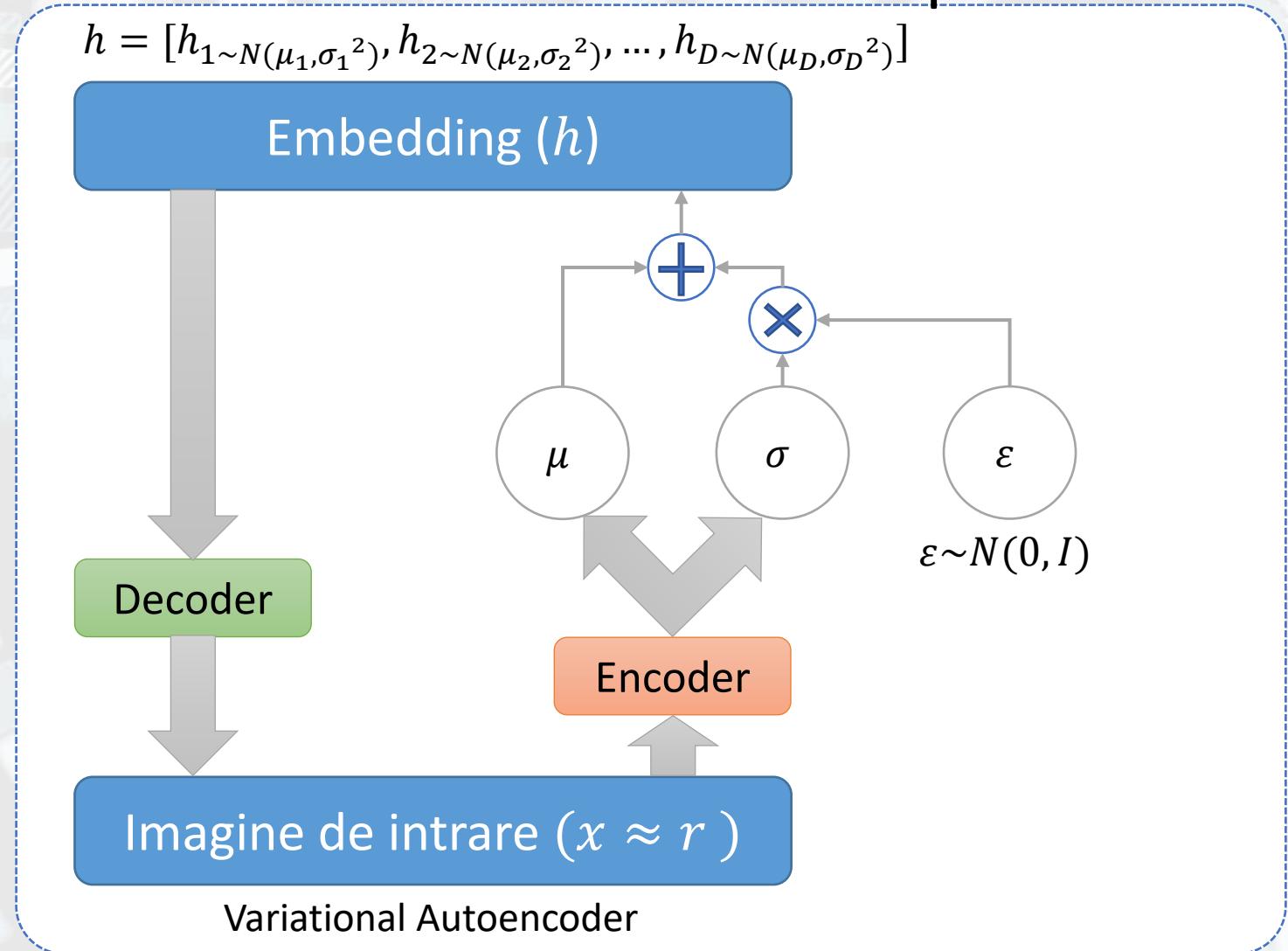
$$h = [h_{1 \sim N(\mu_1, \sigma_1^2)}, h_{2 \sim N(\mu_2, \sigma_2^2)}, \dots, h_{D \sim N(\mu_D, \sigma_D^2)}]$$

- Problemă: eșantionarea dintr-o distribuție de probabilități nu este derivabilă (nu se poate rula backpropagation) => modelul nu poate învăța descriptorul h .
- Soluție: reparametrizare.

$$\begin{aligned} h \sim N(\mu, \sigma^2) &\leftrightarrow \\ h = \mu + \sigma * \varepsilon, \quad \varepsilon \sim N(0,1) \end{aligned}$$

- Partea de eșantionare se mută de la descriptor la ε , în afara rețelei VAE.

Variational Autoencoders – reparametrizare



Variational Autoencoders

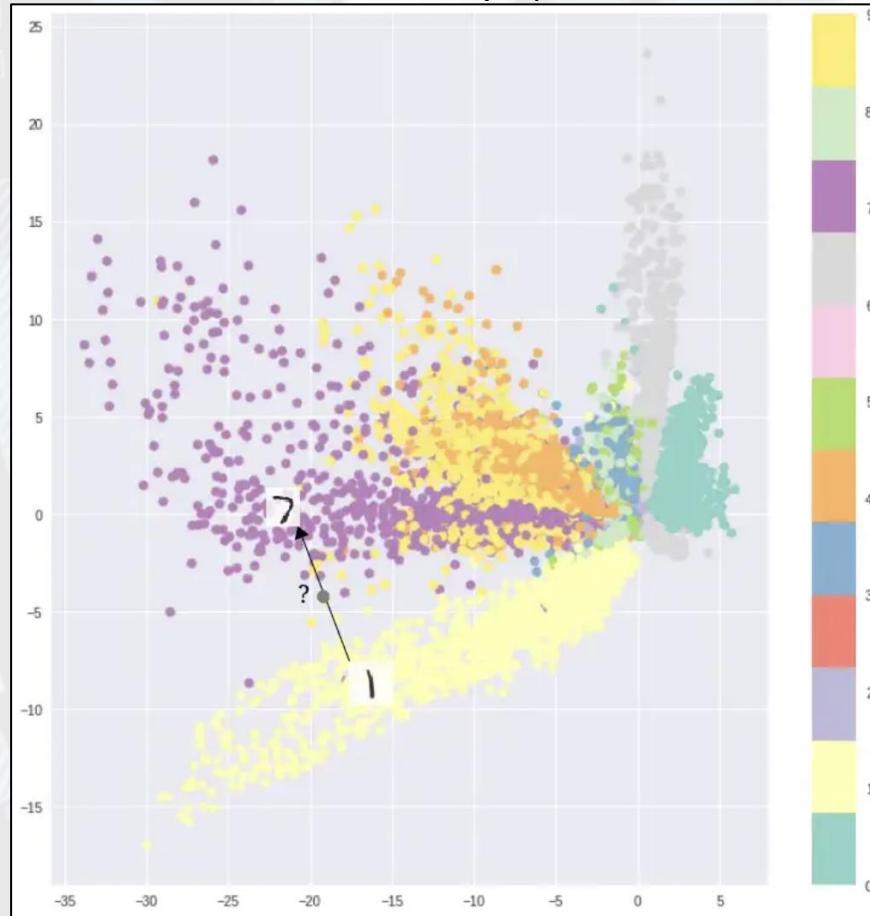
- Antrenarea se face prin minimizarea funcției de cost Evidence Lower Bound (ELBO):

$$L = loss_{reconstruction} + loss_{similarity}$$
$$loss_{reconstruction} = \mathbb{E}_{q_\phi} [\log p_\theta (x|z)]$$
$$loss_{similarity} = -D_{KL}(q_\phi(z) || p(z))$$

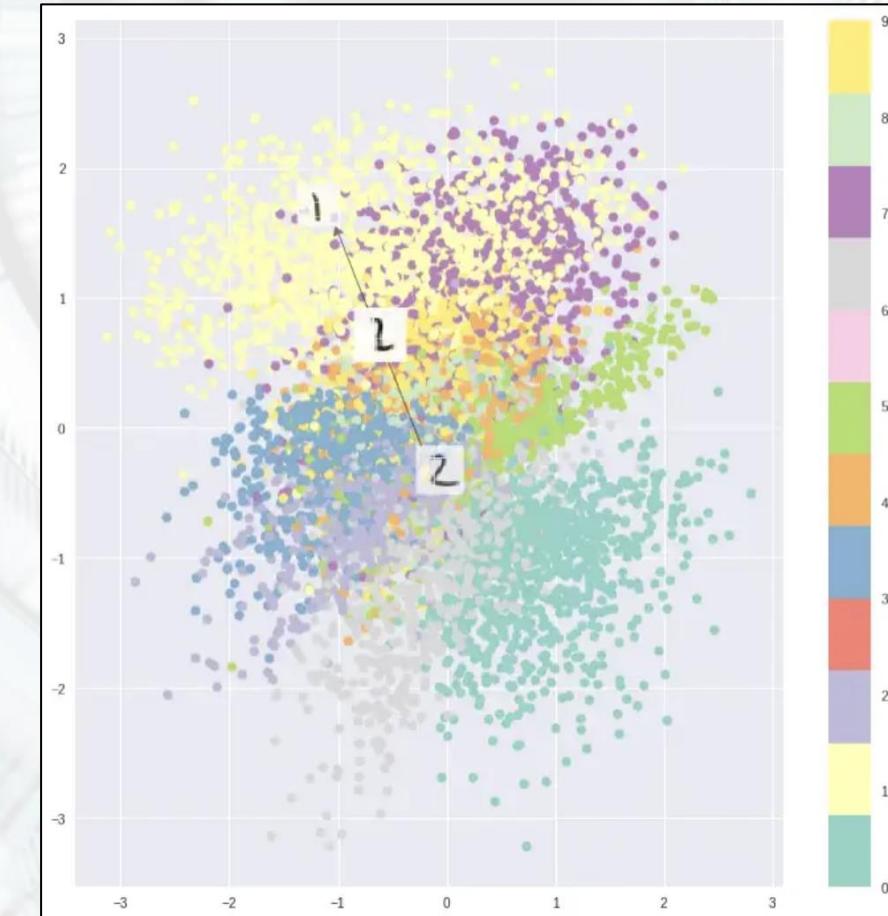
- $loss_{reconstruction}$ reprezintă pierderea asociată reconstrucției imaginii de intrare;
- $loss_{similarity}$ reprezintă distanța între distribuția învățată de codor și distribuția ideală.
- $z = h$ pe diagrama din slide-ul anterior

Variational Autoencoders

Autoencoder – spațiu discret



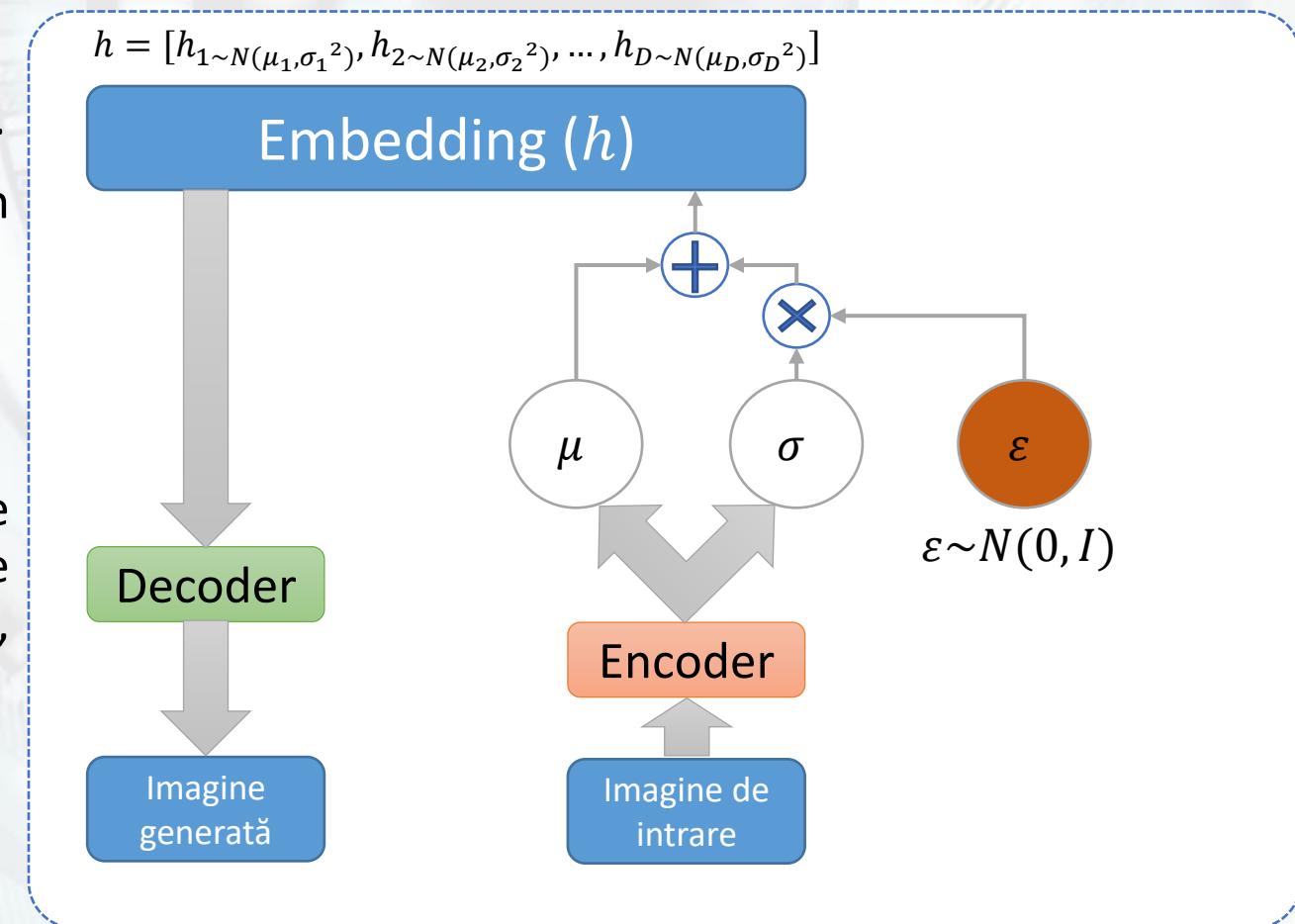
Variational autoencoder – spațiu continuu



Variational Autoencoders – aplicații

1. Generare eșantioane noi:

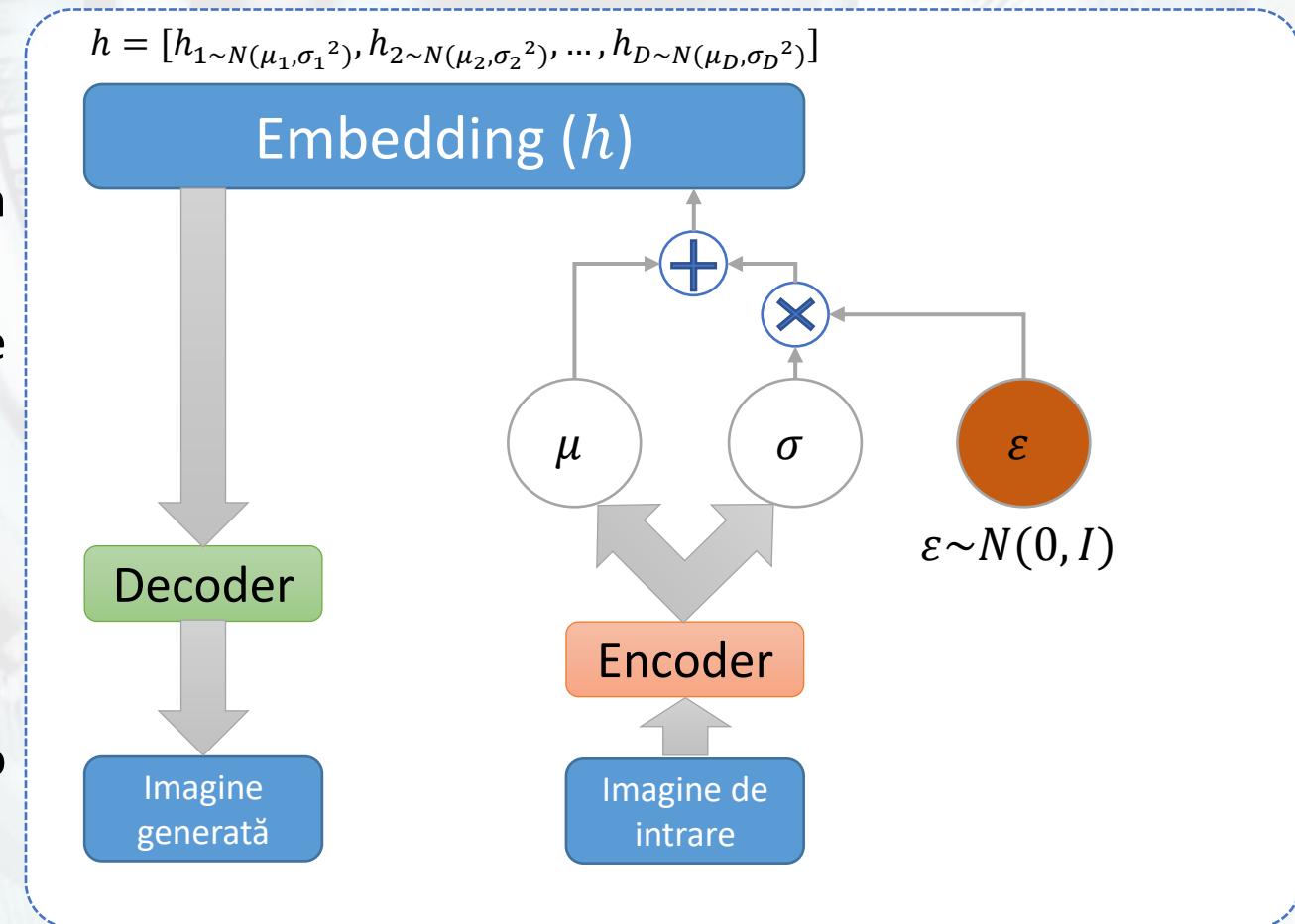
- Se antrenează VAE pe baza de date de train.
- Se procesează o imagine de test și se obțin μ și σ .
- Se eșantionează o valoare aleatoare ε .
- Din μ , σ și ε se generează vectorul latent h .
- h este decodat în spațiul imaginii de către decodor. Imaginea rezultată este o imagine nouă (cu probabilitate aproape egală cu 1), neîntâlnită în baza de date de antrenare.



Variational Autoencoders – aplicații

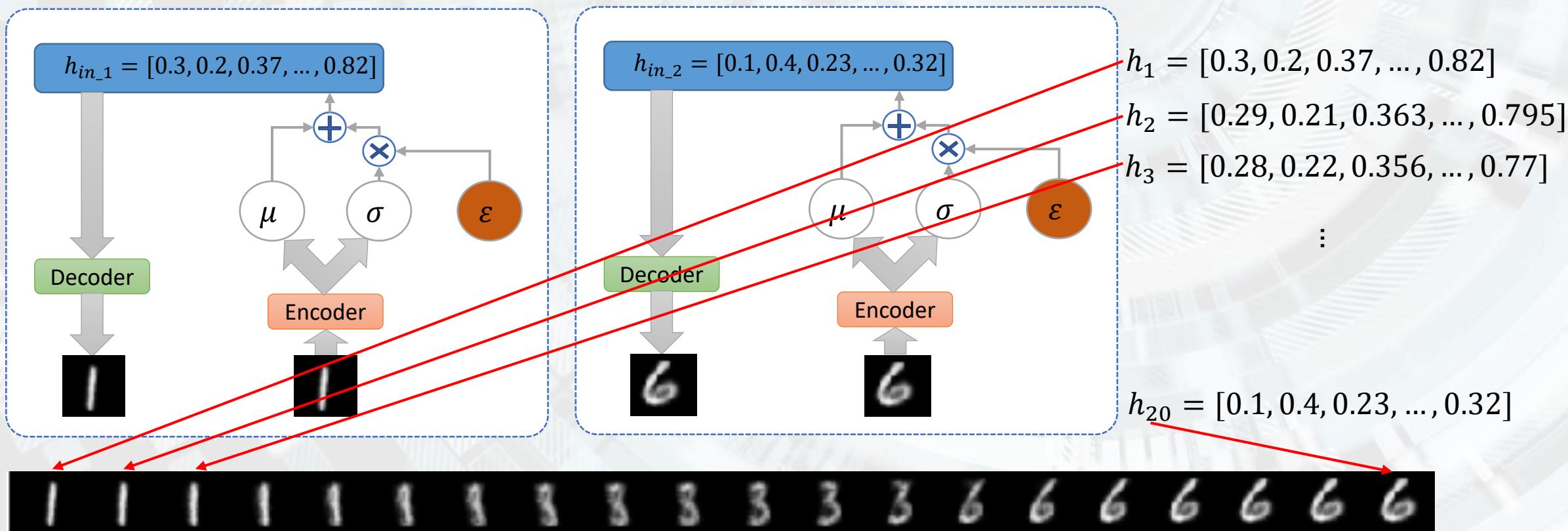
2. Interpolare imagini:

- Se antrenează VAE pe baza de date de train.
- Se procesează 2 imagini de test și se obțin h_{in_1} și h_{in_2} .
- Se stabilește un număr de pași de interpolare N, e.g. 10.
- Se generează vectorii latenți de interpolare
$$h_i = h_{in_1} + (h_{in_2} - h_{in_1}) * \frac{i}{N}, i = 0 \dots N,$$
unde N este numărul de pași de interpolare.
- Fiecare vector latent este decodat într-o imagine nouă.



Variational Autoencoders – aplicații

2. Interpolare imagini:



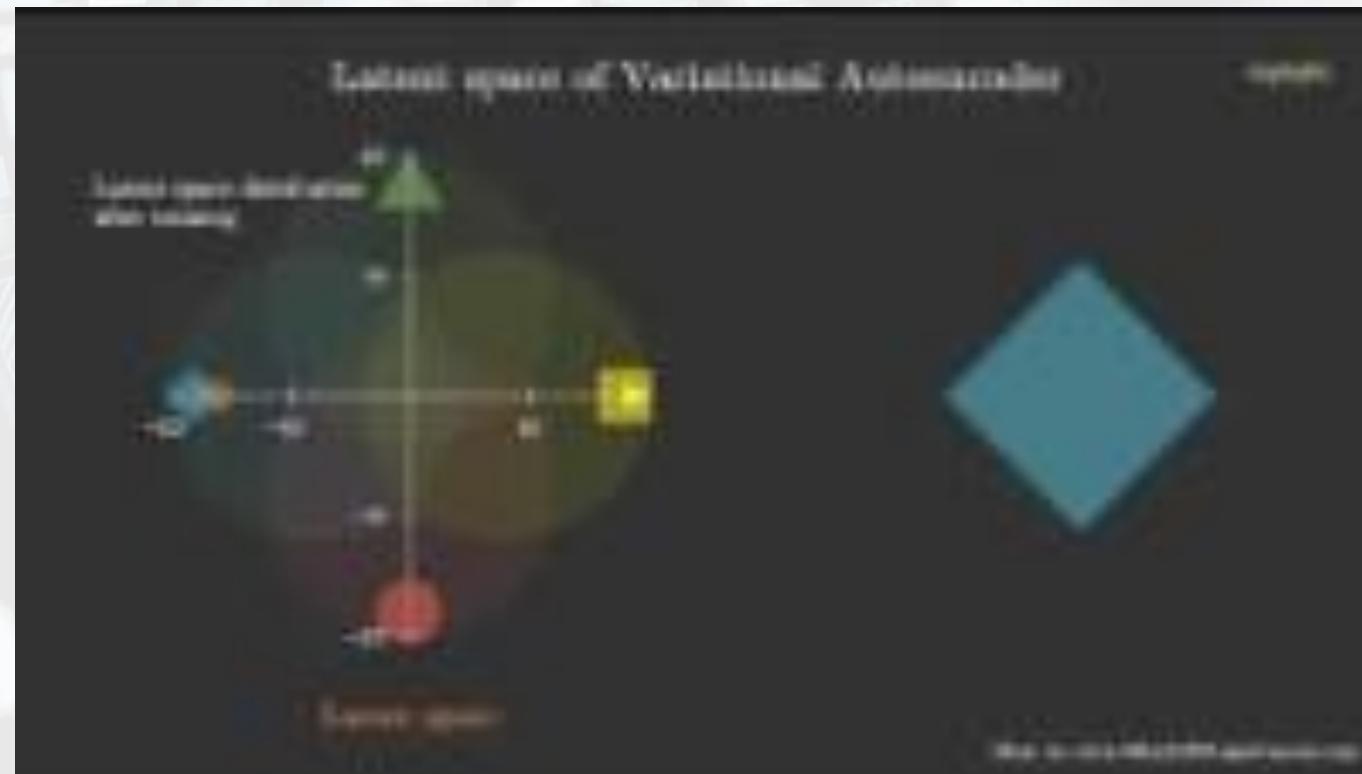
Variational Autoencoders – aplicații

2. Interpolare imagini



Variational Autoencoders – aplicații

2. Interpolare imagini:



https://www.youtube.com/watch?v=sV2FOdGqlX0&t=1s&ab_channel=AqeelAnwar

Variational Autoencoders – aplicații

3. Modificare trăsături specifice:



Trăsături comune?

Zâmbet



Ochelari

Variational Autoencoders – aplicații

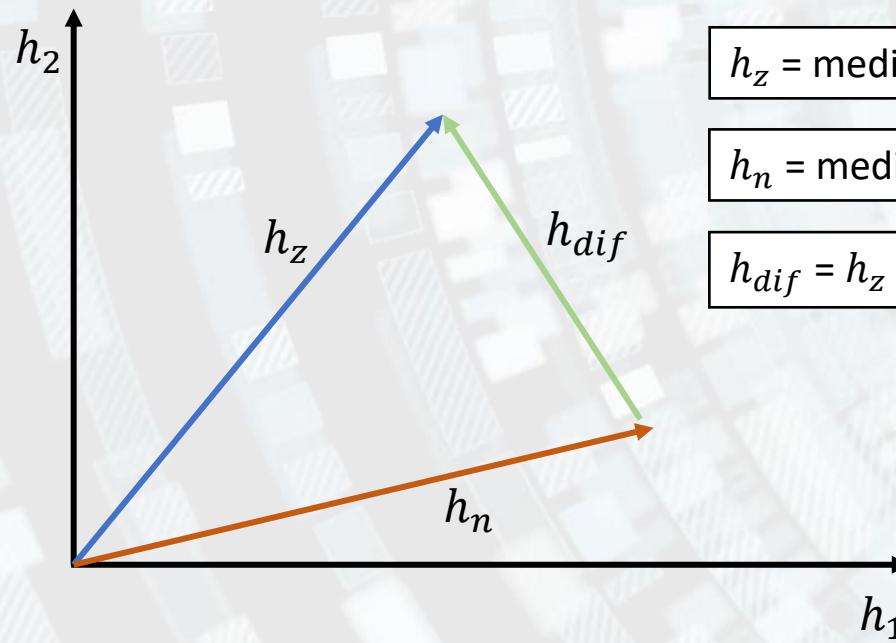
3. Modificare trăsături specifice:

- Valabil doar pentru anumite baze de date în care sunt menționate trăsături specifice.

	5_o_Clock_Shadow	Arched_Eyebrows	Attractive	Bags_Under_Eyes	Bald	Bangs	Big_Lips	Big_Nose	Black_Hair	Blond_Hair	Blurry	Brown_Hair	Bushy_Eyebrows	Chubby
000001.jpg	-1	1	1		-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
000002.jpg	-1	-1	-1		1	-1	-1	-1	1	-1	-1	-1	1	-1
000003.jpg	-1	-1	-1		-1	-1	-1	1	-1	-1	-1	1	-1	-1
000004.jpg	-1	-1	1		-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
000005.jpg	-1	1	1		-1	-1	-1	1	-1	-1	-1	-1	-1	-1
000006.jpg	-1	1	1		-1	-1	-1	1	-1	-1	-1	1	-1	-1
000007.jpg	1	-1	1		1	-1	-1	1	1	1	-1	-1	1	-1
000008.jpg	1	1	-1		1	-1	-1	1	-1	1	-1	-1	-1	-1
000009.jpg	-1	1	1		-1	-1	1	1	-1	-1	-1	-1	-1	-1
000010.jpg	-1	-1	1		-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
000011.jpg	-1	-1	1		-1	-1	-1	-1	1	-1	-1	-1	-1	-1
000012.jpg	-1	-1	1		1	-1	-1	-1	-1	1	-1	-1	1	-1
000013.jpg	-1	-1	-1		-1	-1	-1	-1	-1	-1	1	-1	-1	-1
000014.jpg	1	1	-1		-1	-1	-1	1	1	1	-1	-1	1	-1
000015.jpg	1	-1	-1		1	-1	-1	-1	1	-1	-1	-1	-1	-1
000016.jpg	1	-1	-1		1	-1	-1	-1	-1	-1	-1	-1	-1	-1
000017.jpg	-1	-1	-1		-1	-1	-1	-1	-1	1	-1	-1	-1	-1
000018.jpg	-1	1	-1		-1	-1	-1	1	-1	1	-1	-1	-1	-1
000019.jpg	-1	1	1		-1	-1	-1	-1	-1	-1	1	-1	-1	-1
000020.jpg	-1	-1	-1		-1	-1	-1	-1	-1	1	-1	-1	-1	1
000021.jpg	-1	-1	-1		-1	-1	-1	-1	1	-1	-1	-1	-1	-1
000022.jpg	-1	1	-1		-1	-1	-1	1	-1	-1	1	-1	-1	-1
000023.jpg	1	-1	1		-1	-1	-1	-1	1	-1	-1	1	-1	-1
000024.jpg	-1	1	1		-1	-1	-1	1	-1	-1	1	-1	-1	-1
000025.jpg	1	-1	-1		1	-1	-1	-1	1	-1	-1	-1	-1	-1

Variational Autoencoders – aplicații

3. Modificare trăsături specifice:

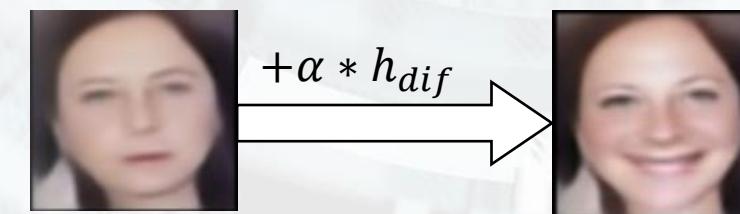


h_z = media tuturor vectorilor latenți ai eșantioanelor în care apar persoane zâmbind

h_n = media tuturor vectorilor latenți ai eșantioanelor în care nu apar persoane zâmbind

$h_{dif} = h_z - h_n$ = vectorul care codează zâmbetul

$$h_{nou} = h_{original} + \alpha * h_{dif}$$



Variational Autoencoders

Limitare: deși VAE sunt capabile să genereze imagini complet noi, acestea sunt, de obicei, încețoșate. Acest lucru se datorează zgomotului introdus de funcția de cost L2, care mediază diferențele dintre pixeli.

Soluție: utilizarea unui ansamblu care ia în calcul și „realismul” imaginii.

M4.4. Generative Adversarial Networks (GANs)

Generative Adversarial Networks

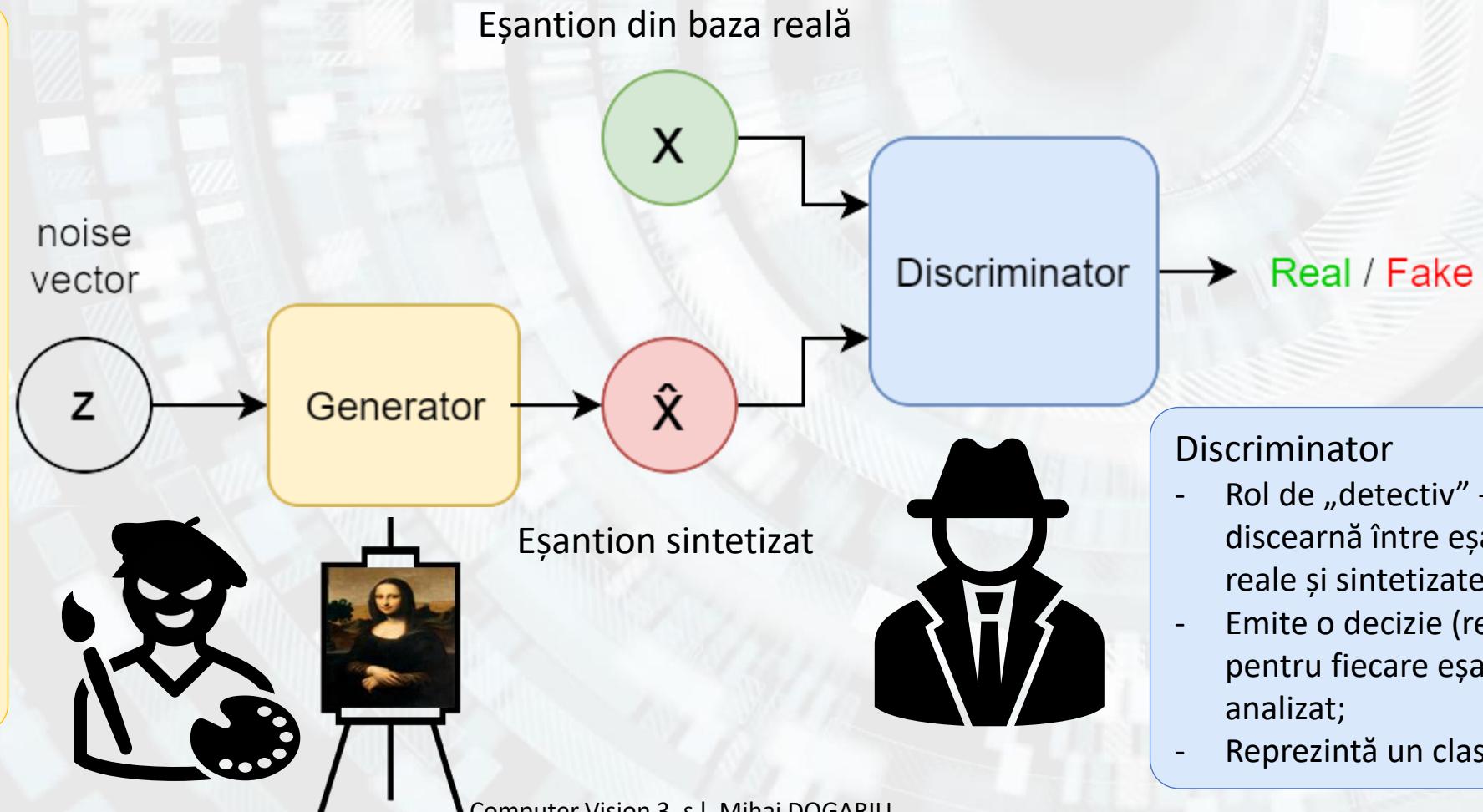
Rețelele generative adversariale [4] reprezintă o clasă de rețele neuronale specializate în generarea de exemple realiste, asemănătoare cu datele de antrenare.

- Modelul descoperă caracteristicile distribuției de probabilitate a datelor de antrenare și le folosește pentru a genera eșantioane noi.
- Ansamblul este compus din 2 rețele neuronale: un generator și un discriminator.
 - Generatorul are sarcina de a genera exemple noi.
 - Discriminatorul are sarcina de a detecta exemplele generate sintetic.
- GAN se bazează pe teoria jocurilor: joc cu sumă 0 (zero-sum game).

Generative Adversarial Networks

Generator

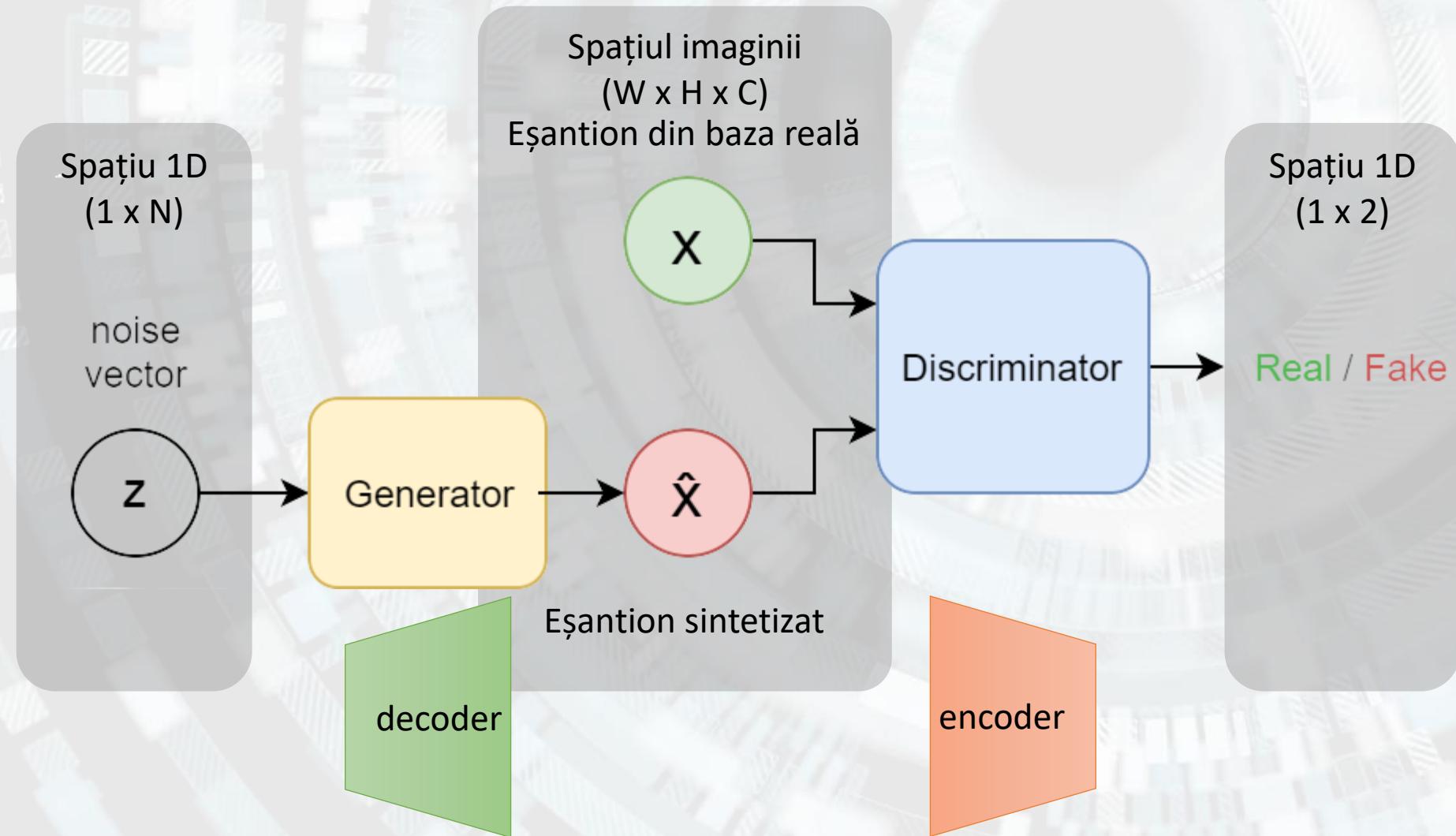
- Rol de „falsificator” – încearcă să păcălească discriminatorul;
- Preia vectorul de zgomot (normal, Gaussian) și îl transformă în spațiul datelor;
- Generează date cât mai plauzibile (cât mai asemănătoare cu cele reale).



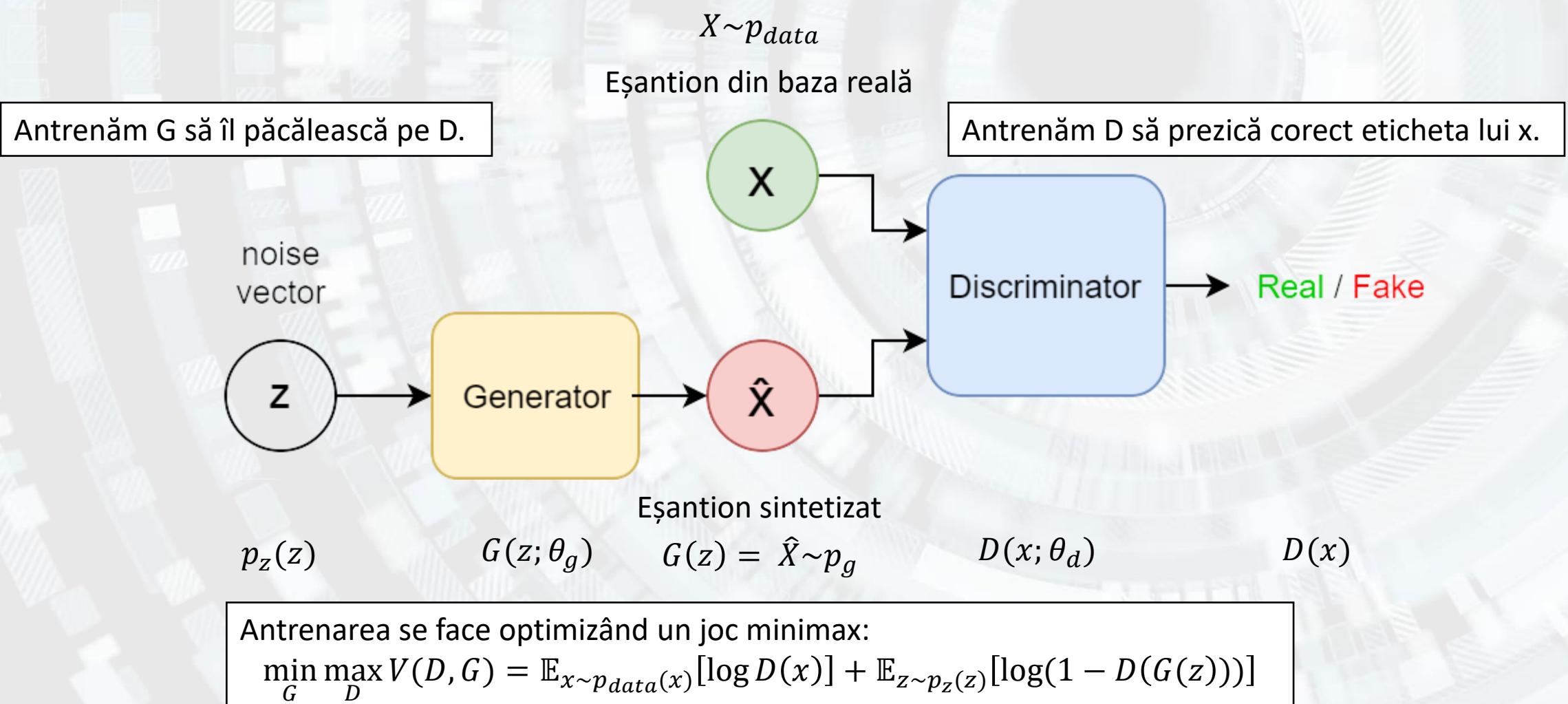
Discriminator

- Rol de „detectiv” – încearcă să discearnă între eșanțioane reale și sintetizate;
- Emite o decizie (real/fals) pentru fiecare eșantion analizat;
- Reprezintă un clasificator binar.

Generative Adversarial Networks



Generative Adversarial Networks



Generative Adversarial Networks

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**

for k steps **do**

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

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

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.

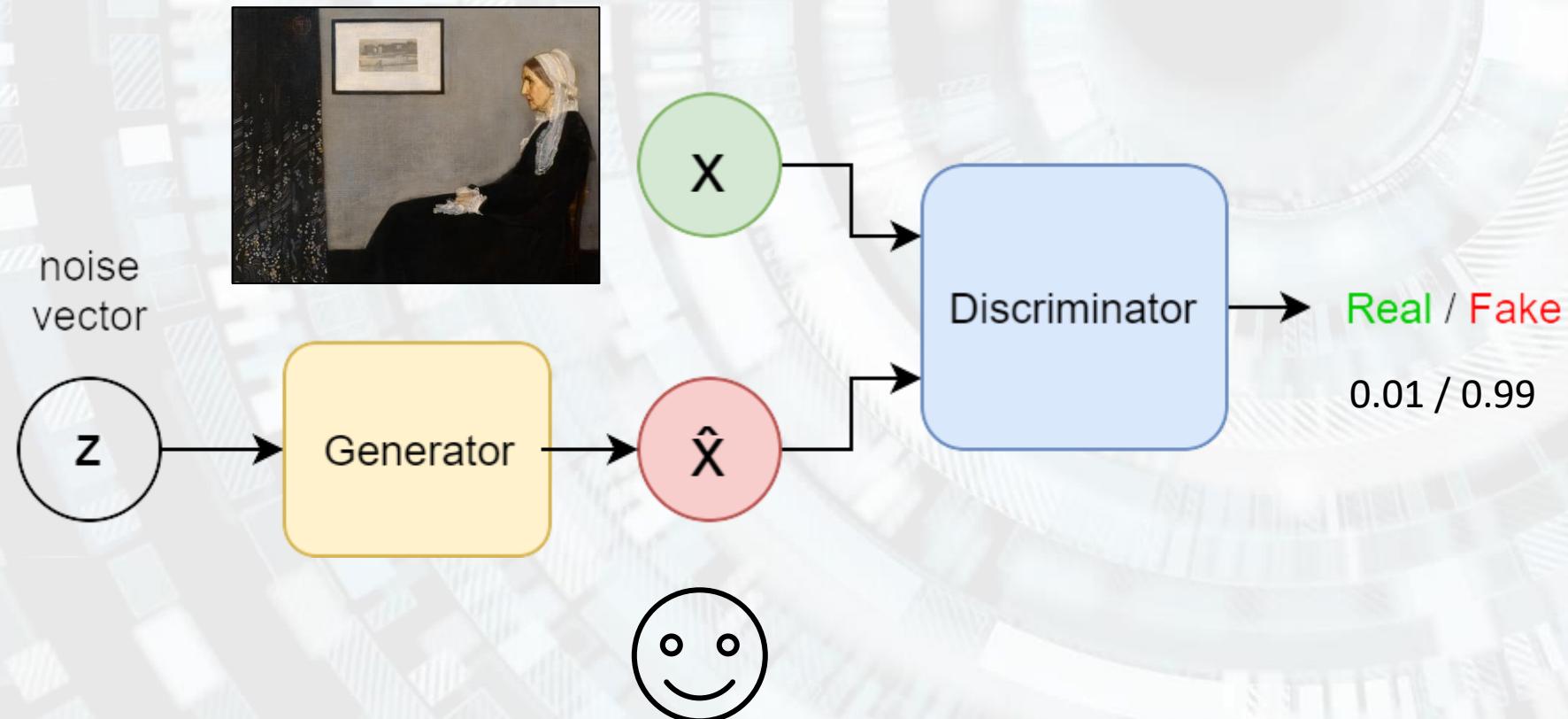
- Update the generator by descending its stochastic gradient:

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

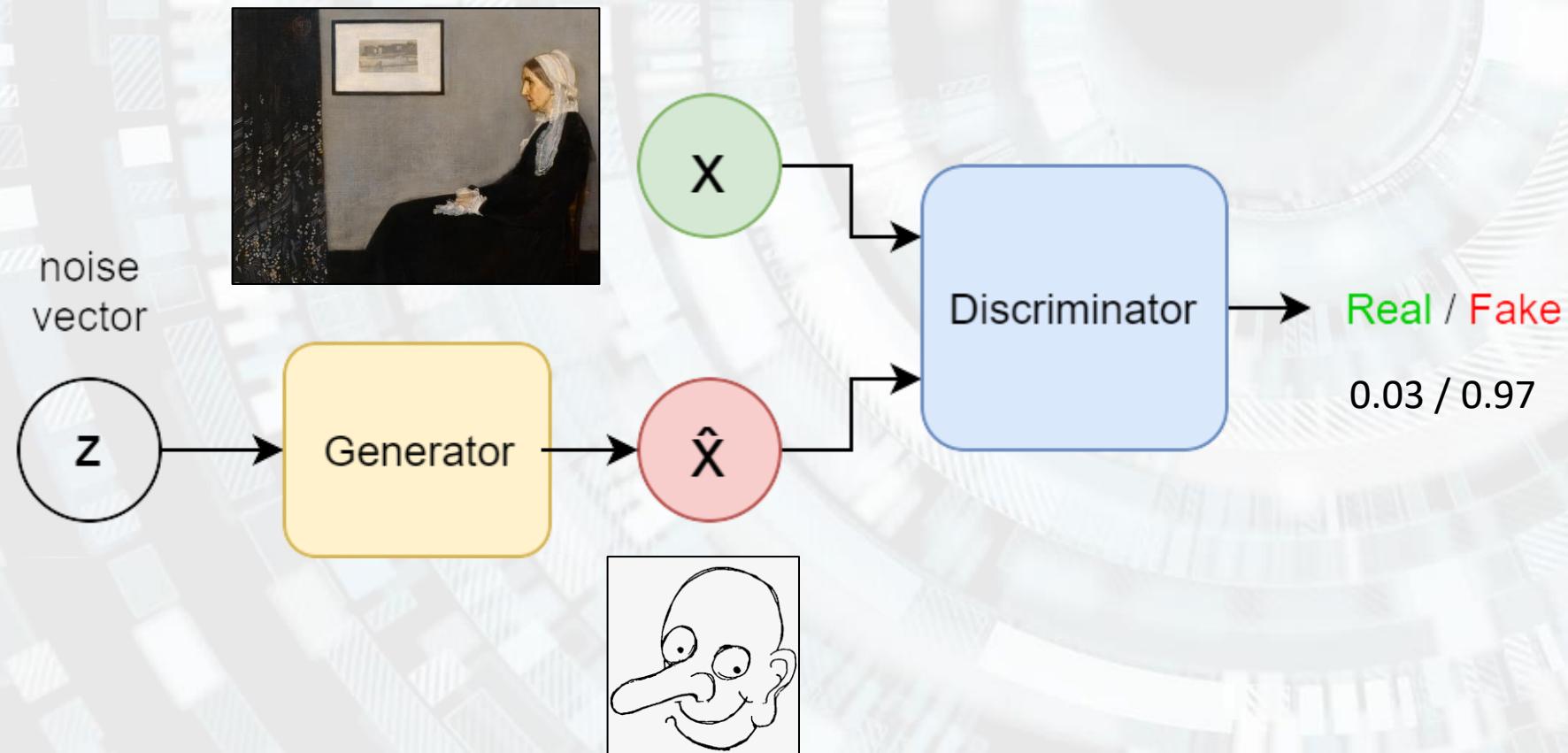
end for

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

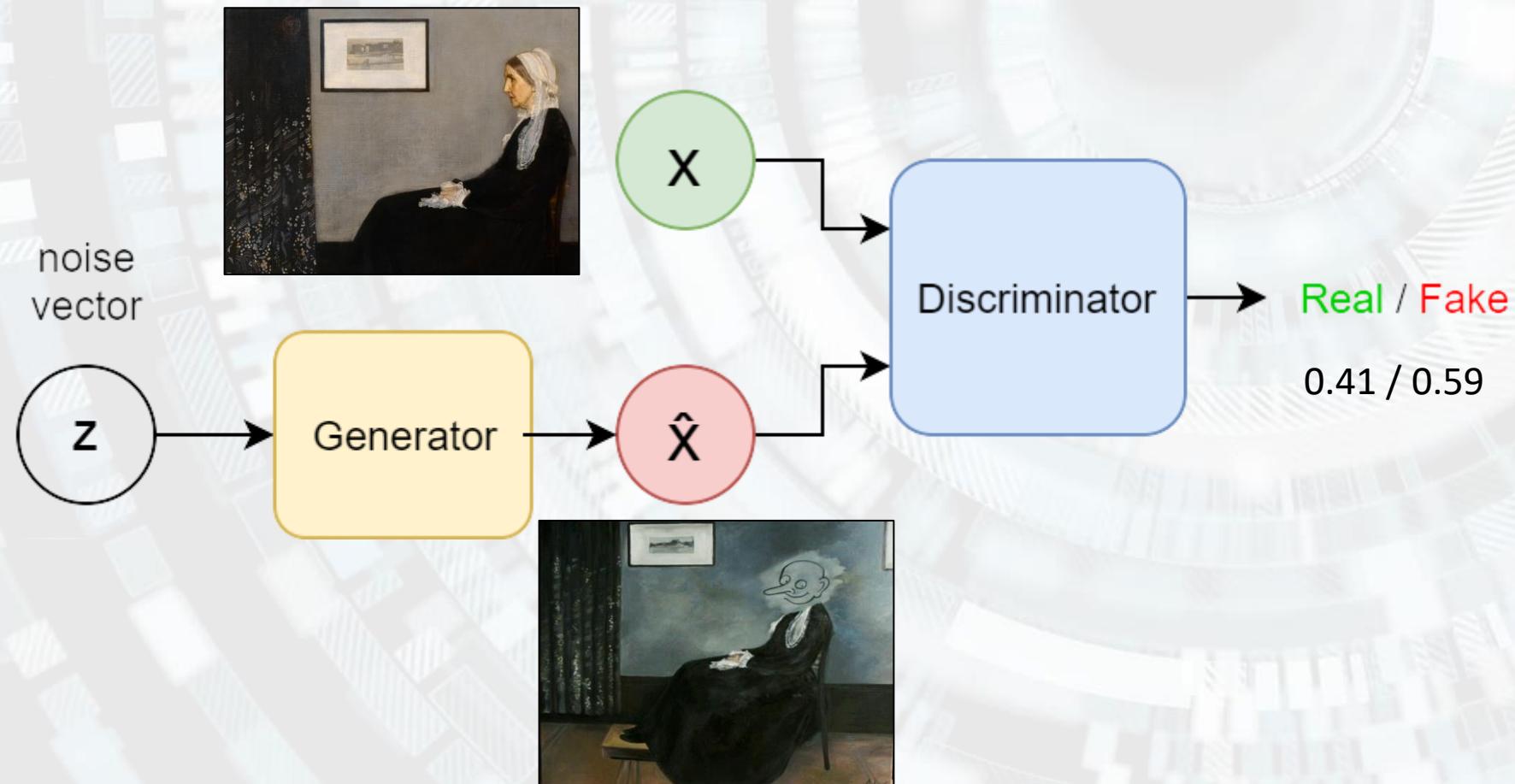
Generative Adversarial Networks



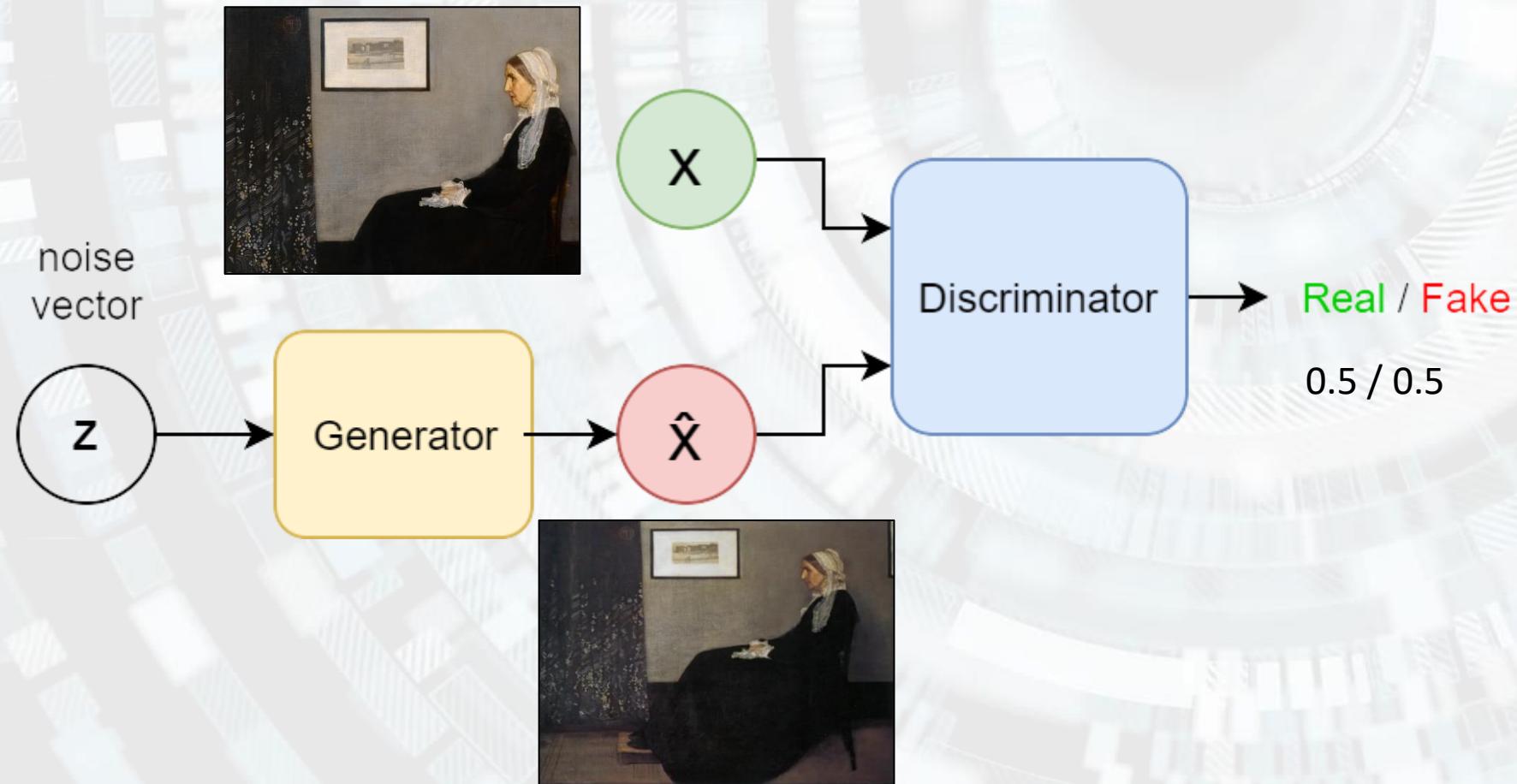
Generative Adversarial Networks



Generative Adversarial Networks



Generative Adversarial Networks



Generative Adversarial Networks

- Modelele GAN micșorează divergența Jensen-Shanon între distribuția datelor reale și cea a modelului ($JSD(p_{data} \parallel p_G)$).
- Convergența se atinge atunci când discriminatorul confundă eșantioanele reale cu cele sintetizate ($p_{data} = p_G$). În acest caz,
 $D(x) = \frac{1}{2}, \forall x.$
- Antrenarea unui GAN presupune atingerea unui echilibru Nash – niciuna dintre părțile implicate (generator/discriminator) nu își poate îmbunătăți starea făcând modificări la parametrii proprii.

Generative Adversarial Networks

Limitări:

- Antrenarea este foarte instabilă și convergența greu de atins => mode collapse: generatorul învață să genereze un set redus de exemple ce păcălesc discriminatorul.

8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	3	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8



Generative Adversarial Networks

Limitări:

- GAN-urile au nevoie de baze de date foarte mari pentru antrenare (o distribuție a datelor cât mai densă și uniformă). Altfel, se obține overfitting, asemănător cu mode collapse.

Generative Adversarial Networks

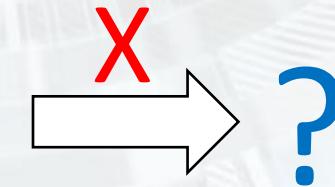
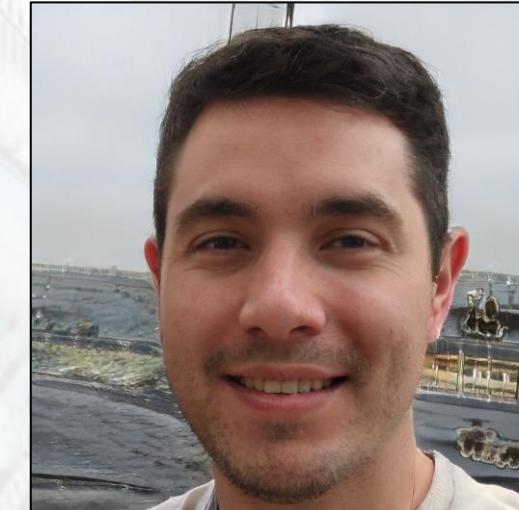
Limitări:

- Evaluarea GAN-urilor depinde de sarcina pe care o îndeplinesc. Nu există o metodă standardizată de a evalua aceste modele.
 - Inception Score (IS) [5] – se folosește un model Inceptionv3 pre-antrenat pe Imagenet. Se procesează un număr mare de eșantioane generate (~30k).
 - Fiecare imagine generată trebuie să aibă o singură clasă prezisă cu încredere mare => imaginea generată este clară sau unică.
 - Clasele prezise pe întregul set de date sintetizate trebuie să fie uniform distribuite => imaginile generate sunt diverse.
 - Fréchet Inception Distance (FID) [6] – folosește tot Inceptionv3 pre-antrenat pe Imagenet.
 - Se elimină ultimul strat al rețelei pre-antrenate Inceptionv3.
 - Se extrag trăsăturile prin Inceptionv3 din eșantioanele reale și din cele sintetizate.
 - Se calculează o distanță între măsurile statistice ale eșantioanelor extrase.

Generative Adversarial Networks

Limitări:

- Conceptual, GAN-urile sunt antrenate să genereze eșantioane noi pornind de la zgomot => obținerea zgomotului care a generat un eșantion nu este facilă.



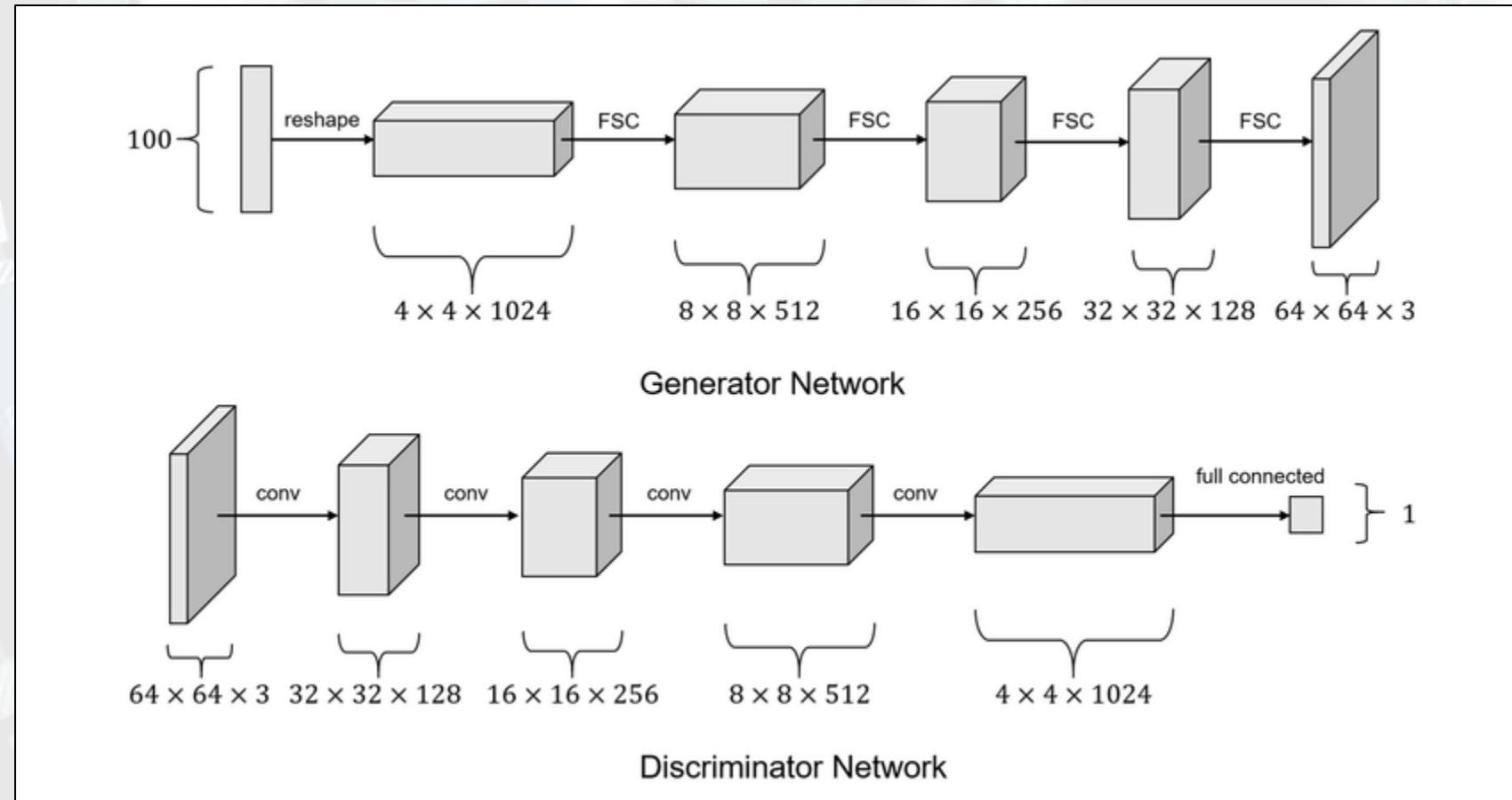
Generative Adversarial Networks

Avantaje:

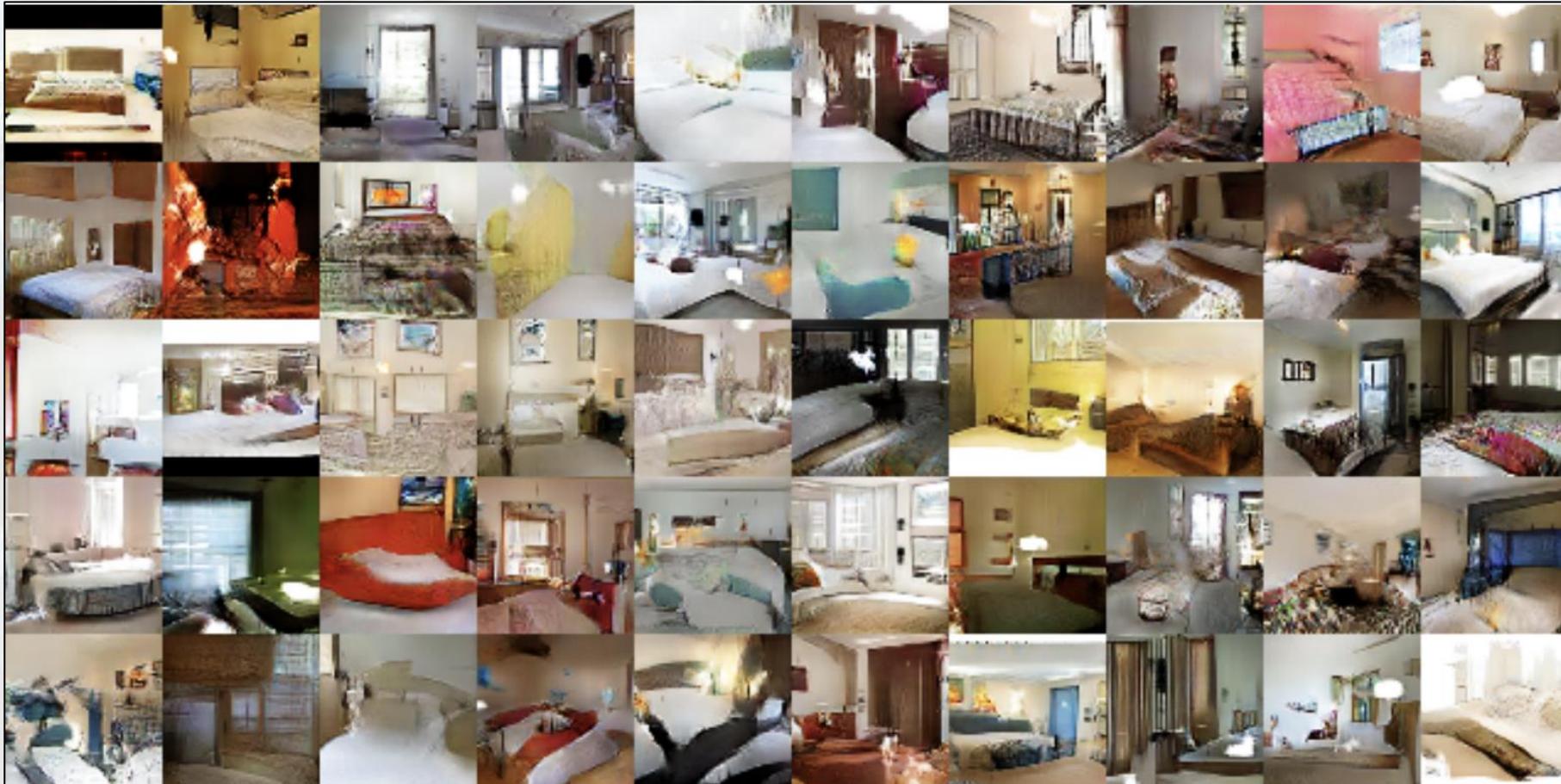
- Imagini generate foarte clare;
- Inferență rapidă (se pot genera sute de imagini/s);
- Ușor de utilizat pentru transferul de stil;
- Deschid aplicații în toate domeniile (medical, auto, artistic, inginerie, educație etc.)
- Numeroase implementări disponibile open-source.

GANs – exemple de modele

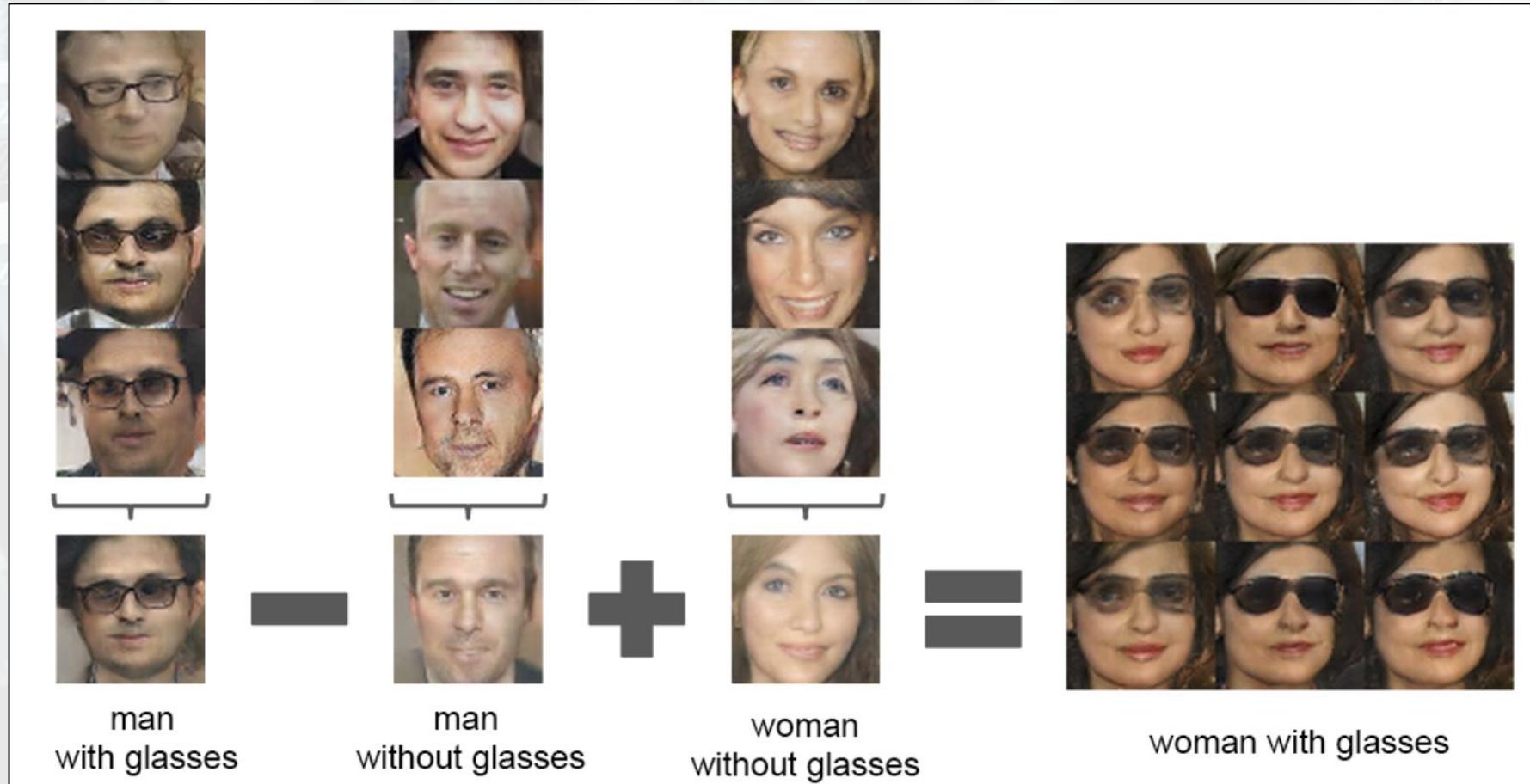
Generative Adversarial Networks – DCGAN [7]



Generative Adversarial Networks – DCGAN



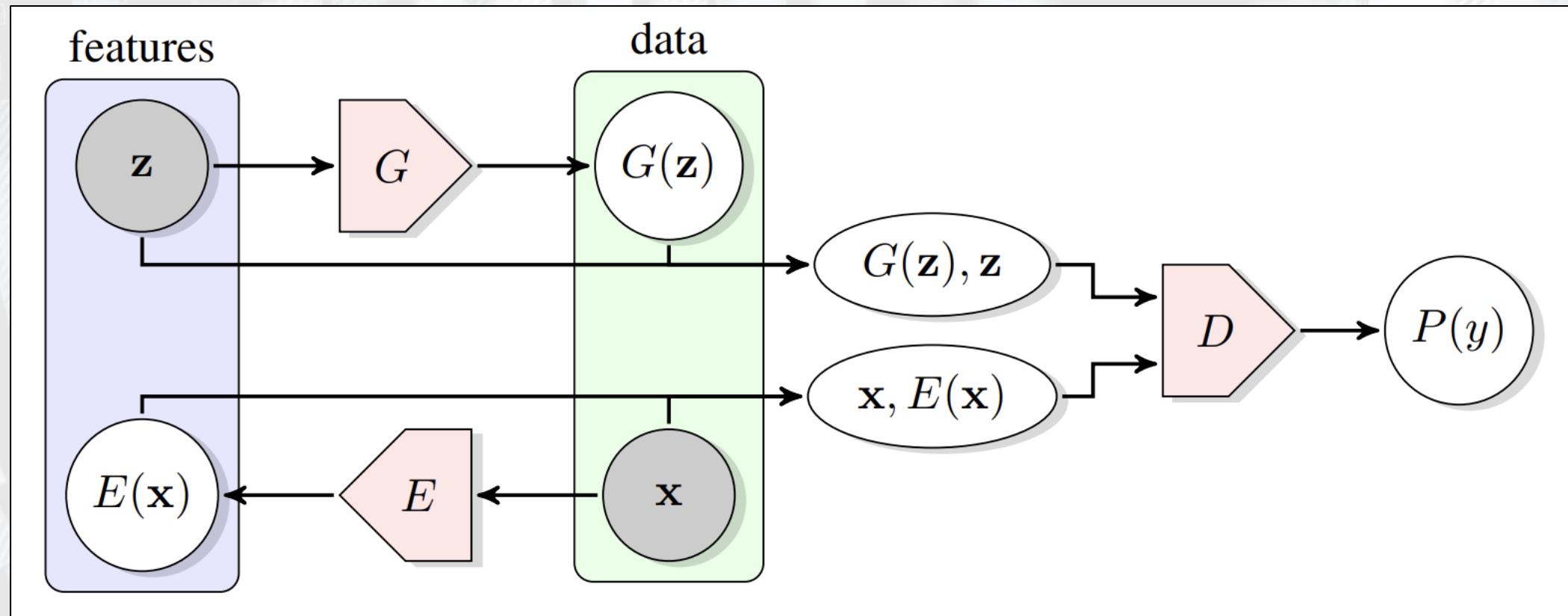
Generative Adversarial Networks – DCGAN



Generative Adversarial Networks – DCGAN



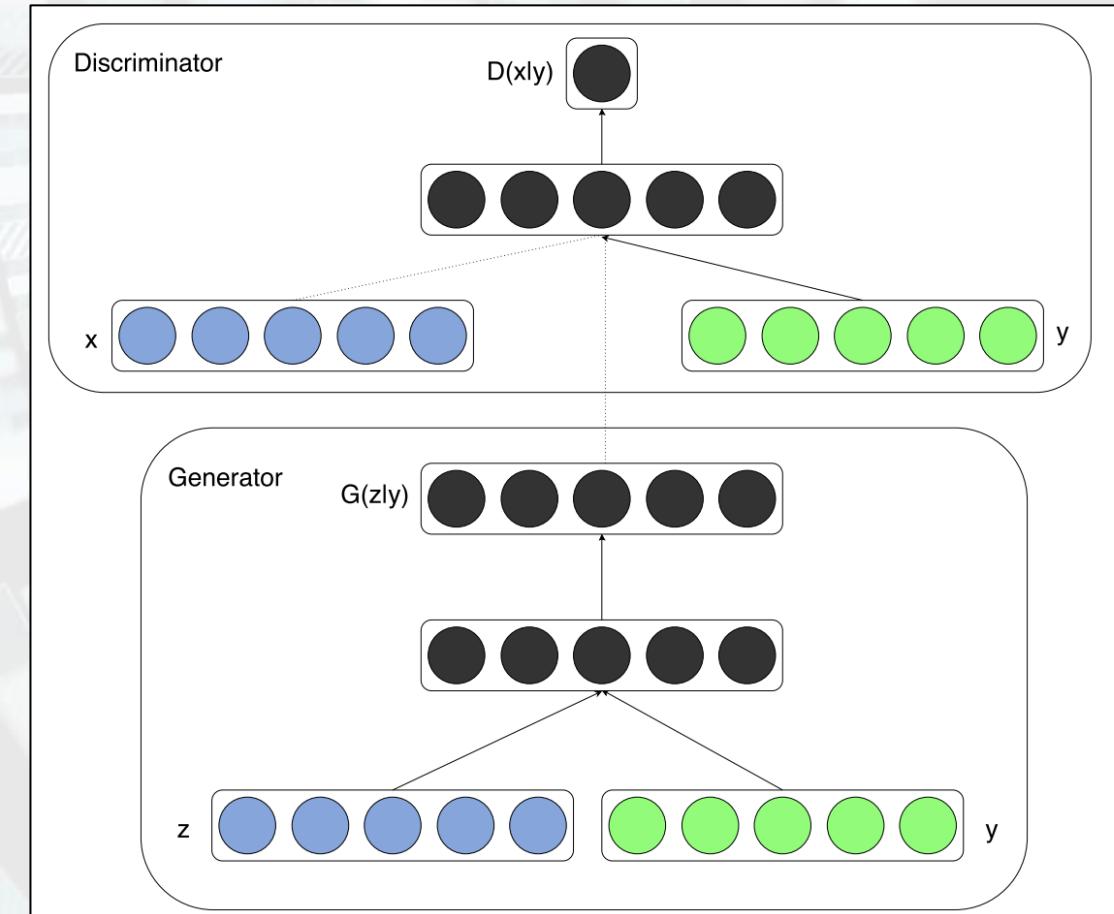
Generative Adversarial Networks – BiGAN [8]



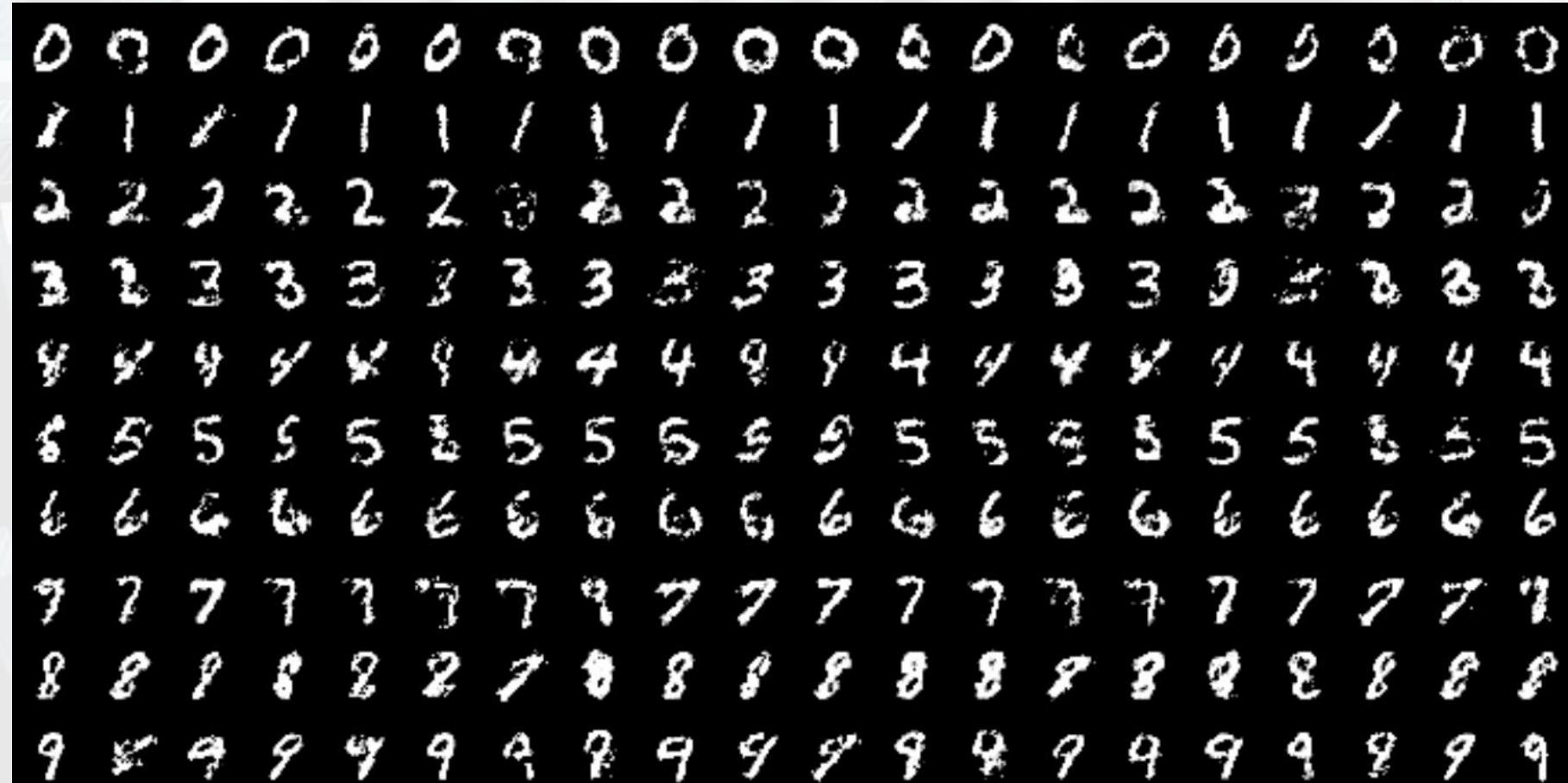
Generative Adversarial Networks – BiGAN

$G(\mathbf{z})$	
\mathbf{x}	
$G(E(\mathbf{x}))$	
$G(\mathbf{z})$	
\mathbf{x}	
$G(E(\mathbf{x}))$	

Generative Adversarial Networks – cGAN [9]

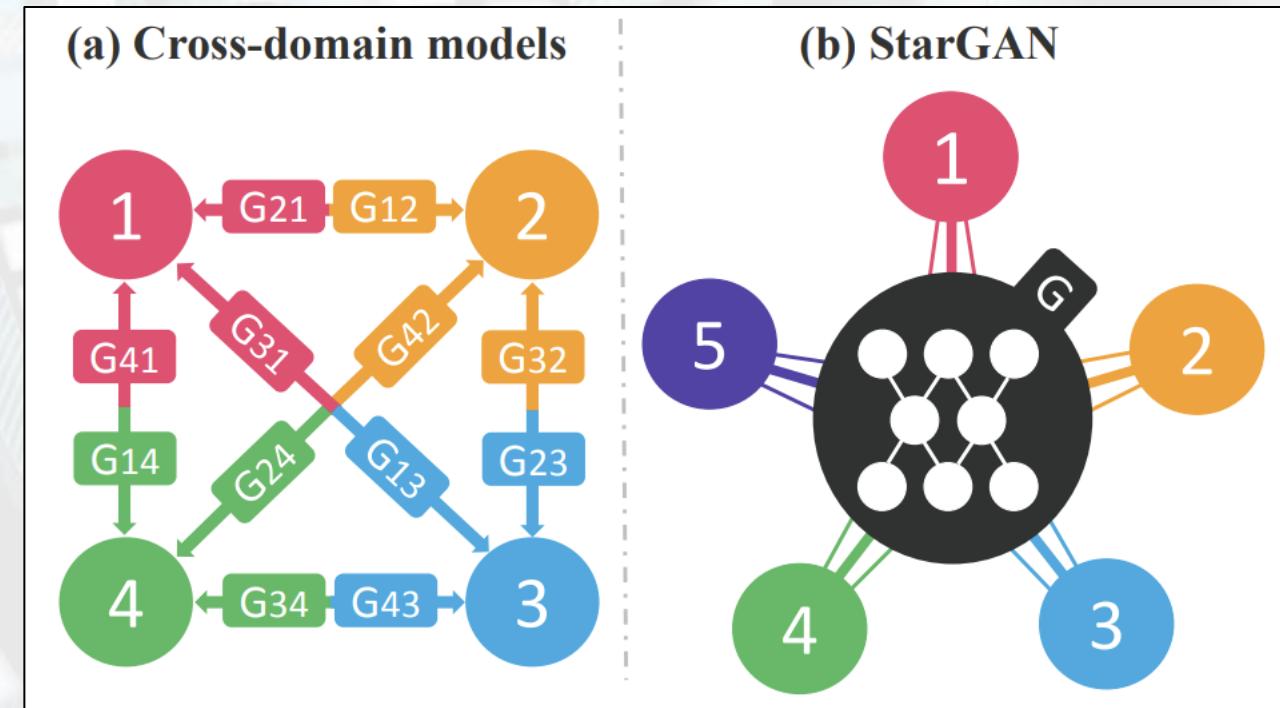


Generative Adversarial Networks – cGAN



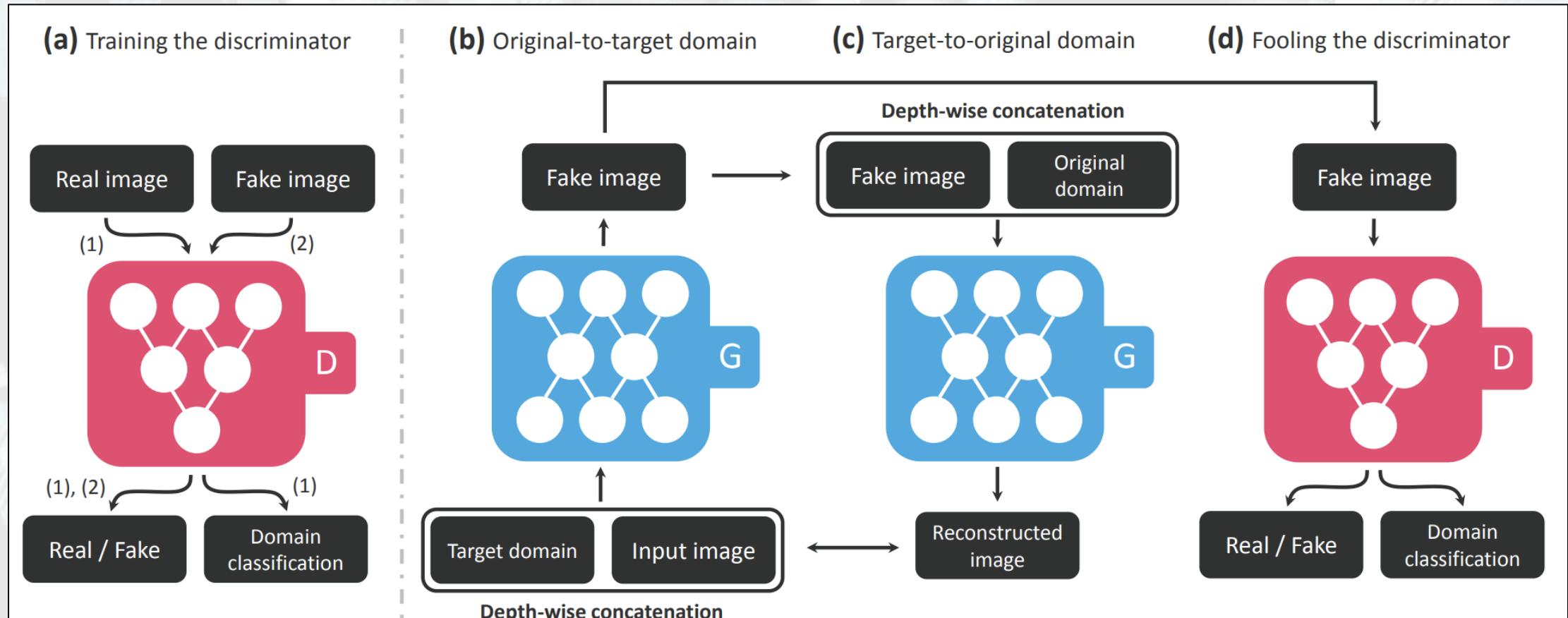
Generative Adversarial Networks – StarGAN

[10]

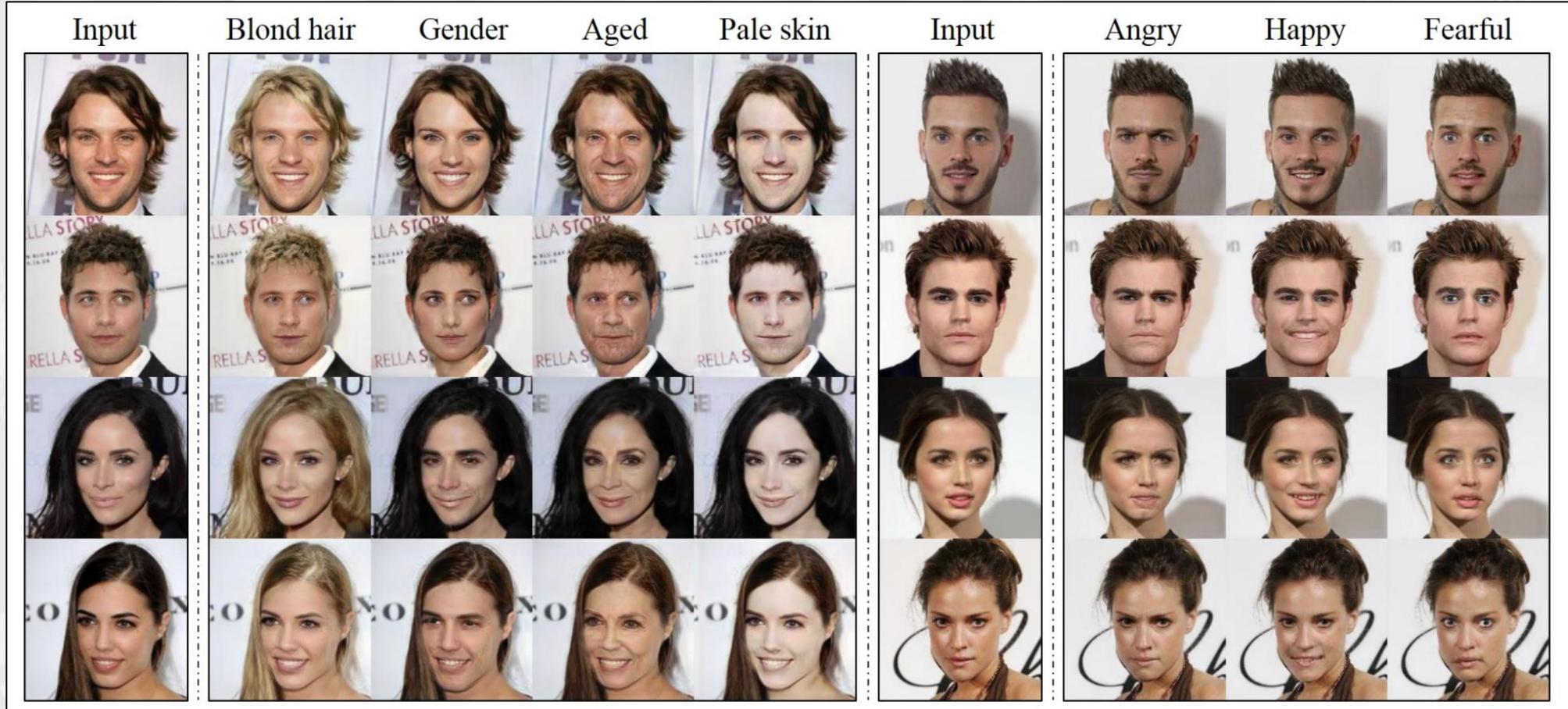


Generative Adversarial Networks – StarGAN

[10]

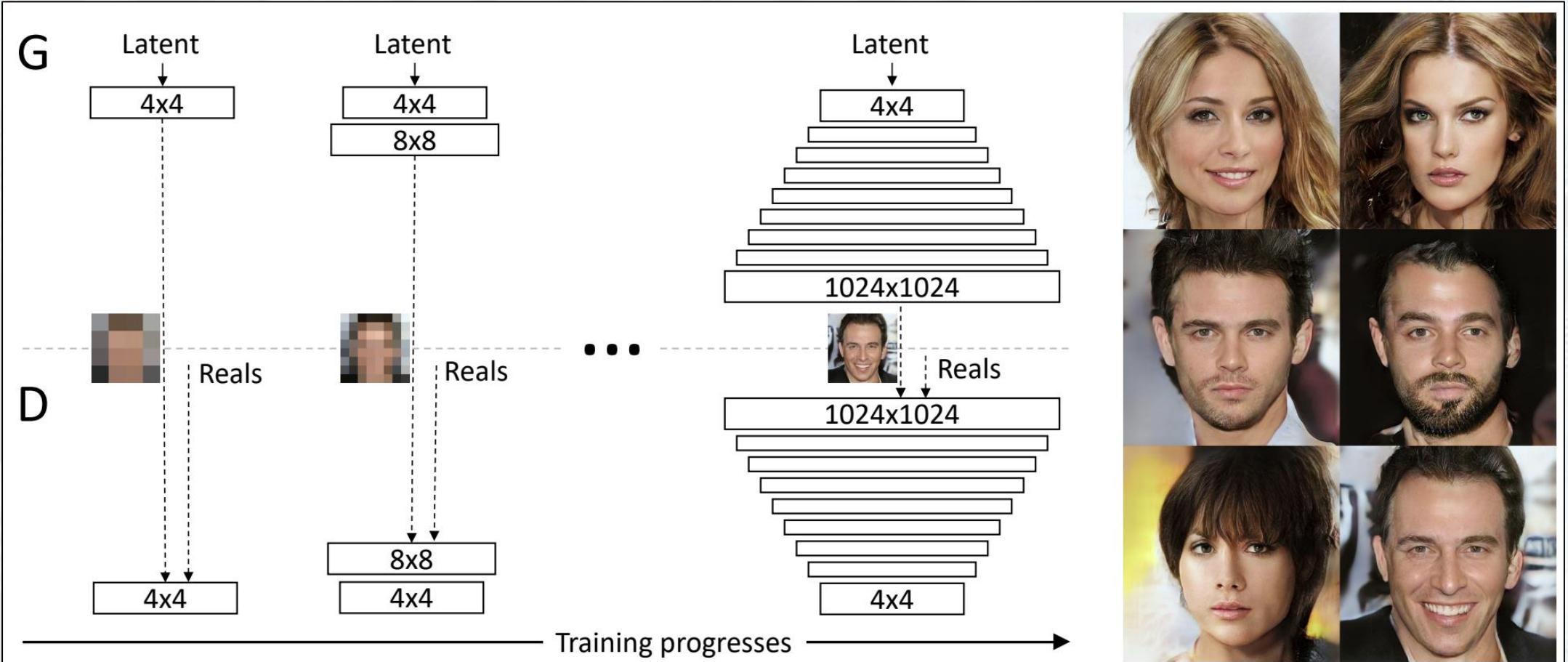


Generative Adversarial Networks – StarGAN [10]

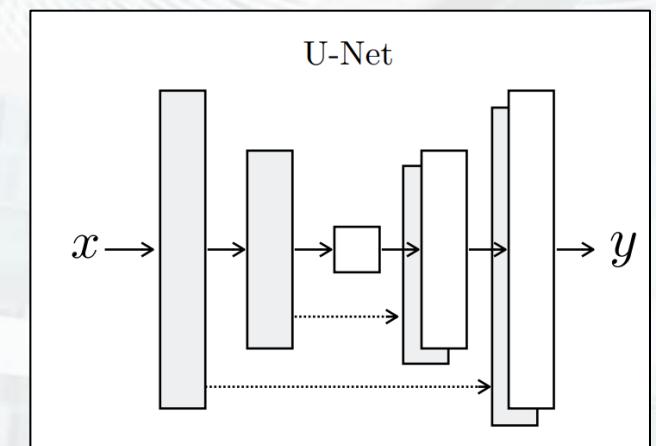
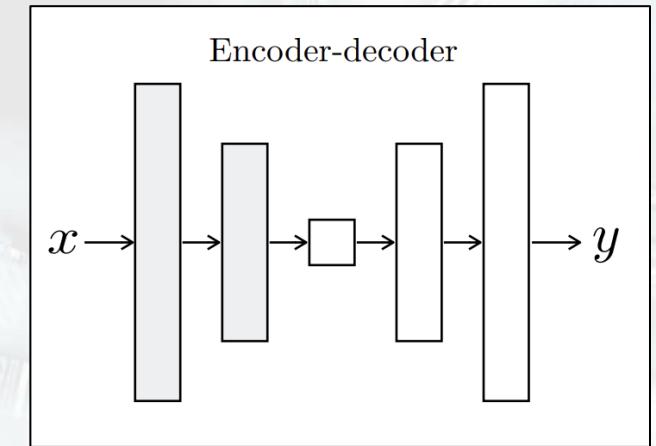
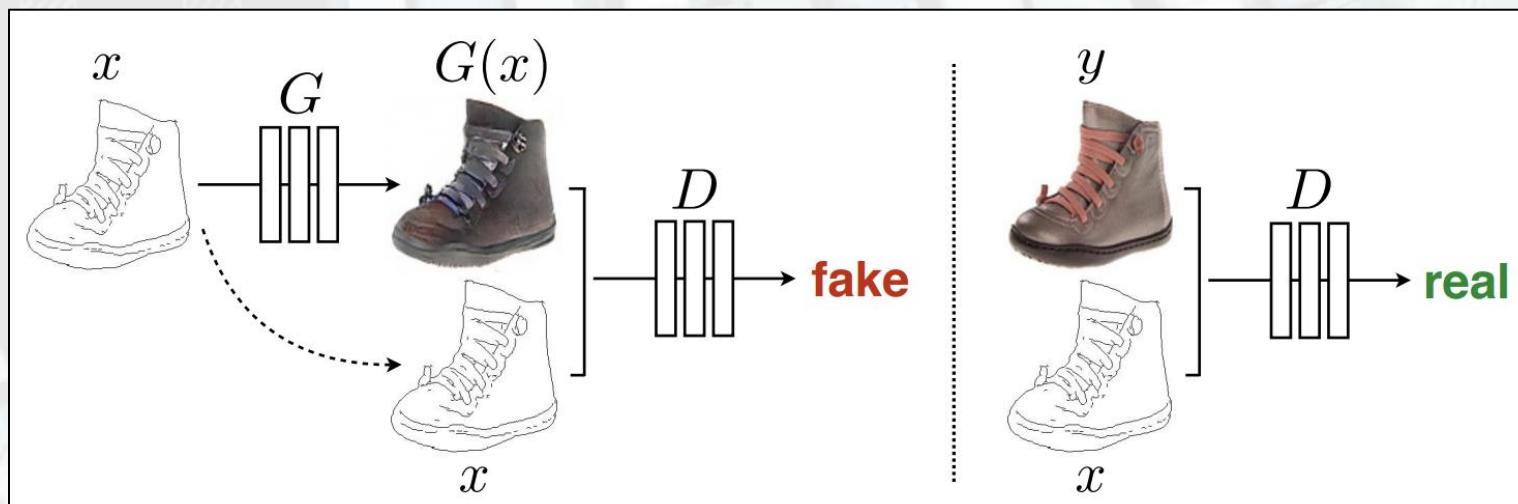


Generative Adversarial Networks – ProGAN

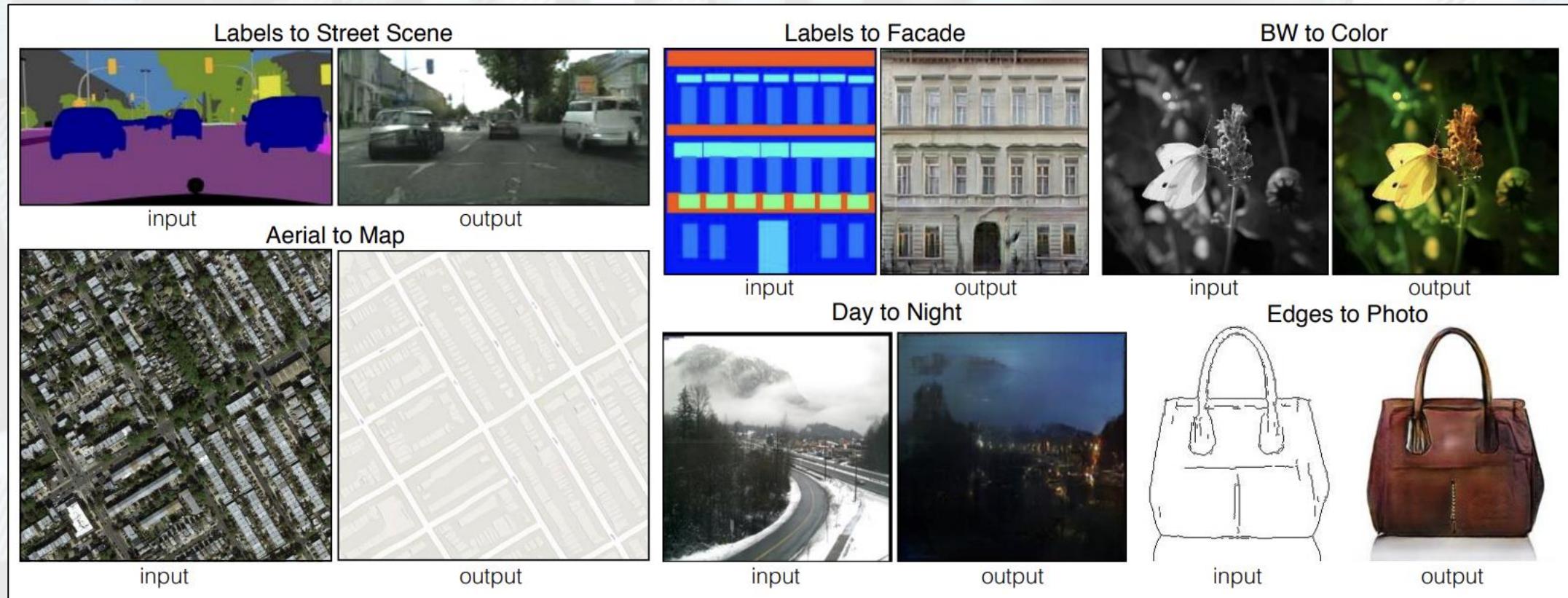
[11]



Generative Adversarial Networks – im2im [12]

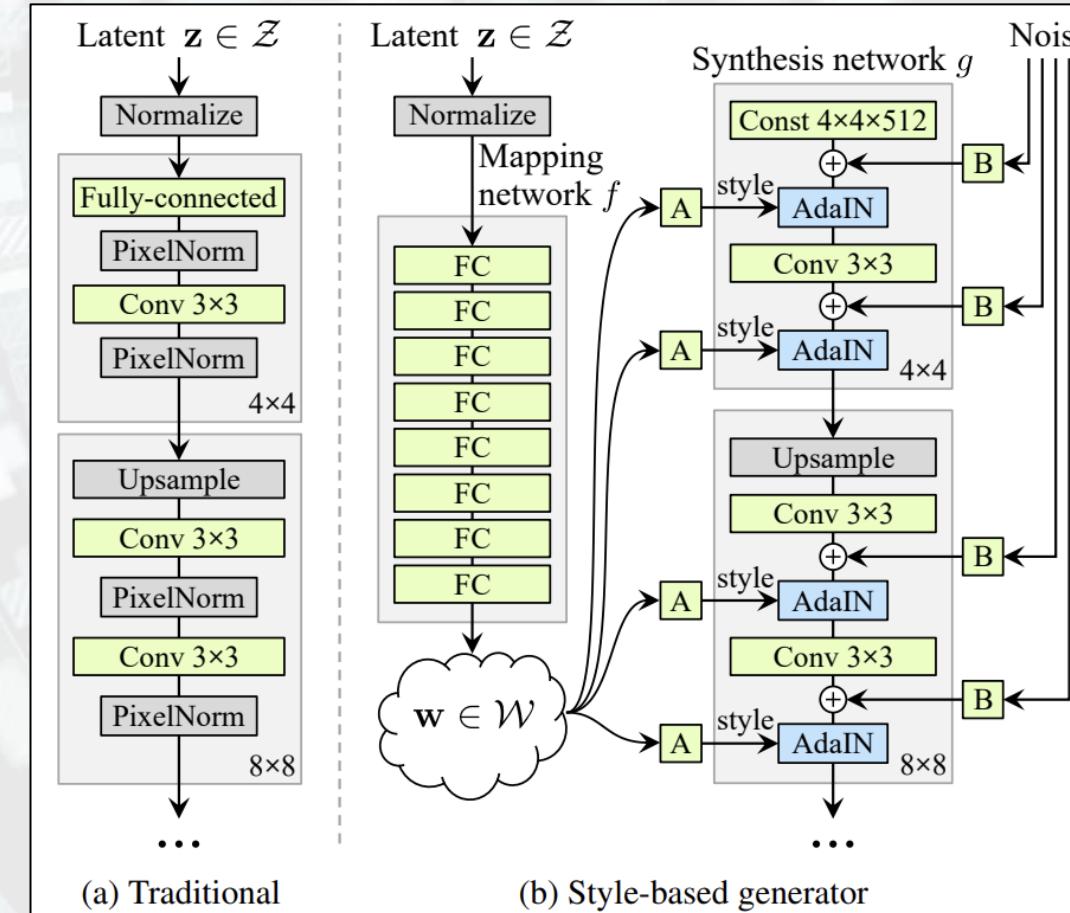


Generative Adversarial Networks – im2im [12]



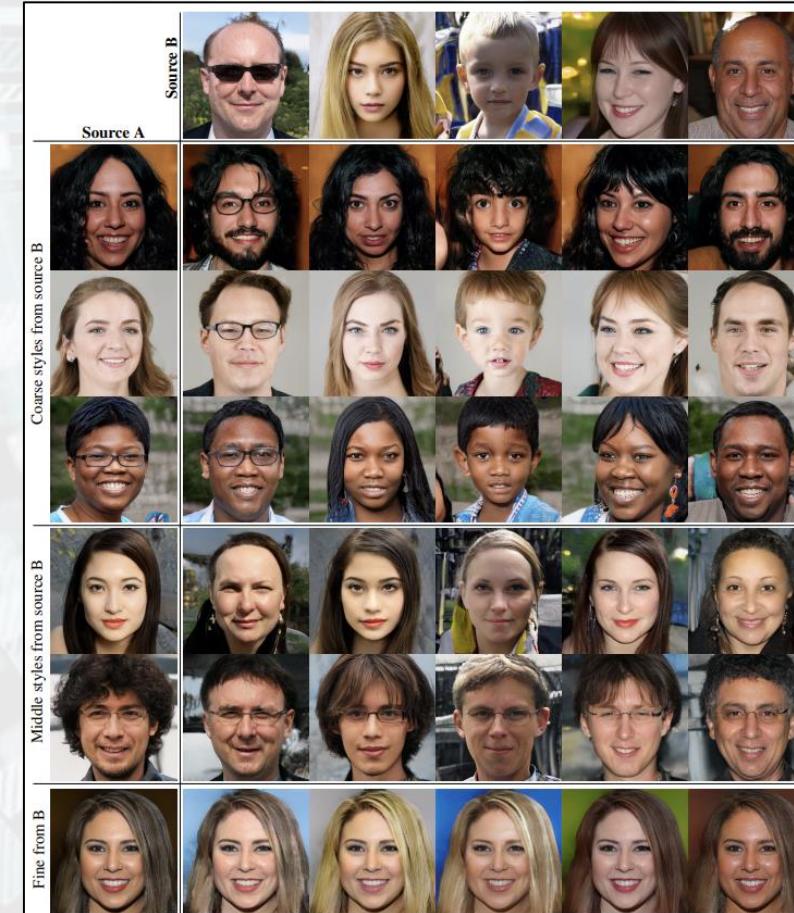
Generative Adversarial Networks – StyleGAN

[13]



Generative Adversarial Networks – StyleGAN

[13]



Sfârșit M4

Bibliografie

- [1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
- [2] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
- [3] Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., & Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. arXiv preprint arXiv:1801.07648.
- [4] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.
- [5] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. Advances in neural information processing systems, 29.
- [6] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30.
- [7] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.
- [8] Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. arXiv preprint arXiv:1605.09782.
- [9] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Bibliografie

- [11] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.
- [12] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).
- [13] Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4401-4410).