Achieving Predictability in the Execution of Deep Neural Networks in Safety Critical Applications

Daniel Casini, Alessandro Biondi, Giorgio Buttazzo

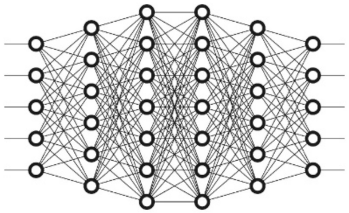
ReTiS Lab, Scuola Superiore Sant'Anna, Pisa, Italy

Sant'Anna
Scuola Universitaria Superiore Pisa

Retis
Real-Time Systems Laboratory

This Talk

Motivations



A case study

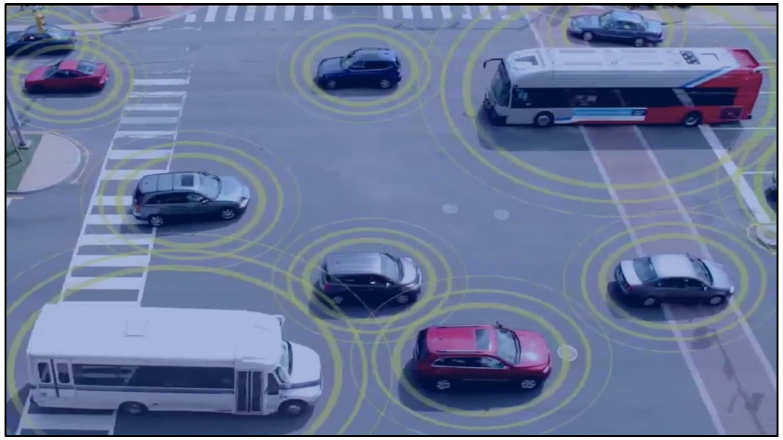
Understanding the workload generated by a Deep Neural Network

Our work

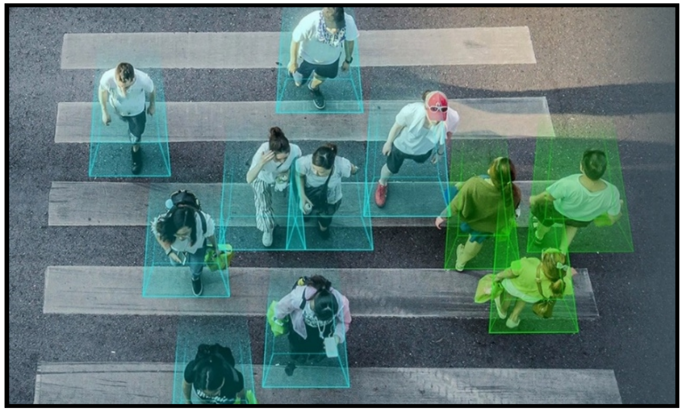
Timing isolation for DNN and real-time tasks executed on a multiprocessor platform

DNNs are everywhere

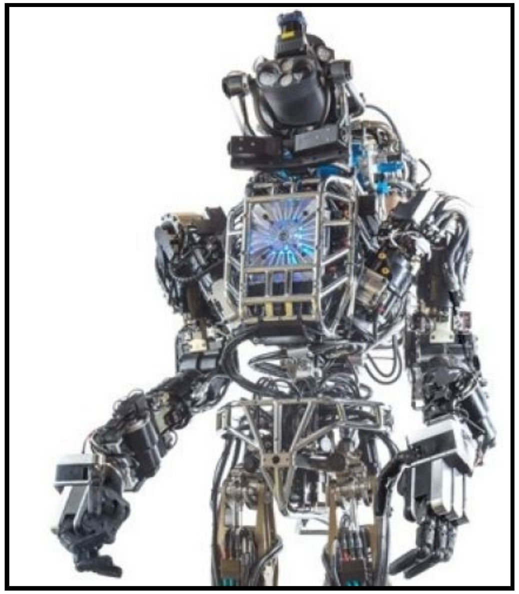
Autonomous Driving



Surveillance



Advanced Robotics



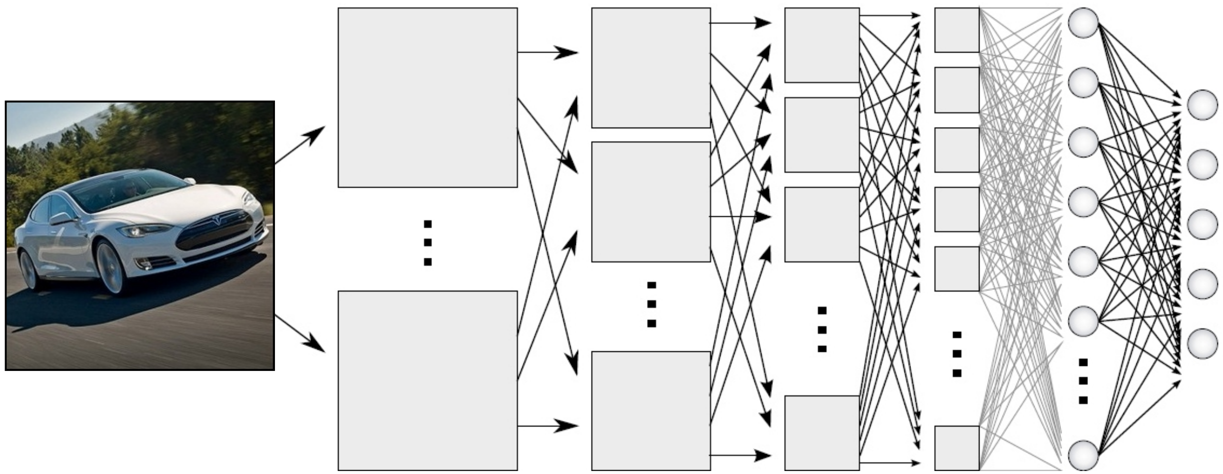
Healthcare

Image Recognition

- The ILSVRC Challenge is a competition held from 2010 in which networks compete in classifying objects from images to labels, with 1000 possible categories

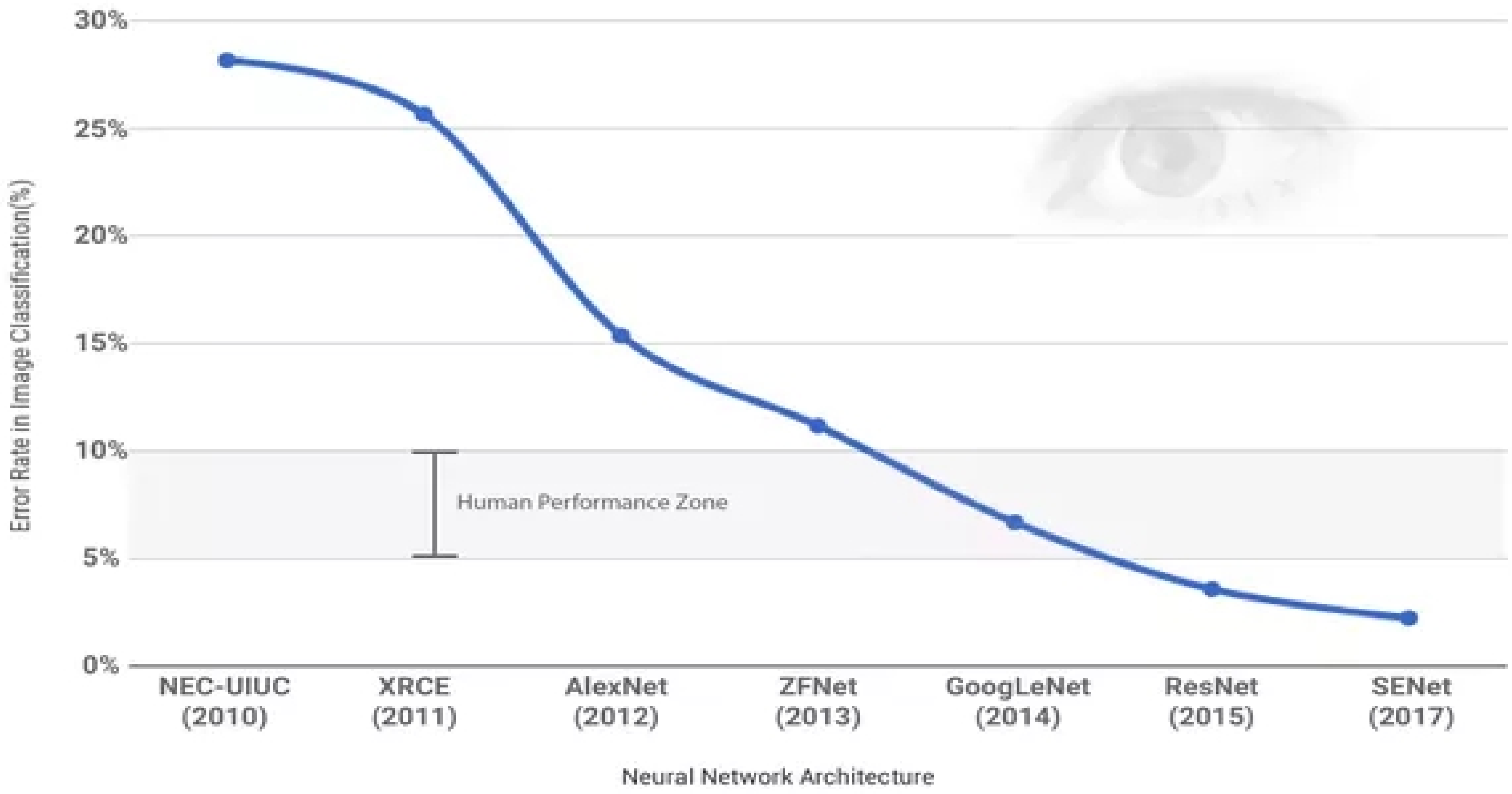
Training set: 1.2 million images (1,000 categories)

Test set: 150,000 images



Are DNNs good enough?

The winning network of 2017 (SENet), achieved an accuracy of 97.74%



Source: http://blog.paralleldots.com/data-science/must-read-path-breaking-papers-about-image-classification/

How to achieve predictability in the execution of DNN workload?

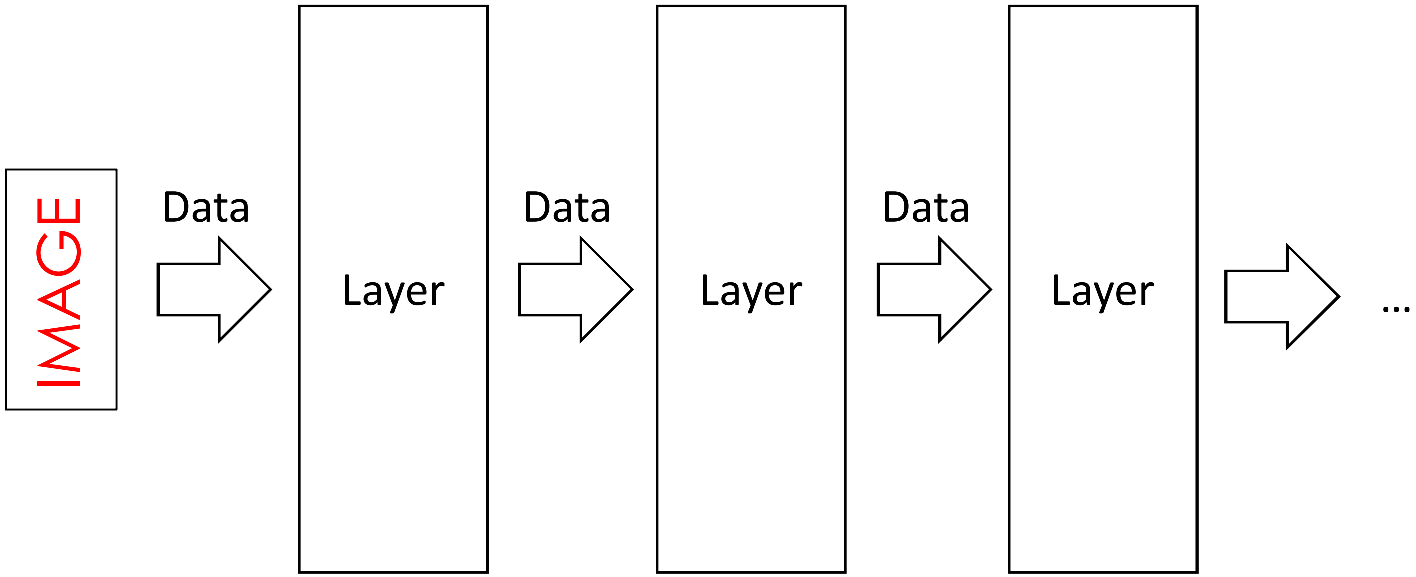
Understanding Complex DNN Workload

- What is a suitable model for the workload generated by complex DNNs?
- What is the resource consumption?
- Case study
  - InceptionV3: powerful image recognition DNN
  - Tensorflow: open-source machine learning framework by Google

- Stock Tensorflow with `eigen` math library on CPUs
- Strongly parallel workload with two levels of parallelism

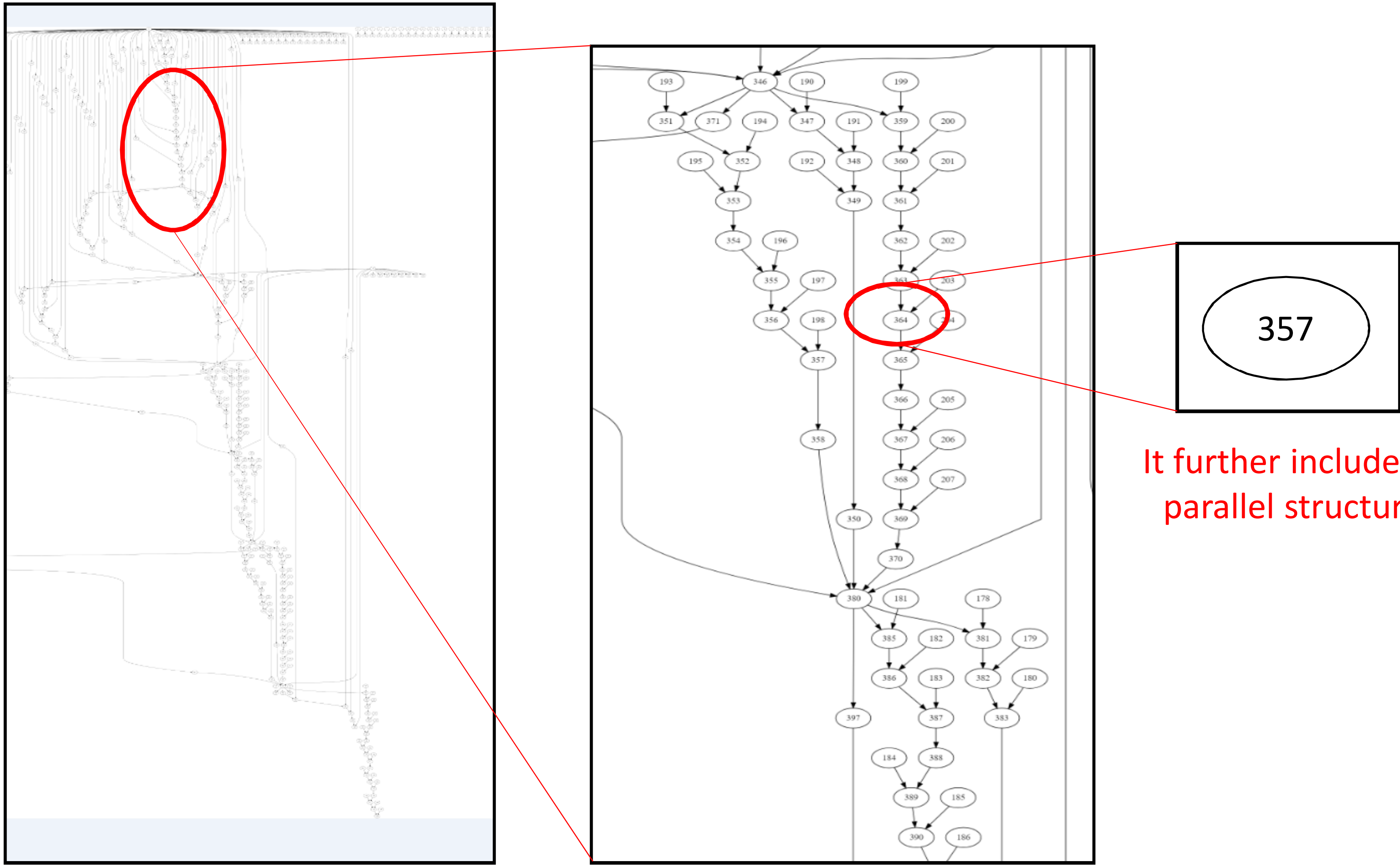
Understanding Complex DNN Workload

- A DNN is composed of a pipeline of layers, where each one implements an operation
- Key issue when it is used in a critical system: guaranteeing that a real-time workload composed of DNNs completes within a deadline (*inference* phase only)



InceptionV3 on Tensorflow

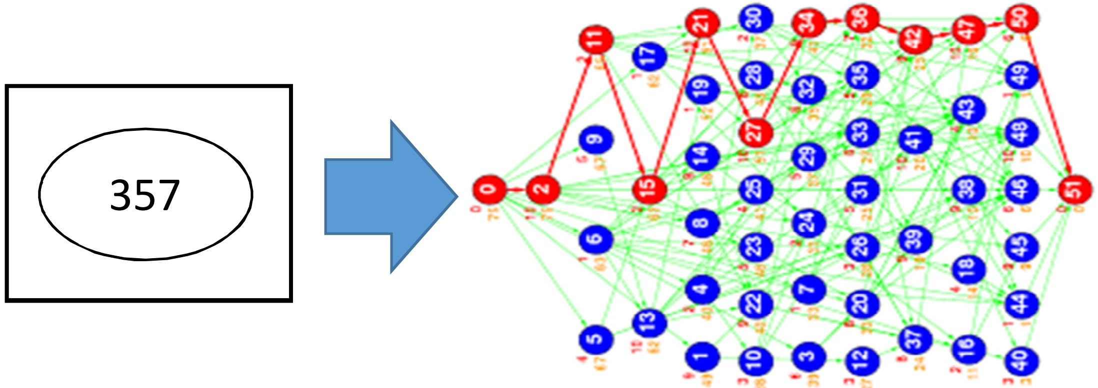
- First level is represented by a complex graph (~700 nodes)



It further includes a parallel structure

InceptionV3 on Tensorflow

- Nodes typically correspond to mathematical computations (e.g., tensor convolutions) whose implementation is platform-dependent and extremely parallel – this is the *second level* of parallelism

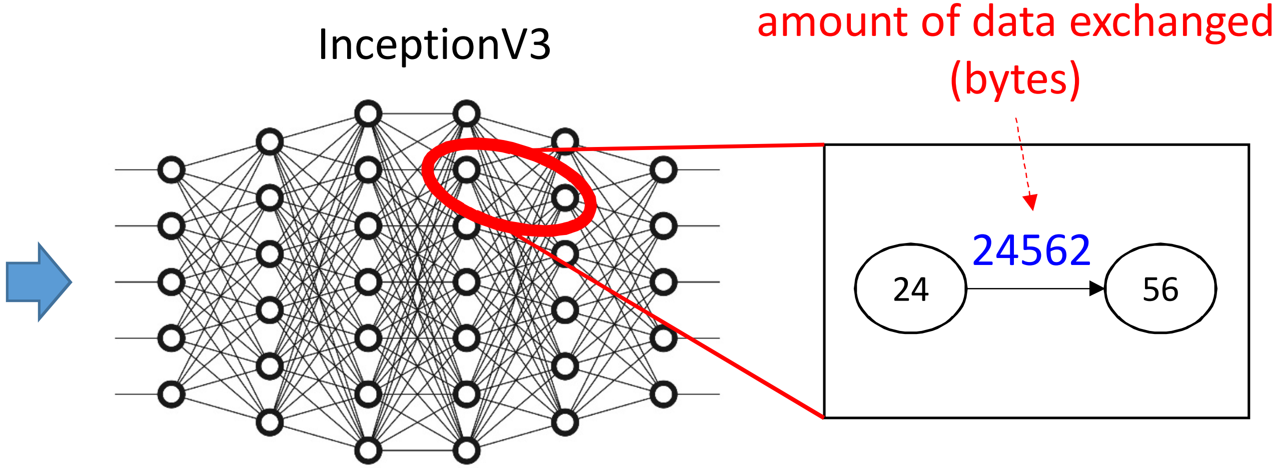


- InceptionV3 on a 8-core Intel i7 machine @ 3.5GHz
  - **More than 34000 nodes** (!) where only about **1.2%** of them have execution times larger than 100 microseconds
  - **Complex** workload with **extreme parallelism** → difficult to understand and manage

InceptionV3 on Tensorflow

- Nodes of the parallel tasks exchange a lot of data

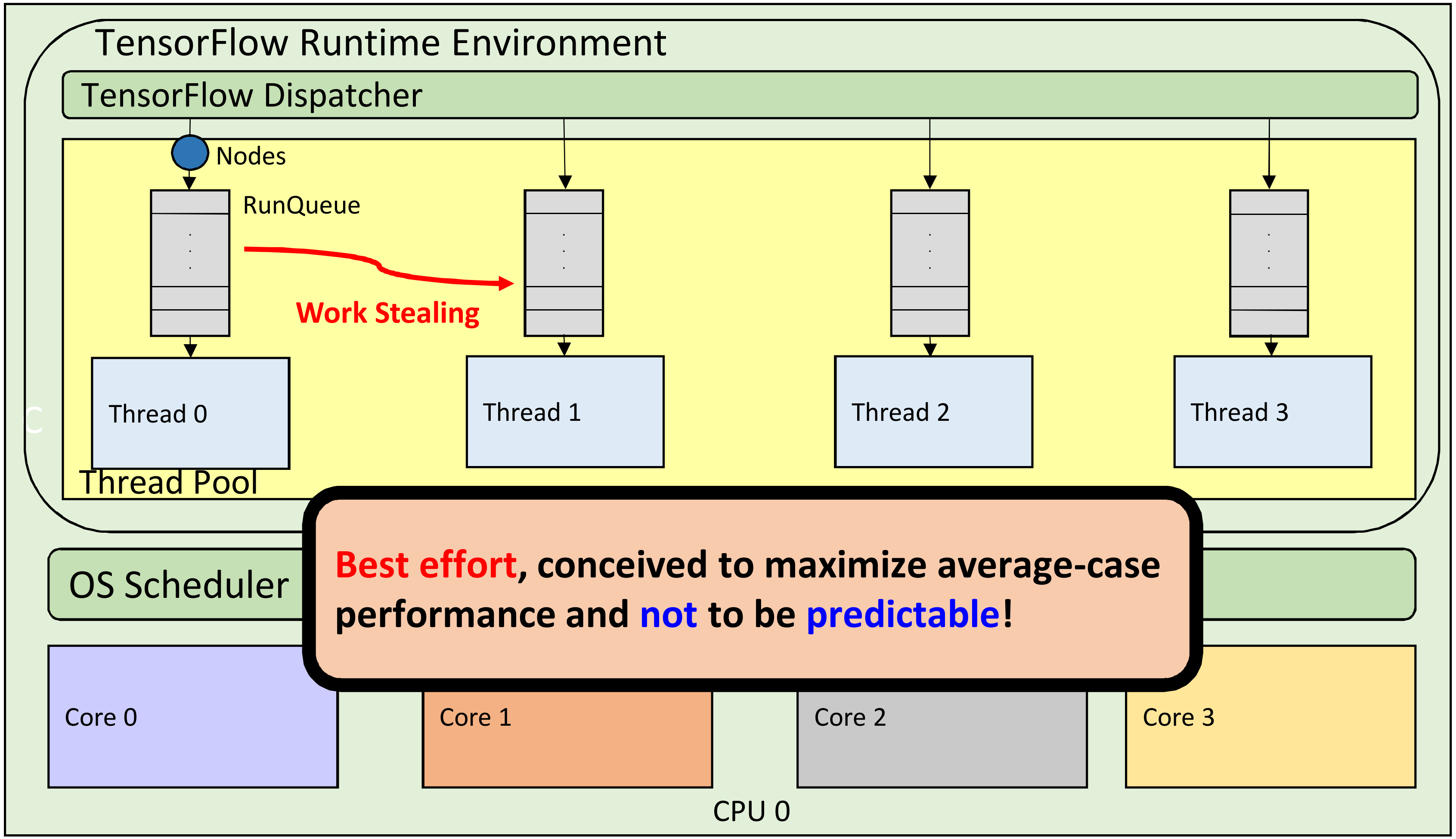
Input image (**60Kb**)



Total of data exchanged by nodes at the first level of parallelism amounts to **603856960 bytes ~= 604 Megabytes**

How Tensorflow works on CPUs?

- TensorFlow assigns ready nodes to threads of a thread pool, one for each level

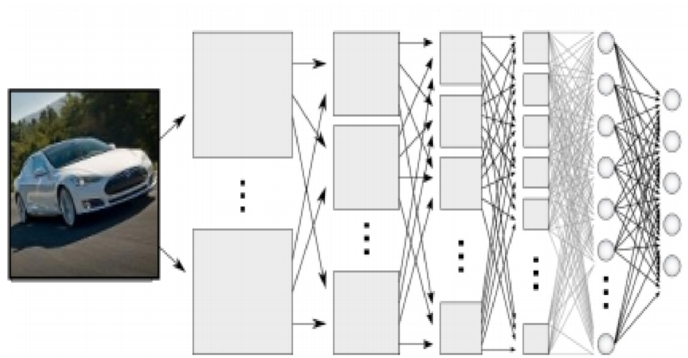


Best effort, conceived to maximize average-case performance and not to be predictable!

Predictable DNN Engines

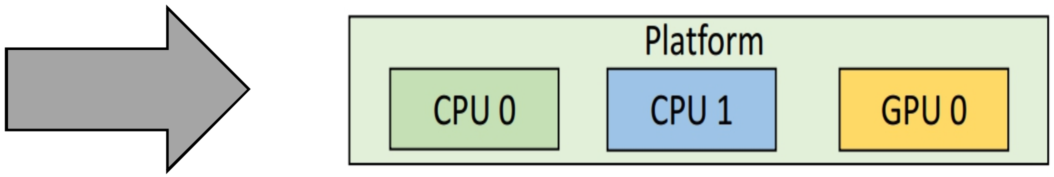
- Deep Neural Network (DNN) engines are typically conceived for best effort applications
  - No support for execution predictability
  - Prone to attacks and malfunctioning

- Need for improved inference engines to support predictable computing
  - Real-time scheduling and memory management, predictable allocation
  - Isolation to contain & control memory contention



DNN

Execution & Mapping



TensorFlow™

Caffe

TensorFlow for Safety-Critical Systems?

- TensorFlow is a complex software written in C++
- Large usage of dynamic memory
- Large usage of complex and advanced features of C++
- No safety programming guidelines are followed
- Wide usage of pointers
- …

Far from safety-critical software standards (e.g., ISO26262):

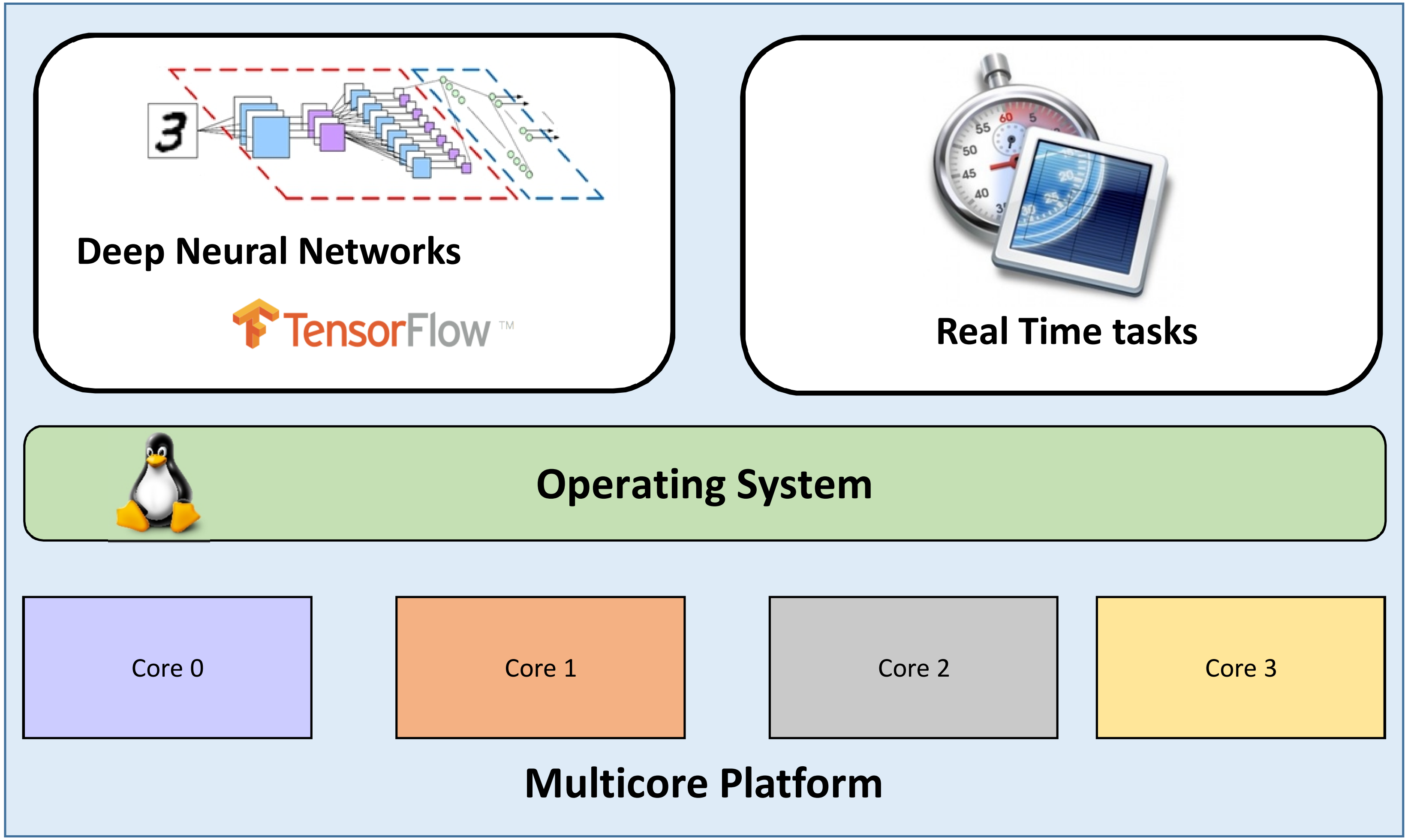
- **Static** memory
- Mandatory programming guidelines
- Limited use of pointers
- …

Large effort is required to make it usable in a safety-critical environment

Ongoing work

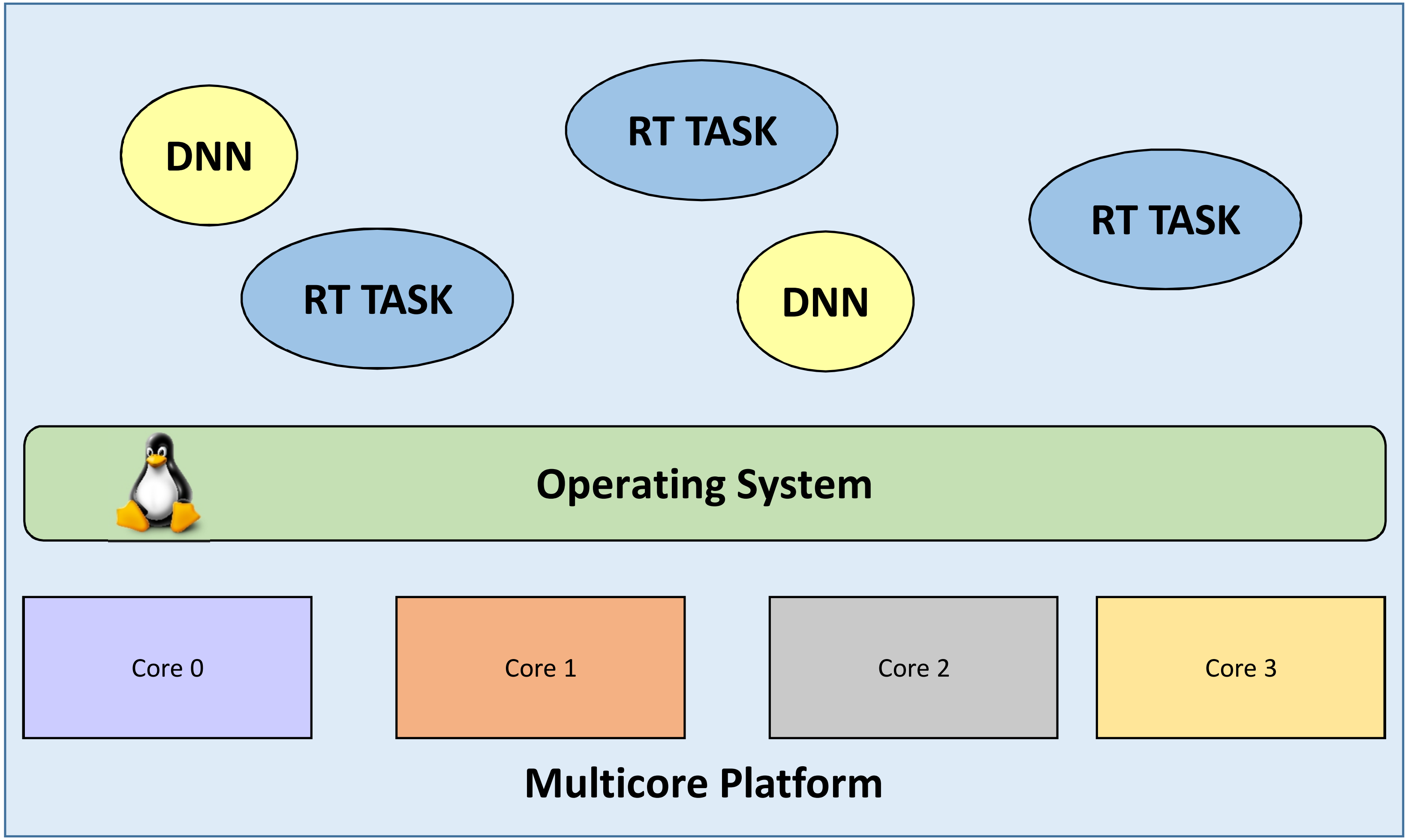
Scenario

Workload composed of DNN and regular real-time tasks



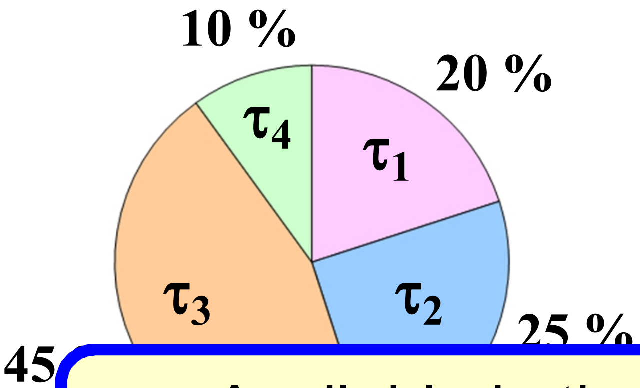
Scenario

but unfortunately they can interfere each other…



Resource Reservation

Resource partition



Resource enforcement

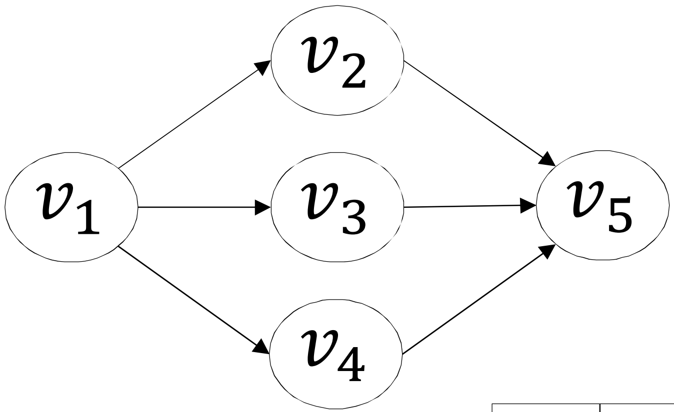
Available in the **SCHED_DEADLINE** scheduling class of **Linux**

Each task subset is assigned a fraction α_i of CPU bandwidth and behaves as it were executing alone on a slower processor of speed α_i

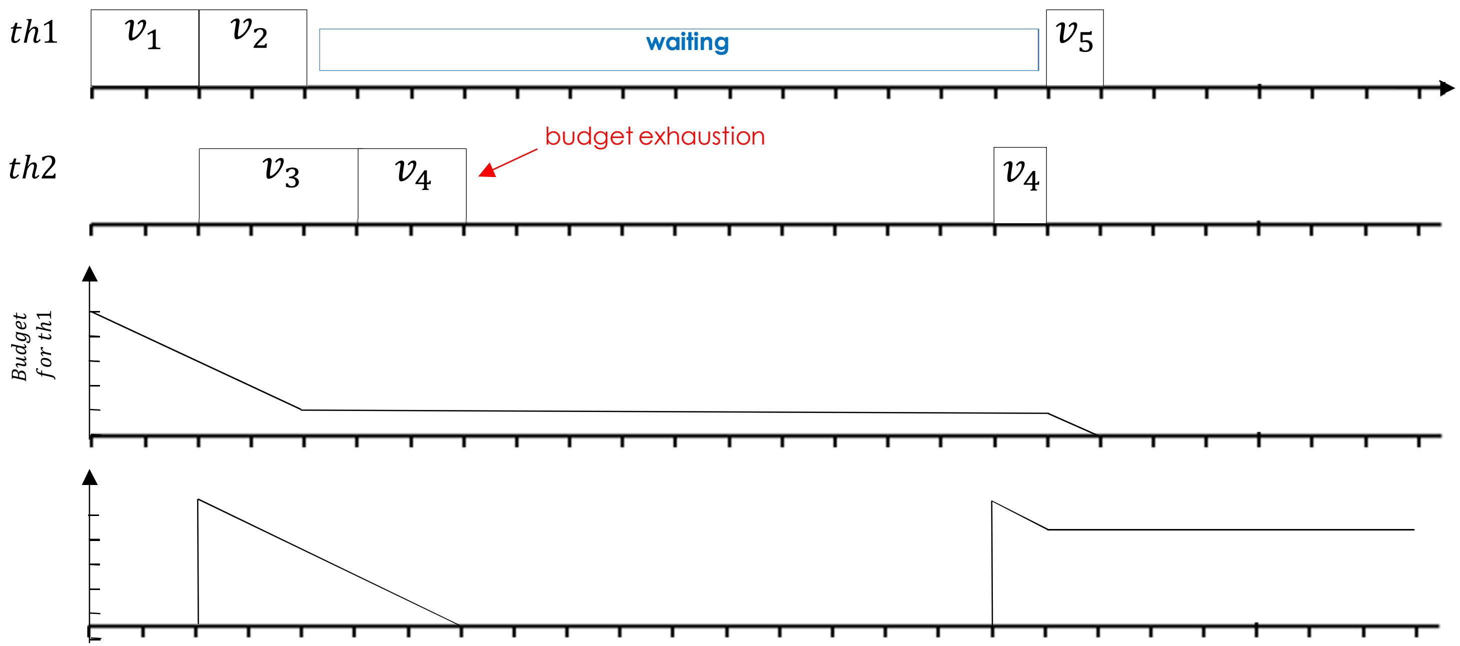
- … prevents … more than its reserved amount.
- If a task executes more, it is delayed, preserving the resource for other tasks.

Problem

- The usage of resource reservation when the workload is subject to precedence constraints can cause performance degradation



Budget exhaustion prevents successor nodes from being spawned



Our work

- Reduce the performance hit due to the usage of resource reservation by allowing budget overrun

Nodes are typically very small, allow them overrunning (with payback) can improve performance

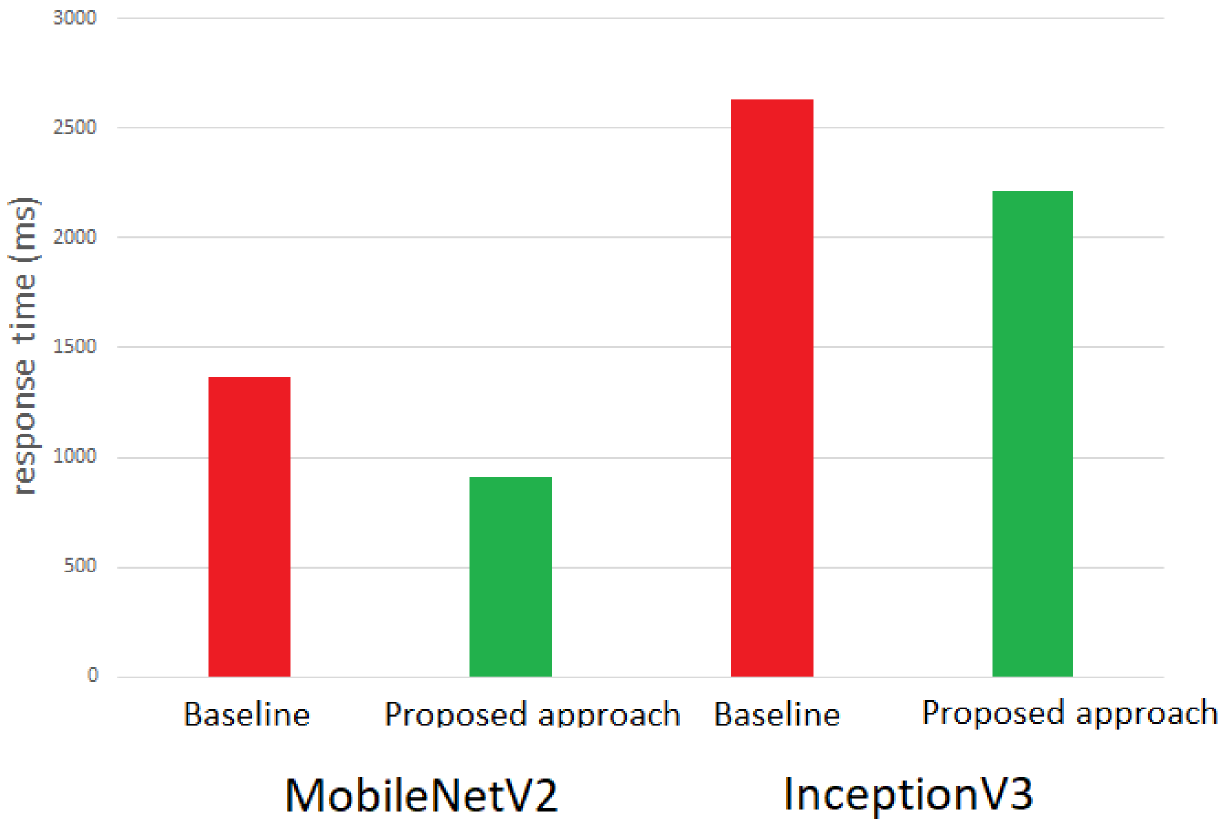
- Exploit **data locality** of DNN workload by implementing localized stealing

Nodes exchange a considerable amount of **memory:** Steal work from neighboring cores to take advantage of shared levels of cache

Some results

8-core Intel i7 running at 3.50Ghz, Tensorflow v1.5

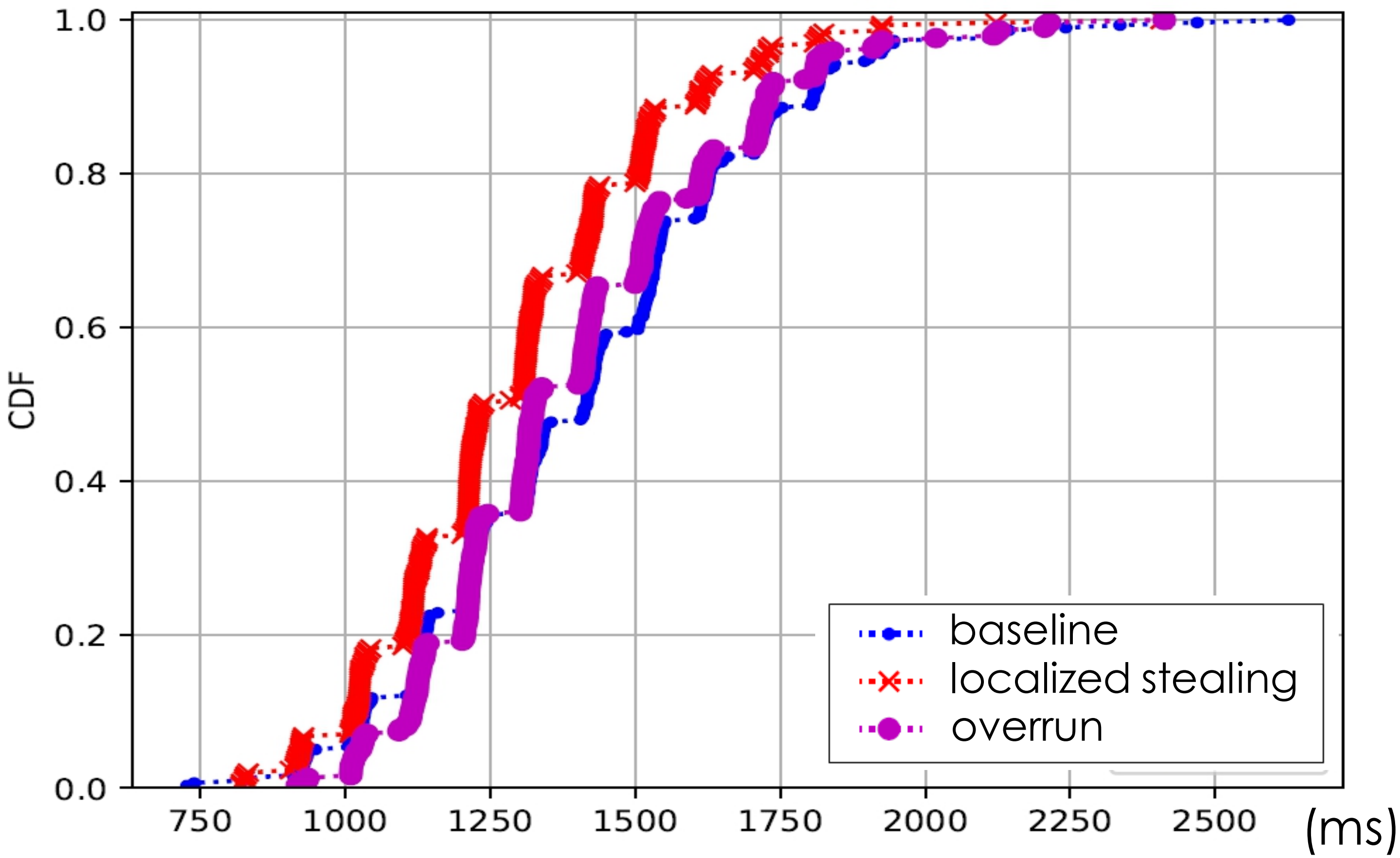
2 DNNs + real-time tasks



Some results

8-core Intel i7 running at 3.50Ghz, Tensorflow v1.5

2 DNNs + real-time tasks



InceptionV3

Thank you!

Daniel Casini
daniel.casini@sssup.it