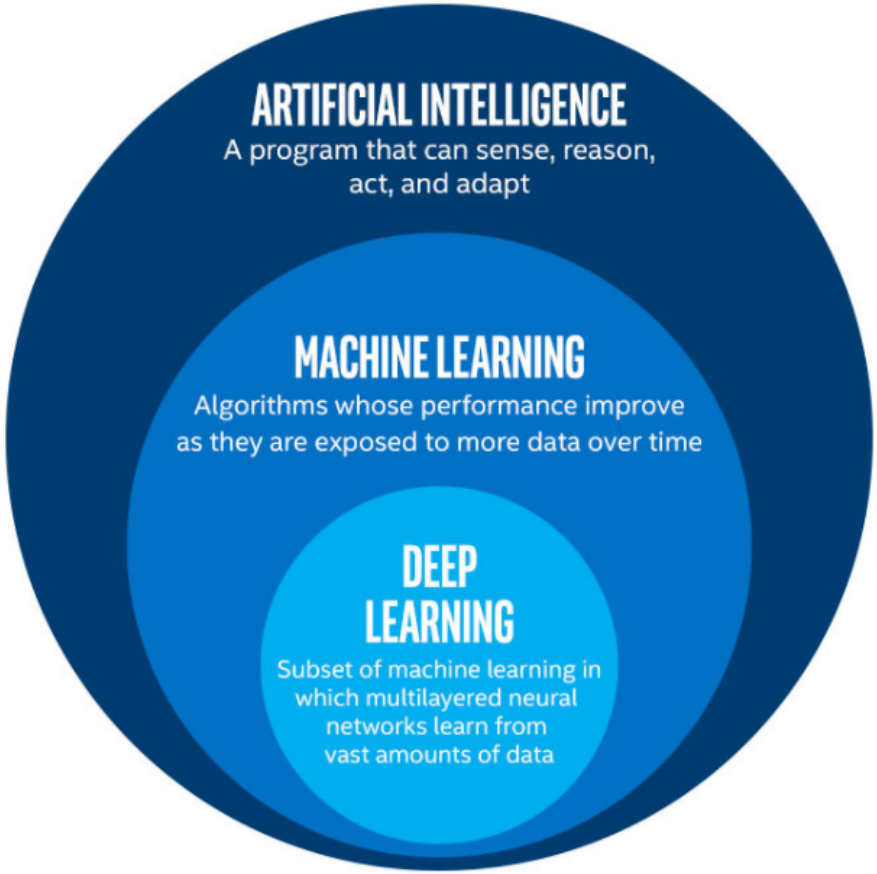
Deep Learning in PyTorch Workshop CLOUDUT

15.03.2021

Raluca Brehar
Departament Calculatoare
Universitatea Tehnica din Cluj-Napoca
Raluca.Brehar@cs.utcluj.ro

Context

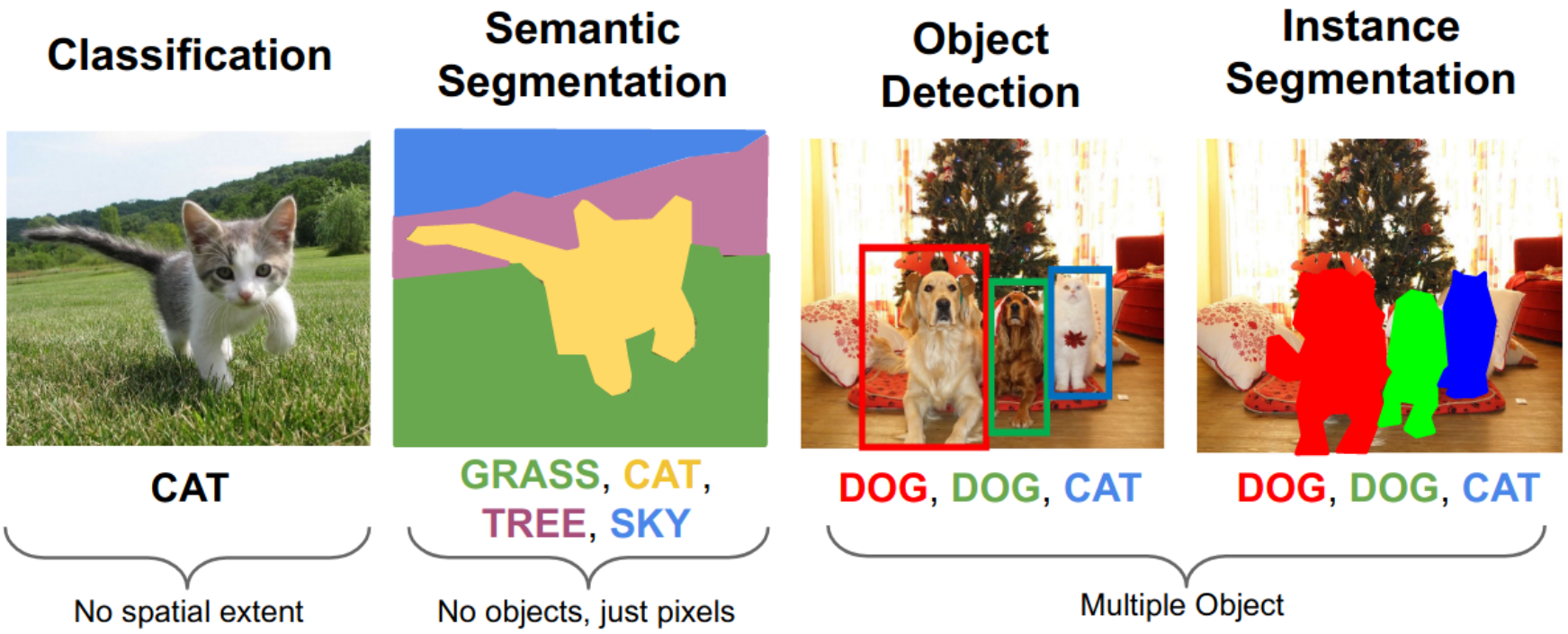
- Artificial intelligence
- Machine learning
- Deep learning
- Deep learning for Computer Vision



https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Context

Computer Vision Tasks



Sursa: http://cs231n.stanford.edu/slides/2020/lecture_12.pdf

Obiective

- Utilizare infrastructură CloudUT în aplicaţii care necesită:
  - calcul GPU masiv pentru probleme de învăţare profundă
  - spaţiu de stocare (aplicaţiile de învăţare profundă au nevoie de colecţii mari de imagini adnotate pentru a antrena modele cu performanţe ridicate).
- Pregatire aplicatie pe maşina locală
  - Daca resursele locale nu sunt suficiente se poate apela la Cloud
- Portare aplicaţie în CloudUT (rezolva problemele de la un anumit nivel de complexitate →permite scalabilitatea pentru rezolvarea unor probleme complexe)
- Toate aplicaţiile utilizează PyTorch

Scenariu demonstrativ

- **Pregătire aplicaţie pe maşina locală**
- Proiectarea modelului local – detaliat (numar mic de epoci, batch size mic , gasirea learning rate optiom si a altor hiperparametrii, dataset redus)
- **Portare aplicaţie în CloudUT :**
- Configurarea masinii pe baza necesitatilor de resurse estimate:
  - CloudUT ofera o masina virtuala – care se poate configura (resurse GPU, memorie, spatiu de stocare) – se face o cerere adresata inginerului de sistem care configureaza masina conform specificatiilor
  - Serviciile pe care le fac inginerii de sistem → instaleaza masina cu sistem de operare cerut
  - Exemplificare conectare la masina din CloudUT ?
- Copiere datelor pe care se antreneaza / evalueaza modelul (exemplu ?)
- Copierea modelului
- Instalaratea in Cloud a librariilor necesare (creare / clonare env nou anaconda)
- Comparaţie resurse utilizate (memorie, timp) şi performanţa modelelor antrenate la rularea pe maşina locală şi în CloudUT

Scenariu demonstrativ

- Configuraţie maşina locală:
  - Sistemul de operare Ubuntu 18.04
  - placă grafică NVIDIA GeForce RTX 2080 Ti/PCIe/SSE2 cu 11GB memorie
  - Procesor Intel i7-3770K CPU 3.5GHz (4 nuclee de procesare cu 8 nuclee virtuale) + 16GB RAM;
- Configuraţie maşina virtuală din CloudUT:
  - Ubuntu 20.04
  - NVIDIA V100Q cu 32GB memorie
  - procesor Intel Xeon Gold 6230 2.1GHz (8 nuclee de procesare) + 128GB RAM.

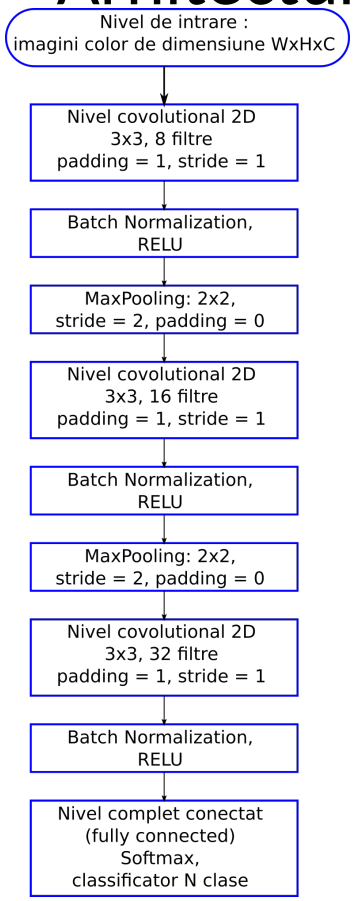
Aplicaţii scenariu demonstrativ

1) Recunoaşterea cifrelor scrise de mână utilizând rețele neuronale convoluționale liniare (clasificare)

2) Segmentarea semantică a imaginilor color

1. Recunoaşterea cifrelor

- Arhitectura reţelei




```python
import torch
import torch.nn as nn
import time
import copy
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

class Unit(nn.Module):
    def __init__(self, in_channels, out_channels, ksize=3, str=1, pad=1):
        super(Unit, self).__init__()

        self.conv = nn.Conv2d(in_channels=in_channels, kernel_size=ksize, out_channels=out_channels, stride=str,
                              padding=pad, bias=False)
        self.bn = nn.BatchNorm2d(num_features=out_channels)
        self.relu = nn.ReLU()

    def forward(self, input):
        output = self.conv(input)
        output1 = self.bn(output)
        output2 = self.relu(output1)

        return output2


class basicCNN(torch.nn.Module):
    def __init__(self, nf=8, num_classes=2, w_=_128, h_=_128):  # parametrii si valorile default
        super(basicCNN, self).__init__()
        self.layer1 = Unit(in_channels=3, out_channels=nf)  #wxhx8
        self.mp1 = nn.MaxPool2d(kernel_size=2, stride=2)# w/2xh/2x8

        self.layer2 = Unit(in_channels=nf, out_channels=2*nf) #w/2 x h/2 x16
        self.mp2 = nn.MaxPool2d(kernel_size=2, stride=2)# w/4 x h/4 x16

        self.layer3 = Unit(in_channels=2*nf, out_channels=_3*nf)  # w/4 x h/4 x32
        self.fc = nn.Linear(int(w/4 * h/4) * nf*3, num_classes, bias=True)

    def forward(self, input):
        out = self.layer1(input)
        out = self.mp1(out)

        out = self.layer2(out)
        out = self.mp2(out)

        out = self.layer3(out)
        out = out.view(out.size(0), -1)  # Flatten them for FC
        out = self.fc(out)
        return out
```

1. Recunoaşterea cifrelor

- Date utilizate
- MNIST (28x28x1)
- https://en.wi

- Parametrii variaţi la antrenare:
  - Număr de epoci
  - Batch size

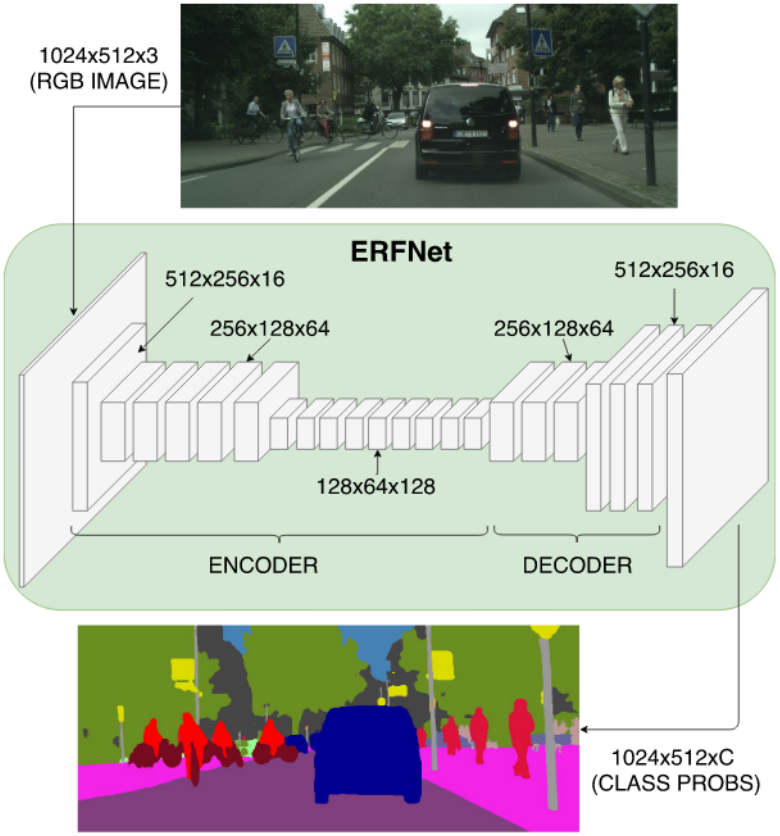
1. Recunoaşterea cifrelor

Masina locala			
Epoci	Batch Size	Timp antrenare	Acuratete
50	128	0:04:56	87 %
50	256	0:04:36	86%
50	512	0:04:26	81%
CloudUT			
50	128	0:04:29	88%
50	256	0:04:19	85%
50	512	0:04:14	78%

https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

2. Segmentare semantica

- Reteaua ERFNet[1]



(1) Encoder:
Straturile 1-16:
- residual and downsampling blocks + interleaved dilated convolutions
- permite straturilor mai adanci sa adune mai multa informatie de context (imbunatateste clasificarea)
(2) Decoder:
Straturile 17-23:
- fac upsample la hartile de trasaturi pentru a se potrivi cu rezolutia nivelelor de deconvolutie

2. Segmentare semantica

- Reteaua ERFNet[1] antrenata pe setul de date Cityscapes[2]
  - Cityscapes dataset:
    - Imagini urbane cu 30 de clase semantice adnotate la nivel de pixel
    - 5000 de imagini adnotate
- Parametrii variati la antrenare:
  - Numar de epoci: 50, 100
  - Batch size: 2, 3, 4, 8
  - Dimensiunea imaginilor din setul de antrenare: 512x1024

2. Segmentare semantica

Masina locala imagini 512x1024					
Epoci	Batch Size	Antrenare encoder	Antrenare decoder	IoU on VAL	Timp mediu pe imagine
50	2	2h:27m:55s	2h:55m:55s	65.44 %	59ms
50	3	2h:15m:25s	2h:40m:42s	66.29 %	56ms
50	4	RuntimeError:	CUDA out of memory.	*	
CloudUT					
50	2	2h:58m:43s	3h:08m:54s	65.89%	68ms
50	3	2h:48m:43s	2h:48m:43s	66.58%	63ms
50	4	2h:04m:40s	2h:30m:43s	66.24%	60ms
50	6	1h:29m:57s	2h:19m:33s	68.43%	45ms
50	8	1h:15m:57s	2h:09m:53s	66.35%	28ms

Crestere de viteza de 1.4 ori la rularea in Cloud pentru batchsize mare, si crestere de acuratete !
Pentru batch size mic – performantele sunt asemanatoare cu ce se obtine pe masina locala !

*Tried to allocate 32.00 MiB (GPU 0; 10.76 GiB total capacity; 9.19 GiB already allocated; 41.81 MiB free; 68.85 MiB cached)

Concluzii

- **Beneficiile aduse de rularea in CloudUT**
  - Timp de execuţie / antrenare
  - Batch size mai mare – poate ajuta la creşterea performanţei modelelor
- **Paşii necesari pentru utilizarea CloudUT de către colective de cercetare pentru aplicaţii de învăţare profundă:**
  - Proiectarea şi implementarea modelului pe maşina locală
  - Stabilirea configuratiei masinii pe care se doreste rularea in CloudUT
  - Portarea datelor şi a modelului pe maşina din CloudUT
  - Crearea unui mediu anaconda nou si instalarea pachetelor necesare pentru modelul care se doreşte a fi rulat (se poate clona mediul de baza care are pytorch, torchvision, numpy, skitlearn)
  - Antrenarea modelului in Cloud, masurare performanta, îmbunătăţiri

Referinte

- [1] "ERFNet: Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation", E. Romera, J. M. Alvarez, L. M. Bergasa and R. Arroyo, Transactions on Intelligent Transportation Systems (T-ITS), [Accepted paper, to be published in Dec 2017].

- [2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Multumesc pentru atentie!

Raluca Brehar
Departament Calculatoare
Universitatea Tehnica din Cluj-Napoca
Raluca.Brehar@cs.utcluj.ro