NVIDIA FOR DEEP LEARNING

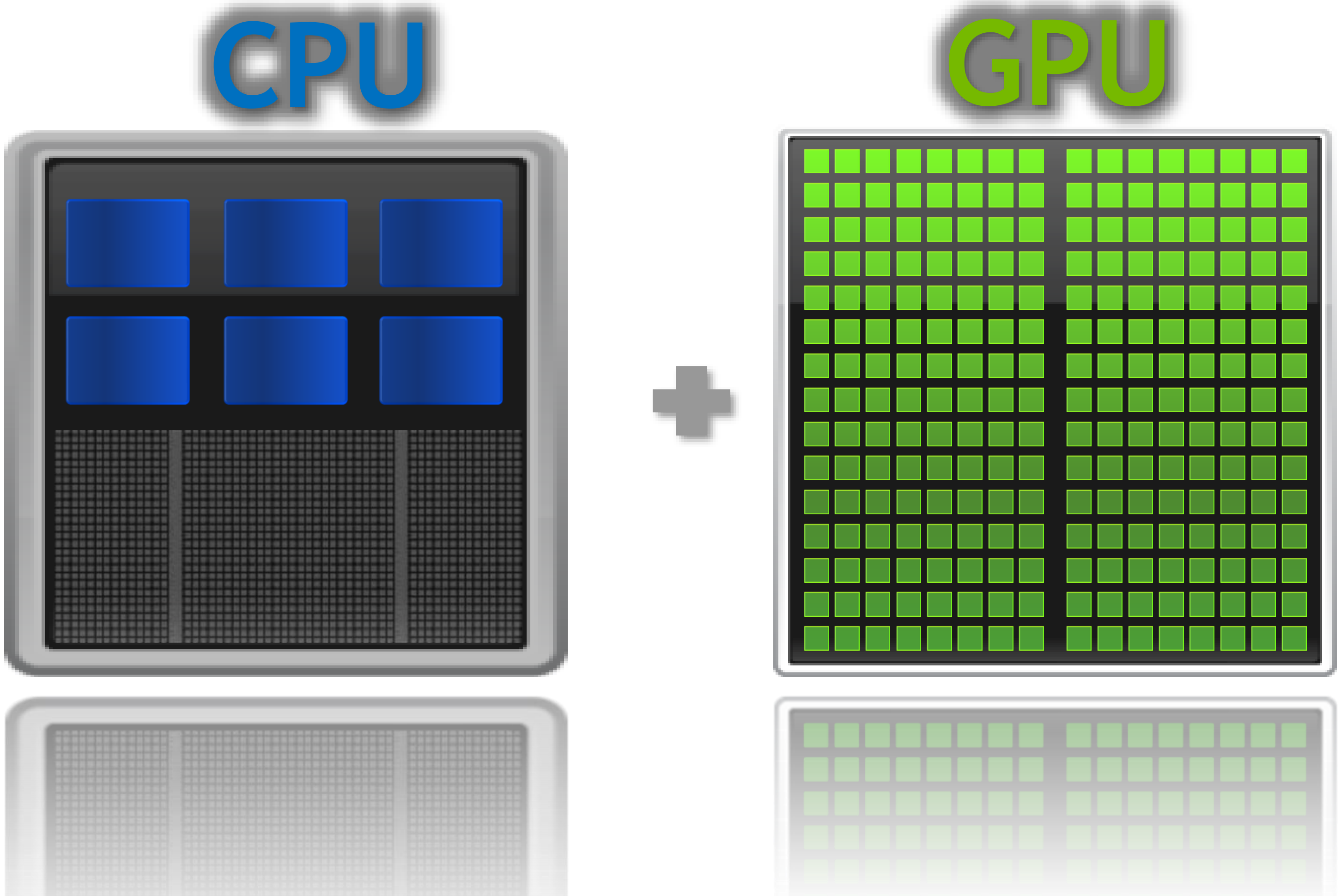
Bill Veenhuis | bveenhuis@nvidia.com

NVIDIA

Nvidia is the world’s leading ai platform

ONE ARCHITECTURE — CUDA

GPU: Perfect Companion for Accelerating Apps & A.I.



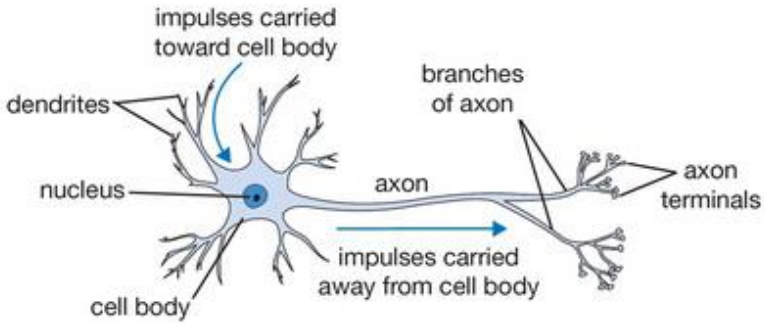
AGENDA & TOPICS

- Intro to AI
- Deep Learning Intro
- NVIDIA’s DIGITS
- Autoencoding Enhancement
- TensorRT

Intro to AI

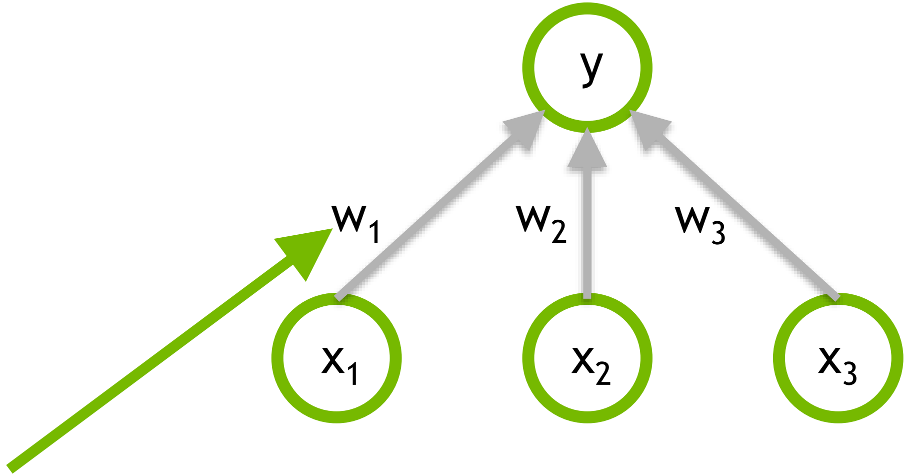
ARTIFICIAL NEURONS

Biological neuron


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

From Stanford cs231n lecture notes

Artificial neuron



Weights (W_n)
= parameters

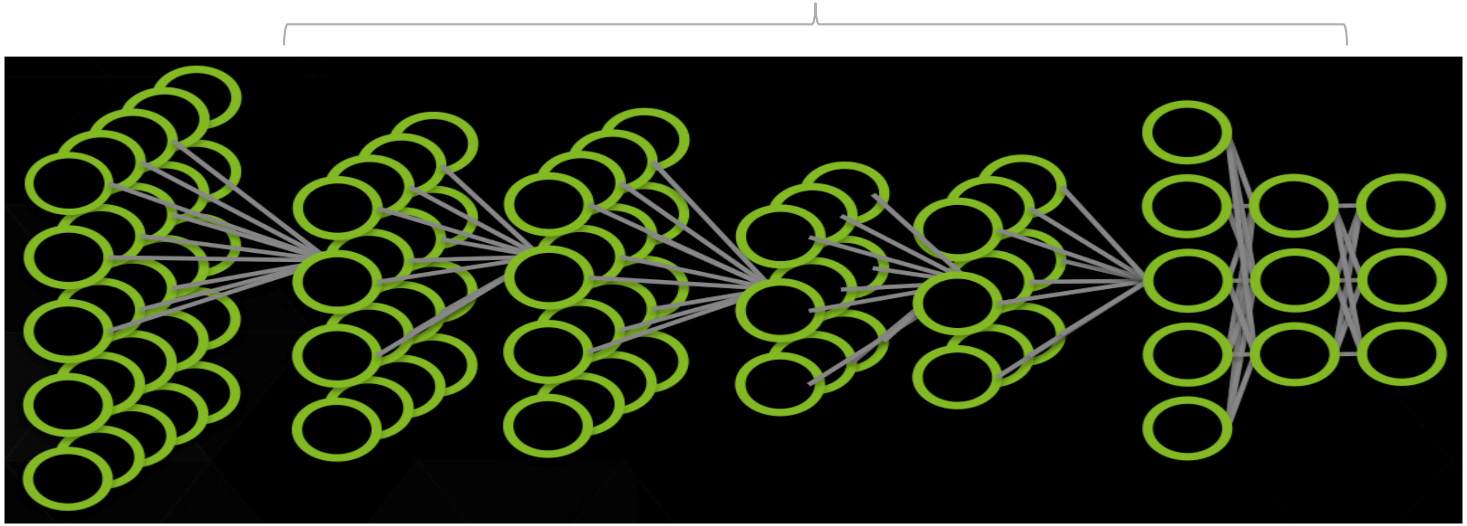
$y=F(w_1x_1+w_2x_2+w_3x_3)$

ARTIFICIAL NEURAL NETWORK

A collection of simple, trainable mathematical units that collectively learn complex functions

Hidden layers

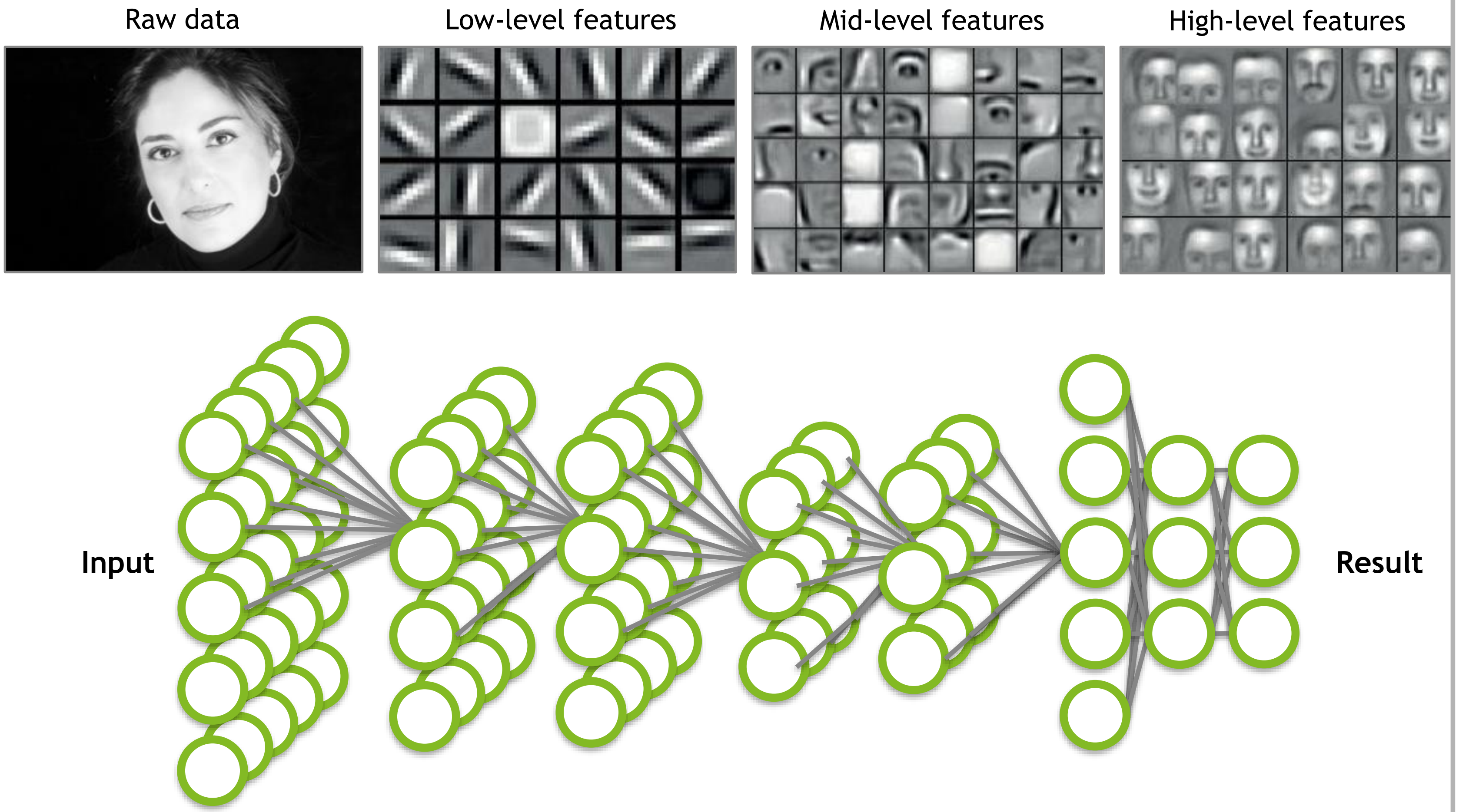
Input layer



Output layer

Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

DEEP NEURAL NETWORK (DNN)



Application components:

Task objective
e.g. Identify face

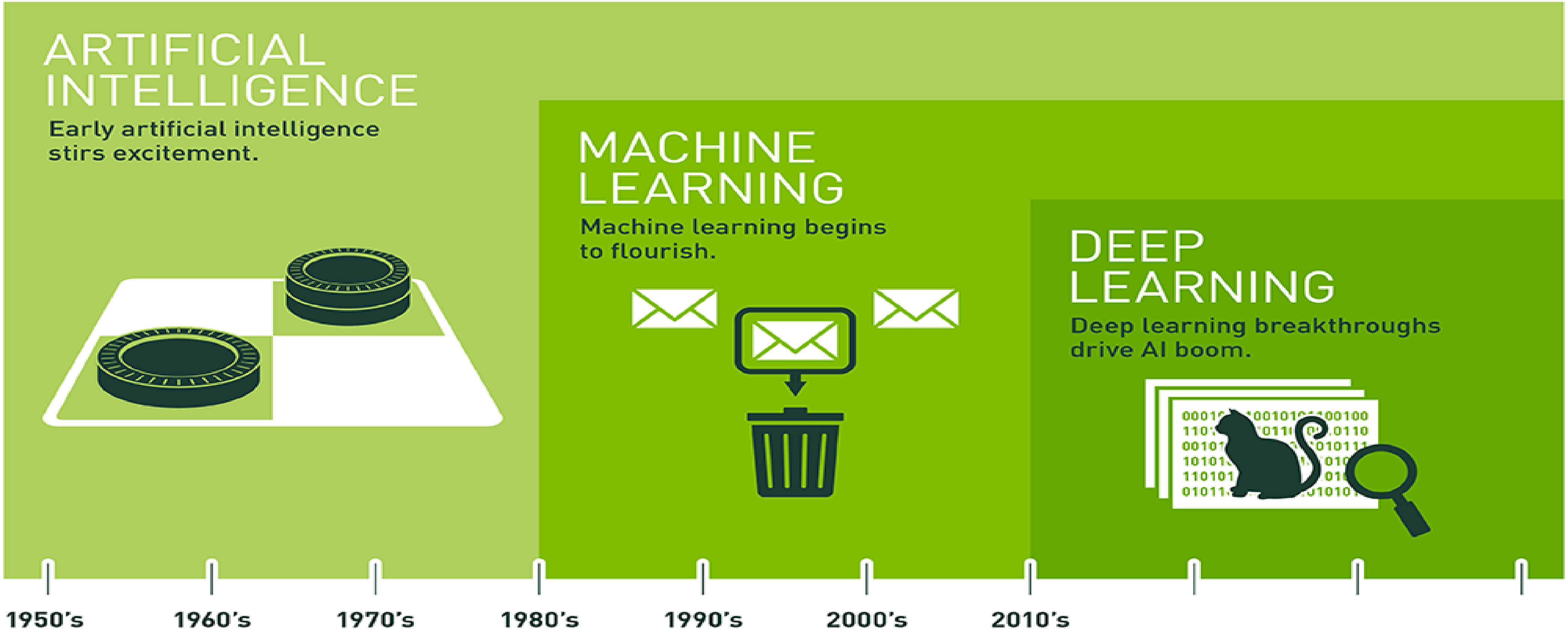
Training data
10-100M images

Network architecture
~10s-100s of layers
1B parameters

Learning algorithm
~30 Exaflops
1-30 GPU days

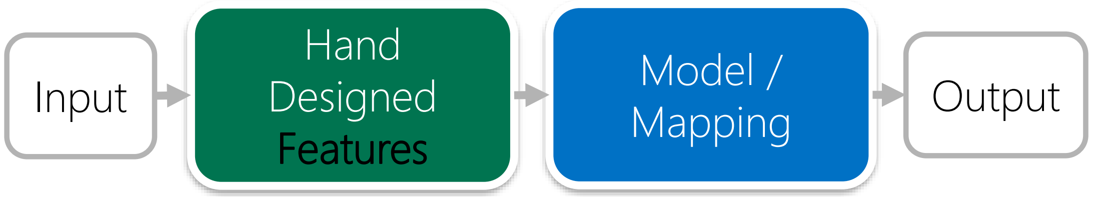
WHAT IS DEEP LEARNING?

Accomplishing complex goals


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

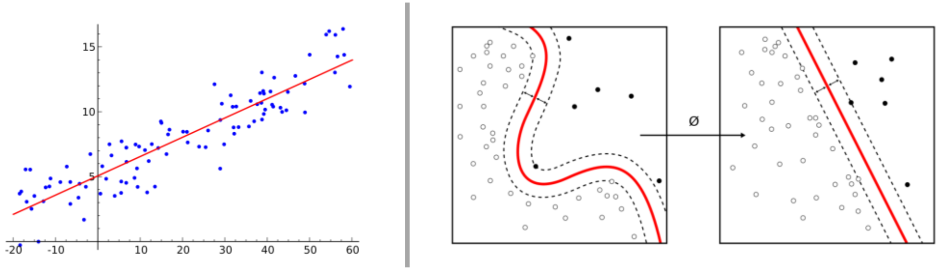
Difference in Workflow

Classic Machine Learning [1990 : now]

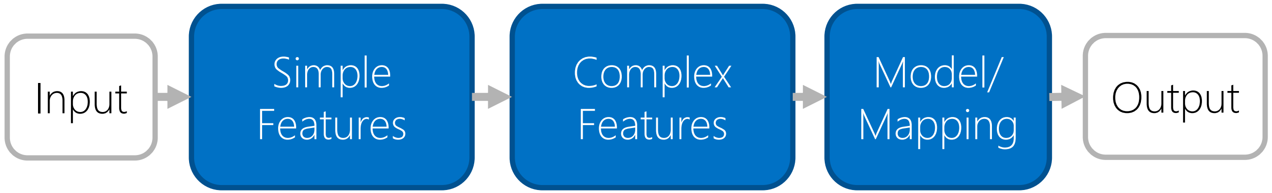


Input → Hand Designed Features → Model / Mapping → Output

Examples [Regression and SVMs]

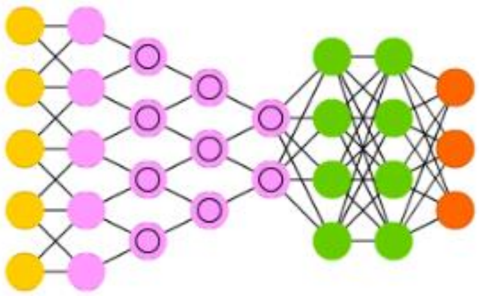


Deep/End-to-End Learning [2012 : now]



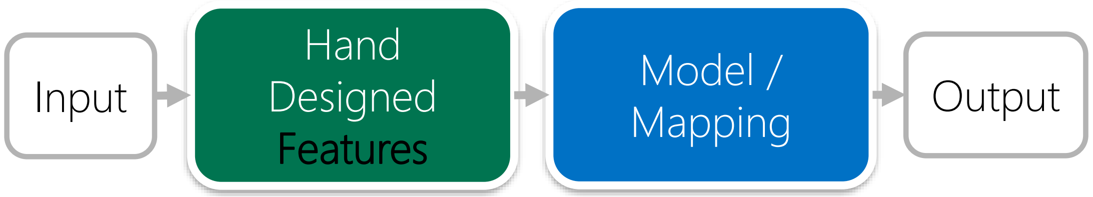
Input → Simple Features → Complex Features → Model/ Mapping → Output

Example [Conv Net]



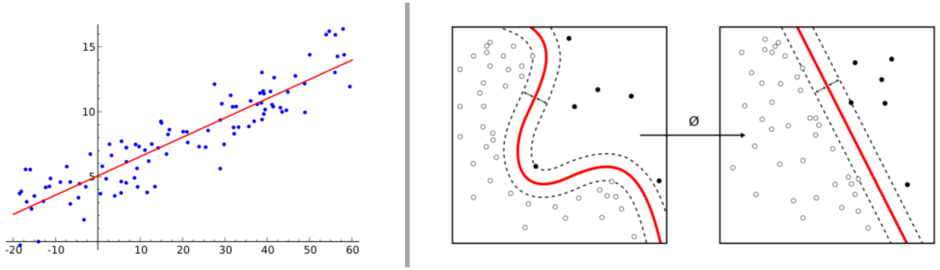
Traditional Workflow

Classic Machine Learning [1990 : now]



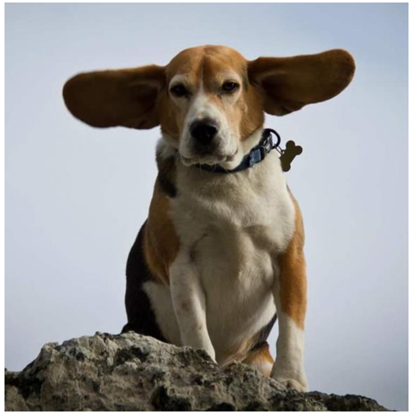
Input → Hand Designed Features → Model / Mapping → Output

Examples [Regression and SVMs]



Challenge in Slack channel: How would you describe this image to someone (or something) blind?

Difficult: From it’s raw pixels.
Medium: From geometric primitives (lines, curves, colors)
Easy: Using any words that you may know

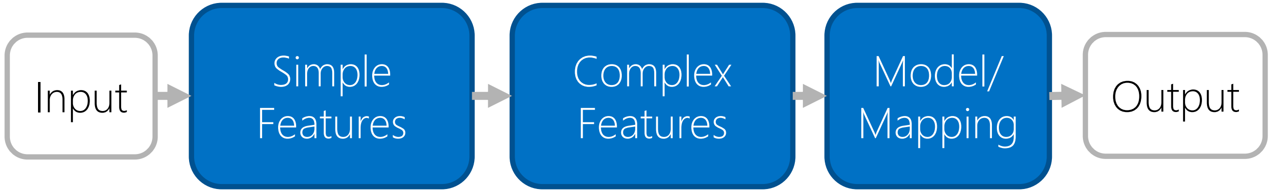


Deep Learning Workflow

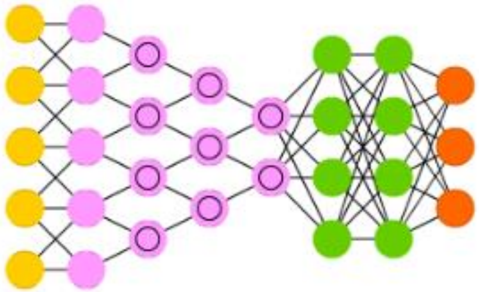
Experience: Trust Neural Network to learn features and model by providing inputs and outputs.

Key Skill: Experience (data) creation

Deep/End-to-End Learning [2012 : now]



Example [Conv Net]



NVIDIA’S DIGITS

NVIDIA’S DIGITS

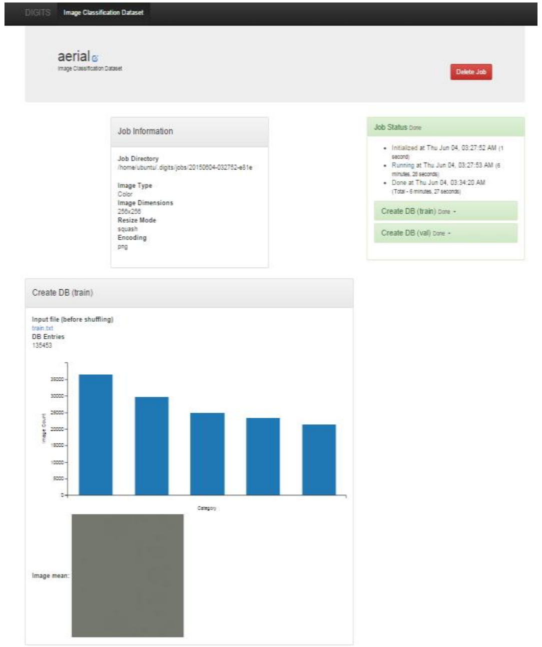
Interactive Deep Learning GPU Training System

- Simplifies common deep learning tasks such as:
    - Managing data
    - Designing and training neural networks on multi-GPU systems
    - Monitoring performance in real time with advanced visualizations
- Completely interactive so data scientists can focus on designing and training networks rather than programming and debugging
- Open source

NVIDIA’S DIGITS

Interactive Deep Learning GPU Training System

Process Data



aerial
Image Classification Dataset

Job Information

Job Directory
/home/ubuntu/.digits/jobs/20150604-032752-e81e

Image Type
Color
Image Dimensions
256x256
Resize Mode
squash
Encoding
png

Job Status Done

- Initialized at Thu Jun 04, 03:27:52 AM (1 second)
- Running at Thu Jun 04, 03:27:53 AM (6 minutes, 26 seconds)
- Done at Thu Jun 04, 03:34:20 AM (Total - 6 minutes, 27 seconds)

Create DB (train) Done

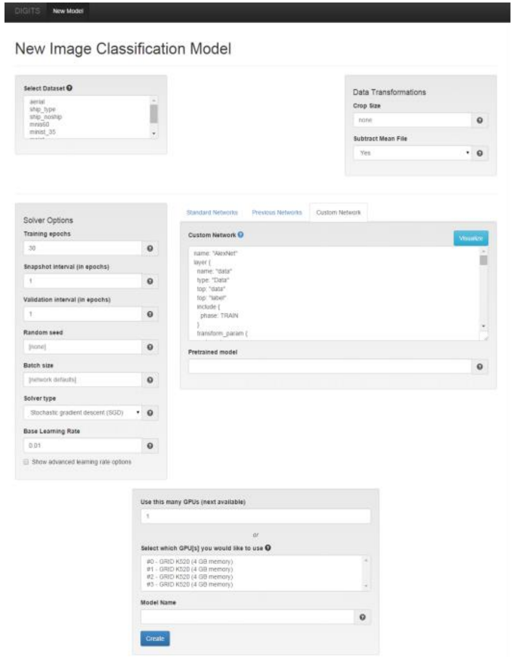
Create DB (val) Done

Create DB (train)

Input file (before shuffling)
train.txt
DB Entries
135453

Image mean:

Configure DNN



New Image Classification Model

Select Dataset
aerial
ship_type
ship_noship
mnist60
mnist_35

Data Transformations
Crop Size: none
Subtract Mean File: Yes

Solver Options
Training epochs: 30
Snapshot interval (in epochs): 1
Validation interval (in epochs): 1
Random seed: [none]
Batch size: [network defaults]
Solver type: Stochastic gradient descent (SGD)
Base Learning Rate: 0.01
□ Show advanced learning rate options

Custom Network

```
name: "AlexNet"
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
```

Pretrained model

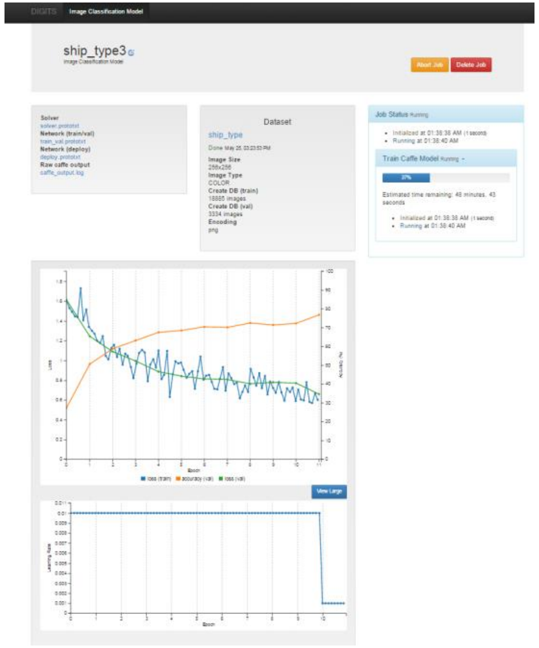
Use this many GPUs (next available): 1

or

Select which GPU[s] you would like to use
#0 - GRID K520 (4 GB memory)
#1 - GRID K520 (4 GB memory)
#2 - GRID K520 (4 GB memory)
#3 - GRID K520 (4 GB memory)

Model Name

Monitor Progress



ship_type3
Image Classification Model

Solver
solver.prototxt
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Raw caffe output
caffe_output.log

Dataset
ship_type
Done May 25, 03:23:53 PM
Image Size
256x256
Image Type
COLOR
Create DB (train)
18885 images
Create DB (val)
3334 images
Encoding
png

Job Status Running

- Initialized at 01:38:38 AM (1 second)
- Running at 01:38:40 AM

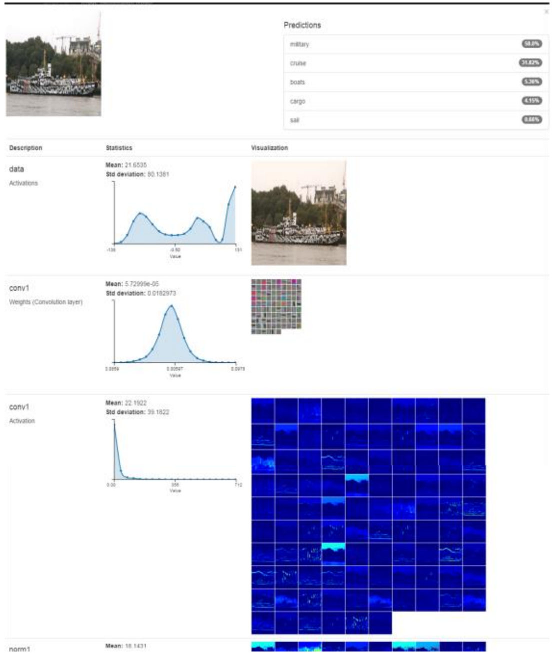
Train Caffe Model Running

37%

Estimated time remaining: 48 minutes, 43 seconds

- Initialized at 01:38:38 AM (1 second)
- Running at 01:38:40 AM

Visualization

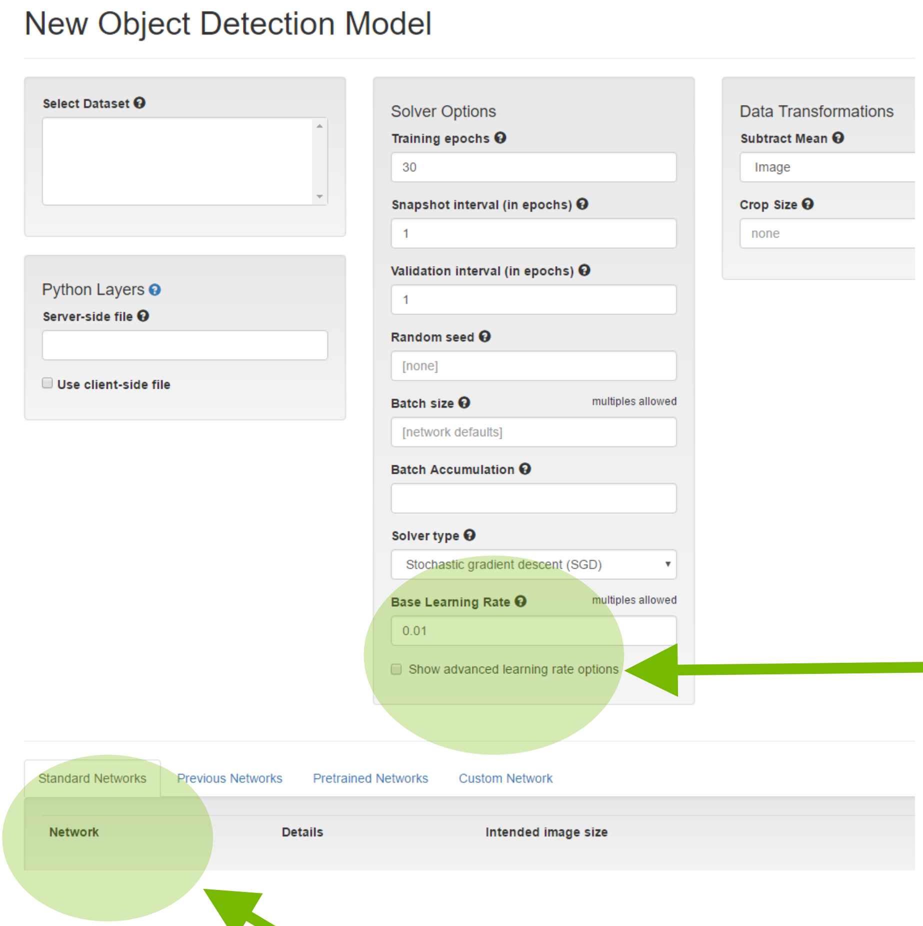


Predictions

military	58.0%
cruise	31.82%
boats	5.36%
cargo	4.15%
sail	0.66%

Description	Statistics	Visualization
data Activations	**Mean:** 21.6535 **Std deviation:** 80.1381	
conv1 Weights (Convolution layer)	**Mean:** 5.72999e-05 **Std deviation:** 0.0182973	
conv1 Activation	**Mean:** 22.1922 **Std deviation:** 39.1822	
norm1	**Mean:** 18.1431	

DIGITS - MODEL



New Object Detection Model

Select Dataset

Python Layers

Server-side file

□ Use client-side file

Solver Options

Training epochs: 30

Snapshot interval (in epochs): 1

Validation interval (in epochs): 1

Random seed: [none]

Batch size (multiples allowed): [network defaults]

Batch Accumulation:

Solver type: Stochastic gradient descent (SGD)

Base Learning Rate (multiples allowed): 0.01

□ Show advanced learning rate options

Data Transformations

Subtract Mean: Image

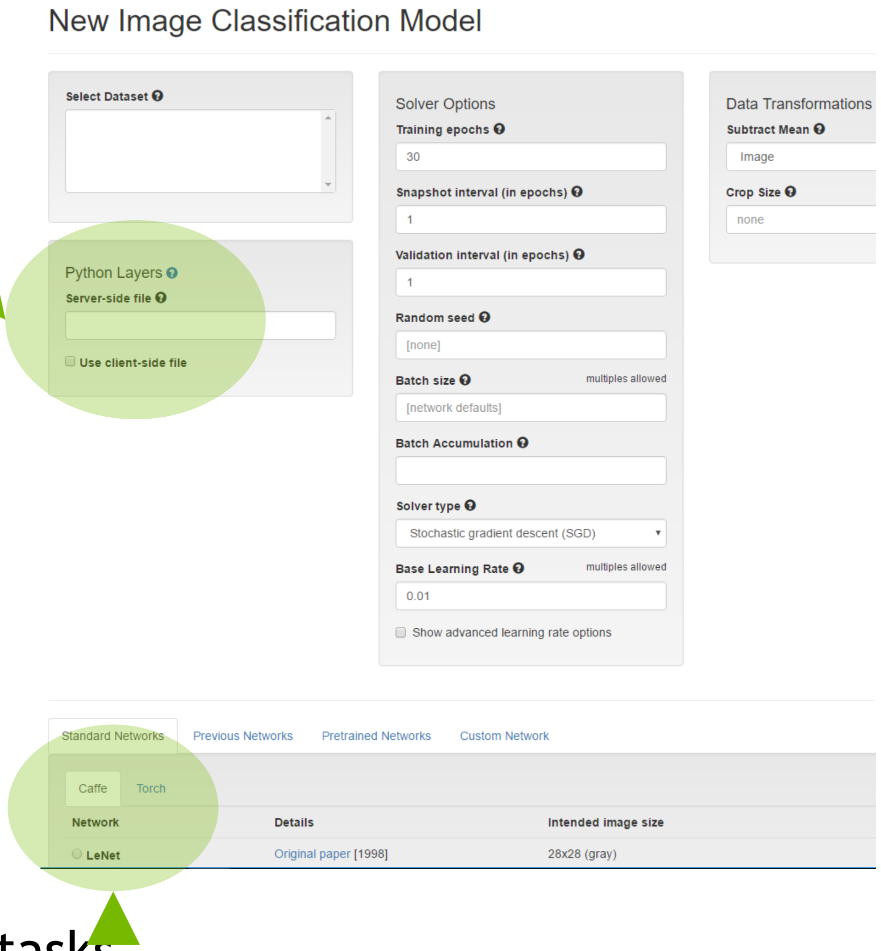
Crop Size: none

Standard Networks | Previous Networks | Pretrained Networks | Custom Network

Network	Details	Intended image size

Define custom layers with Python

Can anneal the learning rate



New Image Classification Model

Select Dataset

Python Layers

Server-side file

□ Use client-side file

Solver Options

Training epochs: 30

Snapshot interval (in epochs): 1

Validation interval (in epochs): 1

Random seed: [none]

Batch size (multiples allowed): [network defaults]

Batch Accumulation:

Solver type: Stochastic gradient descent (SGD)

Base Learning Rate (multiples allowed): 0.01

□ Show advanced learning rate options

Data Transformations

Subtract Mean: Image

Crop Size: none

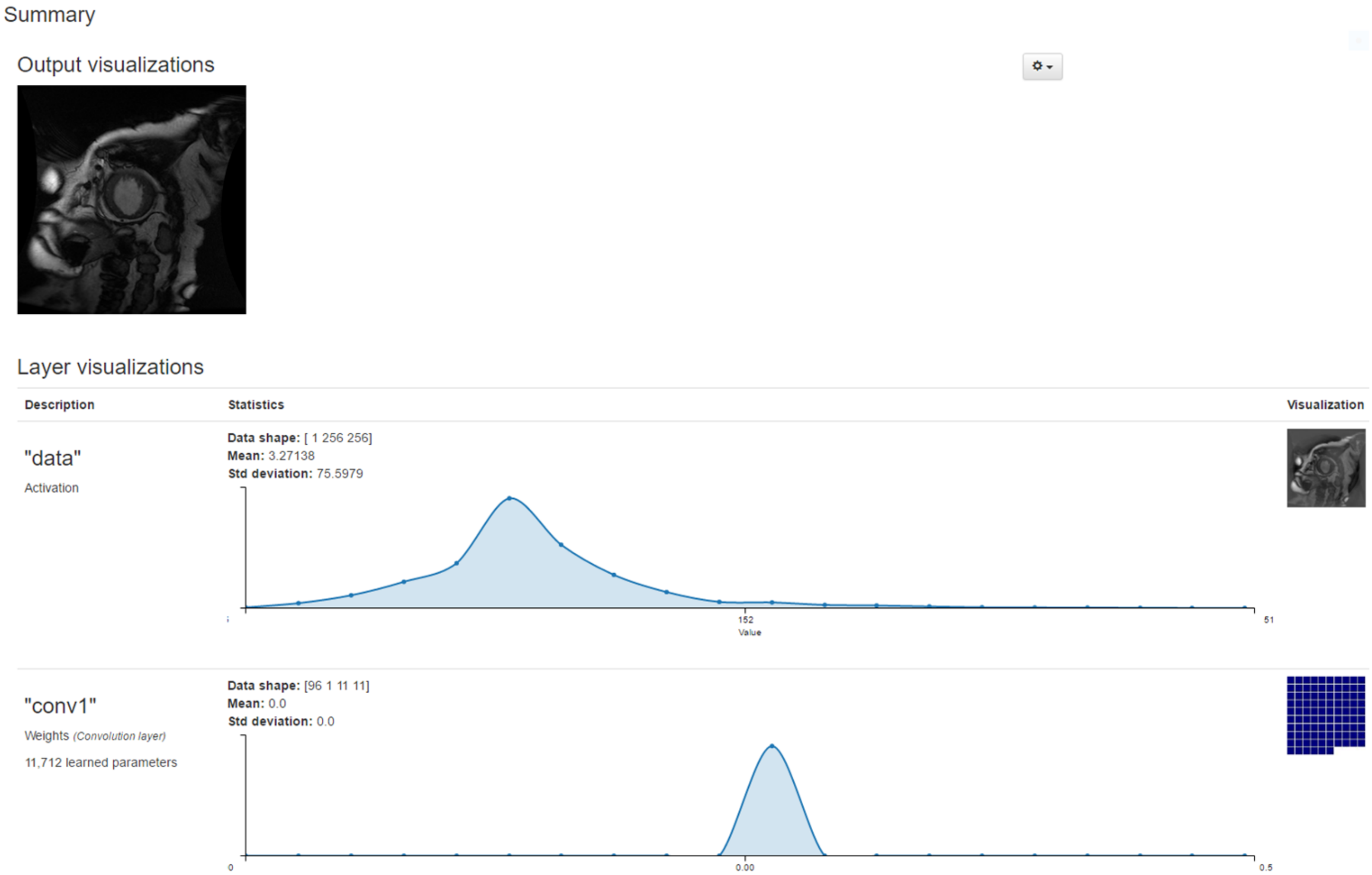
Standard Networks | Previous Networks | Pretrained Networks | Custom Network

Caffe | Torch

Network	Details	Intended image size
○ LeNet	Original paper [1998]	28x28 (gray)

Differences may exist between model tasks

DIGITS – VISUALIZATION RESULTS



Summary

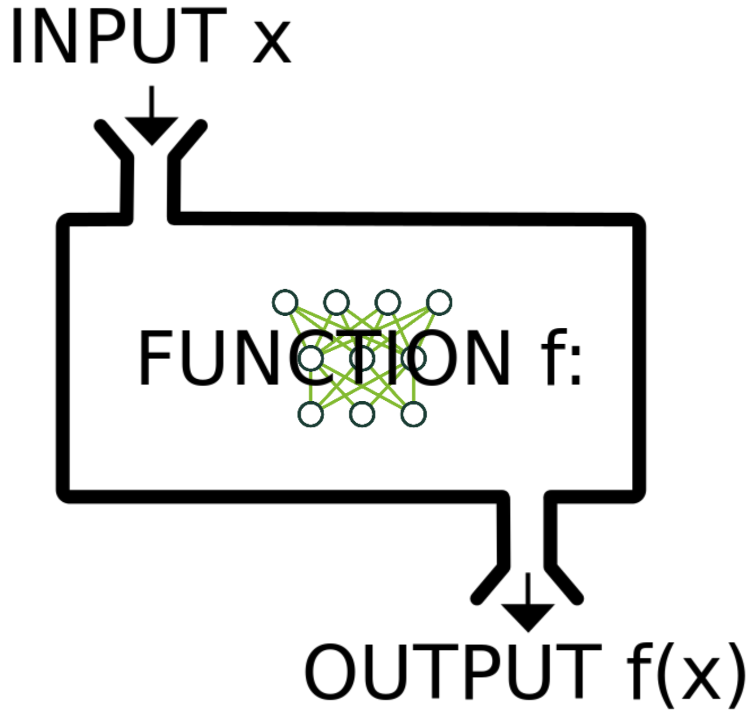
Output visualizations

Layer visualizations

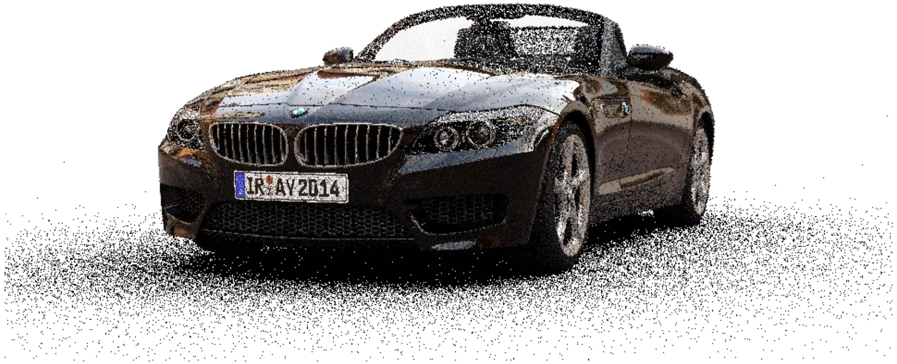
Description	Statistics	Visualization
"data" Activation	**Data shape:** [1 256 256] **Mean:** 3.27138 **Std deviation:** 75.5979	
"conv1" Weights *(Convolution layer)* 11,712 learned parameters	**Data shape:** [96 1 11 11] **Mean:** 0.0 **Std deviation:** 0.0	

ENHANCING IMAGES WITH AN AI AUTOENCODER

A great candidate for Deep Learning!



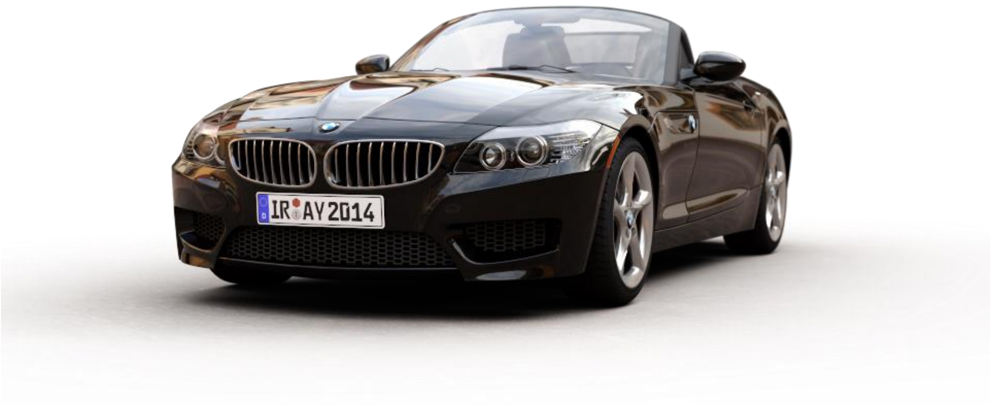
Training Set of images.



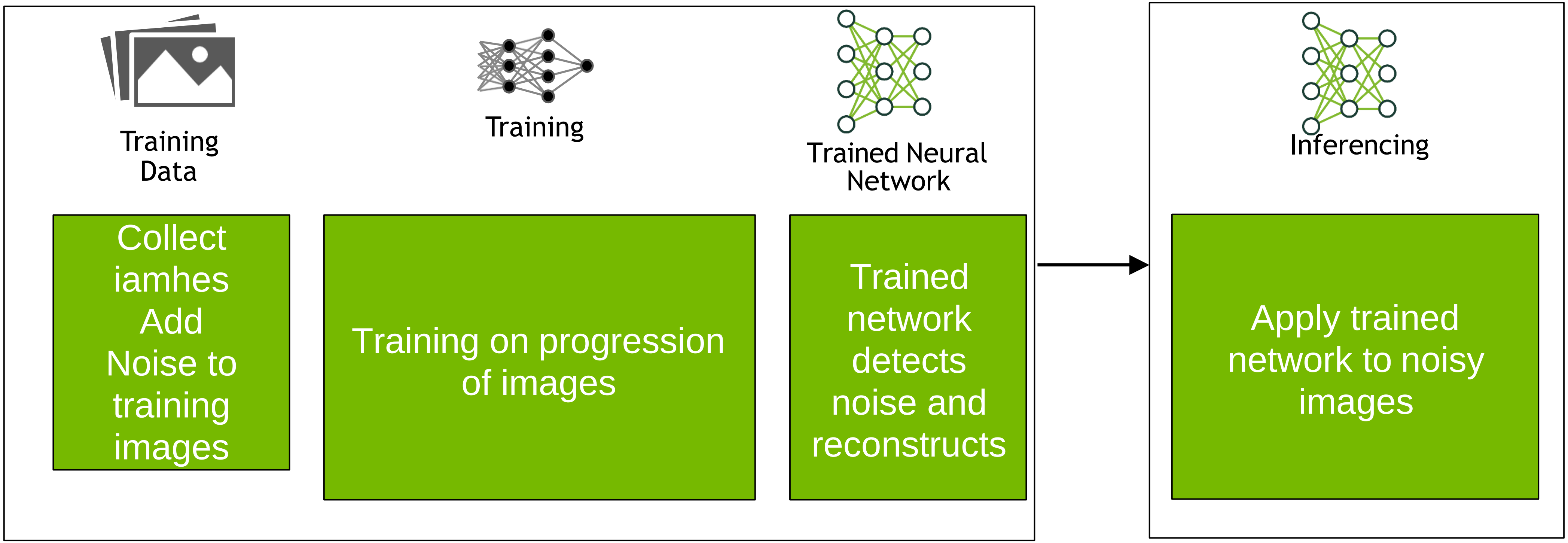
1 sample per pixel

- It requires pairs of noisy and noise-free images. The network will learn to remove the noise from the images.

- We can then deploy this trained model to any image we want to denoise. (inference)



Deep Learning for Image Denoising



Training Data

Training

Trained Neural Network

Inferencing

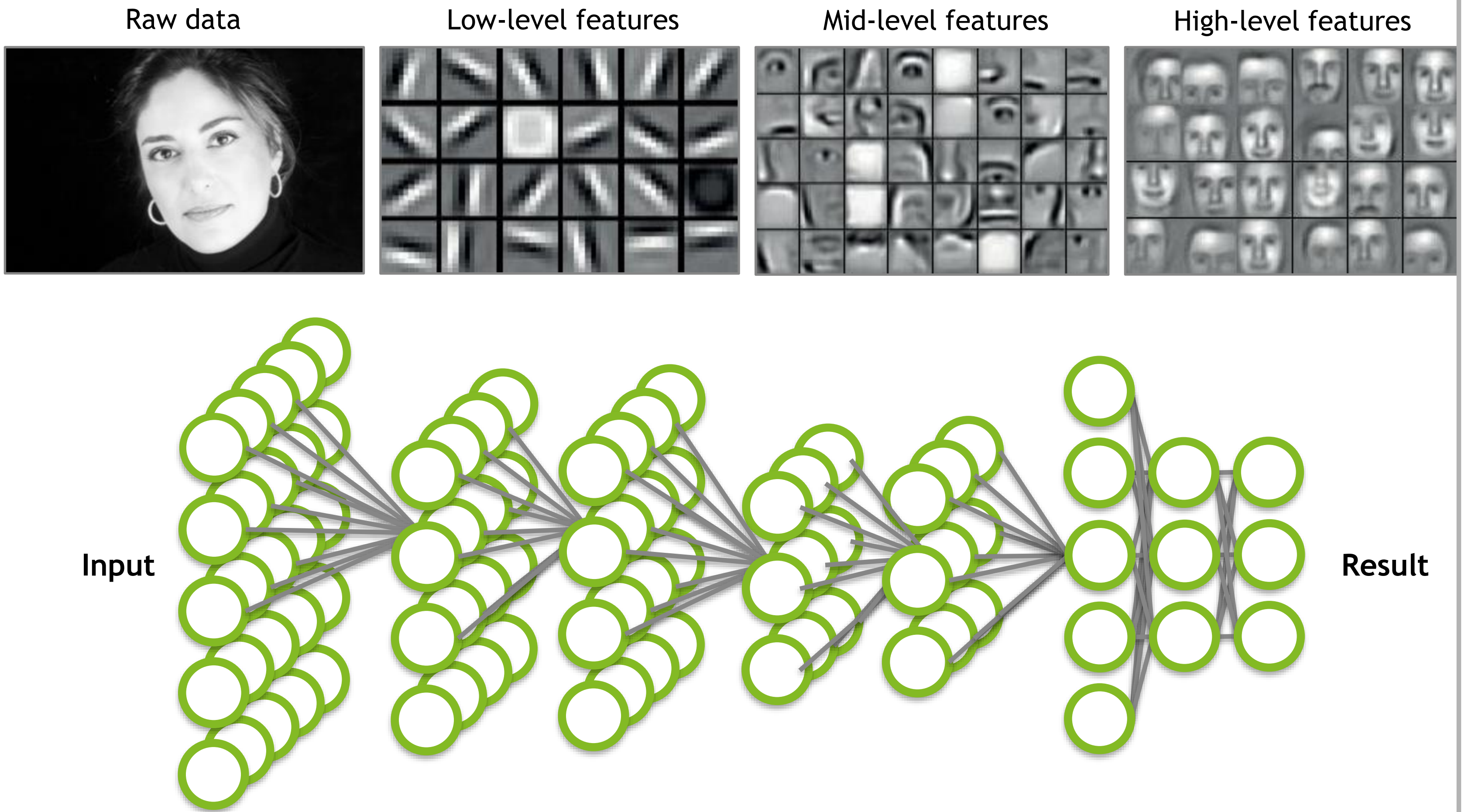
Collect iamhes Add Noise to training images

Training on progression of images

Trained network detects noise and reconstructs

Apply trained network to noisy images

Learning about images (CNN)



Application components:

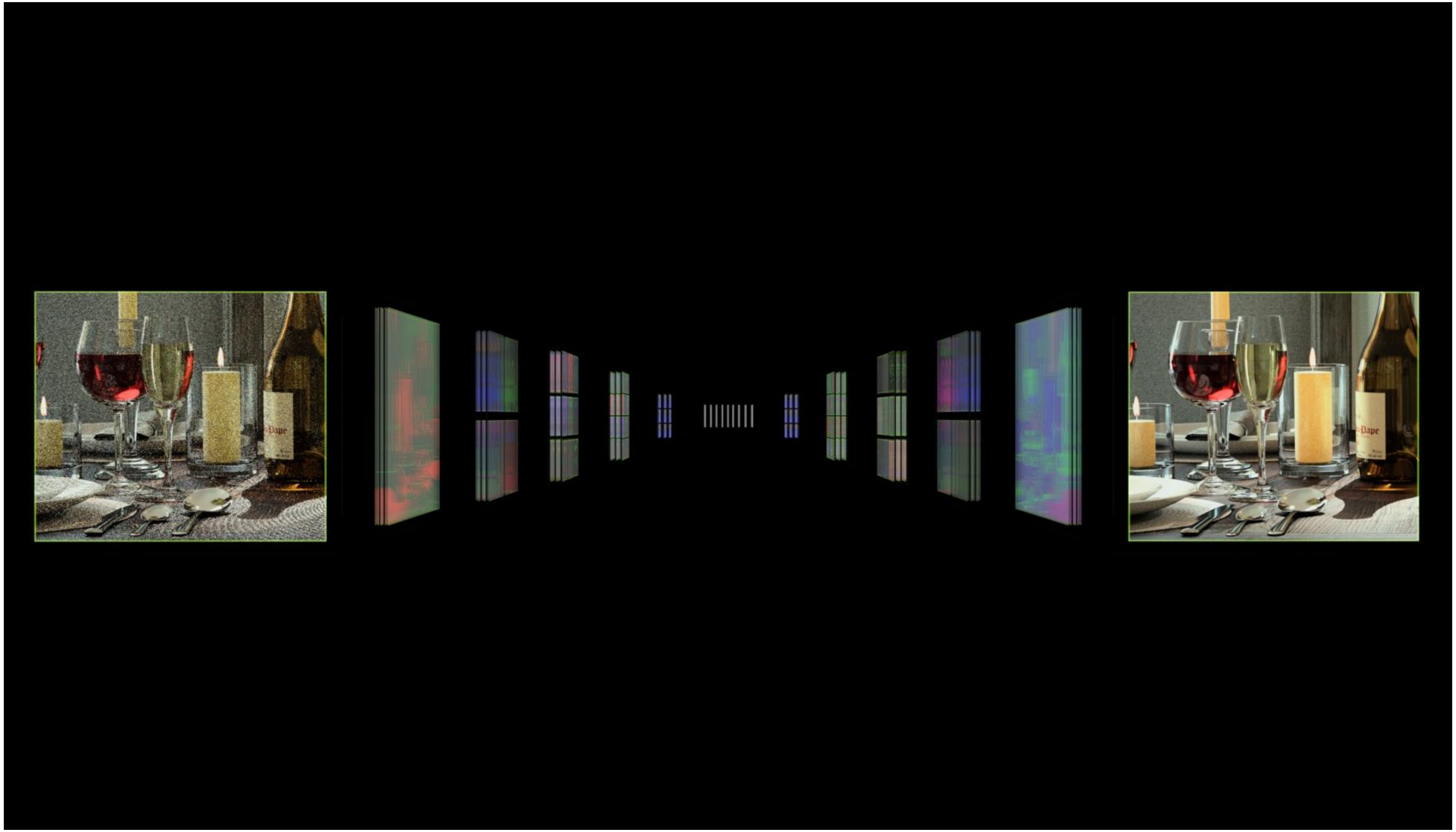
Task objective
e.g. Identify face

Training data
10-100M images

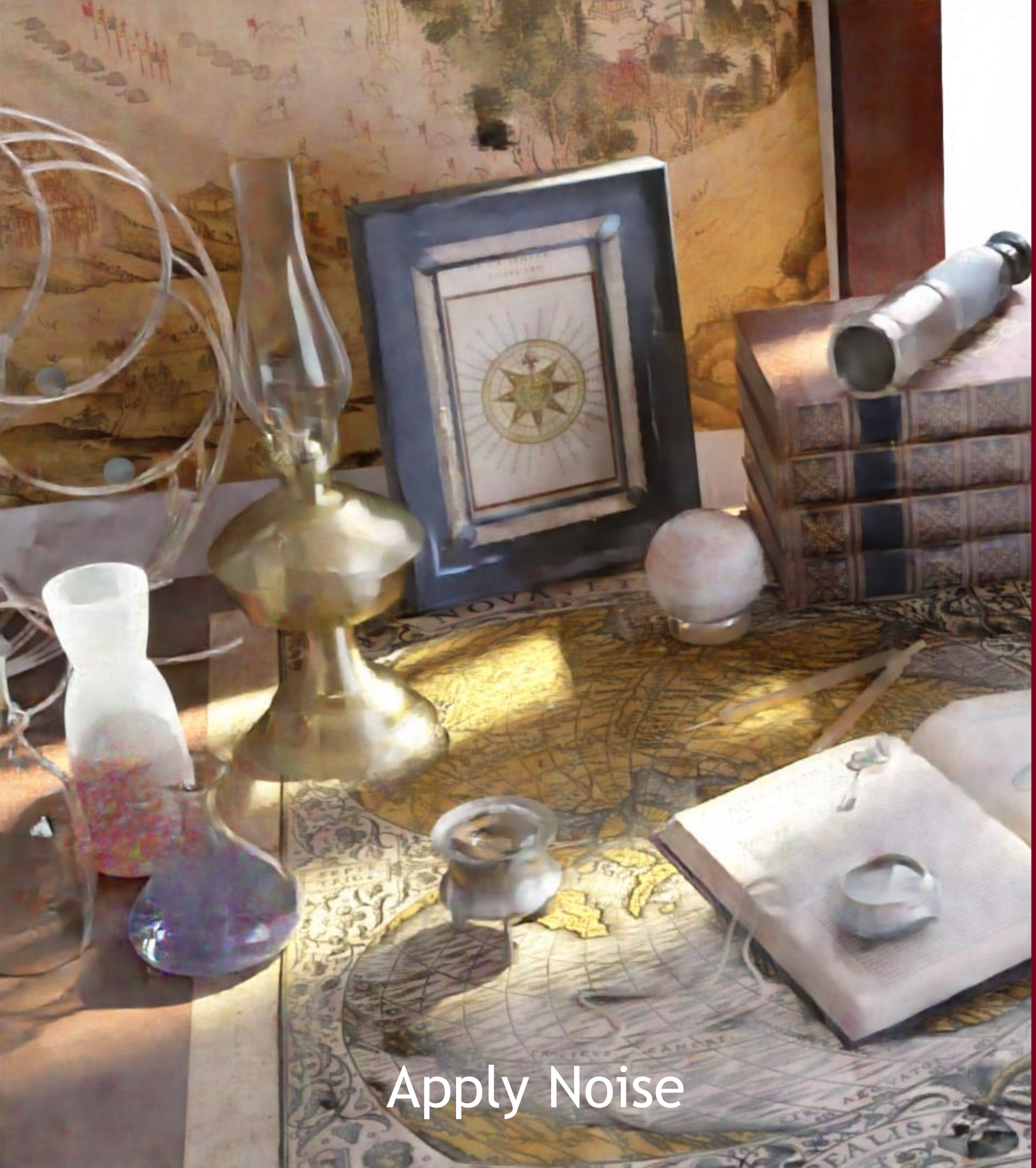
Network architecture
~10s-100s of layers
1B parameters

Learning algorithm
~30 Exaflops
1-30 GPU days

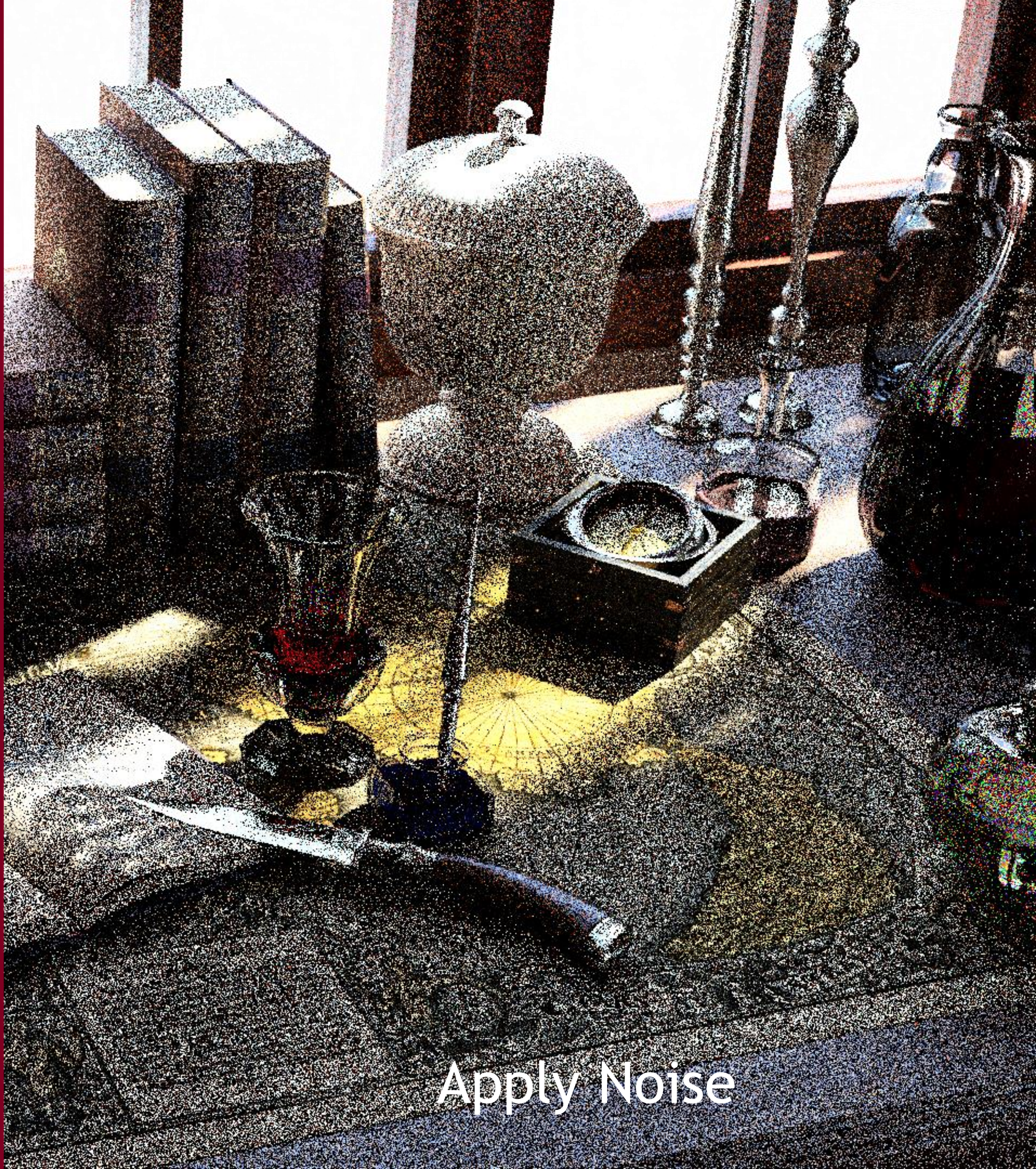
Autoencoder – in Action



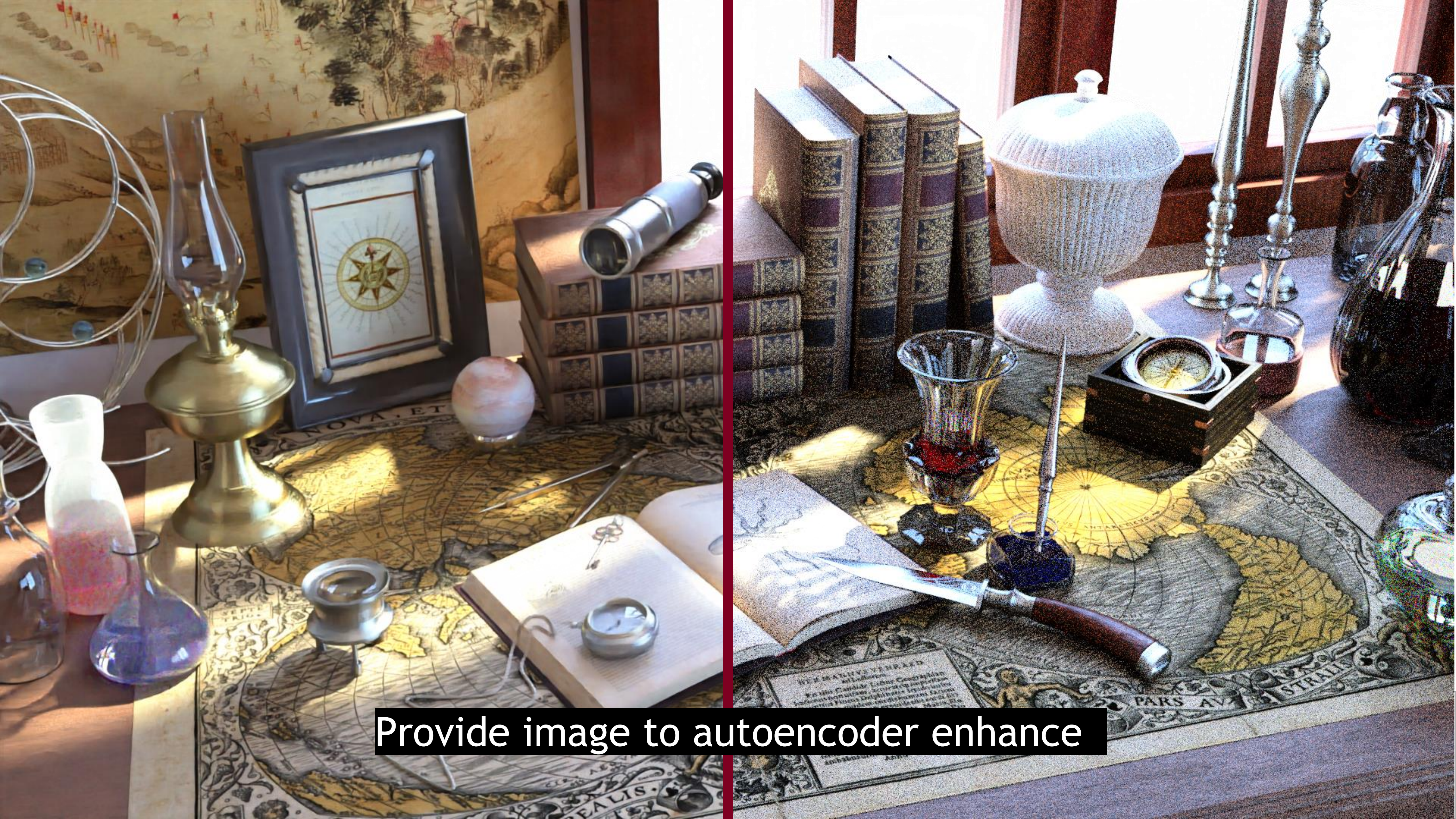
NOVA ET
HERMANNVS VENRAED
ad Lectorem.
En tibi Candide Lector Geographiam
hactenus non uiſam, accuratéq; impreſſam
Orontius Fineus Delphinates lepido uultu
offert, quæ quidem cordis humani faciem
formamq; obtinet (& prouide tibi cordi ſit)
atq; etiam Prouintias, Inſulas, Maria, Flu
mina, Montes hactenus non uiſa, neq; Pto
lomeo, neq; Eudoxo, neq; Eratoſteni, aut
Macrobio cognita, ſed quę in tenebris in
hunc uſq; diem iacuerunt, tuo obtutui præ
ſentat. Tu igitur hoc munuſculum (ſi ſapis)
ambabus ulnis ſuſcipito boniq; conſulito.
Anno 1531.
PARS AV STRALIS
REALIS



Apply Noise



Apply Noise



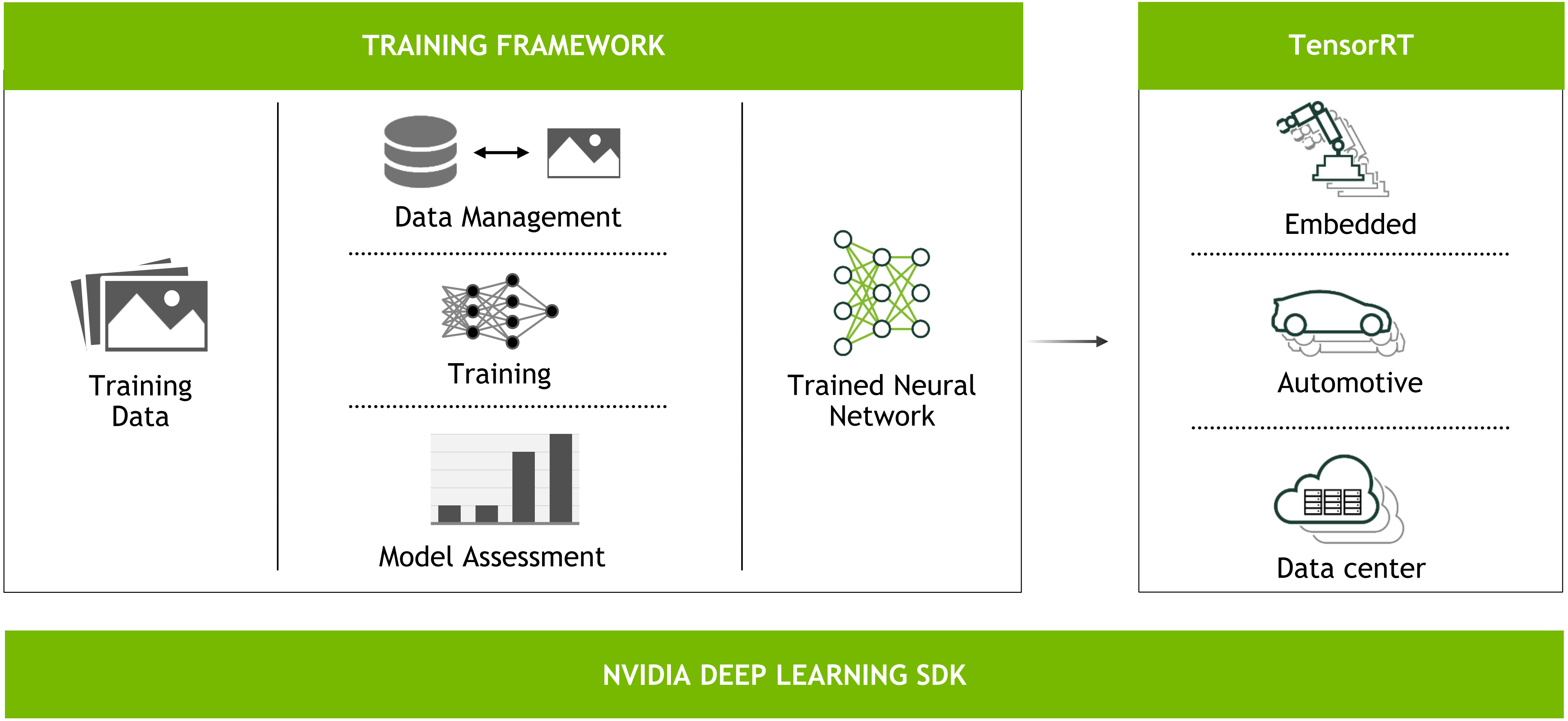
Provide image to autoencoder enhance

NOVA ET
HERMANNVS VENRAED ad Lectorem.
PARS AV STRALIS
REALIS

TensorRT

SOFTWARE INFERENCING PERFROMANCE EHANCEMENT

NVIDIA DEEP LEARNING SOFTWARE PLATFORM



TRAINING FRAMEWORK

Training Data

Data Management

Training

Model Assessment

Trained Neural Network

TensorRT

Embedded

Automotive

Data center

NVIDIA DEEP LEARNING SDK

developer.nvidia.com/deep-learning-software

NVIDIA TensorRT

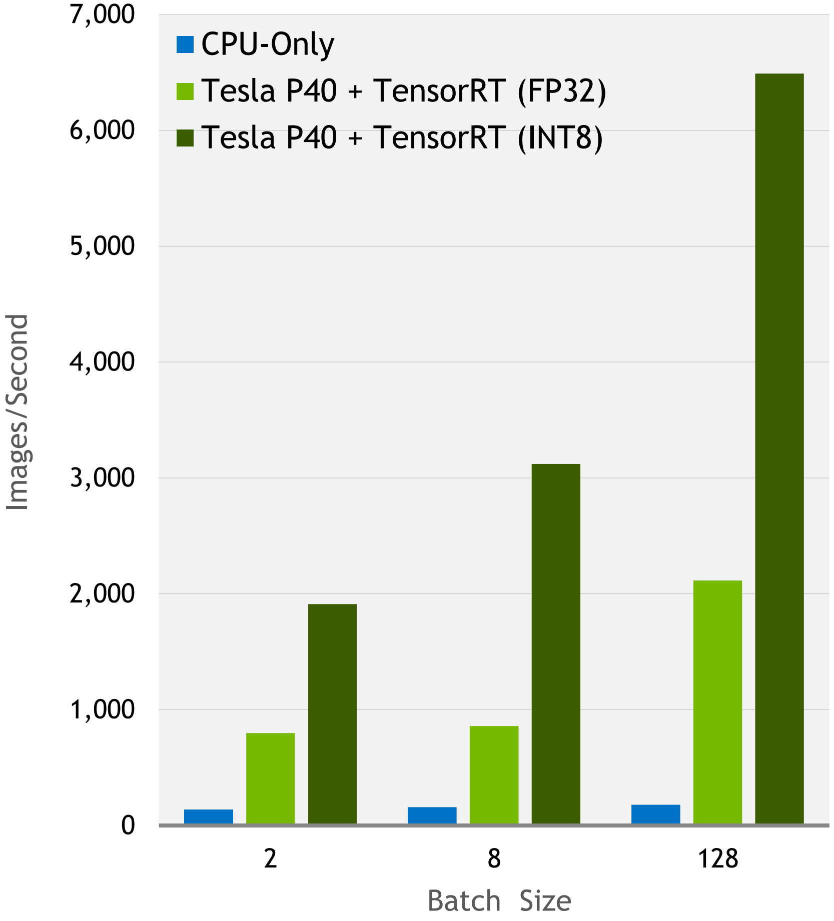
High-performance deep learning inference for production deployment

High performance neural network inference engine for production deployment

- Generate optimized and deployment-ready models for datacenter, embedded and automotive platforms
- Deliver high-performance, low-latency inference demanded by real-time services
- Deploy faster, more responsive and memory efficient deep learning applications with INT8 and FP16 optimized precision support

developer.nvidia.com/tensorrt

Up to 36x More Image/sec



GoogLenet, CPU-only vs Tesla P40 + TensorRT
CPU: 1 socket E4 2690 v4 @2.6 GHz, HT-on
GPU: 2 socket E5-2698 v3 @2.3 GHz, HT off, 1 P40 card in the box

TENSORRT

Networks Supported

- Image Classification (AlexNet, GoogleNet, VGG, ResNet)
- Object Detection
- Segmentation

Not Yet Supported

- RNN/LSTM
- 3D convolutions
- Custom user layers

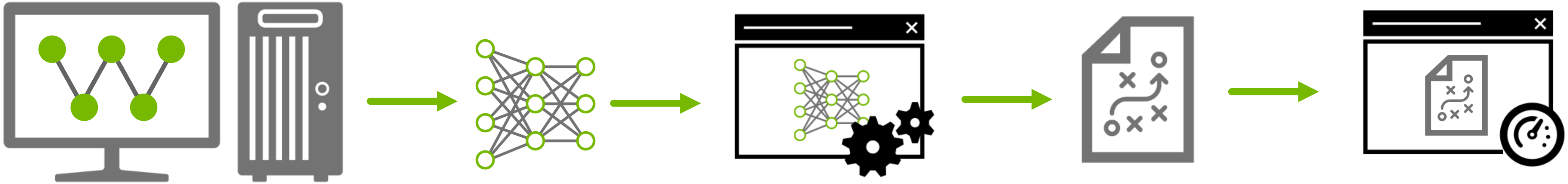
TENSORRT

Layers Types Supported

- **Convolution:** Currently only 2D convolutions
- **Activation:** ReLU, tanh and sigmoid
- **Pooling:** max and average
- **Scale:** similar to Caffe Power layer (shift+scale*x)^p
- **ElementWise:** sum, product or max of two tensors
- **LRN:** cross-channel only
- **Fully-connected:** with or without bias
- **SoftMax:** cross-channel only
- **Deconvolution**

TENSORRT

Workflow



Training Framework

NEURAL NETWORK

OPTIMIZATION USING TensorRT

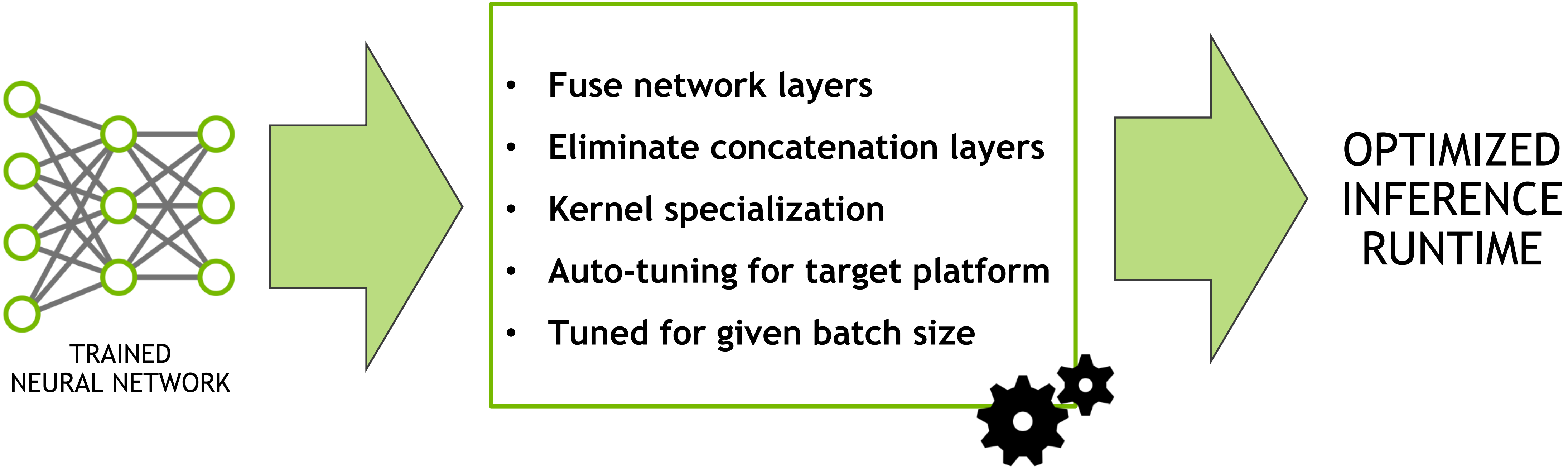
PLAN

RUNTIME USING TensorRT

developer.nvidia.com/tensorrt

TENSORRT

Optimizations



TRAINED NEURAL NETWORK

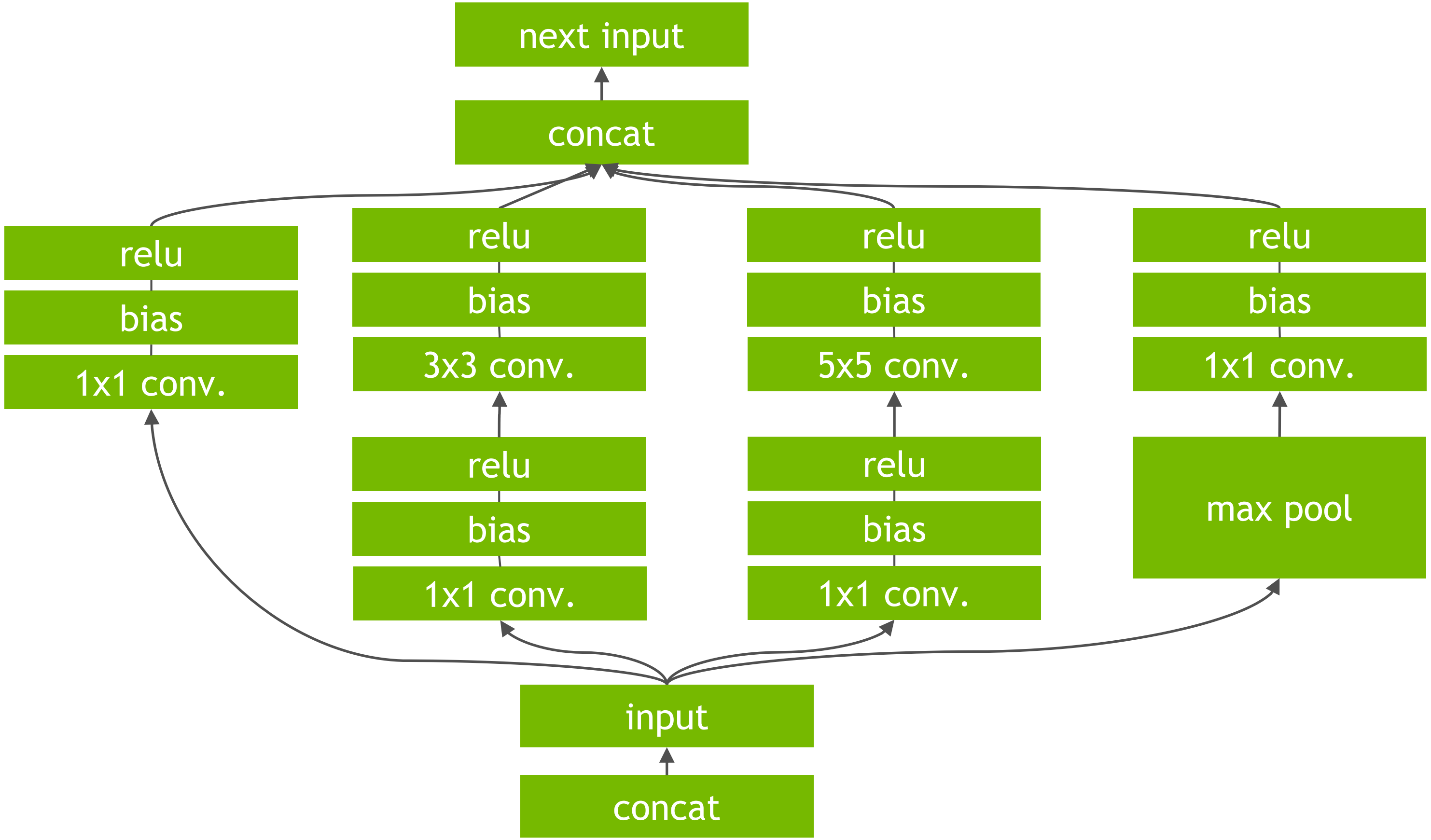
- Fuse network layers
- Eliminate concatenation layers
- Kernel specialization
- Auto-tuning for target platform
- Tuned for given batch size

OPTIMIZED INFERENCE RUNTIME

developer.nvidia.com/tensorrt

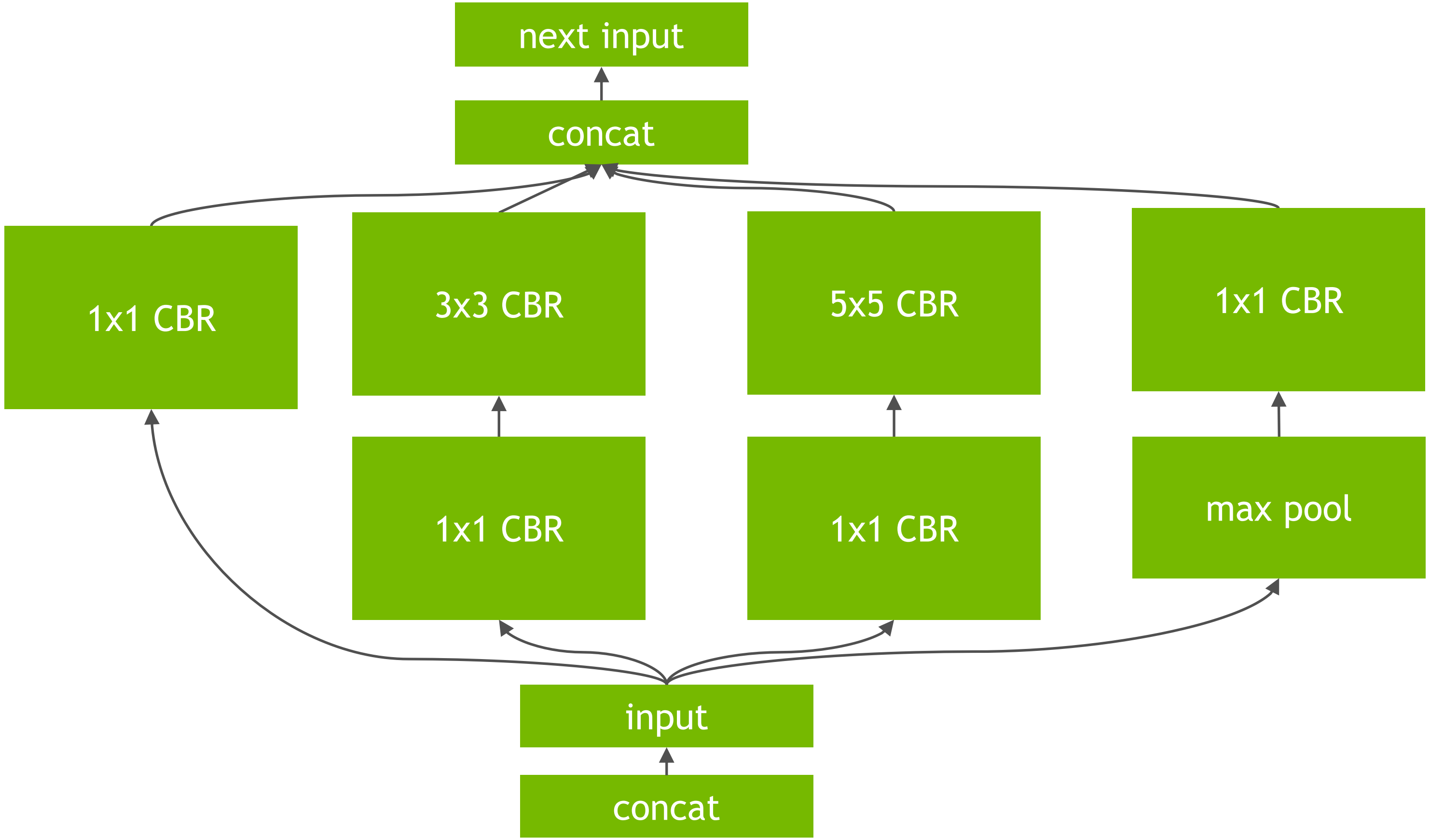
GRAPH OPTIMIZATION

Unoptimized network



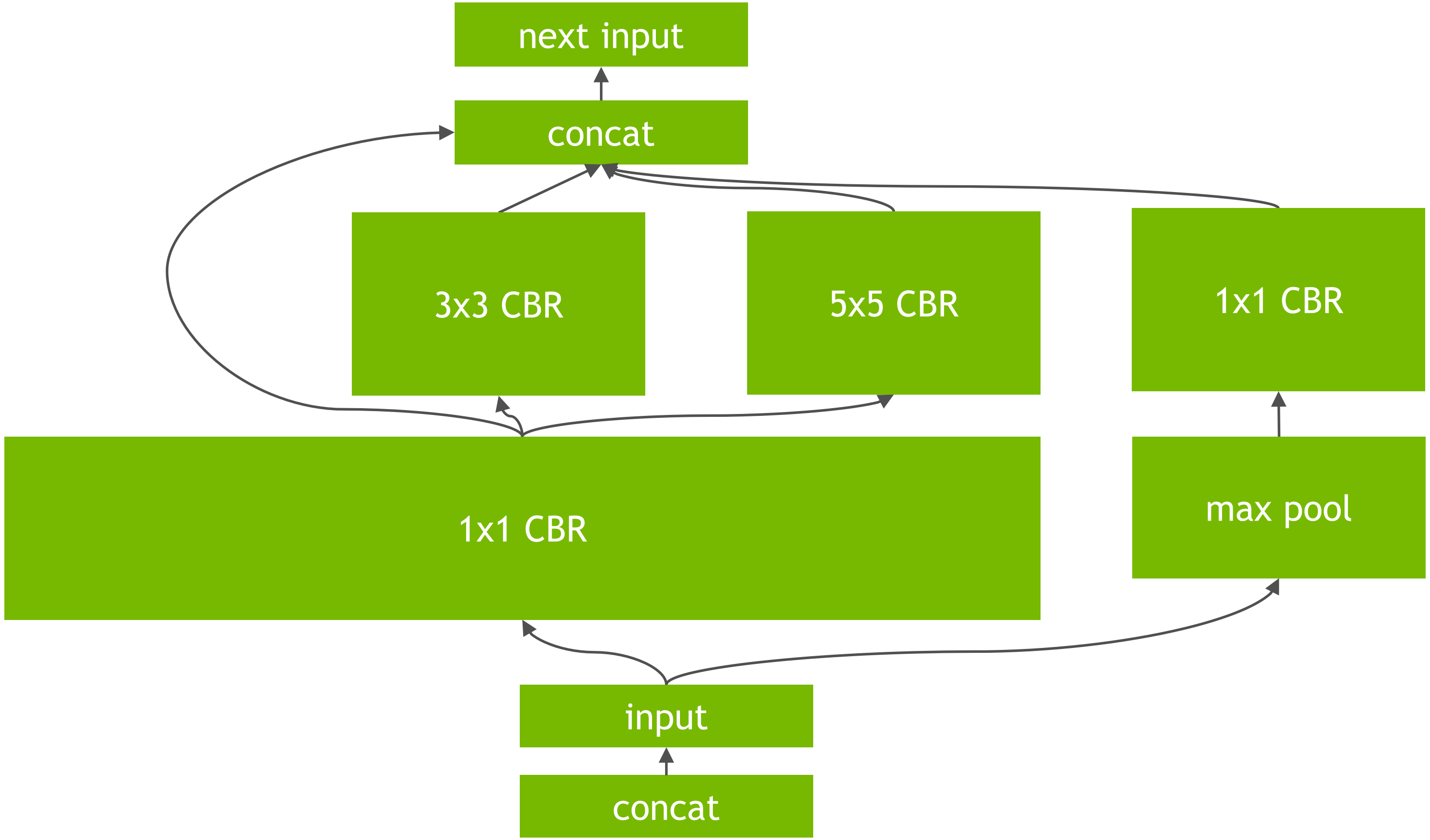
GRAPH OPTIMIZATION

Vertical fusion



GRAPH OPTIMIZATION

Horizontal fusion

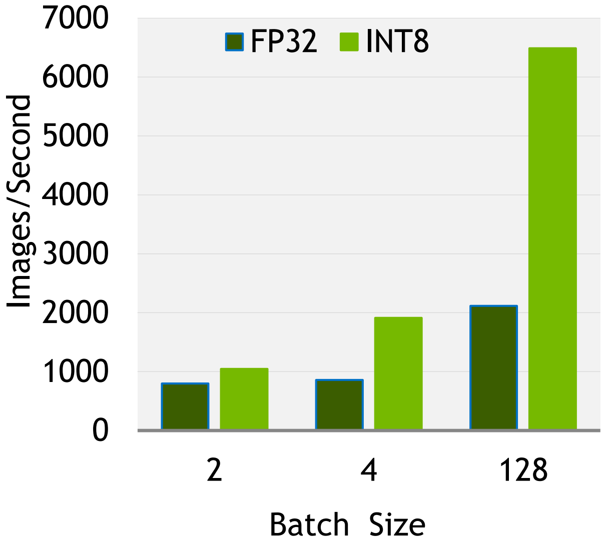


INT8 PRECISION

New in TensorRT

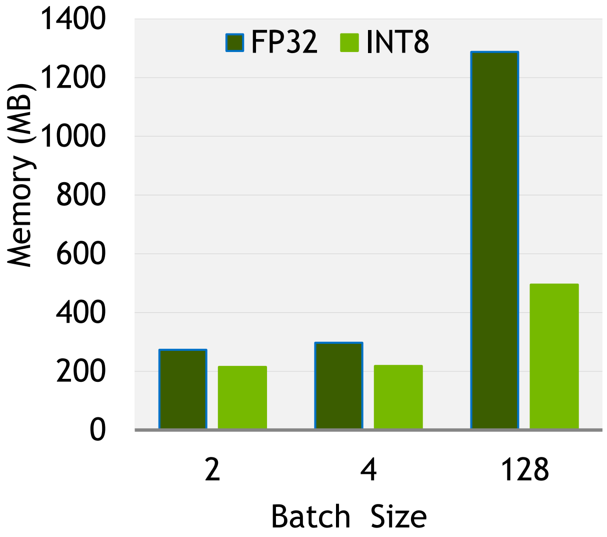
PERFORMANCE

Up To 3x More Images/sec with INT8 Precision



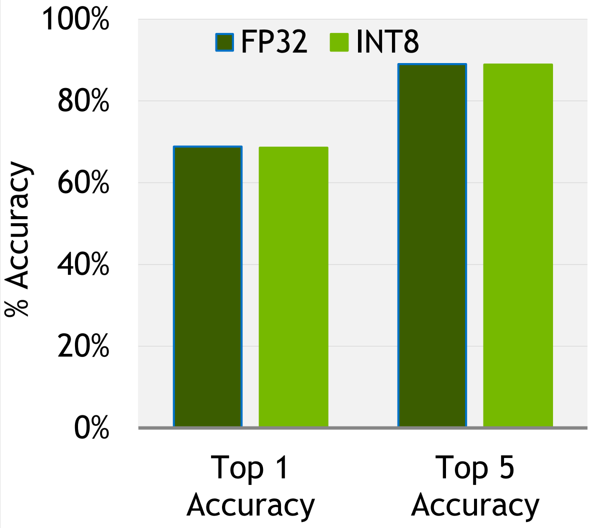
EFFICIENCY

Deploy 2x Larger Models with INT8 Precision



ACCURACY

Deliver full accuracy with INT8 precision



***GoogLenet**, FP32 vs INT8 precision + TensorRT on*
***Tesla P40 GPU**, 2 Socket Haswell E5-2698 v3@2.3GHz with HT off*

THANK YOU