Integrating AI into Your Accelerated Cloud Applications - Xilinx ML(Machine Learning) Suite

정 웅
DSP Specialist / 부장
2018-07-03

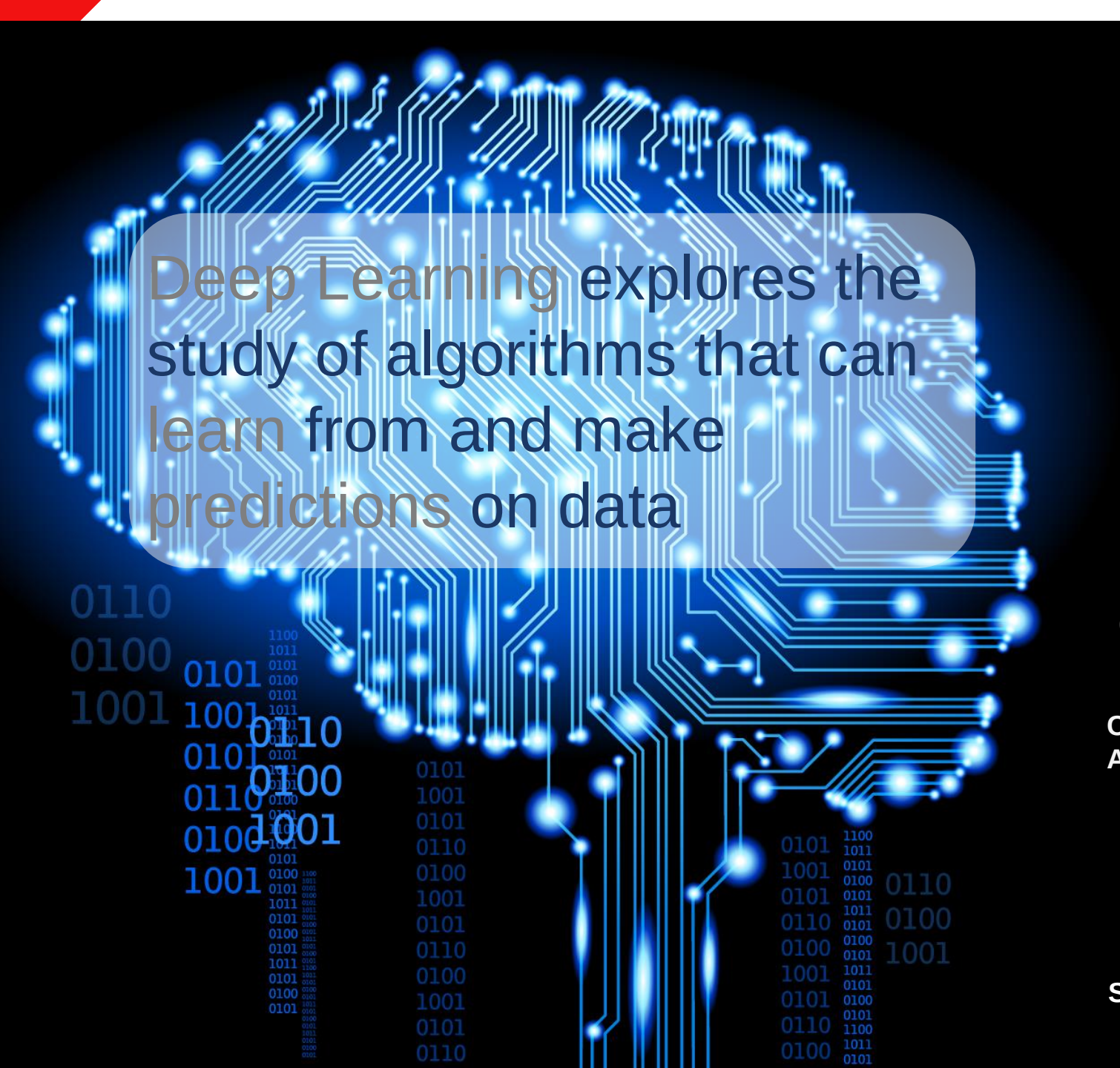
XILINX.

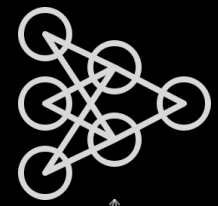
Agenda

- Xilinx & Deep learning
- ML-suite for CNN
  - Processing Engine
  - Middle Ware
  - Getting Start Today
- Tutorial Walk Through
  - Deploying(Compiling and Quantizing) with Caffe
  - Python APIs

Xilinx & Deep Learning

XILINX

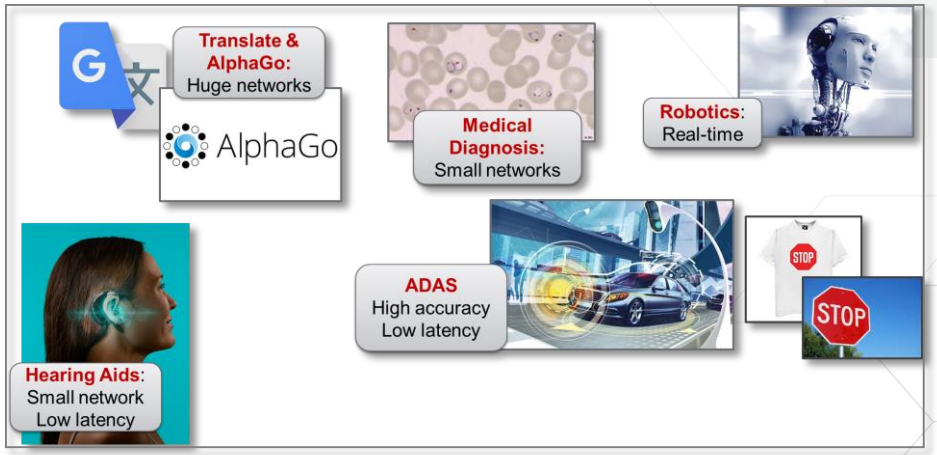




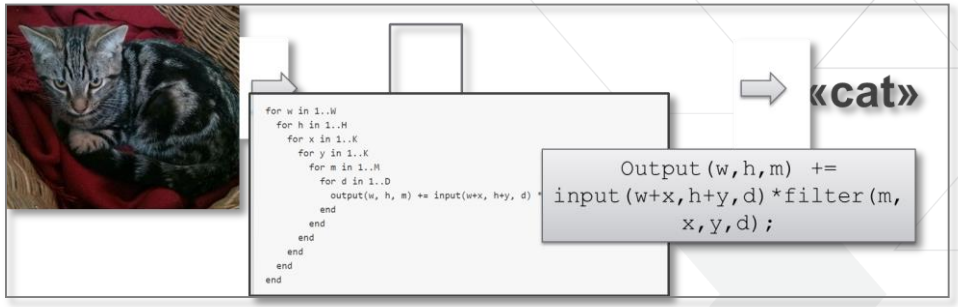
Deep Learning is Re-Defining many Applications

Deep Learning Challenges – Solution Flexibility is Key

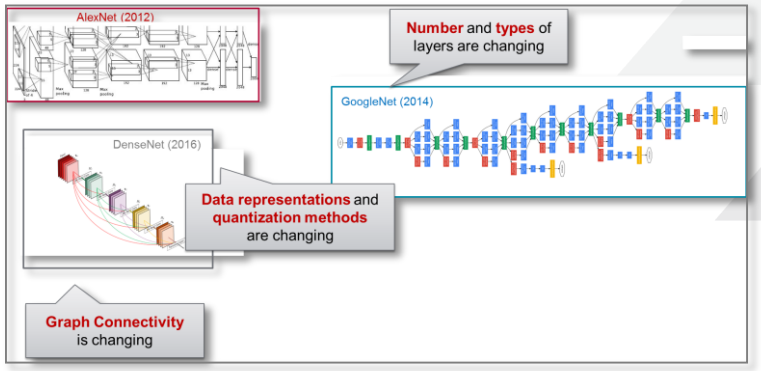
Challenge 1:Different use cases require different networks & different figures of merits
Speed, latency, energy, accuracy



Challenge 2: billions of multiply-accumulate ops & tens of megabytes of parameter data
Sea of operators, custom math & memory hierarchy

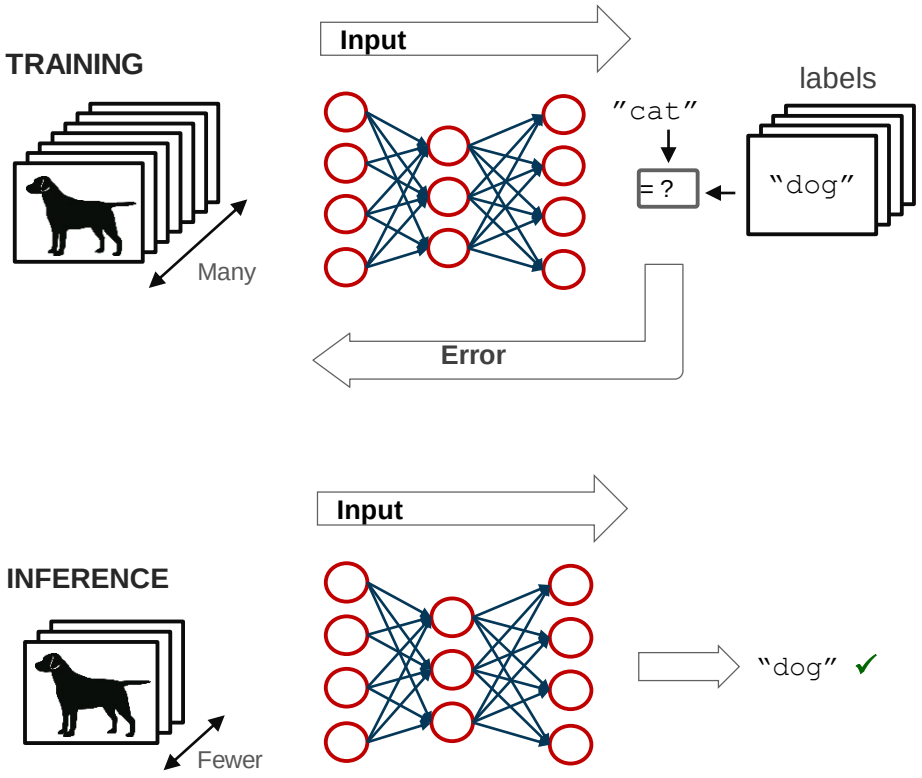


Challenge 3: Continuous stream of new algorithms
Hardening architectures is risky



XILINX

The Divergence of Training and Inference in Deep Learning



Training: Process for machine to "learn" and optimize model from data

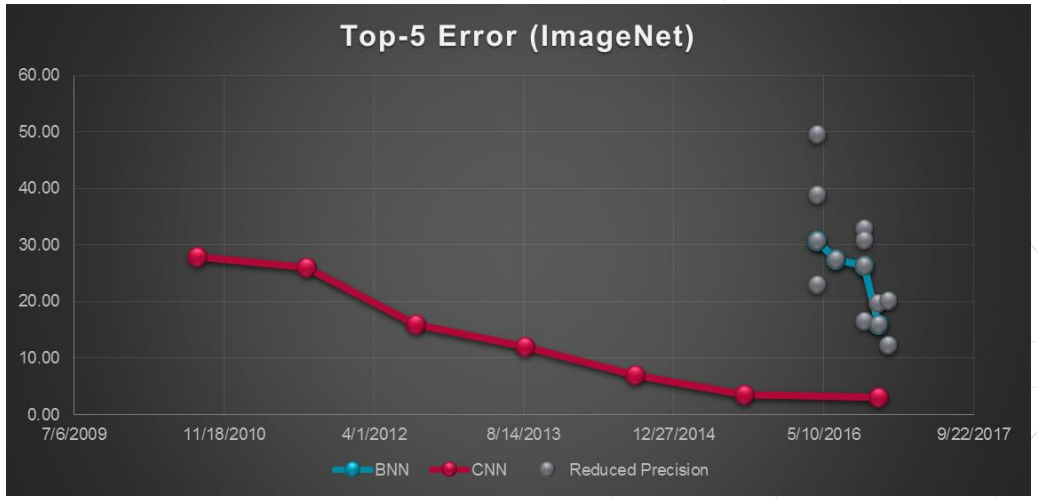
Inference: Using trained models to predict/estimate outcomes from new observations in efficient deployments

https://arxiv.org/pdf/1510.00149v5.pdf

Latest Inference Algorithmic Research

Optimize Compute with Reduced Precision CNNs & BNNs

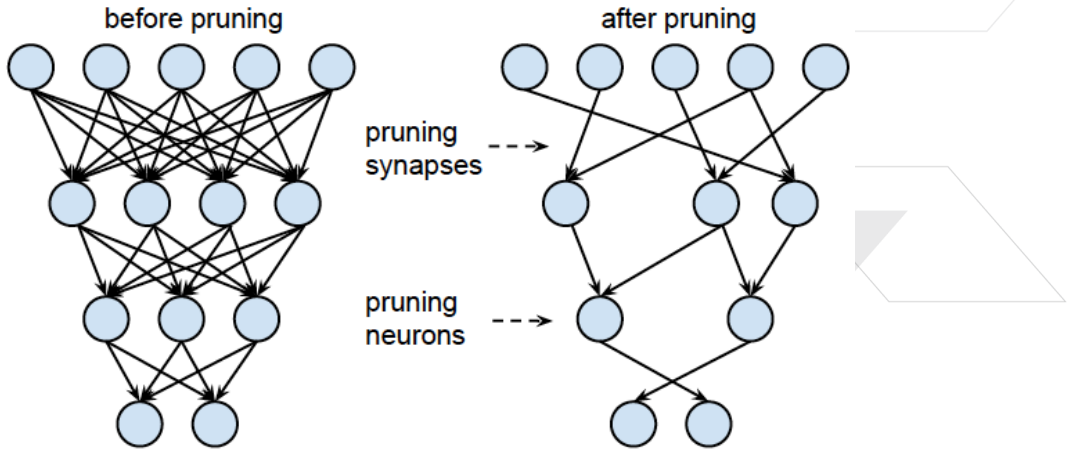
- 8 bit solution loses no significant accuracy
- BNNs are improving rapidly
- Custom Floating point provide significantly more compute density



Not only weights, but also sparsity pattern encodes information

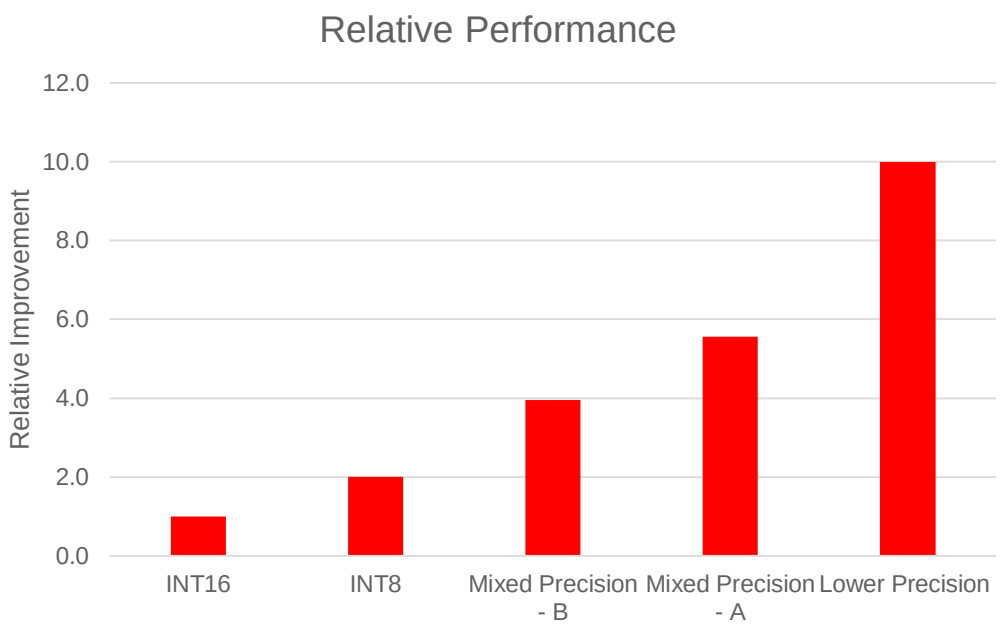
Model Pruning for higher effective performance: LSTM

- TIMIT 3-hour dataset – 20x
- LibreSpeech 100-hour dataset – 10x
- CustomerS1000-hour dataset – 20x
- CustomerS 3000-hour dataset – 5x

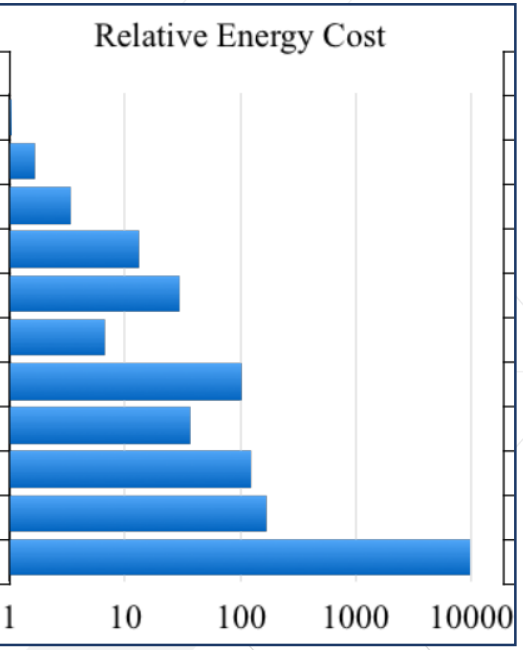


Reduced Precision saves Memory and can scale compute

- **Implementation cost and power per operation is greatly reduced**
  - **100 TOP/s inference is possible on Data center FPGA today**



Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

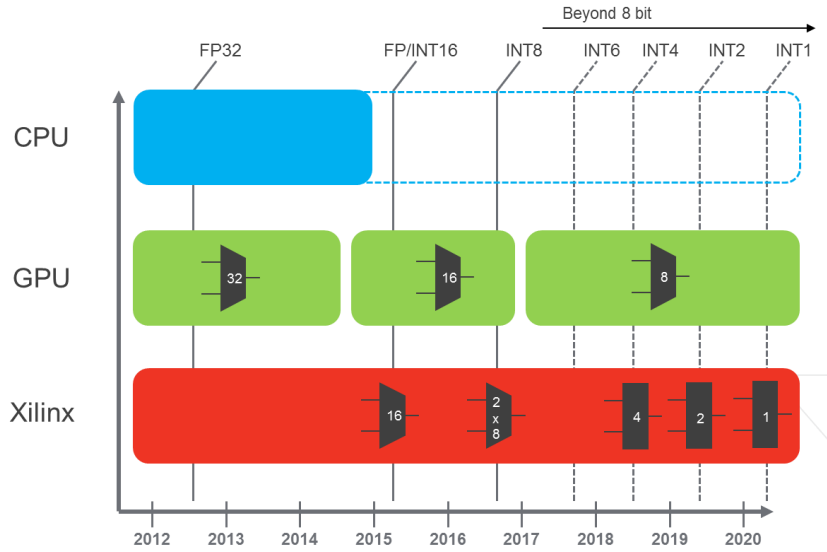


Source: Bill Dally (Stanford), Cadence ***Embedded Neural Network Summit, February 1, 2017***

- **Memory cost is greatly reduced**
  - Large networks can fit entirely into on-chip memory
  - (OCM) (UltraRAM, BRAM)

Xilinx Features for Implementing Efficient Inference Engines

Flexible Architecture for Any Precision

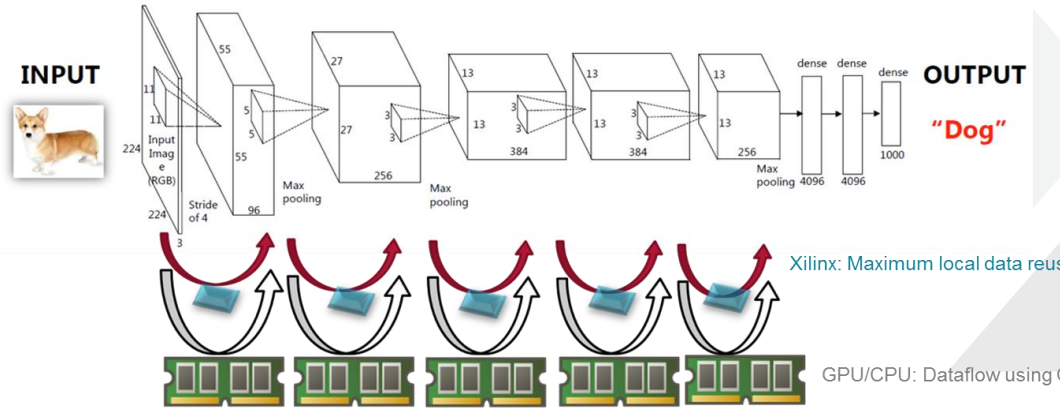


Limited to 32 bit operations

New devices required to support change in precision efficiently

Reconfigurable to scale and optimize for different precisions

Flexible On-chip Memory for low latency

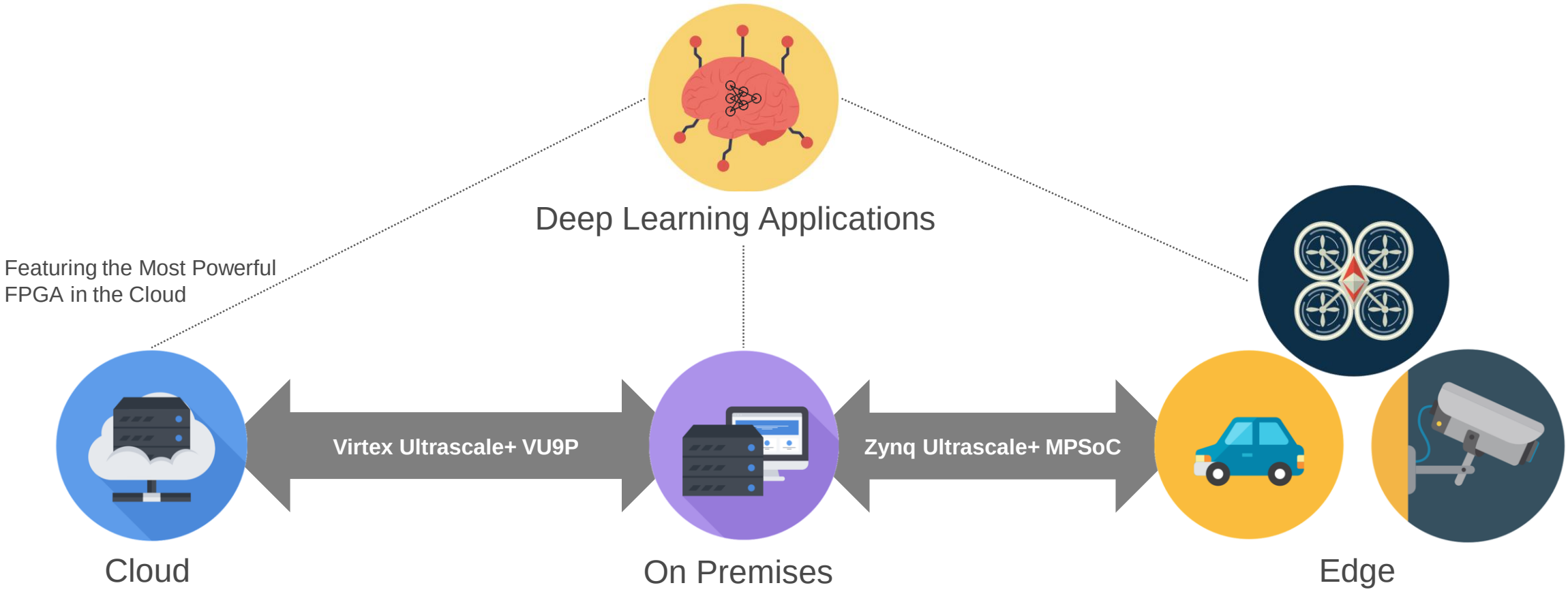


Xilinx: Maximum local data reuse "layer merging" between layers

GPU/CPU: Dataflow using Off chip memory

XILINX

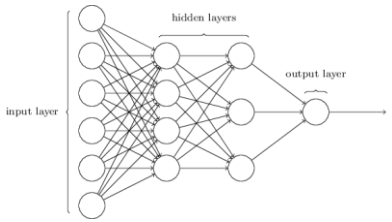
Accelerating AI Inference into Your Cloud Applications

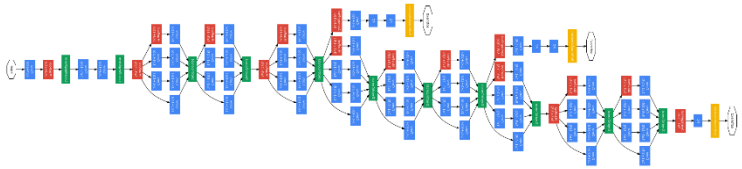


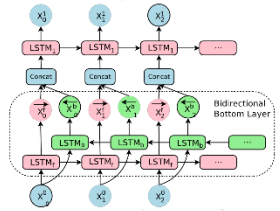
Xilinx ML-suite

XILINX

Deep Learning Models







Multi-Layer Perceptron

- Classification
- Universal Function Approximator
- Autoencoder

Convolutional Neural Network

- Feature Extraction
- Object Detection
- Image Segmentation

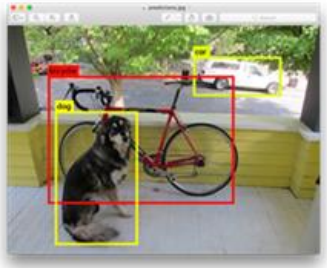
Recurrent Neural Network

- Sequence and Temporal Data
- Speech to Text
- Language Translation

Classification

“Dog”

Object Detection



Segmentation

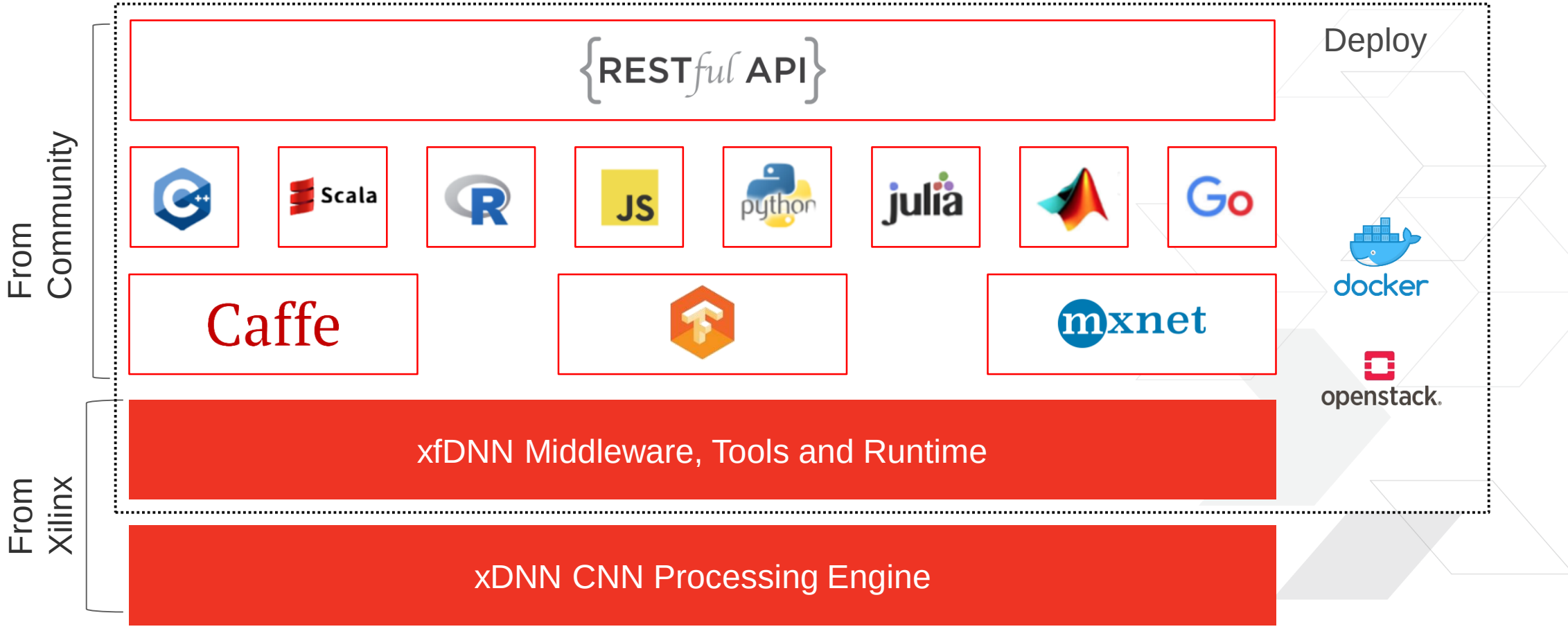
ML Suite Features Today

- **Based on xDNN v2**
- **Supported Frameworks:**
  - Caffe
  - MxNet
  - Tensroflow
- **Examples**
  - DeepDetect REST Tutorial
  - DeepDetect Webcam
  - Image Classification
  - x8 FPGA Pooling GoogLeNet v1 Demo on AWS F1 instance
- **xfDNN Tools**
  - Compiler
  - Quantizer
- **Easy to use Python Interface**
- **Precompiled Models**
  - 8/16-bit GoogLeNet v1
  - 8/16-bit ResNet50
  - 8/16-bit Flowers102
  - 8/16-bit Places365
  - 8/16-bit GoogleNet v3
  - Face Detection
  - Yolo-v2 for object detection
- **Supported Server Platforms**
  - Intel x86
- **Supported FaaS**
  - AWS F1
- **Xilinx SDx boards**
  - VCU1525
  - VCU1526

ML suite - SW
Xilinx Inference Middleware - xfDNN

XILINX

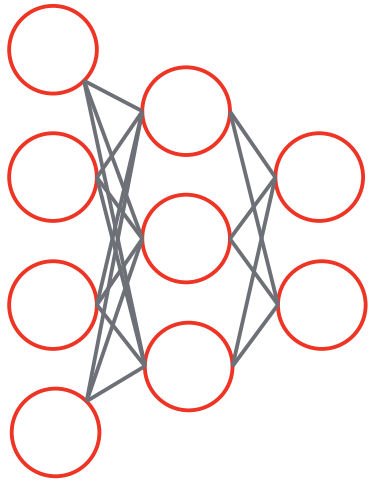
Seamless Deployment with Open Source Software



*TensorFlow Q4 2017

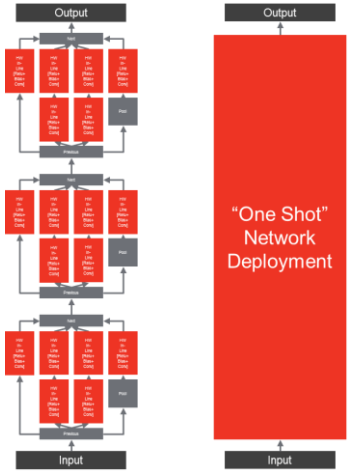
xfDNN Inference Toolbox

Graph Compiler



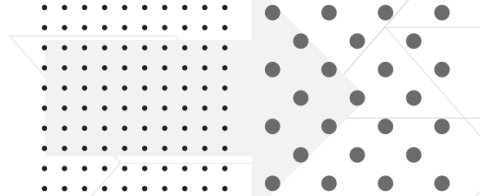
- Build network graphs from Frameworks
- Optimizes for Inference
- Generate code for xDNN IP
- HW/SW partition

Network Optimization



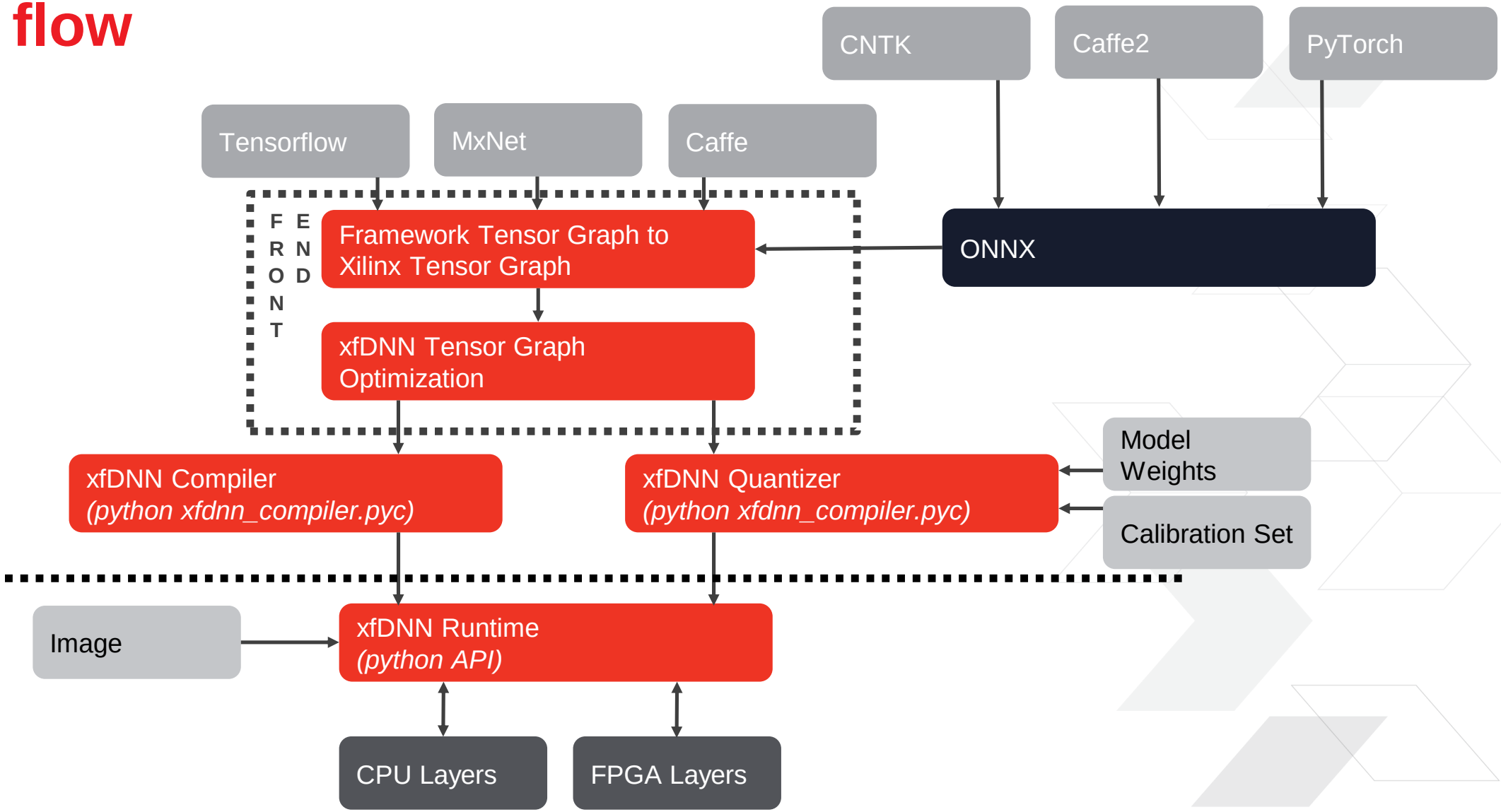
- Fused Layer Optimization
- On-Chip Memory enables Streaming
- "One Shot" Inference eliminates CPU calls

xfDNN Quantizer



- Easily deploy pre-trained floating point models on 8 bit.
- Maintains accuracy without needing lengthy retraining
- Easy and Fast

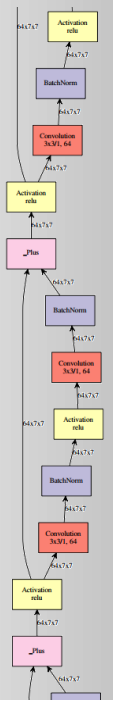
xfDNN flow



https://github.com/Xilinx/ML-Development-Stack-From-Xilinx

xfDNN Graph Compiler

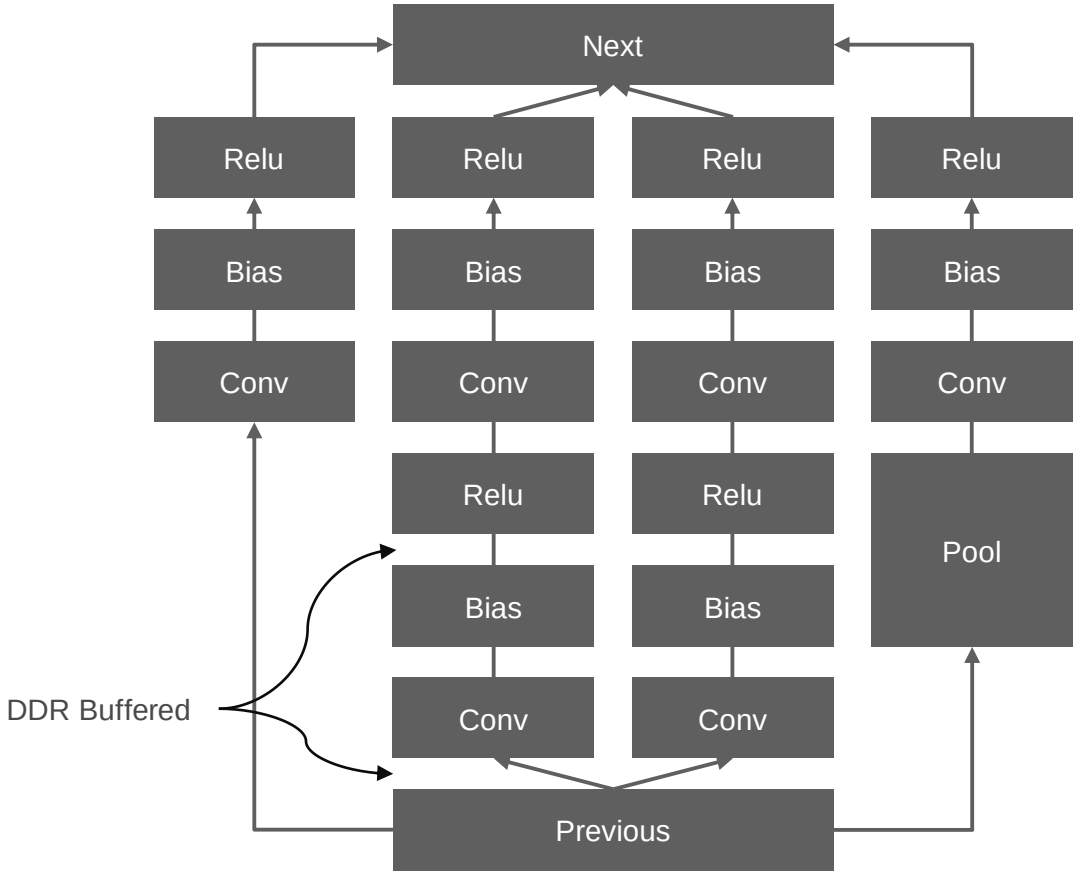
Pass in a Network



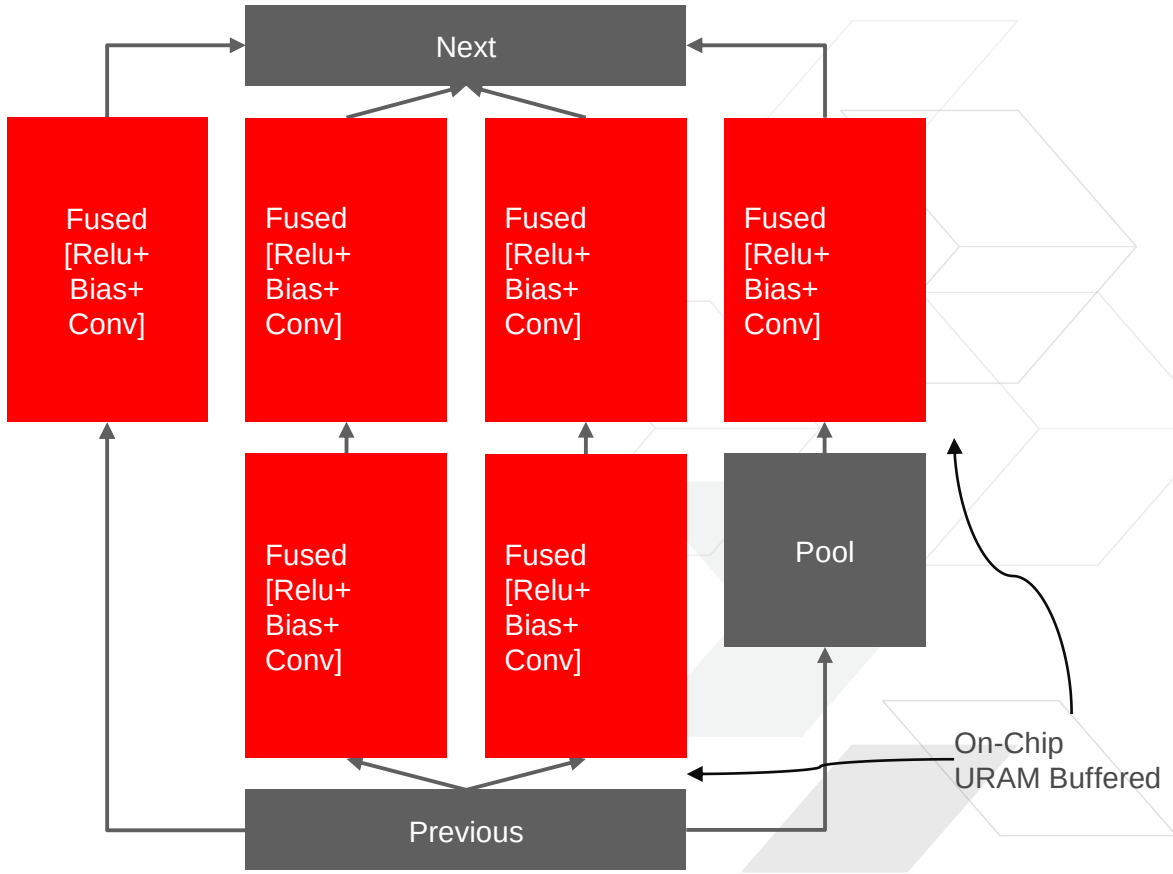
xfDNN
Graph Compiler

Microcode for xDNN is Produced

xfDNN Network Optimization
Layer to Layer



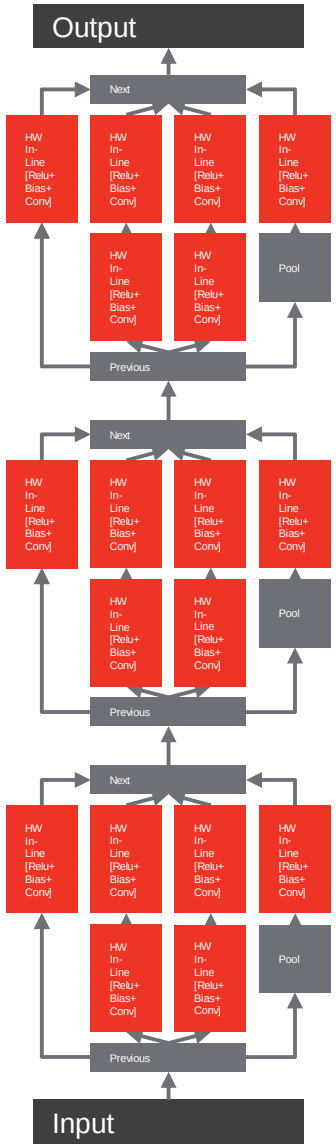
Unoptimized Model



xfDNN Intelligently Fused layers
Streaming optimized for URAM

XILINX

xfDNN Network Deployment



Fused Layer Optimizations

- Compiler can merge nodes
  - (Conv or EltWise)+Relu
  - Conv + Batch Norm
- Compiler can split nodes
  - Conv 1x1 stride 2 -> Maxpool+Conv 1x1 Stride 1

On-Chip buffering reduces latency and increases throughput

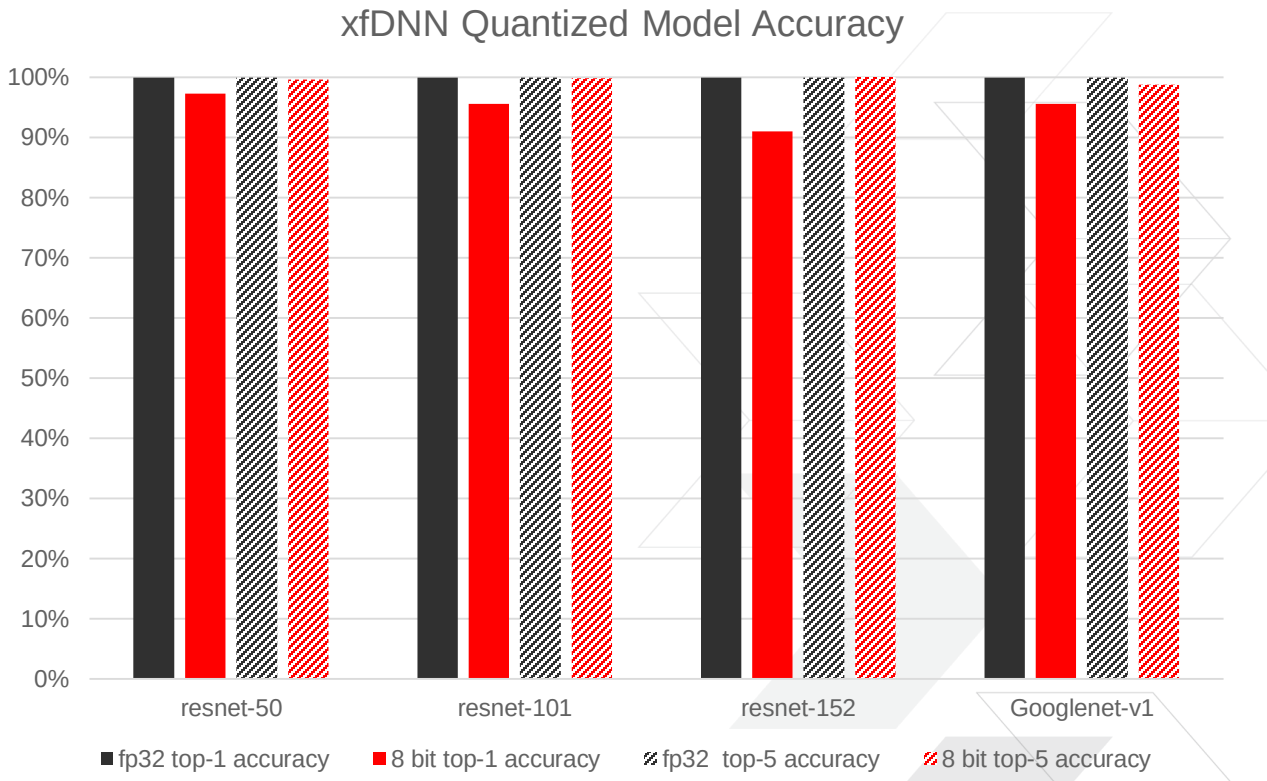
- xfDNN analyzes network memory needs and optimizes scheduler
  - For Fused and "One Shot" Deployment

"One Shot" deploys entire network to FPGA

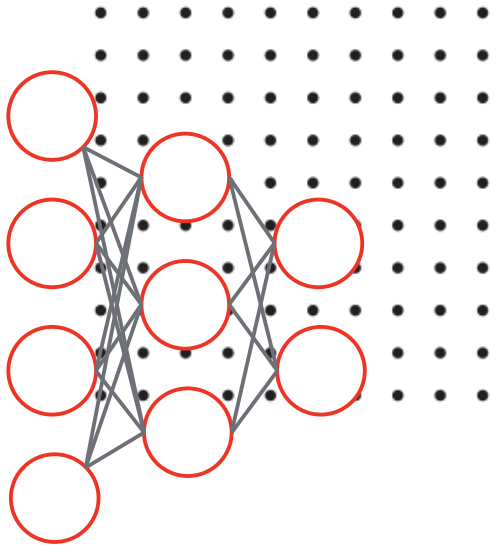
- Optimized for fast, low latency inference
- Entire network, schedule and weights loaded only once to FPGA

xfDNN Quantizer: FP to Fixed-Point Quantization

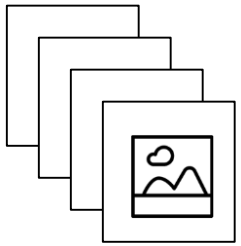
- **Problem:**
  - Nearly all trained models are in 32-bit floating-point
  - Available Caffe and TensorFlow quantization tools take hours and produce inefficient models
  -
- **Introducing: xfDNN Quantizer**
  - A customer friendly toolkit that automatically analyses floating-point ranges layer-by-layer and produces the fixed-point encoding that looses the least amount of information
    - Quantizes GoogleNet in under a minute
    - Quantizes 8-bit fixed-point networks within 1-3% accuracy of 32-bit floating-point networks
    - Extensible toolkit to maximize performance by searching for minimal viable bitwidths and prune sparse networks



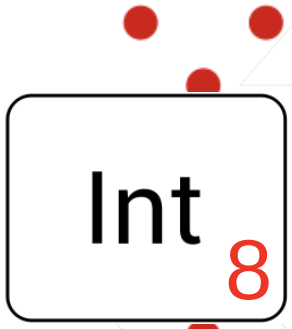
xfDNN Quantizer: Fast and Easy



1) Provide FP32 network and model
   - E.g., prototxt and caffemodel



2) Provide a small sample set, no labels required
   - 16 to 512 images

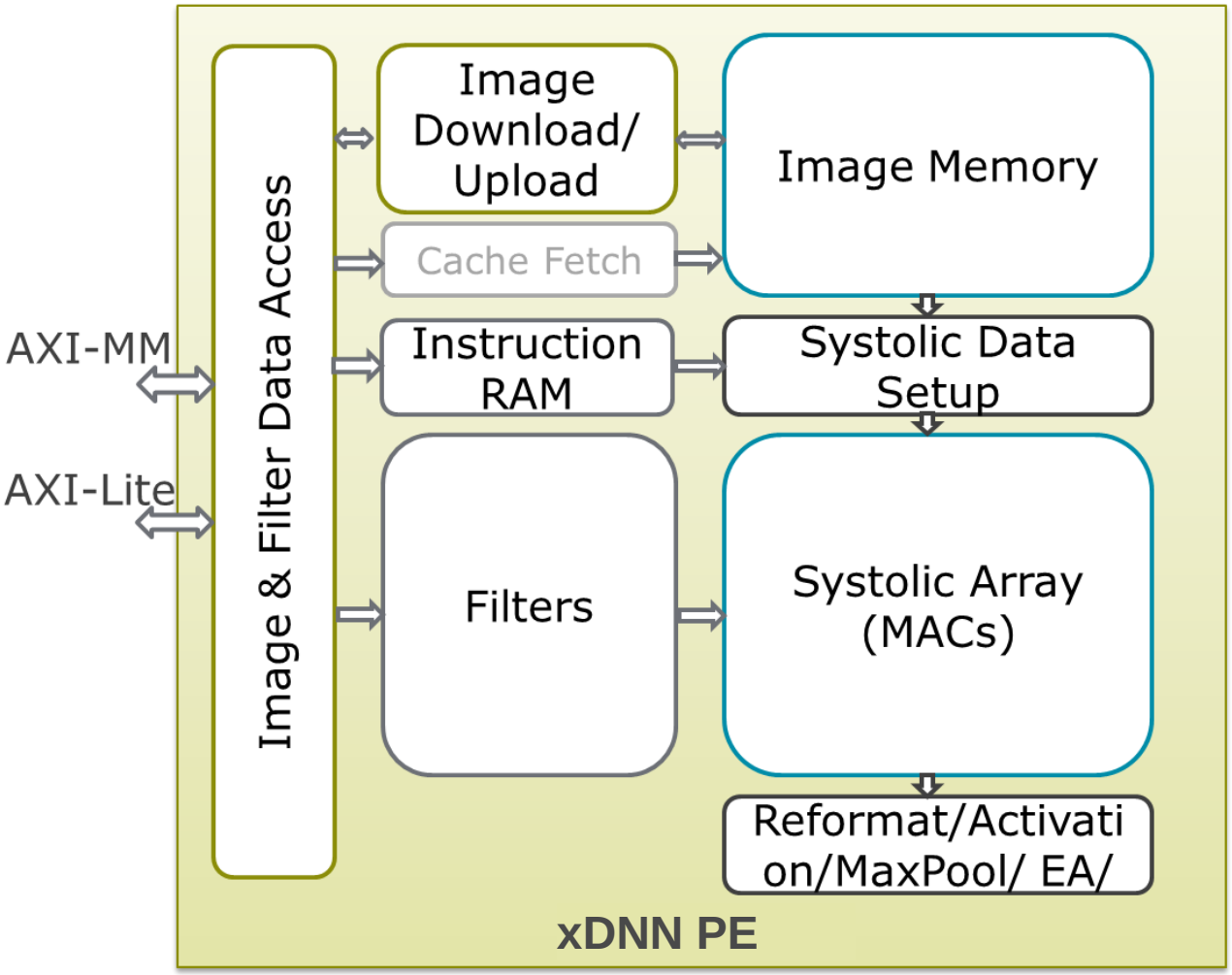


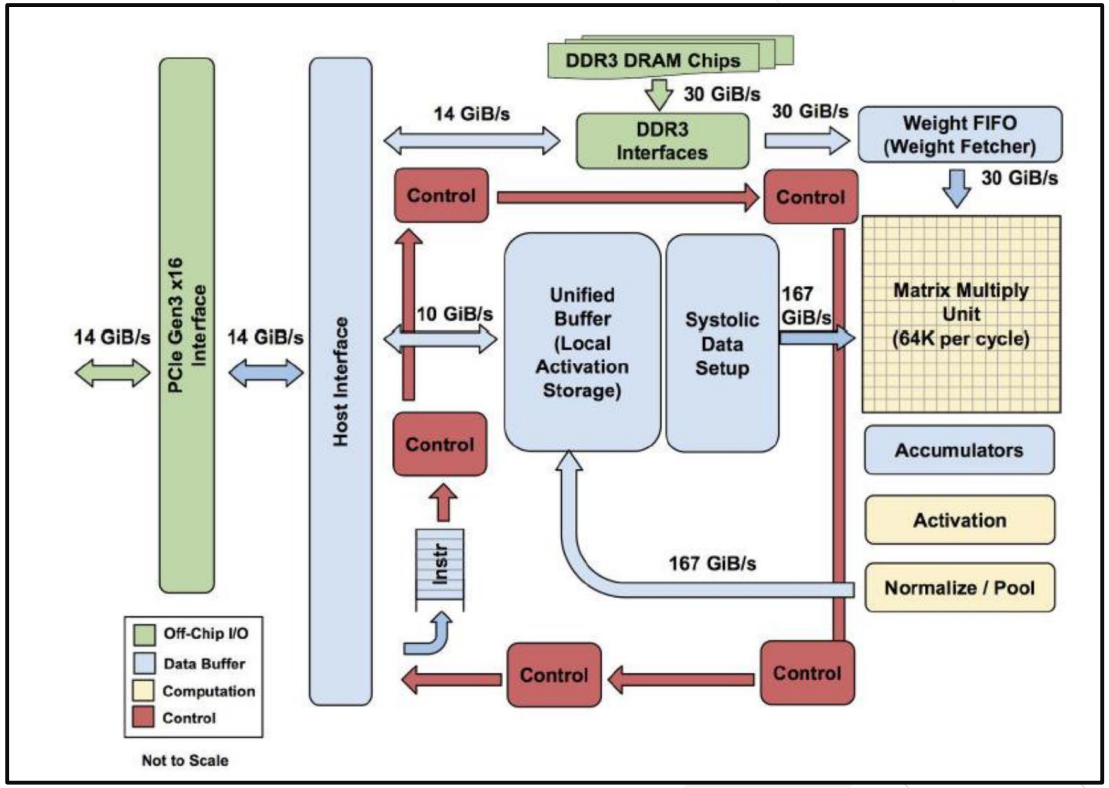
3) Specify desired precision
   - Quantizes to <8 bits to match Xilinx's DSP

ML-suite - HW
xDNN Processing Engine

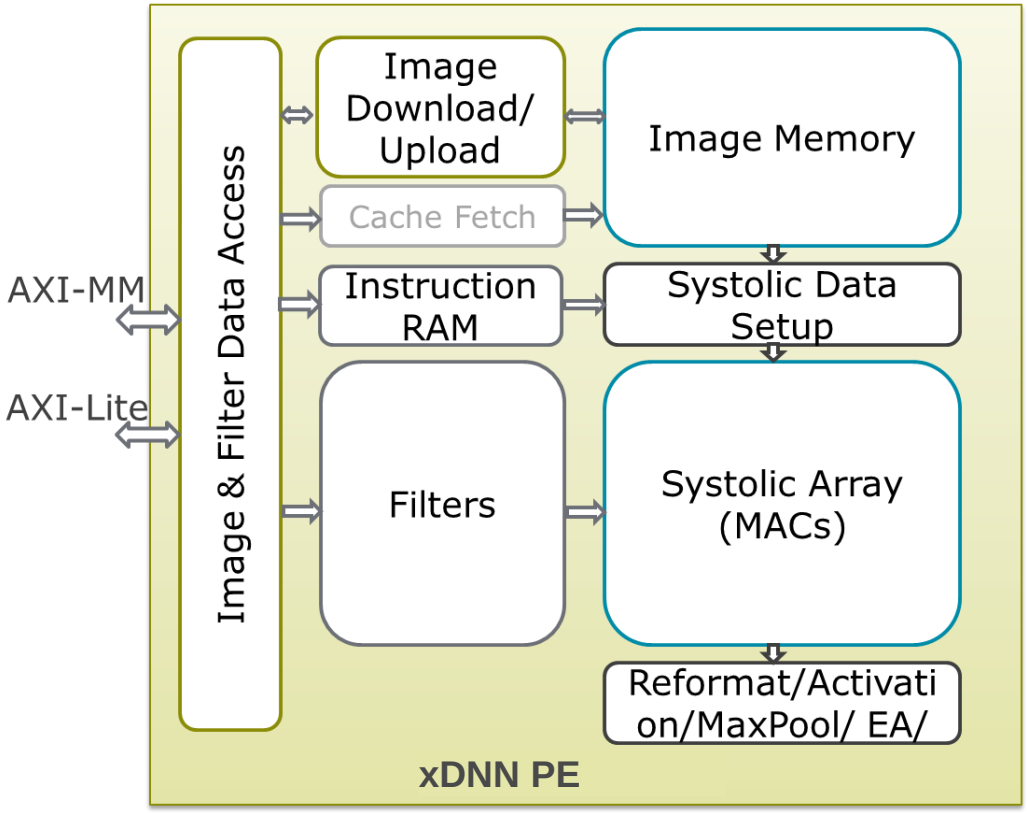
XILINX

Xilinx CNN Processing Engine – xDNN



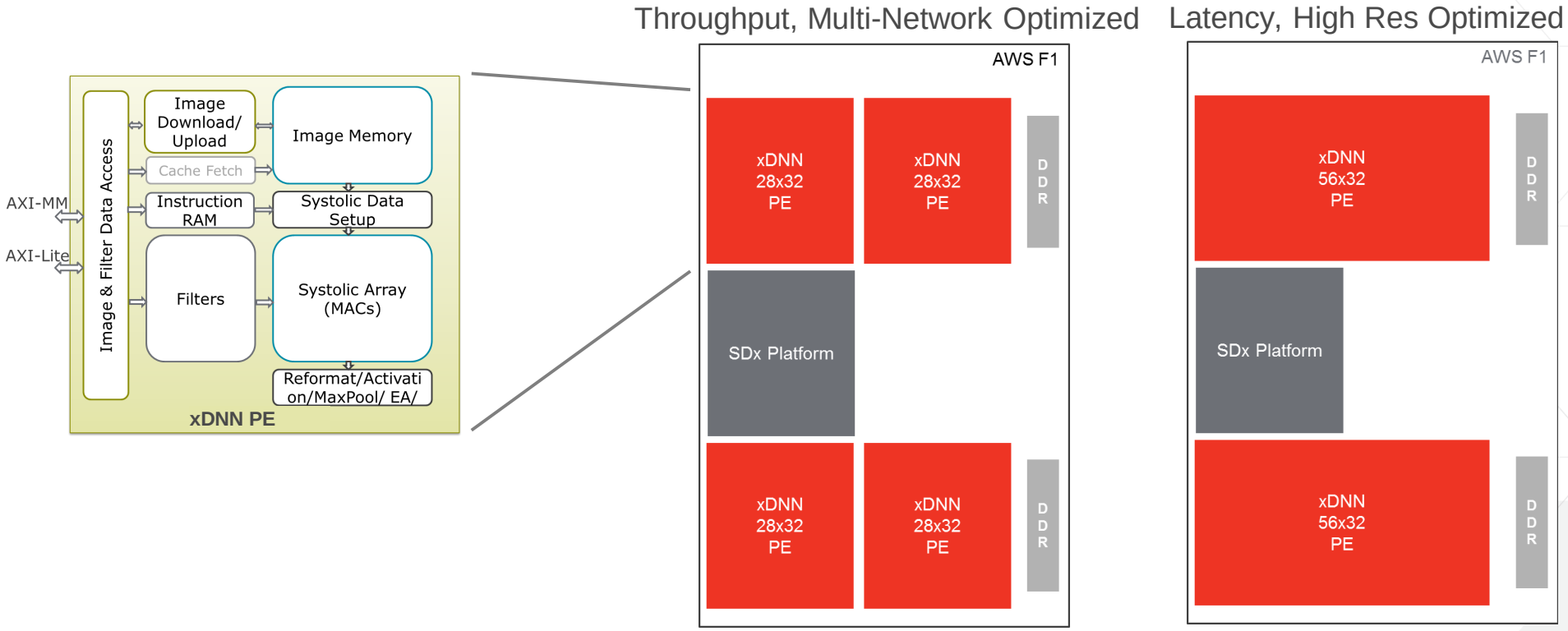


Wide Range of Deep Learning Models Supported v2



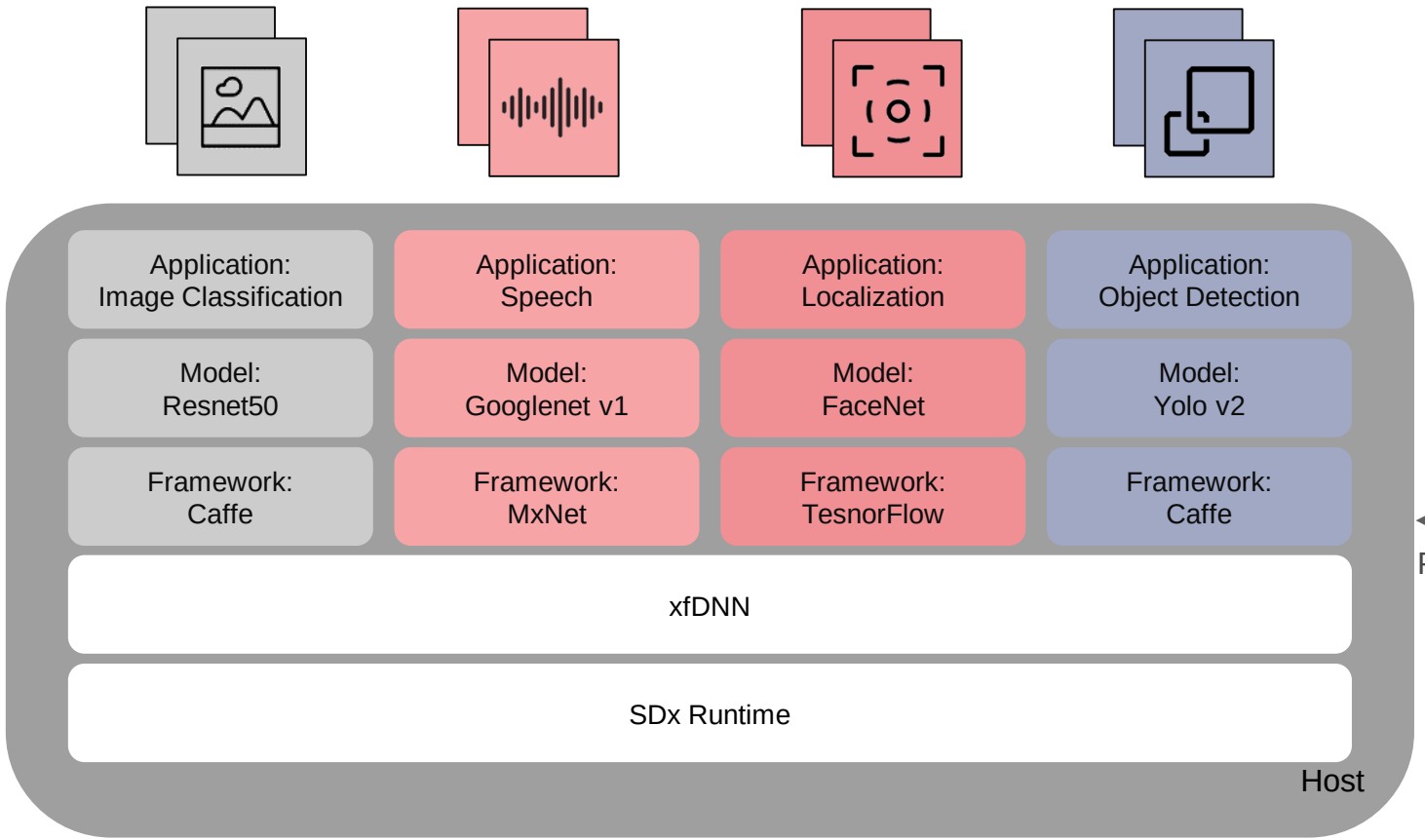
Features		Description	
Supported Operations	Convolution / Deconvolution / Convolution Transpose	Kernel Sizes	W: 1-15; H:1-15
		Strides	W: 1,2,4,8; H: 1,2,4,8
		Padding	Same, Valid
		Dilation	Factor: 1,2,4
		Activation	ReLU
		Bias	Value Per Channel
		Scaling	Scale & Shift Value Per Channel
	Max Pooling	Kernel Sizes	W: 1-15; H:1-15
		Strides	W: 1,2,4,8; H: 1,2,4,8
		Padding	Same, Valid
	Avg Pooling	Kernel Sizes	W: 1-15; H:1-15
		Strides	W: 1,2,4,8; H: 1,2,4,8
		Padding	Same, Valid
	Element-wise Add	Width & Height must match; Depth can mismatch.	
	Memory Support	On-Chip Buffering, DDR Caching	
	Expanded set of image sizes	Square, Rectangular	
	Upsampling	Strides	Factor: 2,4,8,16
Miscellaneous	Data width	16-bit or 8-bit	

xDNN PEs Optimized for Your Cloud Applications



PE	#DSPs	Cache	16 bit GOP/s	8 bit GOP/s	Googlenet V1 latency	Optimized For	Examples Networks
28x32	896	5 MB	896	1,792	5.51	Multi-Network, Maximum Throughput	ResNet50 (224x224)
56x32	1792	5 MB	1,792	3,584	3.23	Lowest Latency Streaming	Yolov2 (224x224)
56x32	1792	8 MB	1,792	3,584	3.23	Lowest Latency, High Resolution	Yolov2 (605x605), ResNet50 (512x512)

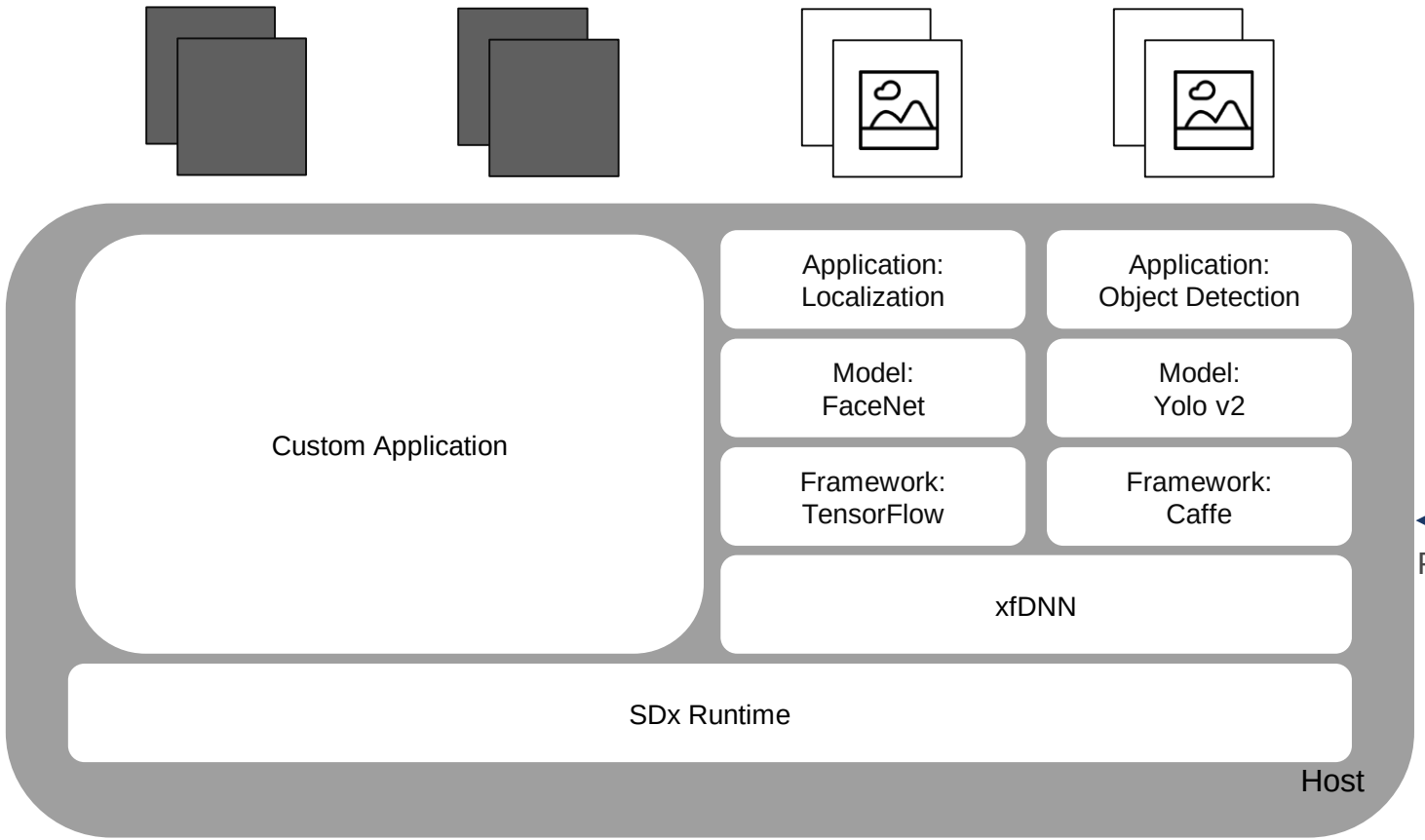
Flexible: Multi-Network Configuration



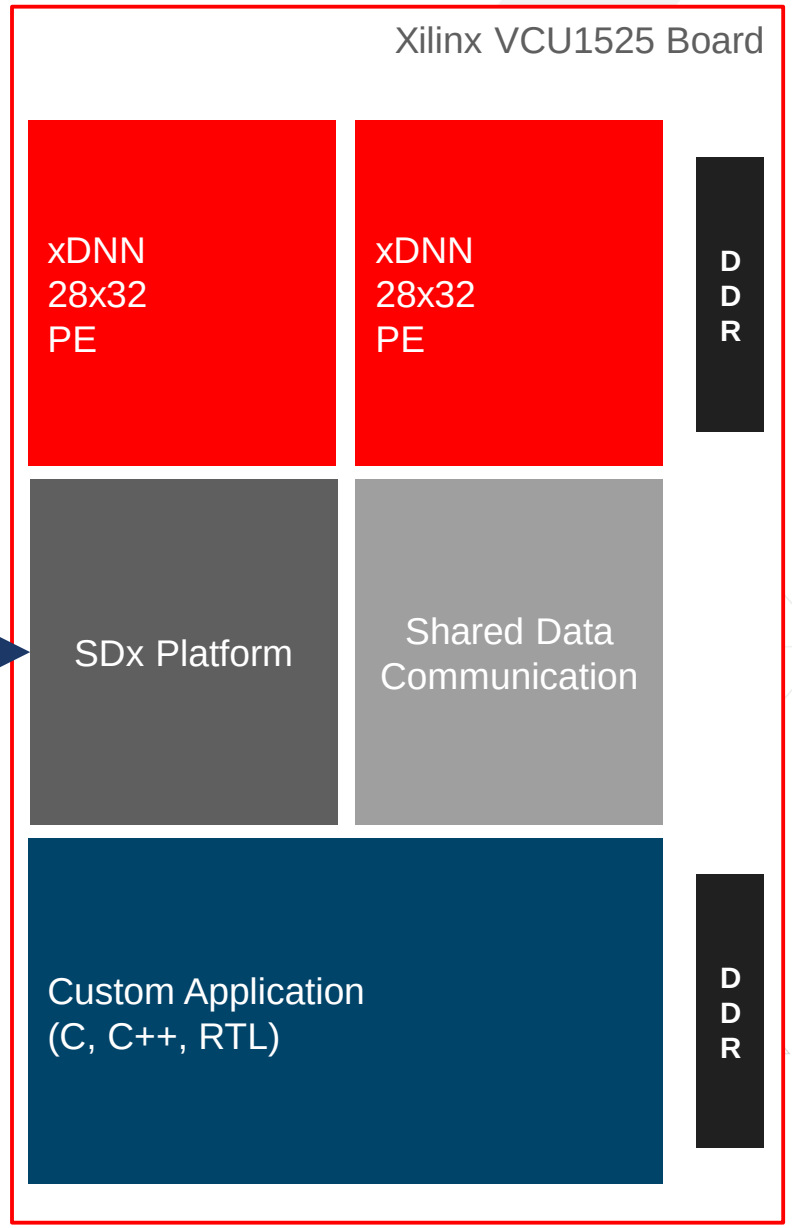
1 FPGA Provides 4 Virtual Accelerators
For Real Time Deep Learning

XILINX

Flexible: Bring Your own IP!

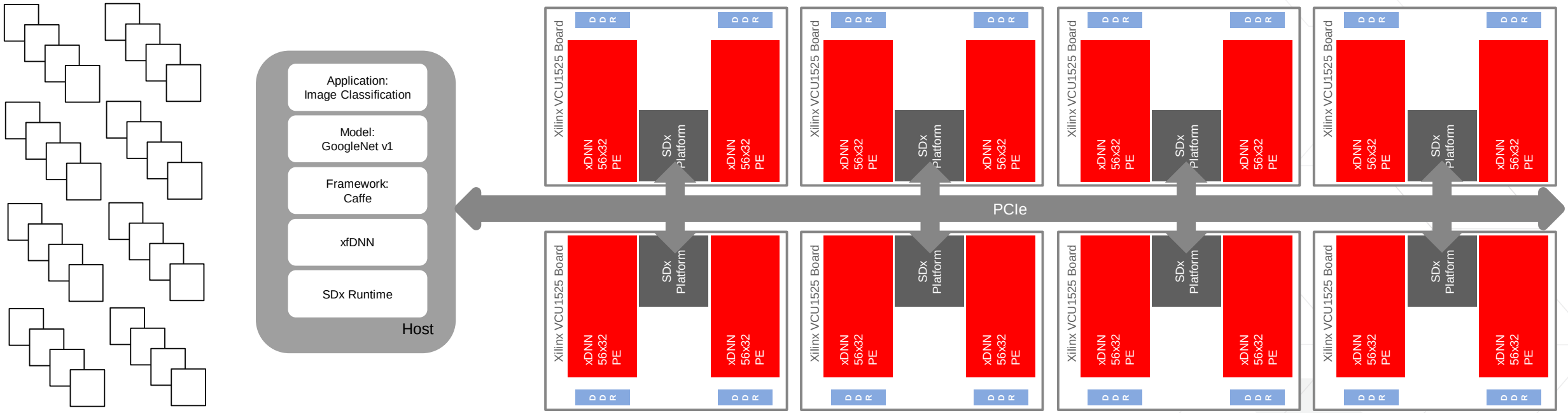


PCIe



Integrate Custom Applications Directly
with xDNN Processing Engines

Scalable: Pooled Acceleration



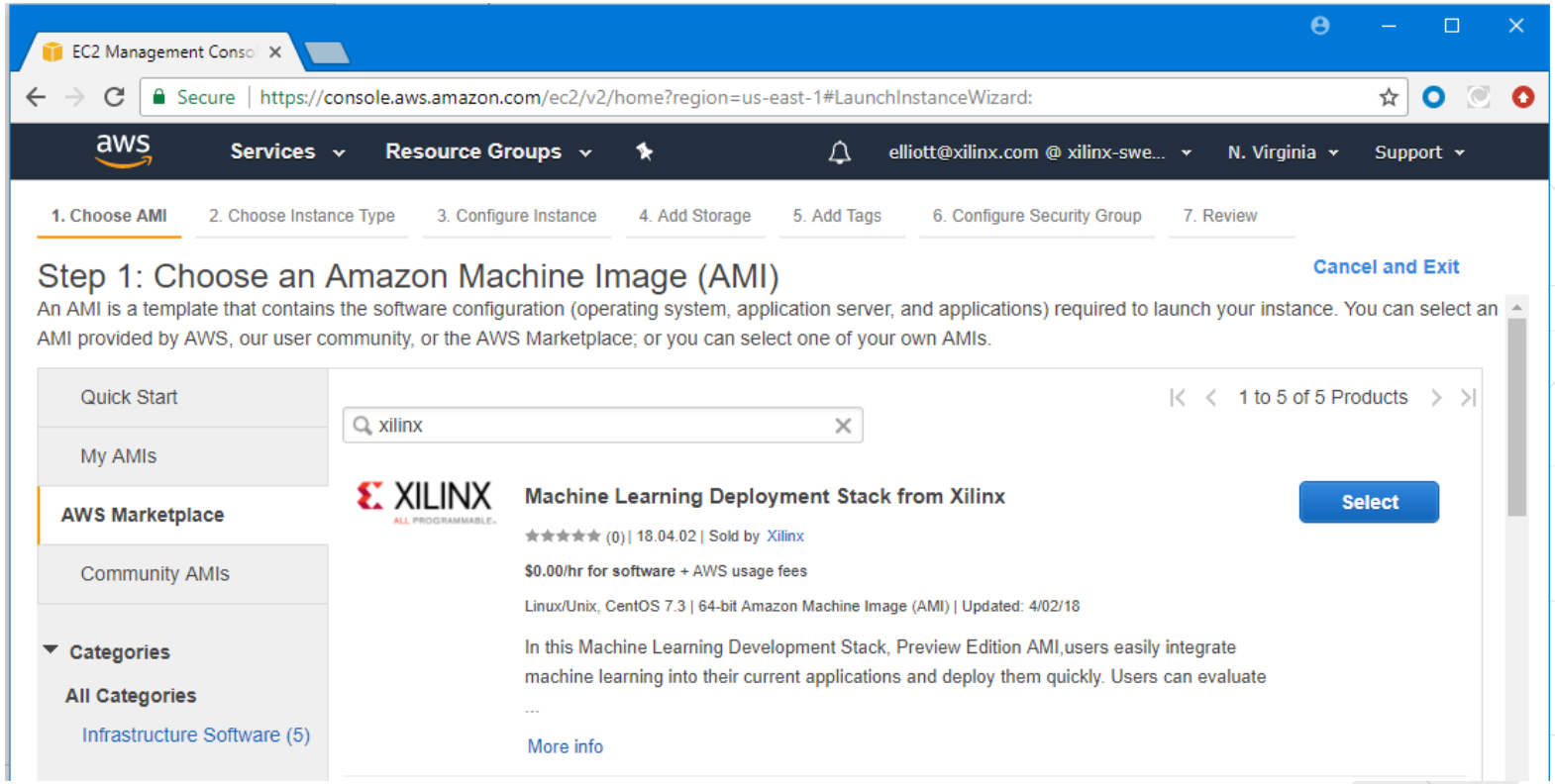
- Software based pooling provides 7.2x performance over single FPGA for Image Classification

- Pooled FPGAs on AWS F1 instance f1.16xlarge
  - >10000 Images/sec for GoogLeNet-V1@int8

Getting Started Today

Xilinx's ML Suite - Amazon EC2 Marketplace

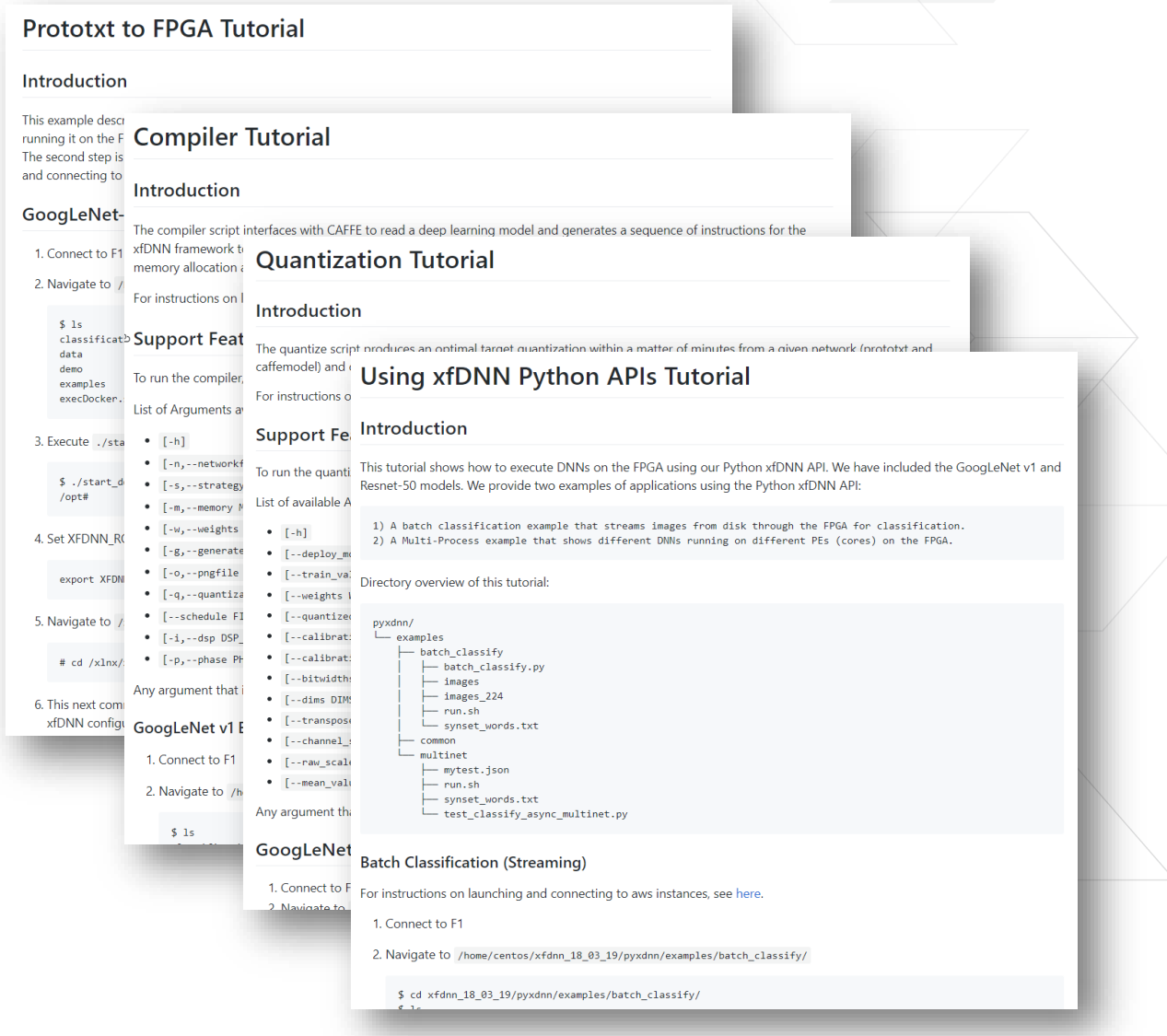
- **ML Suite v18.04.02**
  - Supported Frameworks:
    - Caffe
    - MxNet
    - Tensorflow (Beta)
    - Python Support
    - Darknet
  - Pre-trained Models
    - Caffe 8/16-bit
      - GoogLeNet v1
      - ResNet50
      - Flowers102
      - Places365
    - Python 8/16-bit
      - Yolov2
    - MxNet 8/16-bit
      - GoogLeNet v1
  - xfDNN Tools
    - Compiler
    - Quantizer



https://aws.amazon.com/marketplace/pp/B077FM2JNS

Getting Started with Xilinx ML Suite

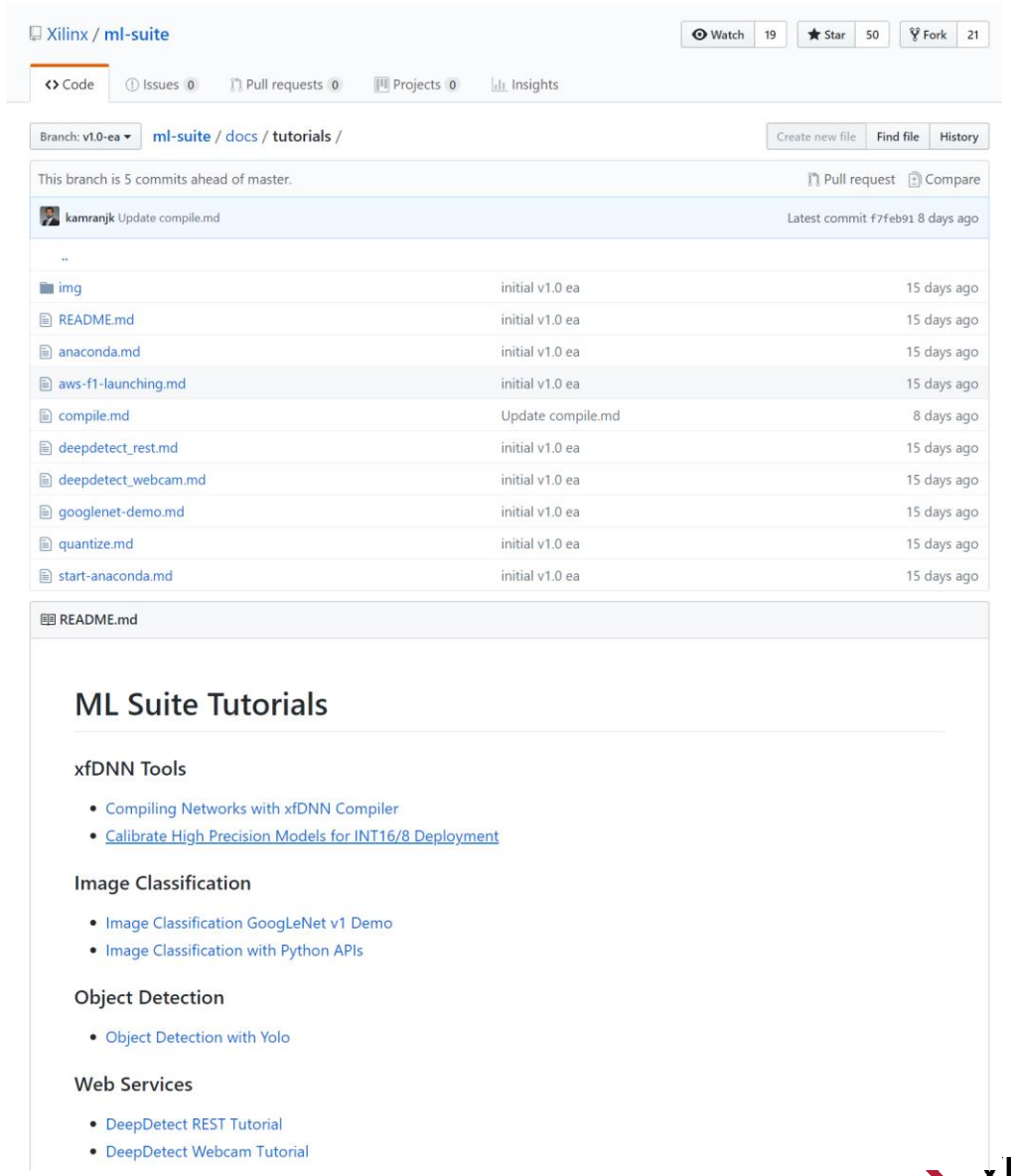
- https://github.com/Xilinx/ML-Suite
- **Tutorials:**
  - Python
    - Batch and Multi-Process
    - Yolo
  - Max Throughput Demo
    - Single FPGA: Image Classification GoogLeNet-v1 Demo
    - Pooled FPGAs: Image Classification GoogLeNet-v1 Demo
  - Classification through RESTful APIs
    - DeepDetect REST
    - DeepDetect Webcam
  - Frameworks
    - Caffe
    - MxNet
    - Tensorflow (Beta)
  - xfDNN Tools
    - Quantization
    - Compiler
    - Prototxt to FPGA



XILINX

Getting Started with Xilinx ML Suite

- https://github.com/Xilinx/ml-suite/tree/v1.0-ea Tutorials:
  - xfDNN Tools
    - Quantization
    - Compiler
  - Image Classification
    - Image Classification GoogLeNet-v1 Demo
    - Image Classification with Python APIs
  - Object Detection
    - Object Detection with Yolo
  - Web Service
    - DeepDetect REST
    - DeepDetect Webcam
- Available Pre-Trained models
  - Caffe
    - MobileNet
    - Places365
    - ResNet50
    - Yolov2
    - Flowers102
    - GoogleNetv1
  - Tensorflow
    - GoogleNetv1
    - FCN
    - Flowers102
    - Place365
    - ResNet50
    - VGG16



Tutorial Walk Through

Using Caffe to Compile your Network

- Compiler for Caffe || Tensor, https://github.com/Xilinx/ml-suite/blob/master/compile.md

1. Connect to F1
2. cd /home/centos/xfdnn_18_03_19/caffe/ || Cd /home/centos/xfdnn_18_03_19/tensorflow/
3. ./start_docker.sh
4. export XFDNN_ROOT=/xlnx
5. cd /xlnx/xfdnn_tools/compile/
6. Caffe

```
python tests/xfdnn_compiler.pyc -n /xlnx/xfdnn_tools/models/caffe/bvlc_googlenet_quantized/GoogleNetWithOutLRN_dummydata_deploy.prototxt \
    -s all \
    -m 4 \
    -i 28 \
    -g network.cmd
```

7. Tensor

```
python tests/xfdnn_compiler_tensorflow.pyc \
--input_graph=../models/tensorflow/bvlc_googlenet/fp32/bvlc_googlenet.pb \
--name=googlenetv1 \
--outputnames='pool5_7x7_s1' \
--weight=googlenetv1_data \
```

Using Caffe to Quantize your Network

- Quantizer for Caffe || Tensor, https://github.com/Xilinx/ml-suite/blob/master/quantize.md

1. Connect to F1
2. cd /home/centos/xfdnn_18_03_19/caffe/ || Cd /home/centos/xfdnn_18_03_19/tensorflow/
3. ./start_docker.sh
4. export XFDNN_ROOT=/xlnx
5. cd /xlnx/xfdnn_tools/quantize/
6. Caffe

```
python quantize.pyc \
--deploy_model /xlnx/models/bvlc_googlenet_without_lrn/fp32/bvlc_googlenet_without_lrn_deploy.prototxt
--train_val_model /xlnx/models/bvlc_googlenet_without_lrn/fp32/bvlc_googlenet_without_lrn_train_val.prototxt
--weights /xlnx/models/bvlc_googlenet_without_lrn/fp32/bvlc_googlenet_without_lrn.caffemodel
--quantized_train_val_model q.train_val.prototxt
--calibration_directory ../../imagenet_val/
--calibration_size 8
--bitwidths 8,8,8
--dims 3,224,224
--transpose 2,0,1
--channel_swap 2,1,0
--raw_scale 255.0
--mean_value 104,117,123
--input_scale 1.0
```

7. Tensor(will be updated in v1.0-ea branch soon)

Python Examples

326 lines (283 sloc) | 11.6 KB — Raw | Blame | History

Using xfDNN Python APIs Tutorial

Introduction

This tutorial shows how to execute DNNs on the FPGA using our Python xfDNN API. We have included the GoogLeNet v1 and Resnet-50 models. We provide two examples of applications using the Python xfDNN API:

```
1) A batch classification example that streams images from disk through the FPGA for classification.
2) A Multi-Process example that shows different DNNs running on different PEs (cores) on the FPGA.
```

Directory overview of this tutorial:

```
pyxdnn/
└── examples
    ├── batch_classify
    │   ├── batch_classify.py
    │   ├── images
    │   ├── images_224
    │   ├── run.sh
    │   └── synset_words.txt
    ├── common
    └── multinet
        ├── mytest.json
        ├── run.sh
        ├── synset_words.txt
        └── test_classify_async_multinet.py
```

Batch Classification (Streaming)

For instructions on launching and connecting to aws instances, see here.

1. Connect to F1
2. Navigate to `/home/centos/xfdnn_18_03_19/pyxdnn/examples/batch_classify/`

```
$ cd xfdnn_18_03_19/pyxdnn/examples/batch_classify/
$ ls
batch_classify.py  images  images_224  run.sh  synset_words.txt
```

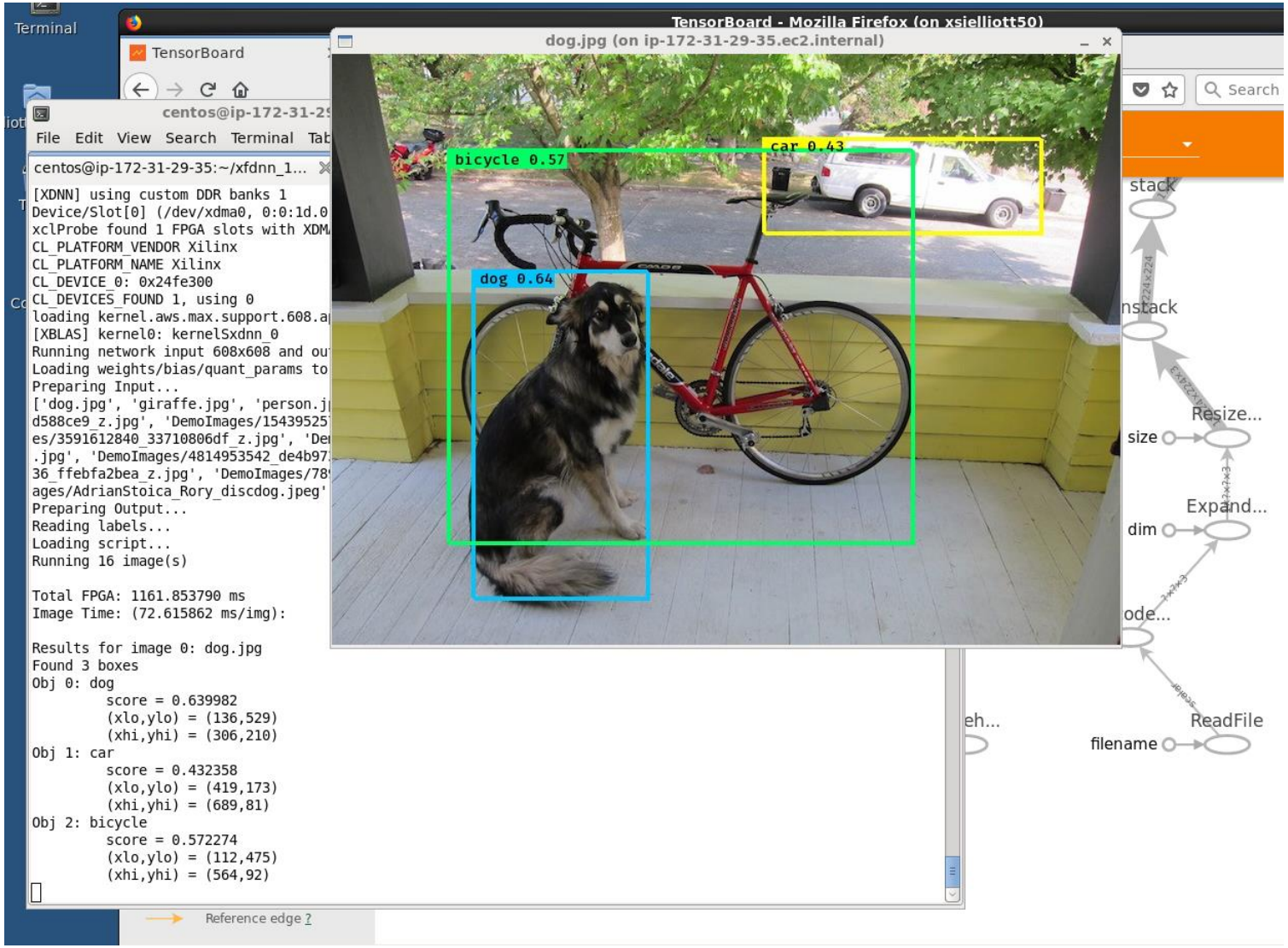
Streaming Images

1. Navigate to *pyxdnn/examples*
2. Set Up batch_classify.py with Network/Model
3. Run Python Script *python batch_classify.py*

Multi-Network

1. Navigate to *pyxdnn/examples*
2. Assign Networks/Models in *mytest.json*
3. Run Python Script *python test_classify_async_multinet.py*

Python Object Detection w/YOLOv2



Adaptable.
Intelligent.

XILINX