Szoftver mély neuronhálók alkalmazásához

12. előadás

Kovács Bálint, Varga Viktor
ELTE IK Mesterséges Intelligencia Tanszék

Előző órán - Felügyelt tanulás

Adott: A tanító minta (training set), input-címke párok halmaza

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n,\ y \in Y \subset \mathbb{R}^k$$

Feladat: A címke (az elvárt output) minél jobb becslése az inputból.

Azaz, keresünk olyan h_θ függvényt (hipotézisfüggvényt), melyre:

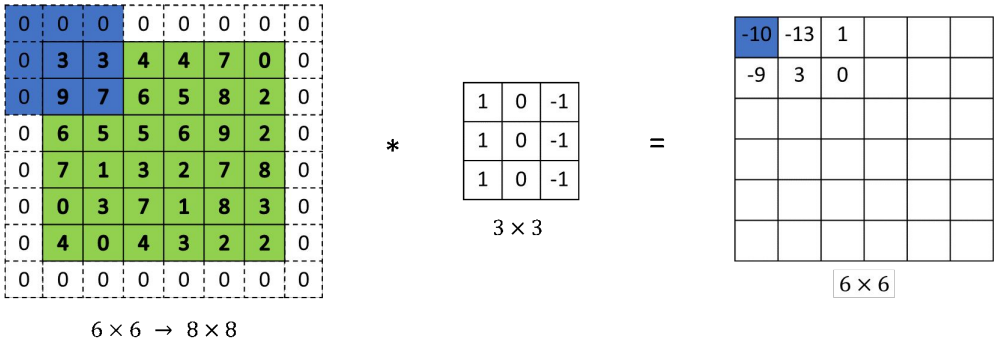
$$h_\theta(x) = \hat{y} \approx y$$

Kép forrása: http://datahacker.rs/what-is-padding-cnn/

Előző órán - A konvolúciós háló hiperparaméterei

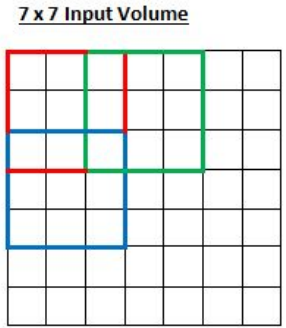
Padding:

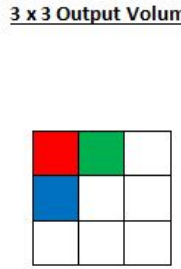
```
Conv2D(..., padding="same", ...)
```



Stride (lépésköz):

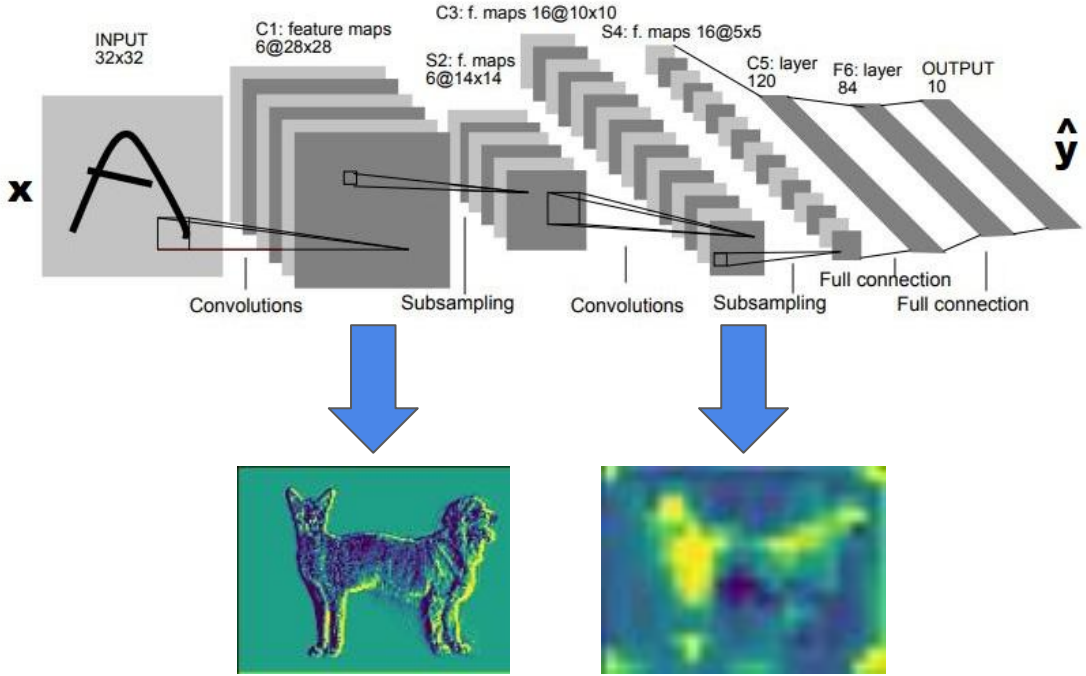
```
Conv2D(..., strides=(2, 2), ...)
MaxPooling2D(..., strides=(2, 2), ...)
```





Előző órán - Mit tanul egy konvolúciós háló?

Heatmap vizualizáció



Forrás: Yosinski et al. 2015, Deep Visualization Toolbox

Előző órán - Mit tanul egy konvolúciós háló?

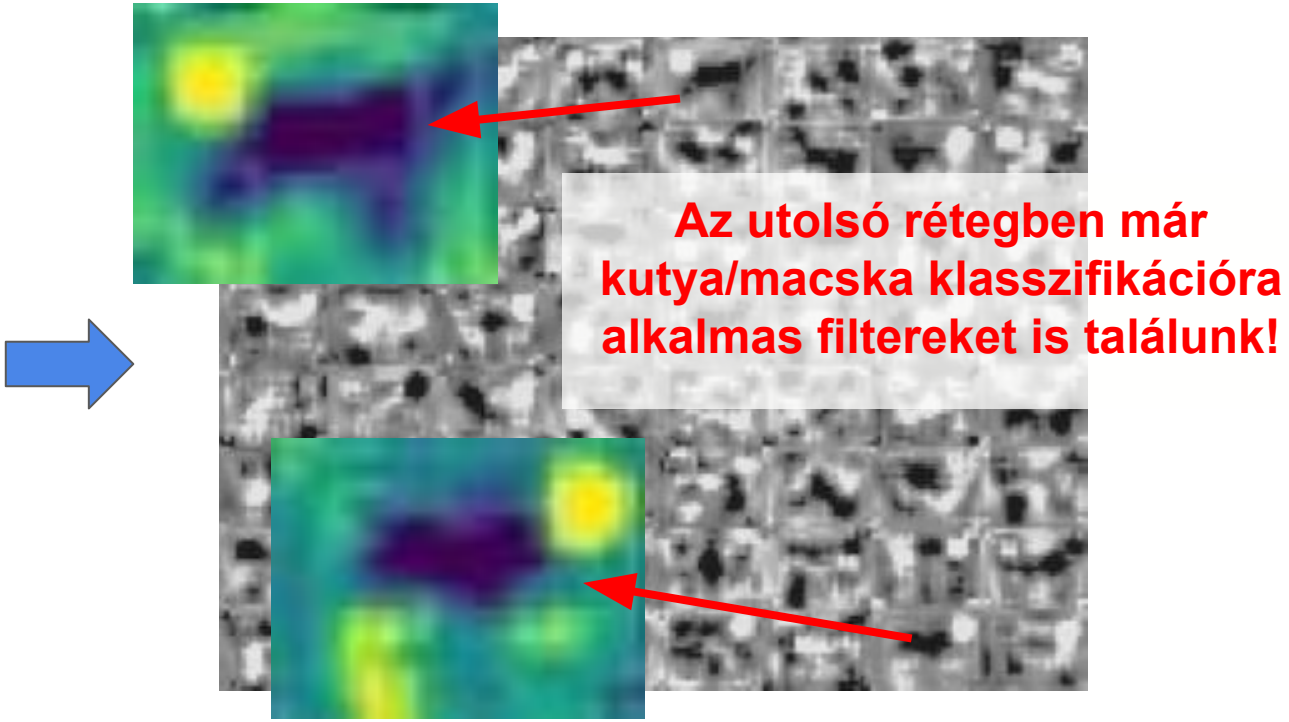
Heatmap vizualizáció

Első konv. rétegek: él-, sarok- és színátmenet-detektálás

Utolsó konv. rétegek: magasszintű objektum-detektálás

Előző órán - Mit tanul egy konvolúciós háló?

Heatmap vizualizáció - utolsó réteg



Az utolsó rétegben már kutya/macska klasszifikációra alkalmas filtereket is találunk!

Forrás: Zeiler et al. 2013, https://arxiv.org/pdf/1311.2901.pdf

Előző órán - Mit tanul egy konvolúciós háló?

Input, mely maximalizálja egy-egy heatmap pixeleit

Első konv. réteg

Középső konv. réteg

Utolsó konv. réteg

Előző órán - Transfer learning

Hogyan tanuljunk kis méretű adatbázison?

a) **Tanítsunk be egy mély neuronhálót** → túltanulás

b) **Tanítsunk be egy kisebb neuronhálót** → magas hibaarány

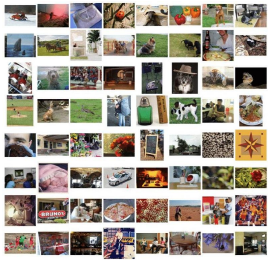
c) **Transfer learning** → enyhébb túltanulás, alacsony hibaarány
HA találunk megfelelő előtanítási feladatot/adatbázist

Előző órán - Transfer learning

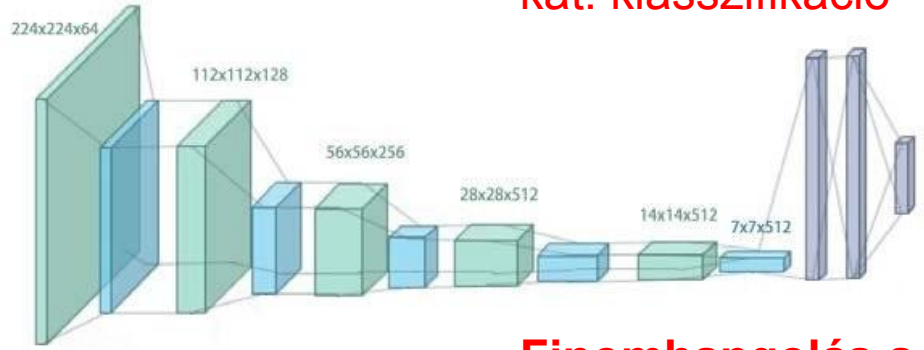
Transfer learning

Előtanítási feladat: 1000. kat. klasszifikáció

1)



nagy adatbázis



Finomhangolás a célfeladatra: bináris klasszifikáció

2)

kisebb adatbázis is elég

akár az előtanított súlyok befagyasztásával...

Előző órán - Transfer learning

Transfer learning - az előtanítást nem minden esetben kell nekünk magunknak elvégezni!

1)
```
model_pre = VGG16(input_shape=(128,128,3), include_top=False,
weights='imagenet', input_tensor=inp_t)
```

Finomhangolás a célfeladatra:
bináris klasszifikáció

2) 

kisebb adatbázis is elég

akár az előtanított súlyok befagyasztásával...

Előző órán - Transfer learning

Transfer learning a gyakorlatban - esetek

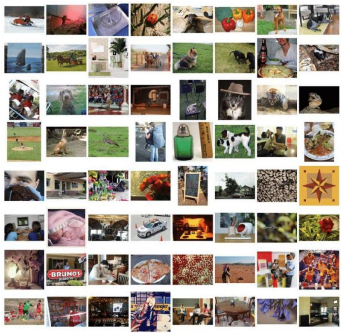
- **Minél kisebb a célfeladat adatbázisa:**
  Annál kevesebb réteget/paramétert tanítsunk a finomhangolás során!

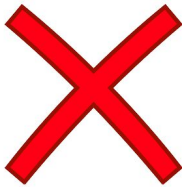
- **Minél inkább különbözik a célfeladat az előtanítási feladattól:**
  Annál kevesebb előtanított réteget tartsunk meg a finomhangoláshoz!

Előző órán - Transfer learning

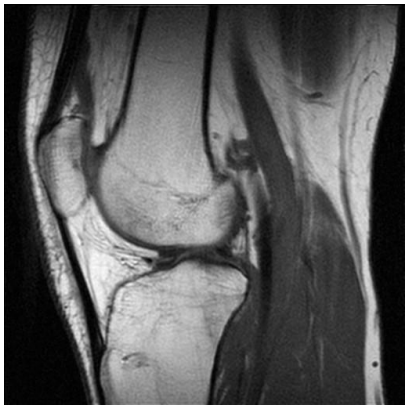
Nem minden célfeladathoz könnyű előtanító adatbázist, vagy előtanított modellt találni…

Előtanítás





Célfeladat



Előző órán - keras.models.Model, Functional API

```
inp_t = Input(shape=(n,))
t = Dense(h, activation='relu')(inp_t)
t = Dense(1, activation='sigmoid')(t)
model = Model(inputs=[inp_t], outputs=[t])
model.compile(loss='binary_crossentropy',optimizer='sgd')
```

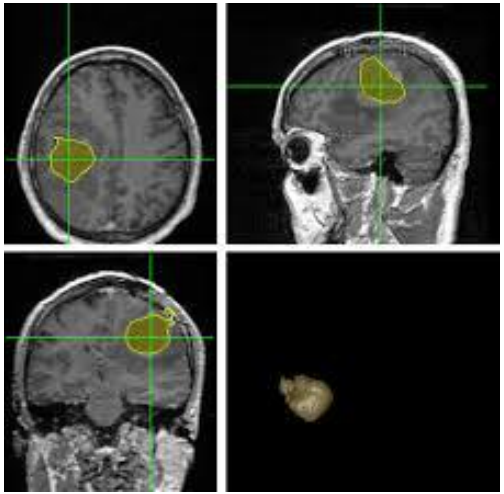
Első réteg input tenzor

Első réteg output tenzor == második réteg input tenzor

`inp_t, t`: **Tensorflow Tensor** típusú változók

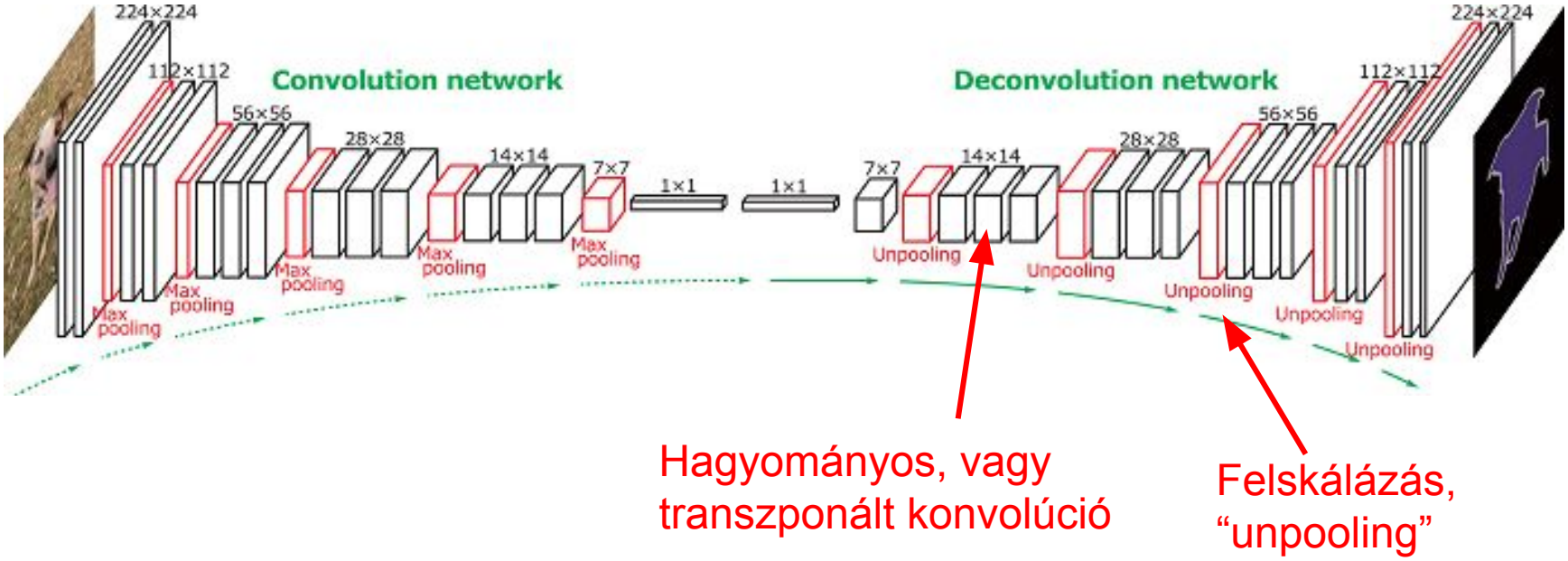
Előző órán - Képszegmentálás alkalmazások

- **ADAS** (Advanced driver-assistance systems)
- **Térképészet, földmérés**
- **Orvosi képalkotás**



Előző órán - Fully-convolutional network (FCN)

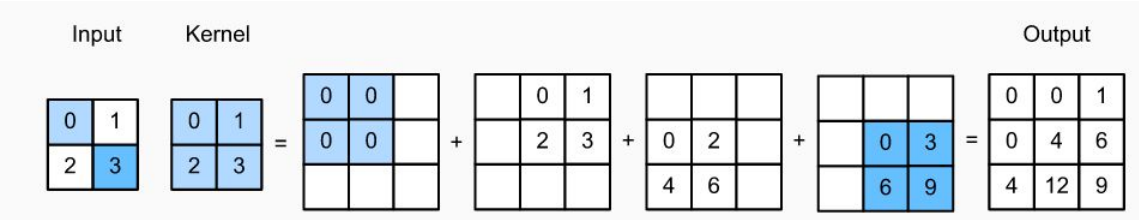
A hálónk egyáltalán ne tartalmazzon teljesen összekötött rétegeket!



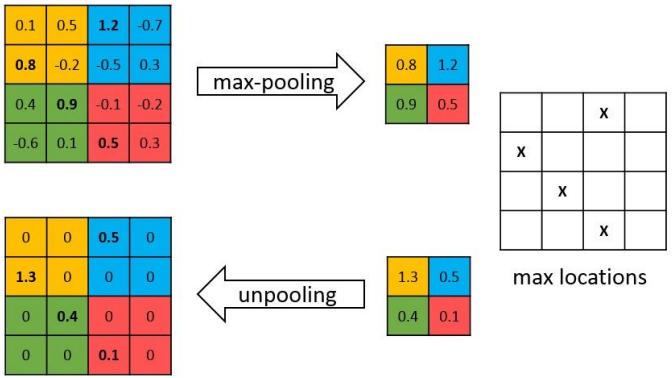
Forrás: Divyanshu Mishra @ towardsdatascience.com

Előző órán - Fully-convolutional network (FCN)

Transzponált konvolúció:



Unpooling:

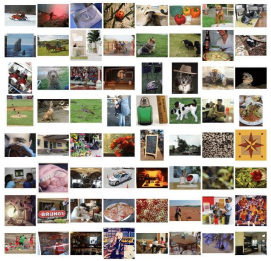


Transfer learning

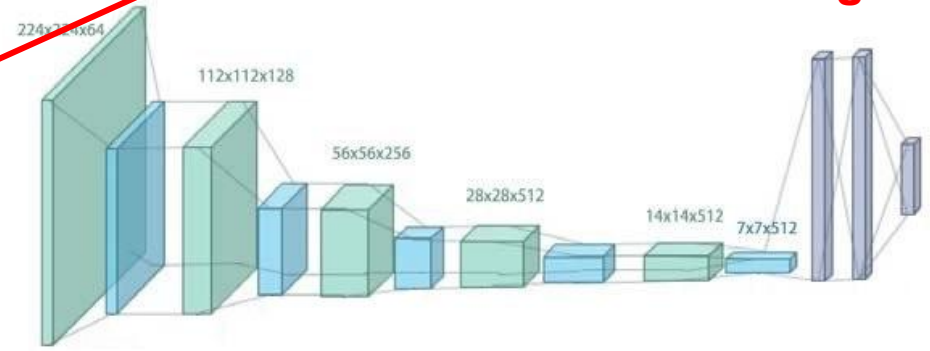
Az előtanítás sokat javíthat a célfeladaton elért eredményünkön, de **nagyméretű előtanító adatbázis meglétét igényli.**

Transfer learning

1)

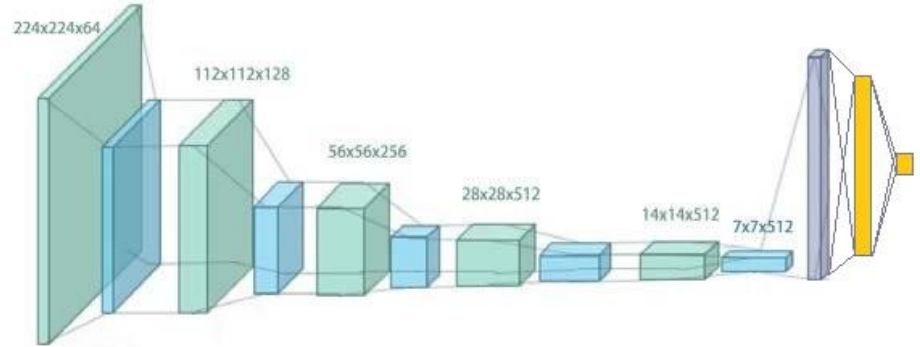


nagy CÍMKÉZETT adatbázis



2)

kisebb adatbázis is elég



Transfer learning

Transfer learning

Az előtanítás sokat javíthat a célfeladaton elért eredményünkön, de nagyméretű előtanító adatbázis meglétét igényli.

Nagyméretű címkézett adatbázisok előállítása drága lehet.

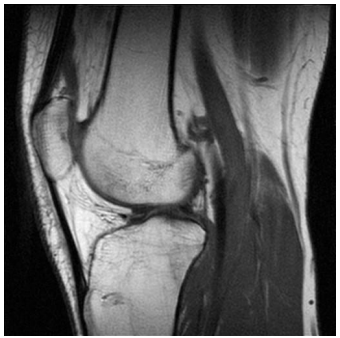
Transfer learning

Nagyméretű címkézett adatbázisok előállítása drága lehet.

Példa: Az ImageNet adatbázist valószínűleg nem tudjuk felhasználni előtanító adatbázisként térdizületek szegmentálására MRI felvételeken.

MRI felvételek címkézése szakértelmet igényel

→ rendkívül drága lenne



Felügyel**etlen** tanulás

Nagyméretű címkézett adatbázisok előállítása drága lehet.

A világban nagyon sok fajta adat nagy mennyiségben a rendelkezésünkre áll, azonban semmilyen címke nincs hozzájuk rendelve.

Vajon haszontalan a címkézetlen adat?

Felügyel**etlen** tanulás

Nagyméretű címkézett adatbázisok előállítása drága lehet.

A világban nagyon sok fajta adat nagy mennyiségben a rendelkezésünkre áll, azonban semmilyen címke nincs hozzájuk rendelve.

Vajon haszontalan a címkézetlen adat?

Nem. → **Felügyeletlen tanulás**

Eddig - Felügyelt tanulás

Adott: A tanító minta (training set), input-címke párok halmaza

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n,\ y \in Y \subset \mathbb{R}^k$$

Feladat: A címke (az elvárt output) minél jobb becslése az inputból.

Azaz, keresünk olyan h_θ függvényt (hipotézisfüggvényt), melyre:

$$h_\theta(x) = \hat{y} \approx y$$

Felügyel**etlen** tanulás (unsupervised learning)

- **Tanító minta** (training set):

$$\{x^{(0)}, x^{(1)}, \ \dots \ , x^{(m)}\}$$

$$x \in X \subset \mathbb{R}^n$$

Feladat: ???

Felügyel**etlen** tanulás (unsupervised learning)

- **Tanító minta** (training set):

$$\{x^{(0)}, x^{(1)}, \ldots, x^{(m)}\}$$

$$x \in X \subset \mathbb{R}^n$$

j: mintaelem index

Továbbra is: $x_i^{(j)}$

i: feature/változó index

Feladat: ???

Felügyel**etlen** tanulás (unsupervised learning)

Feladat: Első körben, ahogy felügyelt előtanításnál tettük, próbáljunk olyan súlyokat/filtereket tanulni, amik hasznosak lehetnek más speciális célfeladatokon is!

Inputunk van: x

Címkénk viszont nincs hozzá...

Felügyel**etlen** tanulás (unsupervised learning)

Feladat: Első körben, ahogy felügyelt előtanításnál tettük, próbáljunk olyan súlyokat/filtereket tanulni, amik hasznosak lehetnek más speciális célfeladatokon is!

Inputunk van: x

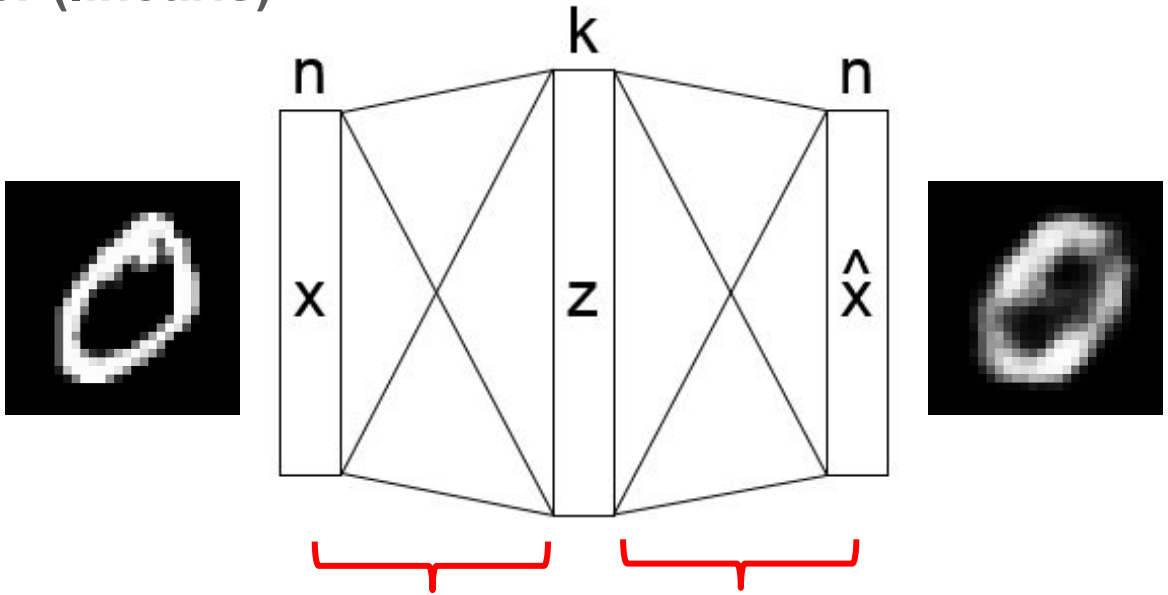
Címkénk viszont nincs hozzá…

Ötlet: Használjuk x-et címkének!

Autoencoder

$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

Autoencoder (lineáris)

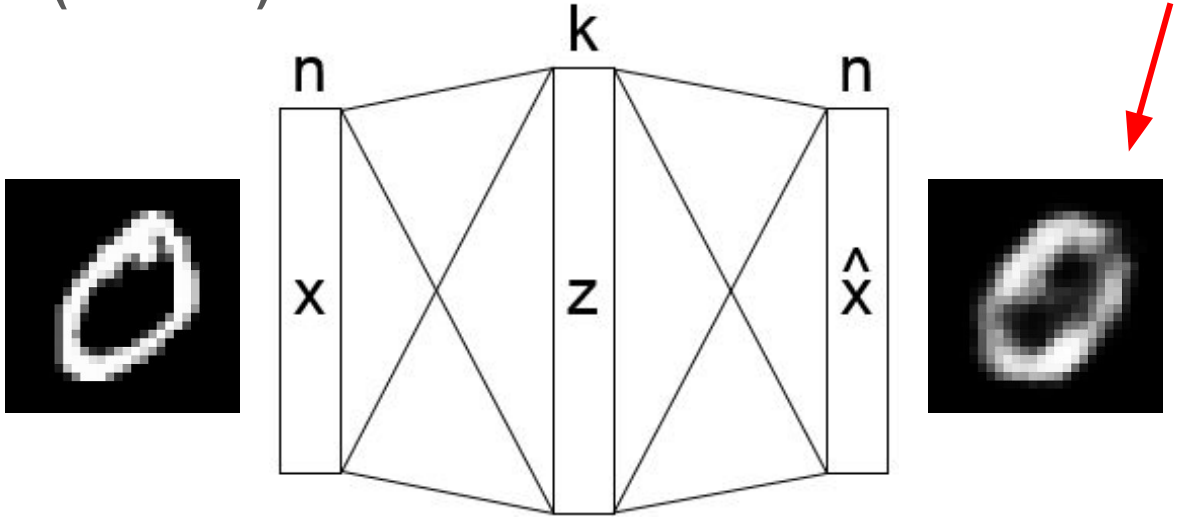


Autoencoder

$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

Autoencoder (lineáris)

Tanuljuk meg **az inputot előállítani** (rekonstruálni) **az outputon.**



Kétrétegű neuronháló, egyelőre aktivációs függvények nélkül.

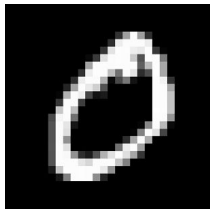
$$W_{enc} \in \mathbb{R}^{k \times n} \qquad W_{dec} \in \mathbb{R}^{n \times k}$$

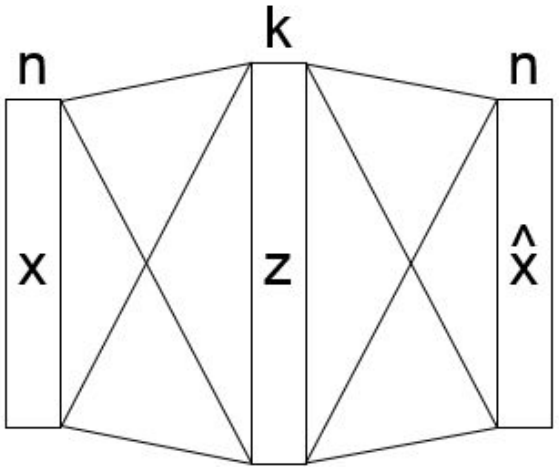
$$b_{enc} \in \mathbb{R}^{k} \qquad b_{dec} \in \mathbb{R}^{n}$$

Autoencoder

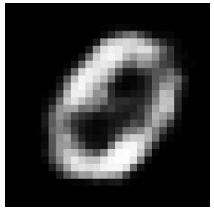
$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

Autoencoder (lineáris)

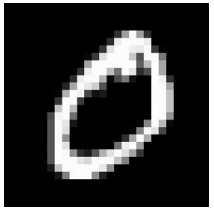




Tanuljuk meg **az inputot előállítani** (rekonstruálni) **az outputon.**



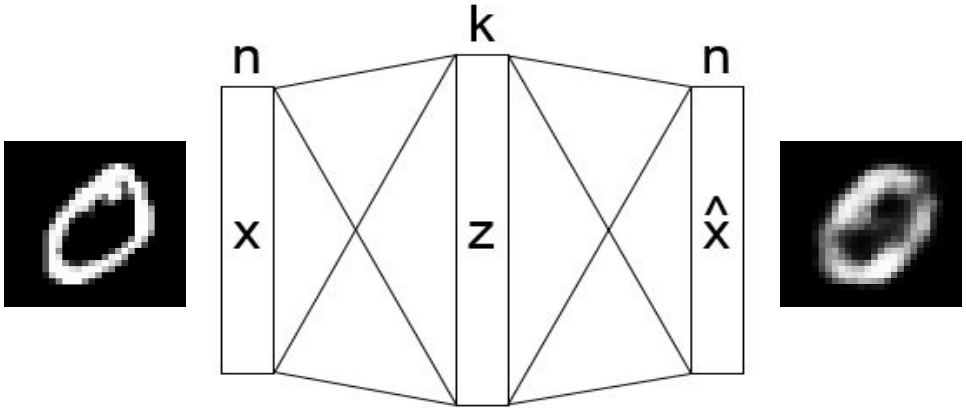
≈



$$J(\Theta) = \frac{1}{n}||\hat{x} - x||_2^2$$

A költségfv. egyszerű MSE

Autoencoder



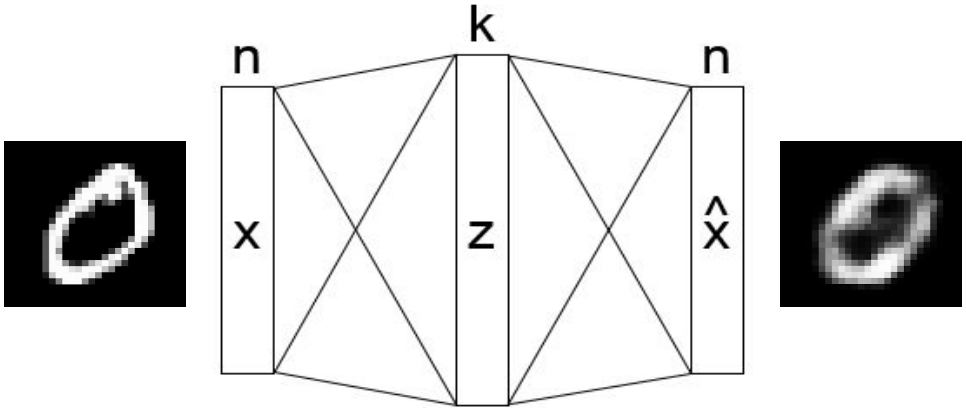
Probléma:

$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

átrendezhető:

$$h(x) = \hat{x} = \underbrace{(W_{dec}W_{enc})}_{\in \mathbb{R}^{n \times n}}x + \underbrace{(W_{dec}b_{enc} + b_{dec})}_{\in \mathbb{R}^{n}}$$

Autoencoder



Probléma:

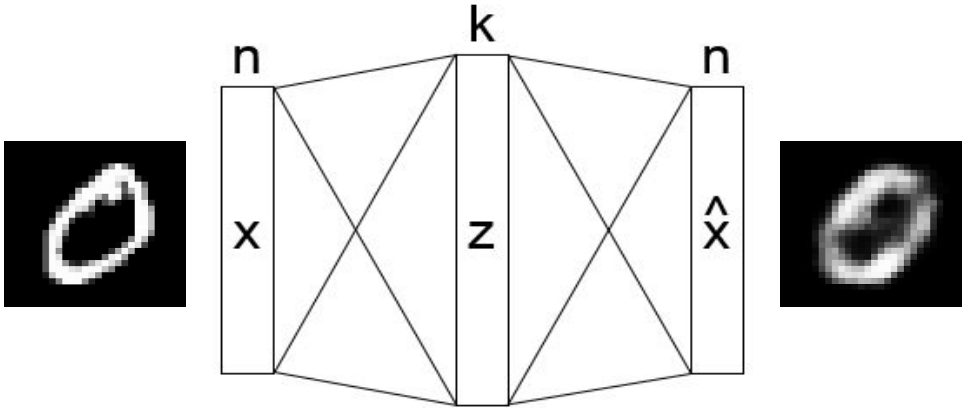
$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

átrendezhető:

$$h(x) = \hat{x} = \underbrace{(W_{dec}W_{enc})}_{\in \mathbb{R}^{n \times n}}x + \underbrace{(W_{dec}b_{enc} + b_{dec})}_{\in \mathbb{R}^{n}}$$

Két lineáris réteg kompozíciója egyetlen réteggel is leírható!

Autoencoder



Probléma:

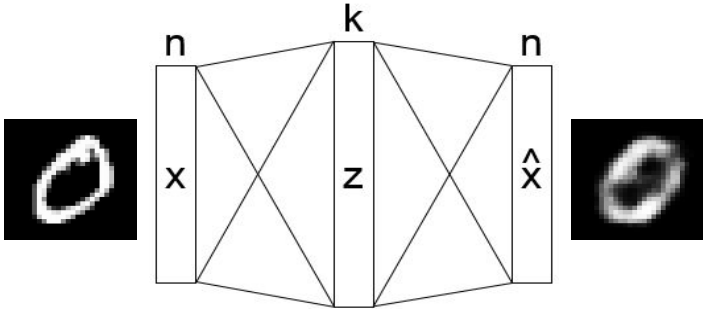
$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

átrendezhető:

$$h(x) = \hat{x} = \underbrace{(W_{dec}W_{enc})}_{\in \mathbb{R}^{n\times n}}x + \underbrace{(W_{dec}b_{enc} + b_{dec})}_{\in \mathbb{R}^{n}}$$

Sőt, semmi nem akadályozza meg a hálót, hogy az egységmátrixot tanulja meg!

Autoencoder



Probléma:

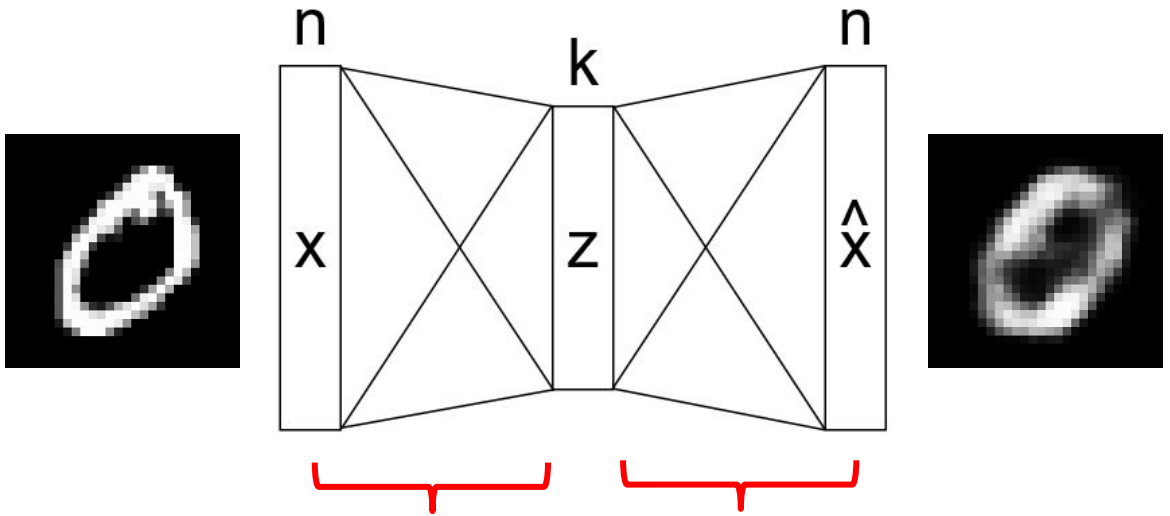
Az autoencoder célja az input rekonstruálása az outputon. Ebben a formában, semmi nem akadályozza meg a hálót, hogy triviális súlyokat (egységmátrixot) tanuljon.

→ **a célját tökéletesen eléri** (0 költség), **de semmi haszna számunkra**

Autoencoder

$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

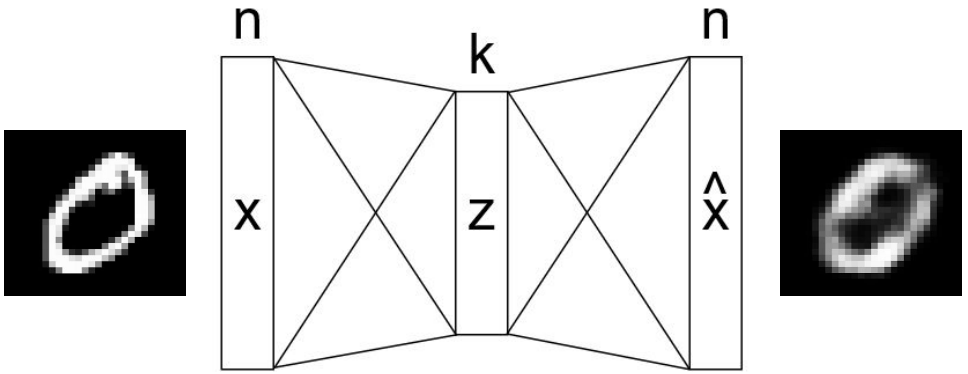
Egy apró módosítás: legyen a rejtett réteg kisebb, mint az input/output!



$W_{enc} \in \mathbb{R}^{k \times n}$ $W_{dec} \in \mathbb{R}^{n \times k}$

$b_{enc} \in \mathbb{R}^{k}$ $b_{dec} \in \mathbb{R}^{n}$

Autoencoder



$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

$W_{enc} \in \mathbb{R}^{k \times n}$

$b_{enc} \in \mathbb{R}^{k}$

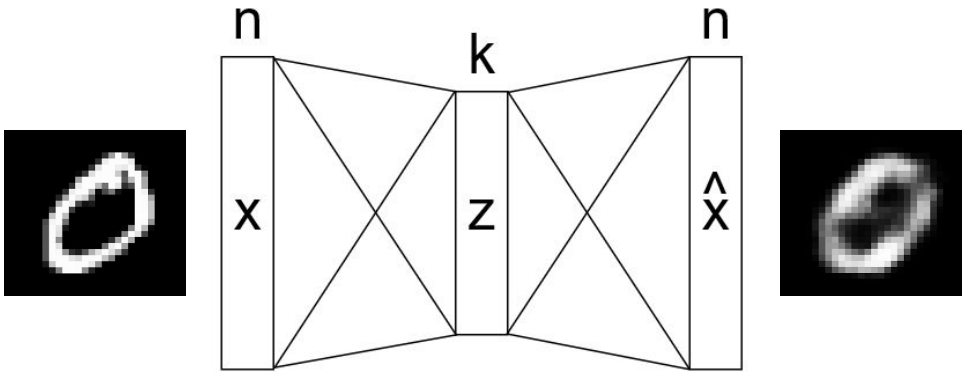
továbbra is átrendezhető:

$$h(x) = \hat{x} = \underbrace{(W_{dec}W_{enc})}_{\in \mathbb{R}^{n \times n}}x + \underbrace{(W_{dec}b_{enc} + b_{dec})}_{\in \mathbb{R}^{n}}$$

$W_{dec} \in \mathbb{R}^{n \times k}$

$b_{dec} \in \mathbb{R}^{n}$

Autoencoder



$$h(x) = \hat{x} = W_{dec}(W_{enc}x + b_{enc}) + b_{dec}$$

$W_{enc} \in \mathbb{R}^{k \times n}$

$b_{enc} \in \mathbb{R}^{k}$

$W_{dec} \in \mathbb{R}^{n \times k}$

$b_{dec} \in \mathbb{R}^{n}$

továbbra is átrendezhető:

$$h(x) = \hat{x} = \underbrace{(W_{dec}W_{enc})}_{\in \mathbb{R}^{n \times n}}x + \underbrace{(W_{dec}b_{enc} + b_{dec})}_{\in \mathbb{R}^{n}}$$

Azonban, mivel n > k, a háló nem tud identitást tanulni: az egységmátrix rangja **n** lenne, de **W_dec · W_enc** rangja legfeljebb **min(n, k) = k** lehet!

Autoencoder

Autoncoder megszorítás

Az autoencoder betanításakor, különböző megszorításokat kell alkalmaznunk, hogy ne legyen triviális a megoldás.

Az egyik ilyen, az inputnál kisebb rejtett rétegek használata: **Alulhatározott** (undercomplete) **autoencoder**

Autoencoder - tömörítés

Alulhatározott (undercomplete) **autoencoder**

Az alulhatározott autoencoder egy tömörített (alulhatározott) reprezentációját tanulja meg az inputnak.

Enkóder: Az autoencoder első része, előállítja a rejtett (tömör) reprezentációt.

Dekóder: Az autoencoder hátsó része, rekonstruálja a rejtett (tömör) reprezentációból az inputot.

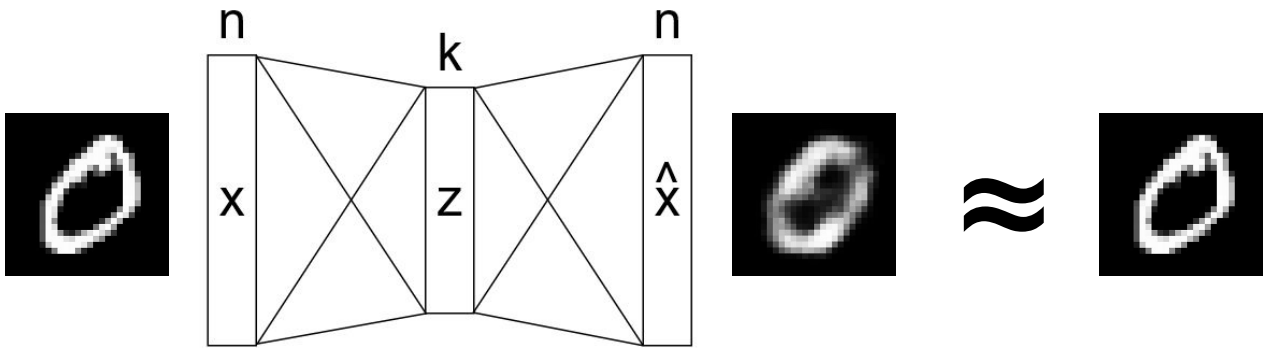
→ **Veszteséges tömörítés!**

Autoencoder - tömörítés

Alulhatározott (undercomplete) **autoencoder**

Veszteséges tömörítés:

- **Cél:** Minél kisebb hibával rekonstruáljuk az inputot az outputon!



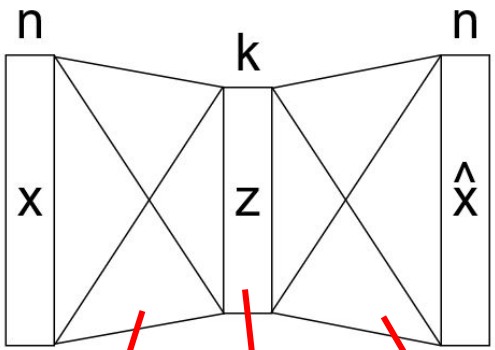
Autoencoder - tömörítés

Veszteséges tömörítés

Példa: 64 x 64-es szürekárnyalatos arcképek tömörítése

Autoencoder - tömörítés

Veszteséges tömörítés - példa



Analógia:

x: tömörítetlen adat

Enkóder: zip.exe

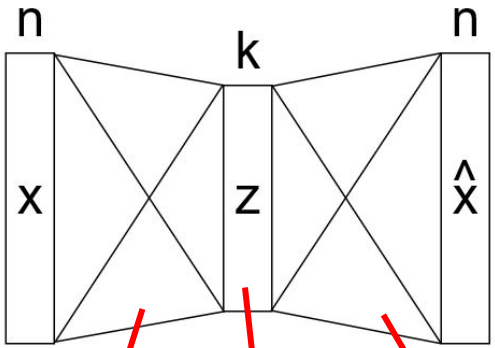
Dekóder: unzip.exe

x^: helyreállított adat

Autoencoder - tömörítés

Veszteséges tömörítés - példa

Ne feledjük: a zip formátum veszteségmentes tömörítési algoritmusokat támogat, a zip csak az analógia kedvéért szerepel itt. Az autoencoder inkább a JPEG, MP3, stb. formátumokkal lenne összehasonlítható.



Analógia:

x: tömörítetlen adat

Enkóder: zip.exe

Dekóder: unzip.exe

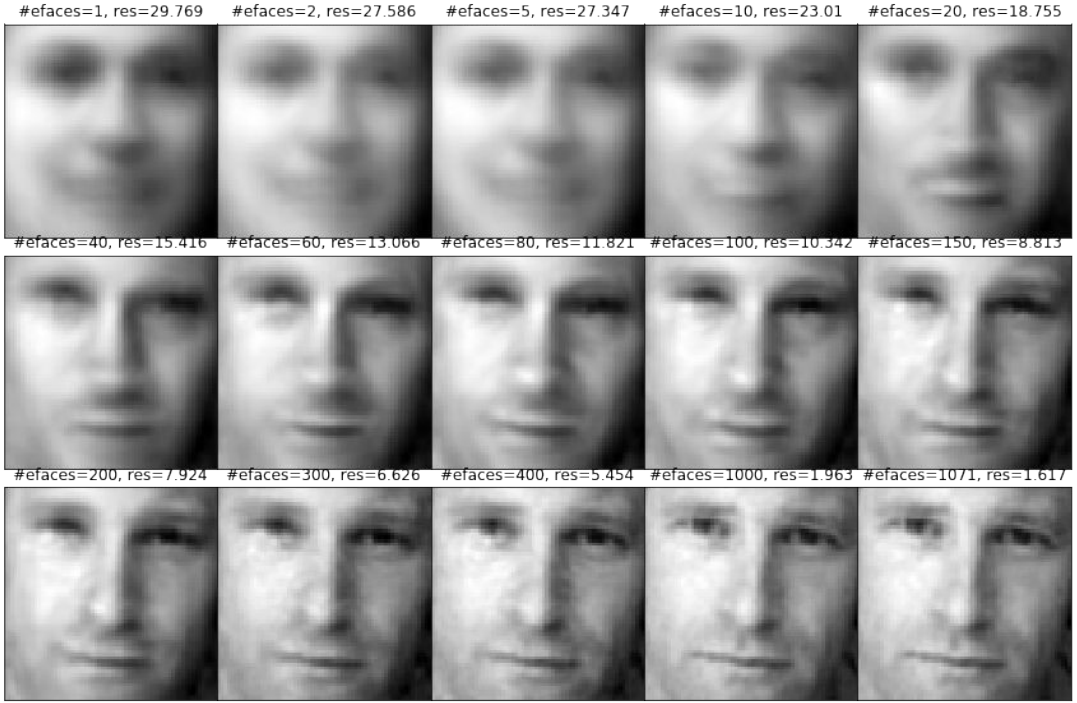
x^: helyreállított adat

Autoencoder - tömörítés

x^: output (tömör reprezentációból rekonstruált input)

Veszteséges tömörítés - példa

x: eredeti input



Autoencoder - tömörítés

x^: output (tömör reprezentációból rekonstruált input)

Veszteséges tömörítés - példa

|z| = 1

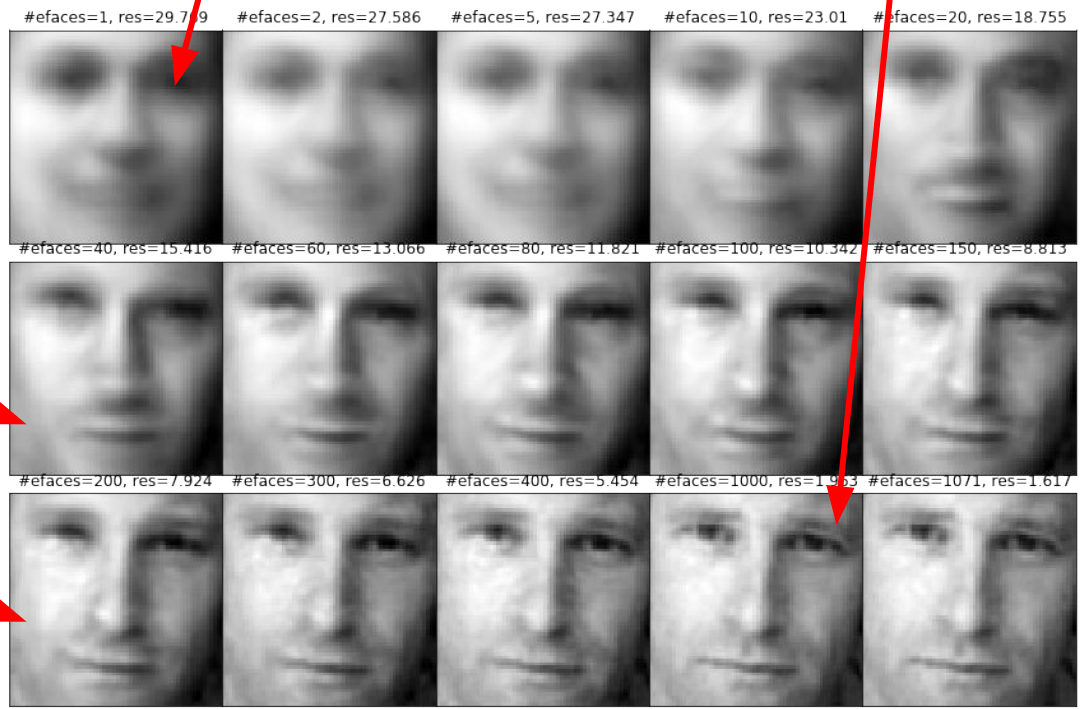
|z| = 1000

|x| = 4096

x: eredeti input

|z| = 40

|z| = 200



Autoencoder - tömörítés

Veszteséges tömörítés - példa

A JPEG és MP3 veszteséges tömörítési algoritmusok fényképekre és zenére lettek (kézzel) optimalizálva.

A JPEG tömörítési szabvány a magas frekvenciájú komponensek erős tömörítésével képes helyet megtakarítani. Fényképeken tipikusan kevés éles színátmenet található, ezért ott jó képminőség várható. Azonban szövegről készült képnél pont emiatt, nem ideális a JPEG tömörítés használata.

Forrás: Sandipan Dey, sandipanweb.wordpress.com/

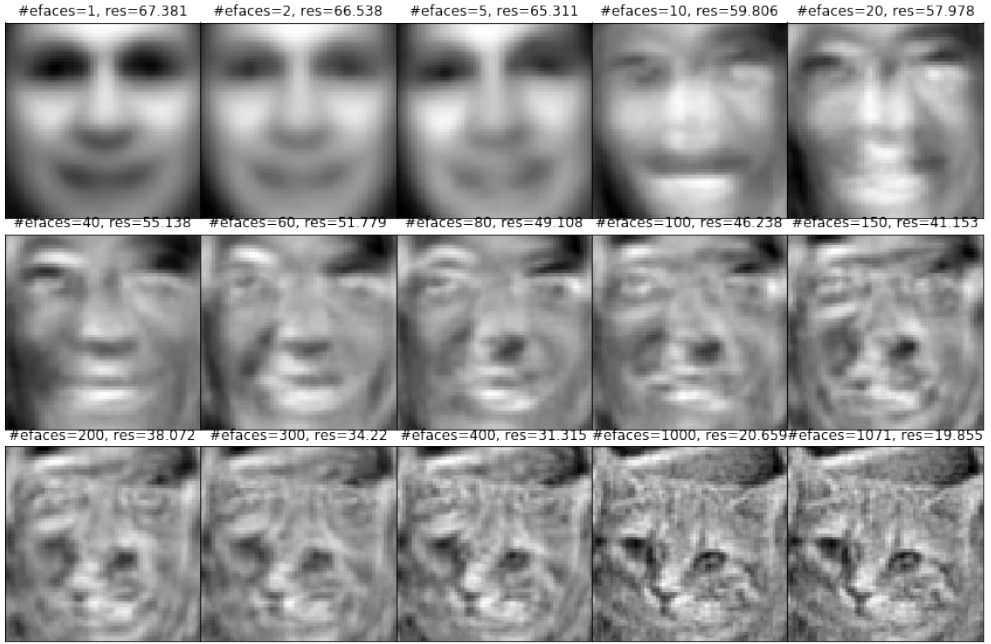
Autoencoder - tömörítés

x^: output (tömör reprezentációból rekonstruált input)
A háló emberi arcokon tanult be...

Veszteséges tömörítés - példa

A gépi tanulással tanult tömörítés hasonlóan, csak olyan adatra fog jól működni, amilyet betanításkor látott.

x: eredeti input



Forrás: Sandipan Dey, sandipanweb.wordpress.com/

Autoencoder - tömörítés

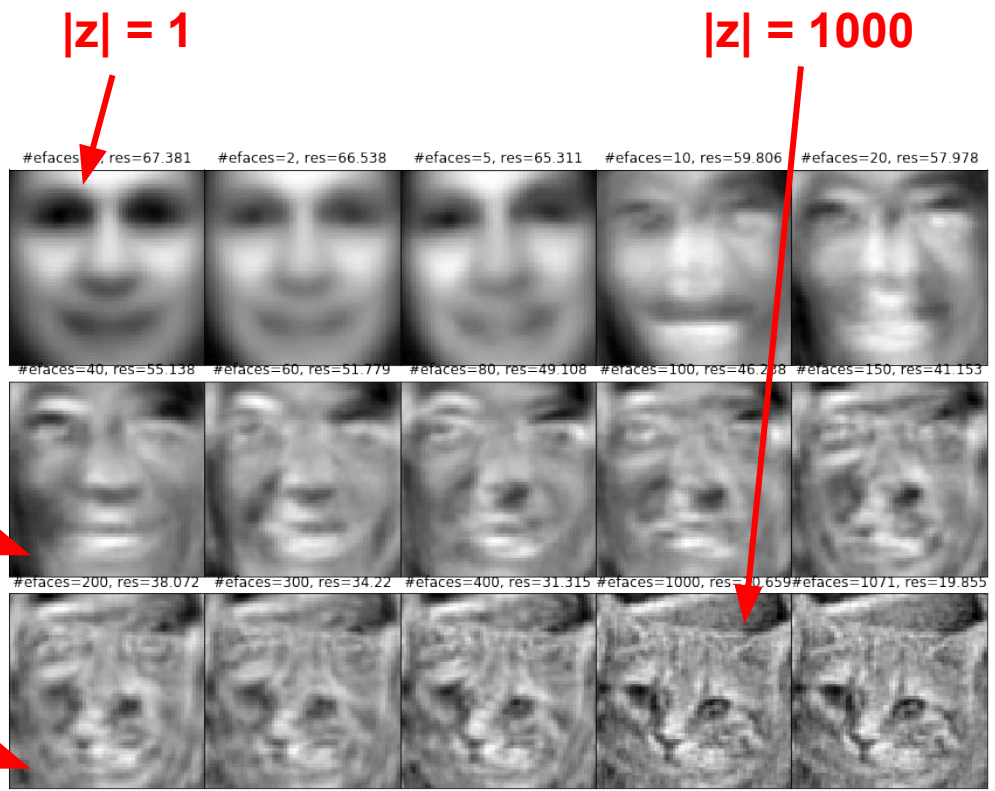
x^: output (tömör reprezentációból rekonstruált input)

A háló emberi arcokon tanult be...

Veszteséges tömörítés - példa

A gépi tanulással tanult tömörítés hasonlóan, csak olyan adatra fog jól működni, amilyet betanításkor látott.

x: eredeti input

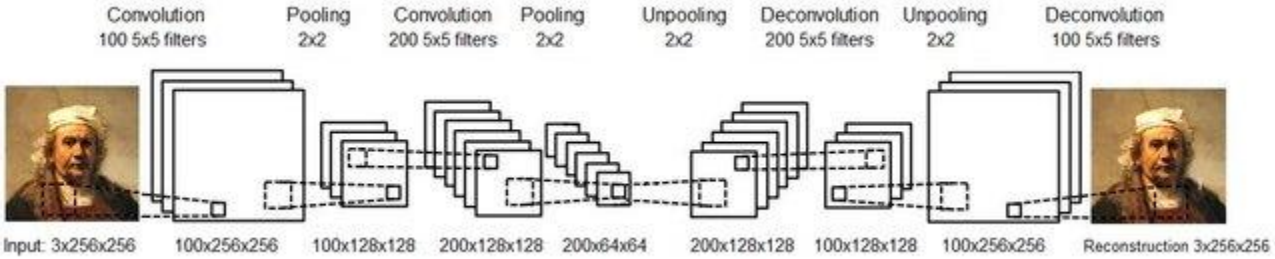


Forrás: David et al., 2016

Autoencoder

Hasonlóan a felügyelt tanulásban alkalmazott neuronhálókhoz, az autoencoder:

- Tartalmazhat aktivációs függvényeket
- Több réteget
- Konvolúciós / pooling rétegeket



Transfer learning

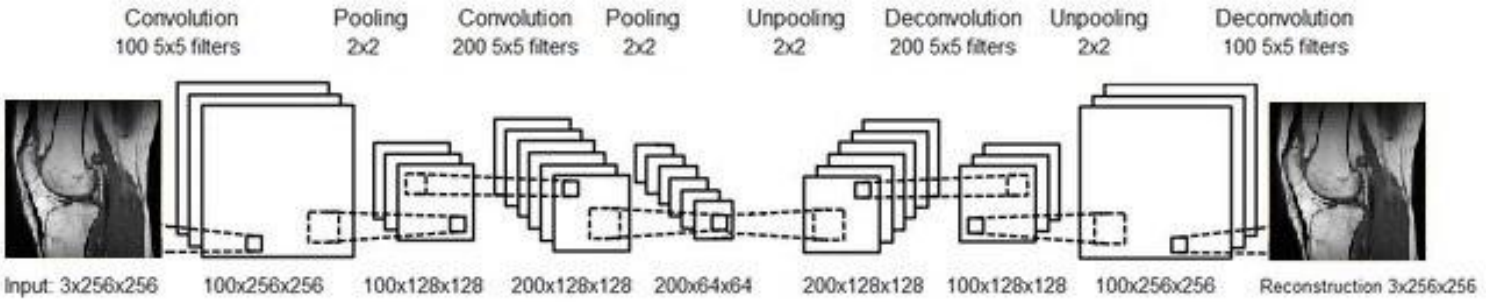
Emlékeztető: Nagyméretű címkézett adatbázisok előállítása drága lehet

Megoldás: ???

Transfer learning

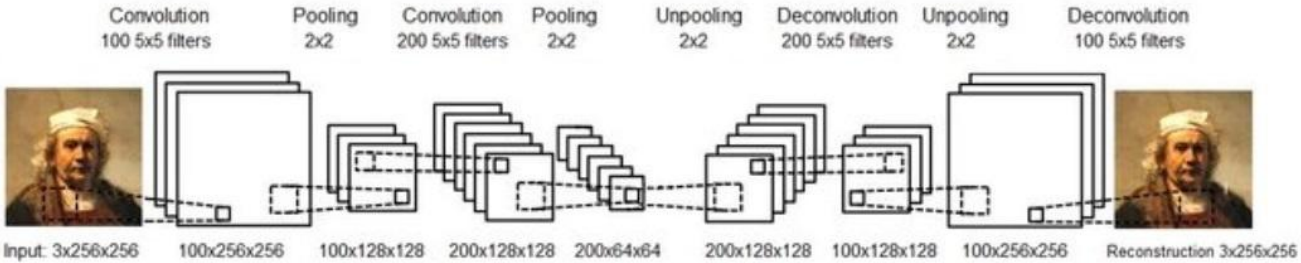
Emlékeztető: Nagyméretű címkézett adatbázisok előállítása drága lehet

Megoldás: Megpróbálkozhatunk a felügyeletlen előtanítással!

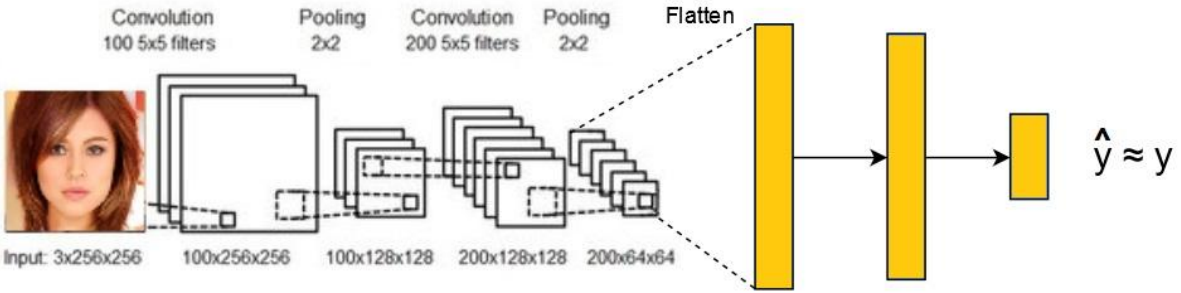


Transfer learning - felügyeletlen

1) **Felügyeletlen előtanítás autoencoderrel** - nagy (akár címkézetlen) adatbázison.



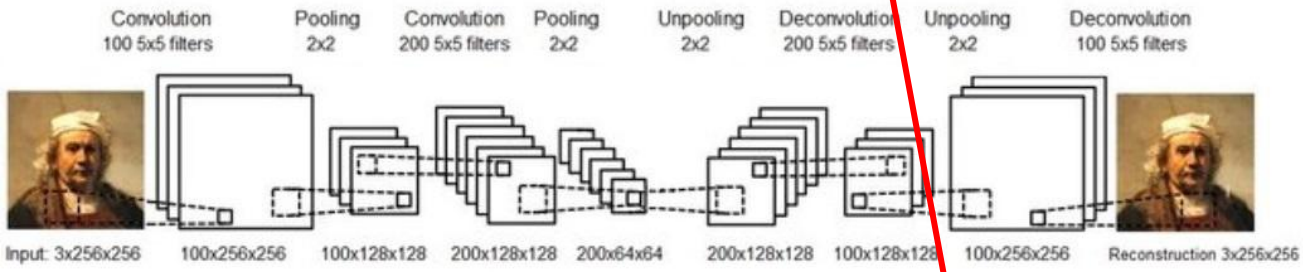
2) **Felügyelt finomhangolás a célfeladaton -** kisebb adatbázis is elegendő lehet.



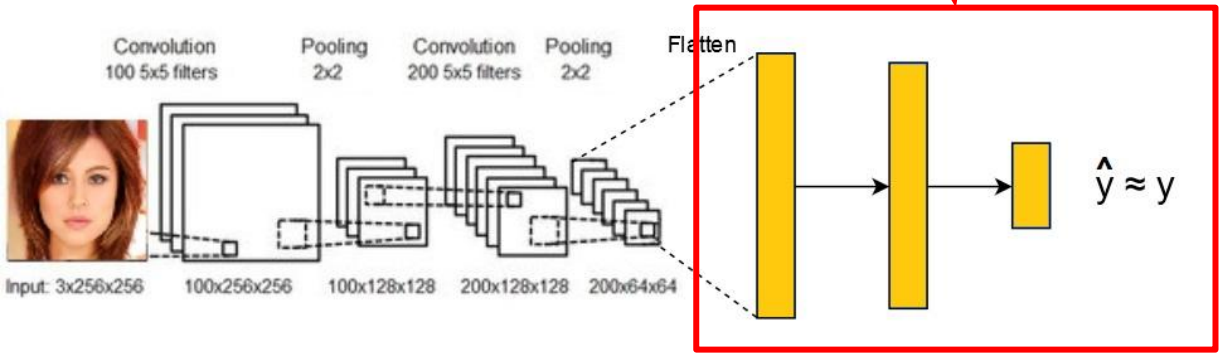
Transfer learning - felügyeletlen

A dekóder rész cseréje a célfeladathoz szükséges rétegekkel

1) **Felügyeletlen előtanítás autoencoderrel** - nagy (akár címkézetlen) adatbázison.



2) **Felügyelt finomhangolás a célfeladaton -** kisebb adatbázis is elegendő lehet.



akár az előtanított súlyok befagyasztásával...

Transfer learning - felügyeletlen

Transfer learning autoencoder-rel

Az előtanító adatbázistól és a célfeladattól függően, sokszor kevésbé hasznos, mint a felügyelt előtanítás.

Transfer learning - felügyeletlen

Transfer learning autoencoder-rel

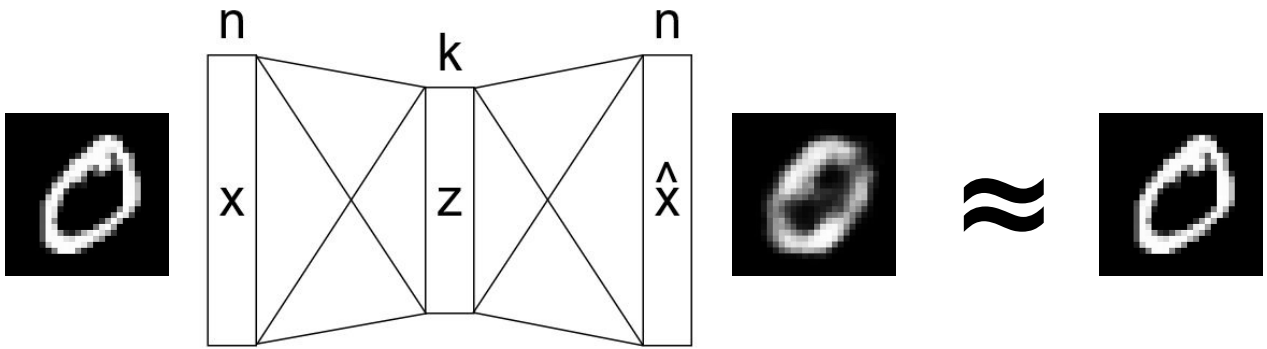
Az előtanító adatbázistól és a célfeladattól függően, sokszor kevésbé hasznos, mint a felügyelt előtanítás.

Nem garantált, hogy az optimális rekonstrukcióhoz tanult súlyok / filterek egy klasszifikációs feladaton is igazán hasznosak. Ettől függetlenül, sokszor jobb, mint a semmi...

Autoencoder

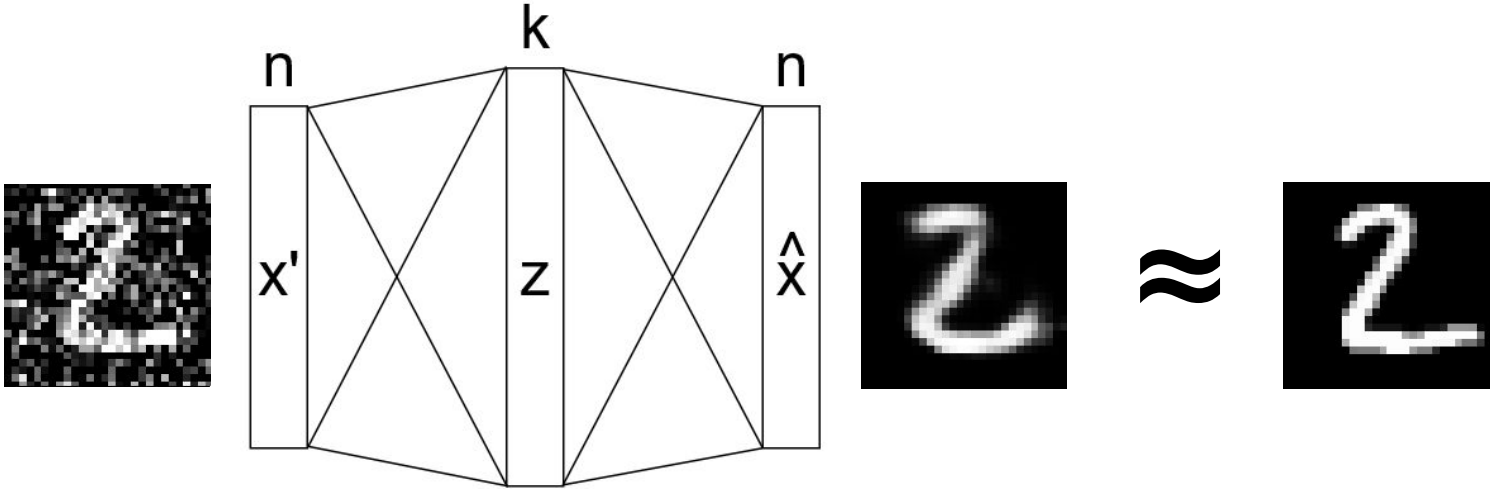
Eddig: Alulhatározott (undercomplete) autoencoder

Ha nem feltétlenül tömörítést szeretnénk tanulni, csak hasznos súlyokat / filtereket (feature extrakció)**, akkor más autoencoder megszorítások is szóba jöhetnek...**



Autoencoder

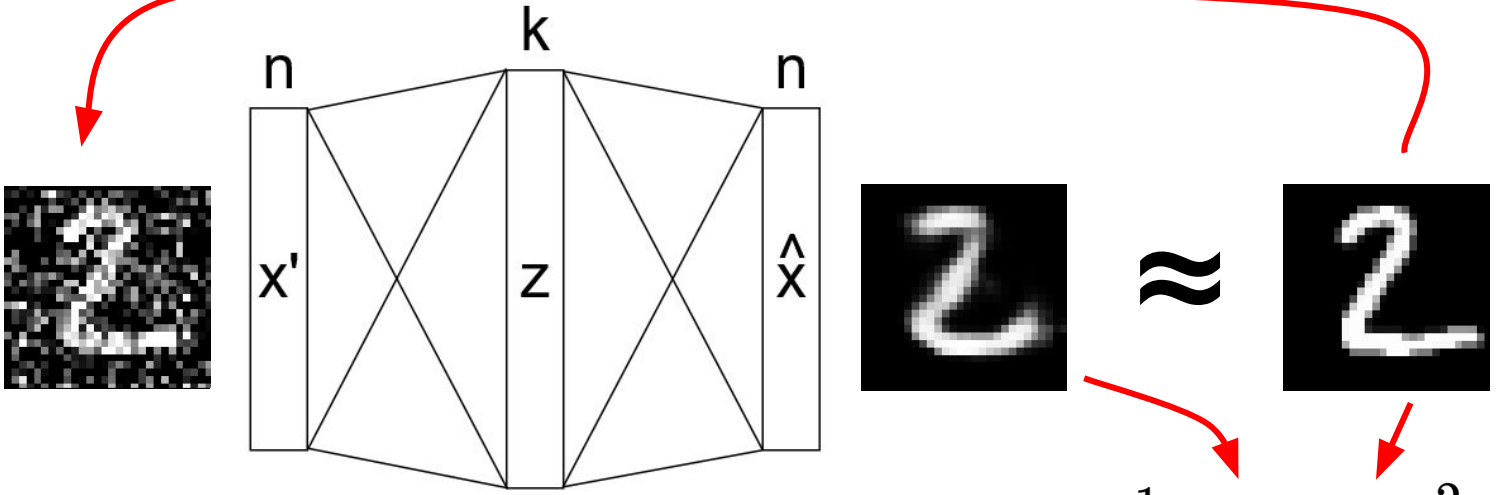
Denoising autoencoder



Autoencoder

Denoising autoencoder

Inputként **x** zajosított változatát (**x'**) adjuk be, de az eredeti **x**-et tanuljuk becsülni



$$J(\Theta) = \frac{1}{n}||\hat{x} - x||_2^2$$

A költségfv. egyszerű MSE

Autoencoder

Denoising autoencoder

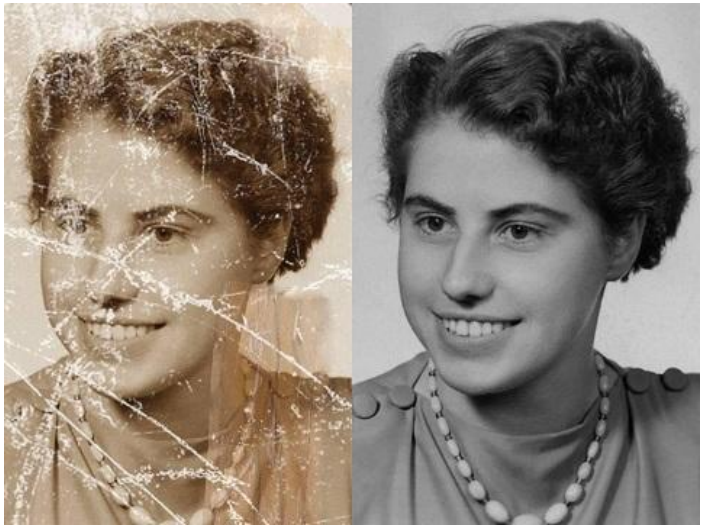
Az input rekonstrukciója helyett, a zajosított input eredeti, zajtalan változatát próbáljuk előállítani.

Mivel nem identitást tanulunk, hanem zajtalanítást, aminek már nincs triviális megoldása, **a rejtett reprezentáció lehet akár nagyobb is, mint az input.**

Forrás: https://github.com/WenchenLi/PhotoRestoration, https://medium.com/@sorenlind

Denoising autoencoder - alkalmazások

Fénykép minőségének javítása (denoising)



x’ x^

x’ x^

Denoising autoencoder - alkalmazások

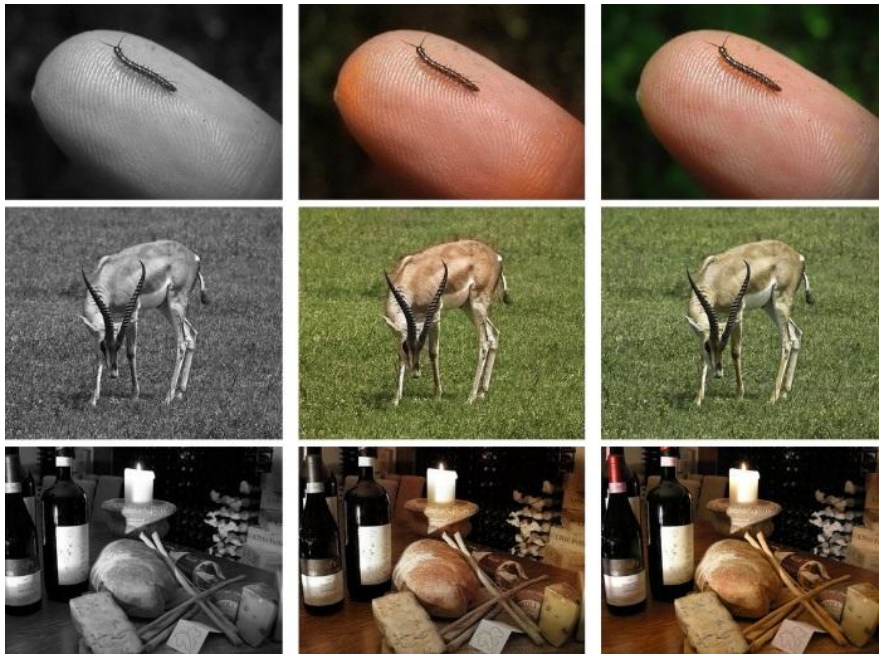
Hiányzó részletek kitöltése (inpainting)

x’ x^ x

Forrás: Larsson et al., 2017

Denoising autoencoder - alkalmazások

Újraszínezés (colorization)

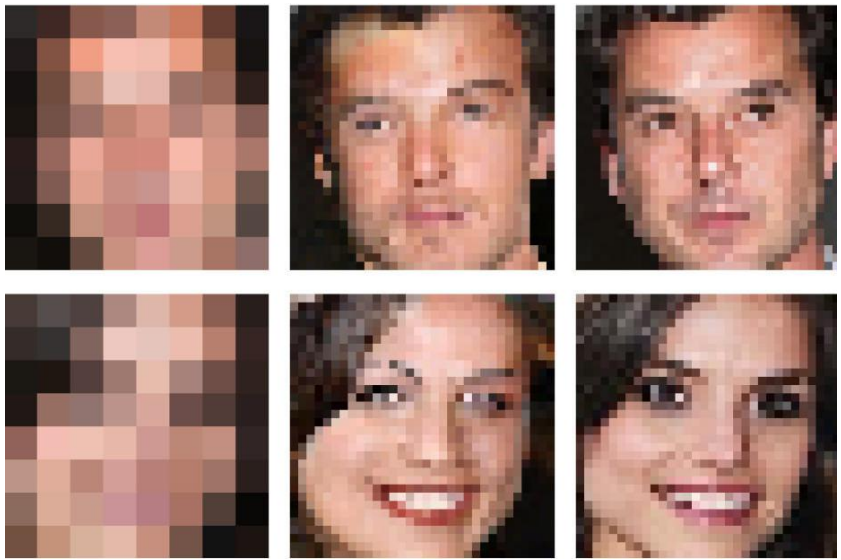


x’ x^ x

Forrás: Dahl et al., 2017

Denoising autoencoder - alkalmazások

Intelligens felskálázás (super-resolution)



Denoising autoencoder - alkalmazások

“De-oldify”

https://www.youtube.com/watch?v=-omKOpkpEm0

A félév összefoglalása - elmélet

- A felügyelt tanulás feladatai: regresszió, klasszifikáció
- Hipotézisfüggvény, paraméterek, költségfüggvény, gradiens módszer
- Lineáris és logisztikus regresszió (mesterséges neuron)
- Teljesen összekötött neuronrétegek, MLP neuronháló architektúra
- Konvolúciós és pooling rétegek, konvolúciós neuronháló
- Túltanulás (overfitting) és kezelése
- Transfer learning, előtanítás, finomhangolás
- Egyszerű képszegmentálás, Fully-Convolutional Network (FCN)
- Felügyeletlen tanulás, autoencoder
- Tömörítés, transfer learning és feature extrakció autoencoder-rel, denoising

A félév összefoglalása - gyakorlat

- Vektorizált programozás (array programming), Numpy
- Keras alapismeretek, Sequential vs. Functional API

Folytatás

- **BSc:**
  - **Mély neuronhálók algoritmusai és fajtái** - ősszel
    (Analízis 2. nem előfeltétel)
- **MSc:**
  - **Proginf, AI szakirány** (AI specialization)
  - **Autonómrendszer-informatikus** (CS for Autonomous Systems) **szak**

Névtelen kritika: Neptun OMHV

Olvasnivaló, programoznivaló

Francois Chollet:

Deep Learning with Python