

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

정 웅

DSP Specialist / 부장

2018-07-03



# Agenda

## > Xilinx & Deep learning

## > ML-suite for CNN

- >> Processing Engine
- >> Middle Ware
- >> Getting Start Today

## > Tutorial Walk Through

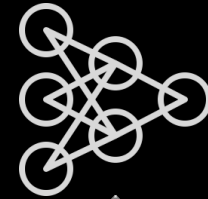
- >> Deploying(Compiling and Quantizing) with Caffe
- >> Python APIs



# Xilinx & Deep Learning



Deep Learning explores the study of algorithms that can learn from and make predictions on data



Deep Learning is Re-Defining many Applications



Cloud Acceleration



Security



Ecommerce Social



Financial



Surveillance



Industrial IOT



Medical Bioinformatics



Autonomous Vehicles

# Deep Learning Challenges – Solution Flexibility is Key

**Challenge 1:** Different use cases require different networks & different figures of merits

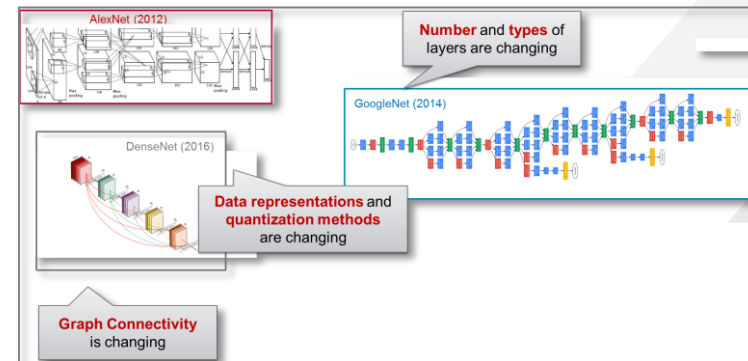
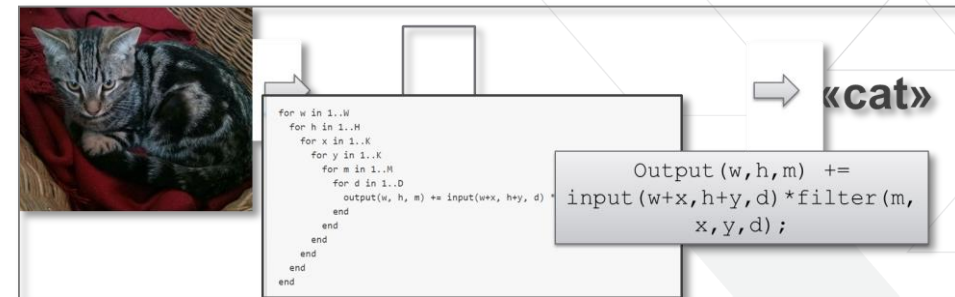
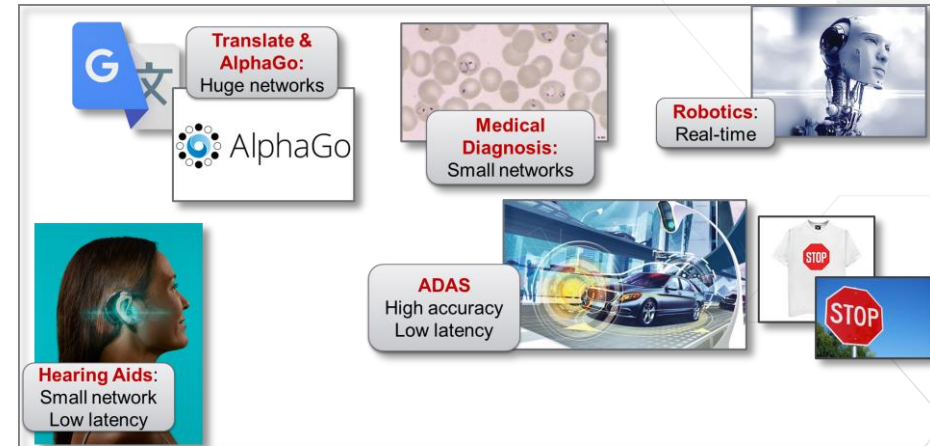
Speed, latency, energy, accuracy

**Challenge 2:** billions of multiply-accumulate ops & tens of megabytes of parameter data

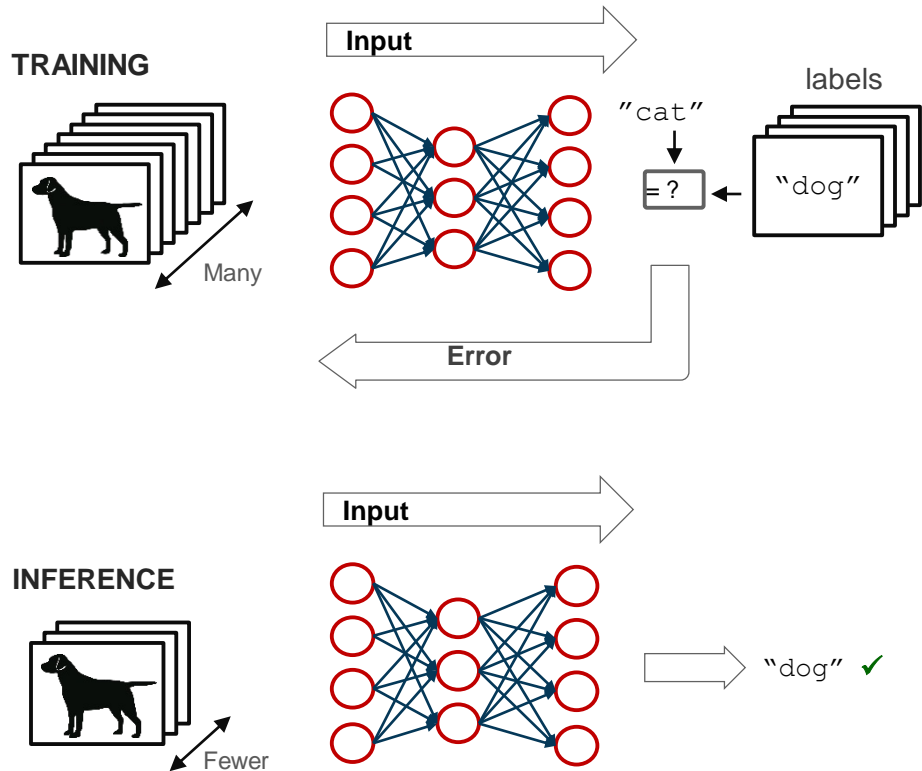
Sea of operators, custom math & memory hierarchy

**Challenge 3:** Continuous stream of new algorithms

Hardening architectures is risky



# The Divergence of Training and Inference in Deep Learning



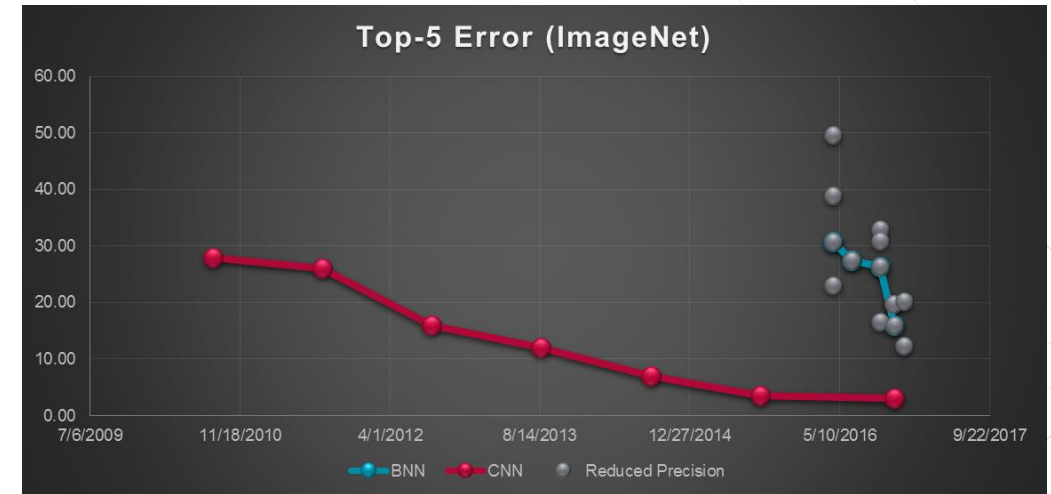
**Training:** Process for machine to “learn” and optimize model from data

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

# Latest Inference Algorithmic Research

## Optimize Compute with Reduced Precision CNNs & BNNs

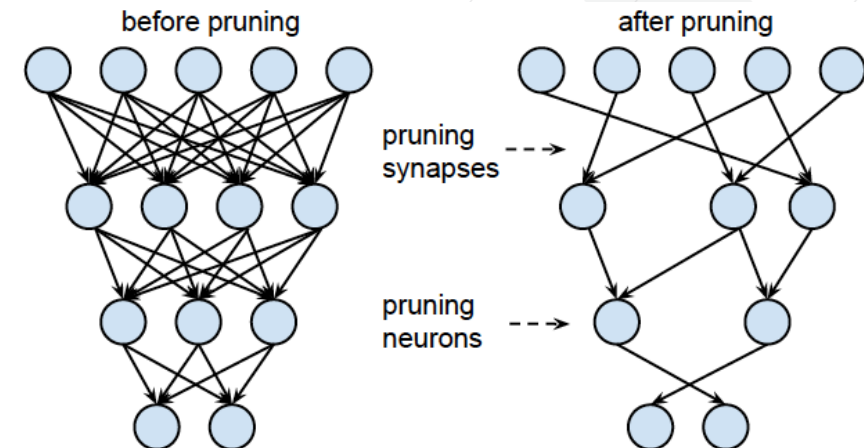
- 8 bit solution loses no significant accuracy
- BNNs are improving rapidly
- Custom Floating point provide significantly more compute density



Not only weights, but also sparsity pattern encodes information

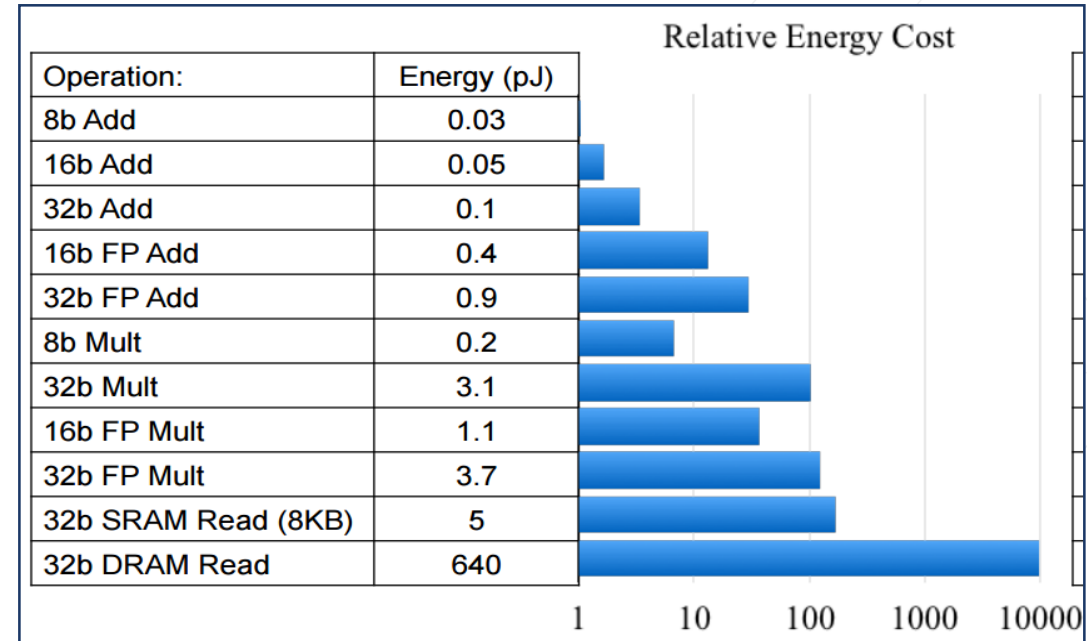
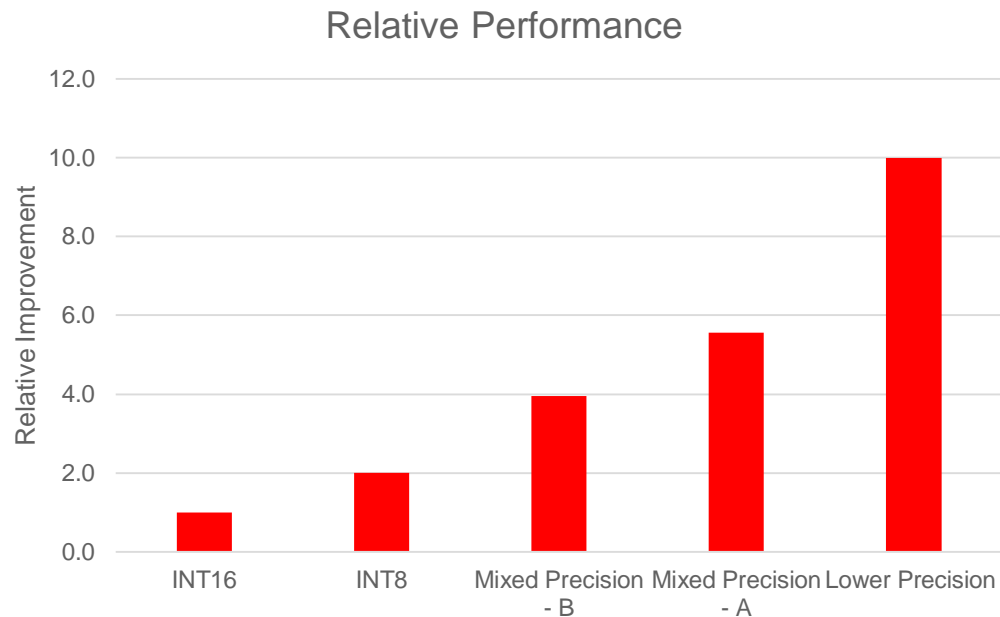
## Model Pruning for higher effective performance: LSTM

- 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



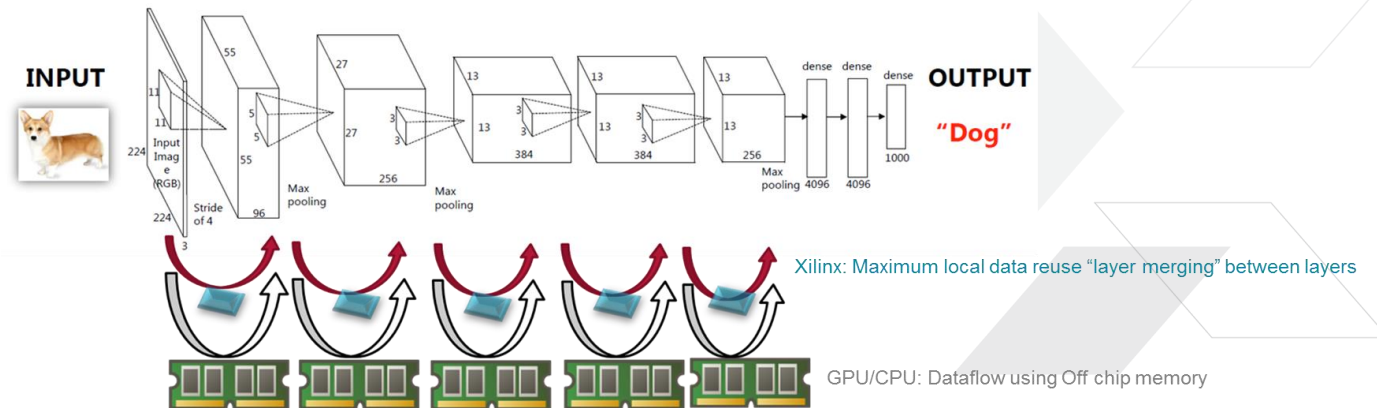
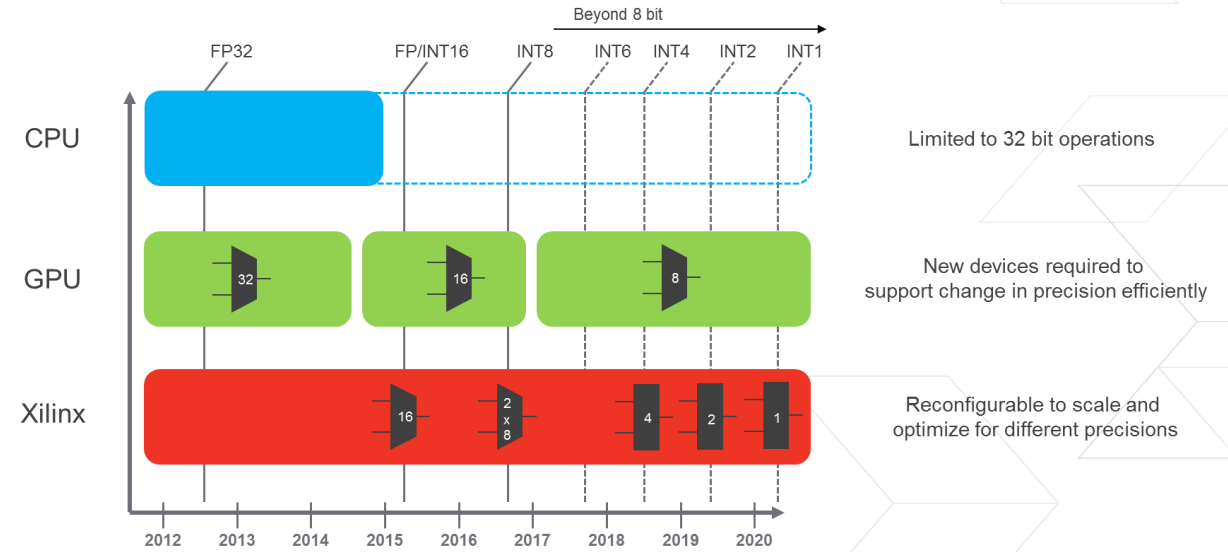
- > Memory cost is greatly reduced
  - >> Large networks can fit entirely into on-chip memory
  - >> (OCM) (UltraRAM, BRAM)

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

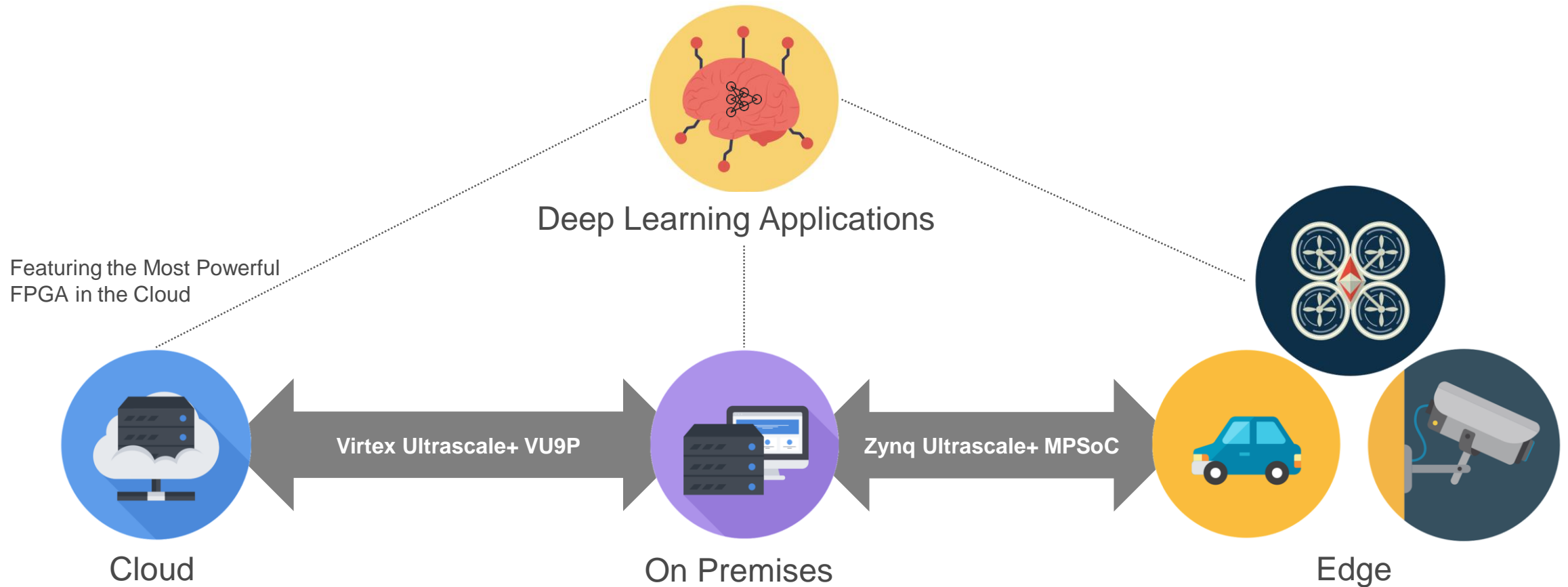
# Xilinx Features for Implementing Efficient Inference Engines

Flexible Architecture for Any Precision

Flexible On-chip Memory for low latency



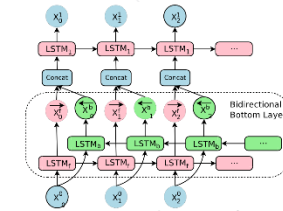
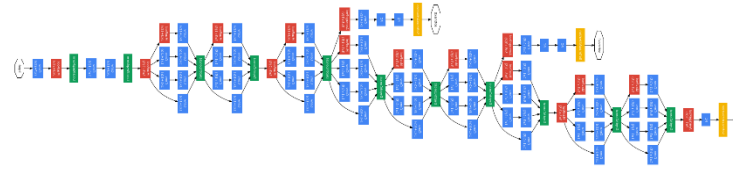
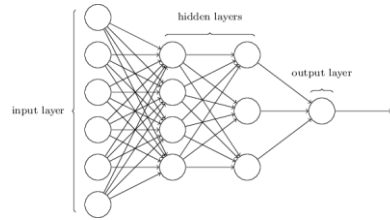
# Accelerating AI Inference into Your Cloud Applications



# Xilinx ML-suite



# Deep Learning Models



## Multi-Layer Perceptron

- Classification
- Universal Function Approximator
- Autoencoder

## Convolutional Neural Network

- Feature Extraction
- Object Detection
- Image Segmentation

## Recurrent Neural Network

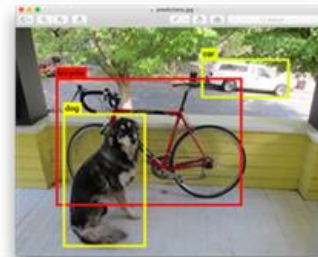
- Sequence and Temporal Data
- Speech to Text
- Language Translation

## Classification

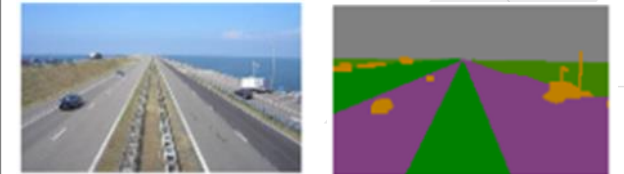


“Dog”

## Object Detection



## Segmentation



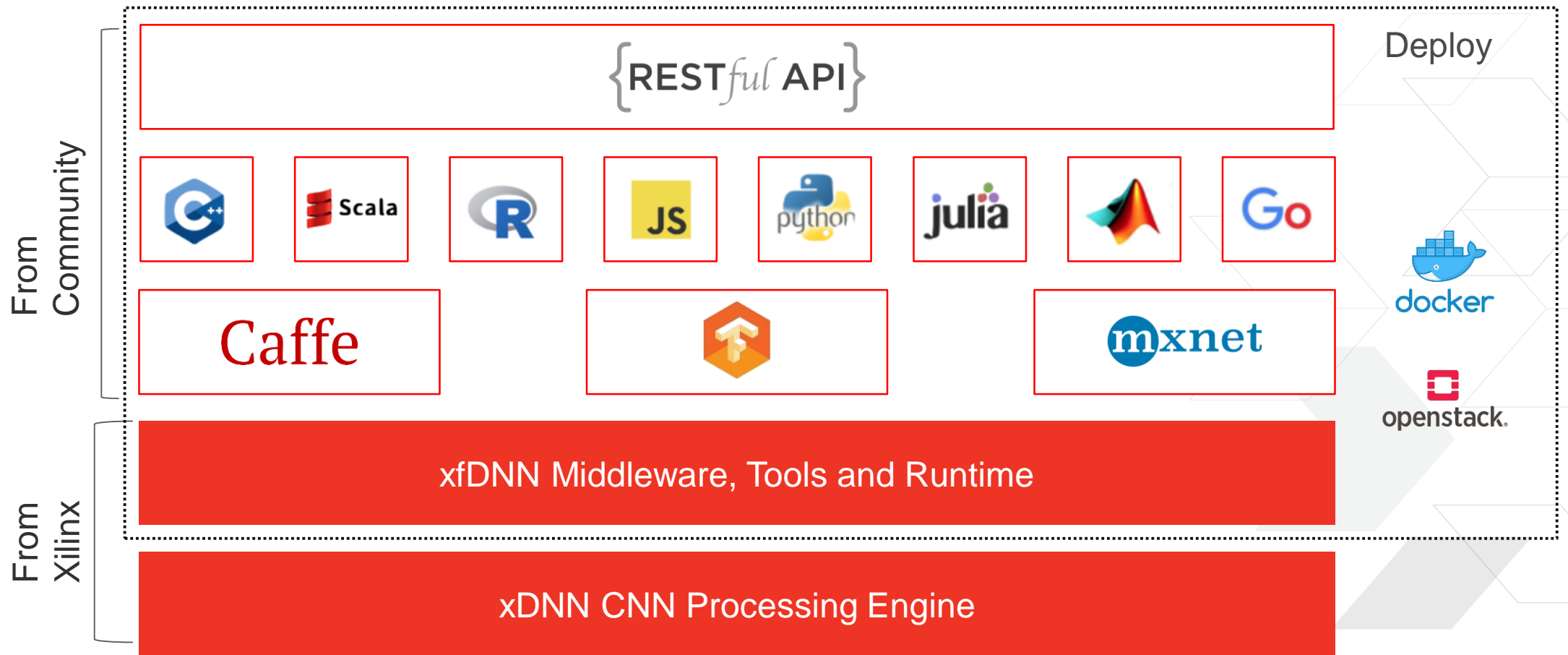
# ML Suite Features Today

- > **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**

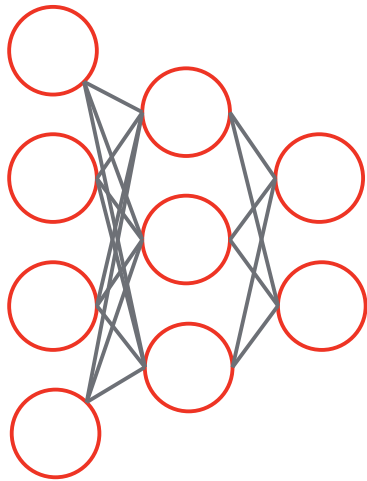
# Seamless Deployment with Open Source Software



\*TensorFlow Q4 2017

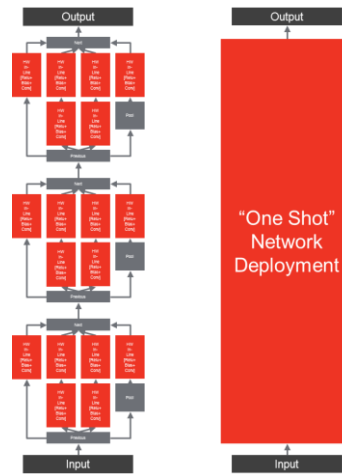
# xfDNN Inference Toolbox

## Graph Compiler



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

## Network Optimization



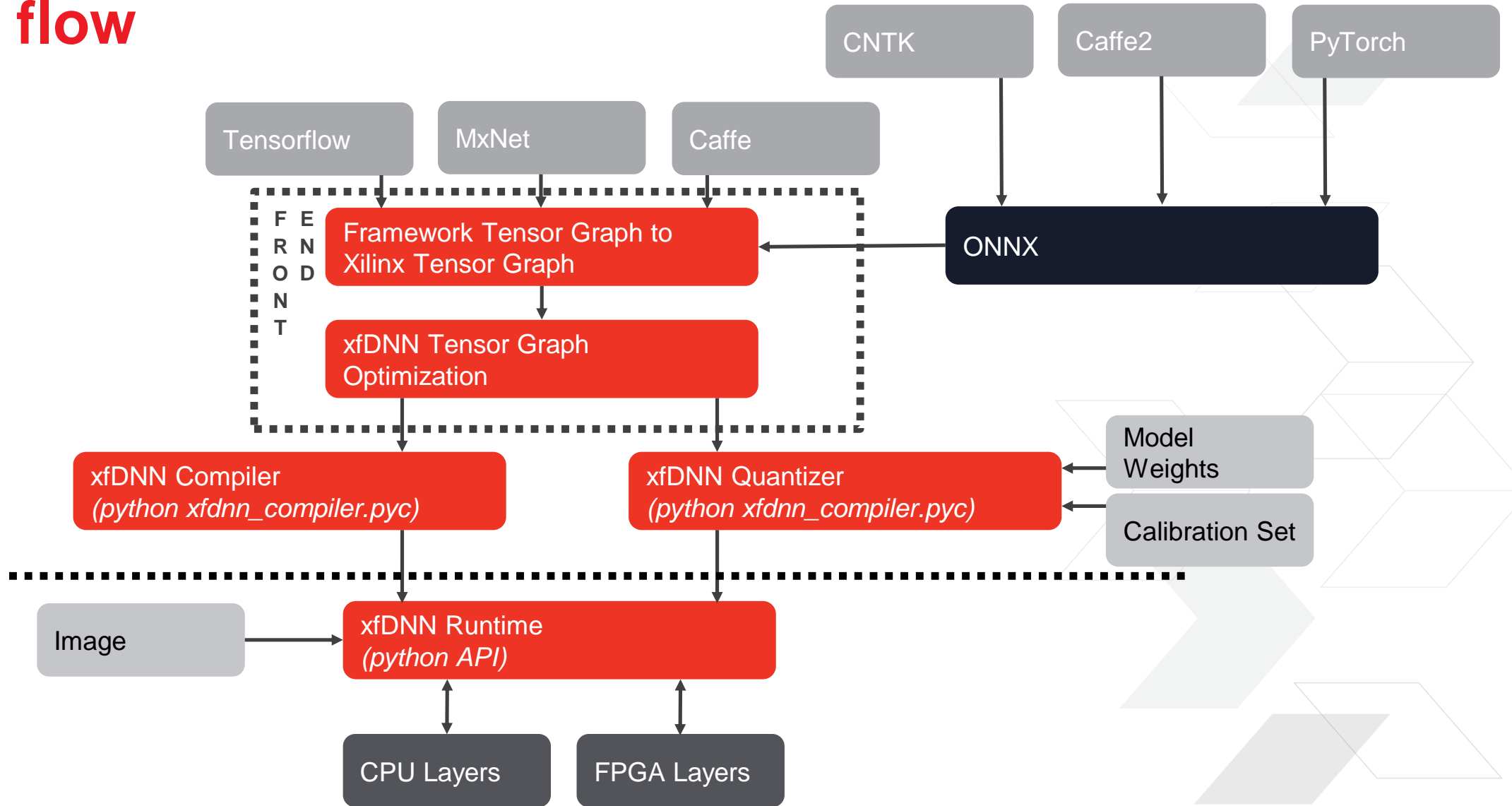
- Fused Layer Optimization
- On-Chip Memory enables Streaming
- “One Shot” Inference eliminates CPU calls

## xfDNN Quantizer



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

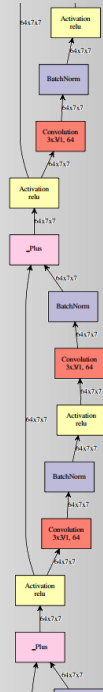
# xfDNN flow



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

# xfDNN Graph Compiler

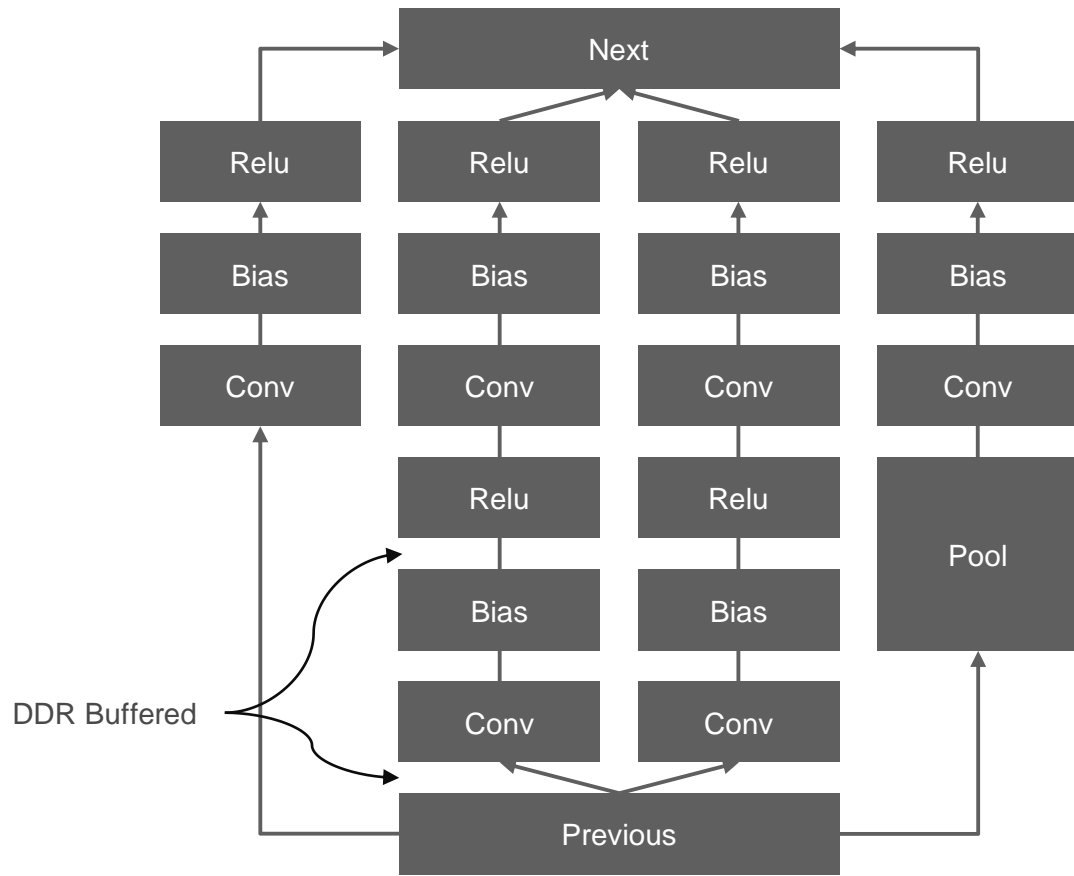
Pass in a Network



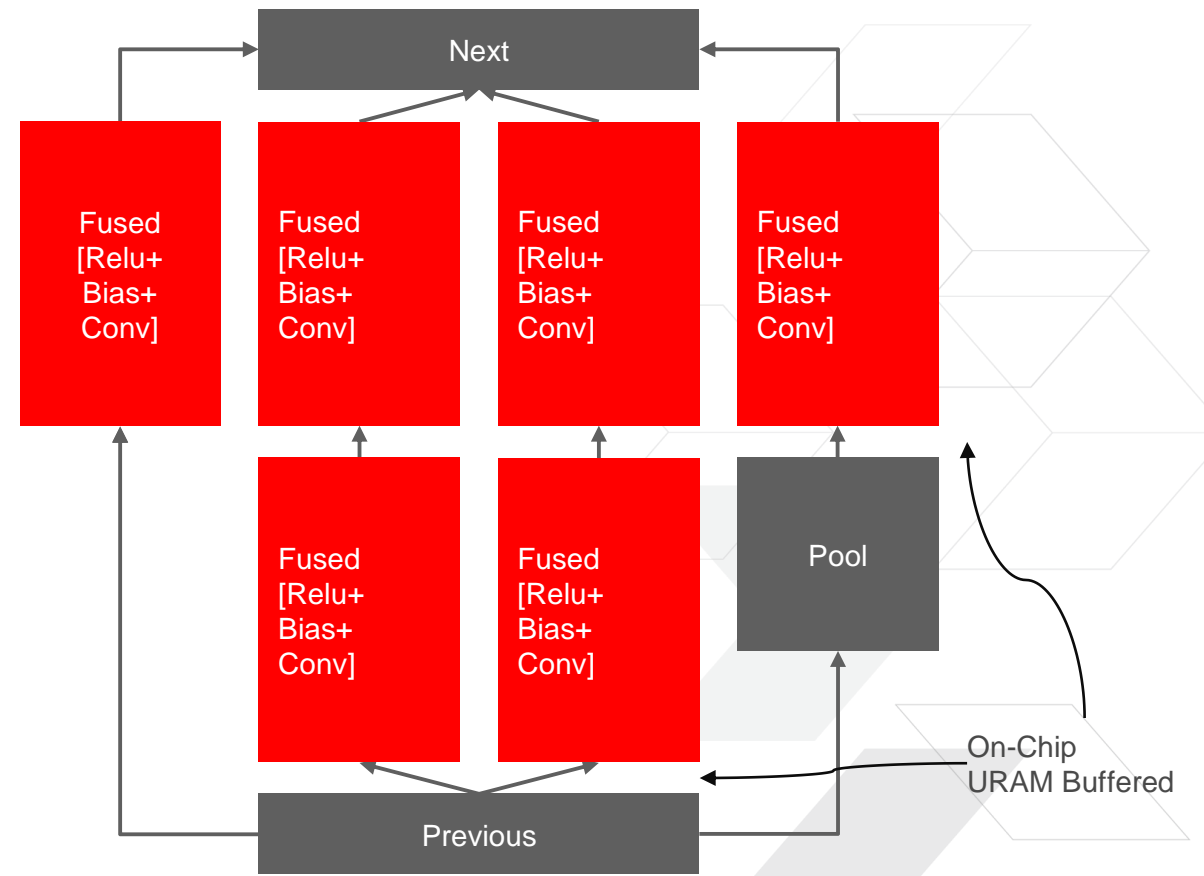
xfDNN  
Graph Compiler

Microcode for xDNN is Produced

# xfDNN Network Optimization Layer to Layer

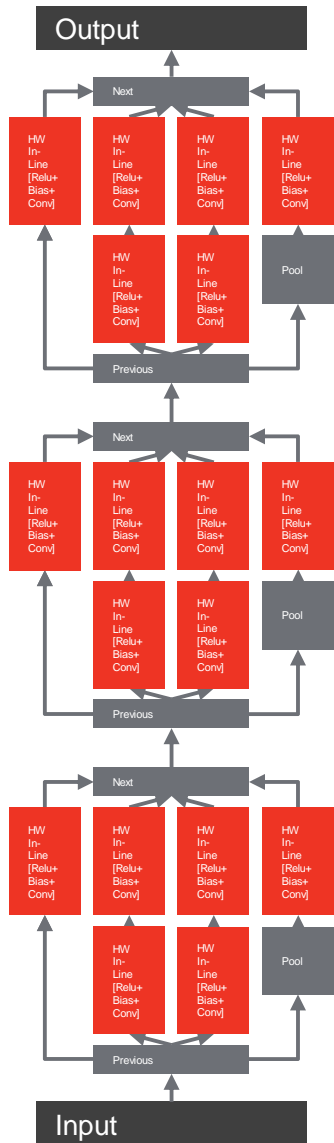


Unoptimized Model



xfDNN Intelligently Fused layers  
Streaming optimized for URAM

# 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

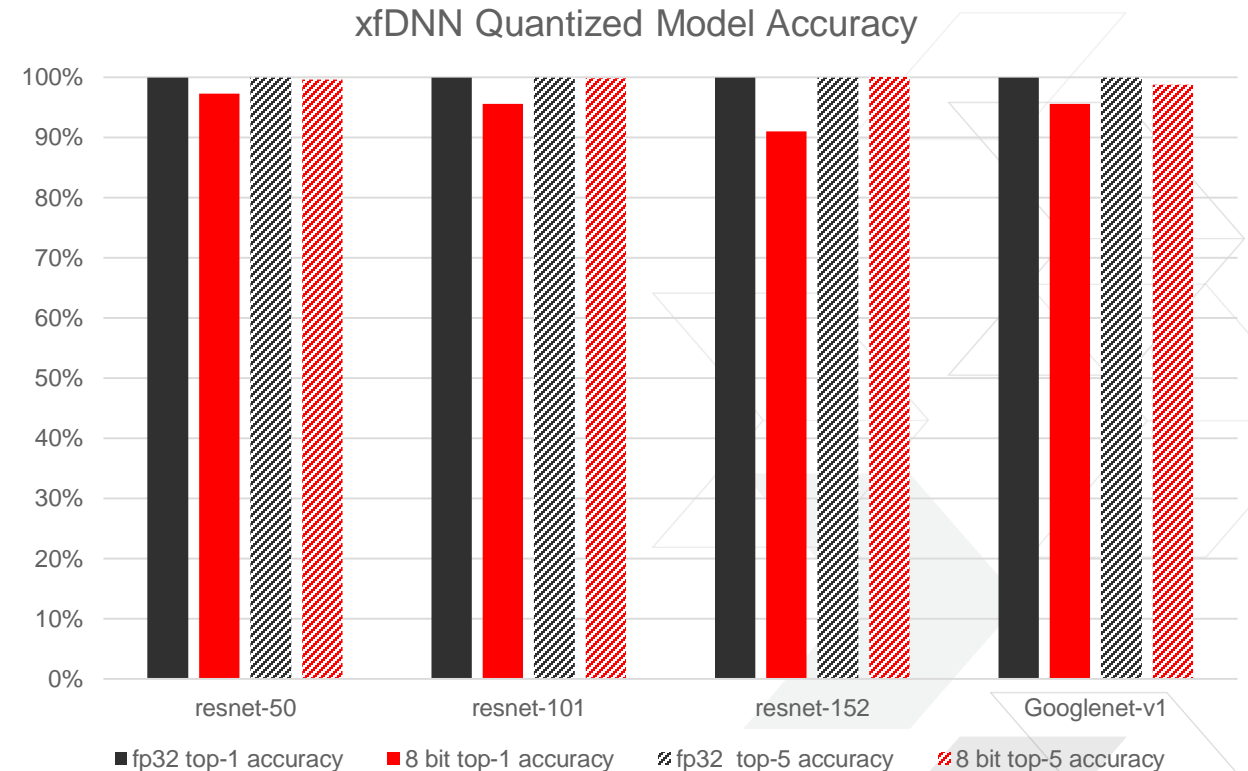
# xfDNN Quantizer: FP to Fixed-Point Quantization

## > Problem:

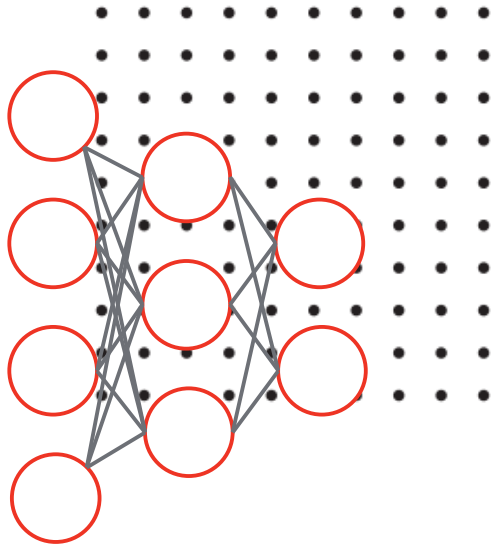
- >> Nearly all trained models are in 32-bit floating-point
- >> Available Caffe and TensorFlow quantization tools take hours and produce inefficient models
- >>

## > Introducing: xfDNN Quantizer

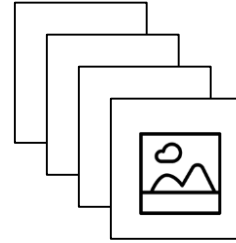
- >> A customer friendly toolkit that automatically analyses floating-point ranges layer-by-layer and produces the fixed-point encoding that loses the least amount of information
  - Quantizes GoogleNet in under a minute
  - Quantizes 8-bit fixed-point networks within 1-3% accuracy of 32-bit floating-point networks
  - Extensible toolkit to maximize performance by searching for minimal viable bitwidths and prune sparse networks



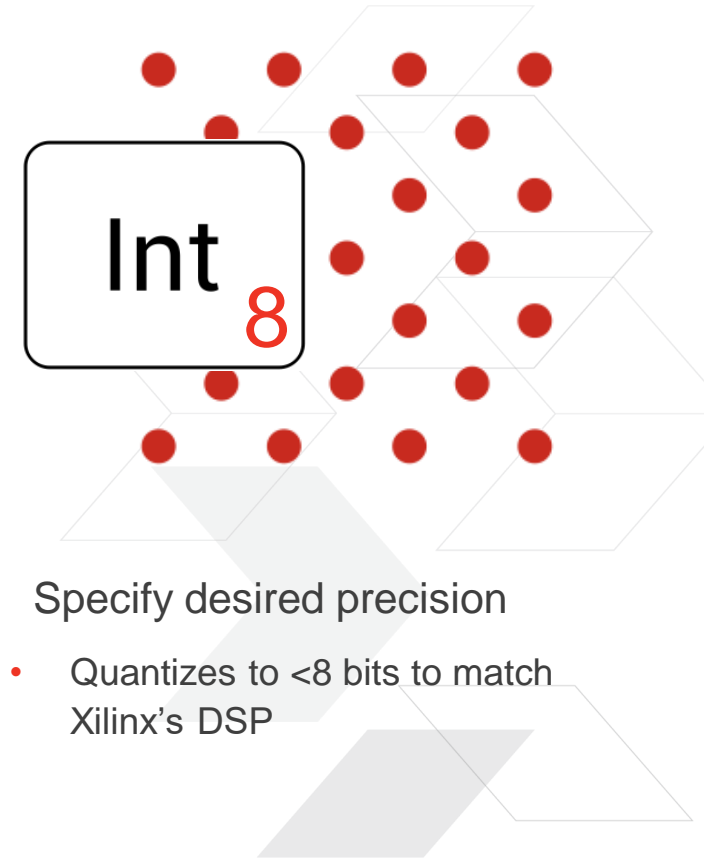
# xfDNN Quantizer: Fast and Easy



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



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

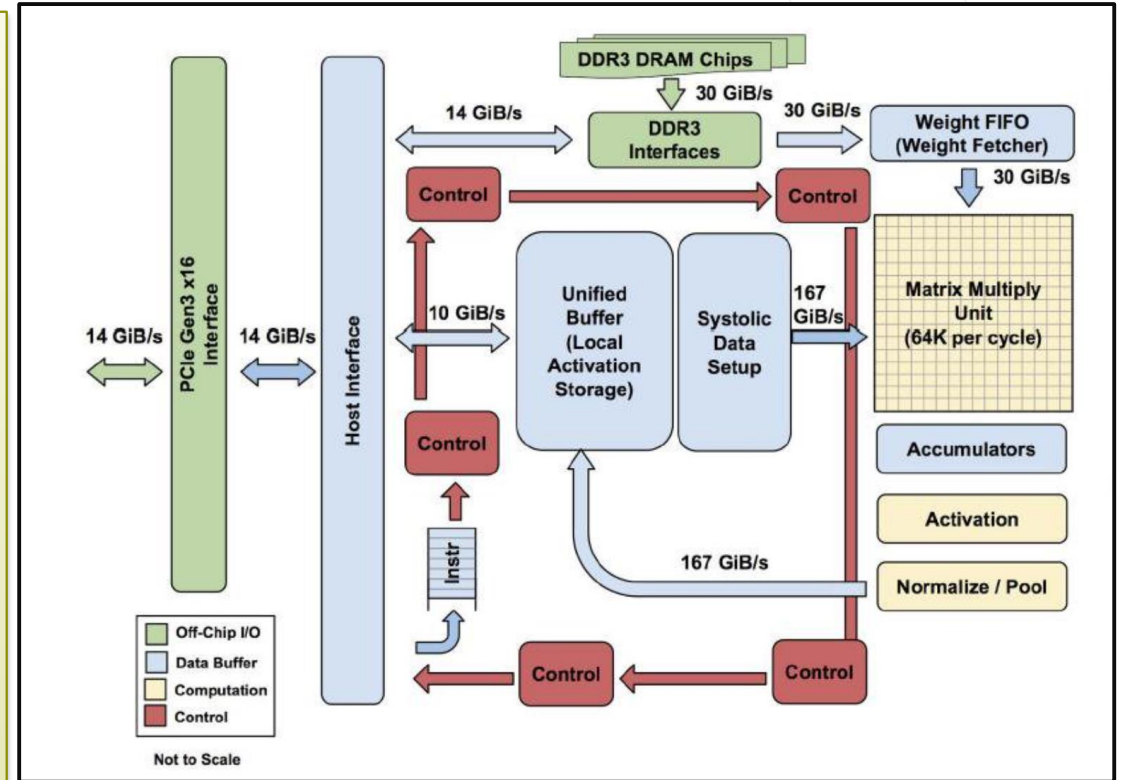
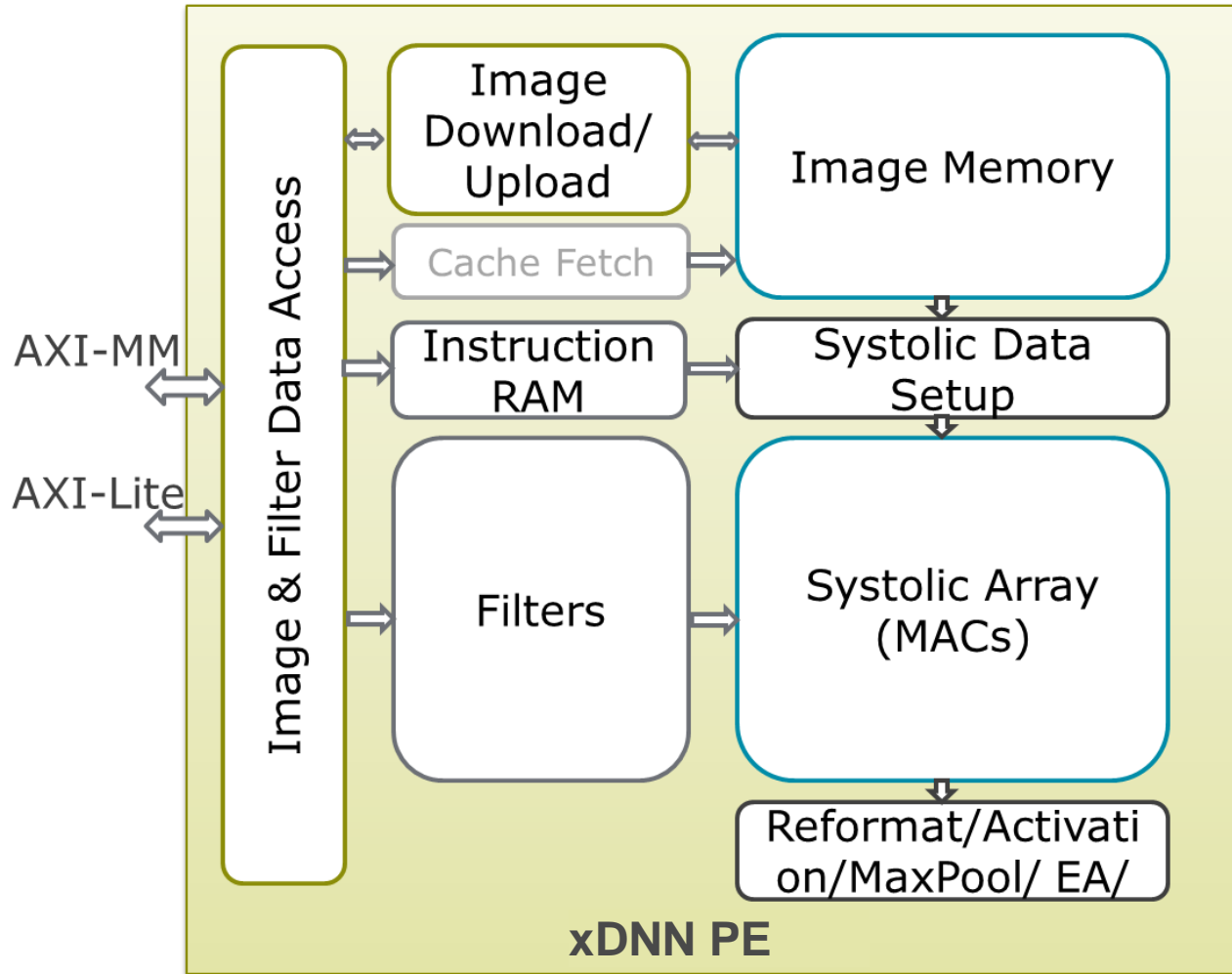


- 3) Specify desired precision
  - Quantizes to <8 bits to match Xilinx's DSP

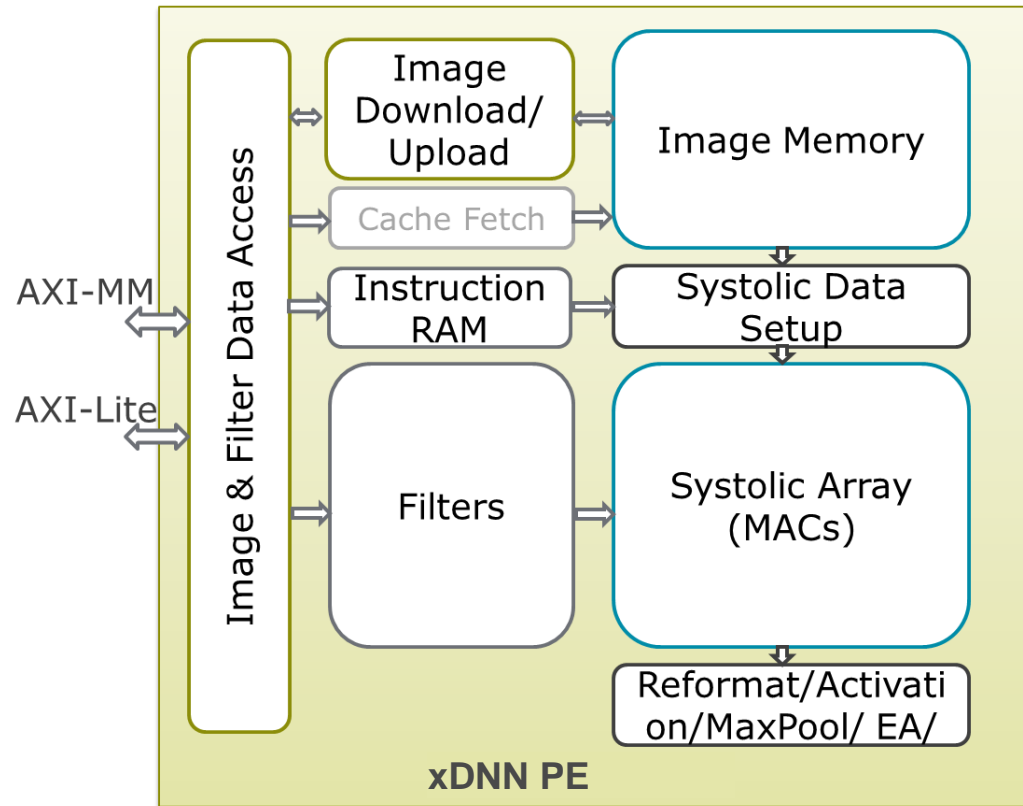
# ML-suite - HW xDNN Processing Engine



# Xilinx CNN Processing Engine – xDNN

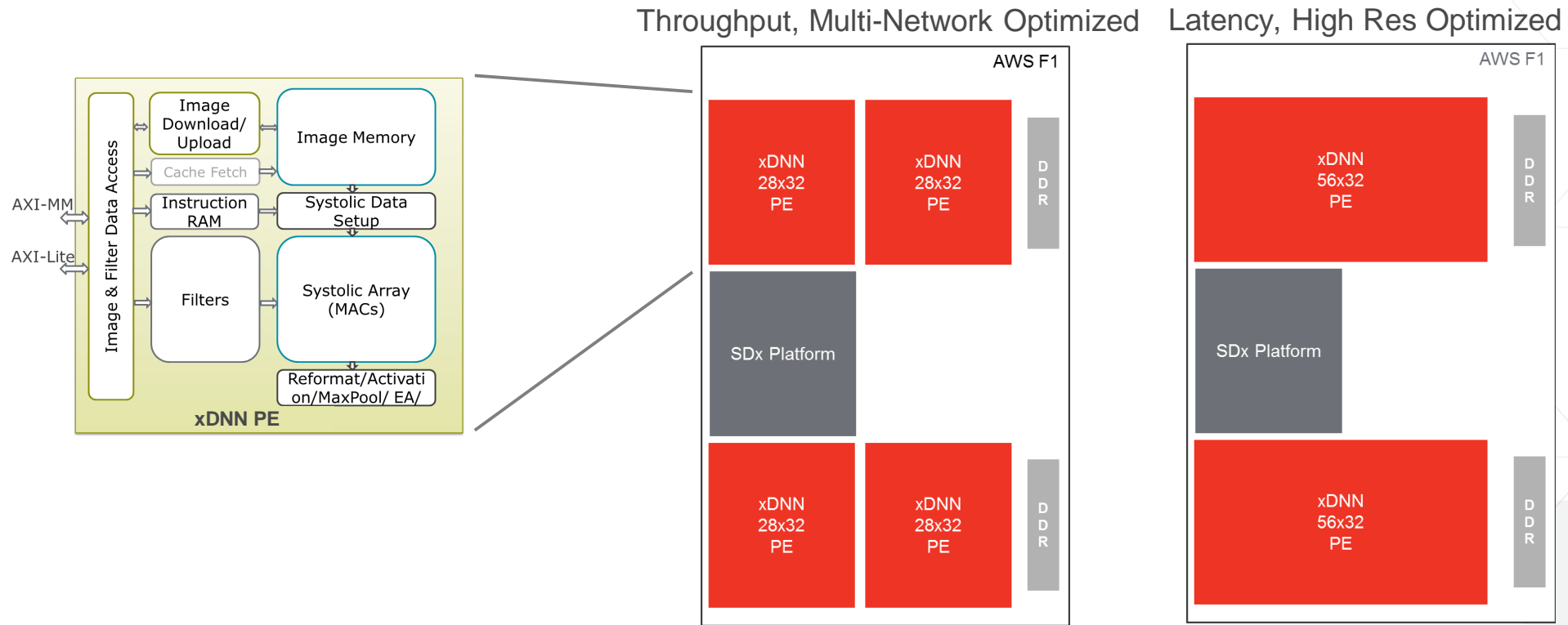


# Wide Range of Deep Learning Models Supported v2



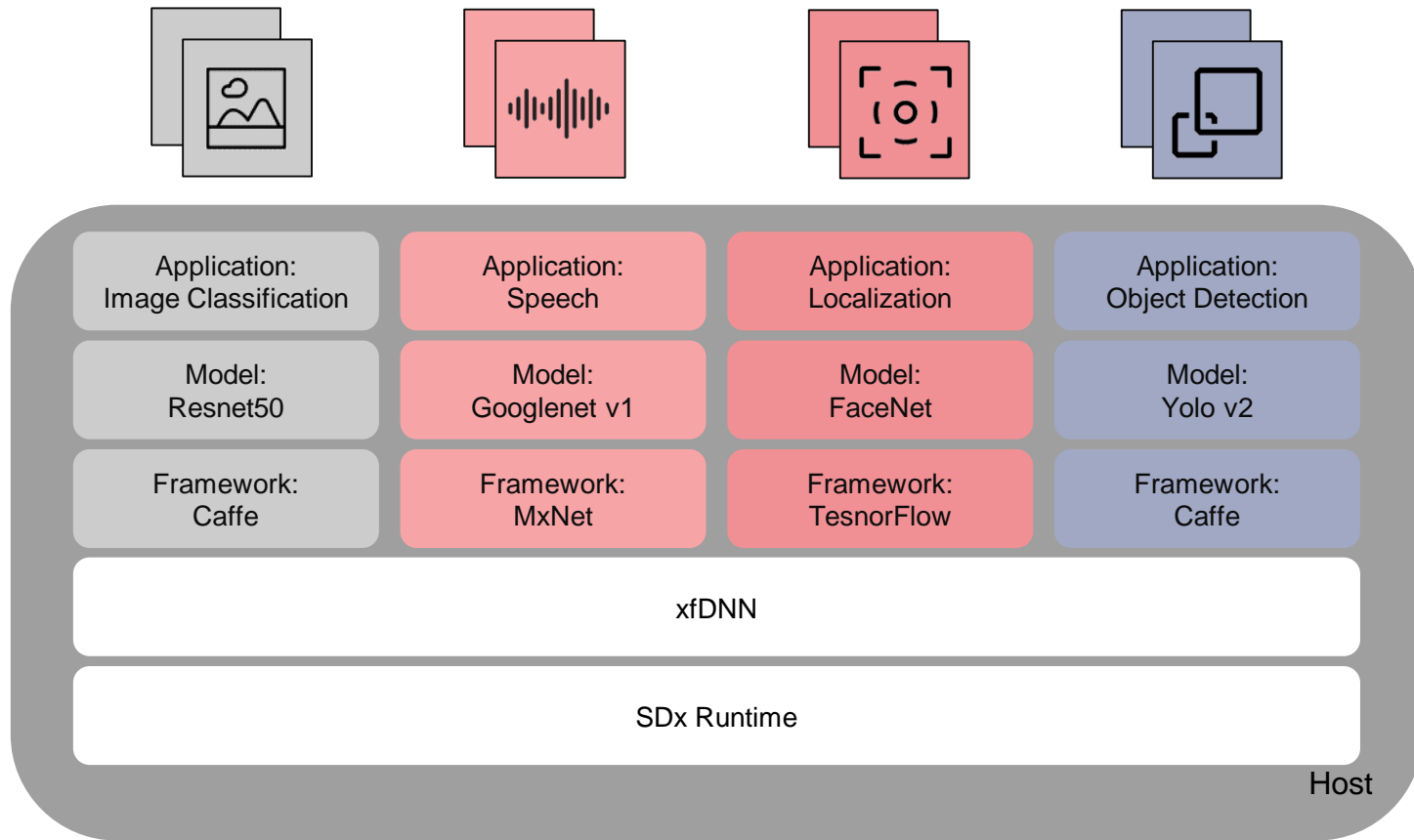
Features		Description	
Supported Operations	Convolution / Deconvolution / Convolution Transpose	Kernel Sizes	W: 1-15; H:1-15
		Strides	W: 1,2,4,8; H: 1,2,4,8
		Padding	Same, Valid
		Dilation	Factor: 1,2,4
		Activation	ReLU
		Bias	Value Per Channel
	Scaling	Scale & Shift Value Per Channel	
	Max Pooling	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

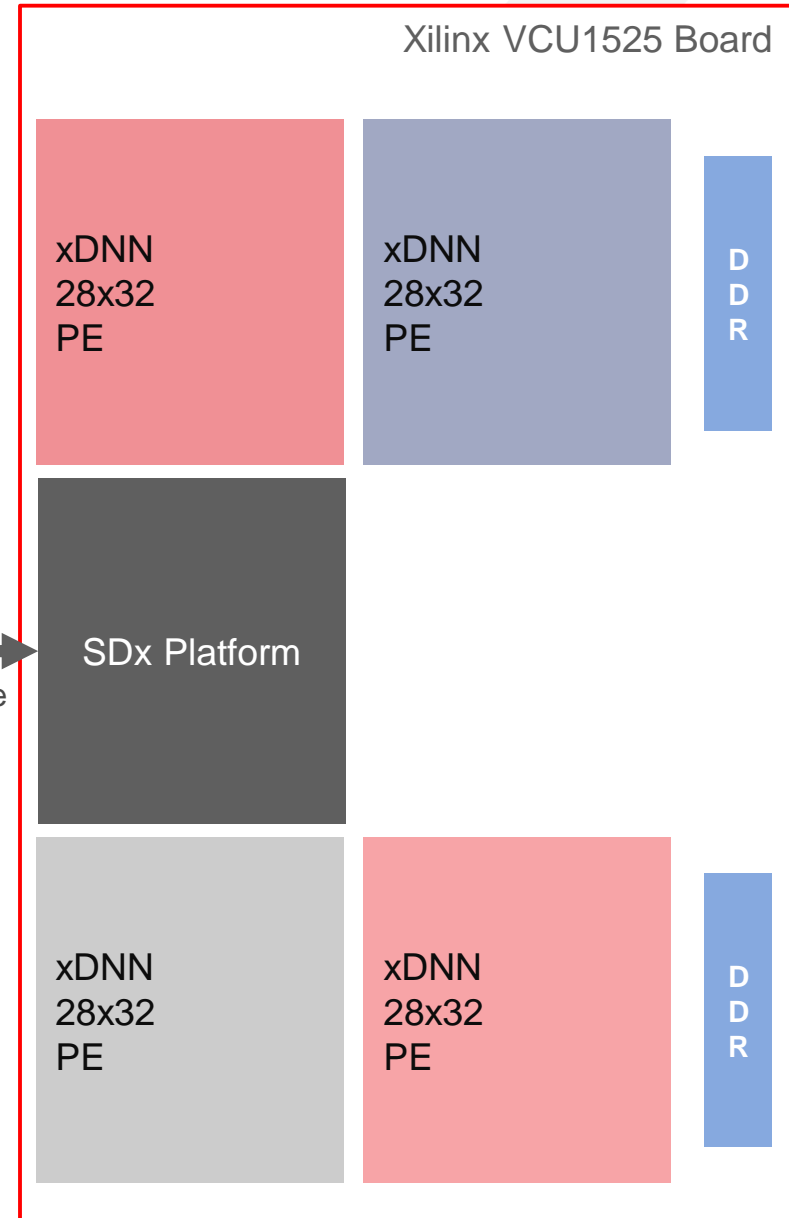


PE	#DSPs	Cache	16 bit GOP/s	8 bit GOP/s	Googlenet V1 latency	Optimized For	Examples Networks
28x32	896	5 MB	896	1,792	5.51	Multi-Network, Maximum Throughput	ResNet50 (224x224)
56x32	1792	5 MB	1,792	3,584	3.23	Lowest Latency Streaming	Yolov2 (224x224)
56x32	1792	8 MB	1,792	3,584	3.23	Lowest Latency, High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

# Flexible: Multi-Network Configuration

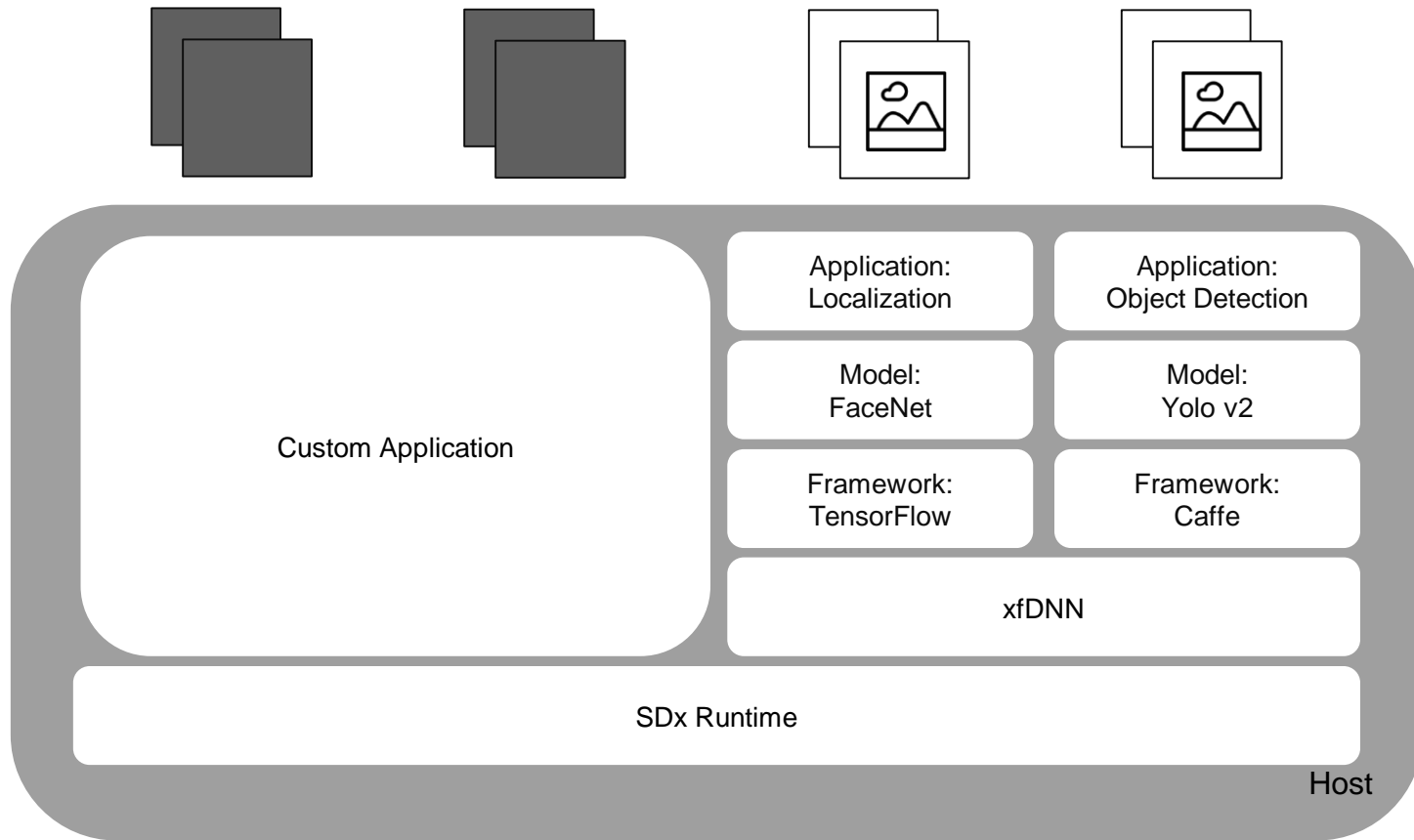


PCle

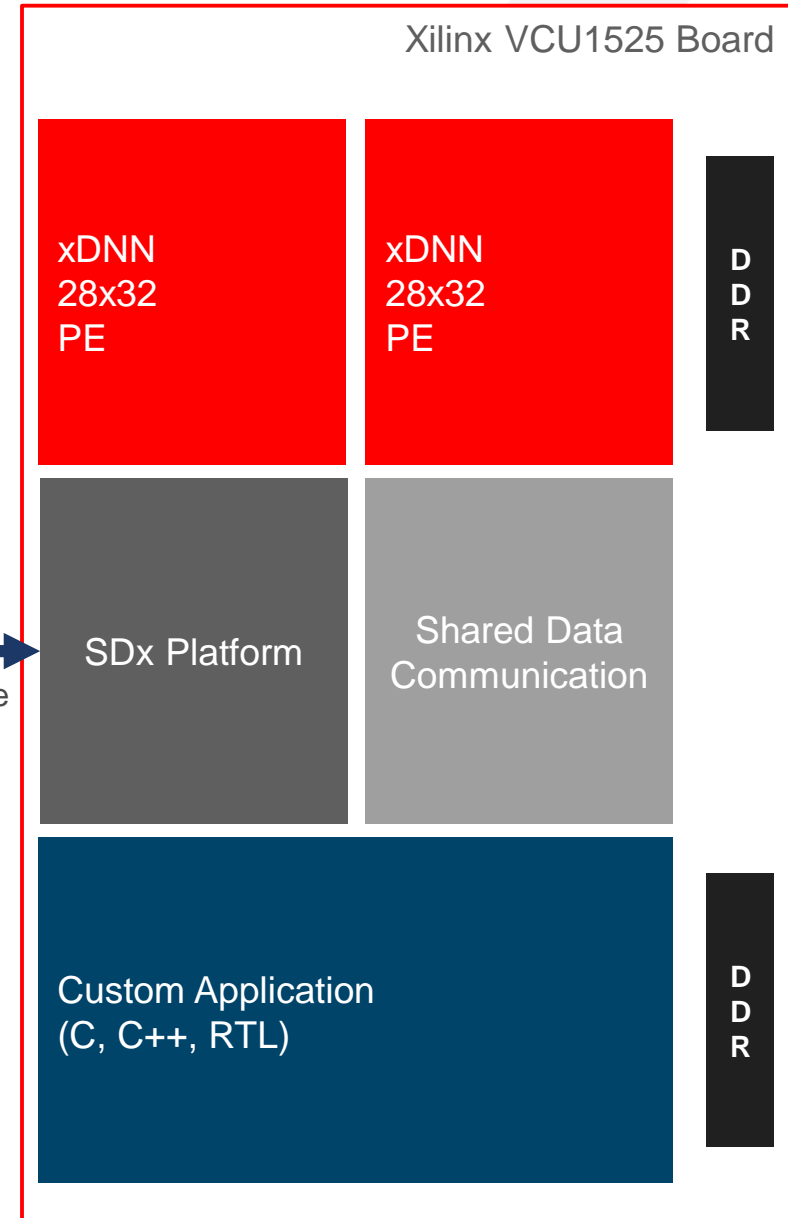


1 FPGA Provides 4 Virtual Accelerators For Real Time Deep Learning

# Flexible: Bring Your own IP!

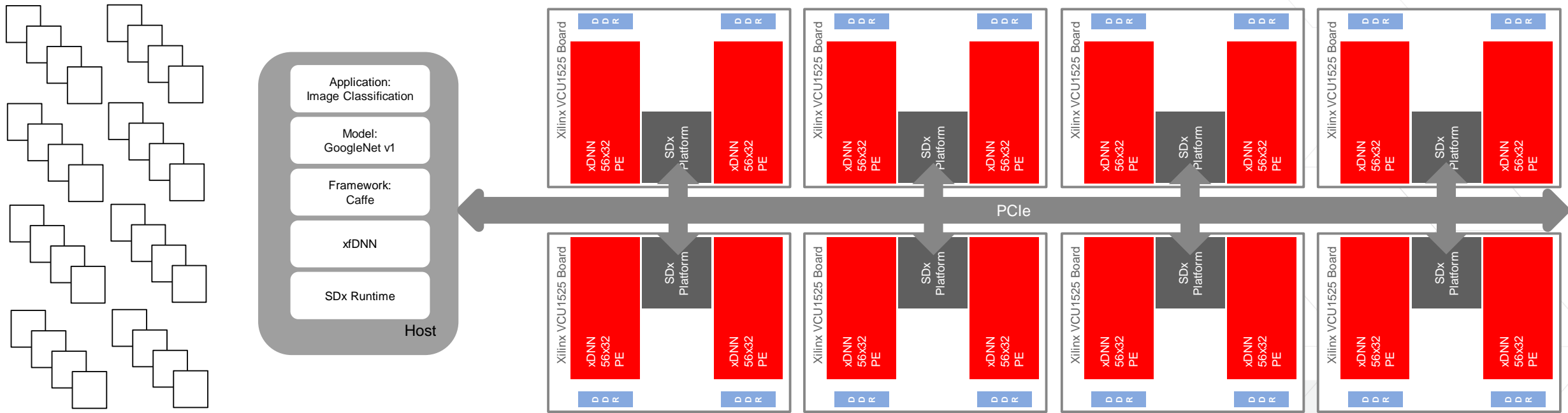


PCle



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

The screenshot shows the AWS EC2 Management Console interface. The browser address bar indicates the URL: <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:>. The page title is "Step 1: Choose an Amazon Machine Image (AMI)". Below the title, there is a search bar with "xilinx" entered. The search results show a single product: "Machine Learning Deployment Stack from Xilinx". The product details include: "★★★★★ (0) | 18.04.02 | Sold by Xilinx", "\$0.00/hr for software + AWS usage fees", and "Linux/Unix, CentOS 7.3 | 64-bit Amazon Machine Image (AMI) | Updated: 4/02/18". A "Select" button is visible next to the product listing.

<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

**Prototxt to FPGA Tutorial**  
Introduction  
This example describes running it on the F...  
The second step is and connecting to

**Compiler Tutorial**  
Introduction  
The compiler script interfaces with Caffe to read a deep learning model and generates a sequence of instructions for the xfdnn framework to memory allocation

**Quantization Tutorial**  
Introduction  
The quantize script produces an optimal target quantization within a matter of minutes from a given network (prototxt and caffemodel) and

**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/

# 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
- GoogleNet1

- Tensorflow

- GoogleNet1
- FCN
- Flowers102
- Place365
- ResNet50
- VGG16

The screenshot shows the GitHub repository for Xilinx/ml-suite. The repository is on the v1.0-ea branch, which is 5 commits ahead of master. The latest commit is by kamranjk, updating compile.md, 8 days ago. The file list includes:

- img (initial v1.0 ea, 15 days ago)
- README.md (initial v1.0 ea, 15 days ago)
- anaconda.md (initial v1.0 ea, 15 days ago)
- aws-f1-launching.md (initial v1.0 ea, 15 days ago)
- compile.md (Update compile.md, 8 days ago)
- deepdetect\_rest.md (initial v1.0 ea, 15 days ago)
- deepdetect\_webcam.md (initial v1.0 ea, 15 days ago)
- googlenet-demo.md (initial v1.0 ea, 15 days ago)
- quantize.md (initial v1.0 ea, 15 days ago)
- start-anaconda.md (initial v1.0 ea, 15 days ago)

The README.md content is as follows:

## ML Suite Tutorials

### xFDNN Tools

- [Compiling Networks with xFDNN Compiler](#)
- [Calibrate High Precision Models for INT16/8 Deployment](#)

### Image Classification

- [Image Classification GoogLeNet v1 Demo](#)
- [Image Classification with Python APIs](#)

### Object Detection

- [Object Detection with Yolo](#)

### Web Services

- [DeepDetect REST Tutorial](#)
- [DeepDetect Webcam Tutorial](#)

# 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/xfdn_18_03_19/caffe/ || Cd /home/centos/xfdn_18_03_19/tensorflow/`
3. `./start_docker.sh`
4. `export XFDNN_ROOT=/xlnx`
5. `cd /xlnx/xfdn_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

The image shows a terminal window on the left and a TensorBoard window on the right. The terminal displays the execution of a YOLOv2 model on an FPGA, including hardware configuration details and the results of object detection on the image 'dog.jpg'.

```
centos@ip-172-31-29-35:~/xfdnn_1...  
[XDN] using custom DDR banks 1  
Device/Slot[0] (/dev/xdma0, 0:0:1d.0)  
xclProbe found 1 FPGA slots with XDM...  
CL_PLATFORM_VENDOR Xilinx  
CL_PLATFORM_NAME Xilinx  
CL_DEVICE 0: 0x24fe300  
CL_DEVICES_FOUND 1, using 0  
loading kernel.aws.max.support.608.a...  
[XBLAS] kernel0: kernelSxdnn_0  
Running network input 608x608 and ou...  
Loading weights/bias/quant_params to...  
Preparing Input...  
['dog.jpg', 'giraffe.jpg', 'person.j...  
d588ce9 z.jpg', 'DemoImages/15439525...  
es/3591612840_33710806df z.jpg', 'Dei...  
.jpg', 'DemoImages/4814953542_de4b97...  
36_ffebfa2bea z.jpg', 'DemoImages/78...  
ages/AdrianStoica_Rory_discdog.jpeg']  
Preparing Output...  
Reading labels...  
Loading script...  
Running 16 image(s)  
  
Total FPGA: 1161.853790 ms  
Image Time: (72.615862 ms/img):  
  
Results for image 0: dog.jpg  
Found 3 boxes  
Obj 0: dog  
  score = 0.639982  
  (xlo,ylo) = (136,529)  
  (xhi,yhi) = (306,210)  
Obj 1: car  
  score = 0.432358  
  (xlo,ylo) = (419,173)  
  (xhi,yhi) = (689,81)  
Obj 2: bicycle  
  score = 0.572274  
  (xlo,ylo) = (112,475)  
  (xhi,yhi) = (564,92)
```

The TensorBoard window shows the image 'dog.jpg' with three bounding boxes: a blue box around the dog (score 0.64), a green box around the bicycle (score 0.57), and a yellow box around the car (score 0.43). The TensorBoard interface includes a search bar, a stack of images, and a 'ReadFile' button.

**Adaptable.**  
**Intelligent.**

