Performance

Performance - Deployment

CATEGORIES OF PERFORMANCE

Requirement	Challenges
High Throughput	**Unable to processing high-volume, high-velocity data** ➢ Impact: Increased cost ($, time) per inference
Low Response Time	**Applications don't deliver real-time results** ➢ Impact: Negatively affects user experience (voice recognition, personalized recommendations, real-time object detection)
Power and Memory Efficiency	**Inefficient applications** ➢ Impact: Increased cost (running and cooling), makes deployment infeasible
Deployment-Grade Solution	**Research frameworks not designed for production** ➢ Impact: Framework overhead and dependencies increases time to solution and affects productivity

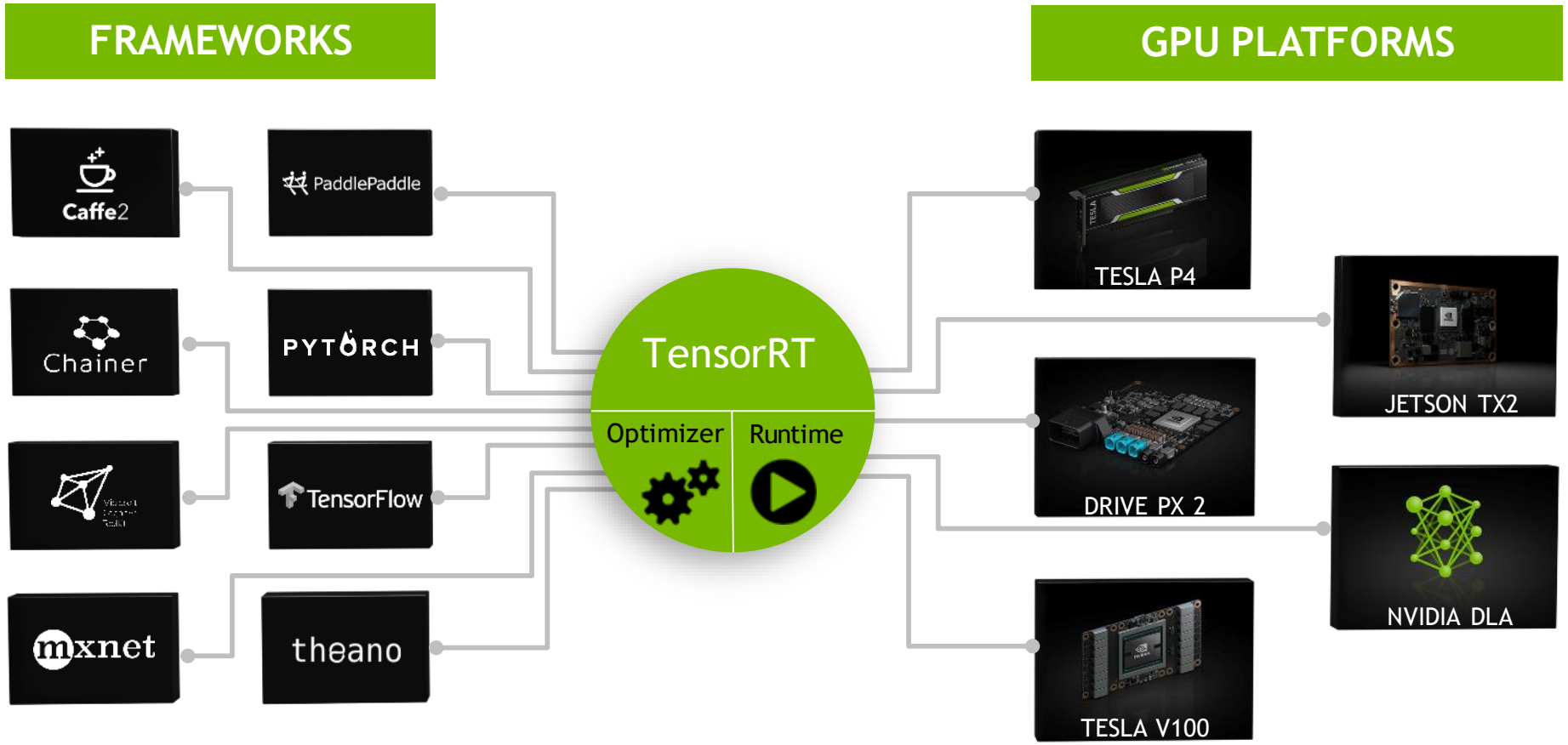
Levers

- Batch size
  - Reduce for less latency
  - Increase for more throughput
- Tools
  - The right deployment platform
  - TensorRT

NVIDIA TENSORRT

NVIDIA TENSORRT

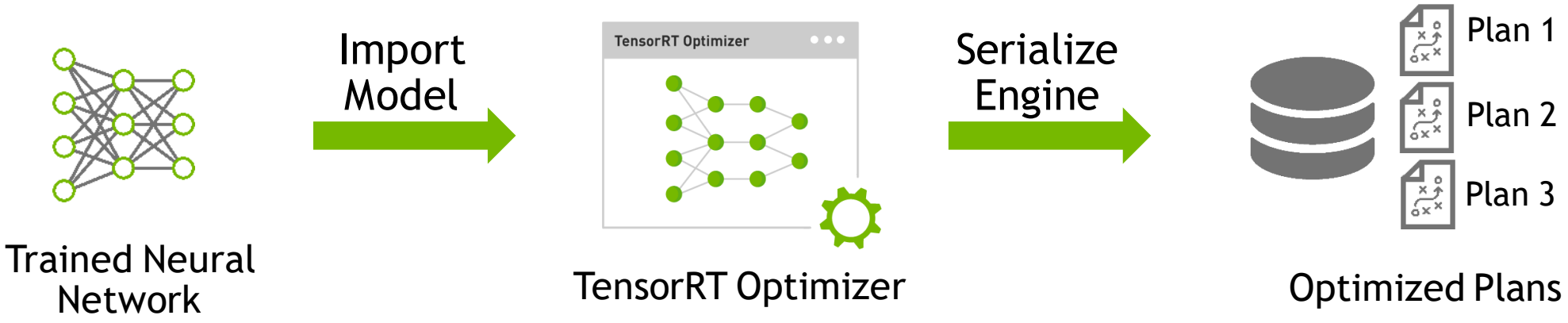
Programmable Inference Accelerator



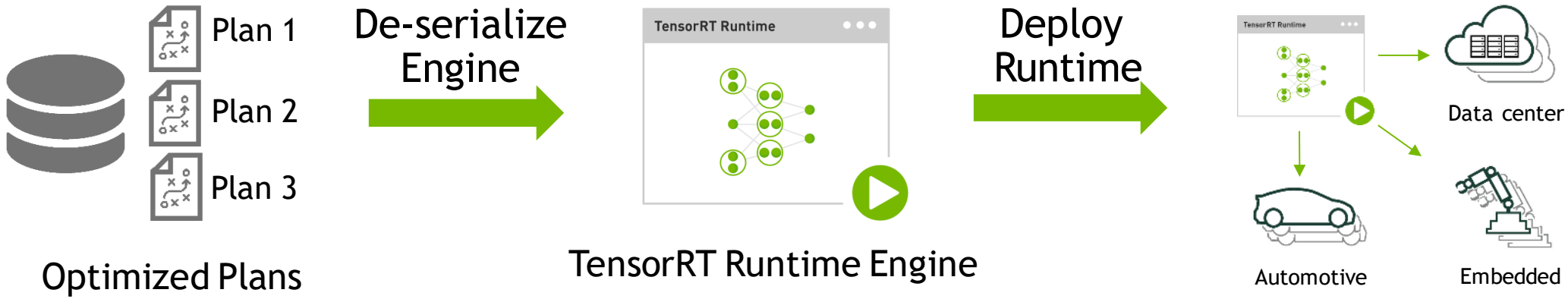
developer.nvidia.com/tensorrt

TENSORRT DEPLOYMENT WORKFLOW

Step 1: Optimize trained model

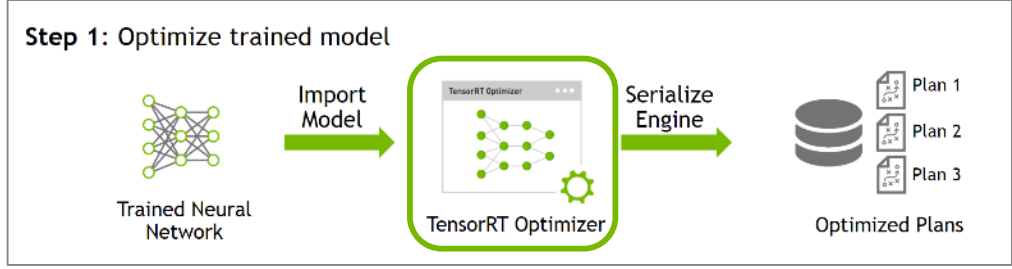


Step 2: Deploy optimized plans with runtime

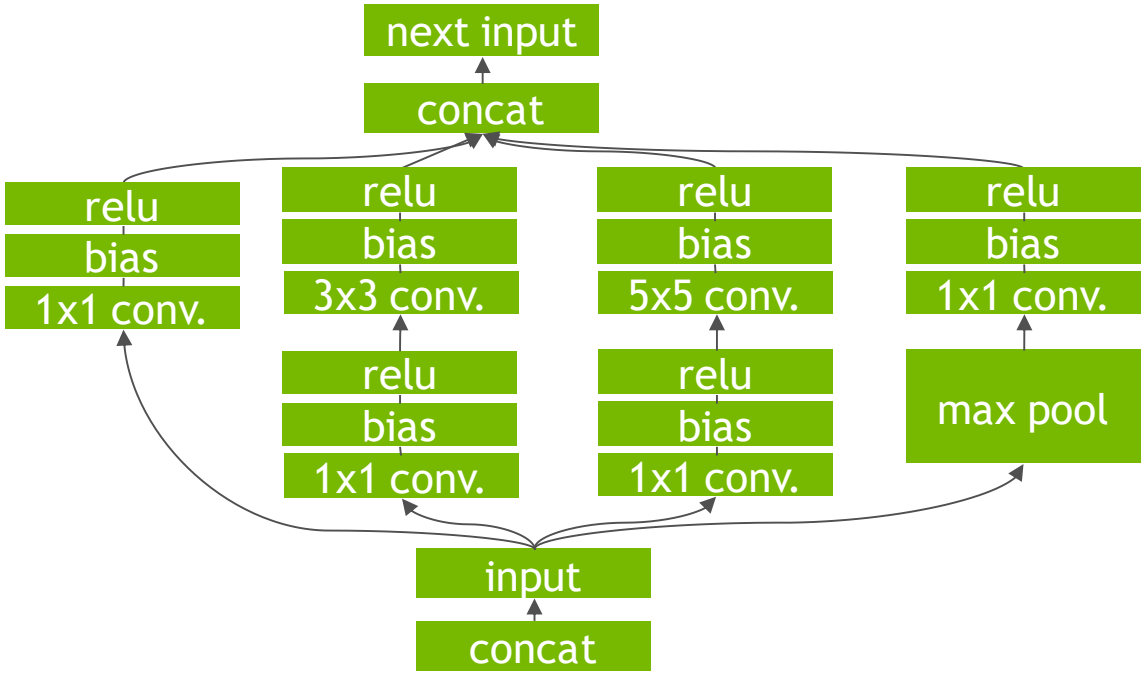


TENSORRT OPTIMIZATIONS

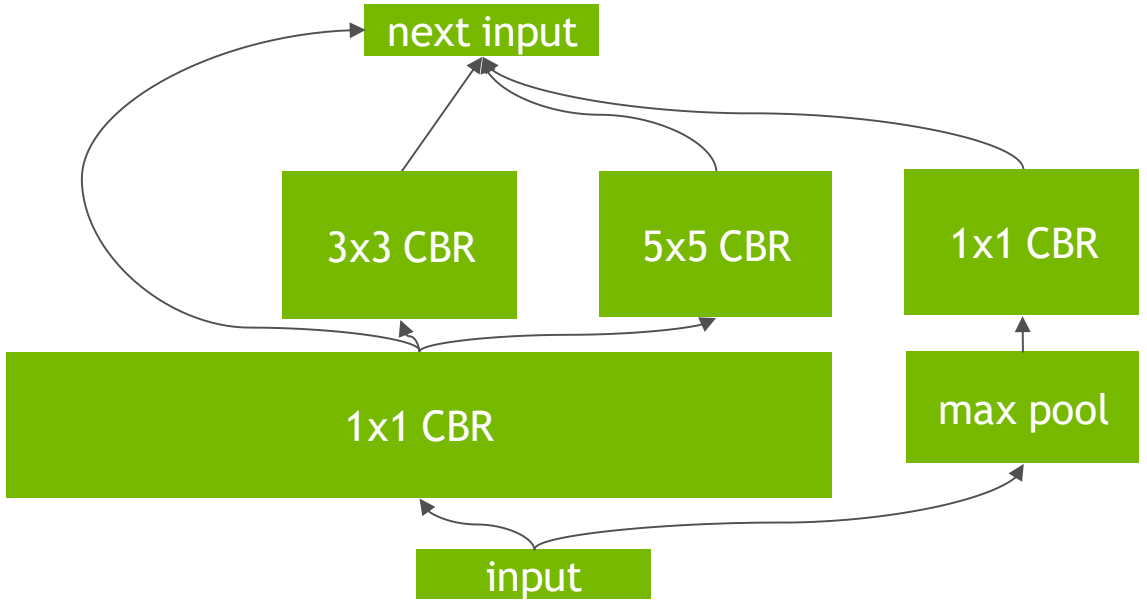
LAYER & TENSOR FUSION



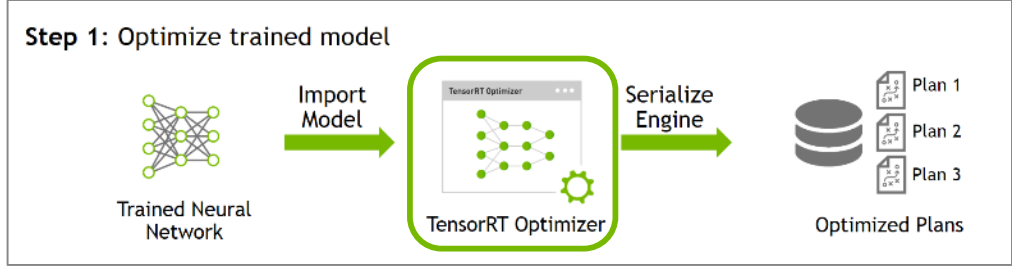
Un-Optimized Network



TensorRT Optimized Network



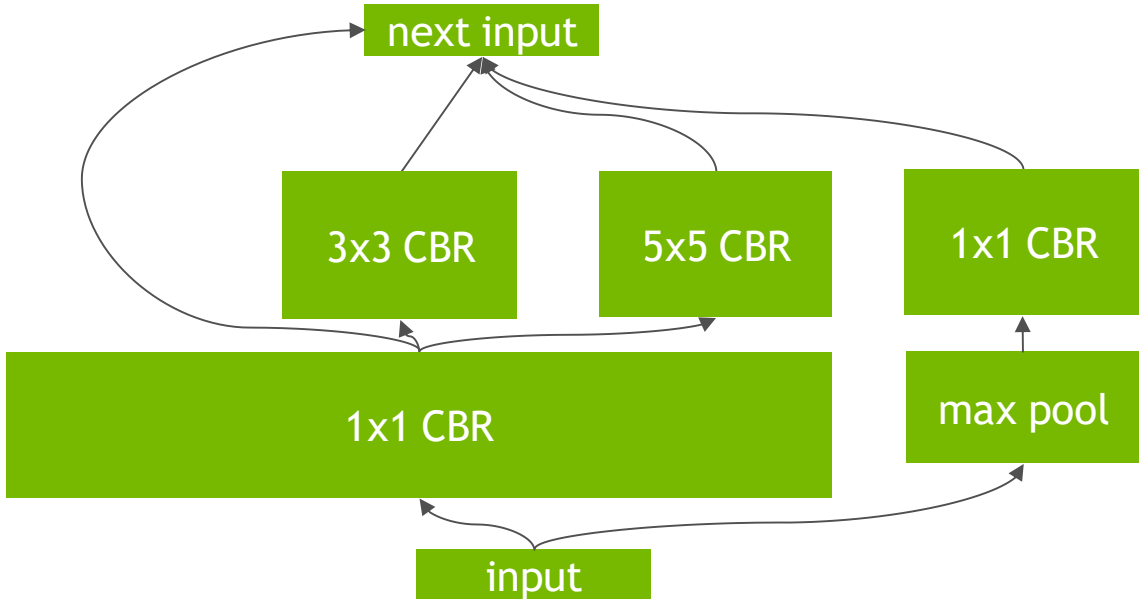
LAYER & TENSOR FUSION



- Vertical Fusion
- Horizonal Fusion
- Layer Elimination

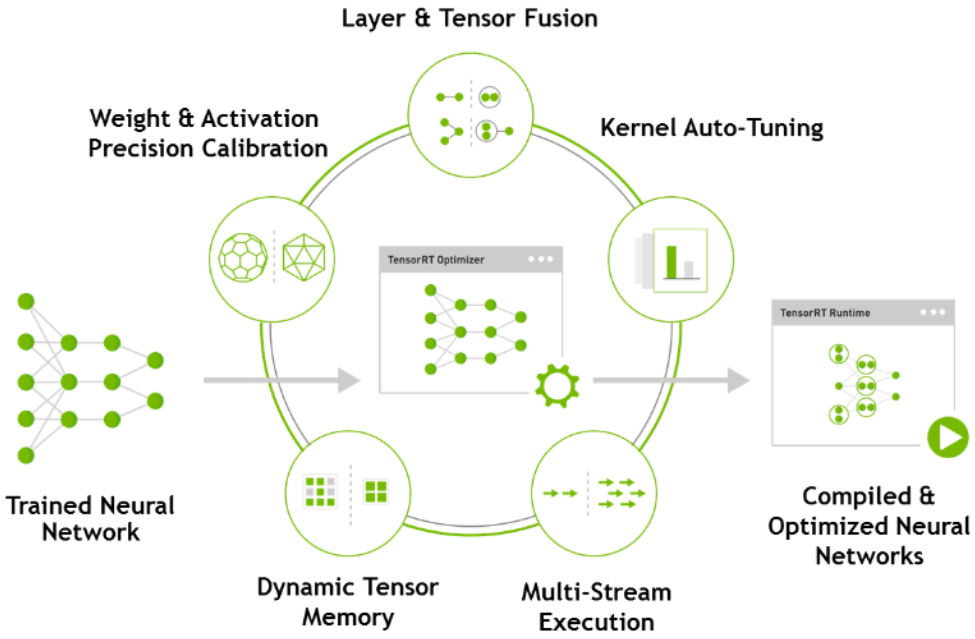
Network	Layers before	Layers after
VGG19	43	27
Inception V3	309	113
ResNet-152	670	159

TensorRT Optimized Network



TENSORRT KEY INFO

- ✓ Generate optimized, deployment-ready runtime engines for low latency inference
- ✓ Import models trained from Caffe or TensorFlow, or use Network Definition API
- ✓ Deploy in FP32 or reduced precision INT8, FP16 for higher throughput
- ✓ Optimize frequently used layers and integrate user defined custom layers



Performance - Training

Next Page

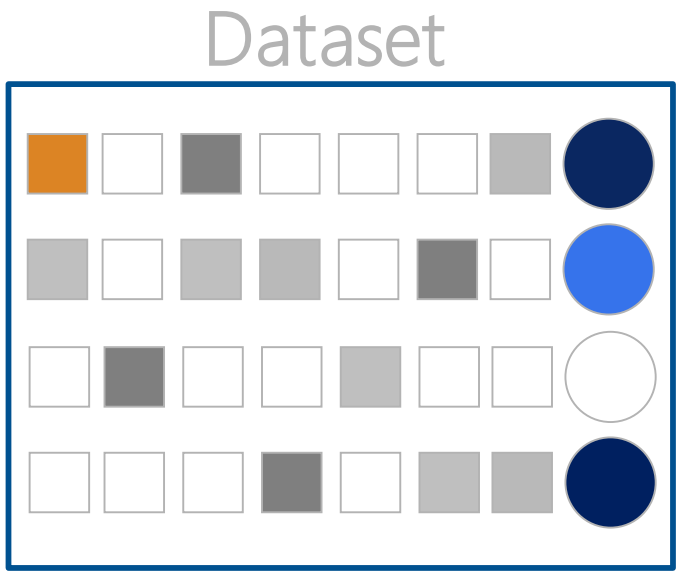
Performance during Training: GPU Task 4

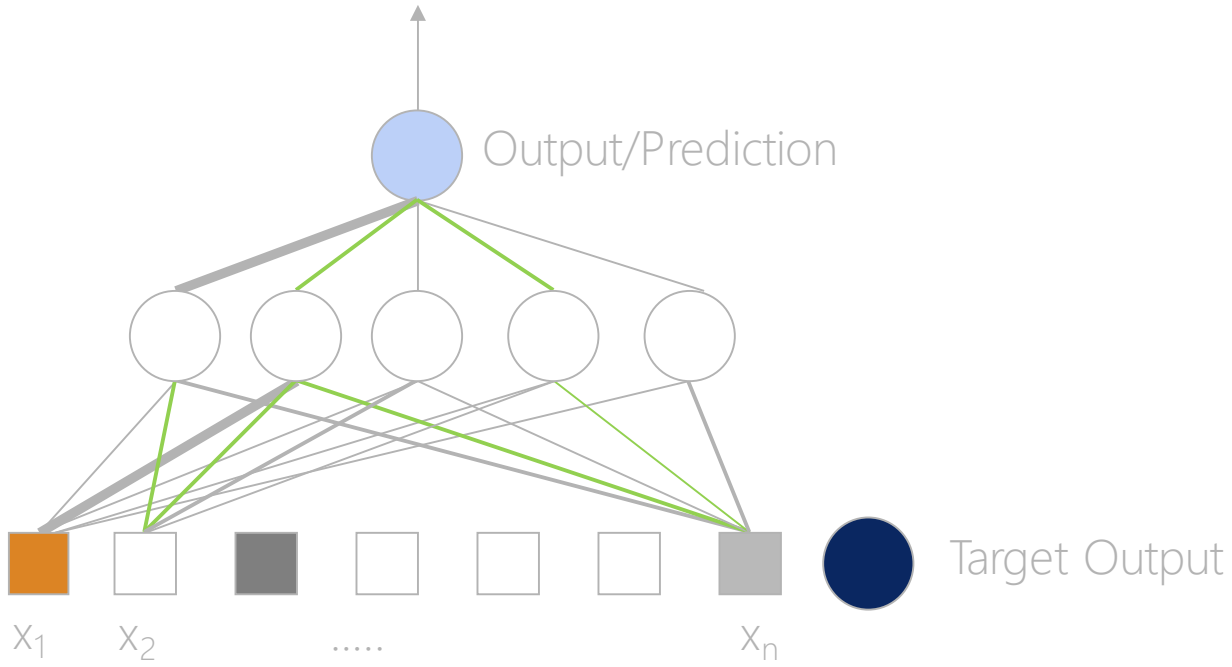
VIEW UNIT IN STUDIO

Bookmark this page

Select **START** to load your next GPU task.

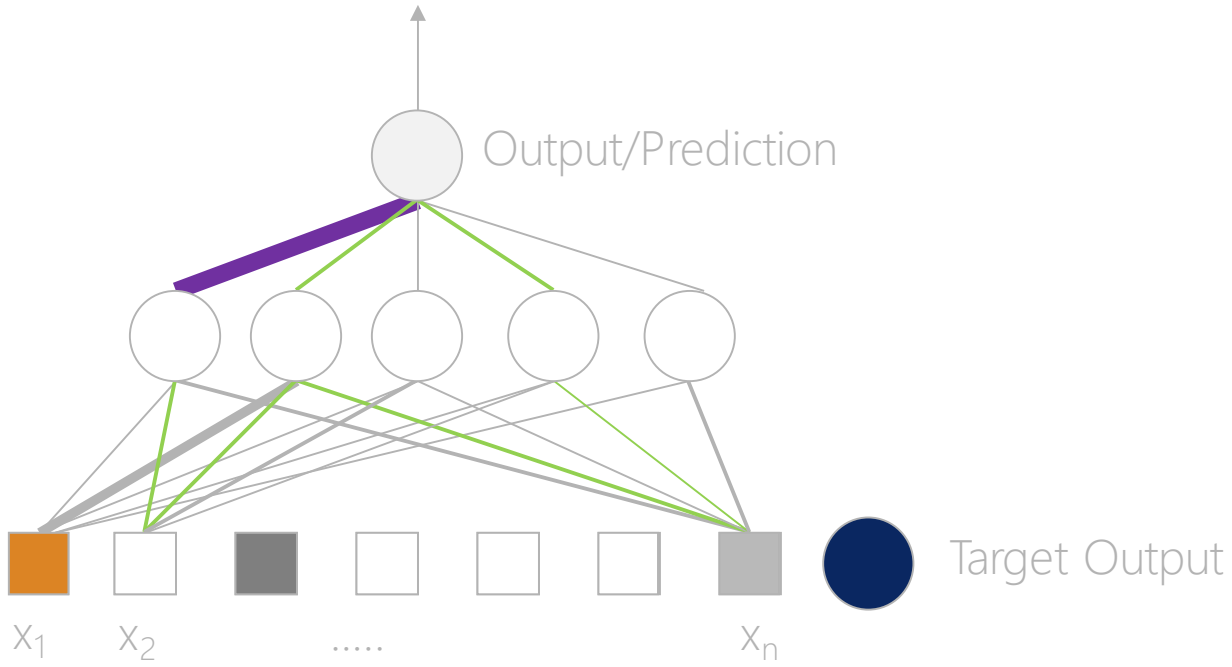
Learning Principle





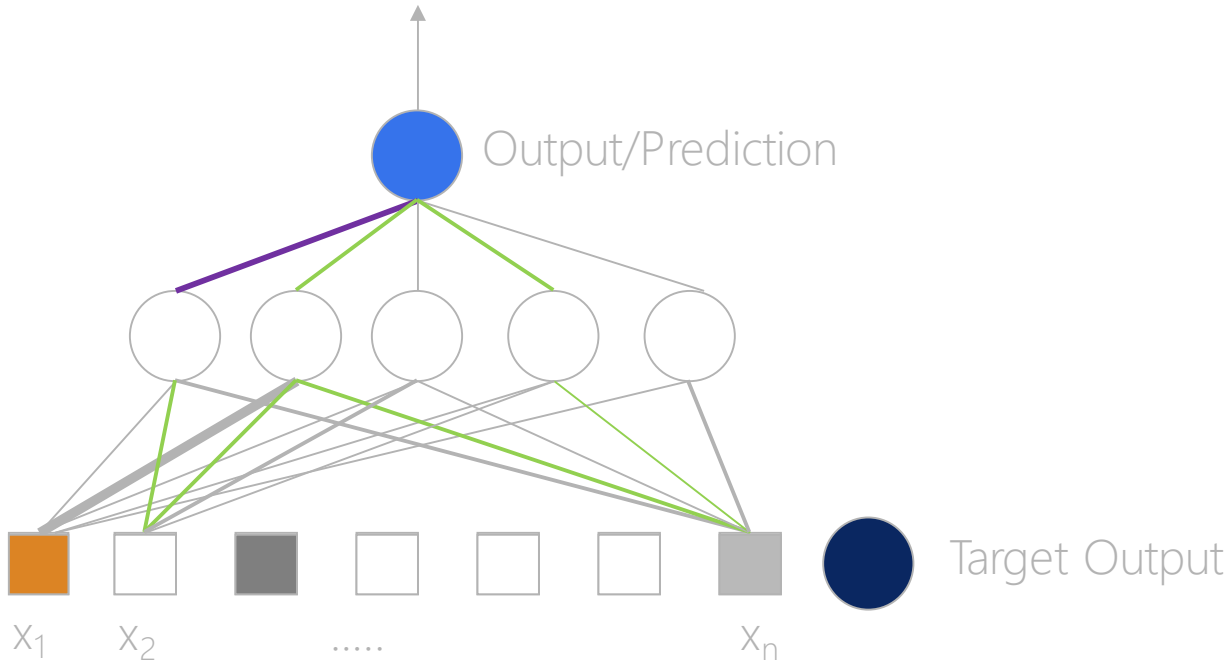
Error:  = 5

Learning Principle



Error:  − = 15

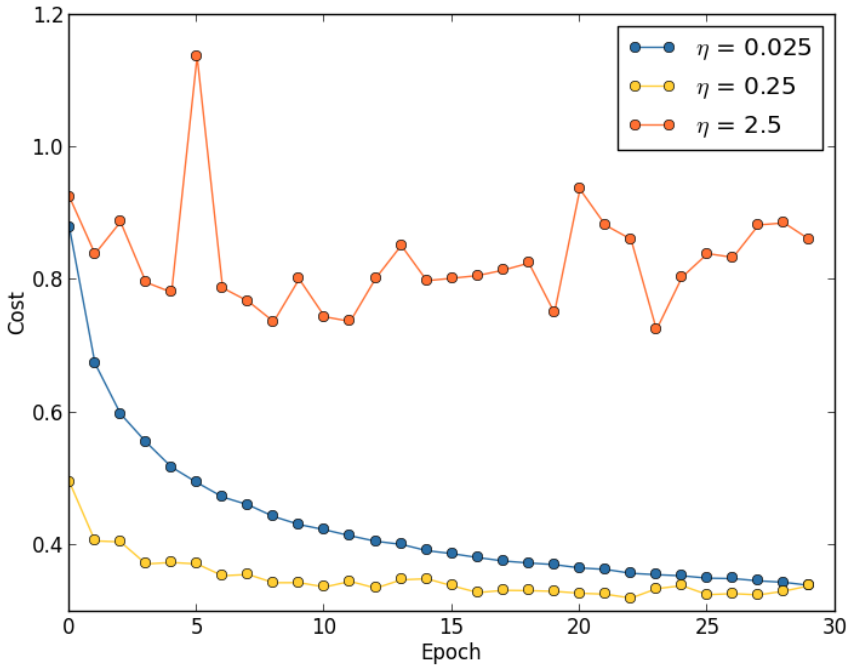
Learning Principle



Error:  −  = 2.5

One HyperParameter: Learning Rate

$$w_t = w_{t-1} + \eta \frac{dE}{dw_t}$$



TECHNIQUES TO IMPROVE MODEL

- More training – GPU Time
- More/better data – Data Science
- Searching Hyperparameters – Learning Design
- Modify the network – Network Architecture – Next Section

Performance Improvement

Ideas?

- Increase accuracy and confidence with similar data

- Generalize performance to more diverse data

More data

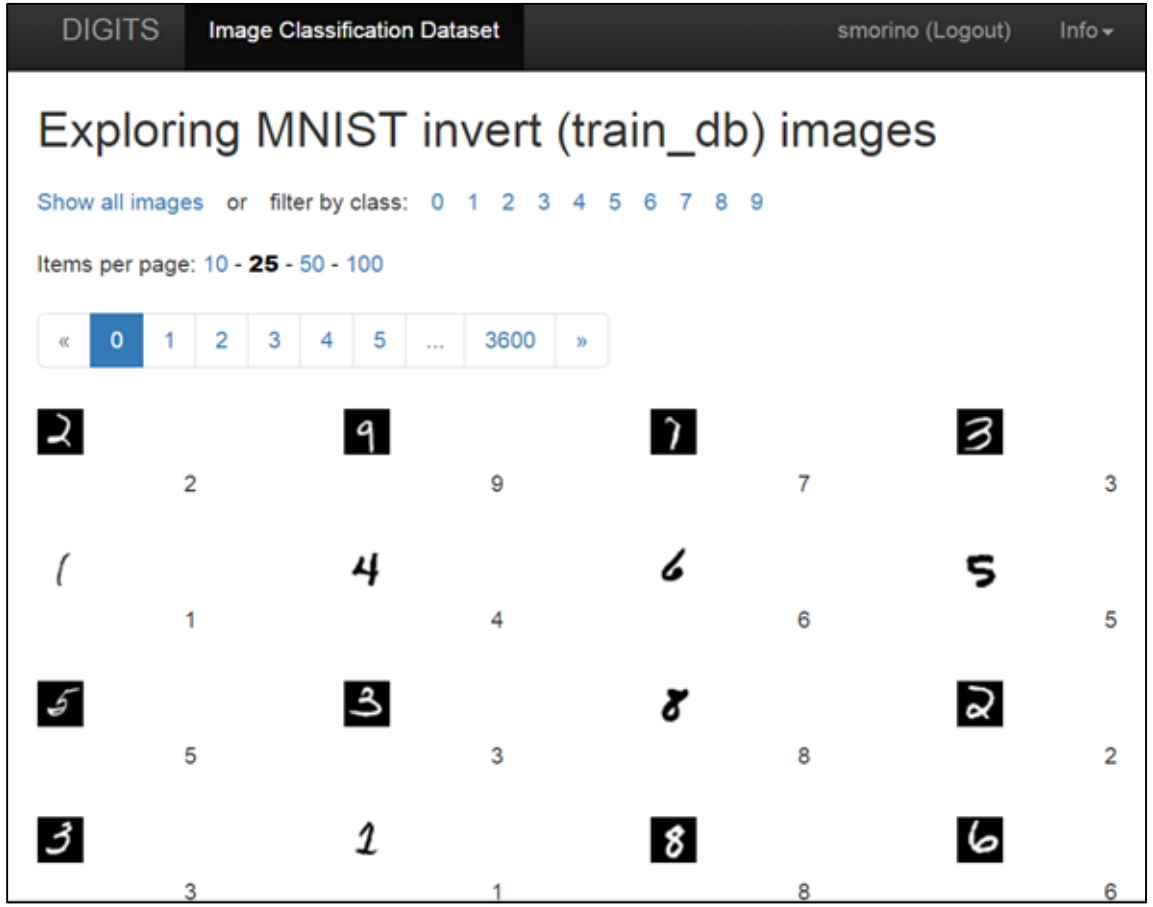
Full dataset (10 epochs)

- 99% of accuracy achieved
- No improvements in recognizing real-world images

	SMALL DATASET	FULL DATASET
	1 : 99.90 %	0 : 93.11 %
	2 : 69.03 %	2 : 87.23 %
	8 : 71.37 %	8 : 71.60 %
	8 : 85.07 %	8 : 79.72 %
	0 : 99.00 %	0 : 95.82 %
	8 : 99.69 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %

DATA AUGMENTATION

Adding Inverted Images



- Pixel(Inverted) = 255 – Pixel(original)
- White letter with black background
  - Black letter with white background
- Training Images: /home/ubuntu/data/train_invert
- Test Image: /home/ubuntu/data/test_invert
- Dataset Name: MNIST invert

DATA AUGMENTATION

Adding inverted images (10 epochs)

	SMALL DATASET	FULL DATASET	+INVERTED
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %