Performance study of a Convolutional Neural Network for Sentinel-2 image classification

Antonio Di Pilato

Università degli Studi di Bari & INFN

antonio.dipilato@uniba.it

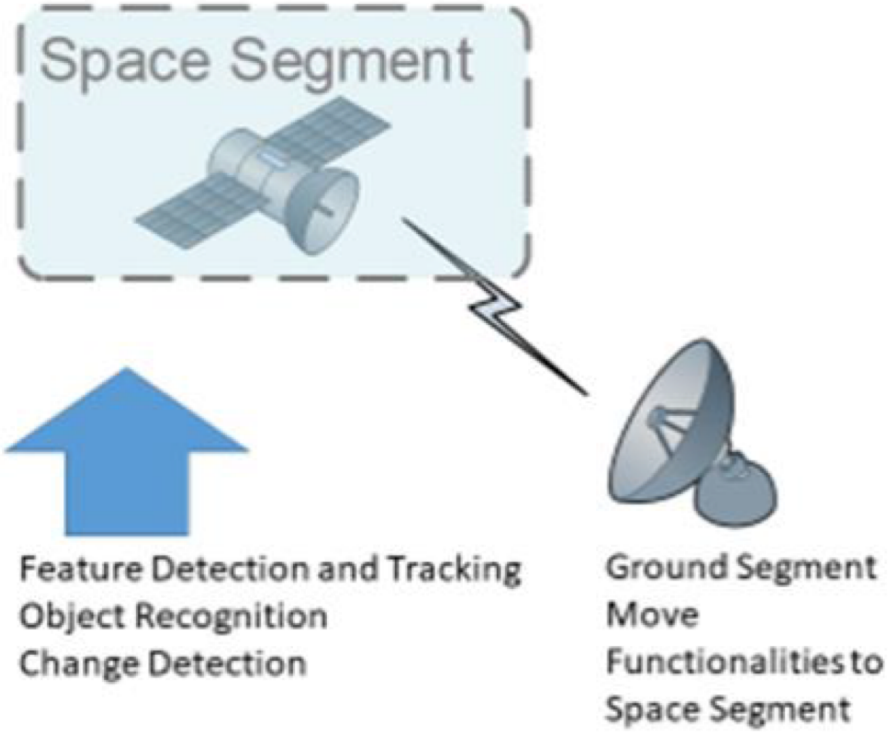
antonio.dipilato@ba.infn.it

Dottorato di Ricerca innovativo a caratterizzazione industriale - Ciclo XXXIII

The Satellite Remote Sensing challenge

- Earth Observation missions are growing in number and data production capabilities → **Big Data** challenge
  - *More mass memory storage is needed*
  - *The downlink bandwith is limited*

- **Idea**: Deep Learning algorithms accelerated by GPUs to detect selected features
  - ✓ Downlink of selected information
  - ✓ Increased processing speed and performance
  - ✓ Continuous stream of information



Installing GPUs onboard requires deep performance studies because of space major constraints (power sources, energy efficiency)

Copernicus and Earth Observation missions

Copernicus: the European Union's Earth Observation Programme

Six thematic information services provided based on satellite Earth Observation and in situ (non-space) data

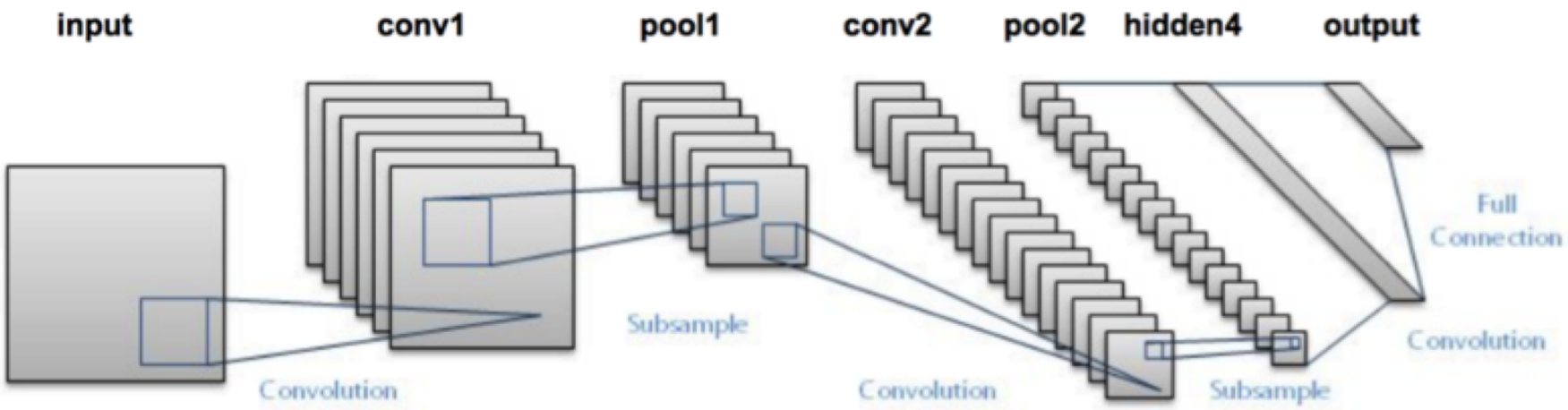
Two groups of missions:

- ***The Sentinels***, developed by ESA
- ***The Contributing Missions***

Each Sentinel mission is based on a constellation of two satellites to fulfill revisit and coverage requirements, providing robust datasets for Copernicus Services

Convolutional Neural Networks

- Generic classifier: $\boldsymbol{y = f(x;\theta)}$
- Optimized architecture for image processing
  - Exploit the spatial structure of the input
- Use several filters to extract features from the previous layer using the same set of parameters (property of ***translation invariance***)



$$\boldsymbol{a^1 = \sigma(b + w * a^0)}$$

where:

- a^1 is the output of the convolutional layer
- a^0 is the input of the convolutional layer
- w is the kernel
- b is the bias
- σ is the activation function

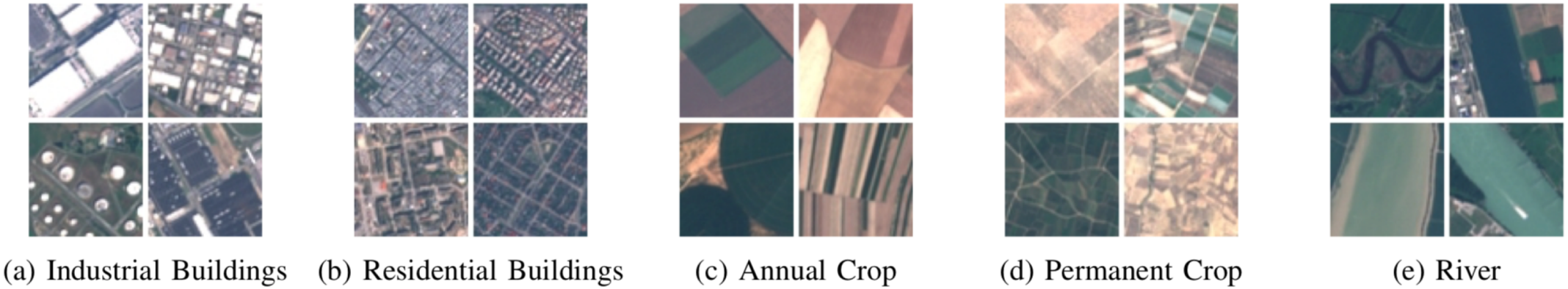
Sentinel-2 images and hardware

- Dataset: **EuroSAT**
- Size: 64 x 64 x 13
- Resolution: 10 m x 10 m
- Categories: 10
- Number of samples: 27000
  - Training samples: 21.600 (80%)
  - Validation samples: 2700 (10%)
  - Test samples: 2700 (10%)
- Batch size for training phase: 32
- Epochs: 25
- Hardware and software:
  - **NVIDIA Tesla K40 and Tesla P100* on ReCaS Data Center**
  - **NVIDIA Jetson Xavier at Planetek Italia**
  - **NVIDIA TensorRT (*software for fast inference*)**

Category	Label
0	*PermanentCrop*
1	*Residential*
2	*HerbaceousVegetation*
3	*Pasture*
4	*River*
5	*Industrial*
6	*Forest*
7	*AnnualCrop*
8	*Highway*
9	*SeaLake*

* thanks to INFN e MAECI/PGR00970 (MAECI_MEX_CMS Project)

Sentinel-2 image samples

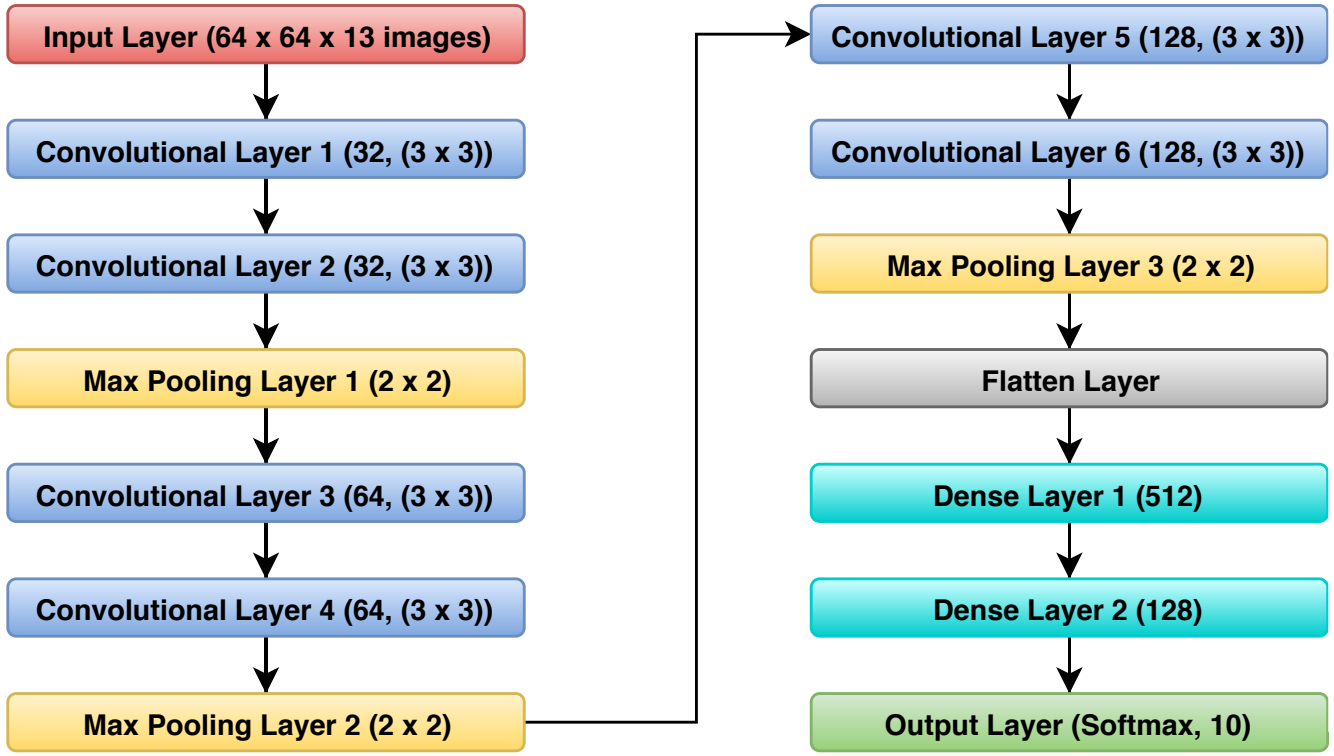


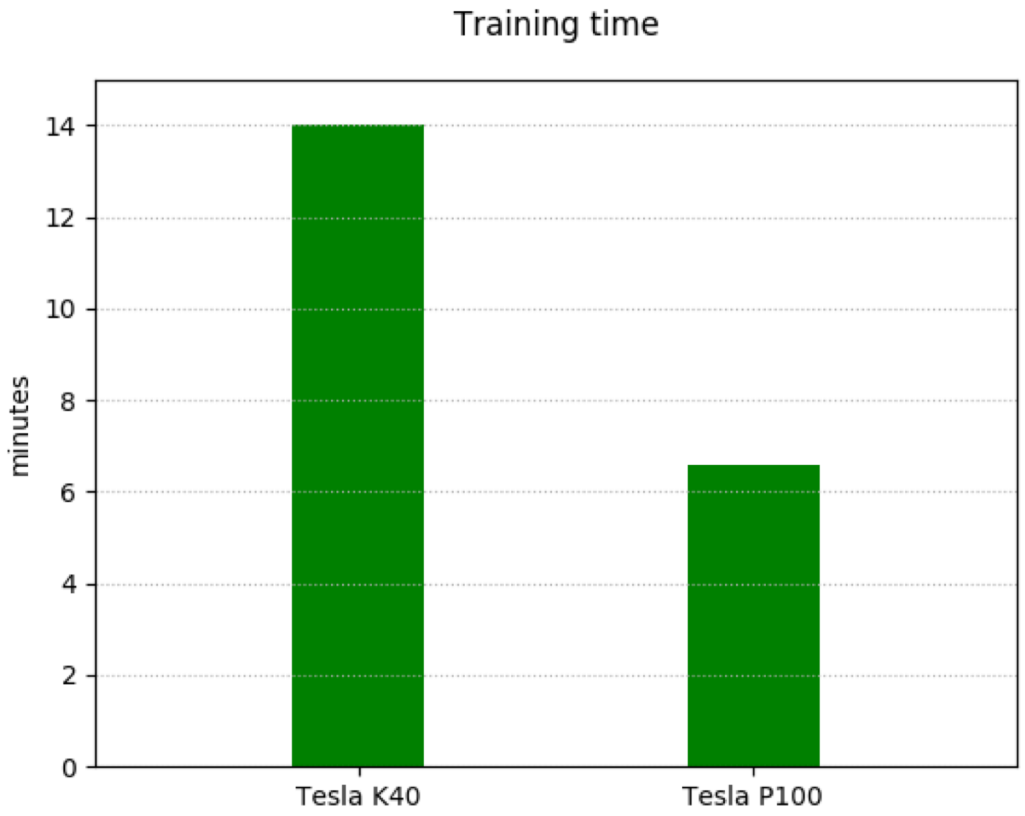
(a) Industrial Buildings (b) Residential Buildings (c) Annual Crop (d) Permanent Crop (e) River

(f) Sea & Lake (g) Herbaceous Vegetation (h) Highway (i) Pasture (j) Forest

P. Helber et al. *EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification*, arXiv:1709.00029v2 (2017).

Basic Model performance



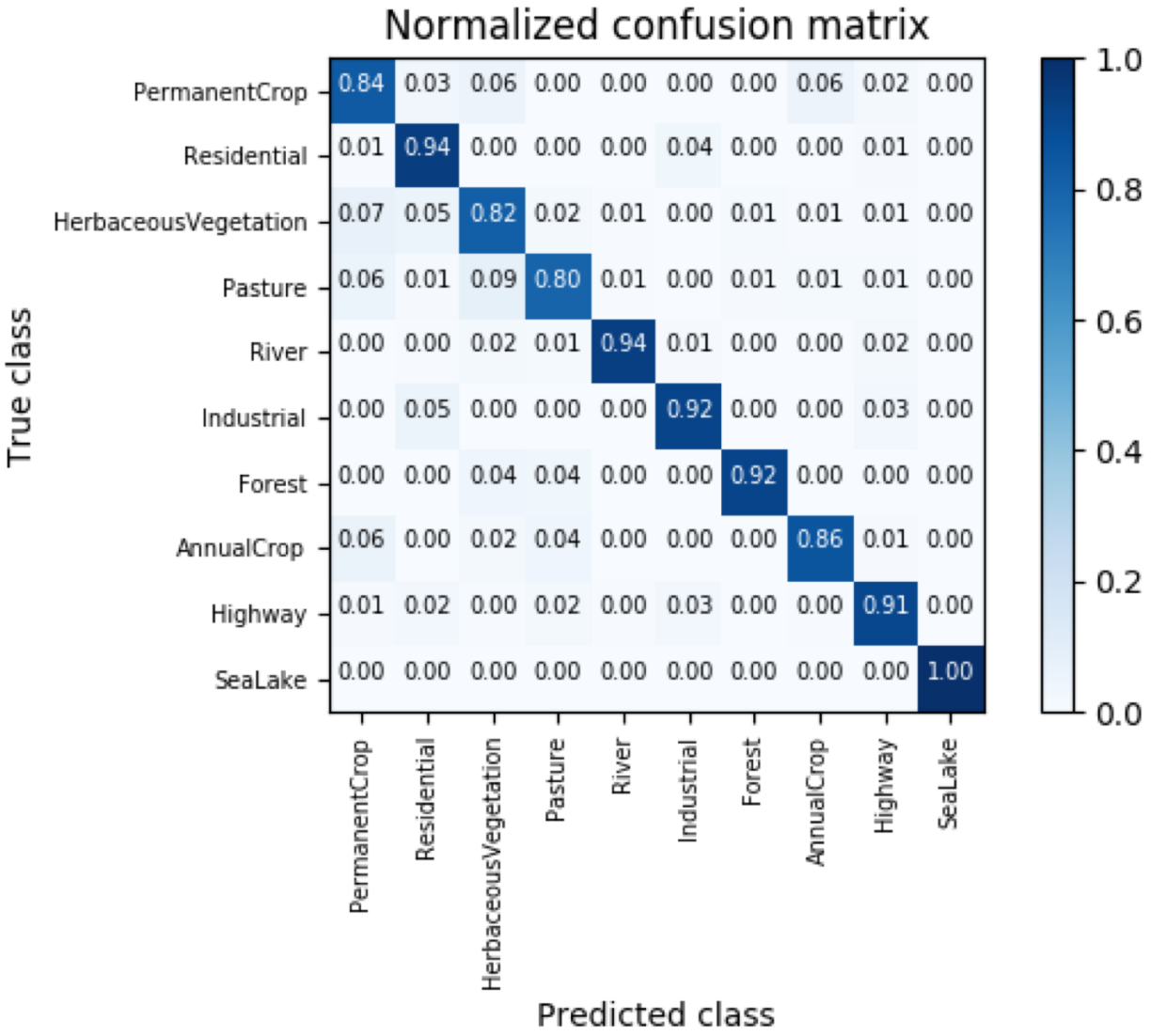


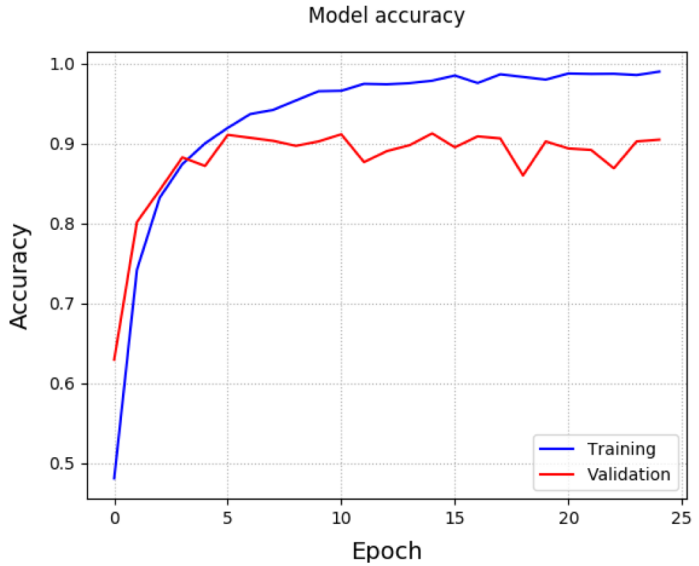
Trained with Keras for Tensorflow-GPU

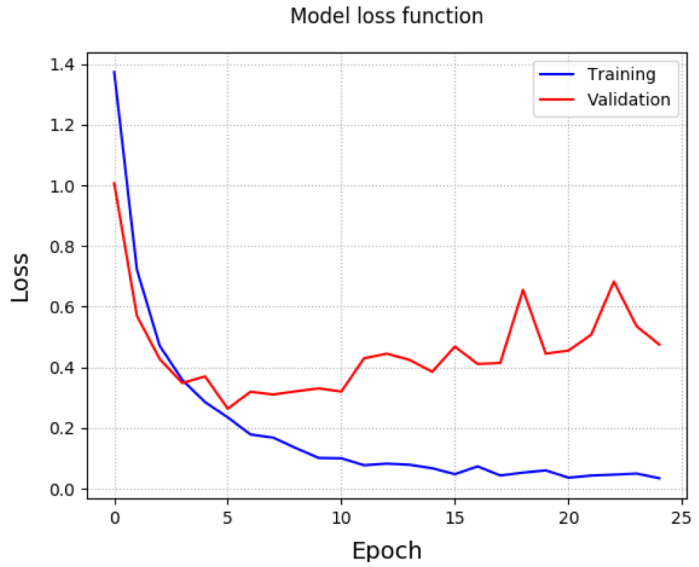
NVIDIA Tesla K40: 14 minutes

NVIDIA Tesla P100: ~ 6.5 minutes

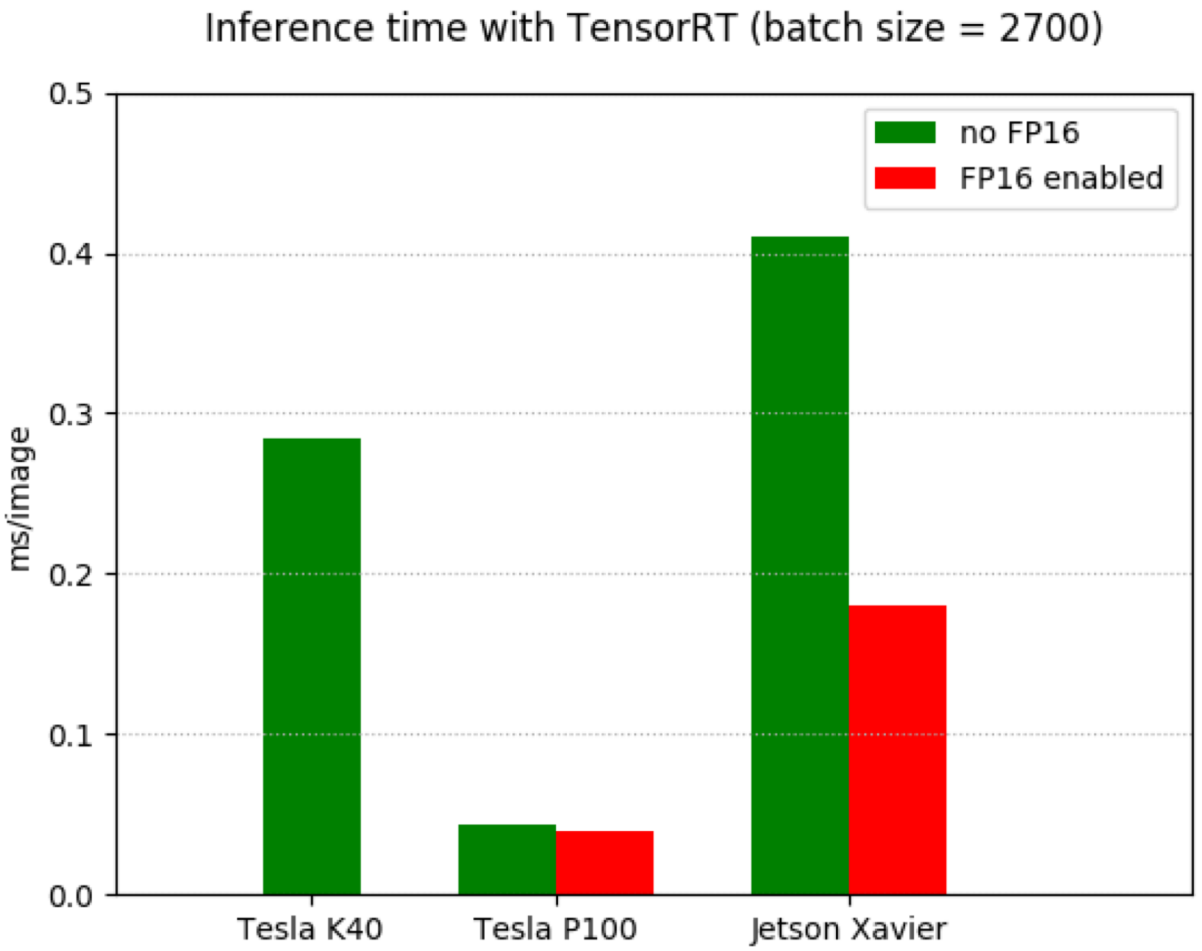
Basic Model performance







Basic Model performance



NVIDIA TensorRT is a high-performance deep learning inference optimizer and runtime that delivers low latency, high-throughput inference for deep learning applications.

NVIDIA Tesla K40: 0.285 ms/image

NVIDIA Tesla P100: 0.044 ms/image

NVIDIA Jetson AGX Xavier: 0.41 ms/image

NVIDIA Tesla P100 (FP16 enabled): 0.039 ms/image

NVIDIA Jetson AGX Xavier (FP16 enabled): 0.18 ms/image

To get an idea…

Inference with **Tensorflow-GPU**:

NVIDIA Tesla K40: 1.3 ms/image

NVIDIA Tesla P100: 0.93 ms/image

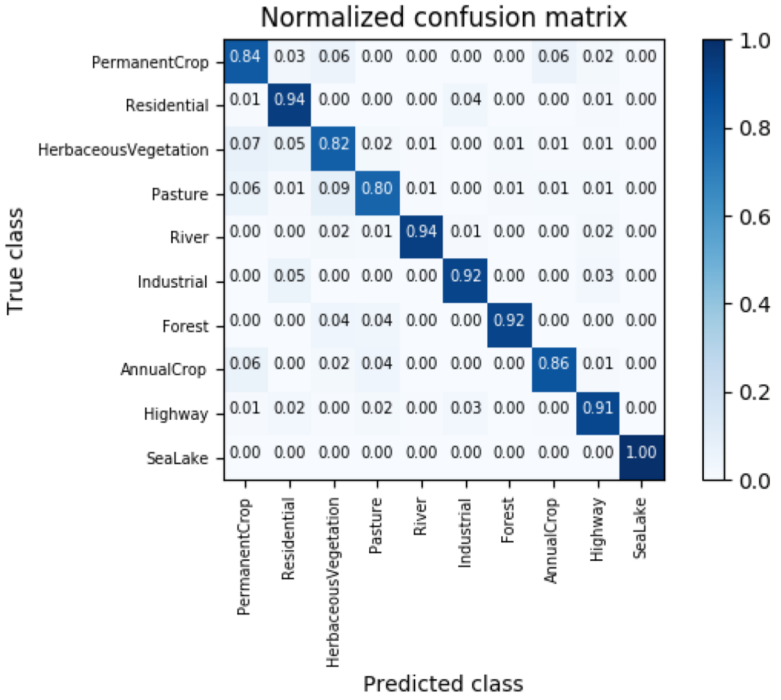
Studying the confusion matrix changes

- Several model tested

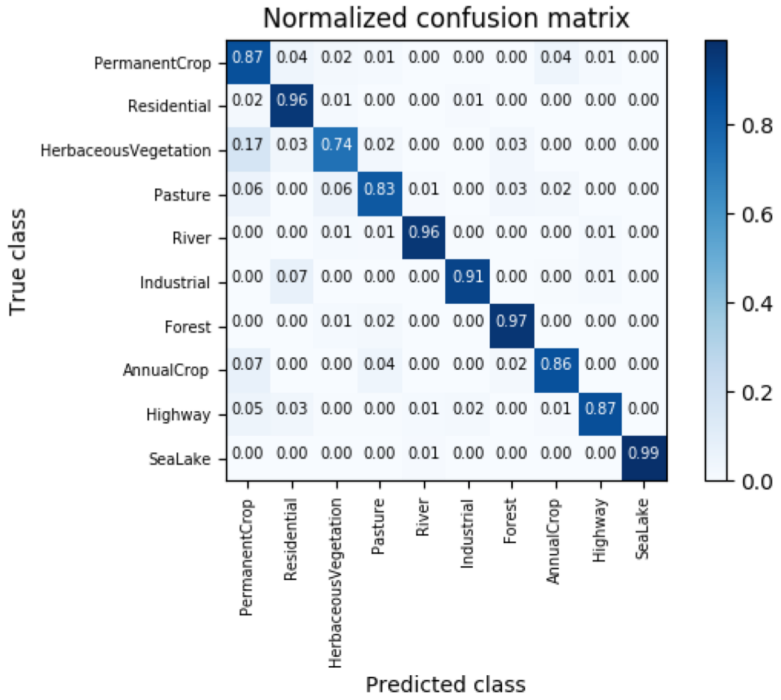
Model name	Description
Basic Model	Seen in the previous slides
1MoreConv Model	Basic Model + 1 Convolutional Stack (Convolutional Layers 7 & 8, 256 3x3-filters each)
1LessConv Model	Basic Model – 1 Convolutional Stack (no Convolutional Layers 5 & 6)
from64 Model	Basic Model with a double number of filters in each Convolutional Stack
from64_1MoreConv Model	from64 Model + 1 Convolutional Stack (Convolutional Layers 7 & 8, 512 3x3-filters each)
from64_1LessConv Model	from64 Model – 1 Convolutional Stack (no Convolutional Layers 5 & 6)

The performance (accuracy, loss, max accuracy) are similar among all the models

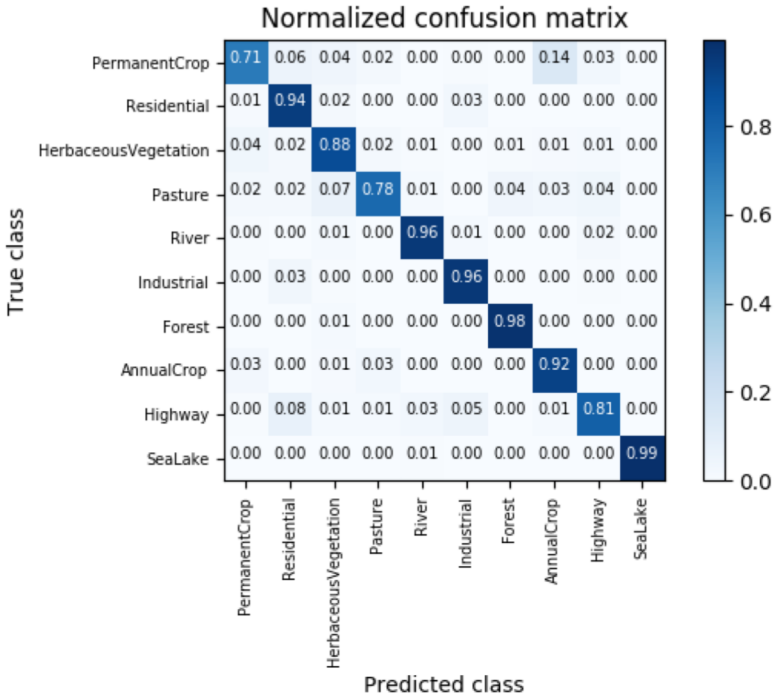
Studying the confusion matrix changes



Basic Model

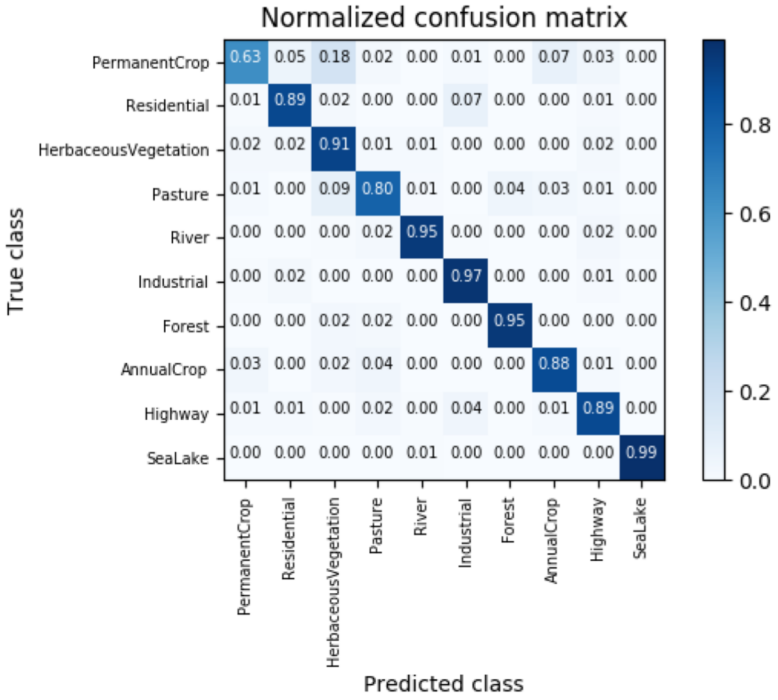


1MoreConv Model

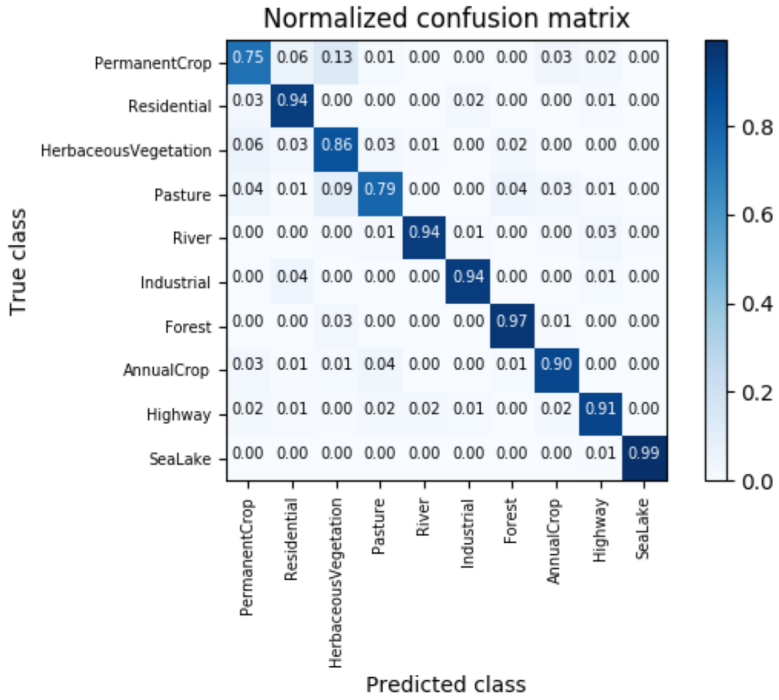


1LessConv Model

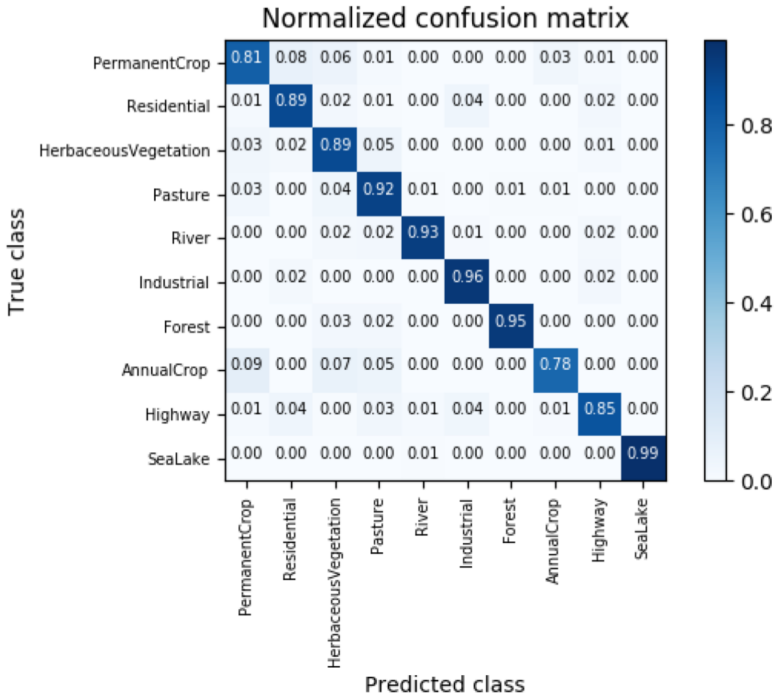
Studying the confusion matrix changes



from64 Model

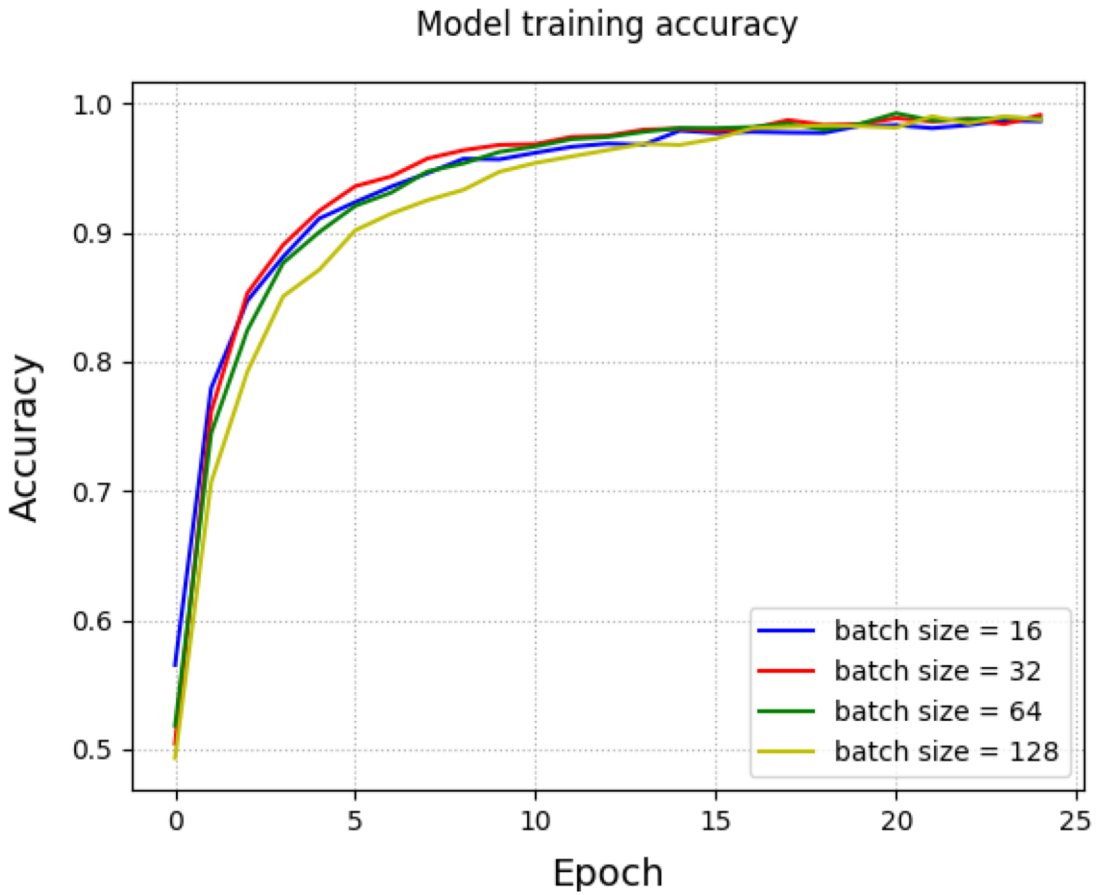


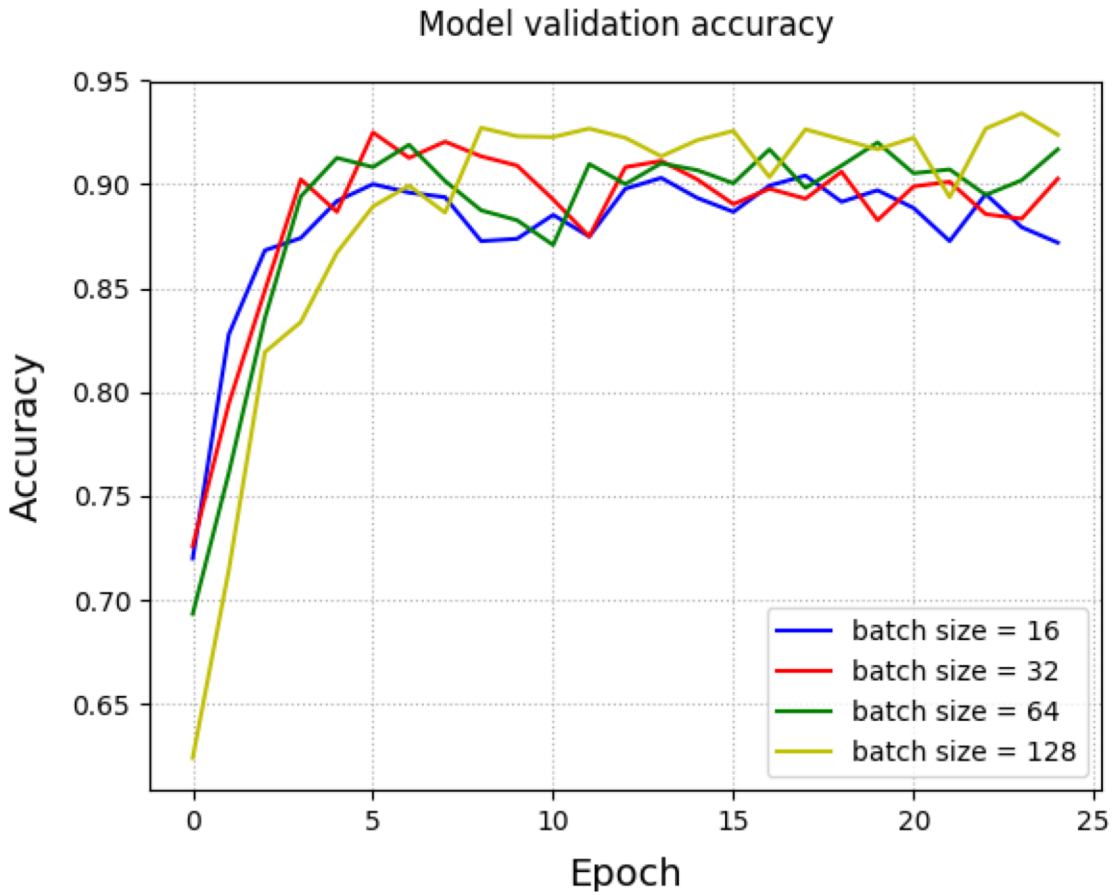
from64_1MoreConv Model



from64_1LessConv Model

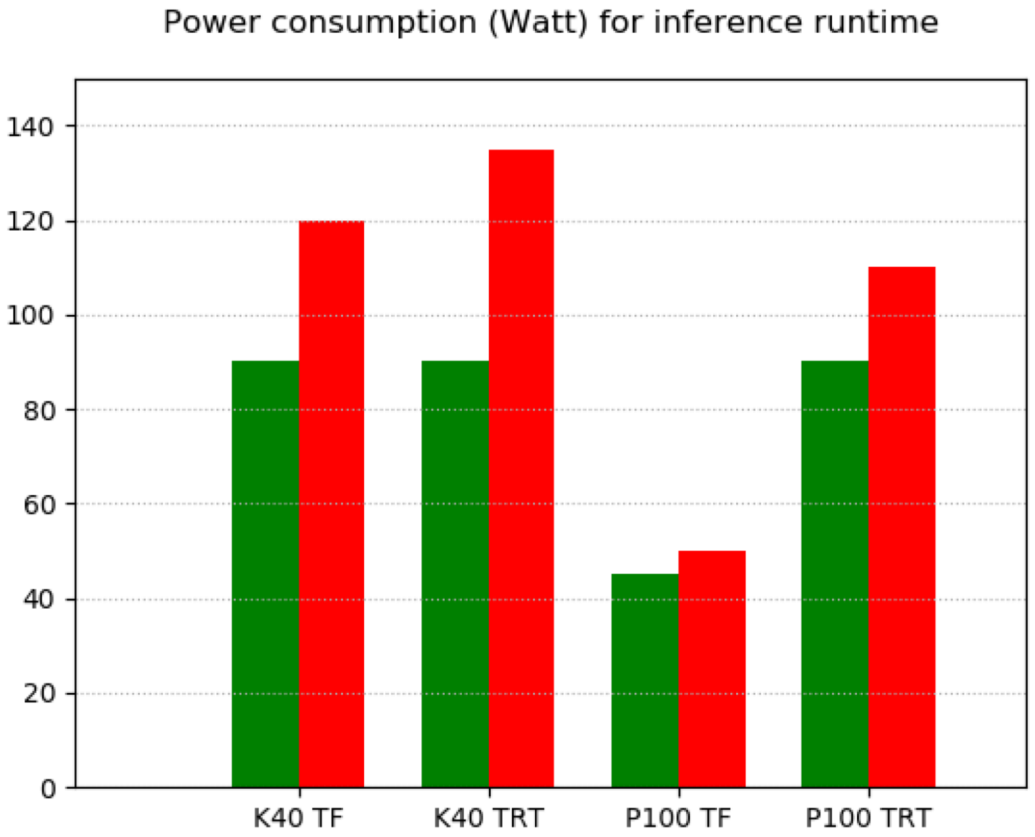
Studying the batch size changes





The **Basic Model** was trained with different values of batch size. As expected a smaller batch size results in slower training and provides a significant overfitting with respect to a bigger batch size.

GPU power consumption



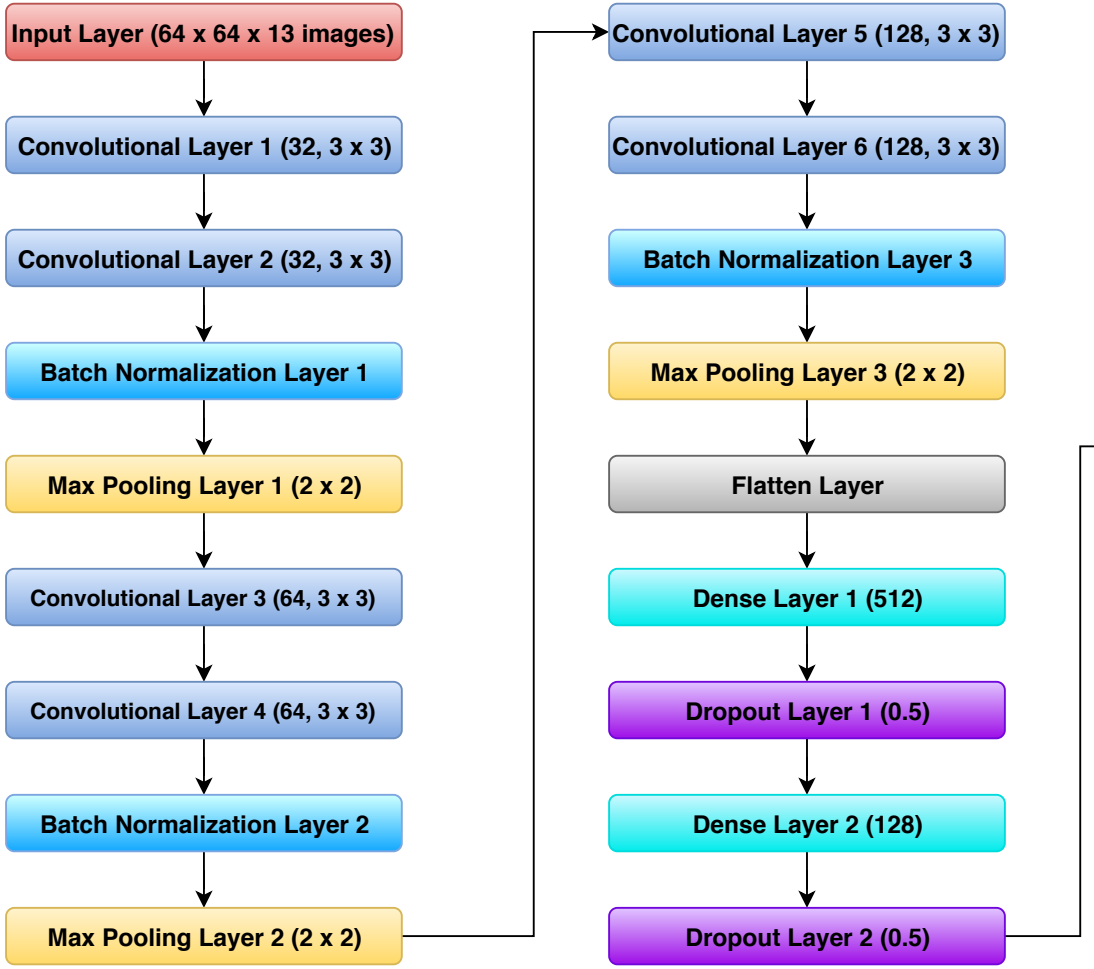
Green: minimum consumption

Red: maximum consumption

	NVIDIA Tesla K40	NVIDIA Tesla P100
Nominal maximum Power consumption	235 W	250 W
Temperature at rest	16 °C	37 °C

	NVIDIA Tesla K40		NVIDIA Tesla P100	
	TensorFlow	TensorRT	TensorFlow	TensorRT
Power consumption	90-120 W	90-135 W	45-50 W	90-110 W
Utilization	43-60 %	80-100 %	25-30 %	55-75 %
Temperature	20 °C	19 °C	38 °C	40 °C

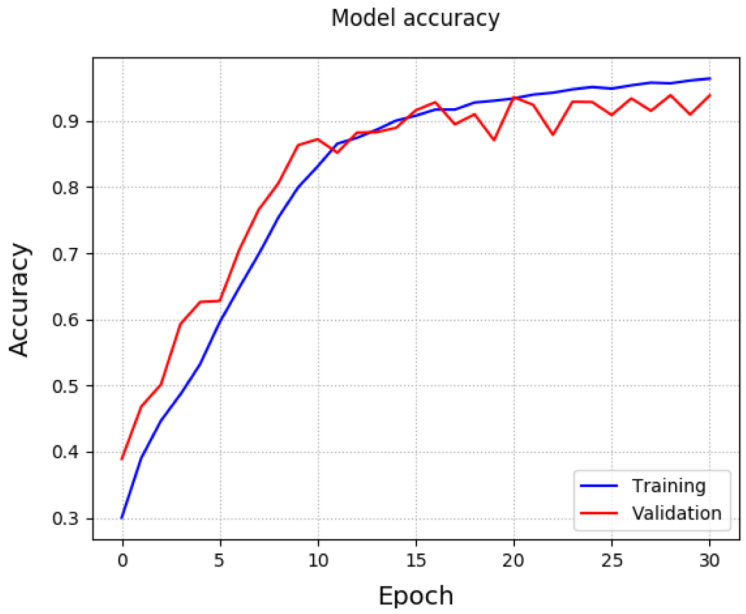
Regularizing the model



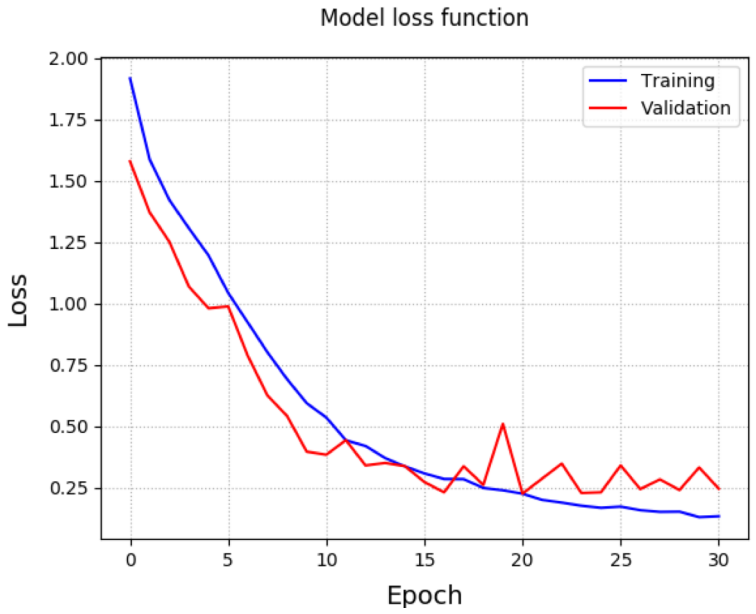
A new model has been tested, to solve the overtraining problem

- Added a **Batch Normalization Layer** after each Convolutional Stack, before the Max Pooling operation (it also speeds-up the training phase)
- Added a **Dropout Layer** after each Dense Layer, shutting down half of its neurons and avoid overtraining

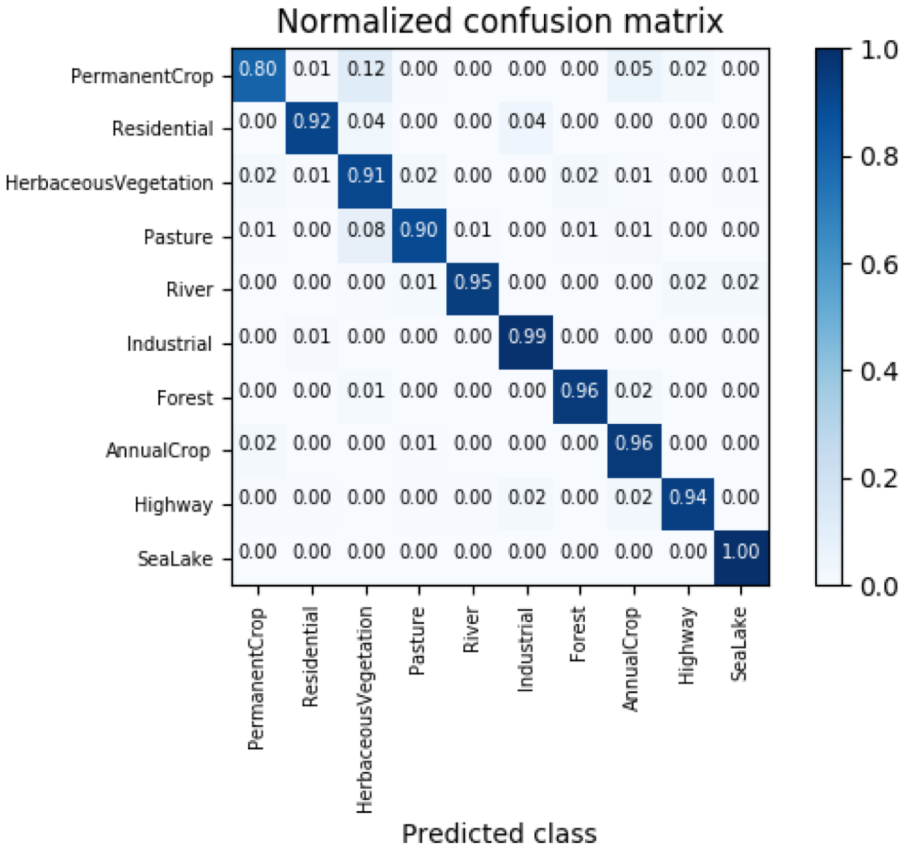
Regularized model performance



The **accuracy** on validation data is as high as on training data: overtraining solved!



The **loss function** converges to a similar value for both training and validation data



The CNN is still misclassifying ***PermanentCrop*** on the 20% of the test data

Questions?

Backup

Sentinel-2 additional info

- High-resolution optical imagery
- 2.4 Tbit onboard mass memory
- High-speed X-Band terminal achieving data rates up to 560 Mbit/s
- Geometric rivisit time: 5 days from two-satellite constellation (at equator)

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

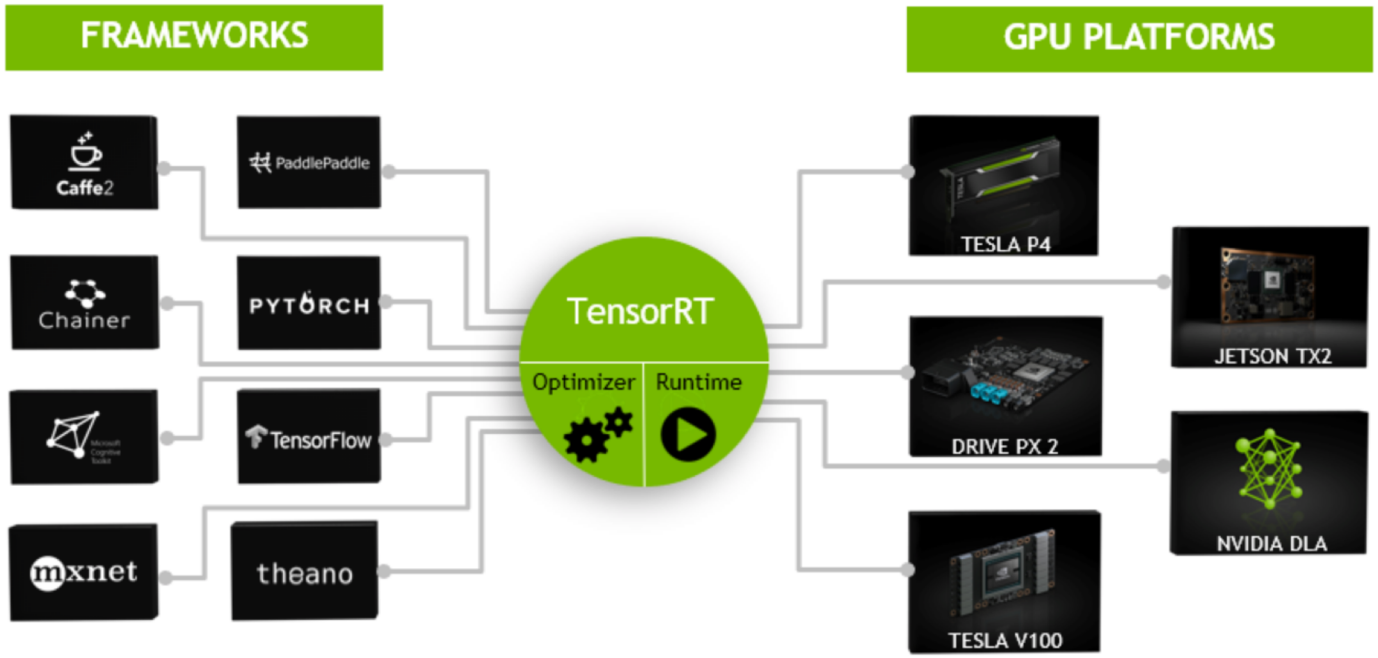
Earth Observation applications

- Atmospheric studies and climate change monitoring
  - Ice melting
  - Air quality and atmospheric composition
  - UV and Solar Radiation Data analyses for health, agriculture and renewable energies
- Land and marine environment monitoring
  - Soil sealing
  - Urban sprawl
  - Subsidences
  - Coastal seawaters quality
- Emergency and security services
  - Border and maritime surveillance
  - Natural or man-made disasters (earthquakes, volcanic eruptions, floods, forest or wild fires, tsunamis, storms, avalanches…)

NVIDIA TensorRT

NVIDIA TensorRT

- Built on CUDA
- C++ and Python APIs available
- up to 40x faster than inference on CPU



Weight & Activation Precision Calibration
Maximizes throughput by quantizing models to INT8 while preserving accuracy

Layer & Tensor Fusion
Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

Kernel Auto-Tuning
Selects best data layers and algorithms based on target GPU platform

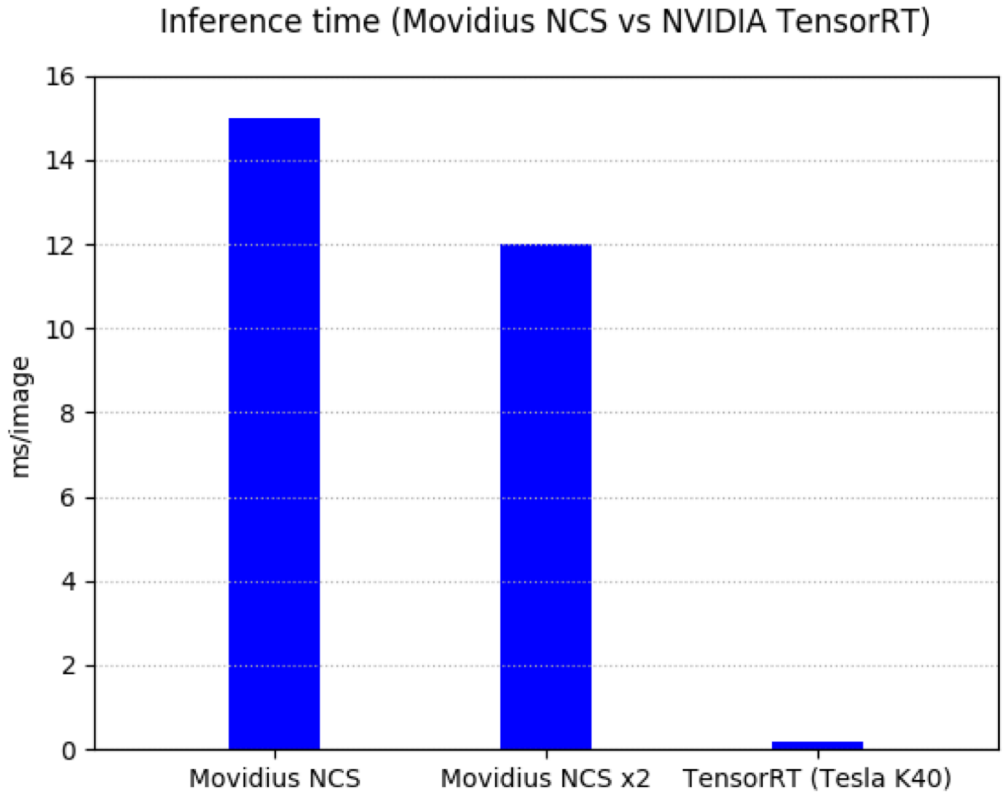
Dynamic Tensor Memory
Minimizes memory footprint and re-uses memory for tensors efficiently

Multi-Stream Execution
Scalable design to process multiple input streams in parallel

NVIDIA TensorRT vs Intel Movidius NCS

Intel Moviudius Neural Compute Stick

- Plug and play simplicity
- Development for high-performance, low-power devices
- Powered by the Intel Movidius Vision Processing Unit (VPU)



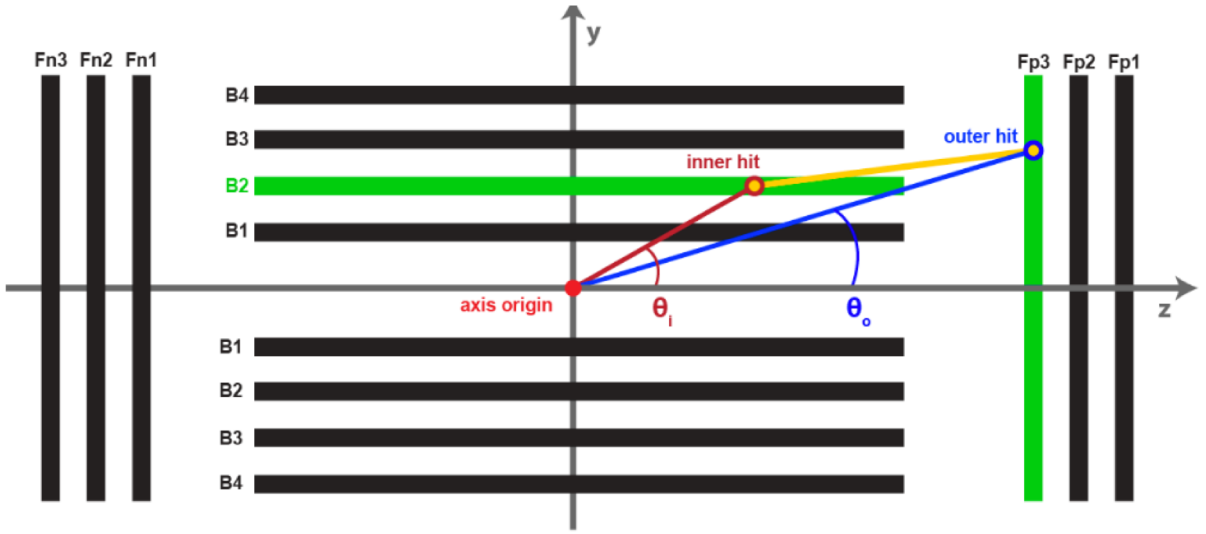
Intel Movidius NCS: 15 ms/image
Intel Movidius NCS x2: 12 ms/image
NVIDIA Tesla K40: 0.3 ms/image

PhD training @CERN: fast inference on CMS data

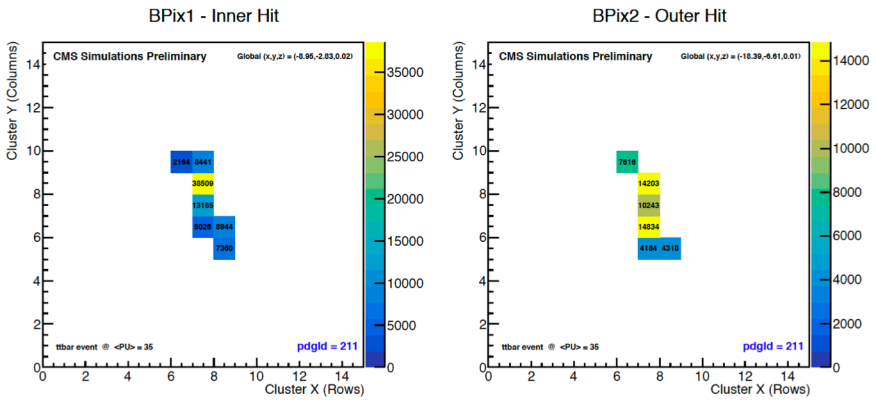
Task: testing fast inference on CMS-Tracker doublets

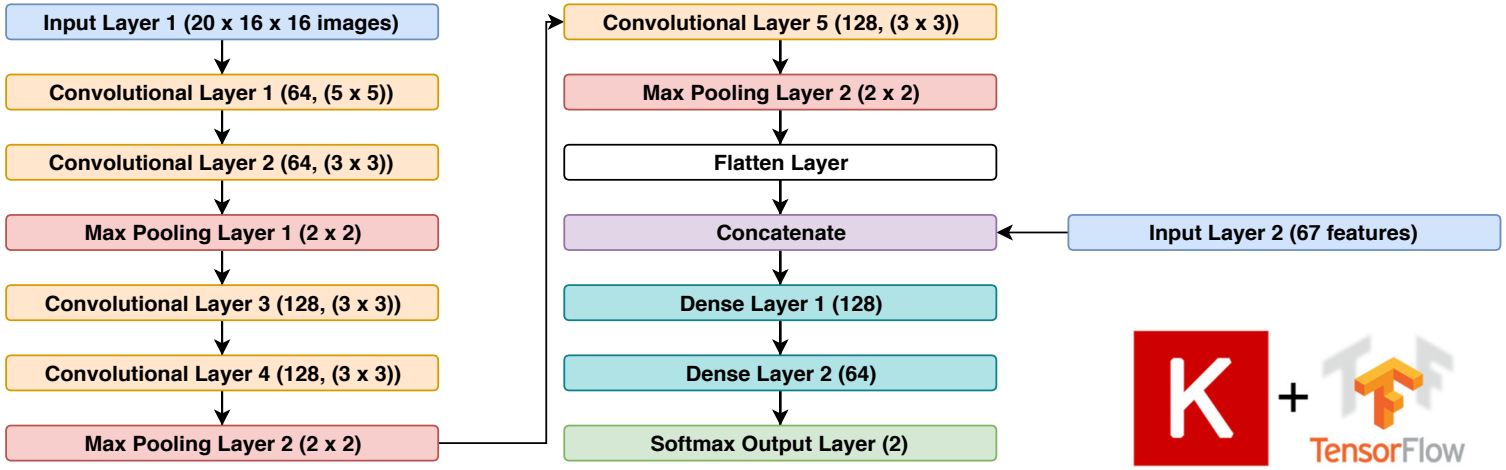
Input: Images 16 x 16 x 20 + set of local information (x, y, z, charge, …)

Output: identify the doublet (true or fake)



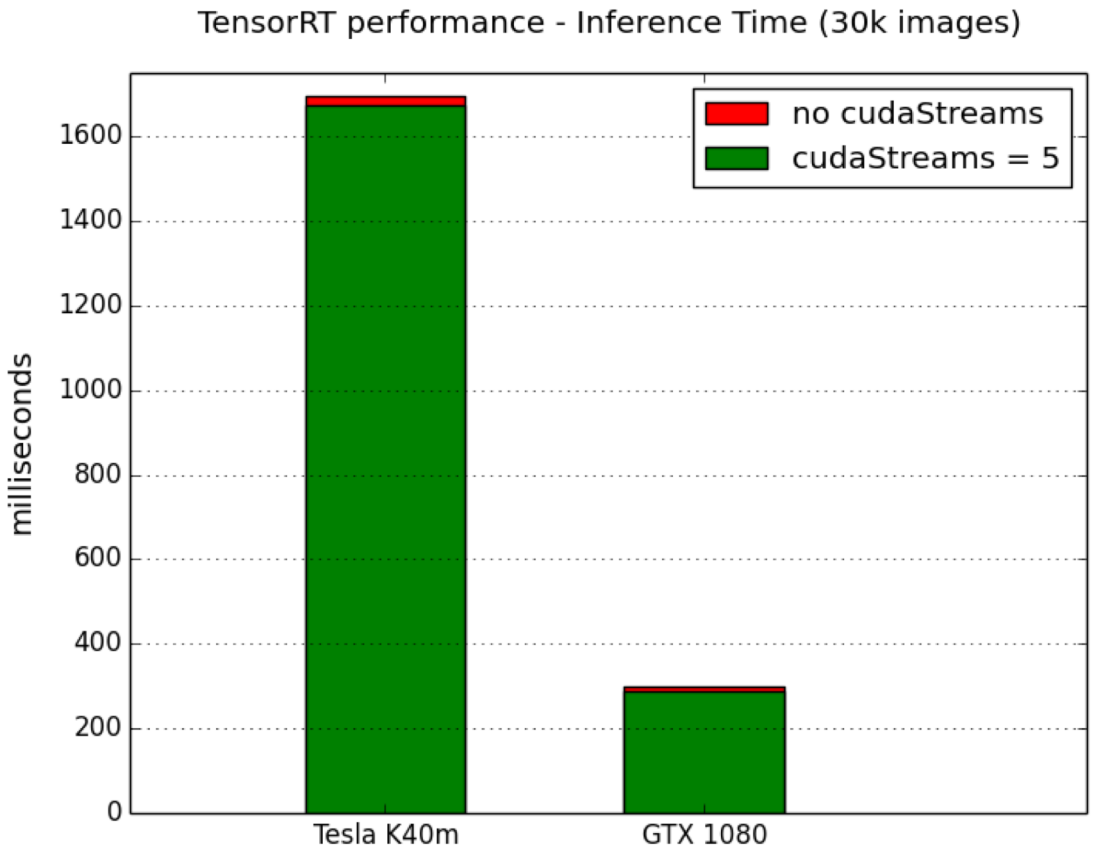
Model: Convolutional Neural Network





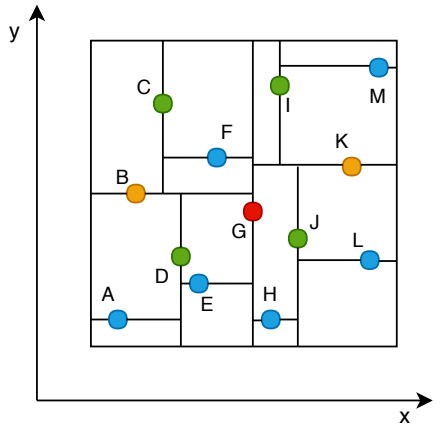
PhD training @CERN: fast inference on CMS data

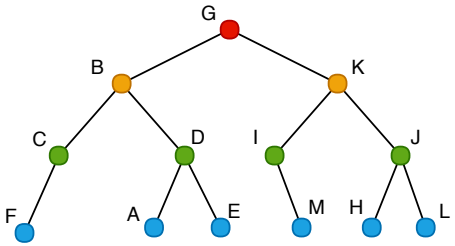
- TensorRT performance studied on **NVIDIA Tesla K40** and **GTX 1080**
- GPU resources are fully exploited by TensorRT to maximize the performance
  - Apparently, using more cudaStreams don't provide any gain in form of concurrency (it's still worth it to create more cudaStreams with respect to CPU performance)

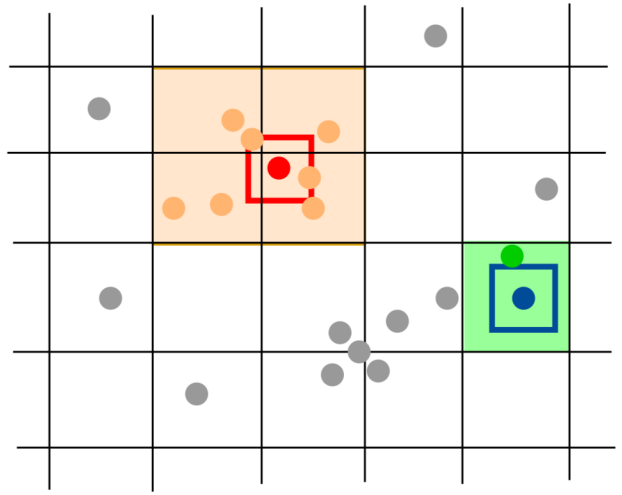


PhD training @CERN: parallel clustering in HGCAL

- The 2D clustering algorithm has been redesigned to be more GPU-friendly
- Previous 2D clustering algorithm:
  - **KDTree** data structure
  - Highly relied on sorting
  - To query d_c-neighborhood of point L, need to follow the full branch G-K-J-L
- New 2D clustering algorithm:
  - **Tiled** data structure
  - No more sorting
  - To query d_c-neighborhood of point *i*, only need to go to tiles touched by box with $(i.x \pm d_c, i.y \pm d_c)$ shown in red and blue squares.



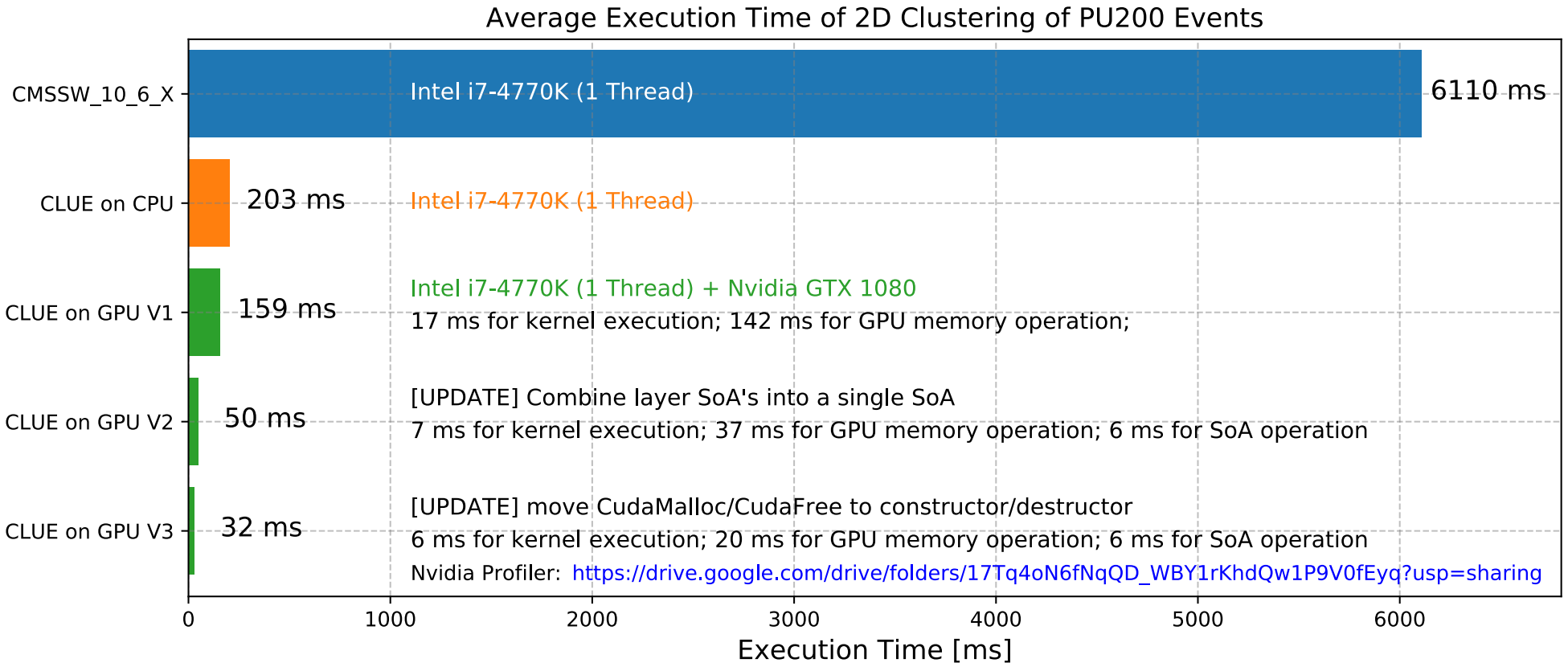




PhD training @CERN: parallel clustering in HGCAL

- GPU Version 1 (Mid May, 2019)
  - direct GPU implementation of CLUE-CPU.
  - represent rechits as SoA.
  - still perform CLUE layer-by-layer.
- GPU Version 2 (Mid June, 2019)
  - cudaMemcpy all rechits on all layers in an event.
  - launch 5 CUDA kernels once per event, processing layers in parallel.
  - still allocate and free GPU memory once per event.
- GPU version 3 (End June, 2019)
  - latest version.
  - allocate GPU memory (cudaMalloc) at the beginning of job in class constructor.
  - free GPU memory (cudaFree) at the end of job in class destructor.
  - 1.9 GB of GRAM is actually used in tests.
- All three versions produce exactly the same clustering result as CPU version in CMSSW_11_0_0.

PhD training @CERN: parallel clustering in HGCAL



- CLUE CPU has about 30X speed up over CMSSW_10_6_X
- CLUE GPU V3 gives an additional 6X over CLUE CPU

PhD training @CERN: PID and energy regression

Main task: Create a model to **predict** the particle ID and energy, given HGCal TICL (**T**he **I**terative **CL**ustering) output (a set of layer clusters).

Input: image size 50 x 30 x 3

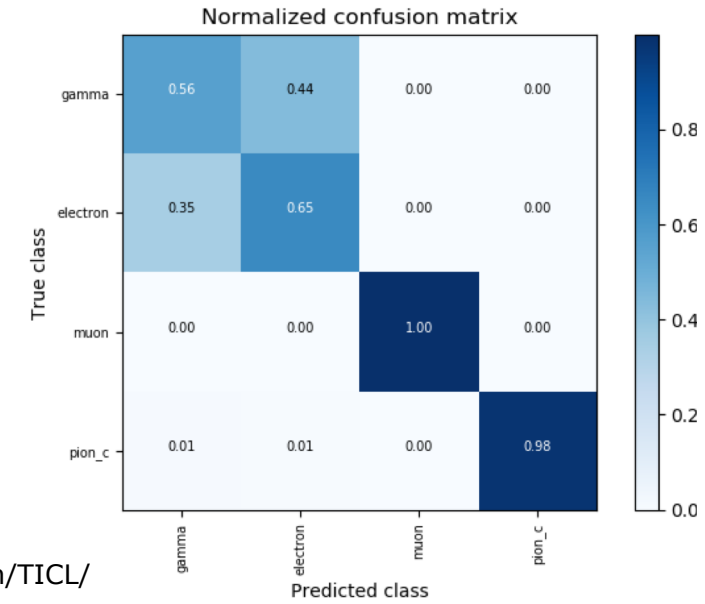
➢ 50 = number of HGCAL layers

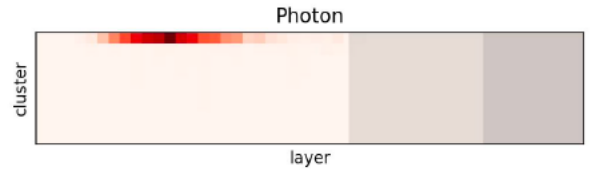
➢ 30 = maximum number of LayerClusters on the same layer

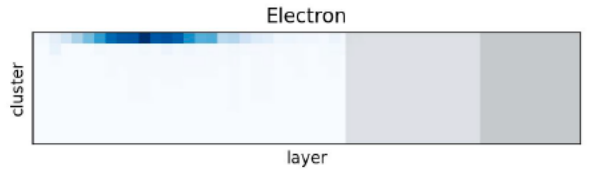
➢ 3 = eta, phi, energy of LayerClusters

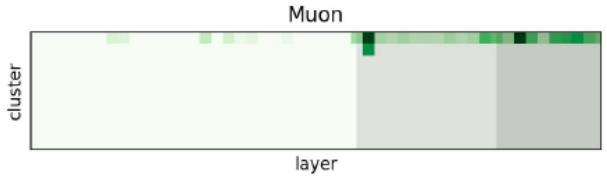
Output: 4 classes + energy value

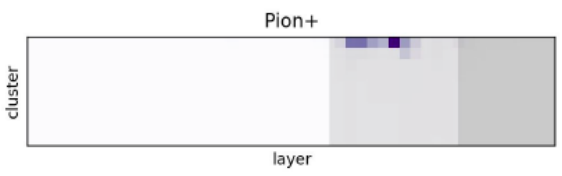
Model:
Convolutional Neural Network

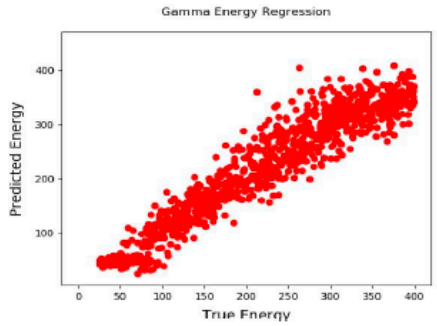


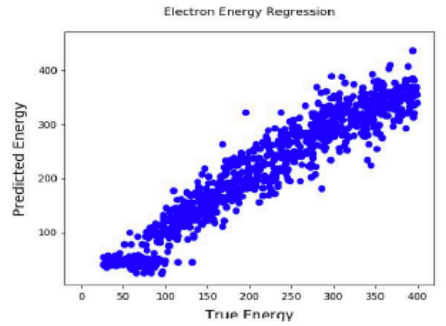


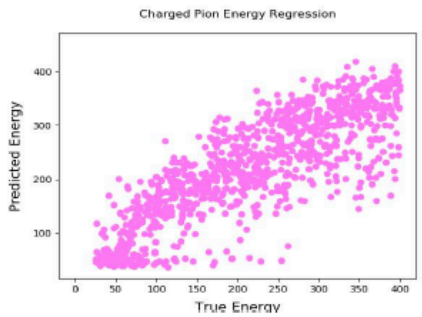


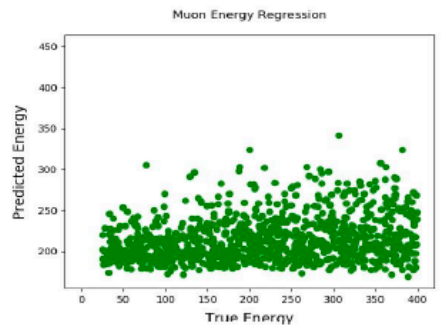












* http://hgcal.web.cern.ch/hgcal/Reconstruction/TICL/