

Szoftver mély neuronhálók alkalmazásához

9. előadás

Kovács Bálint, Varga Viktor
ELTE IK Mesterséges Intelligencia Tanszék

Előző órán - Felügyelt tanulás

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

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

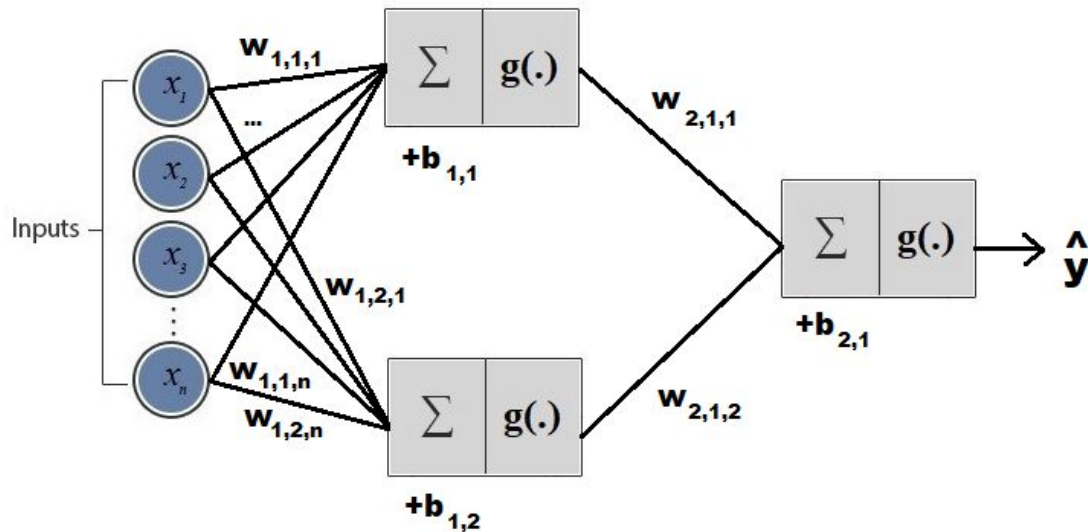
$$x \in X \subset \mathbb{R}^n, y \in Y \subset \mathbb{R}^k$$

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

Azaz, keresünk olyan h_{θ} függvényt (hipotézisfüggvényt), melyre:

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

Előző órán - Multilayer Perceptron (MLP)



$$W_1 \in \mathbb{R}^{2 \times n}$$
$$b_1 \in \mathbb{R}^2$$

$$W_2 \in \mathbb{R}^{1 \times 2}$$
$$b_2 \in \mathbb{R}^1$$

Új jelölés: Theta a súlymátrixok és bias vektorok halmaza

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

Előző órán - Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2 \underbrace{g_1(W_1 x + b_1)}_{\text{Első réteg outputja}} + b_2) = \hat{y} \approx y$$

Költségfüggvények:

- **klasszifikáció:** logistic loss (skalár címke esetén)
- **regresszió:** MSE

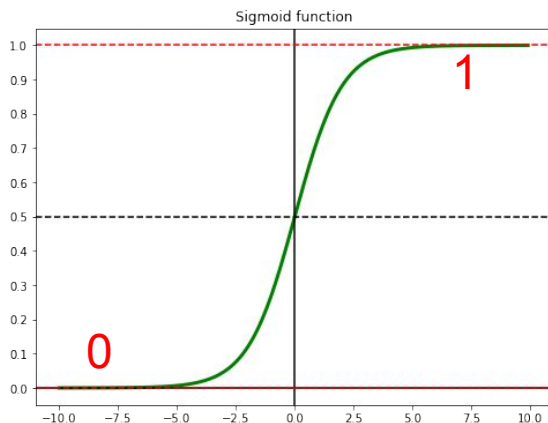
Aktivációs fv. (nemlinearitás) regresszió esetén is kell, azonban ilyenkor az utolsót szokás elhagyni!

(keras.activations.*)

Előző órán - Aktivációs függvények

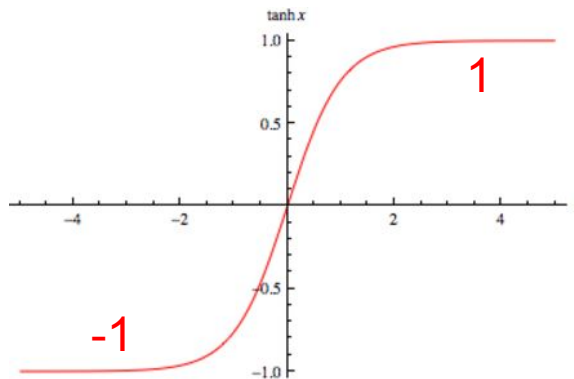
Gyakran használt aktivációs függvények:

sigmoid



$$g(z) = \frac{1}{1+e^{-z}}$$

tanh



$$g(z) = \tanh(z) = \frac{e^{2x}-1}{e^{2x}+1}$$

ReLU

(Rectified Linear Unit)



**Gyorsan számolható és
szinte mindig jól működik**


$$g(z) = \text{ReLU}(z) = \max(0, z)$$

Előző órán - MLP betanítása

Gradiensmódszert fogunk
továbbra is használni...

Szerencsére nem kell kiszámolnunk
kézzel a gradienseket. Ezt a
Tensorflow **automatikus deriválási**
algorithmusa elvégzi helyettünk...

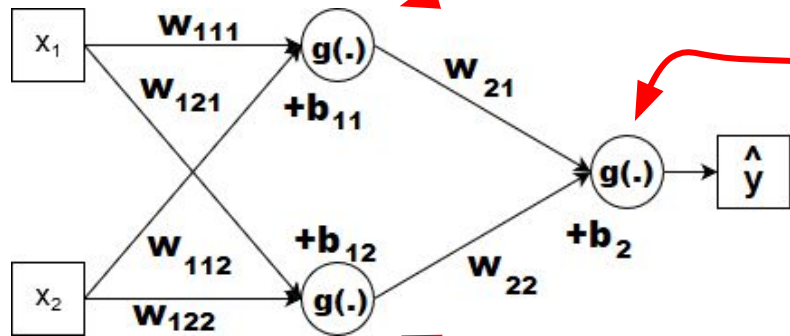
```
repeat until convergence {  
  for  $\forall \theta \in \Theta$  {  
     $grad_{\theta} = \frac{\partial}{\partial \theta} J(\Theta)$   
  }  
  for  $\forall \theta \in \Theta$  {  
     $\theta = \theta - \alpha grad_{\theta}$   
  }  
}
```



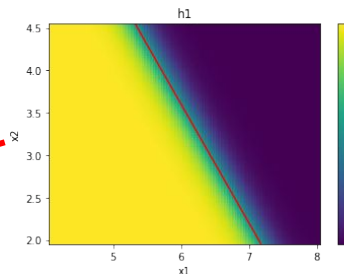
Minden egyes paraméter szerint
deriválnunk kell a költségfüggvényt.

Előző órán - Neuronháló kifejezőereje

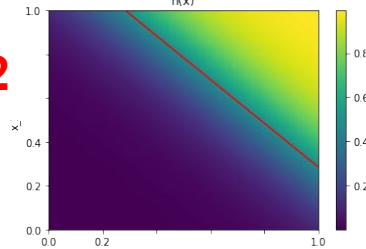
$$w_{111} = -7, w_{121} = -5, b_{11} = 60$$



$$w_{112} = 4, w_{122} = -5, b_{12} = -5$$

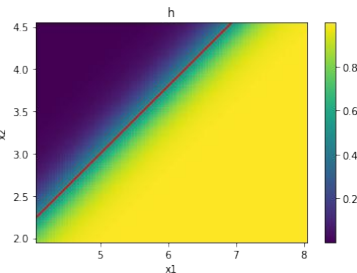


h_2

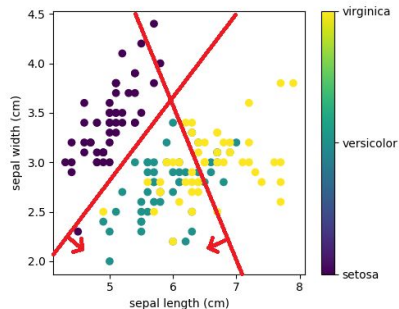
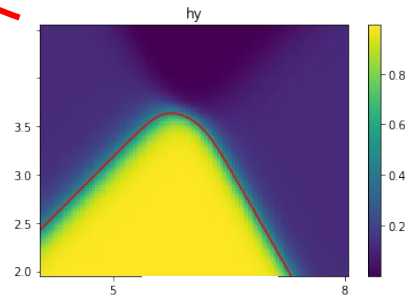


h_1

x_2

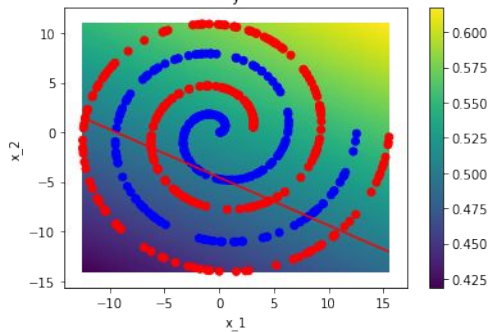
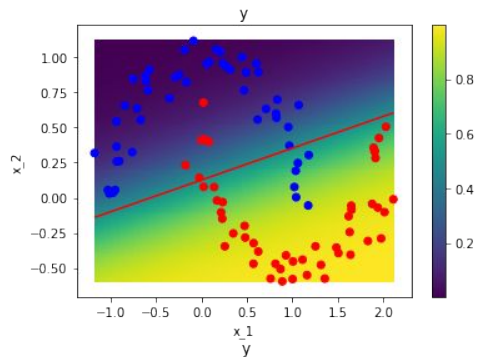


x_1

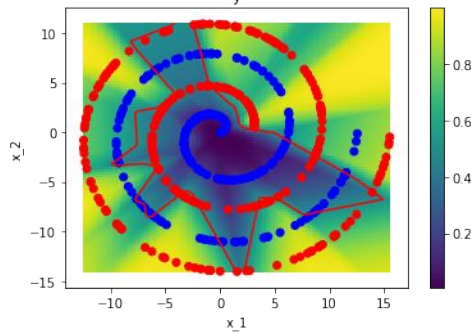
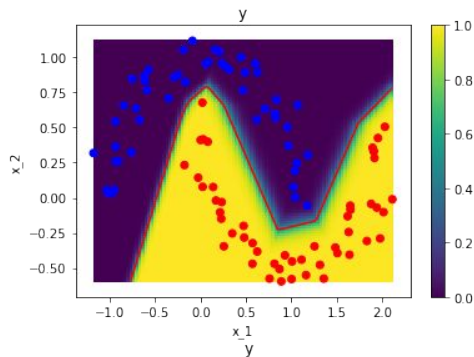


Előző órán - Neuronháló kifejezőereje

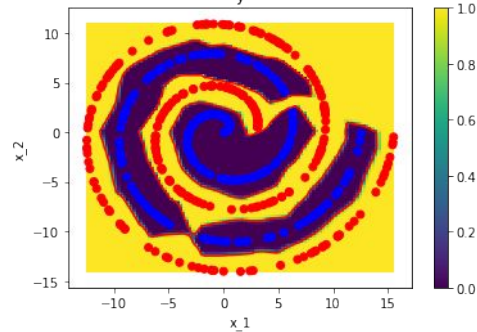
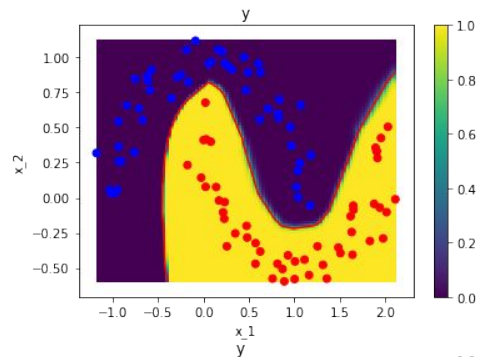
log.reg.



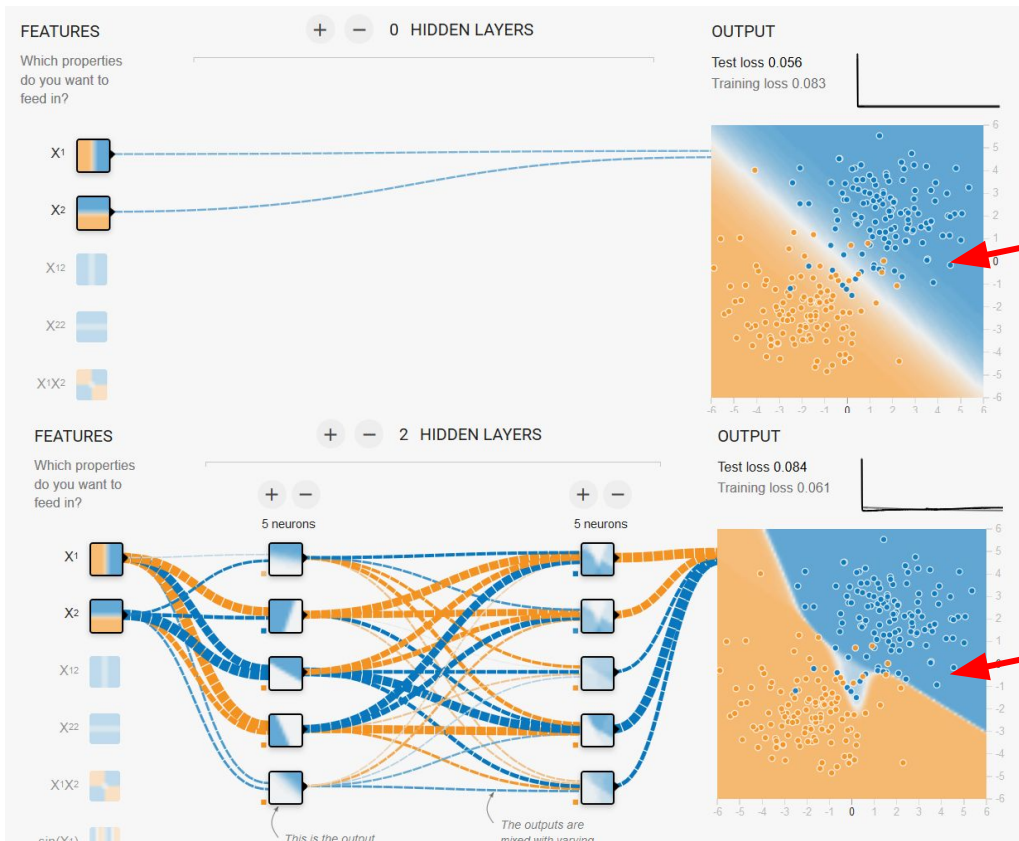
2 réteg, 20+1 neuron



4 réteg, 20+20+20+1
neuron



Előző órán - Alul- és túltanulás neuronhálók esetén



Egyszerű feladatra egyszerű modell jó

Bonyolult modell könnyebben túltanul

Előző órán - Vektor alakú címkét igénylő probléma

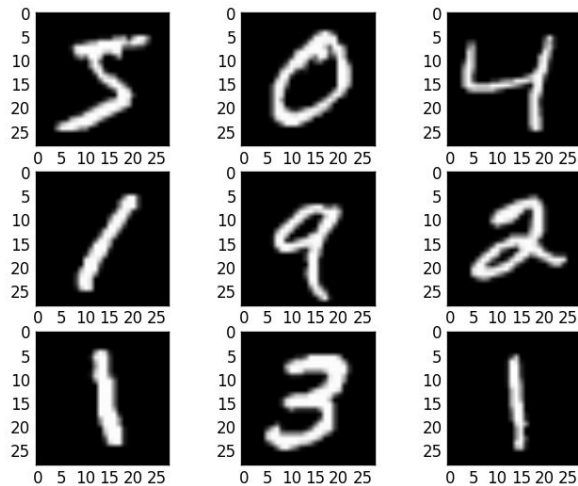
MNIST adatbázis

- Kézzel írt számjegyek
- 28×28 -as méretű képek
- 10 kategória (számjegyek: 0 .. 9)
- 60 ezer tanítópélda,
10 ezer teszt példa

10 kategóriába klasszifikálunk

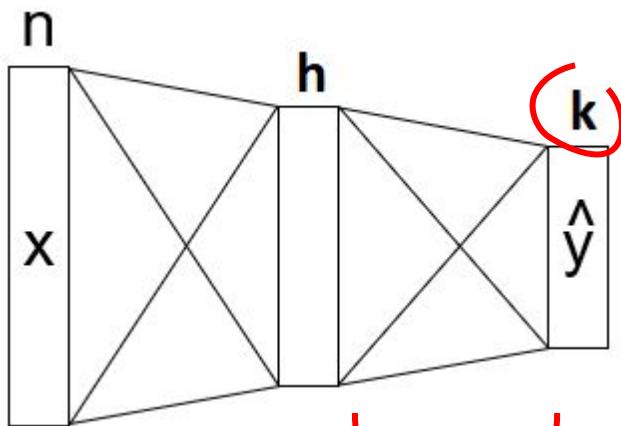
→ **10 elemű vektorokat becsülünk**

784 input változó: Minden pixel fényereje egy változó



Előző órán - Vektor alakú címke becslése MLP-vel

$$h(x) = g_2(W_2 g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$



Legyen y is vektor, hasonlóan x -hez.

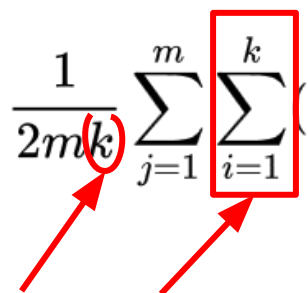
$$\begin{aligned} W_1 &\in \mathbb{R}^{h \times n} \\ b_1 &\in \mathbb{R}^h \end{aligned} \quad \begin{aligned} W_2 &\in \mathbb{R}^{k \times h} \\ b_2 &\in \mathbb{R}^k \end{aligned}$$

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

Előző órán - Vektor alakú címke, regresszió

$$h(x) = g_2(W_2 g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y \quad \leftarrow y^{\wedge}, y \text{ vektor}$$

Költség: A címkevektor elemei szerinti négyzetes költségeket átlagoljuk

$$J(\Theta) = \frac{1}{2mk} \sum_{j=1}^m \|\hat{y}^{(j)} - y^{(j)}\|_2^2 = \frac{1}{2m\cancel{k}} \sum_{j=1}^m \boxed{\sum_{i=1}^k} (\hat{y}_i^{(j)} - y_i^{(j)})^2$$


MSE ahogy eddig, de most a **címkevektor elemei felett is átlagolunk.**

Előző órán - Vektor alakú címke, **klasszifikáció**

$$h(x) = g_2(W_2 g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

↖ **y[^], y vektor**

Utolsó aktivációs fv (g_2): Softmax

Költség: Crossentropy

$$J(\theta) = -\frac{1}{m} \sum_{j=1}^m \sum_{i=1}^k y_i \log(\hat{y}_i)$$

one-hot

0
1
0
0

0.64
0.21
0.06
0.09

Előző órán - Softmax

Softmax függvény:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

$$e^{z_i}$$

Az input vektor i-edik eleme az exponenciálisra emelve.

$$\sum_{j=1}^k e^{z_j}$$

Az exponenciálisra emelt vektorelemek összege.

$$\sigma : \mathbb{R}^k \rightarrow \mathbb{R}^k$$

$$\sigma\left(\begin{array}{c} 2.6 \\ 1.5 \\ 0.2 \\ 0.6 \end{array}\right) = \begin{array}{c} 0.64 \\ 0.21 \\ 0.06 \\ 0.09 \end{array}$$

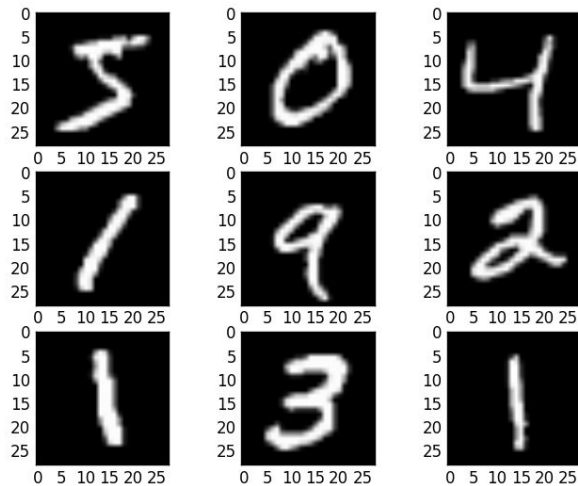
Az eredmény vektor elemeinek összege 1, így értelmezhető valószínűségi eloszlás tömegfv.-eként

MLP alkalmazása kézírás felismerésre

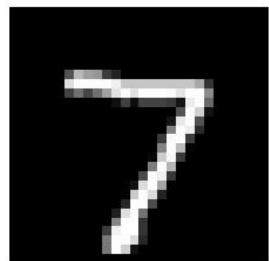
MNIST adatbázis

- kézzel írt számjegyek
- 28×28 -as méretű képek
- 10 kategória (számjegyek: 0 .. 9)
- 60 ezer tanítópélda,
10 ezer teszt példa

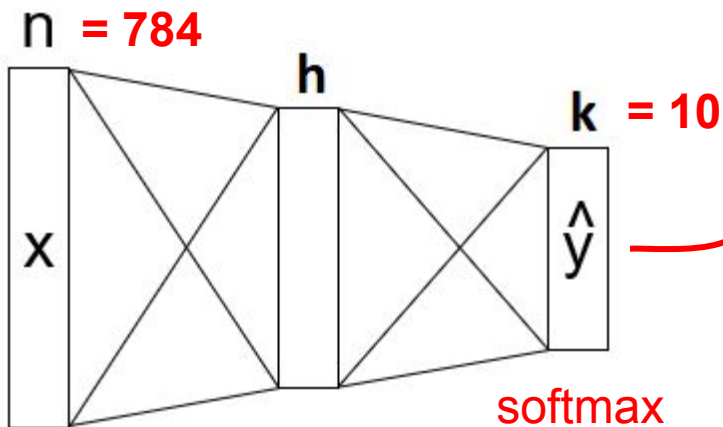
784 input változó: minden pixel
fényereje egy változó



MLP alkalmazása kézírás felismerésre



x



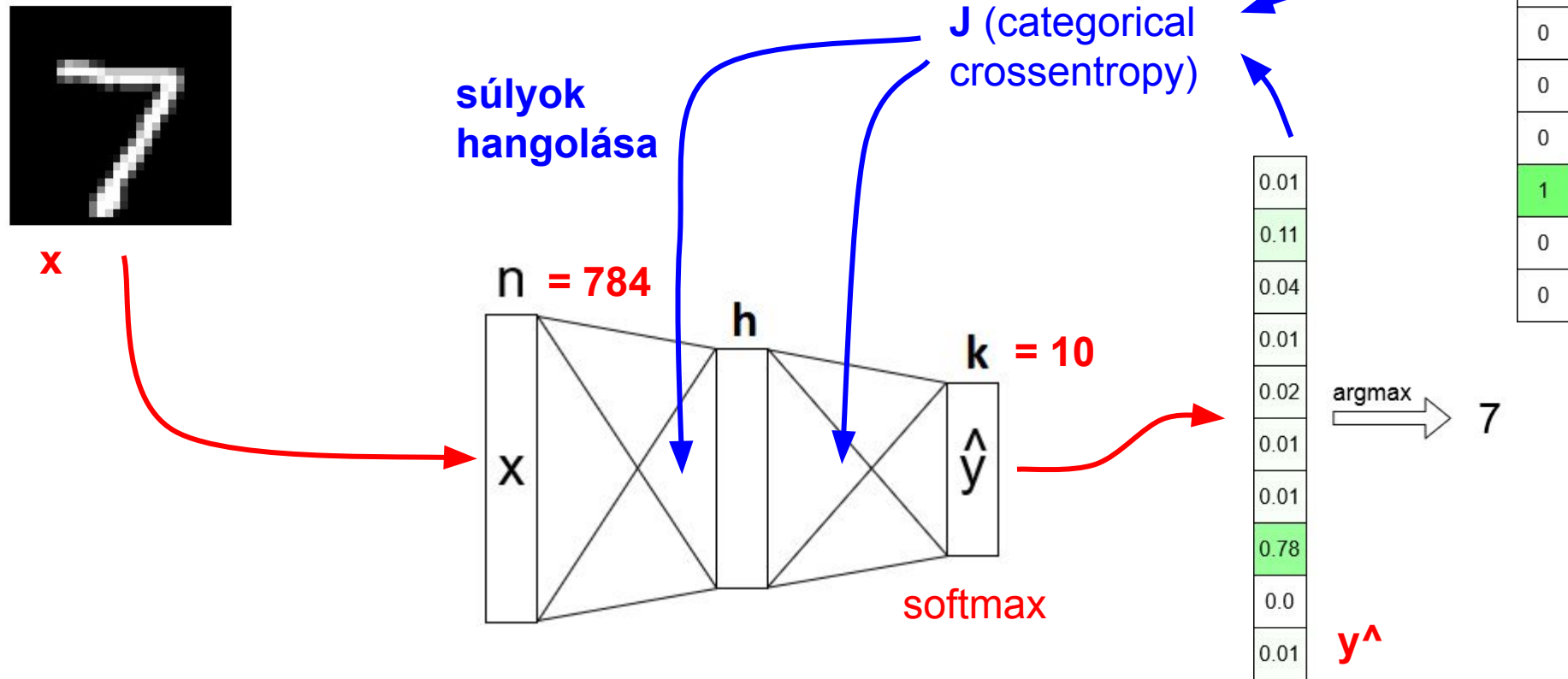
0.01
0.11
0.04
0.01
0.02
0.01
0.01
0.78
0.0
0.01

argmax

7

y^{\wedge}

MLP alkalmazása kézírás felismerésre



Hogyan teljesít az MLP az MNIST adatbázison?

Különböző módszerek teljesítménye az MNIST adatbázison:

<http://yann.lecun.com/exdb/mnist/>

CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN) (log.reg)	none	12.0	LeCun et al. 1998
...			
Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998

Hogyan teljesít az MLP az MNIST adatbázison?

Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
...			
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
...			

Hány paraméter van a hálóban?

Hogyan teljesít az MLP az MNIST adatbázison?

Neural Nets		
2-layer NN, 300 hidden units, mean square error	none	4.7 LeCun et al. 1998
...		
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6 Simard et al., ICDAR 2003
...		

Hány paraméter van a hálóban?
Első réteg: $784 \times 800 + 800$
Második réteg: $800 \times 10 + 10$
Összesen: 636 010 paraméter (!!)

MLP - MNIST

Eddig:

Pl. logisztikus regresszió koleszterinszint becslésére

→ 3 feature, 4 paraméter, több száz mintaelem

Most:

MLP egyetlen rejtett réteggel számjegyek klasszifikálására

→ 784 feature, 636 ezer paraméter, 60 ezer mintaelem

MLP - MNIST

Eddig:

Pl. logisztikus regresszió koleszterinszint becslésére

→ 3 feature, 4 paraméter, több száz mintaelem
<

Most:

MLP egyetlen rejtett réteggel számjegyek klasszifikálására

→ 784 feature, 636 ezer paraméter, 60 ezer mintaelem
>

Mi történhet, ha túl sok
paraméterünk van?

MLP - MNIST

Eddig:

Pl. logisztikus regresszió koleszterinszint becslésére

→ 3 feature, 4 paraméter, több száz mintaelem



Most:

MLP egyetlen rejtett réteggel számjegyek klasszifikálására

→ 784 feature, 636 ezer paraméter, 60 ezer mintaelem



Mi történhet, ha túl sok
paraméterünk van?

Túltanulás (overfitting): A szükségtelenül bonyolult
modell képes a mintaelemek jellegzetességeire egyenként
rátanulni → elveszti az általánosító-képességét

Túltanulás elkerülése

Mit tehetünk a túltanulás elkerülése érdekében?

Túltanulás elkerülése

Mit tehetünk a túltanulás elkerülése érdekében?

Eddig:

- Használjunk egyszerűbb modellt (pl. kevesebb paraméter)!
- Szerezzünk be több tanítóadatot!
- Regularizáció (pl. $\|W\|_2^2$ tag a költségben - L2 reg.)
- Early stopping

Túltanulás elkerülése

Mit tehetünk a túltanulás elkerülése érdekében?

“Szerezzünk be több tanítóadatot!”

- Sajnos ez sokszor nem lehetséges: A címkézett adat **drága**, humán közreműködést igényel.

Túltanulás elkerülése

Mit tehetünk a túltanulás elkerülése érdekében?

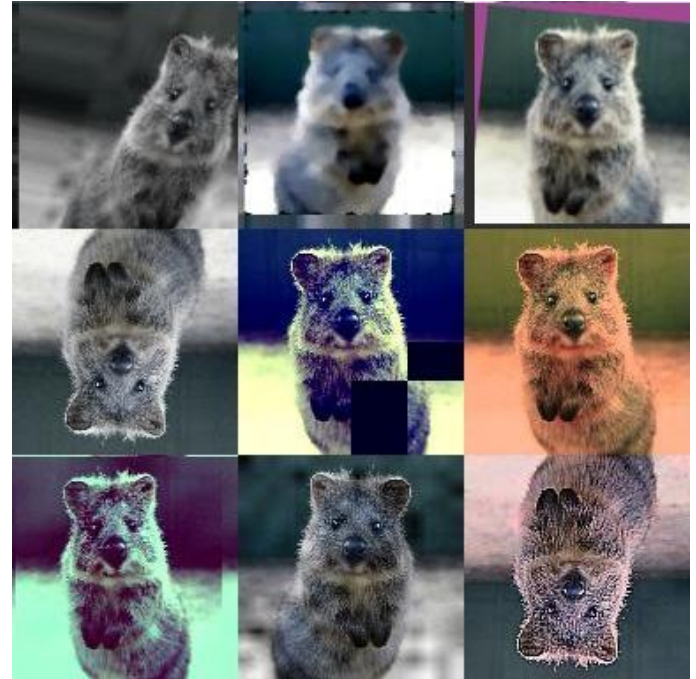
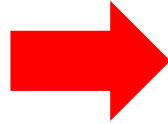
“Szerezzünk be több tanítóadatot!”

- Sajnos ez sokszor nem lehetséges: A címkézett adat **drága**, humán közreműködést igényel.

Próbáljunk meg új tanítópéldákat készíteni a meglevők felhasználásával!

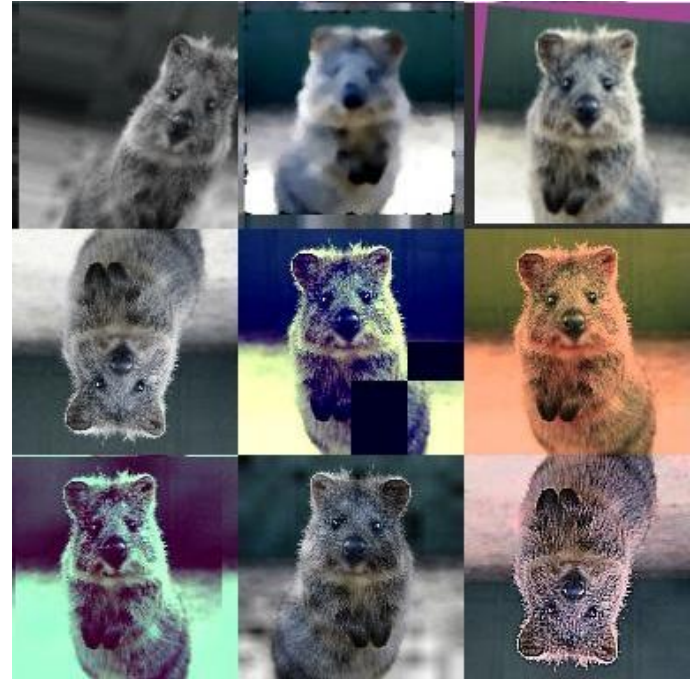
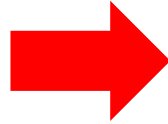
Túltanulás elkerülése - adat augmentáció

Adat-augmentáció (Data augmentation)



Túltanulás elkerülése - adat augmentáció

Adat-augmentáció (Data augmentation)

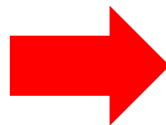


**A hörcsög (?) elforgatva, eltolva,
kinagyítva, átszínezve, ... is hörcsög
marad.**

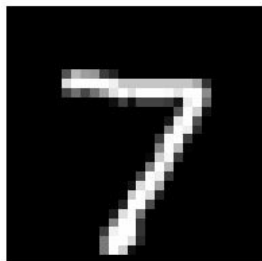
Túltanulás elkerülése - adat augmentáció



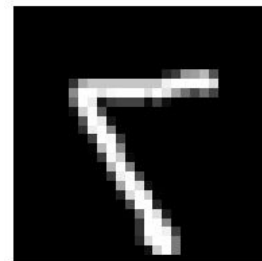
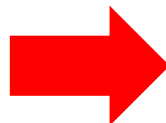
hörcsög



hörcsög



“7”



???

**Nem alkalmazható mindenfajta transzformáció
mindenfajta adatra...**

Túltanulás elkerülése - adat augmentáció

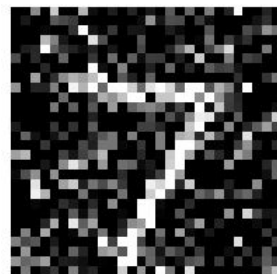
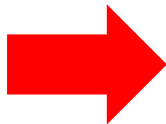
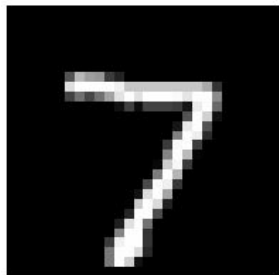
Adat-augmentáció különböző jellegű adatoknál

- **MNIST:** eltolás, kismértékű forgatás, fényerő változtatása, zaj, ...
- **Fényképek:** eltolás, forgatás, tükrözés, torzítás, színek és fényerő változtatása, zaj, bizonyos részletek eltakarása, ...
- **Hangok:** nyújtás, frekvencia változtatása, zaj, ...

Túltanulás elkerülése - zajosítás

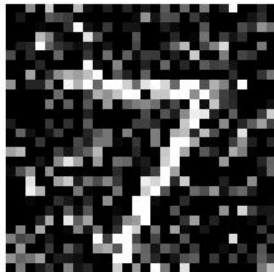
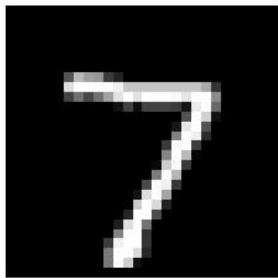
Zajosítás (az adat-augmentáció egy fajtája)

Normál- (Gauss-) eloszlású zajt az input változókhoz adva csökkenthető a túltanulás mértéke.



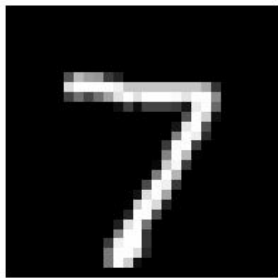
Túltanulás elkerülése - zajosítás

Hogyan segít a zajosítás?



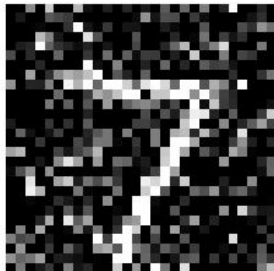
Túltanulás elkerülése - zajosítás

Hogyan segít a zajosítás?



A teljesen összekötött rétegekből álló neuronháló könnyen “memorizál” egyedi eseteket:
pl. ha a pixelértékek összege a képen 6725.81 és 6725.83 közt van, akkor egy 7-es számjegy van a képen.

Ez nyilvánvalóan nem hasznos tudás.

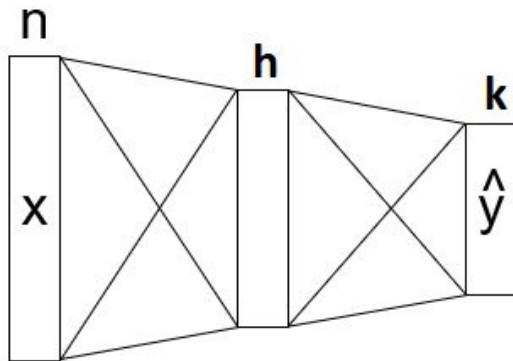


A zajosítás megakadályozza, hogy a háló pontos részleteket “memorizáljon”, így enyhítve a túltanulást.

Túltanulás elkerülése - zajosítás

Nem csak az input zajosítható!

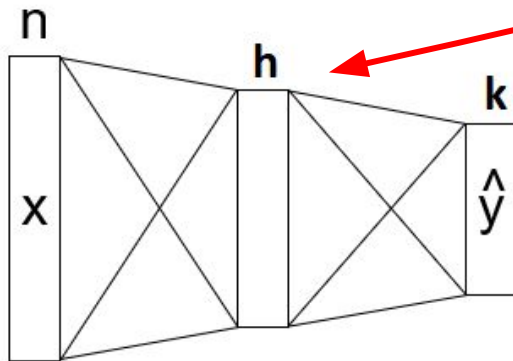
```
model = Sequential()  
model.add(Dense(h, activation='relu', input_dim=n))  
model.add(Dense(k, activation='softmax'))
```



Túltanulás elkerülése - zajosítás

Nem csak az input zajosítható!

```
model = Sequential()  
model.add(Dense(h, activation='relu', input_dim=n))  
model.add(keras.layers.GaussianNoise(0.2))  
model.add(Dense(k, activation='softmax'))
```



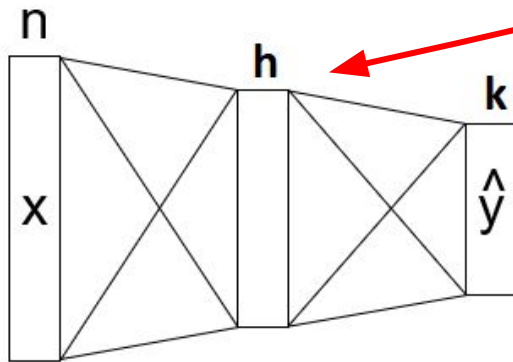
Az egyik rejtett réteg kimenetéhez a betanításkor véletlen zajt adunk.

Túltanulás elkerülése - zajosítás

Nem csak az input zajosítható!

```
model = Sequential()  
model.add(Dense(h, activation='relu', input_dim=n))  
model.add(keras.layers.GaussianNoise(0.2))  
model.add(Dense(k, activation='softmax'))
```

**Predikció esetén ne
zajosítsunk!** A keras
zaj réteg ezt
automatikusan kezeli.

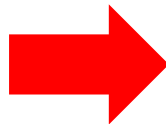


Az egyik rejtett réteg
kimenetéhez a
betanításkor véletlen
zajt adunk.

Túltanulás elkerülése - dropout

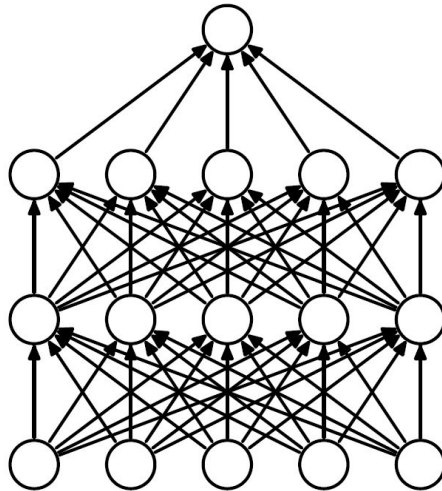
Dropout (a zajosítás egy fajtája)

Kinullázunk bizonyos input változókat.

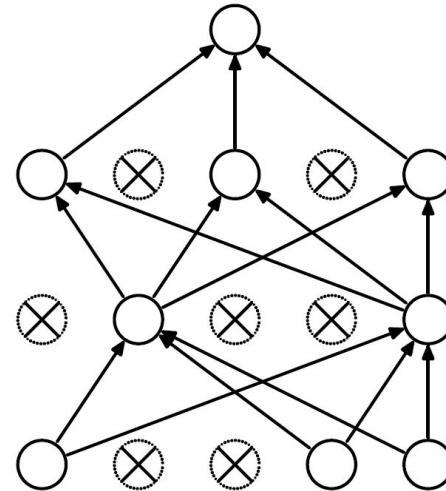


Túltanulás elkerülése - dropout

Dropout-ot leginkább a rejtett reprezentációkon (a rejtett rétegek kimenetén) alkalmaznak.



(a) Standard Neural Net

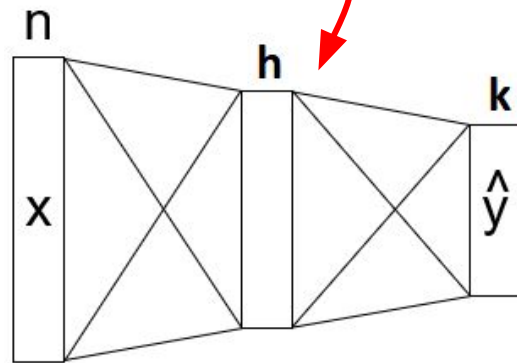


(b) After applying dropout.

Túltanulás elkerülése - dropout

Dropout-ot leginkább a rejtett reprezentációkon (a rejtett rétegek kimenetén) alkalmaznak.

```
model = Sequential()  
model.add(Dense(h, activation='relu', input_dim=n))  
model.add(keras.layers.Dropout(0.2))  
model.add(Dense(k, activation='softmax'))
```



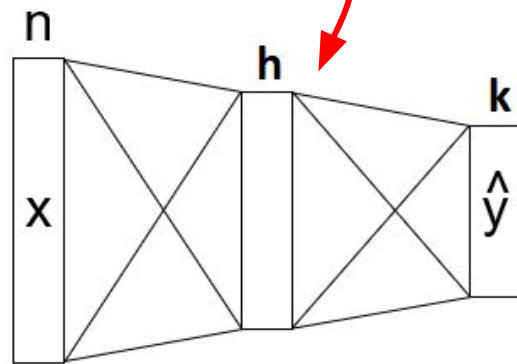
Túltanulás elkerülése - dropout

Dropout-ot leginkább a rejtett reprezentációkon (a rejtett rétegek kimenetén) alkalmaznak.

```
model = Sequential()  
model.add(Dense(h, activation='relu', input_dim=n))  
model.add(keras.layers.Dropout(0.2))  
model.add(Dense(k, activation='softmax'))
```

A rejtett réteg neuronjainak kimenetét adott valószínűséggel kinullázzuk betanításkor.
Minden iterációban újrastorzoljuk a kinullázott indexeket.

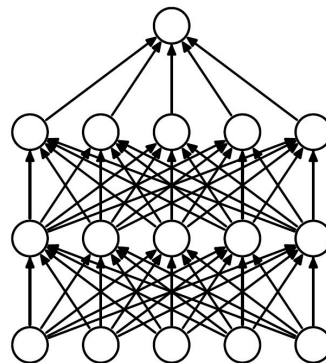
Predikciókor ne használjuk! A Keras dropout réteg ezt automatikusan kezeli.



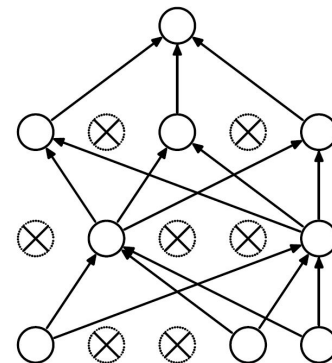
Túltanulás elkerülése - dropout

A dropout értelmezése:

- (Exponenciálisan) sok ritka részháló együttesét tanítjuk be
- Kisebb hálók \rightarrow enyhébb túltanulás
- Sok részháló átlagolása \rightarrow enyhébb túltanulás



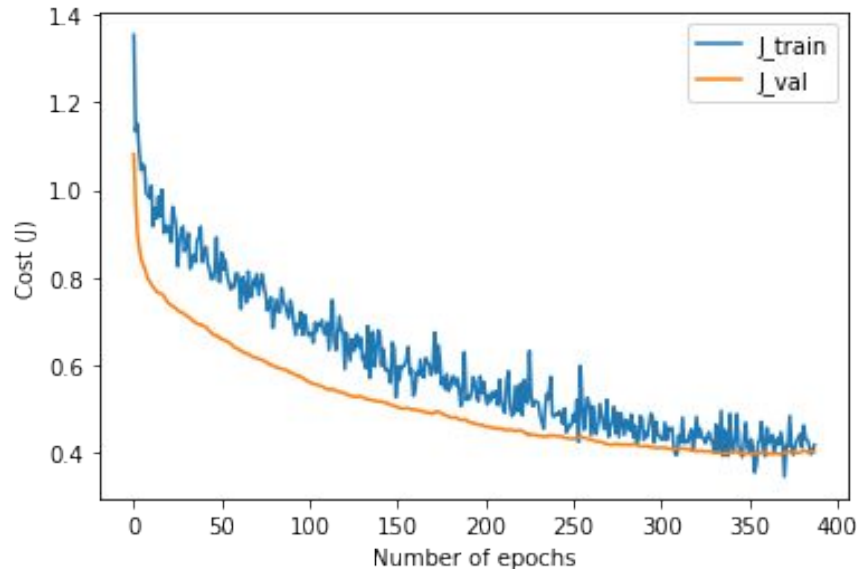
(a) Standard Neural Net



(b) After applying dropout.

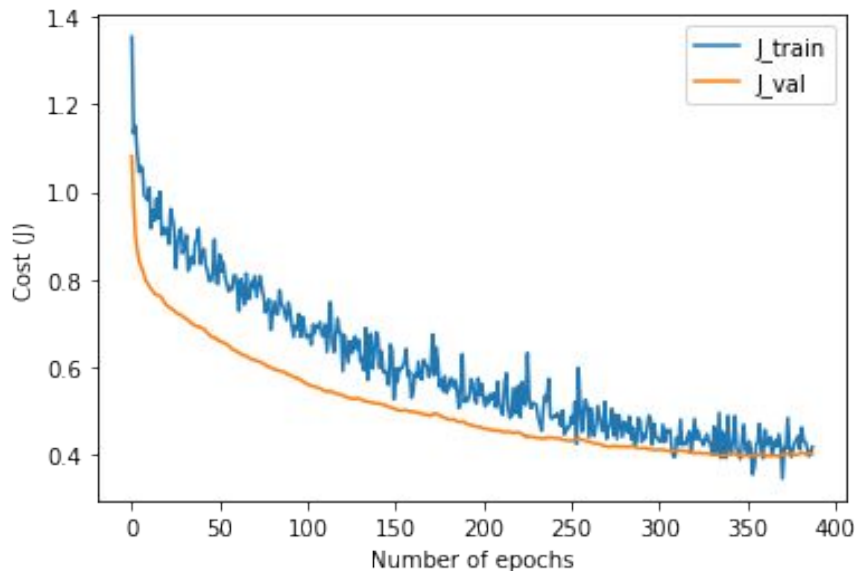
Túltanulás elkerülése - dropout

Dropout alkalmazása során a látszólagos tanítási hiba nagyobb lehet, mint a validációs/teszt hiba.



Túltanulás elkerülése - dropout

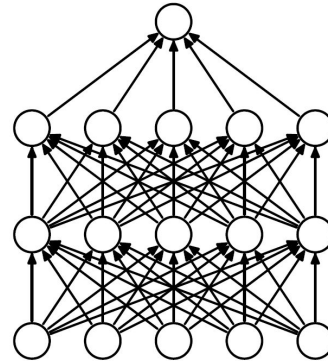
Dropout alkalmazása során a látszólagos tanítási hiba nagyobb lehet, mint a validációs/teszt hiba.



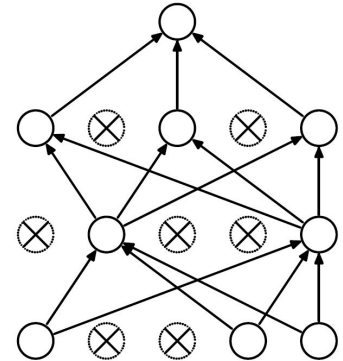
Mivel a betanítás során
alkalmazzuk a zajt, validáció/teszt
közben pedig nem, **a**
tanítómintán mért költség
nagyobb lehet a valóságosnál.

Túltanulás elkerülése - dropout

- Dropout alkalmazásával a betanítás lelassul.
- Azonban, nagyon sok esetben jelentős javulás érhető el a használatával.



(a) Standard Neural Net



(b) After applying dropout.

Hogyan teljesít az MLP az MNIST adatbázison?

Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
...			
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
...			

Hogyan teljesít az MLP az MNIST adatbázison?

Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
...			
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
...			
6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	none	0.35	Ciresan et al. Neural Computation 10, 2010 and arXiv 1003.0358, 2010

Az MNIST adatbázison egy megfelelően nagy MLP típusú háló az említett regularizációs módszerekkel **emberfeletti eredményeket ér el.**

MLP alkalmazása képek klasszifikációjára

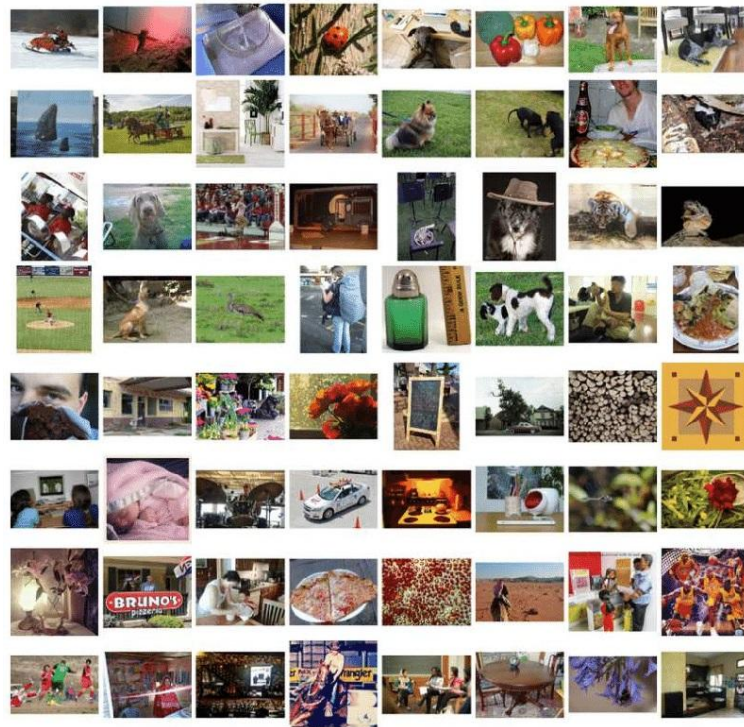
Az MLP jól teljesít kézírásos karakterek felismerésében.

Mi a helyzet a nagyfelbontású fényképekkel?

ImageNet adatbázis

Fényképek különböző objektum-kategóriákkal.

- 1000+ kategória, például:
 - mocsári teknős
 - gofrisütő
 - Norfolk terrier
 - viadukt
- 1 megapixel körüli felbontás, színes
→ **kb. 3 millió input változó**



ImageNet adatbázis

Különböző módszerek pontossága az ImageNet teszthalmazán
(5 tippből eltalálni a megfelelő kategóriát, 1000 kategória közül):

Véletlenszerű találgatás: **0.5%**

Humán: **94.9%**

“FixResNeXt-101 32x48d” (2019): **98%**

ImageNet adatbázis

Különböző módszerek pontossága az ImageNet teszthalmazán
(5 tippből eltalálni a megfelelő kategóriát, 1000 kategória közül):

Véletlenszerű találgatás: **0.5%**

Humán: **94.9%**

“FixResNeXt-101 32x48d” (2019): **98%**

MLP: **~0.5%** :(

MLP alkalmazása képek klasszifikációjára

Miért nem alkalmas az MLP modell nagyfelbontású képek klasszifikációjára?

MLP alkalmazása képek klasszifikációjára

Miért nem alkalmas az MLP modell nagyfelbontású képek klasszifikációjára?

Ugyanazt a mintázatot a kép összes lehetséges pontján meg kell tanulni felismerni.



MLP alkalmazása képek klasszifikációjára

Miért nem alkalmas az MLP modell nagyfelbontású képek klasszifikációjára?

Ugyanazt a mintázatot a kép összes lehetséges pontján meg kell tanulni felismerni.

Ehhez az MLP-nek mindegyik kategóriába tartozó objektumot minden pozícióban, minden méretben, minden elforgatásban látnia kell inputként és meg is kellene tudni tanulni.

Eddig: pl. koleszterinszint becslése - x_1 a páciens tömege, x_2 a páciens életkora, stb., → **minden változó fix jelentést hordoz.**

MLP alkalmazása képek klasszifikációjára

Miért nem alkalmas az MLP modell nagyfelbontású képek klasszifikációjára?

Ugyanazt a mintázatot a kép összes lehetséges pontján meg kell tanulni felismerni.

Ehhez az MLP-nek mindegyik kategóriába tartozó objektumot minden pozícióban, minden méretben, minden elforgatásban látnia kell inputként és meg is kellene tudni tanulni.

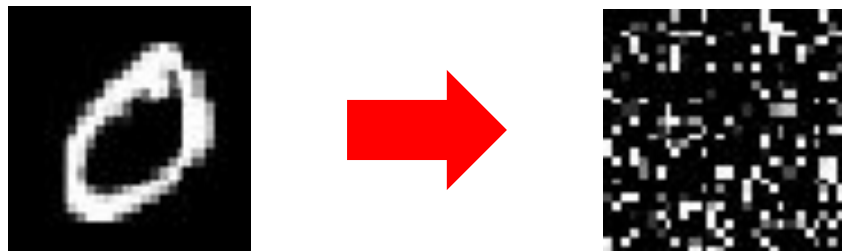
Most: bármelyik input változó reprezentálhatja bármilyen objektumnak bármelyik részletét. **Egy-egy súly/neuron kombinatorikusan sok információt kellene, hogy megtanuljon!**

MLP alkalmazása képek klasszifikációjára

Láttuk, hogy az MNIST karaktereken jól teljesít az MLP modell.

Vajon hogy teljesít, ha egy előre választott permutáció szerint az összes MNIST mintaelem pixeleit megkeverjük?

(Ugyanolyan módon permutált képeken tanítunk és tesztelünk)



MLP alkalmazása képek klasszifikációjára

Láttuk, hogy az MNIST karaktereken jól teljesít az MLP modell.

Vajon hogy teljesít, ha egy előre választott permutáció szerint az összes MNIST mintaelem pixeleit megkeverjük?

Nagyjából ugyanúgy, mint az eredeti képeken, hiszen az MLP nem feltételez semmilyen szomszédsági viszonyt az egyes input változók közt.

Képfeldolgozás esetén ez nem egy jó megközelítés...

Képi mintázatok felismerése

Hogyan érdemes megközelíteni a problémát?

- Ha egy mintázatot megtanulunk felismerni, ismerjük fel attól függetlenül, hogy hol helyezkedik el a képen
→ **Transzláció (eltolás) invariancia**



Képi mintázatok felismerése

Hogyan érdemes megközelíteni a problémát?

- Transzláció (eltolás) invariancia
- Minél nagyobb látszólagos kiterjedésű objektumot próbálunk felismerni, annak annál többféle megjelenése lehet a képen
→ **Tanuljunk apró mintázatokat felismerni**



Képi mintázatok felismerése

Hogyan érdemes megközelíteni a problémát?

- Transzláció (eltolás) invariancia
- Tanuljunk apró mintázatokot felismerni

Ötlet: Haladjunk végig a kép minden lehetséges pozícióján egy kis ablakkal és próbáljuk az ablakban megjelenő mintázatok közül megtanulni felismerni a fontosakat!

Képi mintázatok felismerése

Hogyan érdemes megközelíteni a problémát?

- Transzláció (eltolás) invariancia
- Tanuljunk apró mintázatokot felismerni

Ötlet: Haladjunk végig a kép minden lehetséges pozícióján egy kis ablakkal és próbáljuk az ablakban megjelenő mintázatok közül megtanulni felismerni a fontosakat!

→ **Konvolúciós réteg**

2D Konvolúció - diszkrét eset

1	2	2	-2
1	0	2	4
2	-1	-1	3
0	0	-3	-2

x

*

1	0	3
2	-1	-1
0	1	-3

w

=

9	

y

$$1 \cdot 1 + 2 \cdot 0 + 2 \cdot 3 + 1 \cdot 2 + 0 \cdot (-1) + 2 \cdot (-1) + 2 \cdot 0 + (-1) \cdot 1 + (-1) \cdot (-3) = 9$$

2D Konvolúció - diszkrét eset

1	2	2	-2
1	0	2	4
2	-1	-1	3
0	0	-3	-2

x

*

1	0	3
2	-1	-1
0	1	-3

w

=

9	-20

y

$$2 \cdot 1 + 2 \cdot 0 + (-2) \cdot 3 + 0 \cdot 2 + 2 \cdot (-1) + 4 \cdot (-1) + (-1) \cdot 0 + (-1) \cdot 1 + 3 \cdot (-3) = -20$$

2D Konvolúció - diszkrét eset

1	2	2	-2
1	0	2	4
2	-1	-1	3
0	0	-3	-2

x

*

1	0	3
2	-1	-1
0	1	-3

w

=

9	-20
22	

y

$$1 \cdot 1 + 0 \cdot 0 + 2 \cdot 3 + 2 \cdot 2 + (-1) \cdot (-1) + (-1) \cdot (-1) + 0 \cdot 0 + 0 \cdot 1 + (-3) \cdot (-3) = 22$$

2D Konvolúció - diszkrét eset

1	2	2	-2
1	0	2	4
2	-1	-1	3
0	0	-3	-2

x

*

1	0	3
2	-1	-1
0	1	-3

w

=

9	-20
22	11

y

$$0 \cdot 1 + 2 \cdot 0 + 4 \cdot 3 + (-1) \cdot 2 + (-1) \cdot (-1) + 3 \cdot (-1) + 0 \cdot 0 + (-3) \cdot 1 + (-2) \cdot (-3) = 11$$

2D Konvolúció - diszkrét eset

A kimeneti kép (hő térkép, heatmap) mérete akkora, ahány lehetséges pozícióba tudtuk elhelyezni a képen a filtert.

1	2	2	-2
1	0	2	4
2	-1	-1	3
0	0	-3	-2

x

*

1	0	3
2	-1	-1
0	1	-3

w

=

9	-20
22	11

y



Az **x** kép minden lehetséges pontjára ráhelyezzük a **w** filtert (kernelt) és a skalárszorzatukból előállítunk egy kimeneti pixelértéket.

$$0 \cdot 1 + 2 \cdot 0 + 4 \cdot 3 + (-1) \cdot 2 + (-1) \cdot (-1) + 3 \cdot (-1) + 0 \cdot 0 + (-3) \cdot 1 + (-2) \cdot (-3) = 11$$

A **w** filter tartalmazza a felismerendő mintázatot, ezt tanuljuk gradiensemódszerrel.

2D Konvolúció - diszkrét eset

Input: $x \in \mathbb{R}^{I \times J}$

Filter (kernel): $w \in \mathbb{R}^{U \times V}$

Output (heatmap): $\hat{y} \in \mathbb{R}^{(I-U+1) \times (J-V+1)}$

$$\begin{aligned}\hat{y}[i, j] &= (x * w)[i, j] = \langle x[i..i+U, j..j+V], w \rangle = \\ &= \sum_{u=1}^U \sum_{v=1}^V x[i+u, j+v] \cdot w[u, v]\end{aligned}$$

Konvolúció a fényképek utófeldolgozásában

Filter simításhoz (Gaussian smoothing)

$$\frac{1}{1068} \cdot$$

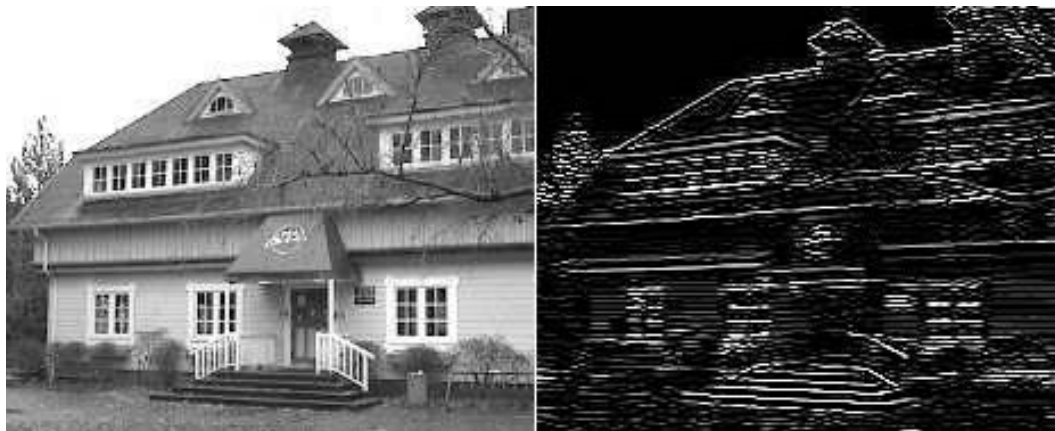
0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0



Konvolúció a fényképek utófeldolgozásában

Filter a vízszintes élek kiemeléséhez

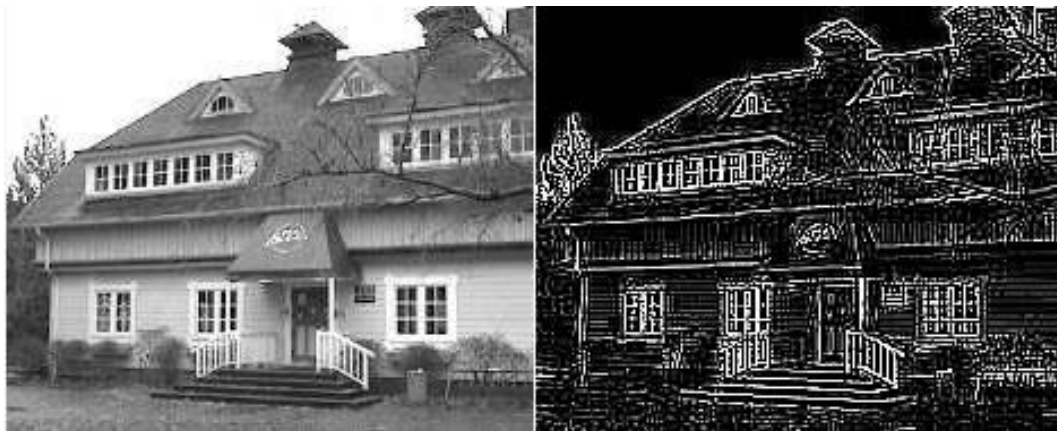
-1	-1	-1
2	2	2
-1	-1	-1



Konvolúció a fényképek utófeldolgozásában

Filter élkimeléshez (edge detection)

-1	-1	-1
-1	8	-1
-1	-1	-1



Konvolúció textúra felismeréséhez

Input

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Filter



1	-1	-1
-1	1	-1
-1	-1	1

=

Output
(heatmap, hőkép)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Konvolúció textúra felismeréséhez

Példa jó illeszkedésre: magas érték a heatmap megfelelő pontján

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Konvolúció textúra felismeréséhez

Példa rossz illeszkedésre: alacsony érték a heatmap megfelelő pontján

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Konvolúciós réteg

Konvolúciós réteg:

- a filterek és a hozzájuk tartozó bias skalárok a **paraméterek**
- aktivációs függvény (g), pl. ReLU


$$\hat{y}[i, j] = g\left(\sum_{u=1}^U \sum_{v=1}^V x[i+u, j+v] \cdot w[u, v] + b\right)$$

$$x \in \mathbb{R}^{I \times J}$$

$$w \in \mathbb{R}^{U \times V}$$

$$b \in \mathbb{R}$$

$$\hat{y} \in \mathbb{R}^{(I-U+1) \times (J-V+1)}$$

2D indexelés és méretek, de más dimenzionalitás esetén is ugyanígy...

Konvolúciós réteg - csatornák

- Egy filter egy mintázatot fog tudni felismerni.
- Szeretnénk egy konvolúciós réteggel több mintázatot felismerni.

→ **Több filter** (és ugyanennyi bias skalár) **egyidejű tanulása**.

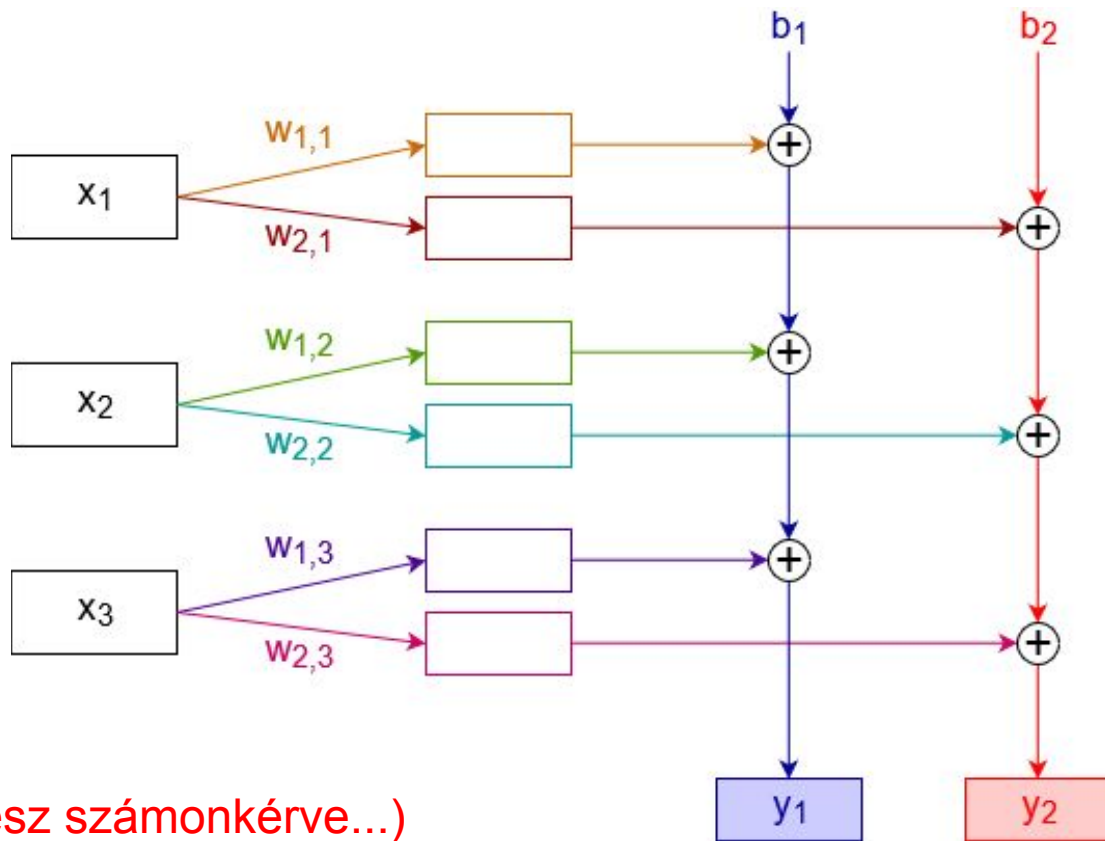
Az inputot minden egyes filterrel konvolváljuk, így ugyanannyi output heatmap-et kapunk, ahány filterünk van.

Az egymástól függetlenül előállított output heatmap-ek az output **csatornái**.

Konvolúciós réteg - csatornák

Ha halmozni szeretnénk a konvolúciós rétegeket, az input is lehet több csatornás.

Ekkor a filterek is több csatornával rendelkeznek.



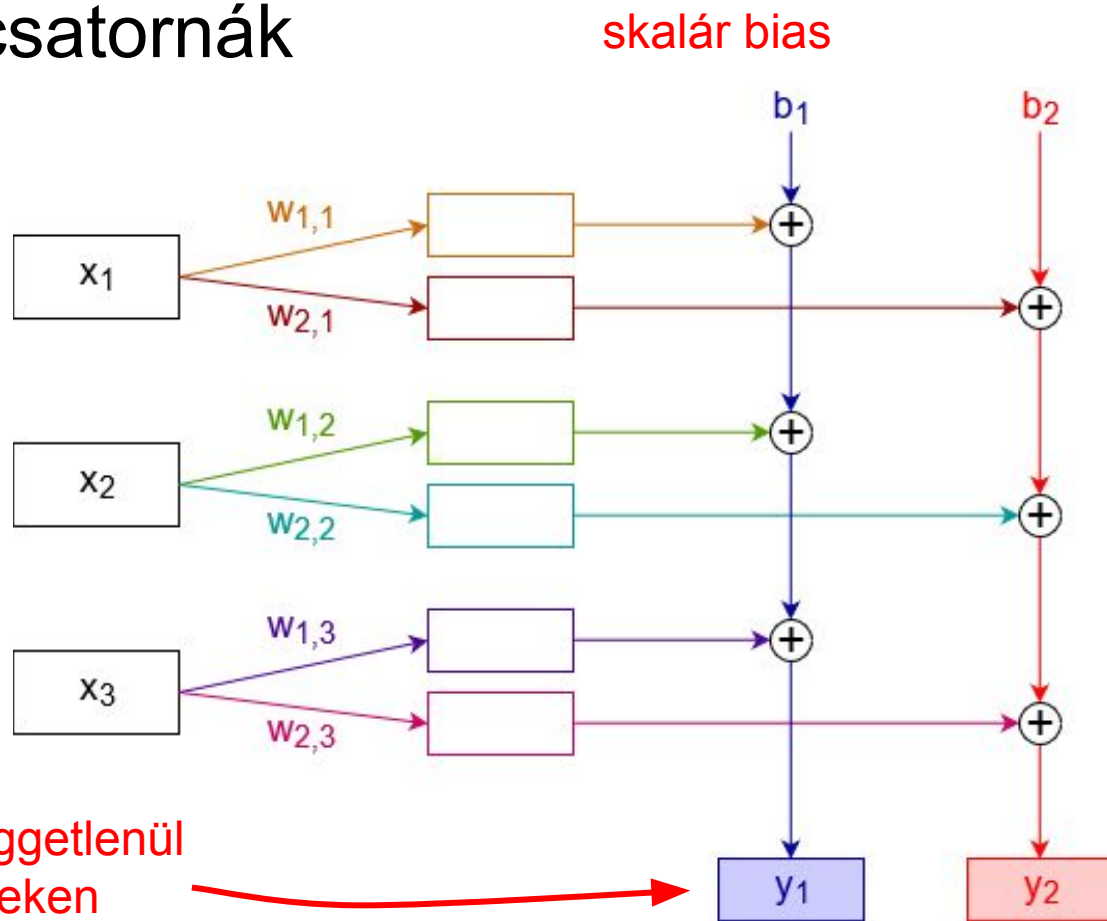
(Ilyen részletességgel nem lesz számonkérve...)

Konvolúciós réteg - csatornák

Példa: 3 input channel,
2 output channel
→ 2 darab, 3 csatornás filter

(Ha színes, pl. RGB kódolású
kép az input, akkor az már eleve
3 (szín)csatornás)

Aktivációs fv. csatornánként függetlenül
alkalmazva az y_i heatmap-eken



Konvolúciós réteg - csatornák

Konvolúciós réteg (több csatorna esetén):

- P darab input csatorna, Q darab output csatorna

$$\hat{y}_q[i, j] = g\left(\sum_{p=1}^P \sum_{u=1}^U \sum_{v=1}^V x_p[i + u, j + v] \cdot w_{p,q}[u, v] + b_q \right)$$

$$x_p \in \mathbb{R}^{I \times J}$$

$$b_q \in \mathbb{R}$$

$$q \in \{1, \dots, Q\}$$

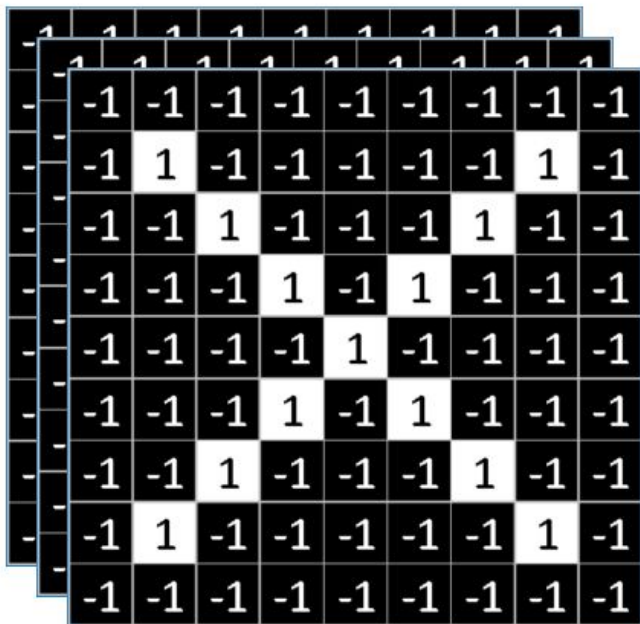
$$w_{p,q} \in \mathbb{R}^{U \times V}$$

$$\hat{y}_q \in \mathbb{R}^{(I-U+1) \times (J-V+1)}$$

(Ilyen részletességgel nem lesz számonkérve...)

Konvolúciós réteg - csatornák

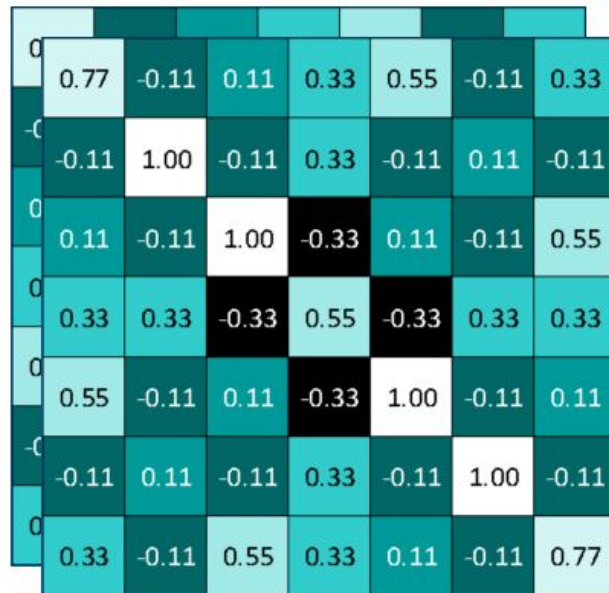
P darab input csatorna



Konvolúciós réteg
Q darab (**P** csatornás)
filterrel.



Q darab output csatorna



Keras, konvolúciós réteg

Egycsatornás input,
pl. MNIST

```
model = Sequential()
```

← Input tömb alakja: (n_mintaelem, 28, 28, 1)

```
model.add(Conv2D(32, (3, 3), activation="relu",\n                 input_shape=(28,28,1)))
```

← Tömb alakja: (n_mintaelem, 26, 26, 32)

```
model.add(Conv2D(64, (5, 5), activation="relu"))
```

← Tömb alakja: (n_mintaelem, 22, 22, 64)

↪ batch_size

Keras, konvolúciós réteg

**Többcsatornás input, pl.
színes képek
(3 input csatorna: R,G,B)**

```
model = Sequential()
```

← Input tömb alakja: (n_mintaelem, 28, 28, 3)

```
model.add(Conv2D(32, (3, 3), activation="relu",\n                 input_shape=(28,28,3)))
```

← Tömb alakja: (n_mintaelem, 26, 26, 32)

```
model.add(Conv2D(64, (5, 5), activation="relu"))
```

← Tömb alakja: (n_mintaelem, 22, 22, 64)

↪ batch_size

Konvolúciós réteg

Mire jutottunk?

- Több apró, egyszerű mintázatot megtanulhatunk felismerni.
- A konvolúciós réteg kimenete a képnek azokon a pontjain ad vissza **magas értéket, ahol a filterekre hasonlító mintázat** található meg.

Konvolúciós réteg

Mire jutottunk?

- Több apró, egyszerű mintázatot megtanulhatunk felismerni.
- A konvolúciós réteg kimenete a képnek azokon a pontjain ad vissza **magas értéket, ahol a filterekre hasonlító mintázat** található meg.

Probléma:

- Ha regressziót/klasszifikációt szeretnénk tanulni, a konvolúciós réteg kimenetén teljesen összekötött rétegeket kell alkalmaznunk.
- Nagyobb mintázatokkal továbbra is bajban vagyunk...

Downsampling / Pooling réteg

Ötlet:

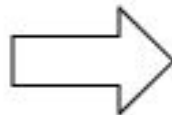
Az egyszerű regressziós/klasszifikációs feladatnál nem kell pixelre pontosan megmondanunk, hol milyen objektum található a képen.

Csökkentsük a heatmap-ek felbontását!

Downsampling / Pooling réteg

Average pooling (2x2)

5	2	-3	0
2	-1	4	-1
-4	-4	-3	0
3	5	0	-1



$$(5 + 2 + 2 + (-1)) / 4 = 2$$

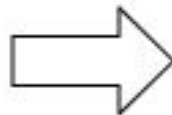
2	0
0	-1

Average pooling: egyszerű leskálázás (downsampling) átlagolással

Downsampling / Pooling réteg

Max pooling (2x2)

5	2	-3	0
2	-1	4	-1
-4	-4	-3	0
3	5	0	-1



$$\max(\{5, 2, 2, -1\}) = 5$$

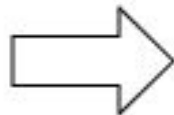
5	4
5	0

Max pooling: leskálázáskor a blokkonkénti maximumértéket visszük tovább

Downsampling / Pooling réteg

Max pooling (2x2)

5	2	-3	0
2	-1	4	-1
-4	-4	-3	0
3	5	0	-1



$$\max(\{5, 2, 2, -1\}) = 5$$

5	4
5	0

Max pooling: csak a maximális értékek megtartása leskálázás során;
Bizonyos feladatoknál (objektum detektálás) esetleg csak arra vagyunk kíváncsiak, hogy egy adott mintázatnak a legintenzívebb jelenléte mennyire intenzív. A gyengébb észlelések intenzitásának mértéke kevésbé fontos...

Keras, pooling réteg

```
model = Sequential()
```

← Input tömb alakja: (n_mintaelem, 28, 28, 3)

```
model.add(Conv2D(32, (3, 3), activation="relu",\n                 input_shape=(28,28,3)))
```

← Tömb alakja: (n_mintaelem, 26, 26, 32)

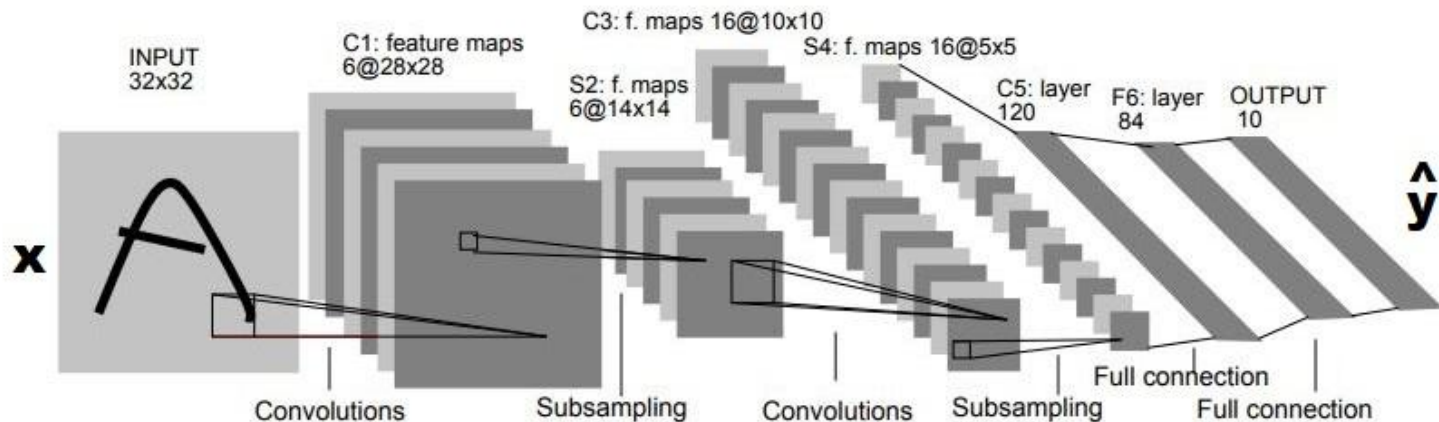
```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

← Tömb alakja: (n_mintaelem, 13, 13, 32)

↪ batch_size

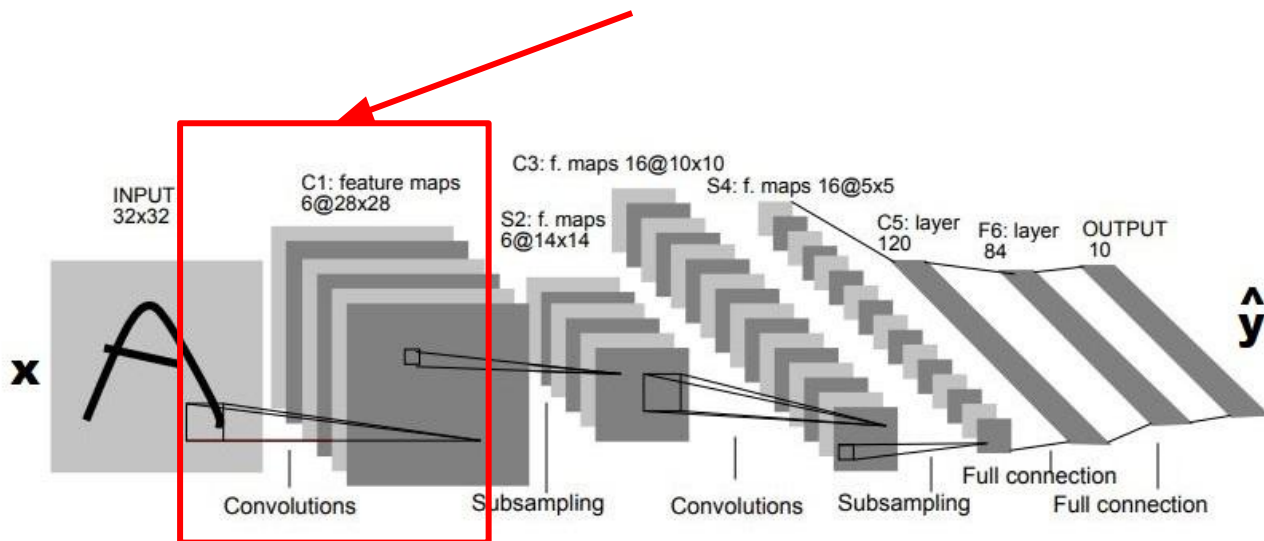
Konvolúciós háló - LeNet-5 típusú architektúra

LeNet-5 típusú architektúra (1998, Y. LeCun et al.)



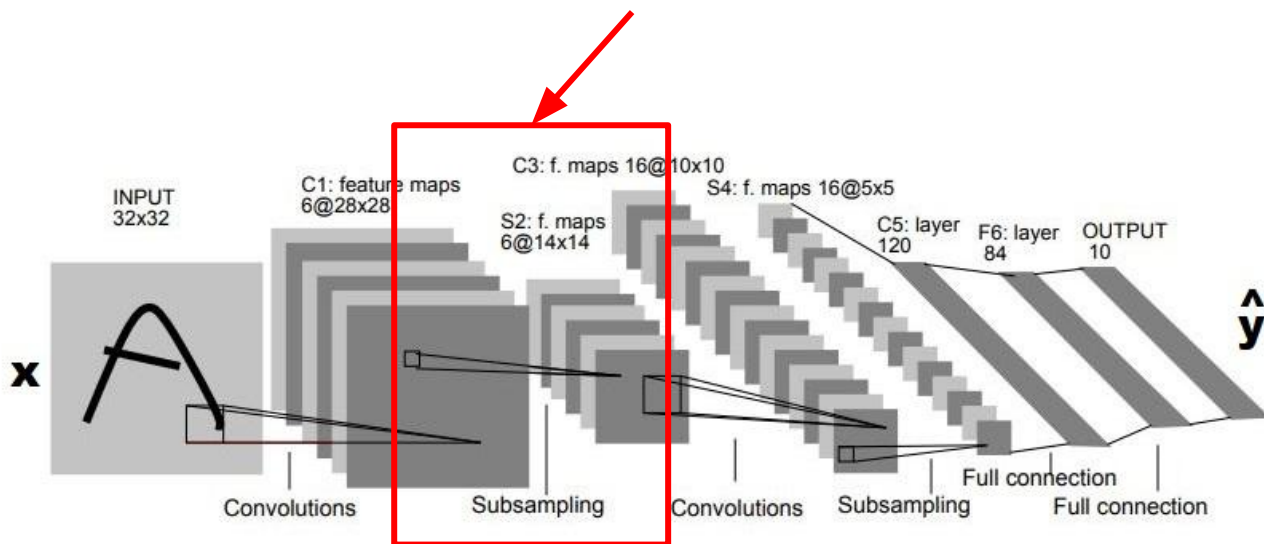
Konvolúciós háló - LeNet-5 típusú architektúra

Első konvolúciós réteg: különböző (itt konkrétan 6 darab, 5x5 méretű) filterek tanulása élek, sarkok, árnyalatok felismerésére → minden filterrel való konvolúcióból 1-1 heatmap az eredmény



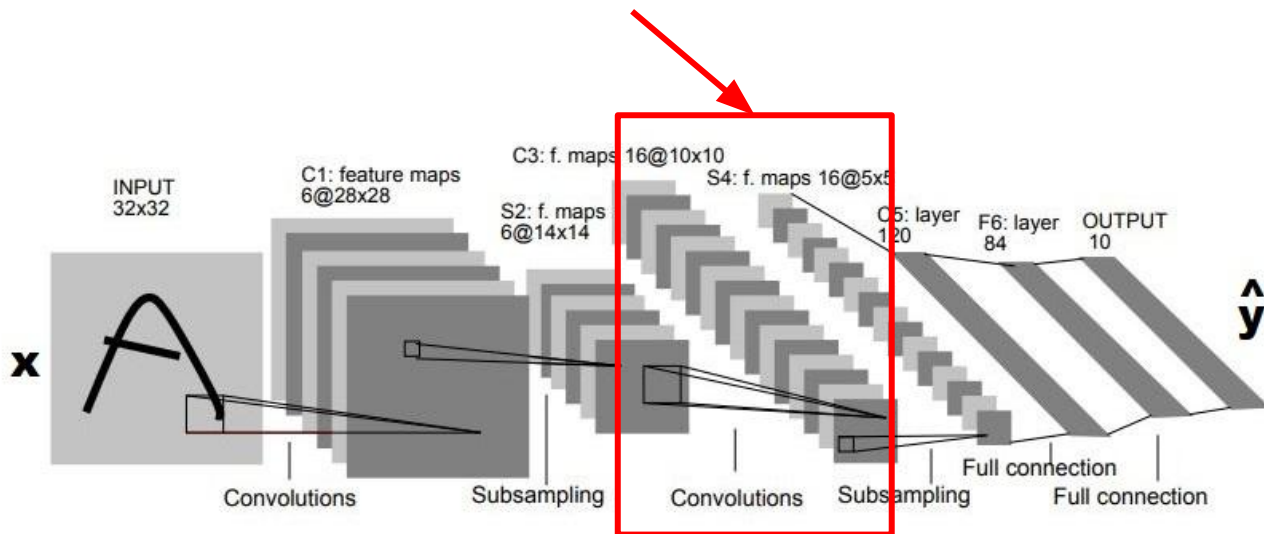
Konvolúciós háló - LeNet-5 típusú architektúra

Első pooling réteg: az előző rétegből kapott (6 darab) heatmap leskálázása csatornánként függetlenül (itt konkrétan 2x2-es leskálázás, azaz mindkét tengely mentén a felére csökken a képméret)



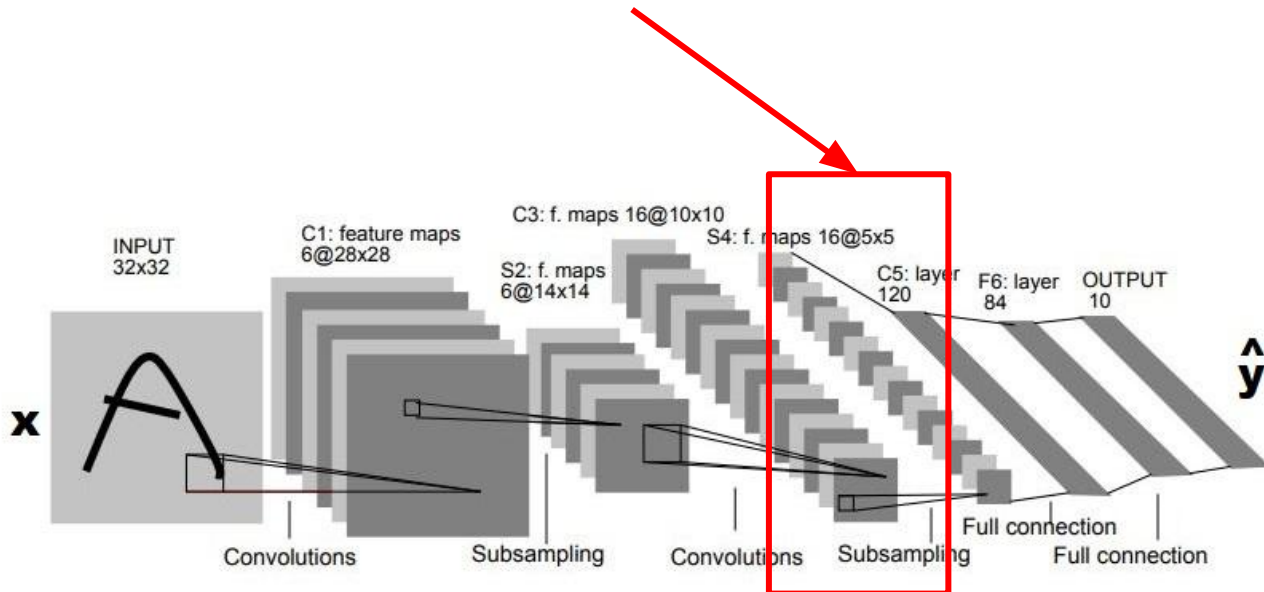
Konvolúciós háló - LeNet-5 típusú architektúra

Második konvolúciós réteg: számjegyek részeit felismerő (itt konkrétan 16 darab, 6 csatornás, 5x5 méretű) filterek tanulása, melyek már az első réteg által felismert éleket, sarkokat reprezentáló heatmap-eken dolgoznak



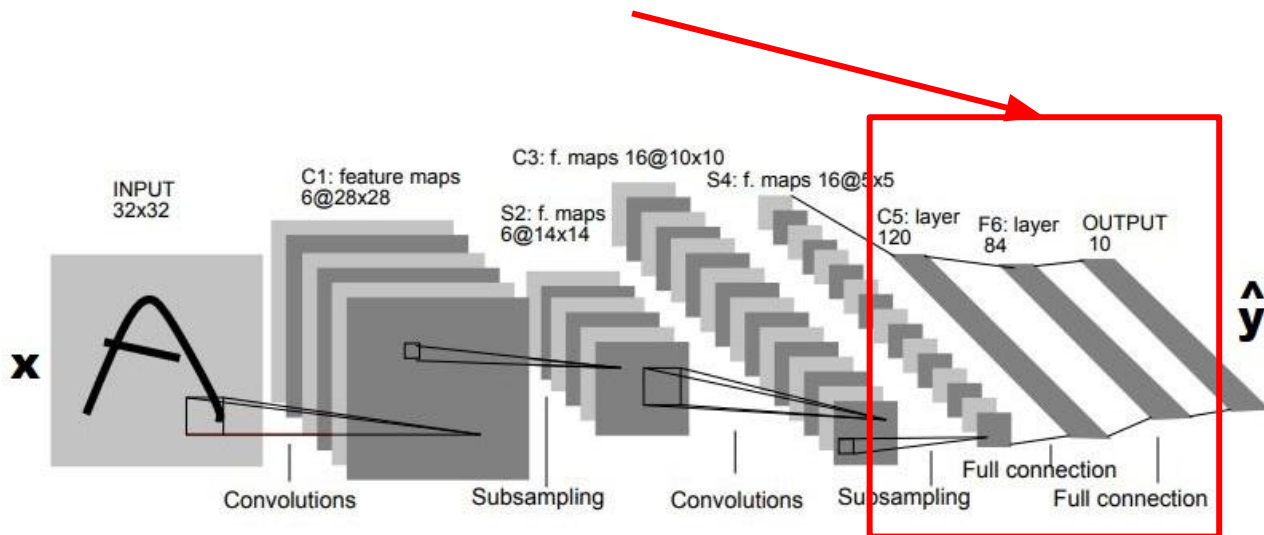
Konvolúciós háló - LeNet-5 típusú architektúra

Második pooling réteg: az előző rétegből kapott (16 darab) heatmap leskálázása (itt konkrétan 2x2, azaz mindkét tengely mentén a felére)



Konvolúciós háló - LeNet-5 típusú architektúra

Teljesen összekötött rétegek: az utolsó pooling réteg heatmap-jeit egyetlen vektorba rendezzük, majd hagyományos MLP-t illesztünk rá klasszifikálásra, vagy regresszióhoz



Konvolúciós háló felépítése

A váltakozó konvolúciós és pooling rétegek a **hierarchikus mintázatfelismeréshez** ideálisak.

Nagyobb, magasabb szintű mintázatokat több kisebb, egyszerűbb mintázat kombinációjaként tanulja a háló felismerni.

Konvolúciós háló felépítése

A váltakozó konvolúciós és pooling rétegek a **hierarchikus mintázatfelismeréshez** ideálisak.

Nagyobb, magasabb szintű mintázatokat több kisebb, egyszerűbb mintázat kombinációjaként tanulja a háló felismerni.

A konvolúciós rétegek felelnek a mintázatok felismeréséért.

A pooling rétegek felelnek azért, hogy a heatmap-ek leskálázásával lehetővé tegyék a következő konvolúciós rétegnek, hogy nagyobb mintázatokat tudjanak felismerni.

(növelik a következő rétegbeli neuronok látóterének méretét)

Keras, LeNet-5 (egyszerűsített)

```
model = Sequential()
model.add(Conv2D(6, (5, 5), activation="tanh",\
                input_shape=(32,32,1)))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), activation="tanh"))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='tanh',))
model.add(Dense(84, activation='tanh'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=sgd)
```

Keras, LeNet-5 (egyszerűsített)

```
model = Sequential()  
model.add(Conv2D(6, (5, 5), activation="tanh", \n                 input_shape=(32, 32, 1)))  
model.add(AveragePooling2D(pool_size=(2, 2)))  
model.add(Conv2D(16, (5, 5), activation="tanh"))  
model.add(AveragePooling2D(pool_size=(2, 2)))  
model.add(Flatten())  
model.add(Dense(120, activation='tanh',))  
model.add(Dense(84, activation='tanh'))  
model.add(Dense(10, activation='softmax'))  
model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

(n_mintaelem, 32, 32, 1)

(n_mintaelem, 28, 28, 6)

(n_mintaelem, 14, 14, 6)

(n_mintaelem, 10, 10, 16)

(n_mintaelem, 5, 5, 16)

(n_mintaelem, 400)

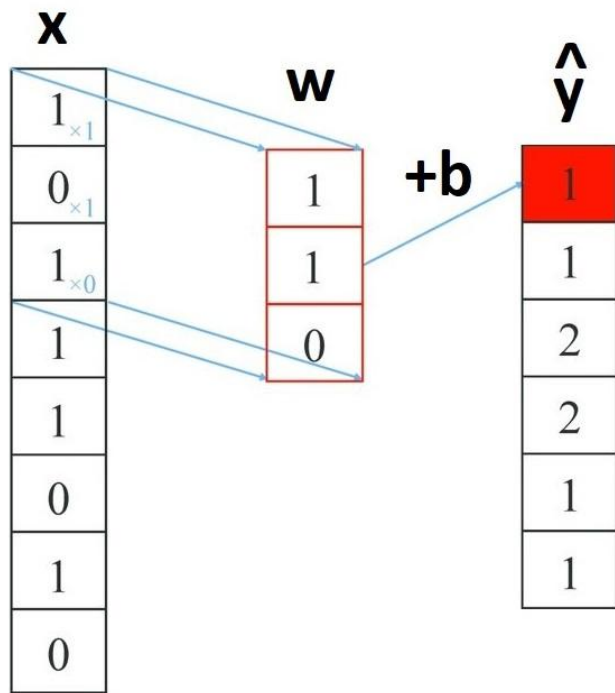
(n_mintaelem, 120)

(n_mintaelem, 84)

(n_mintaelem, 10)

Konvolúciós réteg - 1D

A konvolúció nem csak 2 dimenzió esetén definiálható:



$$\begin{aligned}\hat{y}_i &= g(\langle w, x_{i..(i+U)} \rangle + b) = \\ &= g\left(\sum_{u=1}^U w_u \cdot x_{i..(i+U)} + b\right)\end{aligned}$$

$$w \in \mathbb{R}^U$$

$$b \in \mathbb{R}$$

1D konvolúciós réteg!

Konvolúciós háló

A konvolúciós réteg **transzláció invariáns**.

A konvolúciós háló **ideális**, amikor **egy-, vagy több tengely mentén sorbarendezhetők az input változók**.

Nem feltétlenül csak képfelismerésben használható:

- Hangelemzés, beszédfelismerés (1D, 2D)
- MRI, CT felvételek, 3D modellek (3D)
- Játéktábla (pl. AlphaGo, 2D)
- Pénzügyi adatok, idősorok (1D, 2D)