Introduction to Deep Neural Networks

Georg Zitzlsberger, IT4Innovations

29.06.2023

Univerza v Ljubljani

TU WIEN TECHNISCHE UNIVERSITÄT WIEN

CINECA

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Co-funded by the Erasmus+ Programme of the European Union

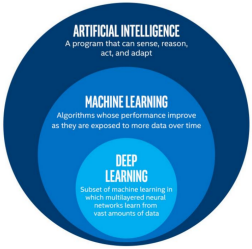
This project has been funded with support from the European Commission.
This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Agenda

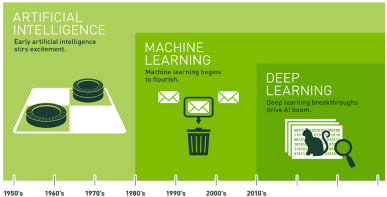
Georg Zitzlsberger, IT4Innovations

- Introduction
- New Generation Silicon
- Neuronal Networks
- Optimizations for Inference
- Summary

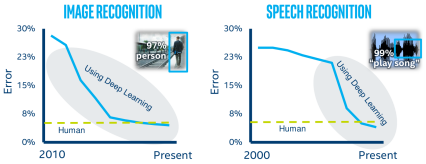
Introduction



(Image: Intel)



(Image: NVIDIA)



(Images: Intel)

Only very brief:

- 1950: Perceptron
  - By Frank Rosenblatt (funded by US Office of Naval Research)
  - 20x20 input photocells
  - Electro-mechanic
  - Not capable enough for multi-class patterns (only one layer)
- First AI winter (1974-1980)
- 1989: Yann le Cun's *Theoretical Framework for Back-Propagation*
- Second AI winter (1987-1993)
- 2012: Dawn of Deep Neuronal Networks with *AlexNet*
- What's next? AI winter or Singularity?

$(f_1, f_2, \ldots, f_K)$

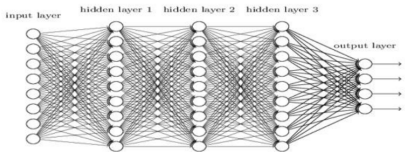
$K \ll N$

!! Takes lot of time

SVM
Random Forest
Naïve Bayes
Decision Trees
Logistic Regression
Ensemble methods

Arjun

(Image: Intel)



Arjun

Features are discovered from data

Extract features at multiple levels of abstraction

Performance improves with more data

High degree of representational power

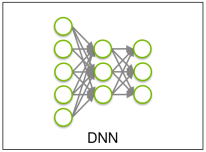
But old practices apply:
Data cleaning, Exploration, Data annotation, hyper-parameters, etc.

(Image: Intel)

- Pro Machine Learning (ML):
  - Best control over feature space
  - Preference if mathematical models exist that can be expressed directly
  - Guarantee of best solution (e.g. SVM with convex kernels)
  - Can work with less data

- Pro Deep Learning (DL):
  - Tackle complex problem spaces w/o feature engineering
  - Ensemble of networks possible
  - Allows to process large amounts of (i.i.d.) data
  - Easy to use across multiple nodes/GPUs

New Generation Silicon



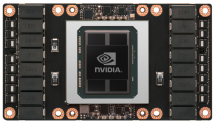
(Image: NVIDIA)

Why now?

- Big Data:
  Large amounts of data are available
- Recent Deep Network Development:
  New Deep Learning methodologies evolved (2010 onwards)
- Hardware:
  Modern systems are fast enough and have the memory needed

NVIDIA Tesla P100 (Pascal)

Georg Zitzlsberger, IT4Innovations



(Image: NVIDIA)

- Peak 4.7 TFLOPS (double precision), 9.3 TFLOPS (single precision), 18.7 (half precision)
- 12/16 GB HBM2 (CoWoS)
- Peak memory bandwidth: 549 GB/s (12 GB), 732 GB/s (16 GB)
- 3584 CUDA cores
- NVLink v1 (SXM) or PCIe x16 Gen3
- 300 (SXM) Watts, 250 Watts (PCIe)

NVIDIA Tesla V100 (Volta)

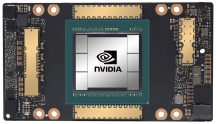
Georg Zitzlsberger, IT4Innovations

(Image: NVIDIA)

- Peak 7.8 TFLOPS (double precision), 15.7 TFLOPS (single precision), 125 (half precision)
- Up to 125 "TensorTFLOPS" for Deep Learning
- 16/32 GB HBM2 (CoWoS), Peak 900 GB/s memory bandwidth
- 5120 CUDA cores + 620 Tensor Cores
- NVLink v2 (SXM), PCIe x16 Gen3
- 300 (SXM) Watts, 250 (PCIe)

NVIDIA A100 (Ampere)

Georg Zitzlsberger, IT4Innovations

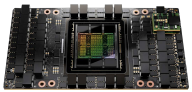


(Image: NVIDIA)

- Peak 9.7 TFLOPS (double precision), 19.5 TFLOPS (single precision), 312 (half precision)
- Up to 312 "TensorTFLOPS" for Deep Learning (624/1248 INT8/4)
- 40/80 GB HBM2 (CoWoS), Peak 1.6/2.0 TB/s memory bandwidth
- 6912 CUDA cores + 432 Tensor Cores
- NVLink v3 (SXM), PCIe x16 Gen3
- 400 (SXM) Watts, 250 (PCIe)

NVIDIA H100 (Hopper)

Georg Zitzlsberger, IT4Innovations

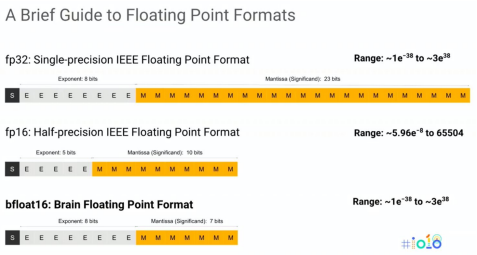


(Image: NVIDIA)

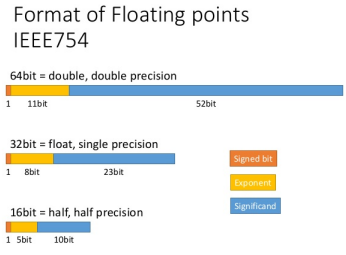
- Peak 34 TFLOPS (double precision), 67 TFLOPS (single precision), 1,979 (half precision)
- Up to 3,958 "TensorTFLOPS" for Deep Learning (INT8)
- 80 GB HBM3 (CoWoS), Peak 3.35 TB/s memory bandwidth
- 16.896 CUDA cores (FP32) + 528 Tensor Cores
- NVLink v4, PCIe x16 Gen5
- 700 (SXM) Watts, 350 (PCIe)

Excursion: Floating Point Types

Georg Zitzlsberger, IT4Innovations



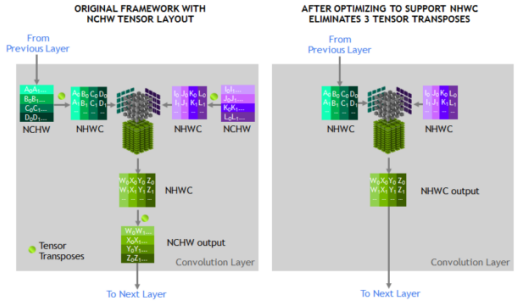
(Image: Google)



(Image: NVIDIA)

- FP32, FP16 and BFloat16[1] are of interest for Deep Learning (DL)
- FP64 is not common for DL but other Machine Learning algorithms
- Also integer types can be used (e.g. INT4, INT8)

[1]Note that we use the term "half precision" for FP16, which does not include BFloat16



(Image: NVIDIA)

More information in the blog ▸ Volta Tensor Core GPU Achieves New AI Performance Milestones

NVIDIA Tesla T4 (Turing)

Georg Zitzlsberger, IT4Innovations

(Image: NVIDIA)

- Peak 8.1 TFLOPS (single precision)
- Up to 65 "TensorTFLOPS" for Deep Learning, 130/260 INT8/4 TOPS for inference
- 16 GB GDDR6, Peak 300 GB/s memory bandwidth
- 2560 CUDA cores + 320 Tensor Cores
- PCIe x16 Gen3
- 70 Watts

- **Single Precision (FP32):**

GPU	P100	V100	T4
Max. TFLOPS	9.32	14.02	8.07
FFT TFLOPS	1.51	2.30	0.66
GEMM TFLOPS	8.79	13.48	3.29
Theoretic Peak TFLOPS	9.3	15.7	8.1

- **Double Precision (FP64):**

GPU	P100	V100	T4
Max. TFLOPS	4.74	7.07	0.25
FFT TFLOPS	0.76	1.15	0.13
GEMM TFLOPS	4.26	5.92	0.25
Theoretic Peak TFLOPS	4.7	7.8	N/A

Results published at microway.com

GPU PERFORMANCE COMPARISON

	P100	V100	Ratio
DL Training	10 TFLOPS	120 TFLOPS	12x
DL Inferencing	21 TFLOPS	120 TFLOPS	6x
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5x
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2x
STREAM Triad Perf	557 GB/s	855 GB/s	1.5x
NVLink Bandwidth	160 GB/s	300 GB/s	1.9x
L2 Cache	4 MB	6 MB	1.5x
L1 Caches	1.3 MB	10 MB	7.7x

(Image: NVIDIA)

Heads-up:

- Real performance highly dependent on layers/operations used
- Uses half/mixed precision parameters that might or might not deliver the same/comparable model accuracy
- Assumes that the entire model and data fits into GPU memory

(Image: NVIDIA)

NVIDIA DGX-1:

- 8x V100 GPUs (also exists with P100 GPUs)
- Deep Learning: 1,000 TFLOPS (peak)
- CUDA cores: 40,960
- Tensor cores: 5,120
- Host system: 2x 20 core Intel E5-2698v4, 256 GB memory
- Power consumption: 3,500 Watts

(Image: NVIDIA)

NVIDIA DGX-2:

- 16x V100 GPUs
- Deep Learning: 2,000 TFLOPS (peak)
- CUDA cores: 81,920
- Tensor cores: 10,240
- Host system: 2x 24 core Intel Xeon Platinum 8168 Processor, 512 GB memory
- 12 NVSwitches (for NVLinks)
- Power consumption: 10,000 Watts

(Image: NVIDIA)

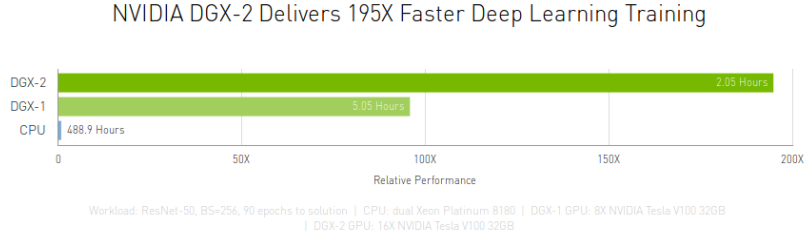
NVIDIA DGX A100:

- 8x A100 GPUs
- Deep Learning: 5,000 TFLOPS (peak)
- CUDA cores: 55,296
- Tensor cores: 3,456
- Host system: 2x 64 core AMD Rome 7742 Processor, 1 TB memory
- 8 NVSwitches (for NVLinks)
- Power consumption: 6,500 Watts

NVIDIA DGX H100:

- 8x H100 GPUs
- Deep Learning: 32,000 TFLOPS (peak, FP8)
- GPU memory: 640 GB
- Host system: 2x 56 core 4th Gen Intel Xeon scalable processors, 2 TB memory
- 4 NVSwitches (for NVLinks)
- Power consumption: 11,300 Watts

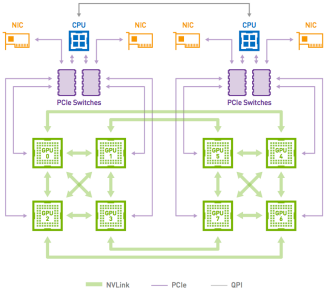
(Image: NVIDIA)



(Image: NVIDIA)

More benchmark results can be found at the NVIDIA Tesla Deep Learning Product Performance web page.

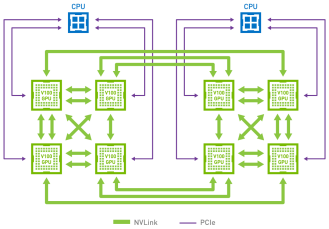
Also compare the official MLPerf results for training and inference here



(Image: NVIDIA)

NVLink (v1):

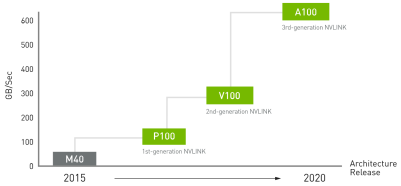
- Tesla P100
- Hybrid Cube Mesh
- 4 links per GPU, 20 GB/s per direction/per NVLink (160 GB/s aggregated)



(Image: NVIDIA)

NVLink 2.0:

- Tesla V100
- Hybrid Cube Mesh (also switch configuration possible)
- 6 links per GPU, 25 GB/s per direction/per NVLink2 (300 GB/s aggregated)



(Image: NVIDIA)

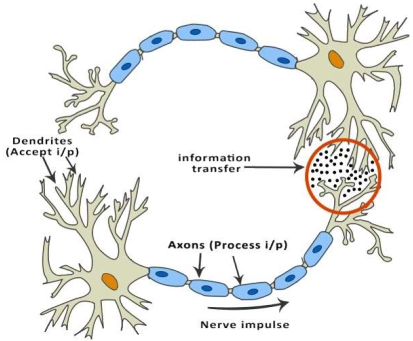
NVLink 3.0:

- ~~Tesla~~ A100
- Hybrid Cube Mesh (also switch configuration possible)
- 12 links per GPU, 25 GB/s per direction/per NVLink3 (600 GB/s aggregated)

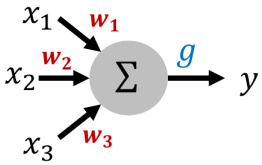
- Pro CPU:
  - More memory for larger models
  - Easier I/O and to set up
  - Some operations/layers can only be executed on the CPU (types or complexity)
- Pro GPU:
  - Very efficient if operations/layers are supported
  - Divide I/O and training between CPU and GPU
  - Best for deep networks (operations $\gg$ data ingestion)

Neuronal Networks

Inspired by biology:



(Image: Intel)



$$y = g\left(\sum_j W_j x_j + b\right)$$

Input from unit j

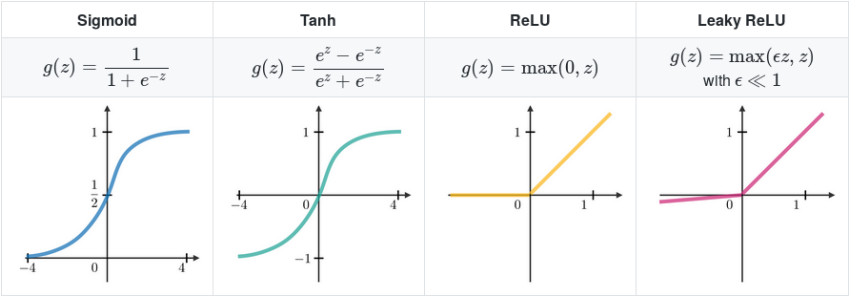
Output of unit

Activation Function

Linear weights

Bias unit

(Image: Intel)

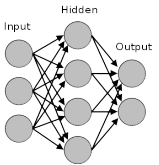


(Image: Afshine Amidi[2])

- Adds non-linearity
- ReLU is currently the most popular (est. 2010)

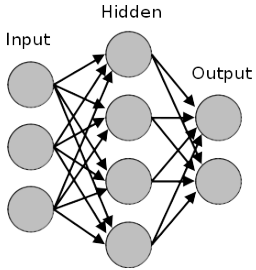
[2] https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning

Example of a "Deep" Neural Network:



(Image: Intel)

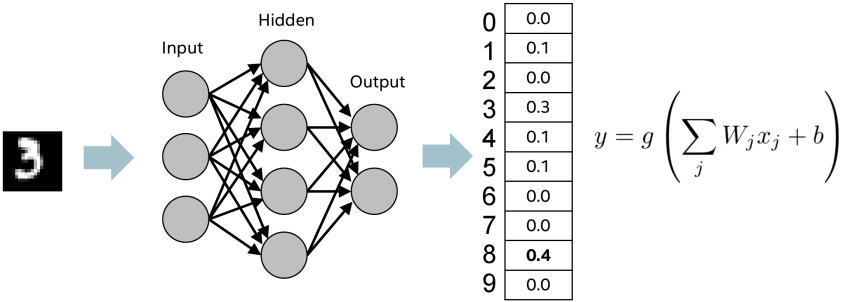
- Layers can have different number of neurons
- Input and output formats can be arbitrary
- There can be multiple (hundreds) of hidden layers
- Typically output is combined with *softmax* function (probabilistic output)
- Example shows fully connected network, which is a special case



(Image: Intel)

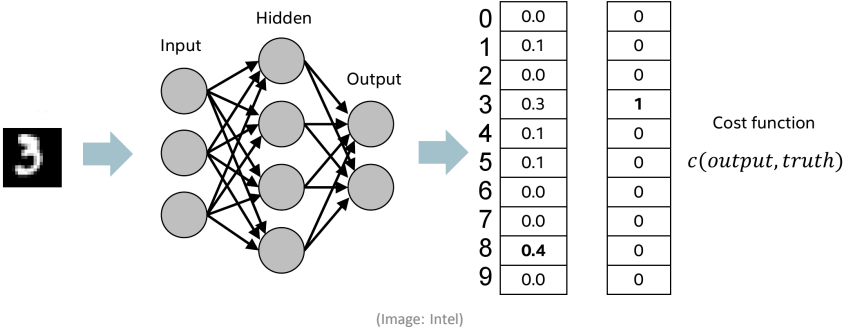
1. Random weights
2. Get a random batch of training data
3. Forward propagation
4. Calculate cost (loss)
5. Backward propagation
6. Update weights and bias
7. Goto step 2.

Example for one digit (image):



(Image: Intel)

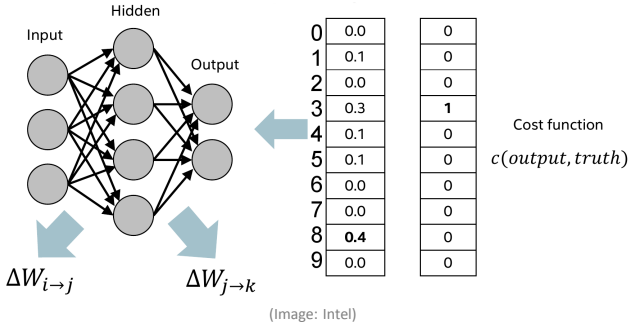
Example of a cost (or loss) function:



(Image: Intel)

- How far off are we from the ground truth?
- Example has labeled data (different if non-labeled data)

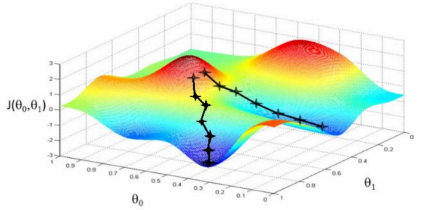
How weights are updated:



(Image: Intel)

- From back to front (problem: vanishing gradient for deep networks)
- Changes of the weights are usually dampened/controlled by a changing learning rate

How to find the best weight updates:

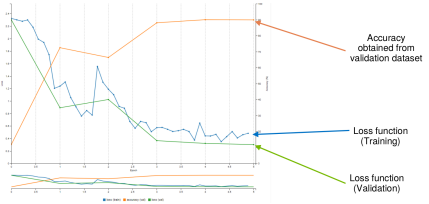


(Image: Intel)

- Gradient descent methods, e.g.:
  - Stochastic Gradient Descent (SGD)
  - Adaptive Moment Estimation (ADAM)
- Example: only two weights (θ_1, θ_2) with cost in 3rd dimension
- Multiple (local) minima are possible

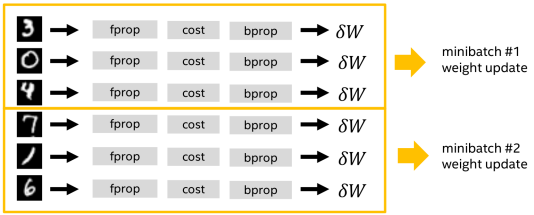
Training - Find the Best Weights

Georg Zitzlsberger, IT4Innovations



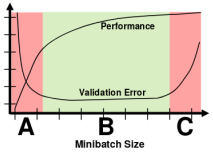
(Image: NVIDIA)

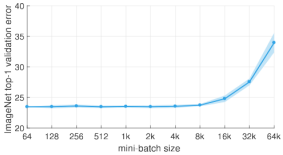
- Data sets separated into training, validation, and testing sets
- Training data set is repeatedly used for training (over epochs)
- Validation data: Track the performance of the network during training
- Testing data set: Final independent performance validation



(Image: Intel)

- Back-propagation is very expensive compared to forward-propagation
- Group training data in batches (so-called *minibatch*) of size *N*
- $N = \frac{training_{size}}{\#batches}$
- A *minibatch* allows parallel forward-propagation



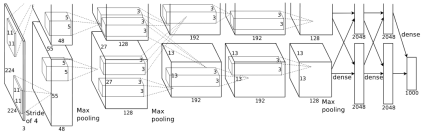


(Image: Ben-Nun, et al.)

- A higher mini-batch size increases performance
- **However:**
  - The larger the batch, the worse the training performance
  - The more memory is needed to store the parameters (problem for GPUs)
- Sweet spot needs to be found empirically

AlexNet:

- Won ImageNet Challenge 2012
- 5 conv. + 3 fully connected layers
- 60 million parameters



(Image: Krizhevsky, et al.)

An overview of more can be found here

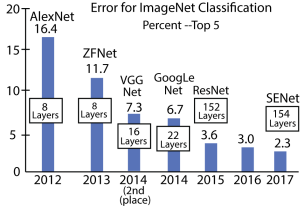
ResNet:

- Won ImageNet Challenge 2015
- Mitigates *vanishing gradient* problem
- 25 million parameters

(Image: He, et al.)

Image Classification Errors

Georg Zitzlsberger, IT4Innovations

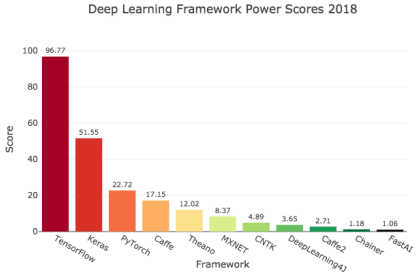


(Image: principlesofdeeplearning.com)

- Trend: More layers
- Error (performance) converges
- Ensemble networks were used last

- In two phases:
  - Training (time consuming)
  - Inference (usage)
- High quality and quantity training (and validation/testing) data is needed
- Output is probabilistic
- Programming with frameworks:
  - TensorFlow
  - CNTK
  - Theano

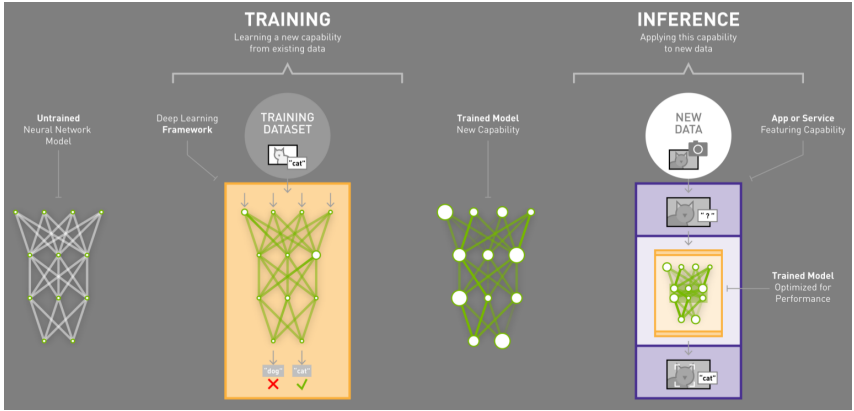
  } Keras
  - PyTorch
  - Caffe{2}
  - …



(Image: keras.io)

Training vs. Inference

Georg Zitzlsberger, IT4Innovations



(Image: NVIDIA)

Model Zoos make it easy to start:

- Use existing models
- Use pre-trained models for transfer learning

Model Zoo examples:

- Tensorflow: here
- PyTorch: here
- Caffe (BVLC): here
- …

Pretrained models are also available (e.g. for object detection with Tensorflow)

Optimizations for Inference

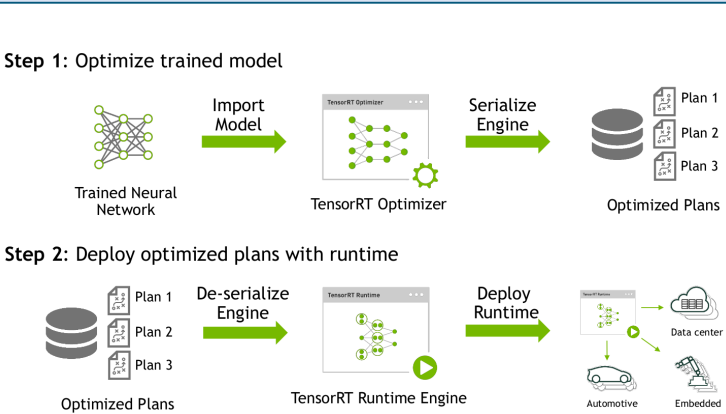
Training needs parallelism, but what about inference?

- Inference is the actual use of the network
- Inference only does forward propagation (weights are fixed)
- Trained networks can be optimized for inference:
  - Optimize away inactive neurons (due to drop-out)
  - Fuse layers and operations and remove redundancies
  - Optimize data structures
  - Optimize for different target architectures
  - Quantize (reduce precision of data types)
  - …

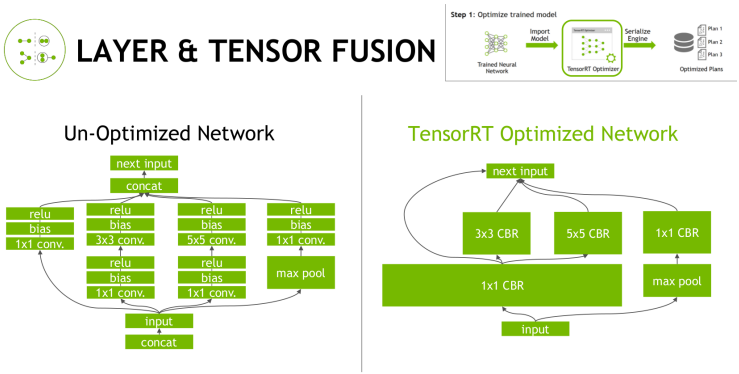
Optimizers exist:

- TensorRT from NVIDIA ▸ TensorRT Documentation
- Intel Deep Learning Deployment Toolkit from ▸ Intel OpenVino
- Both support Open Neural Network Exchange (ONNX) format

ONNX



(Image: NVIDIA)



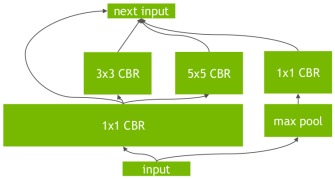
(Image: NVIDIA)

LAYER & TENSOR FUSION

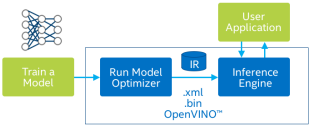
- Vertical Fusion
- Horizonal Fusion
- Layer Elimination

Network	Layers before	Layers after
VGG19	43	27
Inception V3	309	113
ResNet-152	670	159

TensorRT Optimized Network



(Image: NVIDIA)



(Image: Intel)

- Model Optimizer:
  - Optimize for endpoint target device
  - Transform to intermediate representation (IR)
  - Validated with over 100 public models for Caffe, Tensorflow, MXNet and ONNX
- Inference Engine:
  - Execute different layers on different targets (parallelism)
  - Implement custom layers on a CPU

Summary

Summary

Georg Zitzlsberger, IT4Innovations

- Deep Neural Networks have shown and still show remarkable results
- Latest Hardware architecture improvements fuel DL further
- There are many DL frameworks for training and inference
- There are vendor specific extensions (Intel or NVIDIA)

Great State of the Art (academic) overview:

▸ Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

(Tal Ben-Nun and Torsten Hoefler)