DeepLearning alapjai

Gyakorlat

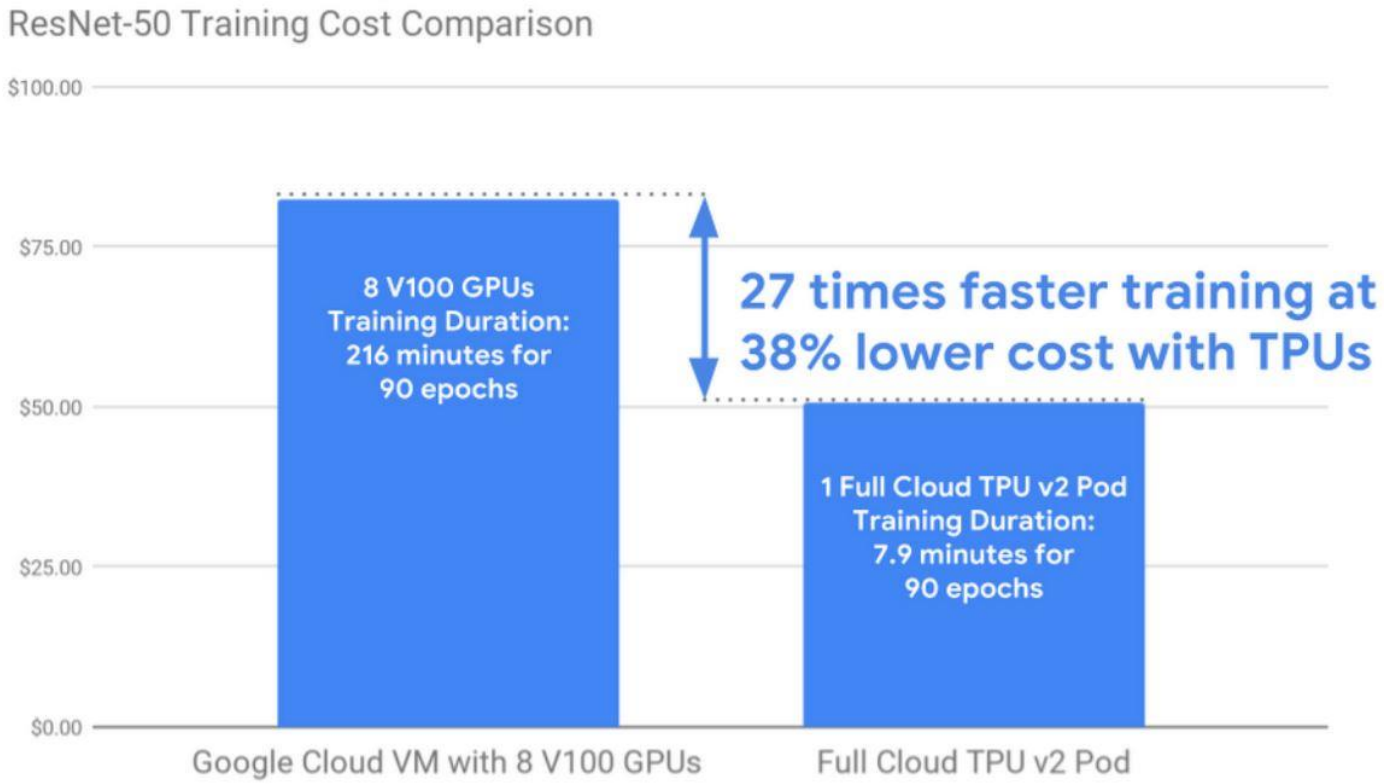
2019. július 12.

Novák Márton
Clementine

GPU, CPU, TPU

- CPU : Central Processing Unit, minden alap otthoni gépben megtalálható, általános műveletek
- GPU: Graphics Processing Unit, dedikált/integrált, drága, grafikai műveletek, **párhuzamos**
- TPU: Tensor Processing Unit, direkt ML műveletek, TensorFlow

TPU célhardware, GPU-ra sok probléma megoldását át lehet ültetni.



https://www.servethehome.com/google-enables-tpu-v2-pod-training-gcp/google-cloud-tpu-v2-pod-v-8x-nvidia-tesla-v100-gpus/

Google Colab

Neuron - > Perceptron -> NN -> DeepNN -> Very Deep CNN -> ?

Tanítás időigényes, majd látni fogjuk… Small scale esetén még a CPU is használható, de nagyobb hálózatoknál bele sem érdemes fogni.

Alternatívák

Otthoni gép: NVIDIA 1080 Ti – 2080 Ti, ~300 ezer – 400ezer.
https://medium.com/the-mission/how-to-build-the-perfect-deep-learning-computer-and-save-thousands-of-dollars-9ec3b2eb4ce2

Cloud: azonnali ktg. olcsóbb, hosszú távon nem biztos + ne felejtsük el leállítani ☺ .
https://medium.com/the-mission/why-building-your-own-deep-learning-computer-is-10x-cheaper-than-aws-b1c91b55ce8c

Google Colab: ingyenesen használható, gyorsabb, mint az átlagos local erőforrás, választható CPU-GPU-TPU.

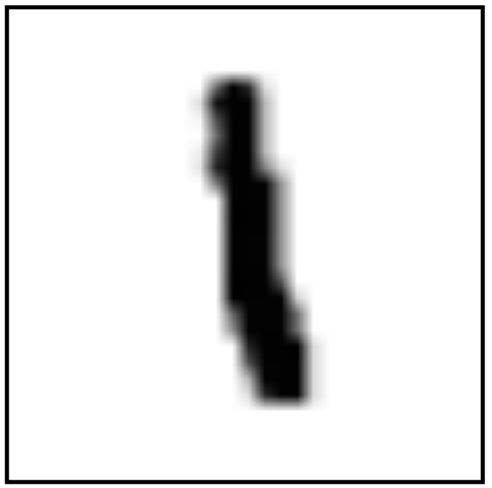
Képfeldolgozás

Probléma:

DeepLearning (egyik) állatorvosi lova: kézzel írt, digitalizált számjegyek felismerése.

MNIST adathalmaz:

- 70.000 minta: 60k train – 10k test
- 28x28 pixeles szürkeárnyalatos képek speciális formátumban ~ mátrix
- Train-test-validation



≈

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	.6	.8	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.5	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.7	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	.9	1	.1	0	0	0	0
0	0	0	0	0	0	0	.3	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Densely Connected Network

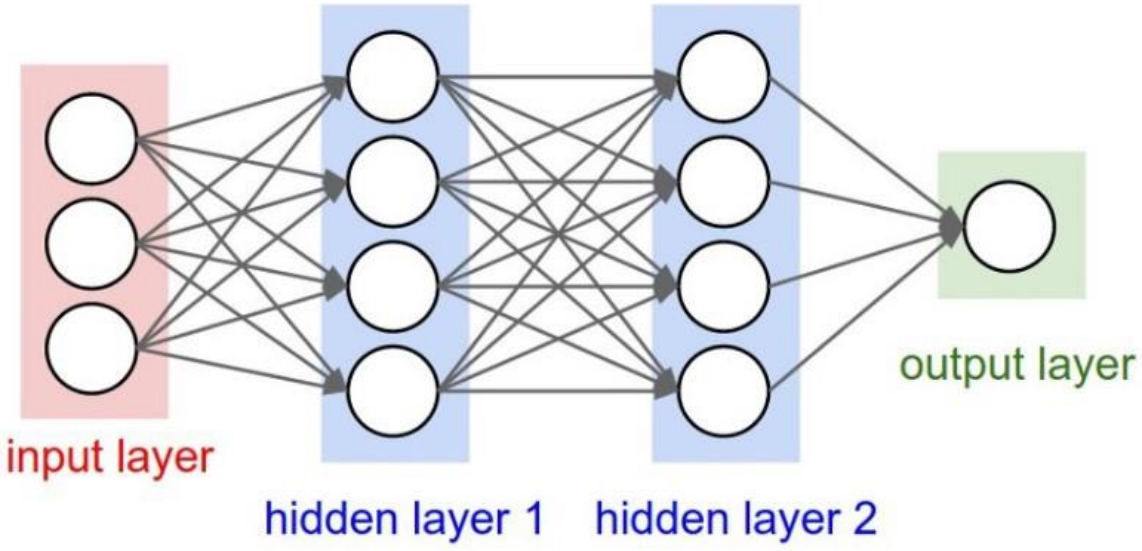
Olyan architektúra, amelyben az n. réteg minden összes az n+1. réteg minden elemével össze van kötve.

Globális mintázatokat próbál felismerni a bejövő adatokon.

Sok általános problémára mégis jó eredményt ad.

Rengeteg paraméter -> hosszas tanítás.

A számjegyeket még egy ember is sokféle képpen írja le, azok különbözők lehetnek pl. a képen belüli pozíció alapján máshol vannak a számok…



https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37

CNN

Konvolúciós réteg + pooling réteg + aktivációs réteg -> ez az alap konvolúciós architektúra, ez tanulja meg, és nyeri ki a fontos tulajdonságokat.

Ehhez még egy, már említett sűrűn összekapcsolt hálót adunk, ami a kinyert attribútumok alapján osztályoz

Konvolúciós réteg: filterek, és a bejövő „kép” konvolválása
Pooling réteg: méretbeli csökkentés, maximum érték vagy átlagolás.

Lokálisan néz mintázatokat, az alap rétegben egyszerű térbeli mintázatokat néz, majd ezekből állnak elő a sokkal absztraktabb filterek. Függőleges, vízszintes élek -> sarkok -> négyszögek -> billentyűk felismerése

Mikor milyen függvényt használjunk?

Kis segítség:

Probléma típusa	Használandó aktivációs függvény	Loss-függvény
Bináris osztályozás	sigmoid	binary_crossentropy
Többosztályos, egycímkés oszt.	softmax	categorical_crossentropy
Többoszt., többcímkés oszt.	sigmoid	binary_crossentropy
Regresszió tetszőleges értékekre	-	mse
Regresszió 0-1 közötti értékekre	sigmoid	mse / binary_crossentropy

François Chollet: *Deep Learning with Python* c. könyvéből

Gyakorlat

1. Töltsük le az MNIST adathalmazt a Google Colabos gépre.
2. Ismerjük meg az adatokat: méret, számosság, konkrét értékek, nézzük meg, hogy hogy néz ki egy adatpont ábrázolva.
3. Készítsük elő az adatokat, hogy azt majd a hálóba betölthessük.
4. Készítsünk egy egyszerű sűrűn összekötött neurális hálót, nézzük meg, hogy milyen eredményeket értünk el vele
5. Csináljunk egy CNN-t, hasonlítsuk össze az eredményeket

Vizualizáció

CNN-nél érdekes, még könnyebben értelmezhetők az entitások kimenetei.

Mit is lehetne vizualizálni?

- Input adatok: már megtettük
- Filterek: konvolúciós mátrixok – az egyes konvolúciós filterek milyen mintázatokat ragadnak meg a képkeből
- **Köztes konvolúciós kimenetek – aktivációs térképek**: hogyan alakítják a filterek a tényleges ábrákat, miket emelnek ki a képből az adott filterek
- Kimenet: végül a képek mely részei reprezentálják a legrelevánsabb osztályt

Gyakorlat

Nézzük meg, hogy hogyan is néznek ki a filterek!

1. Vegyük szemügyre az egyik általunk létrehozott CNN-t.
2. Nyerjük ki az aktivációs térképek értékeit!
3. Ábrázoljuk őket!

Előre tanított hálózatok

Hálót építeni „from scratch” hosszú, repetitív folyamat. Sok komplexebb hálót már betanítottak valami hasonló feladaton, minek végezzük el újra az időigényes tanítási fázist… + így sokkal jobb eredmények érhetők el, mintha mi magunknak tanítanánk rövid ideig a komplex architektúrát

Tfh. van egy előre definiált problémán jól működő háló, ezt szeretnénk a mi problémánk megoldására átalakítani.

CNN esetében két fő lehetőség

1. A konvolúciós részt békén hagyjuk, csak az osztályozó DNN-t tanítjuk újra.
2. Pár utolsó konvolúciós réteget is újratanítunk: csak a nagyon absztrakt, specifikus részek szorulnak módosításra, az alap filterek nem (pl.: élek, lyukak felismerése)

Előre tanított hálózatok

Feladat

ImageNet adathalmaz: képek különböző tárgyakról, mindegyikhez tartoznak tag-ek (célváltozó, hogy mi szerepel a képeken).

Sok jól működő hálózat készült ehhez az osztályozási feladathoz, ezt ki kéne használni.

Ha a feladatunk pl. macskák és kutyák megkülönböztetése, akkor az egyik ilyen hálót egy kis módosítással át lehetne alakítani úgy, hogy az a mi problémánkat is képes legyen megoldani.

Gyakorlat

1. Töltsük le a macskás-kutyás adathalmazt
2. Töltsük be az egyik ImageNet-es hálózati architektúrát
3. Nézzük meg a struktúráját
4. Fagyasszuk be az összes konvolúciós réteget
5. Hozzunk létre egy számunkra releváns DNN-t
6. Csatoljuk a DNN-t a konvolúciós architektúrához

Kérdések?

mnovak@clementine.hu

Clementine