

# Deep Learning: Mélyhálós Tanulás

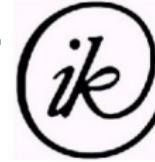
## Mesterséges Neuronhálók 2015 ősz

Milacski Zoltán Ádám

[srph25@gmail.com](mailto:srph25@gmail.com)  
<http://milacski.web.elte.hu>

ELTE Informatika Doktori Iskola

2015-09-21



# Deep Learning anyagok

## Szoftverek:

- NVIDIA Digits (web, NVIDIA): <http://developer.nvidia.com/digits>
- Caffe (command line/Python/MATLAB, UC Berkeley):  
<http://caffe.berkeleyvision.org/>
- torch (Lua/Python/MATLAB, NYU): <http://torch.ch/>
- theano (Python, Université de Montréal):  
<http://deeplearning.net/software/theano/>

## Ingyenes online kurzusok:

- NVIDIA Deep Learning Course (Jon Barker, NVIDIA):  
<http://developer.nvidia.com/deep-learning-courses>
- Neural Networks for Machine Learning (Geoffrey Hinton, University of Toronto):  
<http://www.coursera.org/course/neuralnets>
- Convolutional Neural Networks for Visual Recognition (Andrej Karpathy, Stanford):  
<http://cs231n.github.io/>
- Machine Learning (Andrew Ng, Stanford):  
<http://www.coursera.org/learn/machine-learning>

## Tudományos cikkek:

- Yann LeCun, Joshua Bengio, Geoffrey Hinton, Ilya Sutskever, Christian Szegedy, Alex Krizhevsky, Andrew Ng, Quoc Le, Vincent Vanhoucke, Diederik Kingma és még sokan mások... <http://scholar.google.com>

# Deep Learning 1.

- Legnépszerűbb MI megközelítés napjainkban: aktív kutatási terület, legnagyobb cégek használják adatalemzésre (Google, Facebook, Microsoft, Netflix, Baidu, IBM).
- Sok sikeres alkalmazás: rekordokat döntöget (ImageNet: 25% → 5% hiba)



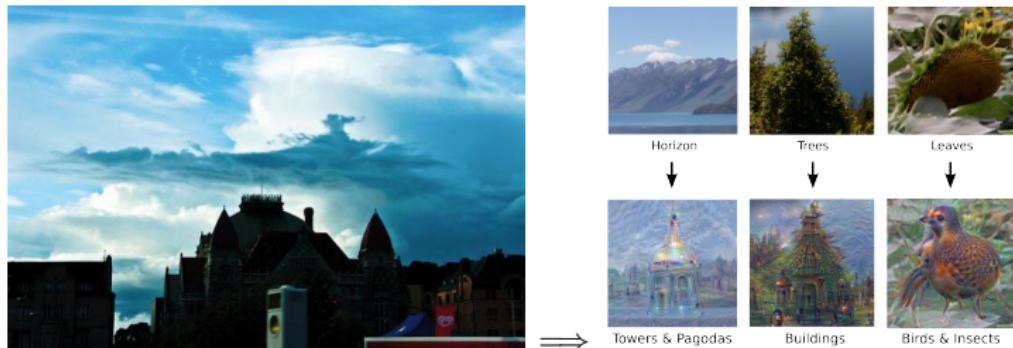
He was elected President during the Revolutionary War and forgave Opus Paul at Rome. The regime of his crew of England, is now Arab women's icons in and the demons that use something between the characters' sisters in lower coil trains were always operated on the line of the **ephemeral** street, respectively, the graphic or other facility for deformation of a given proportion of large segments at RTUS). The B every chord was a "strongly cold internal palette pour even the white blade."

de még spam-szűrés, arcfelismerés, Xbox Kinect, Q-learning...

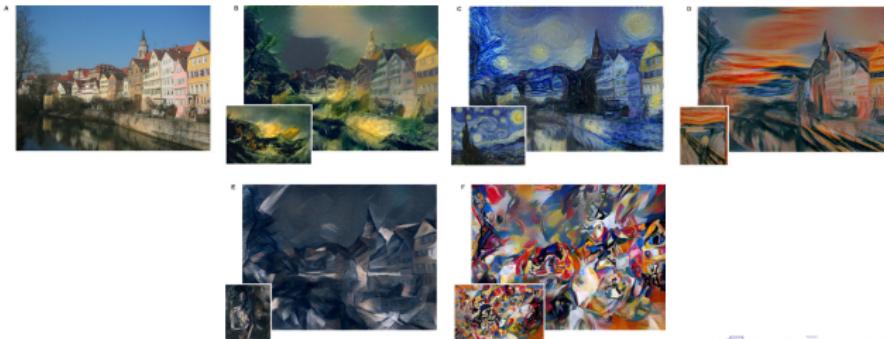
- Zavarba ejtően párhuzamosítható feladat: mátrix-vektor műveletek, GPU gyorsítás (olcsó és takarékos szuperszámítógép).
- Egyszeri lassú betanítás, utána gyors predikció (betanított modell hatékonyan telepíthető beágyazott rendszerekre, mobiltelefonokra).
- Az elmúlt években alapjaiban dőltek meg Machine Learning tézisek:
  - Klasszikus Machine Learning:** feature extrakció és kernel kézzel, tanulunk súlyozást (pl. SVM, HMM, klaszterezés, LDA). Konvex optimalizáció: rengeteg téTEL, pl. garantált globális megoldás.
  - Deep Learning:** vegyük sok mintát és sok paramétert, tanulunk meg a feature-öket IS az adatokból! Erősen nem konvex feladat: nagyon kevés elmeleti garancia, sok mintával és kellő hardverrel mégis jobban általánosít...

## Deep Learning 2.

- Inceptionism: felhők formája és képzelgés, valamire tanítva, másón tesztelve (becsaphatóság kiaknázása)



- Festés stílus másolás: B=Turner, C=van Gogh, D=Munch, E=Picasso, F=Kandinsky



# Deep Learning 3.

- input → output nemlineáris leképezést tanulunk sok (input,output) párból:
  - minden input vektor feature-ökből áll:  $\mathbf{x}_i \in \mathbb{R}^n$ ,  $i = 1, \dots, N$ ,
  - minden output vektor címkékből áll:  $\mathbf{y}_i \in \mathbb{R}^m$ ,  $i = 1, \dots, N$ ,
  - a leképezés paraméterekből áll:  $f_{\theta} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\theta \in \mathbb{R}^p$ ,
  - a leképezés vektoros adatokból címkét próbál jóslani:  $f_{\theta}(\mathbf{x}) \approx \mathbf{y}$ .
  - Cél: előre adott paraméteres leképezés-családból kiválasztani a legjobbat, ami megold valamelyen feladatot (úgy beállítani a  $\theta$  értékeitet, hogy minimalizáljunk egy  $I : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^+$  hibát a jóslott és a tényleges címkék között).
  - Osztályozás:  $\mathbf{y} \in \mathbb{Z}^m$ ,
  - Regresszió:  $\mathbf{y} \in \mathbb{R}^m$ ,
  - Rekonstrukció:  $\mathbf{y} = \mathbf{x}$ ,  $f_{\theta} \neq \text{id}$ .

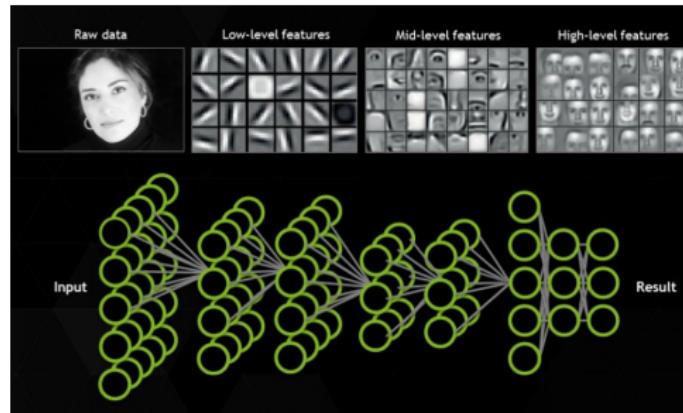
Input	Output
Pixels: 	"lion"
Audio: 	"see at tuhl res taur aun ts"
<query, doc>	P(click on doc)
"Hello, how are you?"	"Bonjour, comment allez-vous?"
Pixels: 	"A close up of a small child holding a stuffed animal"

- Matematikailag:  $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N \implies \min_{\theta} \frac{1}{N} \sum_{i=1}^N I(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$ .
- Túltanulás kikerülendő: még nem látott ( $\mathbf{x}, \mathbf{y}$ ) párokra is kicsi legyen az  $I(f_{\theta}(\mathbf{x}), \mathbf{y})$  hiba.
- Aktivációs függvény:  $g(z) = \max(0, z)$  ReLU függvény. 
- Neuron:  $f_{\mathbf{w}, b}(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x} + b) = g(\sum_{i=1}^n w_i x_i + b)$ ,
- Neuronréteg:  $f_{\mathbf{W}, \mathbf{b}}(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$ ,
- Neuronháló:  $f_{\mathbf{W}_1, \dots, \mathbf{W}_k, \mathbf{b}_1, \dots, \mathbf{b}_k}(\mathbf{x}) = g_k(\mathbf{W}_k \cdots g_2(\mathbf{W}_2 g_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \cdots + \mathbf{b}_k)$ .

## Deep Learning 4.

Neuronháló:  $f_{\mathbf{W}_1, \dots, \mathbf{W}_k, \mathbf{b}_1, \dots, \mathbf{b}_k}(\mathbf{x}) = g_k(\mathbf{W}_k \cdots g_2(\mathbf{W}_2 g_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \cdots + \mathbf{b}_k)$ .

A feature-ök hierarchiába rendeződnek: a mélyebb rétegek egyre magasabb szintű feature-öket tartalmaznak, és minden az előző réteg feature-eit kombinálják össze.



Előnyök:

- Nem kell előzetesen manuálisan feature-öket létrehozni.
- Általános architektúra: sokféle feladatra és adattípusra jó.
- Sok mintával ( $N \approx 100000$ ), GPU-val (NVIDIA CUDA/cuBLAS), új módszerekkel (pl. ReLU, Adam, pretraining, dropout) jól működik.
- Transfer Learning: más által nagy adatbázison betanított háló letölthető és adaptálható saját kicsi adatbázishoz.
- Geoffrey Hinton: "nagyjából így működik az emberi agy, tökéletlen modell is lehet hasznos és tanulságos".

# Optimalizáció

Hogyan találhatunk jó  $\theta = \{W_1, \dots, W_k, b_1, \dots, b_k\}$  paramétereket?

Gradiens-módszerrel (iteratívan)!

- Hiba:  $I(f_\theta(\mathbf{x}), \mathbf{y})$

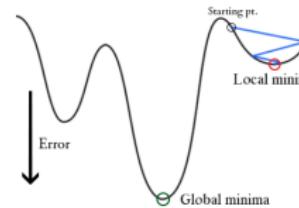
- kettes norma közelítés:  $\frac{1}{2} \|\mathbf{y} - f_\theta(\mathbf{x})\|_2^2$ ,
- egyes norma közelítés:  $\|\mathbf{y} - f_\theta(\mathbf{x})\|_1$ .

- Összhiba:

$$L(\theta) = \left[ \frac{1}{N} \sum_{i=1}^N I(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \right].$$

- Gradiens-módszer: a negatív gradiens  $(-\frac{\partial L(\theta)}{\partial \theta})$  mindenkorban a lokális minimum felé mutat, lépjünk egy kicsit ebbe az irányba!

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\partial L(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}$$



2 kérdés:

- $x_i$  inputok skálázása?
- $\frac{\partial L(\theta)}{\partial \theta}$  gradiens kiszámítása? (véges differenciával való numerikus közelítés pontatlan)
- $\alpha_t$  lépésköz beállítása? (ne lépjünk se túl kicsit, se nem túl nagyon)

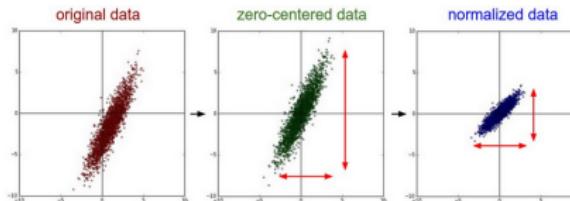
# Gradiens-módszer 1.

Gradiens-módszer:  $\theta_{t+1} = \theta_t - \alpha_t \frac{\partial L(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}$ ,  $\theta = \{W_1, \dots, W_k, b_1, \dots, b_k\}$ .

- $x_i$  inputok skálázása: ha a feature-ök különböző skálákon mozognak, de azonos súllyal szeretnénk figyelembe venni őket (szinte minden igaz), feature-önként standard normalizáljuk az inputokat:

$$\mu = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i, \quad \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \mu)^2},$$

$$\mathbf{x}_i \leftrightarrow \frac{\mathbf{x}_i - \mu}{\sigma}.$$



## Gradiens-módszer 2.

Gradiens-módszer:  $\theta_{t+1} = \theta_t - \alpha_t \frac{\partial L(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}$ ,  $\theta = \{W_1, \dots, W_k, b_1, \dots, b_k\}$ .

- $\frac{\partial L(\theta)}{\partial \theta}$  gradiens kiszámítása: láncszabállyal, dinamikus programozással.

Legyen  $X = [x_1, \dots, x_N]$ ,  $Y = [y_1, \dots, y_N]$ ,  $g'_j(h) = \frac{\partial g(h)}{\partial h}$ ,  $l'(z, y) = \frac{\partial l(z, y)}{\partial z}$ .

```
function PROP(X, Y, W1, ..., Wk, b1, ..., bk, g1, ..., gk, g'1, ..., g'k, l, l')
```

// ForwardPropagation

$Z_0 = X$

for  $j = 1, \dots, k$  do

$H_j = W_j Z_{j-1} + [b_j, \dots, b_j]$

$Z_j = g_j(H_j)$

end for

$L = \frac{1}{N} \sum_{i=1}^N l(Z_{k,\cdot,i}, Y_{\cdot,i})$

// BackPropagation

$\partial Z_k = \frac{1}{N} [l'(Z_{k,\cdot,1}, Y_{\cdot,1}), \dots, l'(Z_{k,\cdot,N}, Y_{\cdot,N})]$

for  $j = k, \dots, 1$  do

$\partial H_j = g'_j(H_j) \circ \partial Z_j$  // itt  $\circ$  az elemenkénti szorzás

$\partial Z_{j-1} = W_j^T \partial H_j$

$\partial b_j = \sum_{i=1}^N \partial H_{j,\cdot,i}$

$\partial W_j = \partial H_j Z_{j-1}^T$

end for

return  $Z_k, L, \partial W_1, \dots, \partial W_k, \partial b_1, \dots, \partial b_k$

end function

## Gradiens-módszer 3.

Gradiens-módszer:  $\theta_{t+1} = \theta_t - \alpha_t \frac{\partial L(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}$ ,  $\theta = \{W_1, \dots, W_k, b_1, \dots, b_k\}$ .

- $\alpha_t$  lépésköz beállítása: Adam algoritmus.

```

function ADAM( $\alpha, \beta_1, \beta_2, \varepsilon, X, Y, \theta_0, g_1, \dots, g_k, g'_1, \dots, g'_k, l, l'$ )
     $M_0 = 0, V_0 = 0$ 
    for  $t = 0, 1, \dots, T - 1$  do
         $[Z_k, L, \partial\theta] = \text{PROP}(X, Y, \theta_t, g_1, \dots, g_k, g'_1, \dots, g'_k, l, l')$ 
         $M_{t+1} = \beta_1 M_t + (1 - \beta_1) \partial\theta$  // Gradiens mozgóátlaga
         $V_{t+1} = \beta_2 V_t + (1 - \beta_2) \partial\theta^2$  // Gradiens négyzet mozgóátlaga
         $\theta_{t+1} = \theta_t - \alpha \frac{\frac{M_{t+1}}{1 - \beta_1^{t+1}}}{\sqrt{\frac{V_{t+1}}{1 - \beta_2^{t+1}} + \varepsilon}}$  // Paraméterek frissítése
    end for
    return  $\theta_T$ 
end function

```

Tipikus beállítás:  $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ .

- $\theta_0$  kezdőpont: ReLU függvény esetén  $W_j \sim \mathcal{N}(0, 0.001^2), b_j = 0.01$ .
- Leállási feltétel:  $\|\partial\theta\|_2 < 10^{-4}$  (közelítőleg lokális szélsőérték).
- Minibatch: minden lépéshoz a minták egy részhalmazával számolunk (csökkentett memóriaigény, több kicsi iteráció, gyorsabb konvergencia, mintákban párhuzamosítható GPU-val). Ekkor célszerű a negatív gradiens irányába lépni (momentum).

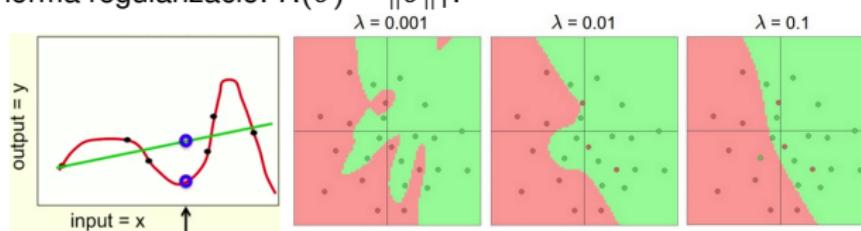
# Regularizáció

Regularizáció:  $R(\theta)$ , plusz költség tag túltanulás ellen (nem engedi, hogy  $\theta$  elemei nagyok legyenek, egyszerűsíti a modellt, jobban általánosít).

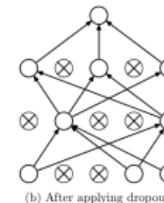
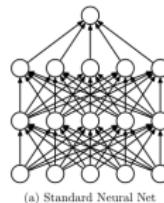
Összhiba regularizációval ( $\lambda > 0$ : két költség trade-off-ja):

$$L(\theta) = \left[ \frac{1}{N} \sum_{i=1}^N l(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \right] + \lambda R(\theta)$$

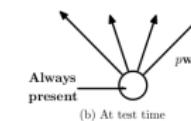
- Kettes norma regularizáció:  $R(\theta) = \frac{1}{2} \|\theta\|_2^2$ ,
- Egyes norma regularizáció:  $R(\theta) = \|\theta\|_1$ .



- Dropout: exponenciálisan sok háló összekombinálása, tanításkor  $p = 0.5$  valószínűséggel aktív minden neuron (input rétegre  $p = 0.8$ ), teszteléskor  $p$ -vel szorzandó minden aktivitás. Jó implementáció: inverted dropout.

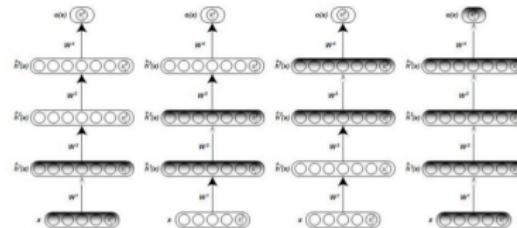


Present with probability  $p$   
(a) At training time



# Greedy Layer-wise Pretraining

Előtanítás: címkék nélküli (felügyeletlen) rekonstrukció, mohó módon, rétegenként. Utána szokásos felügyelt (finom)hangolás címkékkel.



Lassú, de gyakran jobb  $\theta_0$  kezdőpontot ad.

Kis adatbázisoknál segít a legtöbbet! Esetleg ha sok címke nélküli tanítópéldánk van...

Manapság csökken a jelentősége, de a jövőben még nagyobb hálók esetén újra szükség lehet rá...

# Hiperparaméterek

Egy mély hálónak sok hiperparamétere van: rétegek száma  $k$ , rétegek méretei, regularizáció  $\lambda$ , Adam  $\alpha, \beta_1, \beta_2, \varepsilon$ , dropout  $p$ , de akár még aktivációs és költségfüggvények is...

Hogyan állítsuk be ezeket?

- Kereszt-validáció: bontsuk az adatbázist 80% tanító + 10% validáció + 10% teszt részekre. Tanítón sok kombinációt próbálunk ki, validáción mérjük ki a legjobbat. A legjobb kombinációval tanítsunk újra tanító+validáció és teszten használjuk ezt! Majd válasszuk újra a halmazokat, és átlagoljuk az eredményt.
- Bayesian optimization: Megerősítéses Tanulás (RL) szerű iteratív eljárás, próba-szerencse alapon oldja fel az exploráció-exploitációs dilemmát. Gauss-folyamatokat használ. Megtanulja a helyes skálázást a változókban. Sokkal jobb, mint a manuális próbálgatás (Pontosan emlézik és értékel ki sok kombinációt).
  - Spearmint (Python/MATLAB): <http://github.com/HIPS/Spearmint>
  - Hyperopt (Python): <http://github.com/hyperopt/hyperopt>
  - BayesOpt (C/C++/Python/MATLAB): <http://rmcantin.bitbucket.org/html/>

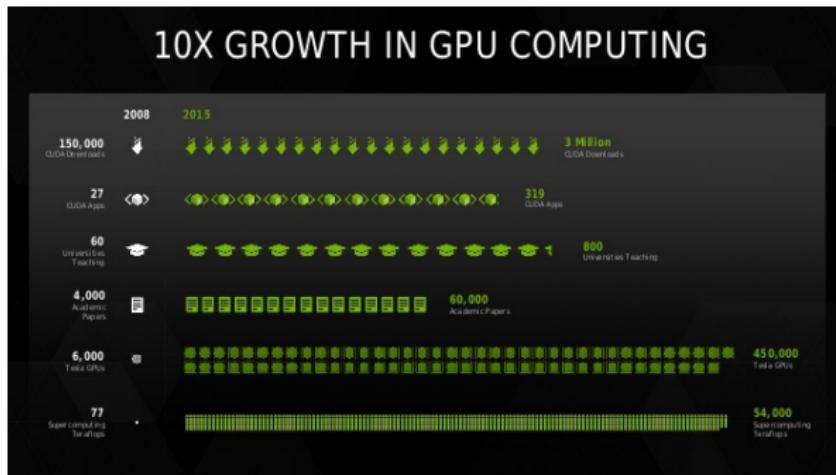
# Speciális neuronhálók

Az alapvető neuronhálós tudást lefedik az előző diák, speciális hálók azonban minden igényelnek további trükköket.

- Feedforward háló (FNN): ezt láttuk részletesen az előző diákon.
- Konvolúciós háló (CNN): mátrixszorzások helyett konvolúciók (kevesebb változó, kisebb feladatméréte).
- Rekurrens háló (RNN): idősorok (dinamika).
- Autoencoder: rekonstrukció, feature tanulás, előtanítás, klaszterezés.
- Hopfield háló: tartalom-címzett memória, robosztus, bináris aktivitások, irányítatlan élek a gráfban. Ma már nem használják, történelmi okokból fontos.
- Boltzmann háló (RBM): ma már nem használják, történelmi okokból fontos.

# A terület jövője

- 2016: NVIDIA Pascal architektúra
  - Újabb nagy ugrást jelenthet... Most még nem késő belekezdeni.
  - 32 GB RAM: nagyobb minibatch-ek, kevesebb másolási művelet.
  - NVLink interfész: PCI Express-hez képest sokkal nagyobb sávszélesség.
  - Vegyes-pontosságú számítások: számolás double-ben, tárolás float-ban.
- CUDA Toolkit: folyamatos szoftveres fejlődés (7.5: ritka mátrixok)



- Nehéz megmondani, hogy mi lesz a neuronhálók jövője, de:
  - a meglévő módszereket tovább fogják tökéletesíteni,
  - adat és hardverkapacitás is csak egyre több lesz,
  - egyre több helyen fognak megjelenni a hétköznapokban,
  - sok szép meglepő eredményt fognak produkálni az elővetkezendő években...

# Köszönöm a figyelmet!

## Köszönöm a figyelmet!

Ez a diasor Jon Barker, Geoffrey Hinton, Andrej Karpathy anyagai; Diederik Kingma, Leon Gatys, Nitish Srivastava cikkei, Ilya Sutskever disszertációja és Yifei Teng, Alexander Mordvintsev (Google Research) blogjai alapján készült. Az algoritmusok és az ábrák a fenti cégek és szerzők szellemi tulajdonát képezik.