

Gyires-Tóth Bálint

Deep Learning a gyakorlatban Python és LUA alapon Tanítás: alap tippek és trükkök

Jogi nyilatkozat

Jelen előadás diái a „*Deep Learning a gyakorlatban Python és LUA alapon*” című tantárgyhoz készültek és letölthetők a <http://smartlab.tmit.bme.hu> honlapról.

A diák nem helyettesítik az előadáson való részvételt, csupán emlékeztetőül szolgálnak.

Az előadás diái a szerzői jog védelme alatt állnak. Az előadás diáinak vagy bármilyen részének újra felhasználása, terjesztése, megjelenítése csak a szerző írásbeli beleegyezése esetén megengedett. Ez alól kivétel, mely diákon külső forrás külön fel van tüntetve.

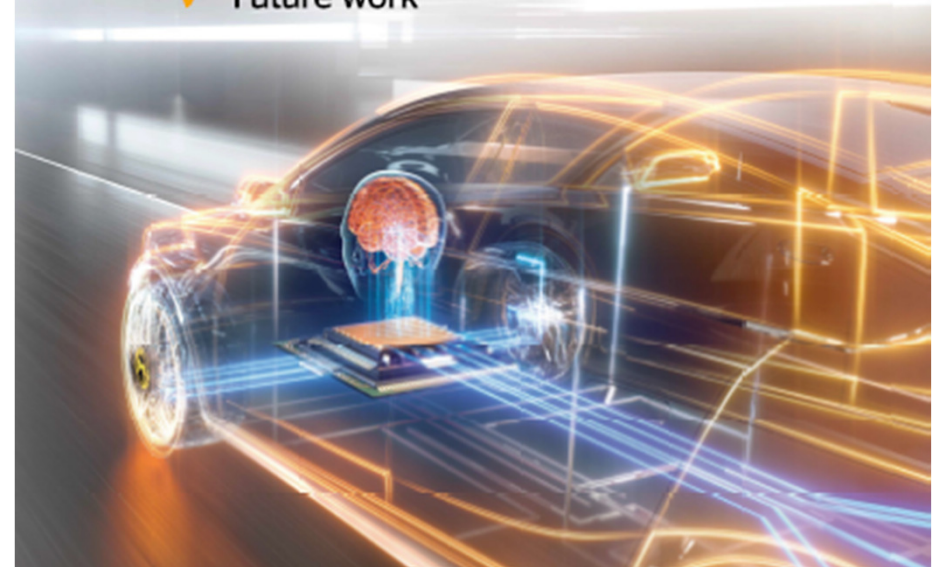


Deep Learning Híradó

Hírek az elmúlt 168 órából



- ✓ AI in Automotive
- ✓ Research and Development
- ✓ For BsC, Msc and PhD programmes
- ✓ Scholarship
- ✓ Future work



<http://bit.ly/piaday2020>

Regisztráció:

www.nvidia.com/gtc

Referencia kód: NVMDIERINGER



Előzetes dátum: október 23.

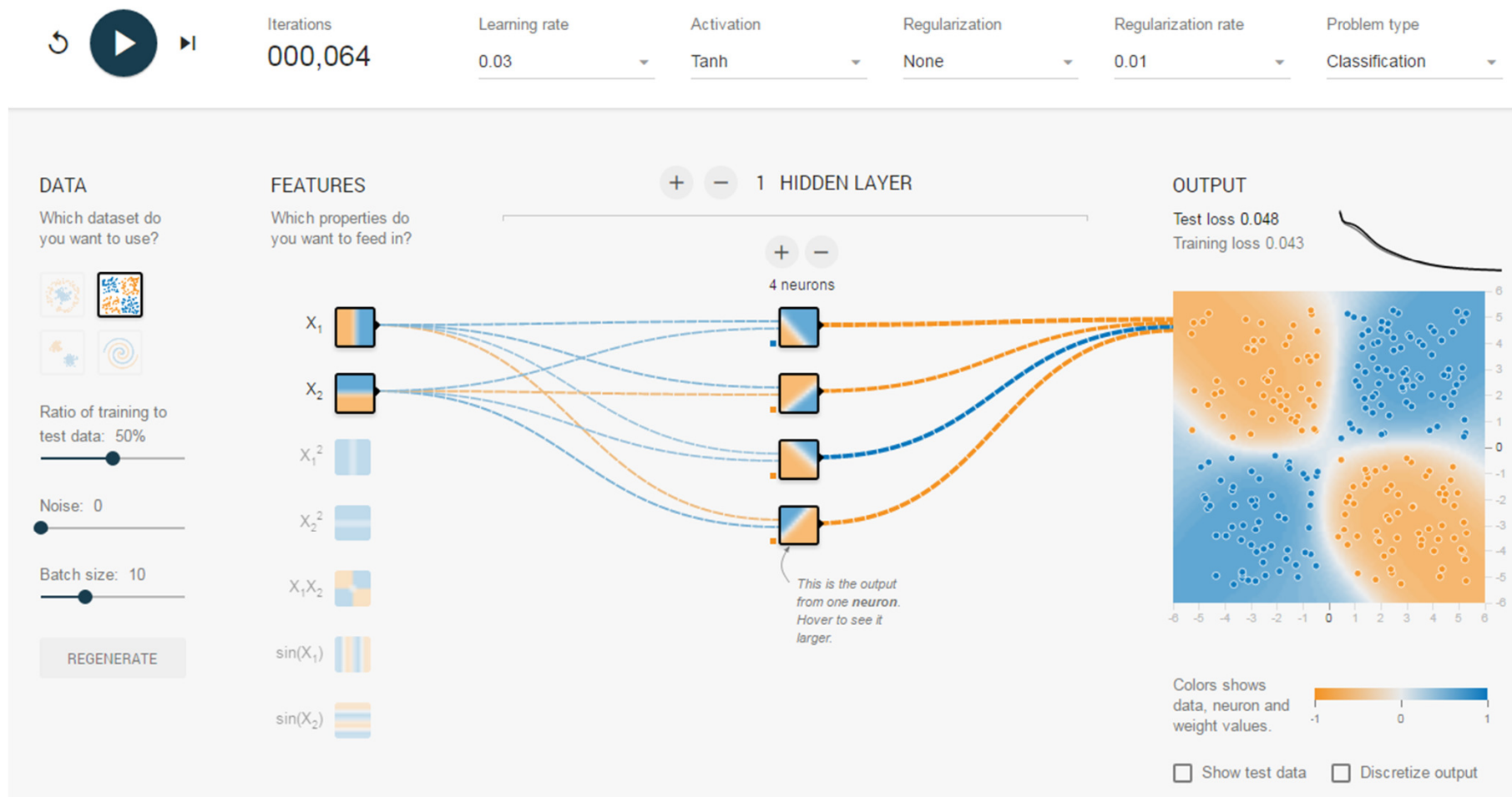


DEEP
LEARNING
INSTITUTE

TEACHING YOU
TO SOLVE PROBLEMS
WITH DEEP LEARNING

Vizualizáció

<http://playground.tensorflow.org/>



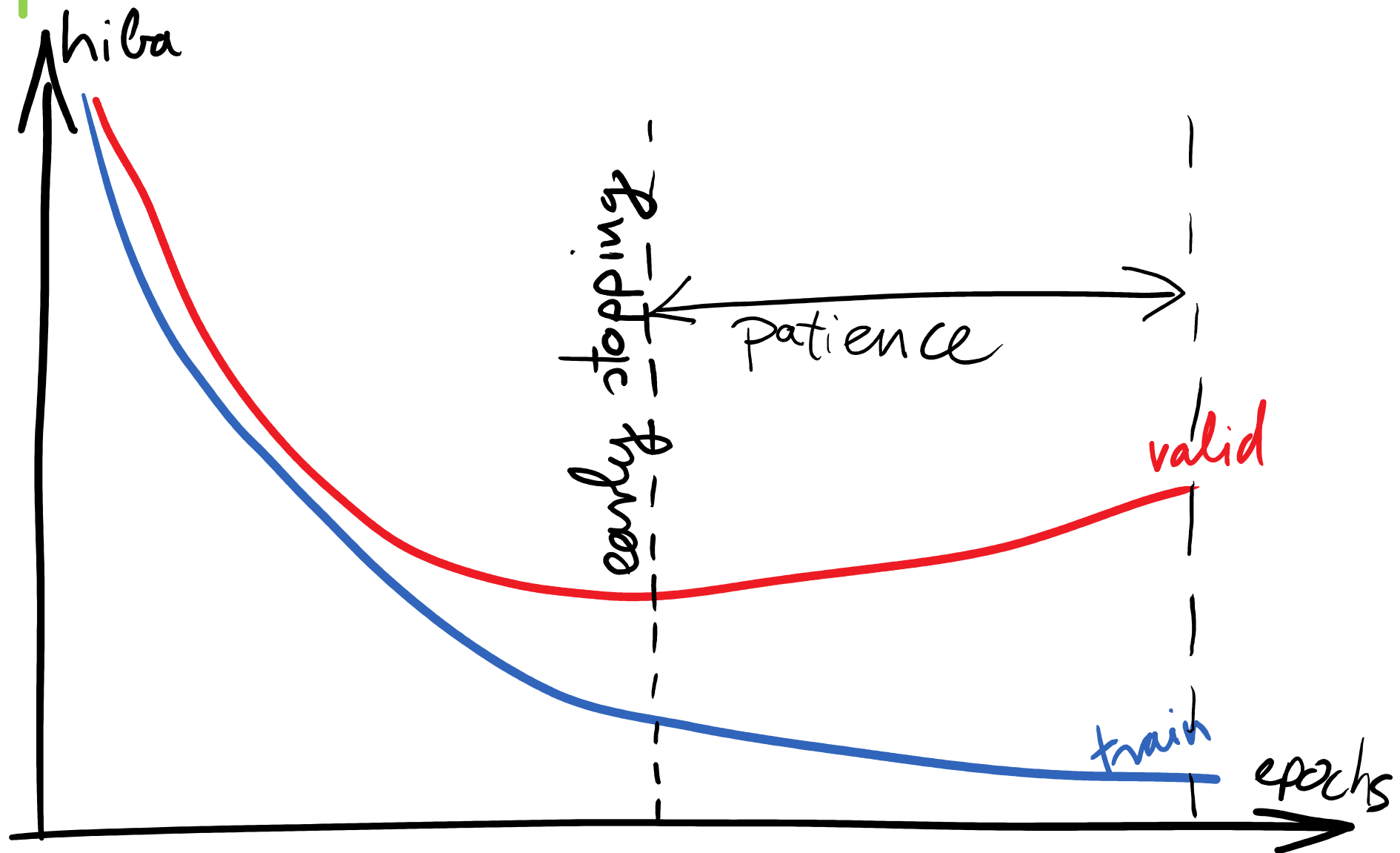
Alapvető problémák

- Hatalmas hiperparaméter tér
- Nagyon lassan vagy nem konvergál
 - Adatok standardizálása és min-max skálázás
 - Mini-batch learning
 - Más költségfüggvény és optimizációs algoritmus
 - Tanítóadatok összekeverése
- Túltanulás – train-validation-test
 - Több adat
 - Early stoping
 - Regularizáció: L1, L2 (Weight decay), Dropout, stb.

Tanító-, validációs- és tesztadatok

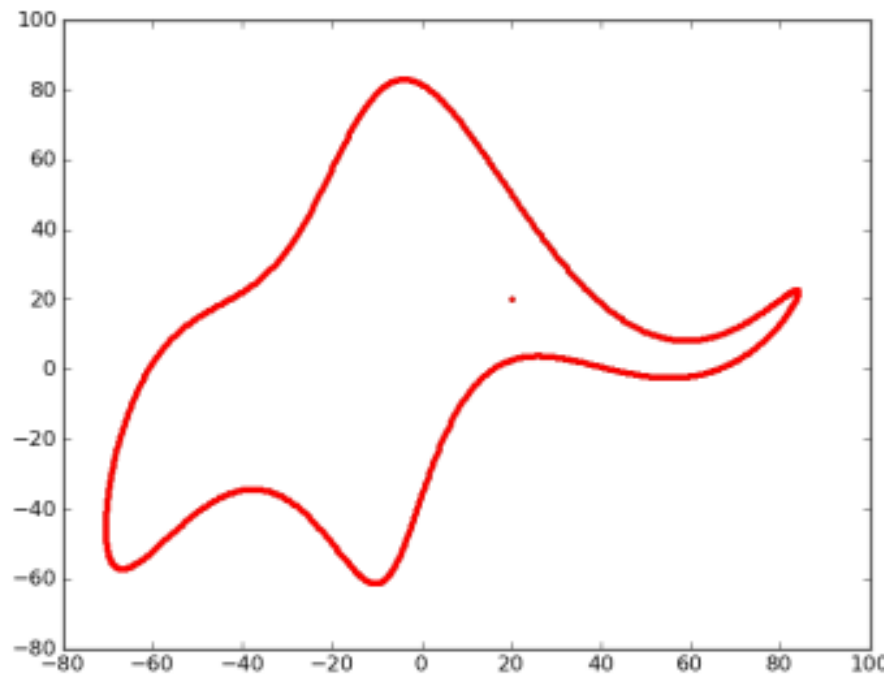
- Adatok szétszedés 3 részre:
 - Tanító, validációs és tesztadatok.
 - Saját függvényrel vagy az `sklearn.cross_validation.train_test_split` függvényt 2x meghívni.
- Adatok összekeverése segíti a tanulást:
 - Két egymást követő minta minél távolabb legyen egymástól (pl. külön osztályból).
 - Legyen nagy a hiba a többi tanítómintához képest.
 - Véletlenszerű.
- *Early stoping after „patience” epochs*
 - Ha a validációs hiba adott lépésig nem csökken, akkor leállítjuk a tanítást és visszatöltjük a legjobb modellt.

Tanító-validációs-teszt adatok I.



Tanító-validációs-teszt adatok II.

- Túltanítás
- Neumann János: *Négy paraméterrel még egy elefántot is le lehet írni, egy ötödikkal pedig el lehet érni, hogy az ormányát is csóválja.*



Forráskód: <http://www.johndcook.com/blog/2011/06/21/how-to-fit-an-elephant/>

L1 és L2 regularizáció

L1 regularizáció

$$C = C_0 + \lambda_1 \cdot \sum_w |w|$$

$$W^{(i)}(t+1) = W^{(i)}(t) - \mu \frac{\partial C}{\partial W^{(i)}(t)} - \mu \lambda_1 \cdot \text{sgn}(W^{(i)}(t))$$

L2 regularizáció (weight decay)

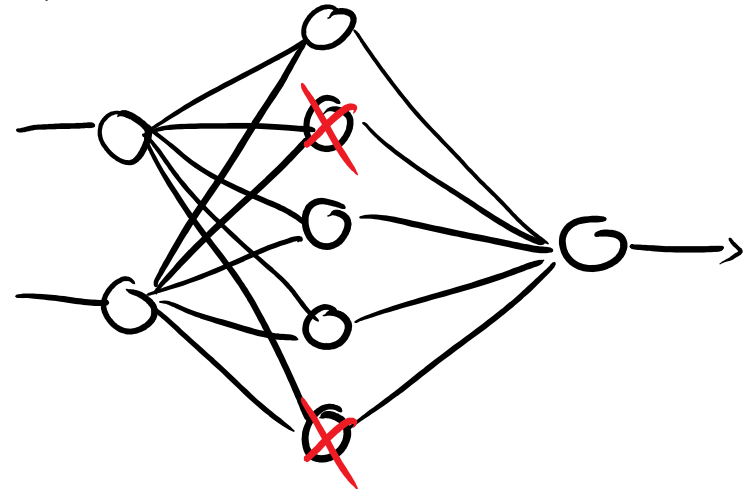
$$C = C_0 + \frac{1}{2} \lambda_2 \sum_w w^2$$

$$W^{(i)}(t+1) = W^{(i)}(t) - \mu \frac{\partial C}{\partial W^{(i)}(t)} - \mu \lambda_2 \cdot W^{(i)}(t)$$

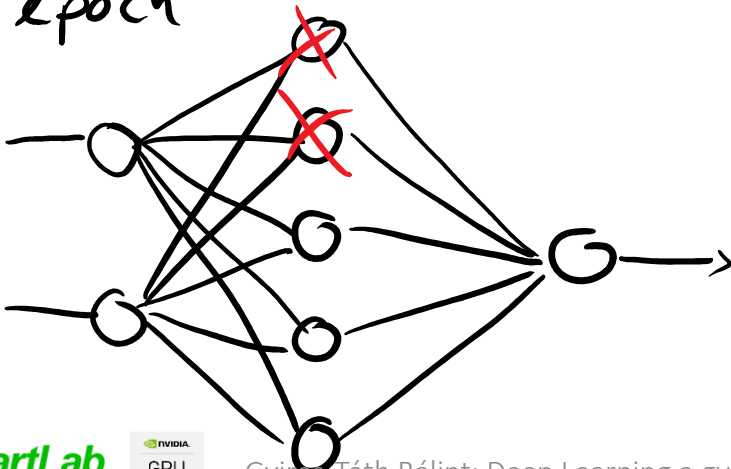
Dropout I.

Hinton prof. egyik „AHA” pillanata: banki ügyintézők.

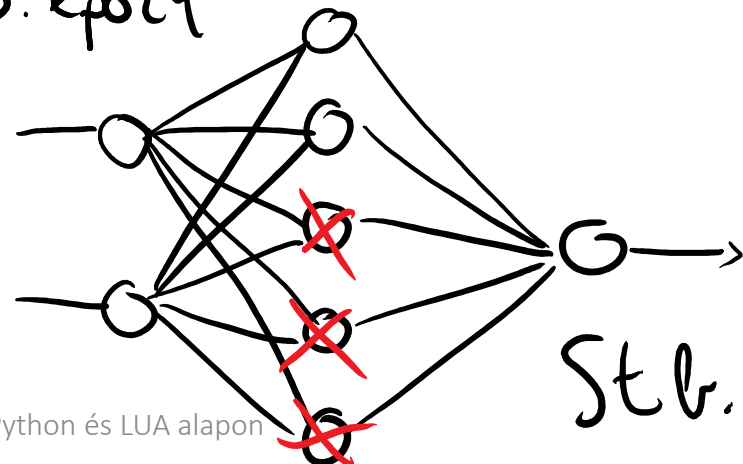
1. epoch



2. epoch



3. epoch



Dropout II.

Az egyes neuronok eldobásának valószínűsége általában 30-50%.

Miért működhet?

Olyan, mintha több hálót tanítanánk, és azok aggregált becslését átlagolnánk.

Így kerül el a túltanítást.

<https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf>

Standardizálás I.

Eltolás kivonása és szórással való osztás.

Gyakorlatban: várható érték kivonása és szórással való osztás.

Csak a tanító adatokra!!!

Max>1

Feature szerint

`sklearn.preprocessing.StandardScaler` vagy:

$$X = (x - \text{mean}(X)) / \text{std}(x)$$

Standardizálás II.

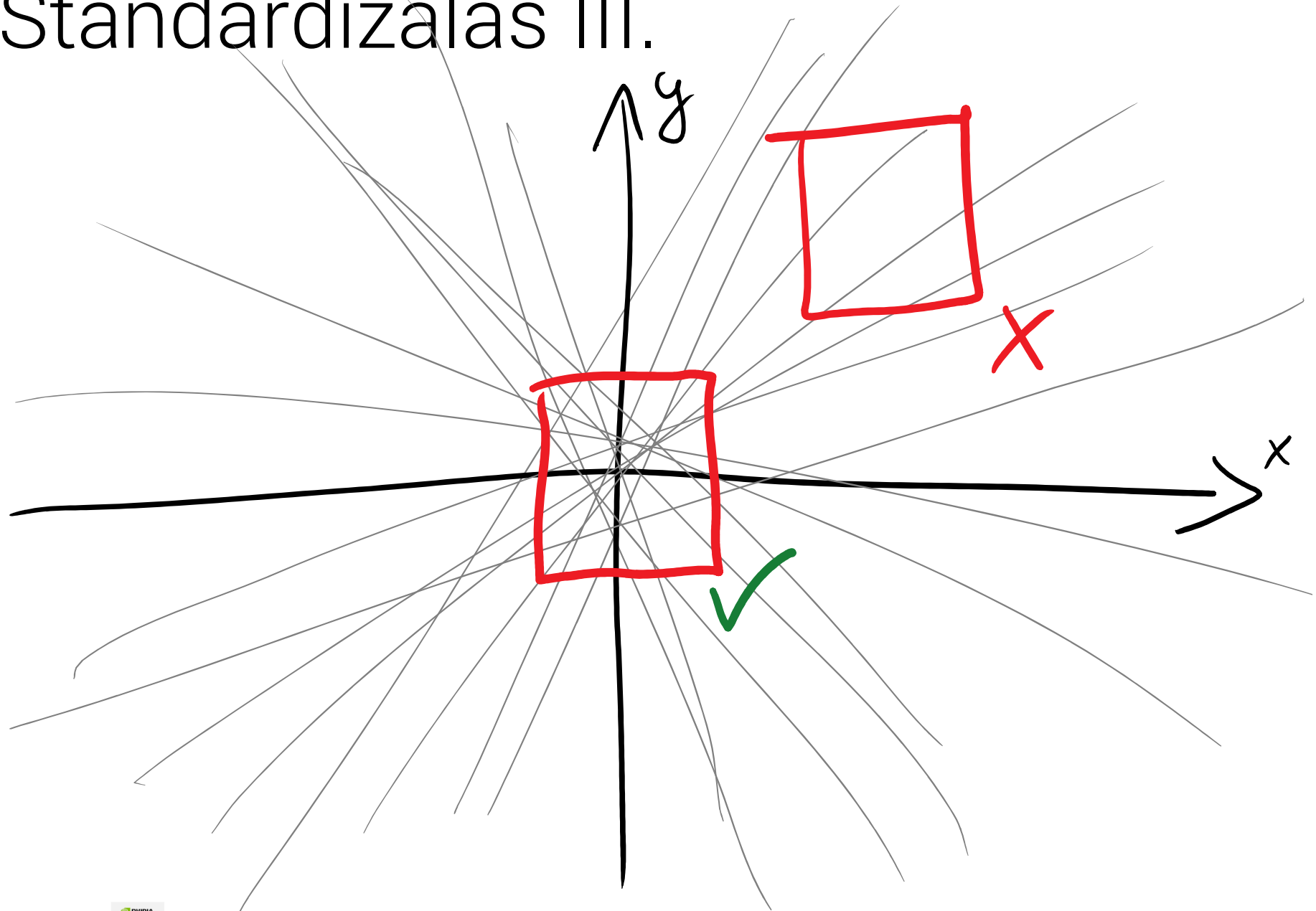
Miért van rá szükség:

- Súlyok inicializálása: azonos nagyságrendben legyen, egyébként szaturáció lehetséges.
- A neuronokhoz tartozó hipersíkok:
 - Irányát a súlyok határozzák meg
 - Eltolást a bias határozza meg

Alacsony bias inicializáláskor → jó, ha az adat is közel van az origóhoz.

Lehetne a súlyokat és a bias-t is a bemenethez illeszteni, de sokkal bonyolultabb.

Standardizálás III.



Min-max skálázás

Adatok átskálázása

Kimeneti aktivációs függvény sigmoid vagy tanh
„Harmóniában” a súlyokkal és a bemenettel

Probléma: *outlier*-ek.

Csak a tanító adatokra!!!

`sklearn.preprocessing.StandardScaler` vagy:

$$\begin{aligned} y_{std} &= (y - \min y) / (\max y - \min y) \\ y_{scaled} &= y_{std} \cdot (\max - \min) + \min \end{aligned} \quad \left| \begin{array}{l} \max = 1 \\ \min = 0 \end{array} \right.$$



Költséggüggvények és optimalizáció

- Leggyakrabban használt költséggüggvények
 - Regresszió: négyzetes hiba (MSE, mean squared error)
 - Osztályozás: keresztentropia
 - Több költséggüggvény kombinálása a kimeneten
- Leggyakrabban használt optimalizációs algoritmusok
 - SGD (Stochastic Gradient Descent), momentum
 - ~~L-BFGS~~
 - ~~AdaDelta~~
 - ~~AdaGrad~~
 - RMSProp
 - ADAM, AdamW

Momentum módszer

Segít elkerülni a súly „fluktuációt” és kijutni a lokális minimumokból.

0.5-0.99 közötti érték.

“Sebessége” lesz a súlyfrissítésnek, amerre a gradiens halad.

$$\Delta w^{(i)}(t) = -\mu \frac{\partial C}{\partial w^{(i)}(t)} - \alpha \cdot \Delta w^{(i)}(t-1)$$

$$w^{(i)}(t+1) = w^{(i)}(t) + \Delta w^{(i)}(t)$$

Újabb változata: Nesterov momentum.

Kahoot!

Névnek a NEPTUN
kódodat add meg!

<https://kahoot.it/>

ADAGRAD

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul), 2121-2159.

$$\alpha_t = \alpha_{t-1} + \left\{ \frac{\partial C}{\partial w_t^{(n)}} \right\}^2$$

$$w_{t+1}^{(n)} = w_t^{(n)} - \frac{\mu \cdot \frac{\partial C}{\partial w_t^{(n)}}}{\sqrt{\alpha_t} + \epsilon} \quad \epsilon = 10^{-4} \dots 10^{-8}$$

. √α_t + ε űj elem

Előny:

- nagyobb gradiens adott súlynál → a tanulási ráta csökken.
- kisebb gradiens adott súlynál → a tanulási ráta nő.

Hátrány:

- Monoton csökken a tanulási ráta, túl hamar leállhat.

RMSProp

G.E. Hinton, Lecture 6 / 29-es dia

http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

$$\alpha_t = \lambda \cdot \alpha_{t-1} + (1-\lambda) \cdot \left\{ \frac{\partial \mathcal{L}}{\partial w_t^{(n)}} \right\}^2$$

$$\lambda = \begin{cases} 0,9 \\ 0,99 \\ 0,999 \end{cases}$$

$$W_{t+1}^{(n)} = W_t^{(n)} - \frac{\mu \cdot \frac{\partial \mathcal{L}}{\partial w_t^{(n)}}}{\sqrt{\alpha_t} + \varepsilon}$$

$$\varepsilon = 10^{-4} \dots 10^{-8}$$

ÚJ ELEM

Az ADAGRAD „tovább gondolt” változata: az *accumulator* a gradiensek súlyozott „mozgóátlaga”.

A tanulási ráta nem csökken monoton módon nem válik monotonná.

ADAM

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \frac{\partial \mathcal{L}}{\partial w_t^{(n)}} & \beta_1 &= 0.9 \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \left(\frac{\partial \mathcal{L}}{\partial w_t^{(n)}} \right)^2 & \beta_2 &= 0.999 \\ w_{t+1}^{(n)} &= w_t^{(n)} - \frac{\mu \cdot m_t}{\sqrt{v_t} + \epsilon} & \epsilon &= 10^{-8} \end{aligned}$$

RMSProp + momentum.

Előny: a gradiens simítva van („nem ugrálhat annyira”).

Ez és az SGD a leggyakrabban használt algoritmusok.

| Algoritmusok összehasonlítása

Online:

- <http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html>
- <https://www.deeplearning.ai/ai-notes/optimization/>

Adam with decoupled weight decay

Loshchilov, Ilya, and Frank Hutter. "Decoupled weight decay regularization." arXiv preprint arXiv:1711.05101 (2017).

- *Az Adam implementáció minden deep learning keretrendszerben hibás volt*
- *Az L2-regularizáció és a weight decay csak SGD esetén azonos*
- *Adaptív módszerek esetén ezek különböznek*

AdamW

Algorithm 1 SGD with L_2 regularization and SGD with decoupled weight decay (SGDW), both with momentum

- 1: **given** initial learning rate $\alpha \in \mathbb{R}$, momentum factor $\beta_1 \in \mathbb{R}$, weight decay/ L_2 regularization factor $\lambda \in \mathbb{R}$
 - 2: **initialize** time step $t \leftarrow 0$, parameter vector $\theta_{t=0} \in \mathbb{R}^n$, first moment vector $m_{t=0} \leftarrow \mathbf{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
 - 3: **repeat**
 - 4: $t \leftarrow t + 1$
 - 5: $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$ ▷ select batch and return the corresponding gradient
 - 6: $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$
 - 7: $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, be used for warm restarts
 - 8: $m_t \leftarrow \beta_1 m_{t-1} + \eta_t \alpha g_t$
 - 9: $\theta_t \leftarrow \theta_{t-1} - m_t - \eta_t \lambda \theta_{t-1}$
 - 10: **until** *stopping criterion is met*
 - 11: **return** optimized parameters θ_t
-

AdamW

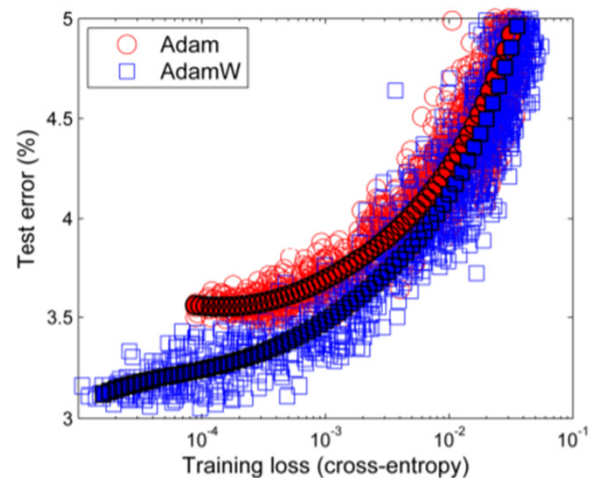
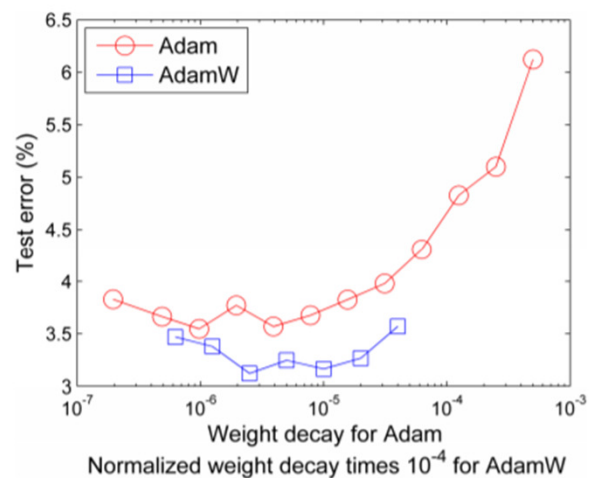
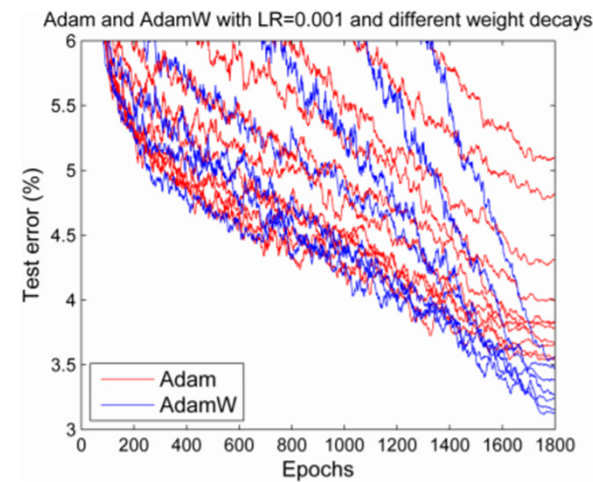
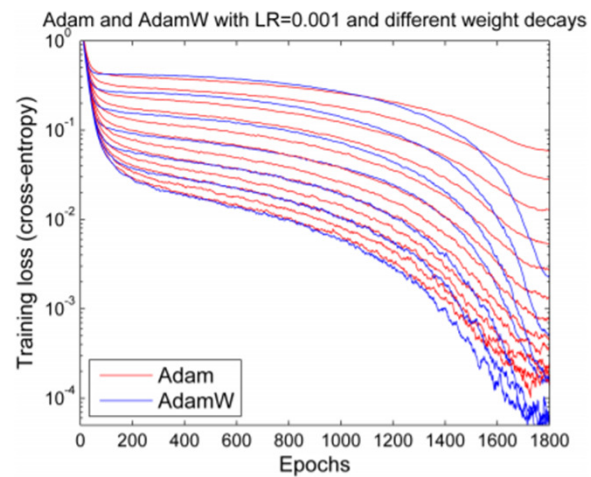
Algorithm 2 Adam with L₂ regularization and Adam with decoupled weight decay (AdamW)

```
1: given  $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $\mathbf{m}_{t=0} \leftarrow \mathbf{0}$ , second moment vector  $\mathbf{v}_{t=0} \leftarrow \mathbf{0}$ , schedule multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$  ▷ select batch and return the corresponding gradient
6:    $\mathbf{g}_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$ 
7:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$  ▷ here and below all operations are element-wise
8:    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$ 
9:    $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$  ▷  $\beta_1$  is taken to the power of  $t$ 
10:   $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$  ▷  $\beta_2$  is taken to the power of  $t$ 
11:   $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$  ▷ can be fixed, decay, or also be used for warm restarts
12:   $\theta_t \leftarrow \theta_{t-1} - \eta_t \left( \alpha \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon) + \lambda \theta_{t-1} \right)$ 
13: until stopping criterion is met
14: return optimized parameters  $\theta_t$ 
```

Helyes és hibás implementáció

✓
$$\tilde{Q}_t \leftarrow \tilde{Q}_{t-1} - \eta_t \frac{\beta_1 m_{t-1} + (1-\beta_1)(\nabla f_t)}{\sqrt{u_t} + \epsilon} + \lambda \tilde{Q}_{t-1}$$

✗
$$\tilde{Q}_t \leftarrow \tilde{Q}_{t-1} - \eta_t \frac{\beta_1 m_{t-1} + (1-\beta_1)(\nabla f_t + \lambda \tilde{Q}_{t-1})}{\sqrt{u_t} + \epsilon}$$



Kis házi feladat 2.

A gyakorlatokon ismertetésre került Numpy alapú neurális hálózat kódjába valósítsd meg a következő funkciókat:

1. Momentum
2. L1 és L2 regularizáció

A forráskód új részeit “# HF2 start eljárás” és “# HF2 end eljárás” (eljárás=momentum, l1reg, l2reg) kommentek közé tedd és kommentbe írd bele, hogy pontosan mit-miért csináltál. **Törekedj a minél rövidebb kódra!**

A kis házi feladatot angol nyelven adjátok be!

Csak Github-ra secret gist-ként feltöltött Jupyter notebook formátumot fogadunk el, amelyben a kódok mellett magyarázat és az eredmények is láthatóak.

Beadási határidő: 2020. október 13. éjfélig

Nagy házi feladat

- Csapatok toborzása Microsoft Teams-en
- 4. hét végéig csapat minden tagjának regisztrációja a <http://smartlab.tmit.bme.hu/oktatas-deep-learning> oldalon.
- BSc, MSc diploma
- PhD téma
- 3 fős csapatok
- 1, 2, 4 fős csapatok
- Téma javaslatok

Nagy házi feladat témaötletek

Ha még nincs meg a végleges téma:

- Duckietown
- Covid-19
- Rapdis.ai alapú machine learning pipeline
- Neuronhálók vizualizálása
- Idősorok modellezése a legújabb architektúrákkal
- Intelligens minibatch válogatás
- Optimizációs algoritmusok fejlesztése
- önfelügyelt tanulás

Ha még nincs meg a végleges csapat: Google Groups

Előzetes dátum: október 23.



DEEP
LEARNING
INSTITUTE

TEACHING YOU
TO SOLVE PROBLEMS
WITH DEEP LEARNING

Regisztráció:

www.nvidia.com/gtc

Referencia kód: NVMDIERINGER

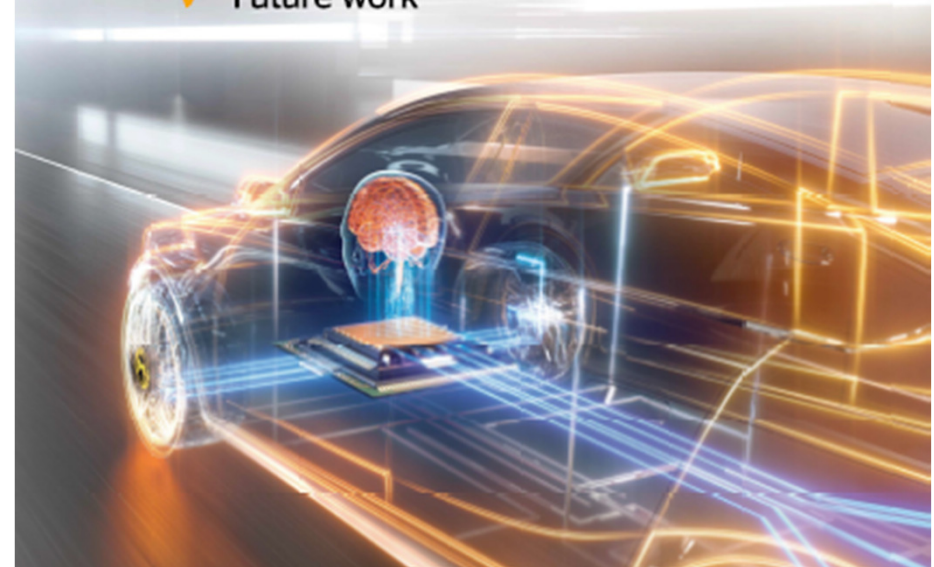


October 5-9

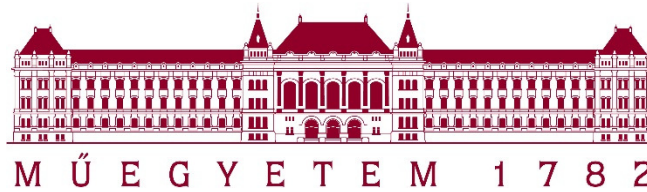




- ✓ AI in Automotive
- ✓ Research and Development
- ✓ For BsC, Msc and PhD programmes
- ✓ Scholarship
- ✓ Future work



<http://bit.ly/piaday2020>



Köszönöm a figyelmet!

`toth.b@tmit.bme.hu`