UNIUNEA EUROPEANĂ

Instrumente Structurale 2014-2020

Efficient Deep Learning in CloudUT

ICCP 2021

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

CloudUT

Raluca Brehar, Ion Giosan, Cristian Vancea

Computer Science Department

Technical University of Cluj-Napoca

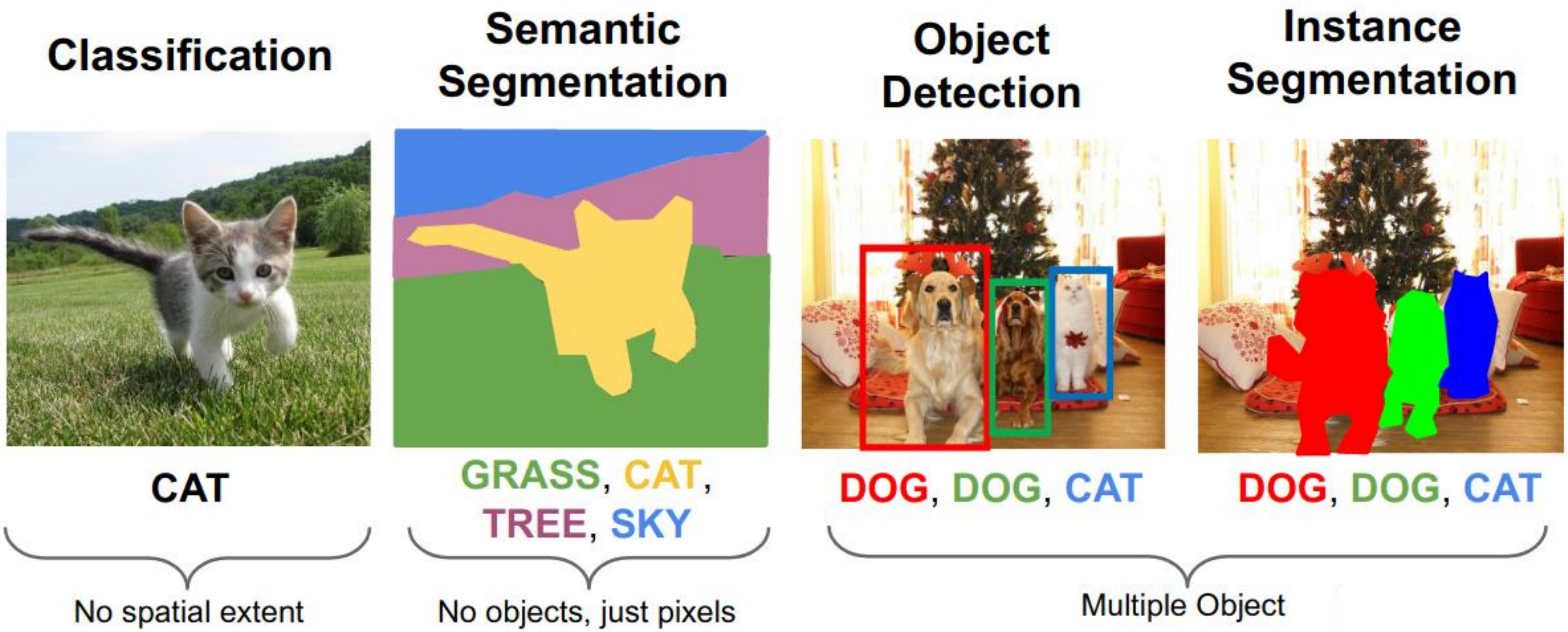
Raluca.Brehar@cs.utcluj.ro

Ion.Giosan@cs.utcluj.ro

Cristian.Vancea@cs.utcluj.ro

Context

Computer Vision Tasks



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

Solutions: Approaches with a high level of complexity based on neural networks.

Objectives

- Usage of the CloudUT infrastructure for applications that need:
  - High GPU processing for deep learning problems
  - Large storage space (deep learning applications need large datasets of annotated images in order to train accurate models).
- Preparation of the application on the local workstation
  - If the necessary resources are not enough, the CloudUT machines can be used.
- The application is ported on CloudUT (solves problems from a certain level of complexity → allows scaling for complex problem solving)
- We have experimented with PyTorch and MATLAB deep learning applications.

Methodology

Develop and run the application on the local machine

- Model design, establish training parameters: epochs, batch size, learning rate, other hyper-parameters

Application porting in CloudUT infrastructure

1. Request resources in CloudUT
2. Based on the request analysis the system engineer provides a virtual machine which is accessible by VPN
3. Copy the data for the application into CloudUT infrastructure (FTP, RDP)
4. Run the application in cloud (allows scalability for complex problem solving)

MATLAB application development

MATLAB

- Provides support for signal processing, optimization of functions, control system design, image and audio processing, machine learning and deep learning
- Needs high computational resources for parallel and distributed computing: memory, CPU, GPU
- Popular in the scientific community
- License based

PyTorch application development

PyTorch

- Open-source availability
- Flexibility
- Distributed model parallel training
- Mobile support
- Robust ecosystem – an active community of researchers and developers have built a rich ecosystem of tools and libraries for extending PyTorch and supporting development in areas from computer vision to reinforcement learning
- Native ONNX support
- Cloud support

Deep learning in MATLAB and PyTorch

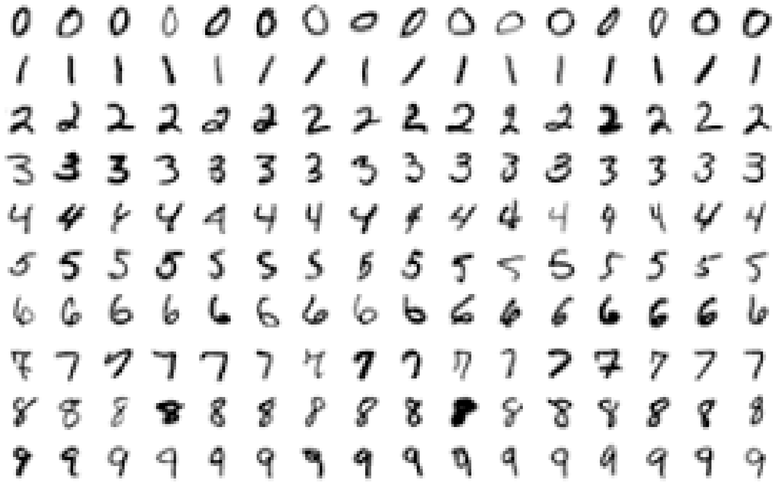
- Support for convolutional neural network design, training and for prediction based on the trained models
- Training/prediction processes are scalable with respect to available computing resources
- The speed increase is proportional to the work capabilities of the machine used for training

Purpose and advantages of working with deep learning models in CloudUT

- For research teams
- Enables the efficient execution of applications
- Usage of CloudUT infrastructure configured to each application needs
- Reduces the execution time
- Possibility to test applications not running on personal computers limited by resources or insufficient storage space.
- Quick access to TUCN network & privacy

Demonstrative applications

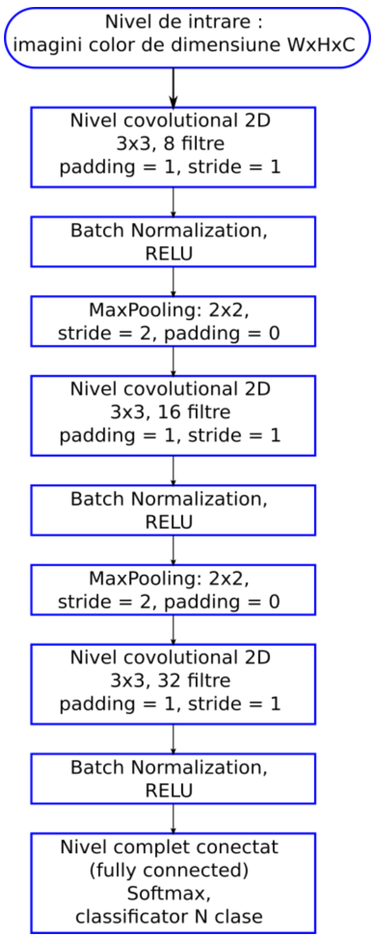
1) Handwritten digit recognition



2) Semantic segmentation of color images

1. Digit recognition

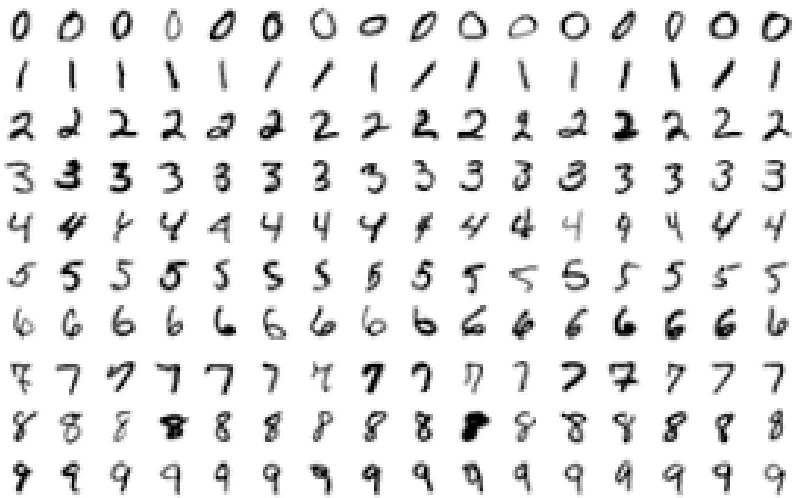
- CNN architecture




```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. Digit recognition

- Dataset: MNIST (28x28x1)
  https://en.wikipedia.org/wiki/MNIST_database



- Hyper-parameters
  - Number of convolutional filters
  - Number of epochs
  - Batch size

1. Test configuration: PyTorch

Local machine configuration

- Ubuntu 18.04
- GPU: NVIDIA GeForce RTX 2080 Ti/PCIe/SSE2 with 11GB memory
- Processor: Intel i7-3770K CPU 3.5GHz (4 processing cores and 8 virtual cores) + 16GB RAM

Virtual machine configuration in CloudUT

- Ubuntu 20.04
- GPU: NVIDIA V100Q with 32GB memory
- Processor Intel Xeon Gold 6230 2.1GHz (8 processing cores) + 128GB RAM

1. Digit recognition in PyTorch

Local Machine			
Epochs	Batch Size	Training Time	Accuracy
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%

NO noticeable increase in accuracy or execution time !
Proves the Cloud infrastructure is suitable for high computation applications.

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

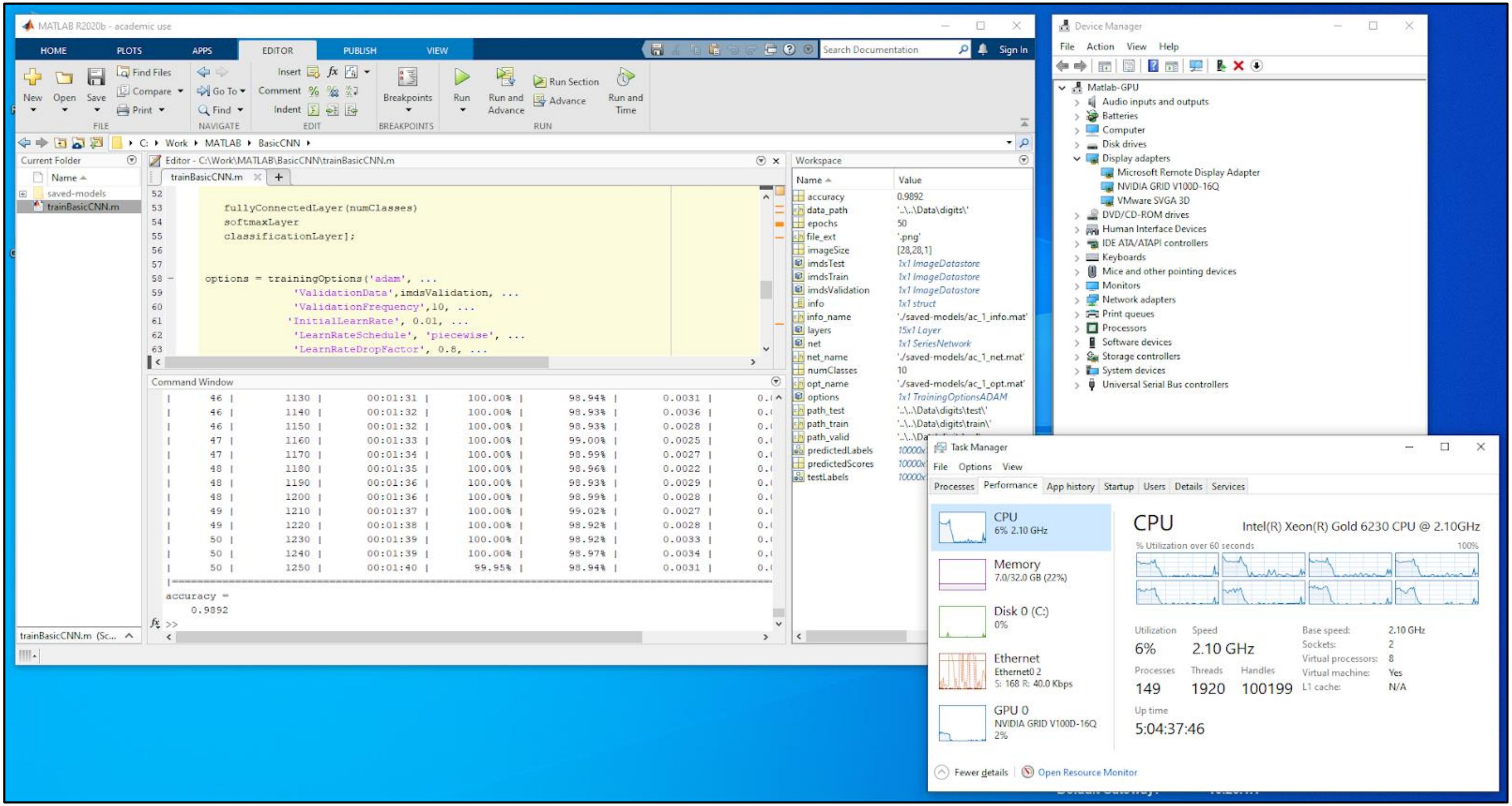
1. Digit recognition in MATLAB

Test configurations

1. Local machine: Windows 10
   - **GPU** NVIDIA GeForce RTX 2080 Ti/PCIe/SSE2 12GB memory
   - **CPU** Intel i7-3770K@3.5GHz (8 cores)
   - 16GB **RAM**
2. Virtual machine 1, in CloudUT: Windows 10
   - **GPU** NVIDIA V100 with 16GB memory
   - **CPU** Intel Xeon Gold 6230@2.1GHz (8 processing cores)
   - 32GB **RAM**
3. Virtual machine 2, in CloudUT: Windows 10
   - **GPU** NVIDIA V100 with 32GB memory
   - **CPU** Intel Xeon Gold 6230@2.1GHz (8 processing cores)
   - 128GB **RAM**

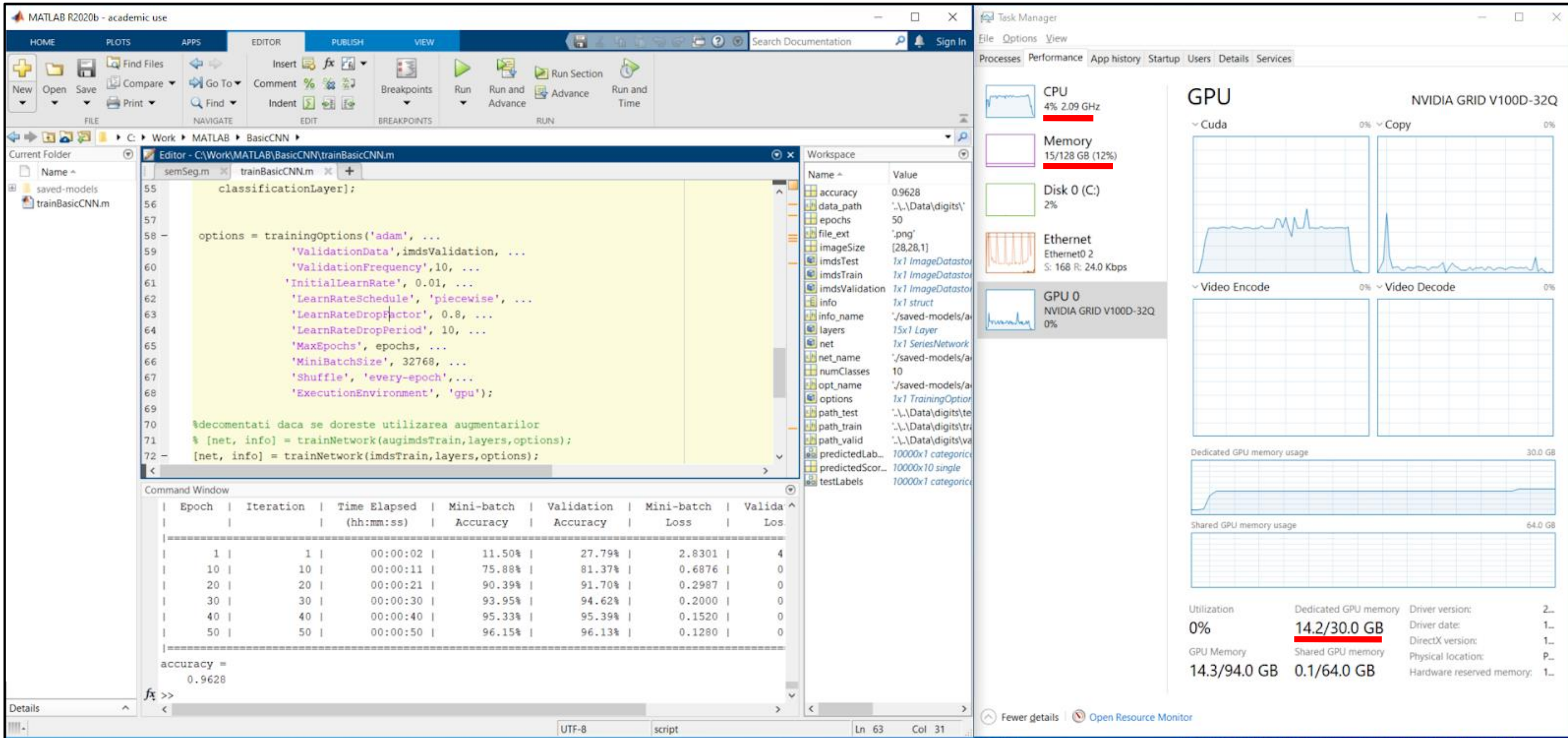
1. Digit recognition in MATLAB

Training session in CloudUT with available resources and their usage on virtual machine 1



1. Digit recognition in MATLAB

Training session in CloudUT with available resources and their usage on virtual machine 2



1. Digit recognition in MATLAB

Comparative results

Parameters			Execution on local workstation		Execution on virtual machine$_1$ (CloudUT)		Execution on virtual machine$_2$ (CloudUT)	
Epochs	Image size	Batch size	Accuracy	Training time (hh:mm:ss)	Accuracy	Training time (hh:mm:ss)	Accuracy	Training time (hh:mm:ss)
50	28x28x1	1024	0.9899	0:05:42	0.9887	0:02:39	0.9898	0:02:44
50	28x28x1	2048	0.989	0:04:22	0.9892	0:01:40	0.989	0:01:39
50	28x28x1	4096	0.9898	0:03:46	0.99	0:01:13	0.9898	0:01:12
50	28x28x1	8192	0.9876	0:03:29	0.9875	0:01:07	0.9875	0:01:00
50	28x28x1	16384	0.9832	0:03:37	0.9829	0:00:58	0.9829	0:01:00
50	28x28x1	32768	0.9628	0:03:32	0.9628	0:01:06	0.9628	0:00:50

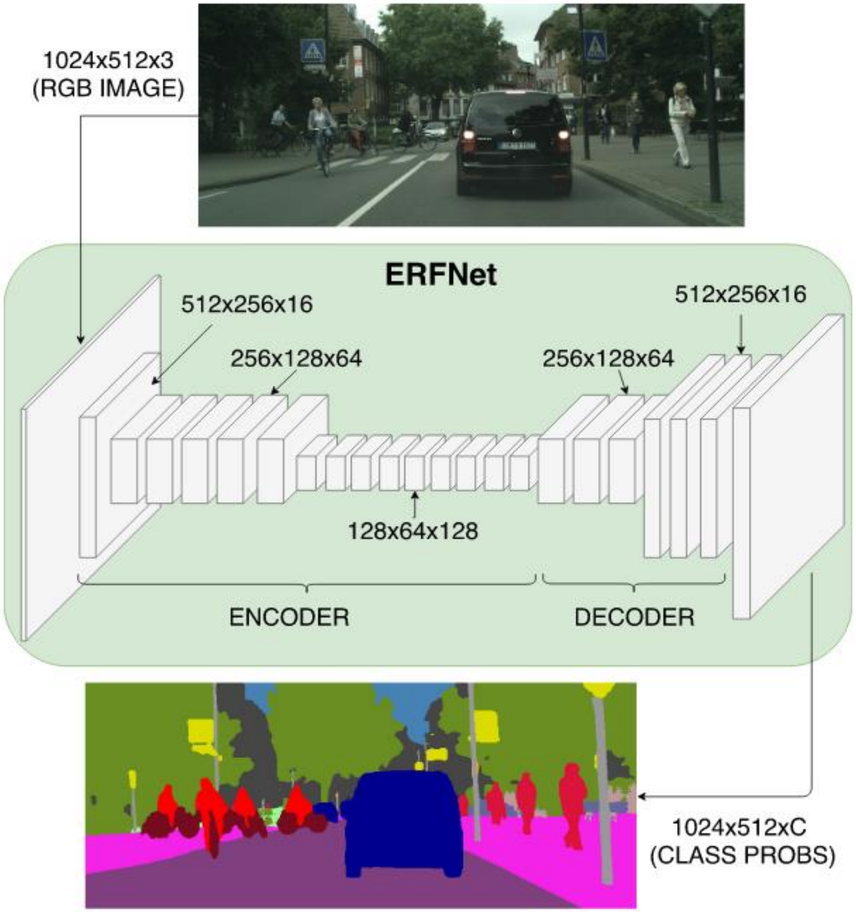
1. Digit recognition in MATLAB

Comparative analysis of execution time

- On virtual machine 1 from CloudUT (compared with the local machine) there is a mean decrease of the execution time of 65% (min. 53%, max. 73%)
- On virtual machine 2 from CloudUT, with a large batch size (32 768 images), there is a time execution decrease of 24% with respect to the virtual machine 1
  - Due to the high processing power of the GPU for a large batch size
  - The computations were made faster due memory increase (32GB for virtual machine 2 vs. 16GB for virtual machine 1)

2. Semantic segmentation in PyTorch

- ERFNet[1] network



(1) Encoder: Layers 1-16

➢ Residual and downsampling blocks + interleaved dilated convolutions

(2) Decoder: Layers 17-23

➢ Up-samples the feature maps to match the resolution of the deconvolution layers.

2. Semantic segmentation in PyTorch

- ERFNet[1] trained on Cityscapes[2]
  - Cityscapes dataset:

    - Urban images containing 30 semantic classes with pixel level annotations.

    - 5,000 annotated images
- Hyper-parameters:
  - Epochs: 50, 100
  - Batch size: 2, 3, 4, 8
  - Image size: 512x1024

2. Semantic segmentation in PyTorch

Local machine results (images of size 512x1024)					
Epochs	Batch Size	Encoder training	Decoder training	IoU on VAL	Average time per image
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.	*	
Virtual machine from CloudUT results (images of size 512x1024)					
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

A speed increase factor of 1.4 while training in Cloud, for a large batch size, increase in accuracy !
For a small batch size the accuracy is comparable on the local machine and on the virtual machine.

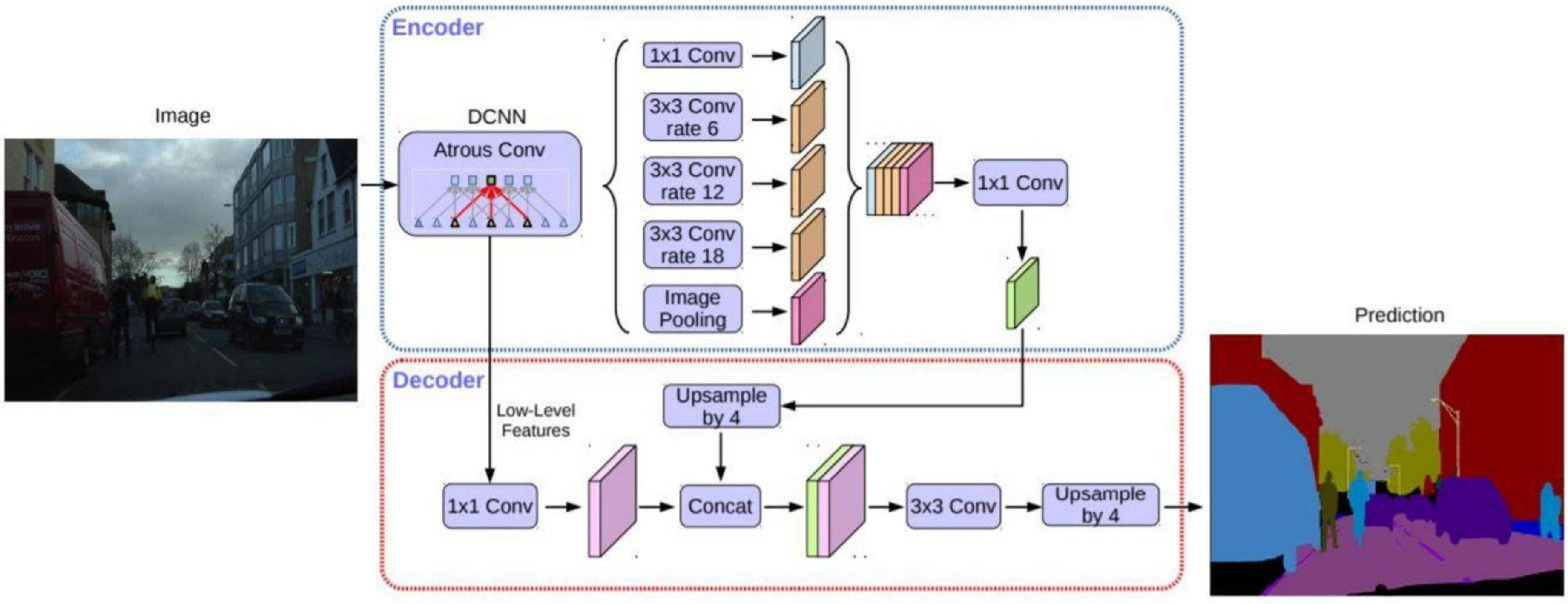
*Tried to allocate 32.00 MiB (GPU 0; 10.76 GiB total capacity; 9.19 GiB already allocated; 41.81 MiB free; 68.85 MiB cached)

2. Semantic segmentation in MATLAB

Deep learning architecture

DeepLabv3+ [1]: Encoder (Resnet-18) + Decoder

- Layers: *Convolution, Batch Normalization, ReLU, Image Pooling*
- Number of layers: 100
- Resnet-18 [2] (1000 classes): pre-trained on 10^6 images from Imagenet (source: http://www.image-net.org)

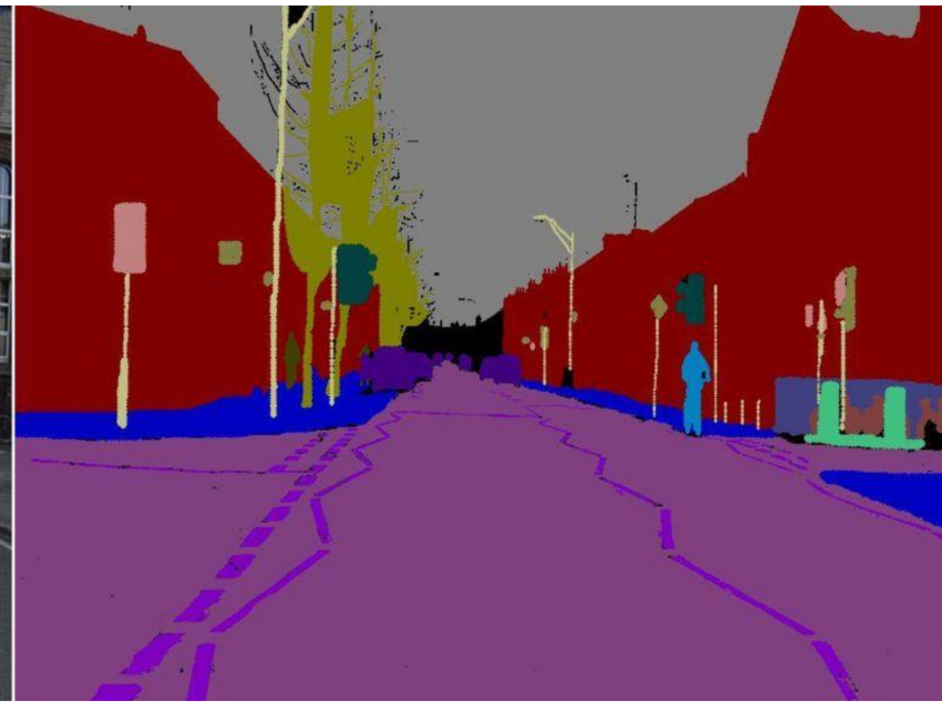


2. Semantic segmentation in MATLAB

CamVid dataset (701 images 960x720x3) – 32 classes

Source: http://web4.cs.ucl.ac.uk/staff/g.brostow/MotionSegRecData

Original image



Annotated image

2. Semantic segmentation in MATLAB

Training parameters

- *Stochastic Gradient Descent with Momentum*
- Learning rate: 0.003 (initial), decay with a factor of 0.3 every 10 epochs.
- Number of epochs: variable
- Batch size: variable

Evaluation metrics

- Accuracy
- Intersection over Union (IoU)
- F1 score on contours (BFScore)

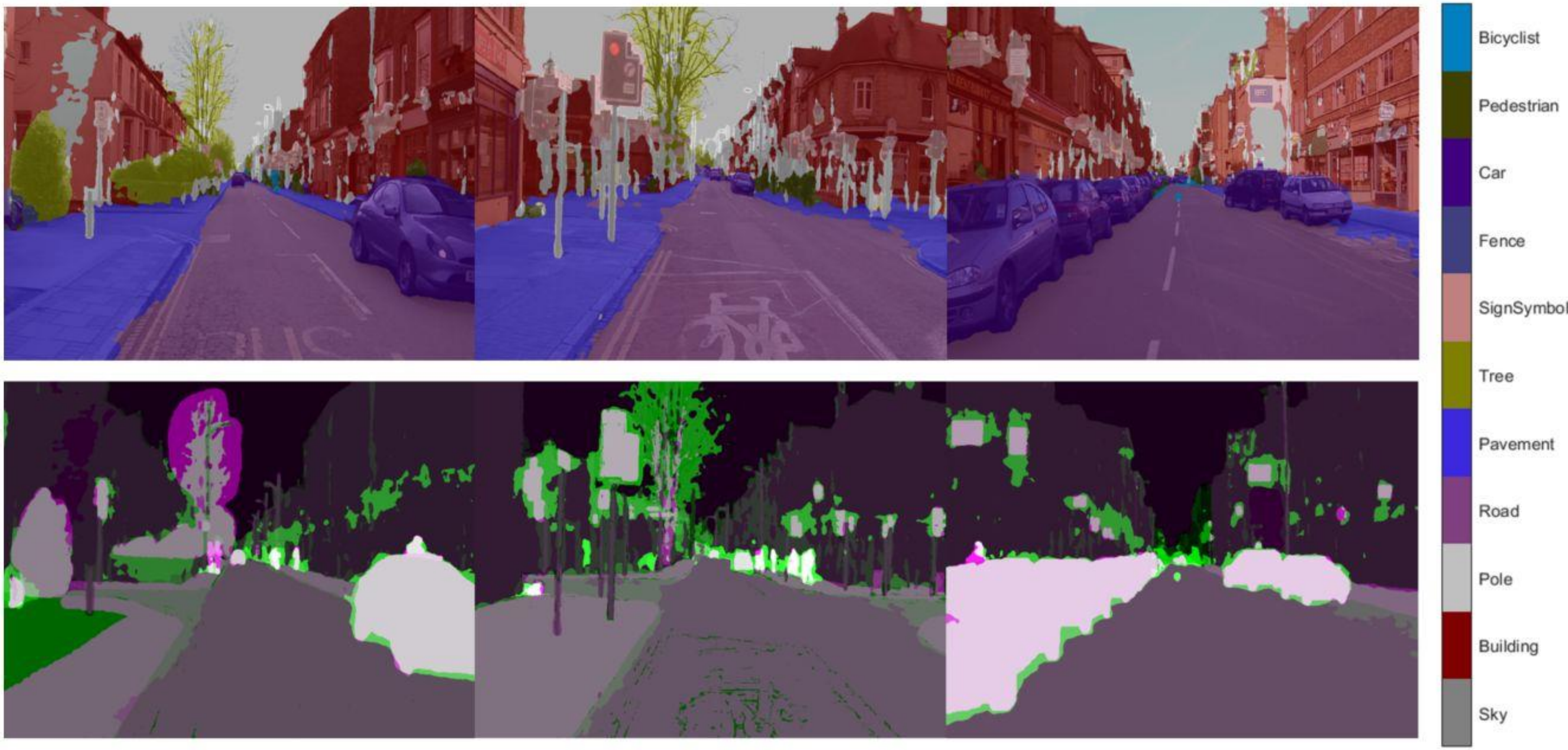
2. Semantic segmentation in MATLAB

Test configurations

1. Local workstation: Windows 10
   - **GPU** NVIDIA GeForce RTX 2060 SUPER 8GB memory
   - **CPU** Intel i7-3770K@3.5GHz (8 threads)
   - 16GB **RAM**
2. Local server: Windows 10
   - **GPU** NVIDIA GeForce GTX 1080Ti with 12GB memory
   - **CPU** Intel i9-7900X@4GHz (20 processing cores)
   - 128GB **RAM**
3. Virtual machine 1, in CloudUT: Windows 10
   - **GPU** NVIDIA V100 with 16GB memory
   - **CPU** Intel Xeon Gold 6230@2.1GHz (8 processing cores)
   - 32GB **RAM**

2. Semantic segmentation in MATLAB

Experimental results (virtual machine 1, CloudUT)



Batch Size = 8, Training: 1550 iterations (30 epochs)

2. Semantic segmentation in MATLAB

Experimental results (virtual machine 1, CloudUT)

Batch Size = 8
Training: 1550 iterations (30 epochs)

Global accuracy	Average accuracy	Average IoU	Weighted IoU	Average BFScore
0.89658	0.86185	0.66822	0.83346	0.70117

Batch Size > 8 => Error: insufficient RAM

Class	Testing partition		
	IoU	Accuracy	BFScore
Sky	0.90804	0.93887	0.90636
Building	0.80341	0.82986	0.66167
Pillow	0.25494	0.74593	0.59589
Road	0.93057	0.94582	0.81708
Pavement	0.7472	0.8924	0.76095
Tree	0.77623	0.88676	0.73405
Sign symbol	0.43684	0.75522	0.55289
Fence	0.59367	0.81132	0.59107
Car	0.79434	0.92213	0.75142
Pedestrian	0.4633	0.86498	0.6267
Bicyclist	0.64193	0.88703	0.5652

Remark: lower efficiency for small size objects

2. Semantic segmentation in MATLAB

Comparative results

Parameters				Execution on virtual machine$_1$ (CloudUT)		Execution on local server		Execution on local workstation	
Epochs	Iterations	Image size	Batch size	Accuracy	Time (hh:mm:ss)	Accuracy	Time (hh:mm:ss)	Accuracy	Time (hh:mm:ss)
1	1	960x720x3	8	0.0721	00:00:45	0.0966	00:00:26	0.0721	00:00:49
2	104	960x720x3	8	0.8375	00:07:44	0.8366	00:06:59	0.8364	00:16:07
4	208	960x720x3	8	0.8475	00:14:43	0.8463	00:13:34	0.8468	00:31:01
8	400	960x720x3	8	0.9042	00:27:34	0.8871	00:25:21	-	-
16	800	960x720x3	8	0.9178	00:55:12	0.9177	00:51:30	-	-
25	1300	960x720x3	8	0.9244	01:27:38	0.9246	01:23:15	-	-
30	1550	960x720x3	8	0.9079	01:43:53	0.9083	01:39:09	-	-

Remark: $P_{virtual\_m1} \approx P_{server} = 2.2 \times P_{workst}$

Insufficient resources

2. Semantic segmentation in MATLAB

Test configurations

1. Local server: Windows 10
   - **GPU** NVIDIA GeForce GTX 1080Ti with 12GB memory
   - **CPU** Intel i9-7900X@4GHz (20 processing cores)
   - 128GB **RAM**
2. Virtual machine 2, in CloudUT: Windows 10
   - **GPU** NVIDIA V100 with 32GB memory
   - **CPU** Intel Xeon Gold 6230@2.1GHz (8 processing cores)
   - 128GB **RAM**

2. Semantic segmentation in MATLAB

Time and resource during training (max. 30 epocs)

Batch size	Necessary iterations	CloudUT - virtual machine 2 -		Local server		CloudUT (max values) - virtual machine 2 -		
		Time (hh:mm)	Performance (%)	Time (hh:mm)	Performance (%)	CUDA [%]	GPU Mem (GB)	RAM (GB)
2	1300	00:29	355	00:26	396	30	8	30
4	1700	00:51	202	01:00	172	30	8	30
6	1750	01:05	158	01:24	123	50	11	33
8	1550	01:14	139	01:39	104	60	12	35
10	1000	01:05	158	01:31	113	62	14	35
12	1050	01:22	126	04:00	43	68	16	37
14	900	01:24	123	03:13	53	70	19	37
16	780	01:24	123	03:10	54	73	20	38

Performance = Time / $\text{Time}_{\text{virtual\_m1(batch size=8)}}$

2. Semantic segmentation in MATLAB

Comparative analysis

MATLAB uses efficient parallelization algorithms => performance scalability with respect to available resources

- Small Batch Size –> reduced GPU consumption (memory/CUDA)
  => speed performance increase up to 150% for virtual machine 2.
- Large Batch Size -> high GPU consumption (memory/CUDA)
  => speed performance increase up to 230-300% for virtual machine 2
- Parallel computing made on GPU allows a stabilization of the working time for large batch size with respect to small batch size, when there are enough computing resources.
- For complex models we recommend virtual machines with at least 32G RAM on GPU.

Conclusions

CloudUT advantages for deep learning

- Decreased execution time depending on the application complexity.
- Allows larger batch size that can impact an increase in accuracy.

Steps for developing an application on CloudUT

1. Develop the application on the local machine.
2. Establish what are the necessary hardware resources for a more efficient training.
3. Ticketing request for the CloudUT administrator => provides a virtual machine.
4. Install on the virtual machine all the needed libraries for your application. Port the necessary data.
5. Run the application on the provided virtual machine. Save the models!

Bibliography

- [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.

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2014-2020

Thank you !

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

CloudUT

Raluca Brehar, Ion Giosan, Cristian Vancea
Computer Science Department
Technical University of Cluj-Napoca
Raluca.Brehar@cs.utcluj.ro
Ion.Giosan@cs.utcluj.ro
Cristian.Vancea@cs.utcluj.ro