tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"DeepMaker - Deep Learning Accelerator on Commercial Programmable Devices"

Masoud Daneshtalab – Mälardalen University

Sweden Area Group – June 17, 2021

www.tinyML.org

TINY ML TALKS webcast

tinyML Talks Sponsors

arm
tinyML Strategic Partner

Deeplite

EDGE IMPULSE
tinyML Strategic Partner

emza visual sense
tinyML Strategic Partner

GREENWAVES TECHNOLOGIES
tinyML Strategic Partner

LatentAI
Adaptive AI for a Smarter Edge
tinyML Strategic Partner

HOTG
tinyML Strategic Partner

maxim integrated™

Qeexo

Qualcomm
tinyML Strategic Partner

Reality AI®
tinyML Strategic Partner

SensiML™
tinyML Strategic Partner

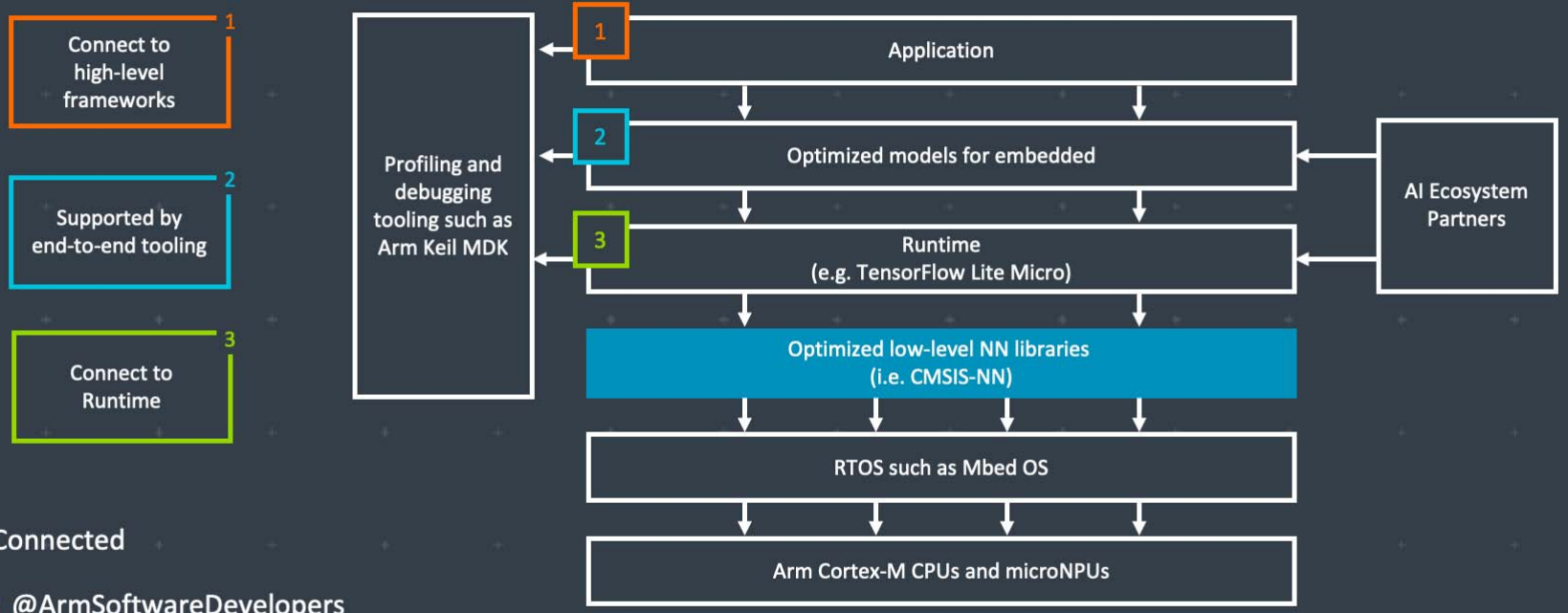
SynSense

SYNTIANT
tinyML Strategic Partner

Additional Sponsorships available – contact Olga@tinyML.org for info

TINY ML TALKS webcast

Arm: The Software and Hardware Foundation for tinyML



Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

arm

TINY ML TALKS webcast

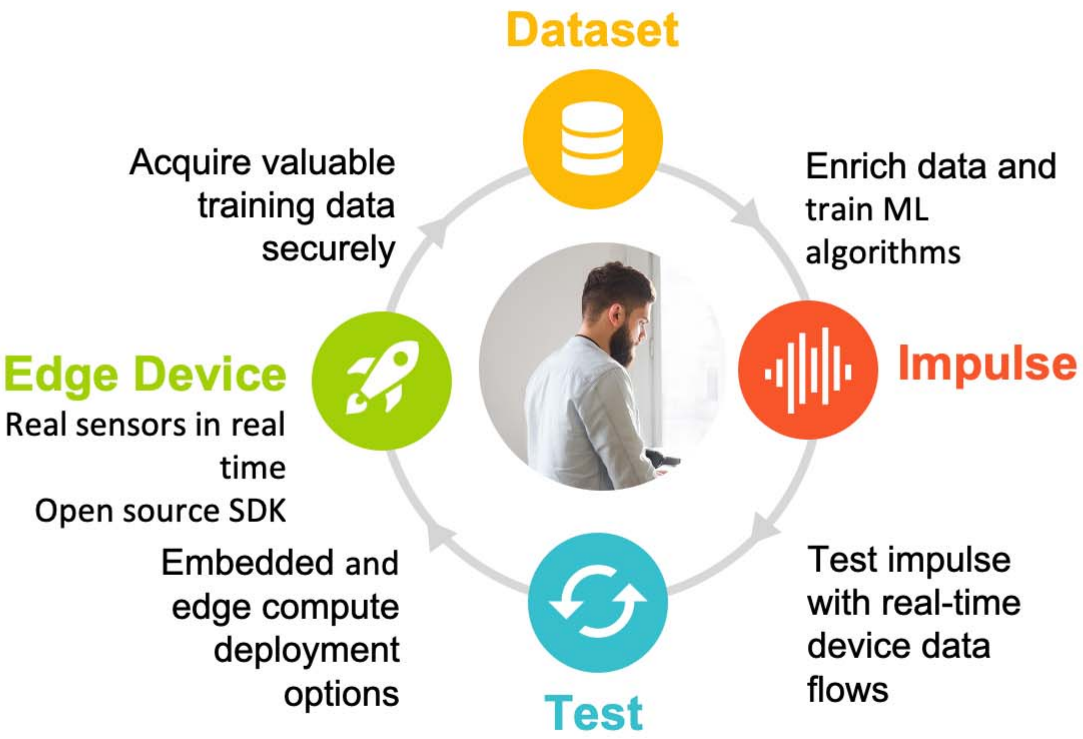
TINY

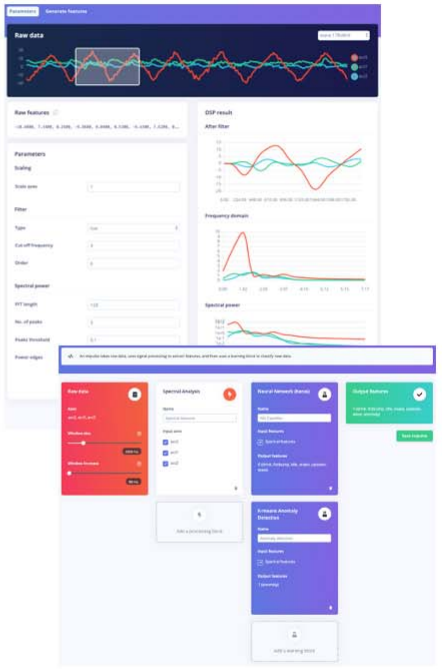
TinyML for all developers

C++ library

Arduino library

WebAssembly





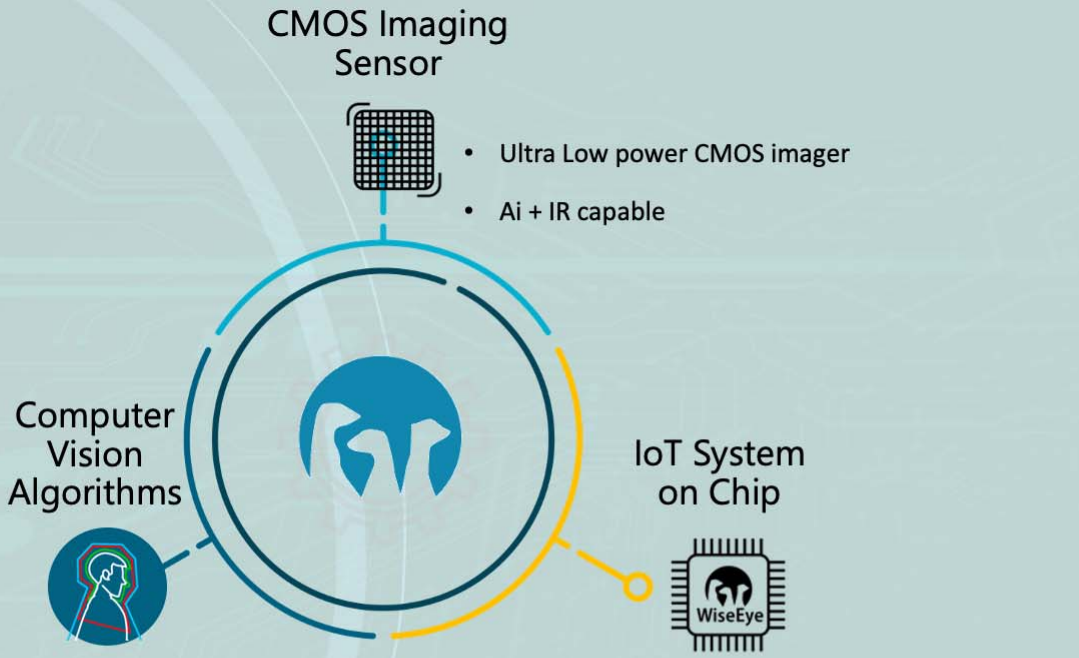
www.edgeimpulse.com

TINY ML TALKS webcast

emza visual sense

The Eye in IoT

Edge AI Visual Sensors



info@emza-vs.com

TINY ML TALKS webcast

Enabling the next generation of Sensor and Hearable products to process rich data with energy efficiency

Visible Image

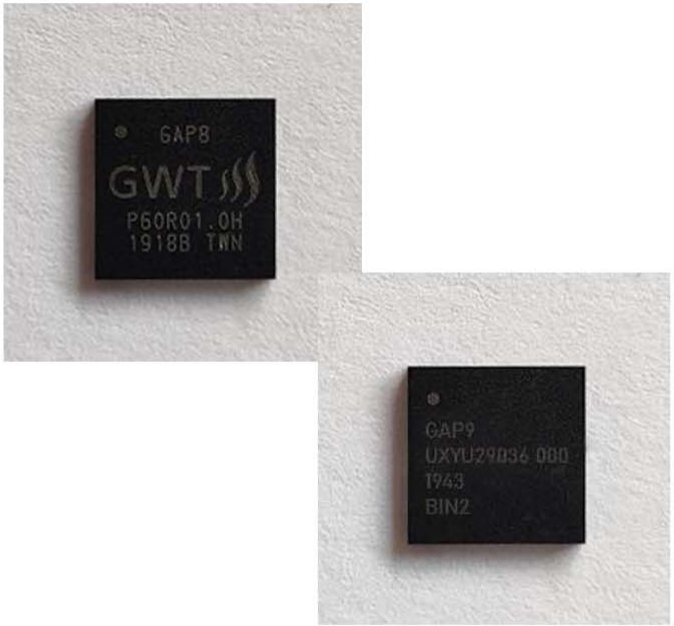
Sound

IR Image

Radar

Bio-sensor

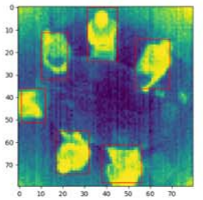
Gyro/Accel



Wearables / Hearables

Battery-powered consumer electronics

IoT Sensors



GREENWAVES TECHNOLOGIES

TINY ML TALKS webcast

LatentAI

Adaptive AI for the Intelligent Edge

Latentai.com

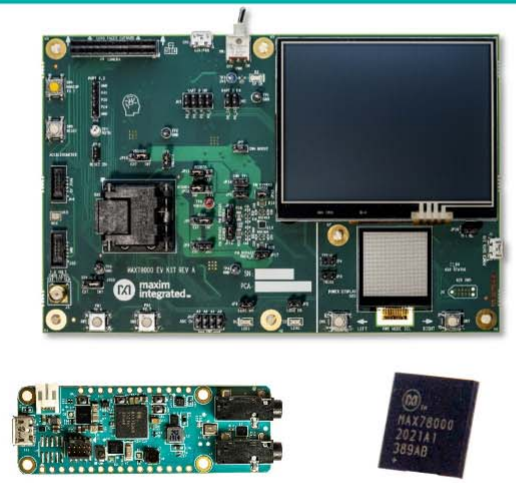
TINY ML TALKS webcast

TINY ML TALKS webcast

maxim integrated™

Maxim Integrated: Enabling Edge Intelligence

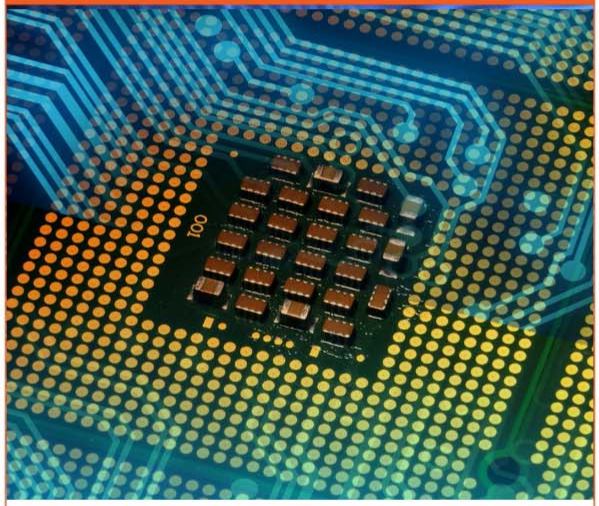
Advanced AI Acceleration IC



The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Low Power Cortex M4 Micros



Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Sensors and Signal Conditioning

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors

maxim integrated™

TINY ML TALKS webcast

Qeexo AutoML

Qeexo

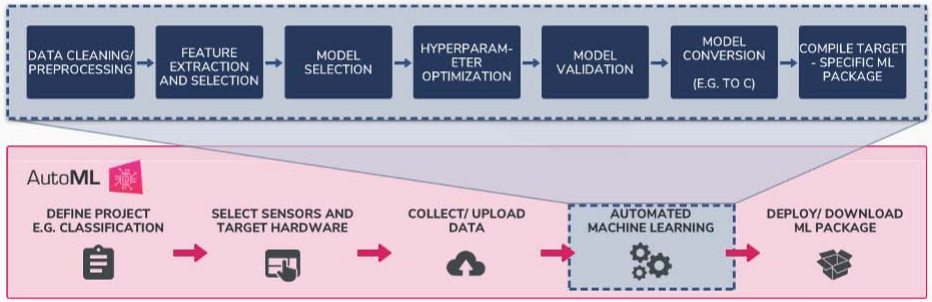
CES INNOVATION AWARDS 2021 HONOREE

Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Supports 17 ML methods:
  - Multi-class algorithms: GBM, XGBoost, Random Forest, Logistic Regression, Gaussian Naive Bayes, Decision Tree, Polynomial SVM, RBF SVM, SVM, CNN, RNN, CRNN, ANN
  - Single-class algorithms: Local Outlier Factor, One Class SVM, One Class Random Forest, Isolation Forest
- Labels, records, validates, and visualizes time-series sensor data
- On-device inference optimized for low latency, low power consumption, and small memory footprint applications
- Supports Arm® Cortex™- M0 to M4 class MCUs

End-to-End Machine Learning Platform



For more information, visit: www.qeexo.com

Target Markets/Applications

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

TINY ML TALKS webcast

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

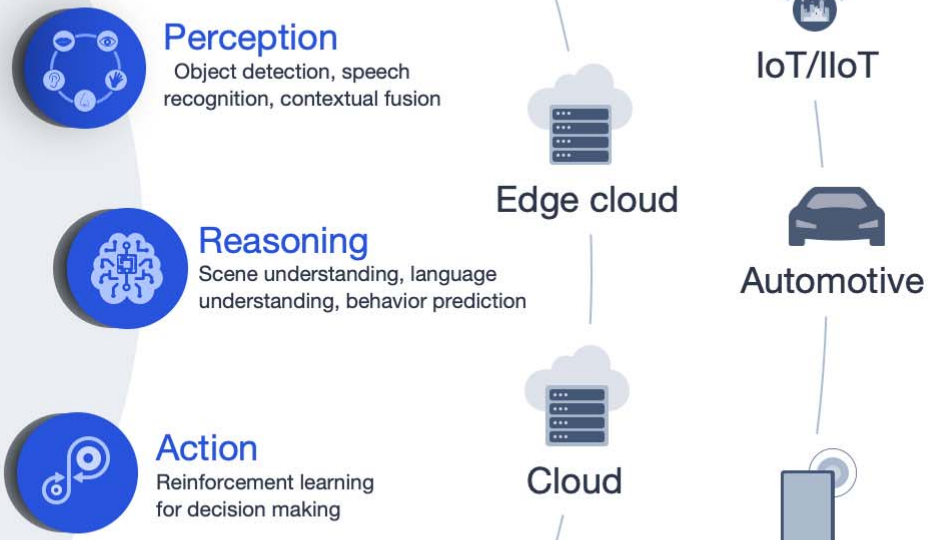
Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry



TINY ML TALKS webcast

Reality AI®

Add Advanced Sensing to your Product with Edge AI / TinyML

https://reality.ai info@reality.ai @SensorAI Reality AI

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions

- Prebuilt sound recognition models for indoor and outdoor use cases
- Solution for industrial anomaly detection
- Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software

- Build prototypes, then turn them into real products
- Explain ML models and relate the function to the physics
- Optimize the hardware, including sensor selection and placement

TINY ML TALKS webcast

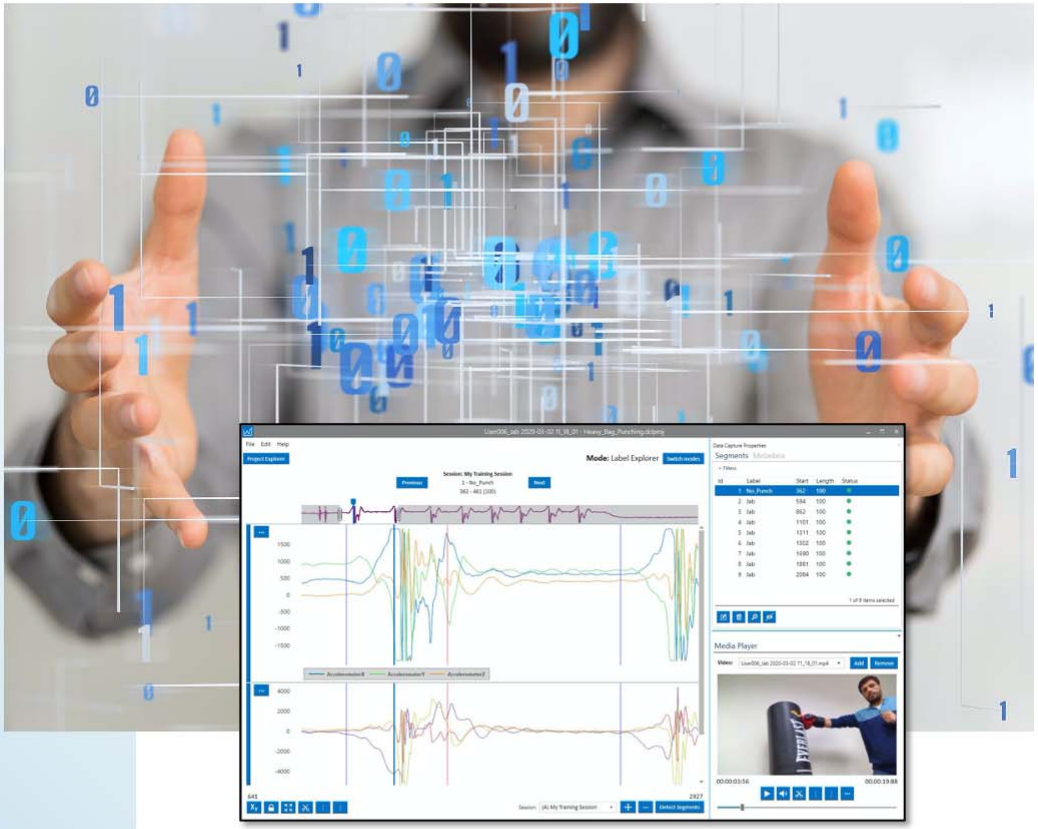
SensiML™

Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

- End-to-end AI workflow
- Multi-user auto-labeling of time-series data
- Code transparency and customization at each step in the pipeline

We enable the creation of production-grade smart sensor devices.



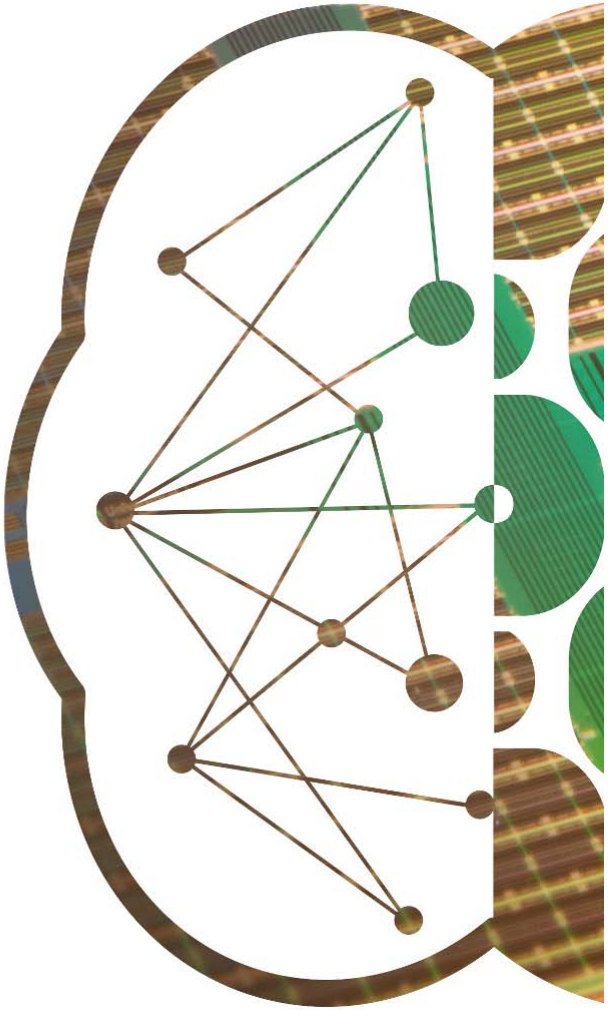
sensiml.com

TINY ML TALKS webcast

SynSense

SynSense builds **sensing and inference** hardware for **ultra-low-power** (sub-mW) **embedded, mobile and edge** devices. We design systems for **real-time always-on smart sensing**, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai



TINY ML TALKS webcast

SYNTIANT

Syntiant Corp. is moving artificial intelligence and machine learning from the cloud to edge devices. Syntiant's chip solutions merge deep learning with semiconductor design to produce ultra-low-power, high performance, deep neural network processors. These network processors enable always-on applications in battery-powered devices, such as smartphones, smart speakers, earbuds, hearing aids, and laptops. Syntiant's Neural Decision Processors™ offer wake word, command word, and event detection in a chip for always-on voice and sensor applications.

Founded in 2017 and headquartered in Irvine, California, the company is backed by Amazon, Applied Materials, Atlantic Bridge Capital, Bosch, Intel Capital, Microsoft, Motorola, and others. Syntiant was recently named a CES® 2021 Best of Innovation Awards Honoree, shipped over 10M units worldwide, and unveiled the NDP120 part of the NDP10x family of inference engines for low-power applications.

www.syntiant.com

@Syntiantcorp

TINY ML FOUNDATION

collaboration with

hackster.io
AN AVNET COMMUNITY

Focus on:
(i) developing new use cases/apps for tinyML vision; and (ii) promoting tinyML tech & companies in the developer community

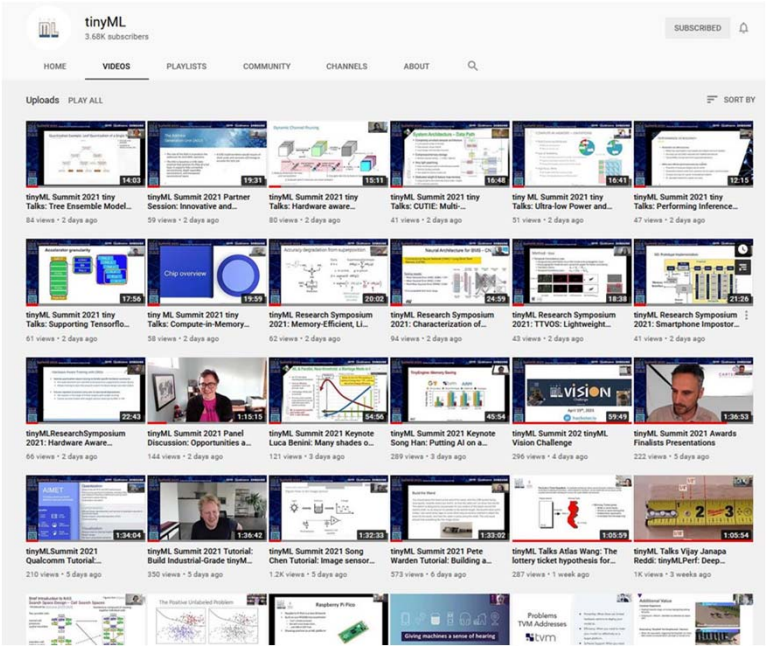
Submissions accepted until August 15th, 2021
Winners announced on September 1, 2021 ($6k value)
Sponsorships available: *sponsorships@tinyML.org*

https://www.hackster.io/contests/tinyml-vision

Successful tinyML Summit 2021:

www.youtube.com/tinyML with 150+ videos

- **5** days of tutorials, talks, panels, breakouts, symposium
  - **4** tutorials
  - **6** keynotes & **6** plenary tinyTalks (more in breakouts)
  - **2** panel discussions
  - **5** disruptive news presentations
  - **17** breakout/partner sessions
  - **6** Best Product and Innovation Award Finalists & Presentations
  - **89** Speakers
- **5006** registered attendees representing:
  - **104** countries, **1000+** companies and **400+** academic institutions
- **26** Sponsoring companies



tinyML Summit-2022, January 24-26, Silicon Valley, CA

Local Committee in Sweden

Ali Balador, Senior Researcher RISE, Assistant Professor MDH, ali.balador@ri.se

Johan Malm, AI Engineer, PhD, Imagimob AB, johan.malm@imagimob.com

Magnus Melander, Evangelist and Co-founder THINGS, magnus@wbird.se

Åke Wernelind, Business Development, Imagimob AB, ake.wernelind@imagimob.com

Next tinyML Talks

Date	Presenter	Topic / Title
Tuesday, July 6	**Shivy Yohanandan,** Xailient	Cracking a 600 million year old secret to fit computer vision on the edge

Webcast start time is 8 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting

TINY ML TALKS webcast

Reminders

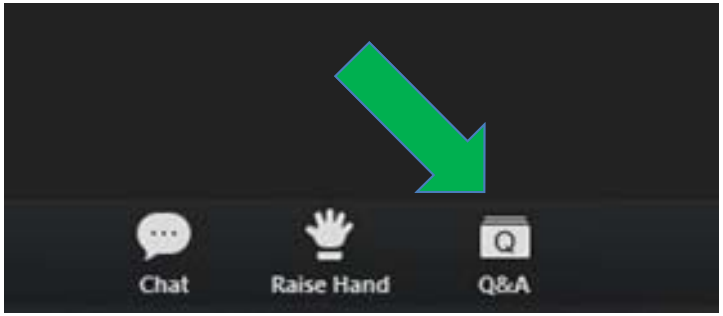
Slides & Videos will be posted tomorrow

PDF

tinyml.org/forums

youtube.com/tinyml

Please use the Q&A window for your questions



TINY ML TALKS webcast

Masoud Daneshtalab

Masoud Daneshtalab is professor at Mälardalen University (MDH) in Sweden and an adjunct Professor at Tallinn University of Technology (TalTech) in Estonia. He is co-leading the Heterogeneous System research group (www.es.mdh.se/hero/).

Since 2016 he has been in the Euromicro board of Directors, a faculty member of the HiPEAC network, and a permanent associate editor of Elsevier MICPRO. His research interests include hardware/software co-design and deep learning acceleration.

He has published 2 books and over 200 refereed international journals and conference papers.

Heterogeneous Systems Research Group

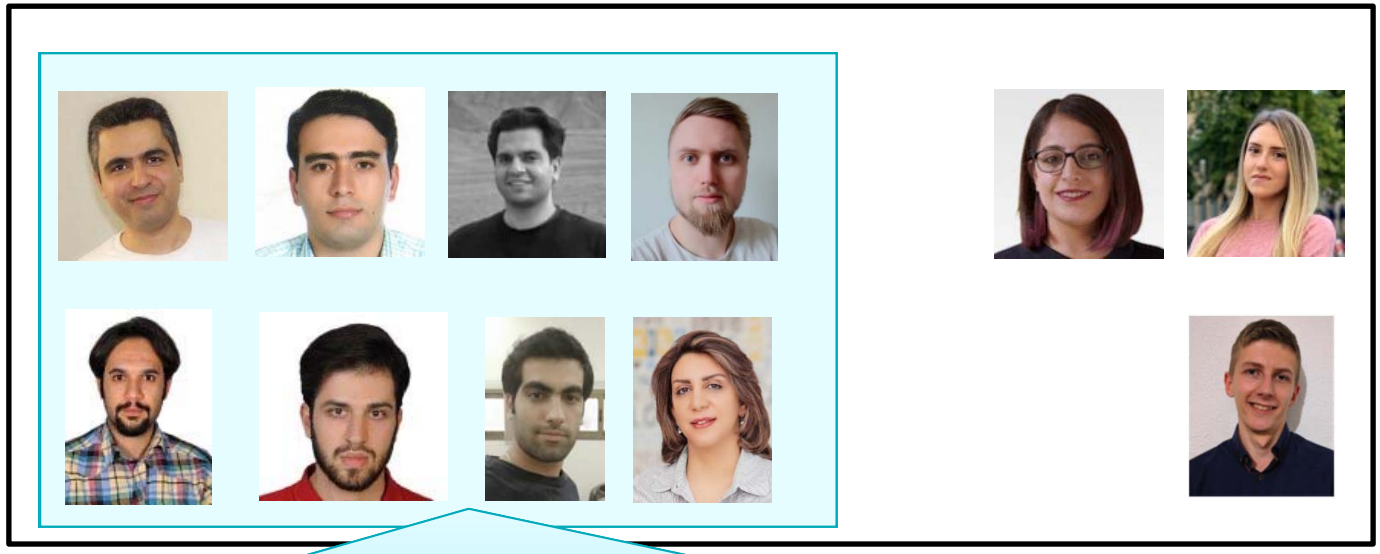
https://www.es.mdh.se/hero/ since 2018

Masoud Daneshtalab

This research group tackles the following scientific areas:

- Deep learning modeling and acceleration
- Hardware/software co-design and integration
- Interconnection and time-sensitive networking (TSN)
- Model-based development of heterogeneous systems

Members: 10 PhD students + 1 postdoc



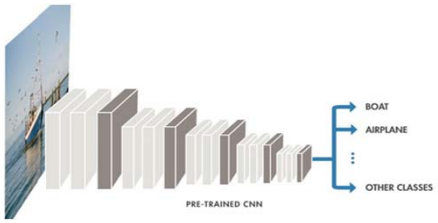
Projects on Deep Learning:

AutoDeep: Automatic Design of Safe, High-Performance and Compact Deep Learning Models for Autonomous Vehicles
SafeDeep: Dependable Deep Learning for Safety-Critical Airborne Embedded Systems,
HERO: Heterogeneous systems: software-hardware integration,
DeepMaker: Deep Learning Accelerator on Commercial Programmable Devices,

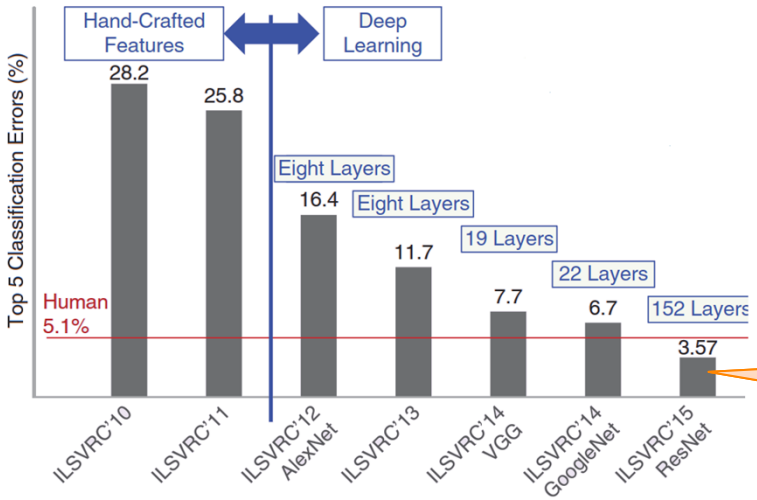
Agenda

- Challenges on deep learning
- Introduction on DeepMaker
- Neural Architecture Search (Frontend)
- FPGA implementation of DNNs (Backend)

Challenges



- Challenges:
  - How the **computation-intensive** DNN could be customized and deployed on the **resource-limited** hardware platform
  - How much to **rely on** their outputs.



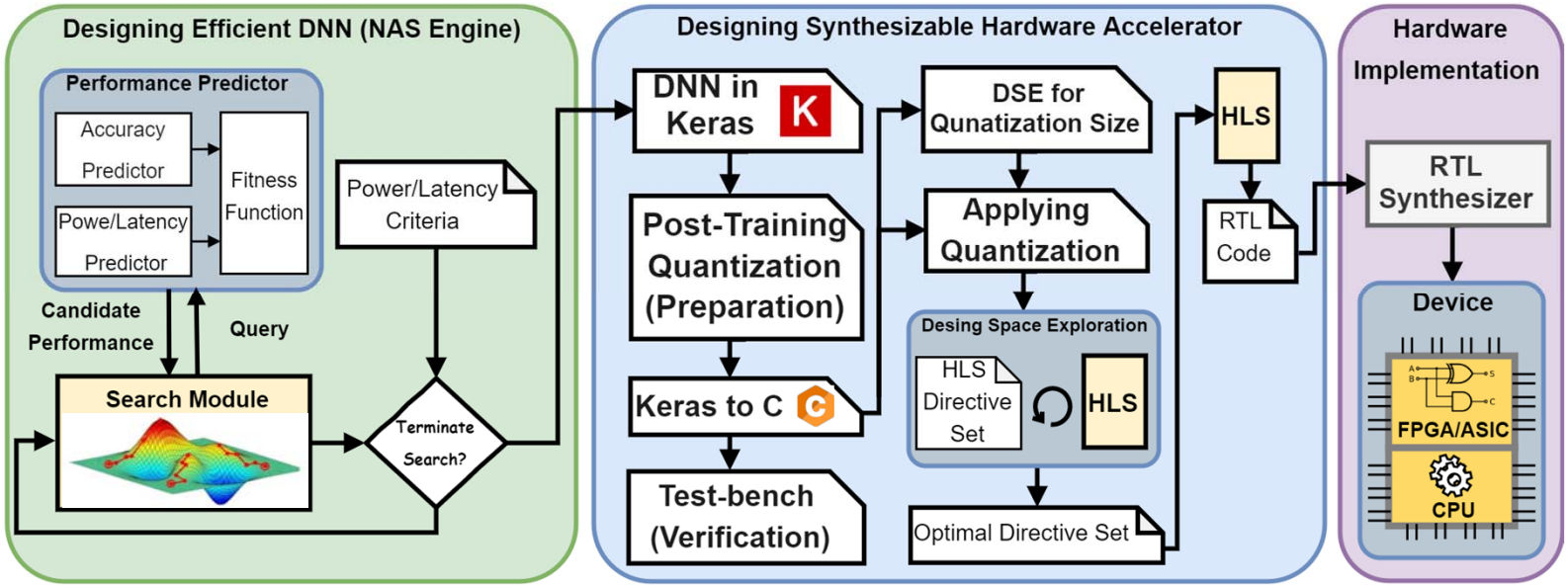
20M parameters:
15 Giga-floating point operations

Increasing the **complexity** of DL algorithms for achieving better accuracy [1].

[1]Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. IEEE Solid-State Circuits Magazine, 2018.

The DeepMaker Framework

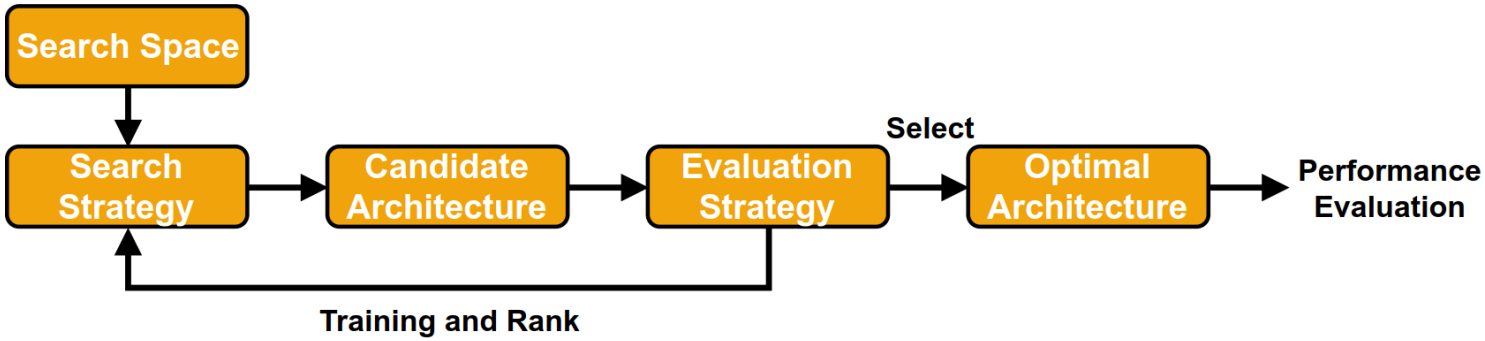
- provides resource-efficient DNNs with a required performance-level for (embedded) computing platforms.
  - Algorithmic techniques based on evolutionary **multi-objective optimization** to generate optimal DNN models (topology and hyper-parameters) in terms of:
    - Accuracy
    - Execution time (Flops)
    - Network complexity (Size)
  - Using **quantization techniques** to reduce the huge memory footprint and bandwidth requirements.



"DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems." Microprocessors and Microsystems (2020).

Neural Architecture Search (NAS)

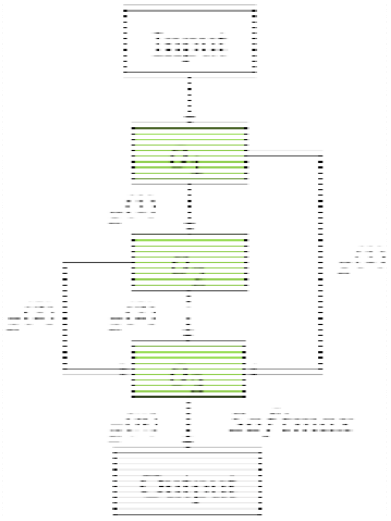
- NAS is to design DNN models automatically without human intervention.
- NAS methods try to design neural architectures with better accuracy.
  - But other critieria can also be considered in addition to accuracy:
    - Execution time (Flops)
    - Network complexity (Size)



Neural Architecture Search (NAS)

Macro NAS

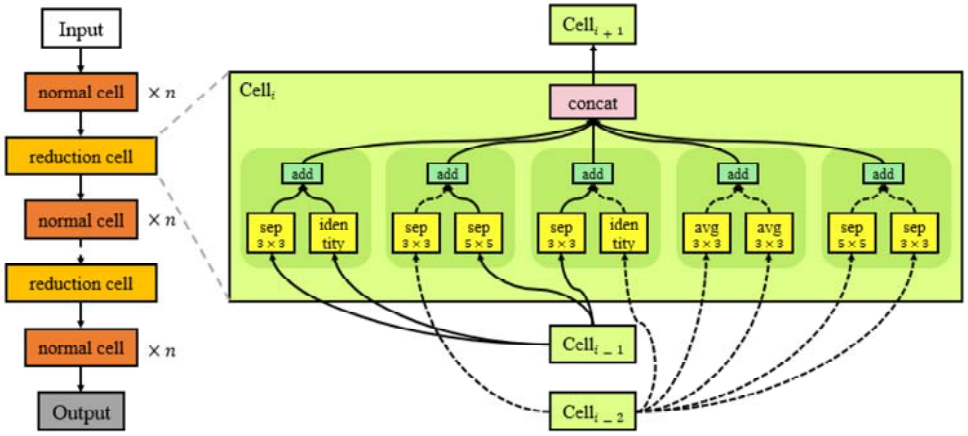
- Macro NAS search spaces with a chain structure.
- ***Oi*** is an operation and the *i-th* operation in the chain structure.
- The input goes through a series of operations to get the final output.



Slow Search, More Flexibility

Micro NAS

- Micro NAS search space often only needs to search a few small cell structures, and then repeatedly stack such cells to form the final architecture.



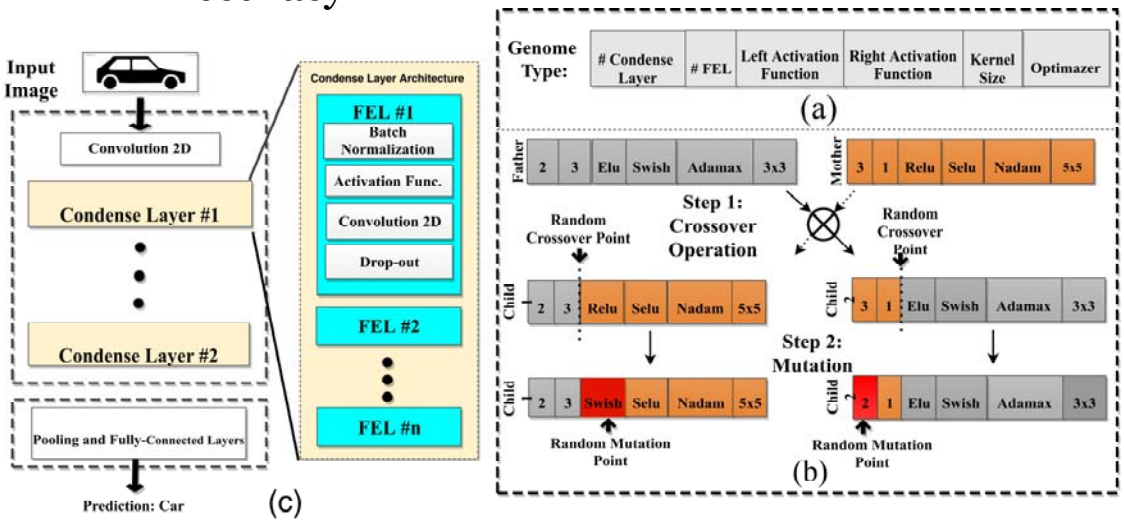
Fast Search, Less Flexibility

Popular Search Strategies

- Random Search
  - SMASH [Brock et al. 2017]
- Evolutionary Algorithm
  - Regularized evolution [Real et al. 2018] / DeepMaker [2020]
- Reinforcement Learning
  - REINFORCE: [Zoph and Le 2017], [Pham et al. 2018 1]
  - Proximal policy optimization (PPO): [Zoph et al. 2018]
- Bayesian Optimization (BO)
  - GP with string kernel
    - Vizier [Chen et al. 2018]
    - Guided ES [Liu et al. 2018 2]
  - Hyperband Bayesian optimization [Wang et al. 2018]
  - BO and optimal transport [Kandasamy et al. 2018]
- Gradient Based Optimization
  - SMASH [Brock et al. 2017]
  - ENAS [Pham et al. 2018 2]
  - DARTS [Liu et al. 2018 1]
  - ProxylessNAS [Cai et al. 2018]

Designing Optimal CNN Architecture

- **Goal:** Designing Energy-efficient CNN for resource-limited devices
- Using a template-based design space inspired by DenseNet architecture.
- Using NSGA-II as the multi-objective search engine
  - FLOPS
  - Accuracy



Parameter	Deep CNN
Activation Function	hard-sigmoid, relu, elu, tanh, sigmoid, softplus, linear
# Condense_Layer	1, 2, 3, 4
# Convolution_Layer	16, 28, 40, 52
Kernel Size	3x3, 5x5
Optimizer	rmsprop, adam, sgd, adagrad, adadelta, adamax, nadam

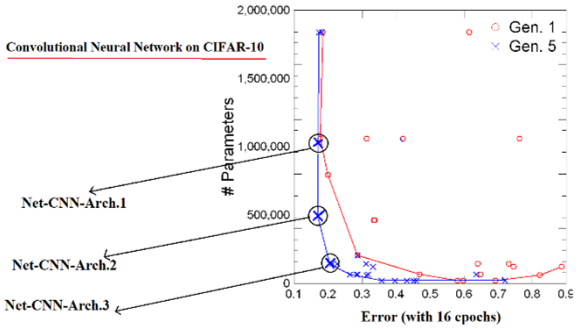
(a) A genome type representing NAS hyperparameters. (b) Crossover & mutation operators. (c) Inspired template architecture of a generated network.

1. "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems." Microprocessors and Microsystems 73 (2020).
2. "NeuroPower: Designing Energy Efficient Convolutional Neural Network Architecture for Embedded Systems." International Conference on Artificial Neural Networks. Springer, 2019
3. "Designing Compact Convolutional Neural Network for Embedded Stereo Vision Systems," in Proceedings of 12th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2018, Vietnam. (best paper award)

Classification Results

Dataset	Approach	Solutions	#Params ($\times 10^6$)	Error Rate (%)	Compression Rate*‡
MNIST	Hand-Crafted	Wan et al. [25]	–	0.21	–
	RL	MetaQNN [2]	5.59	0.35	0.023x
	MO²-EC	* ADONN-Arch.3 [17]	0.13	0.41	–
	MO²-EC	**Our M_Net_1**	**0.065**	**0.71**	**2x**
CIFAR-10	Hand-Crafted	* CondenseNet^{Light} [11]	3.1	3.46	–
	Hand-Crafted	SimpleNet [8]	5.48	4.68	0.53x
	Hand-Crafted	DenseNet (k = 12)-40 [12]	1.0	7.0	3.1x
	Hand-Crafted	ResNet-20 [9]	0.27	8.75	11.48x
	Hand-Crafted	ResNet-110 [9]	1.7	6.43	1.82x
	Hand-Crafted	Gastaldi et al. [5]	26.4	2.86	0.117x
	RL	Block-QNN-22L [29]	39.8	3.54	0.078x
	RL	MetaQNN [2]	6.92	11.18	0.45x
	RL	NAS-v1/v3 [32]	4.2/37.4	5.50/3.65	0.7x/0.083x
	RL	Block-QNN-S [29]	6.1	4.38	0.5x
	EC	Real et al. [21]	5.4	5.4	0.57x
	MO²-EC	NSGA-Net [19]	3.3	3.85	0.94x
	MO²-EC	ADONN-Arch.3 [17]	0.14	14.1	22.14x
	MO²-EC	Loni et al. [18]	0.56	13.8	5.5x
	MO²-EC	**Our C10-Net.1**	**0.065**	**16.49**	**47.7x**
	MO²-EC	**Our C10-Net.2**	**0.21**	**11.05**	**14.7x**
	MO²-EC	**Our C10-Net.3**	**1.0**	**6.81**	**3.1x**
CIFAR-100	RL	MetaQNN [2]	11.18	27.14	0.28x
	RL	Block-QNN-S [29]	6.1	20.65	0.5x
	Hand-Crafted	* CondenseNet^{Light} [11]	3.1	17.55	–
	Hand-Crafted	DenseNet (k = 12)-40 [12]	1.0	27.55	3.1x
	Hand-Crafted	DenseNet (k = 12)-100 [12]	7.0	23.79	0.44x
	Hand-Crafted	SimpleNet [8]	5.48	26.58	0.53x
	MO²-EC	NSGA-Net [19]	3.3	20.74	0.94x
	MO²-EC	**Our C100-Net.1**	**1.1**	**26.63**	**2.82x**
	MO²-EC	**Our C100-Net.2**	**1.89**	**24.87**	**1.64x**

* The baseline for comparing the compressing rate.
‡ The values more than 1.0 indicate improvement. Best results are in **bold**.



- **MNIST** (Compare to MetaQNN by ***Google***): **43x** compression rate, **0.06%** accuracy loss
- **CIFAR-10** (Compare to the most accurate): **26.4x** compression Rate, **4%** accuracy loss
- **CIFAR-10** (Compare to MetaQNN by ***Google***): **6.8x** compression Rate, **4.3%** better accuracy
- **CIFAR-100** (Compare to MetaQNN by ***Google***): **10x** compression Rate, **0.5%** accuracy loss
- **CIFAR-100** (Compare to the most accurate and light-weight network, CondensNet): **2.8x** compression Rate, **9%** accuracy loss

Hardware Implementation Results

(Compared to the ADONN with 0.14M parameters)

Platform			GPU			Intel ® CPU		ARM processor	
Dataset	Network	NID ($\times 10^6$)	Energy efficiency	Speedup (Kernel)	Speedup: D2H/H2D (Comm.)	Energy efficiency	Speedup (Inference Time)	Energy Efficiency	Speedup (Inference Time)
MNIST	* ADONN-Arch.3	7.06	–	–	–	–	–	–	–
	M_Net_1	**15.25**	**1.55x**	**1.49x**	**1.59x/2.95x**	**1.49x**	**1.51x**	**1.56x**	**1.38x**
CIFAR-10	ResNet-20	3.37	4.2x	4.25x	0.62x/0.79x	0.75x	0.81x	1.05x	0.87x
	ResNet-110	0.55	1.53x	1.64x	0.108x/0.15x	0.093x	0.95x	0.16x	0.87x
	DenseNet (k = 12)-100	0.135	0.25x	0.027x	0.05x/0.073x	0.099x	0.112x	0.07x	0.115x
	* ADONN-Arch.3	6.13	–	–	–	–	–	–	–
	C10_Net_1	**13**	1.54x	1.51x	**1.61x/1.57x**	**1.5x**	**1.59x**	**1.52x**	**1.32x**
	C10_Net_3	0.93	0.14x	0.154x	0.25x/0.313x	0.3x	0.372x	0.3x	0.37x
CIFAR-100	* ResNet-110	0.43	–	–	–	–	–	–	–
	DenseNet (k = 12)-100	0.109	0.017x	0.016x	0.44x/0.46x	0.92x	1.03x	0.34x	0.74x
	C100-Net.1	**0.66**	0.11x	0.1x	**2.28x/2x**	**3.2x**	**3.46x**	**1.89x**	**2.38x**
	C100-Net.2	**0.4**	0.06x	0.06x	**1.45x/1.39x**	**2.45x**	**2.53x**	**1.35x**	**1.83x**

* The baseline for comparing the energy efficiency of different architectures. The values more than 1.0 indicate improvement. Best results are in **bold**.

Industrial Use Case #1: Runway Detection for Auto-landing System

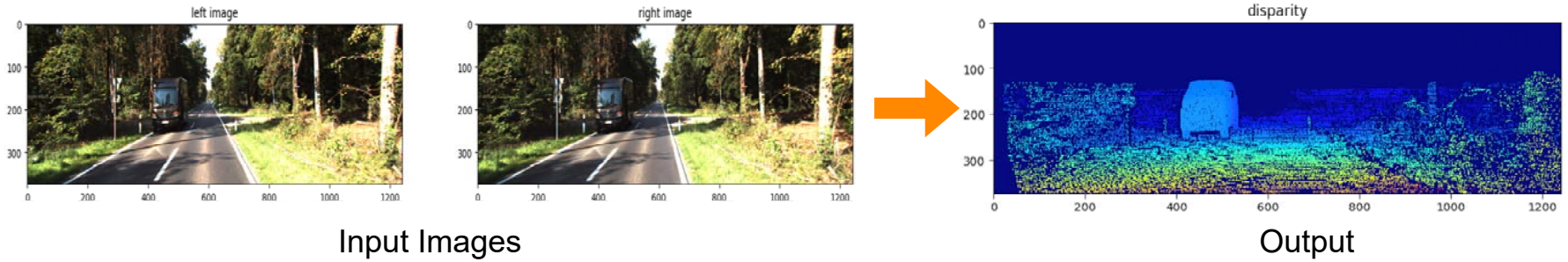
- All the results have been achieved by running on NVIDIA GTX 1080ti.
- **4x** better inference time
- **4.22x** more compact
- **4.15 %** more accurate

Classification Results

Solution	RetinaNet	DeepMaker
AVG. Accuracy	86.95 %	91.1%
Inference Time (ms)	63	**16**
Frame/Second	15	**62**

What is Stereo Matching?

- A piece of Stereo Vision process which aim to estimate depth of contents by getting the images of stereo camera.
- The main process in this part is finding equal pixels in stereo pictures and estimating their distances to generate **Disparity Map.**



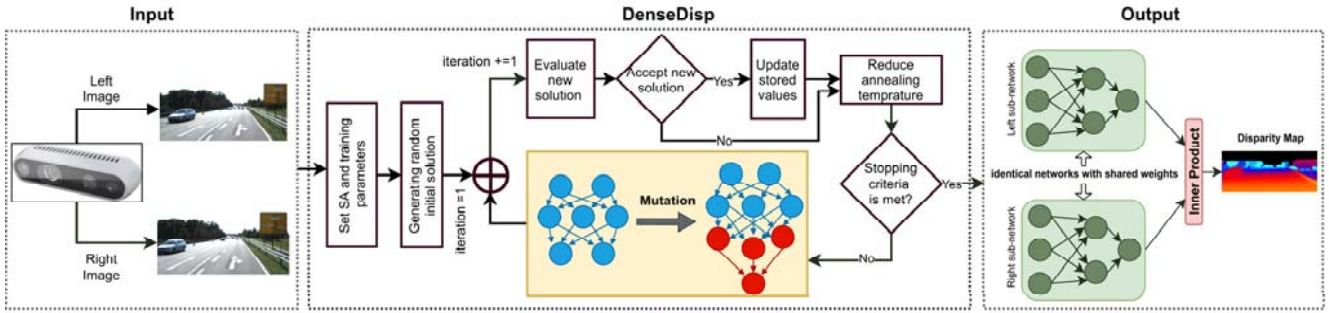
Input Images

Output

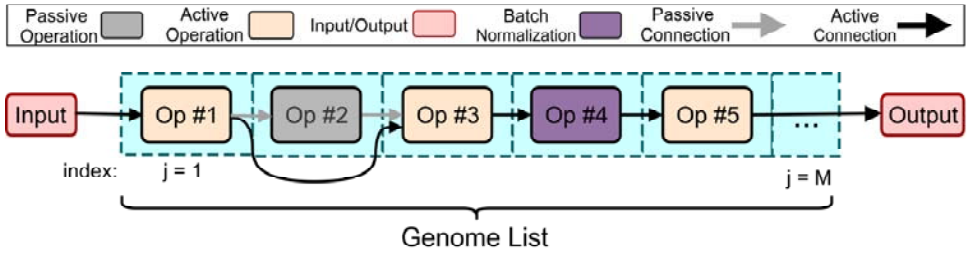
- Although CNN-based stereo matching algorithms provides superior accuracy, they suffer from huge memory footprint and computational cost.
- For example, full training a stereo matching model may costs under loading 4 GPUs for more than 6 hours!

DenseDisp: Optimizing CNN Architecture for Depth Estimation

- To reduce the computational demand of depth estimation task, we utilized multi-objective Simulated Annealing (SA) to design a hardware-friendly architecture.
  - We selected the Siamese neural model due to high flexibility.
  - Using SA to decrease the **search cost achieving 24x reduction** compared to state-of-the-art.



- We used Matrix design space to generate discrete chain macro NAS structure.



"DenseDisp: Resource-Aware Disparity Map Estimation by Compressing Siamese Neural Architecture." IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE (WCCI), 2020.

DenseDisp Experimental Results: Disparity Estimation Performance

- **KITTI-2015** is a dataset of real-world images that consist of 200 training and 200 test stereo pairs with the dimensions of 376×1240 pixels.
- Our DenseDisp architecture achieves **92.03%** accuracy with **0.6 Sec.** inference time and **1.03M parameters / 1.56M multiply-adds (FLOPs),** achieving a cutting-edge accuracy-FLOPs trade-off.

Classification/Complexity Results

Architecture	Accuracy (%) $D1-all$	Params $\times 10^6$	FLOPS $\times 10^6$	Exploration Method	Compression Rate† (%)	Exploration Cost (GPU Days)	GPU Latency (Sec.)	H.W. Platform
AutoDispNet-BOHB-C†	97.82	37	61	RL	1	42	-	1x GTX 1080ti
Content-CNN	95.46	7	2	Hand-Crafted	35.5	-	1	1x Titan Xp
GA-Net-deep	98.19	-	-	Hand-Crafted	-	-	1.8	1x Tesla P40
DenseDisp (Ours)	90.03	1.03	1.56	Meta-heuristic	39.1	2	0.56	1x GTX 2080
DenseDisp + Median Filter (Ours)	92.01	1.03	1.56	-	39.1	2	0.6	1x GTX 2080

† The baseline for comparing the compressing rate. Our considerable results are in green cells.

FPGA implementation Results.

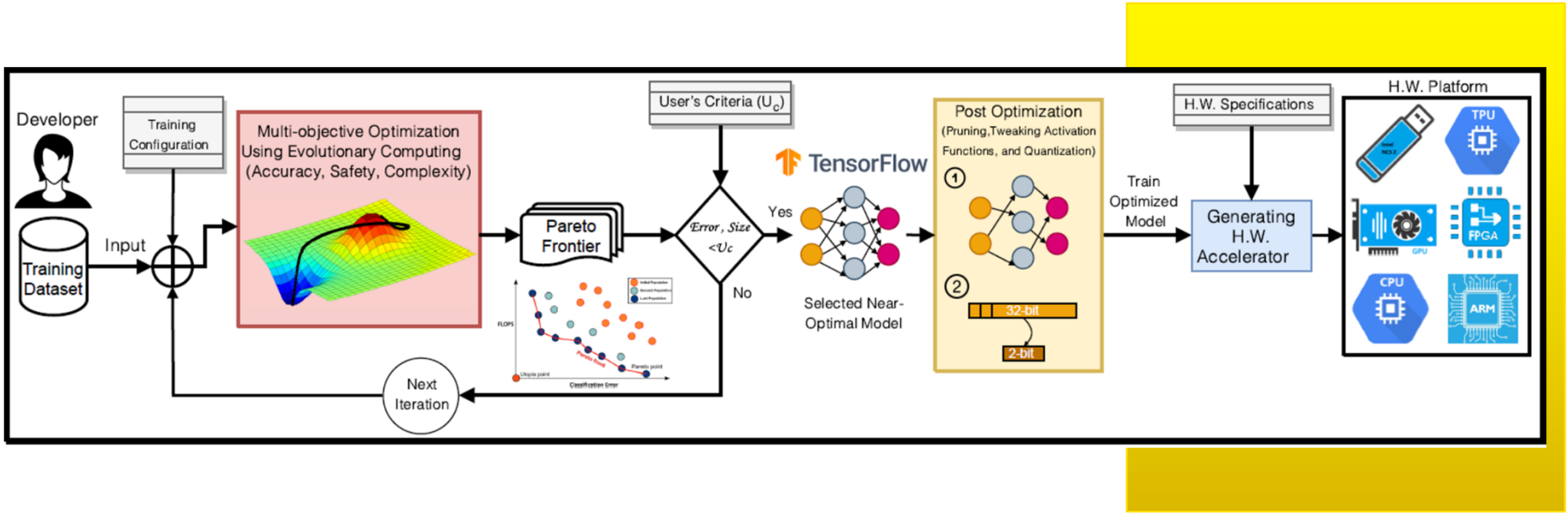
Work FPGA Platform	Method	Disparity Range	FPS	Accuracy (%) (D1-all)
(Schumacher & Greiner, 2014) Virtex-5	SGM	160	199	81.94
(Rahnama et al., 2018) Zynq ZC706 (CPU+FPGA)	ELAS†(Geiger et al., 2010)	-	9.5	86.1
(Rahnama et al., 2019) Zynq ZCU104 (CPU+FPGA)	SGM+ELAS	-	52	91.3‡
Content-CNN (Luo et al., 2016) Zynq UltraScale+	CNN	128	6	95.46
(Ours) Zynq UltraScale+	CNN	128	12	92.3/(94.3‡)

† The Efficient Large-Scale Stereo (ELAS), is the fastest triangulation-based disparity estimation algorithm on CPU.
‡ With using median filter.

DenseDisp was the only network we managed to deployed in Intel NCS2 (Movidius)

DeepMaker - Backend

- Backend:
  - Efficiently map the optimized DNN architecture to target devices:
    - GPU
    - TPU
    - Intel NCS2 (Movidius)
    - FPGA



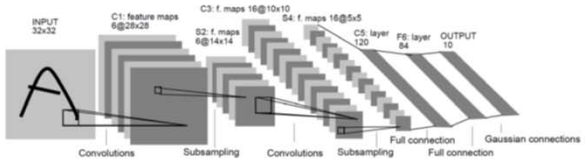
"DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems." Microprocessors and Microsystems (2020).

DeepMaker - Backend

- Acceleration:
  - Running the DNN on the **FPGA or ASIC** to achieve a **higher power performance gains**
  - Gap: DNN design is performed by high-level software/data specialists, whereas the hardware is implemented by low-level hardware ones
  - **HLS** is a promising tool to fill the gap between DNN designers and the programmable logic (FPGA)/ASIC.

- High-level synthesis:
  - Generates a circuit for a design described in high-level languages (mainly C or C++).
  - Easy to use but required to be an expert using directives (without directives, results will be worth than CPU)
  - Multiple architectures (area-latency) through simply adding just a few **directives** (e.g. loop unrolling, pipelining)

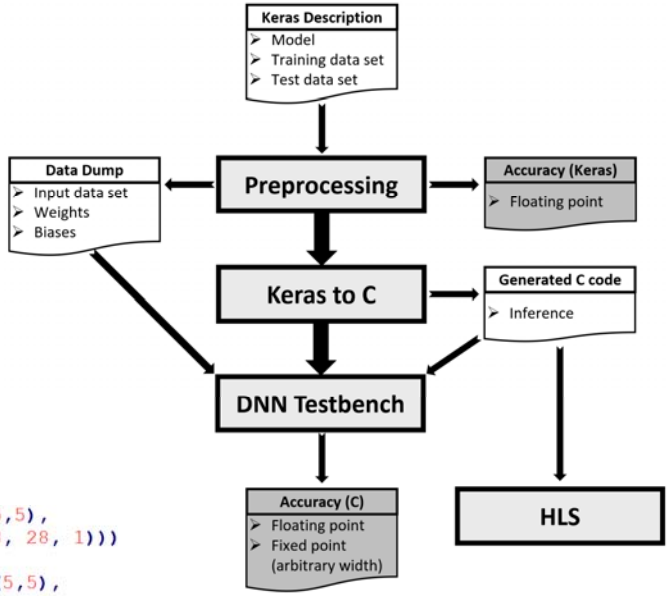

```
model.add(Conv2D(filters=6, kernel_size=(5,5),
        activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D(strides=2))
model.add(Conv2D(filters=16, kernel_size=(5,5),
        activation='relu'))
model.add(MaxPool2D(strides=2))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))
```



DeepMaker - DeepHLS

CNN	BRAM	FF	LUT
VGG	616	7991	21636
AlexNet	326	4200	11257
LeNet-5	16	2348	5763

HLS



Keras Implementation
(e.g. LeNet Network)

```
model.add(Conv2D(filters=6, kernel_size=(5,5),
        activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D(strides=2))
model.add(Conv2D(filters=16, kernel_size=(5,5),
        activation='relu'))
model.add(MaxPool2D(strides=2))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

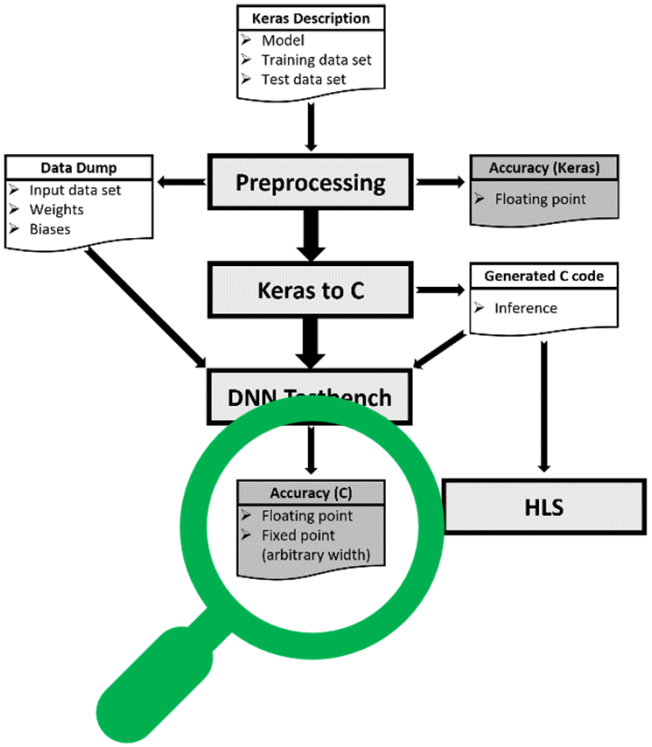
```
//Convolutional Layer
for each output feature
  for each row of the output feature
    for each column of the output feature
      Initialize the cell to zero;

for each input feature
  for each output feature
    for each row of the output feature
      for each column of the output feature
        for each row of the kernel
          for each column of the kernel
            MAC: Respective input cells and weights;

for each output feature
  for each row of the output feature
    for each column of the output feature
      Apply biases and the activation function;

//Pooling layer
for each output feature
  for each row of the output feature
    for each column of the output feature
    {
      for each row of the kernel
        for each column of the kernel
          Find the pooling result
      Assign the pooling result to respective output cell
    }
```

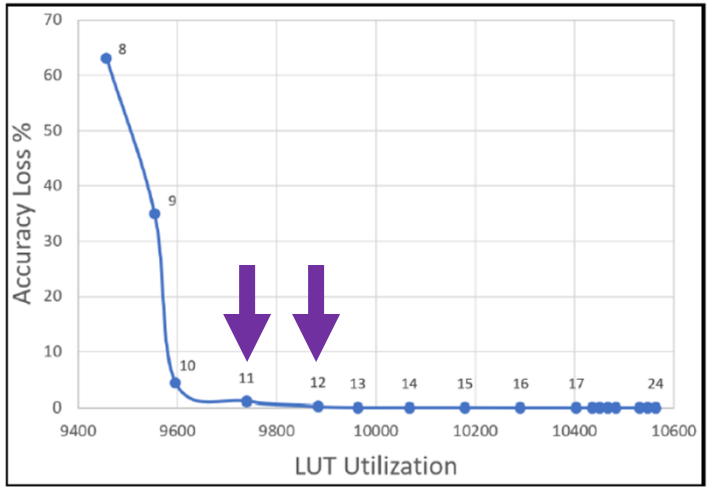
Extraction of the layers' information

Layer	Layer	Padding	Filters	Kernel	Stride	Activation	Input	Output
1	Conv2D	Valid	6	(5, 5)	(1, 1)	Relu	(28, 28, 1)	(24, 24, 6)
2	MaxPool2D	-	-	(2, 2)	(2, 2)	-	(24,24,6)	(12,12,6)
3	Conv2D	Valid	16	(5, 5)	(1, 1)	Relu	(12,12,6)	(8,8,16)
4	MaxPool2D	-	-	(2, 2)	(2, 2)	-	(8,8,16)	(4,4,16)
5	Flatten	-	-	-	-	-	(4,4,16)	(256,1,1)
6	Dense	-	-	-	-	Relu	(256,1,1)	(120,1,1)
7	Dense	-	-	-	-	Relu	(120,1,1)	(84,1,1)
8	Dense	-	-	-	-	Softmax	(84,1,1)	(10,1,1)

Implementation in C / C++

DeepMaker - DeepHLS



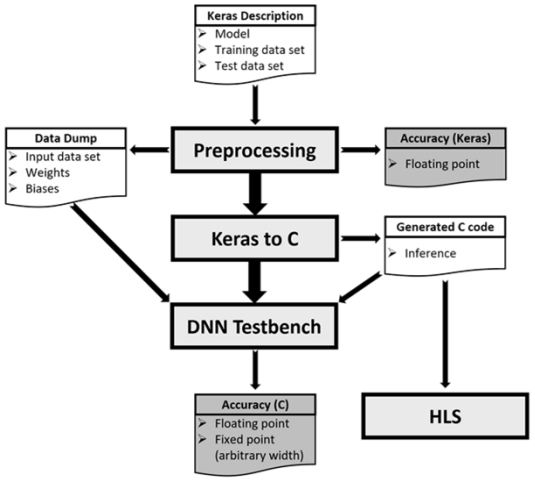


The impact of the fixed-point size on accuracy and resource utilization

(AlexNet)

DeepMaker – DeepHLS

Features summary



A full toolchain including the **Keras to C conversion** and **conversion accuracy verification**.

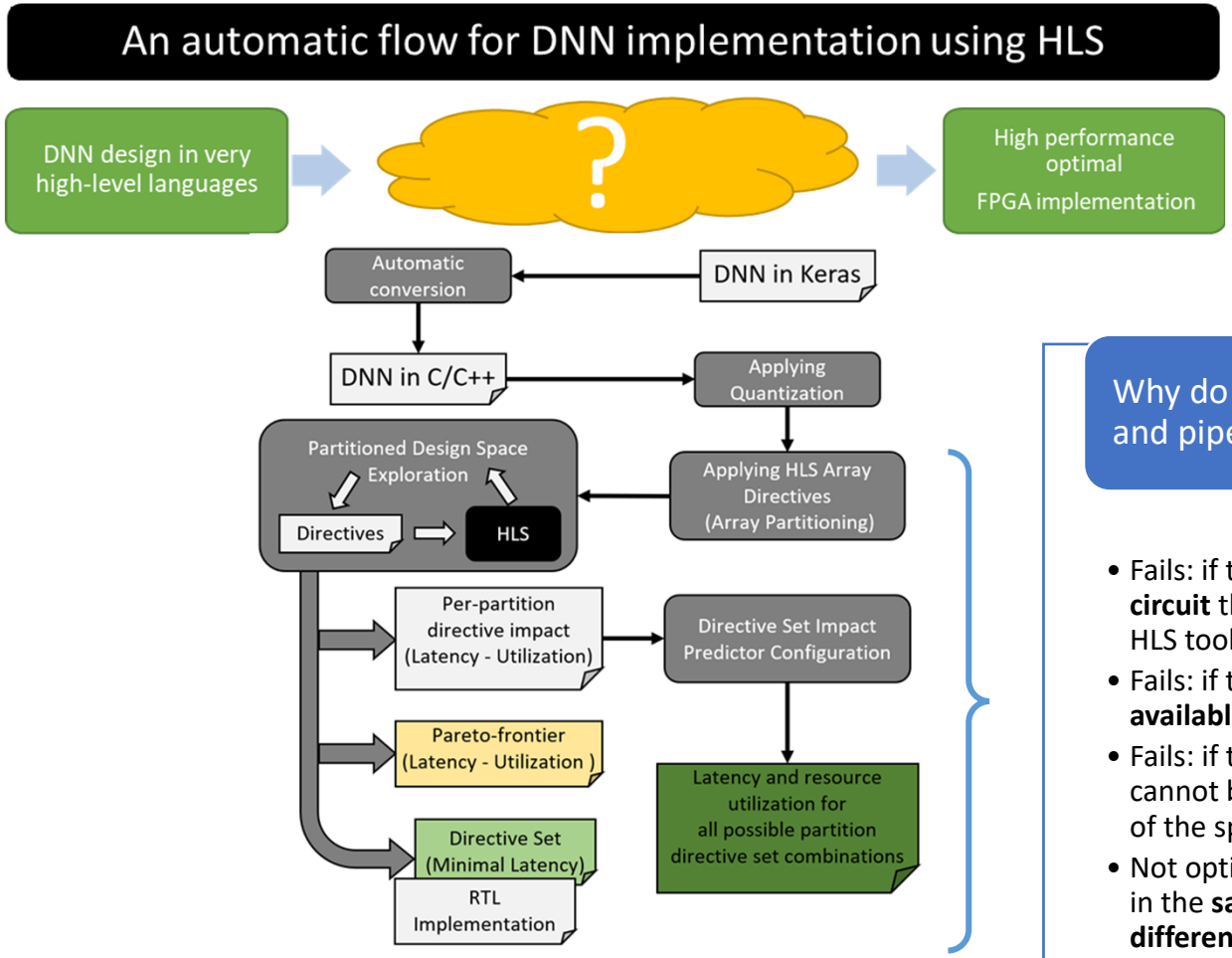
The generated C has easy-to-use **knobs** to switch the HLS implementation from **floating point to fixed point** with arbitrary quantization levels.

The generated C code is in **ANSI C**, and thus, it can be used in almost **all** the open-source and commercial **HLS tools**.

The C code is **flat**, allowing the HLS users to easily **add directives** such as pipeline and unroll in a fine-grained way. This also enables the **intra-layer optimizations**.

For verification and accuracy evaluation, a **file-to-memory interface** is implemented that allows the toolchain to be used for **large networks** with many parameters and large datasets.

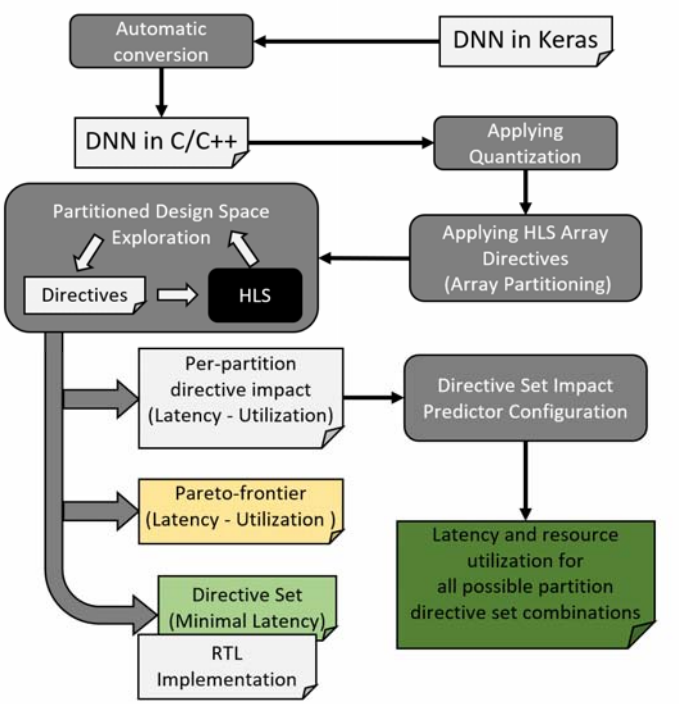
DeepMaker – Backend - Second stage



Why do we need DSE? Why not fully unroll and pipeline every loop?

- Fails: if the unroll factor results in a **very large circuit** that exceeds the processing capacity of the HLS tool for **scheduling and routing**
- Fails: if the required resources are more than the **available resources** on the specified chip
- Fails: if the **concurrent memory accesses** needed cannot be provided due to the limited bandwidth of the specified memory unit
- Not optimal: two different directive set may result in the **same** performance improvement but **different** resource utilization

DeepMaker – Backend - Second stage (step 1)



Array Partitioning

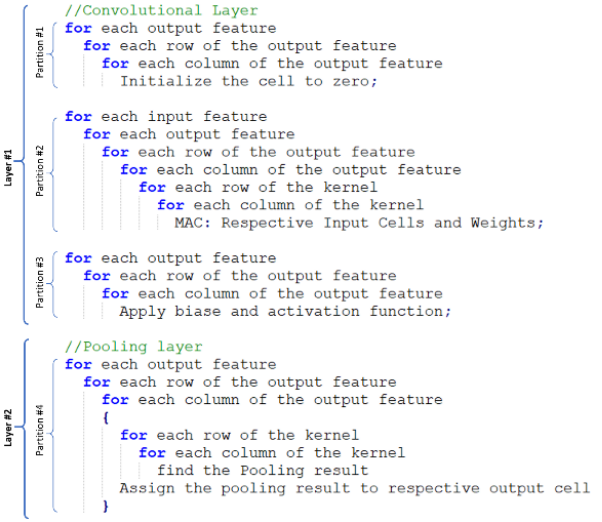
- Is applied to intermediate data elements (layer outputs)
- Helps simultaneous access to multiple elements

DeepMaker – Backend - Second stage (step 2)

Code Partitioning Before DSE:

```
//Convolutional Layer
for each output feature
  for each row of the output feature
    for each column of the output feature
      Initialize the cell to zero;

for each input feature
  for each output feature
    for each row of the output feature
      for each column of the output feature
        for each row of the kernel
          for each column of the kernel
            MAC: Respective Input Cells and Weights;

for each output feature
  for each row of the output feature
    for each column of the output feature
      Apply biase and activation function;

//Pooling layer
for each output feature
  for each row of the output feature
    for each column of the output feature
    {
      for each row of the kernel
        for each column of the kernel
          find the Pooling result
      Assign the pooling result to respective output cell
    }
```

For-loops without parents mark the beginning of a new partition.



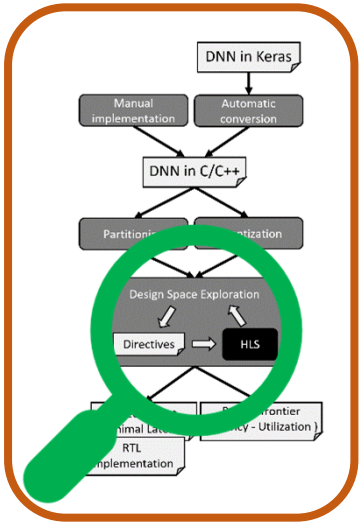
DeepMaker – Backend - Second stage (step 3)

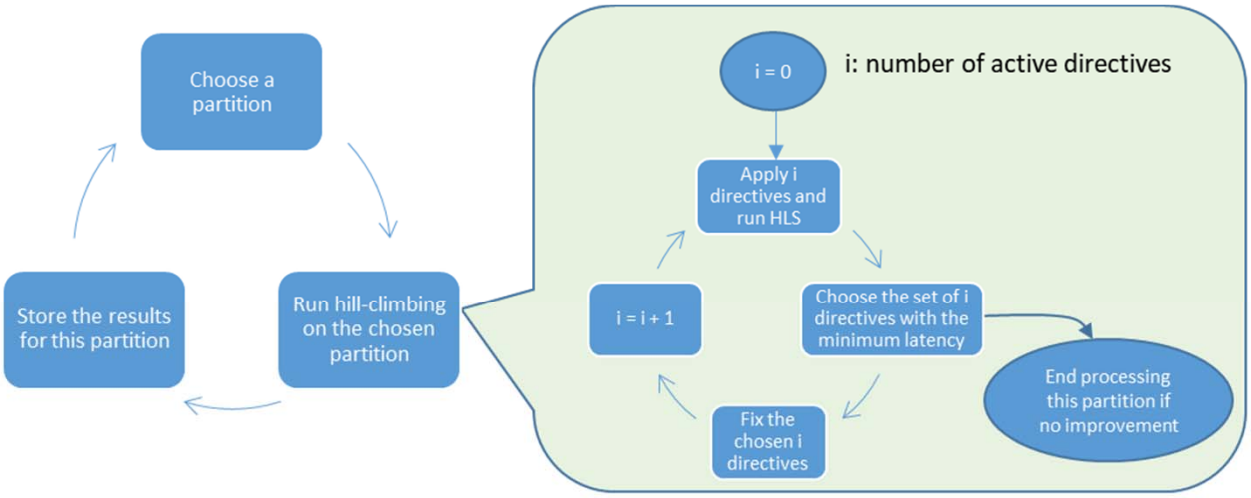
```
//Convolutional Layer
for each output feature
  for each row of the output feature
    for each column of the output feature
      Initialize the cell to zero;

for each input feature
  for each output feature
    for each row of the output feature
      for each column of the output feature
        for each row of the kernel
          for each column of the kernel
            MAC: Respective Input Cells and Weights;

for each output feature
  for each row of the output feature
    for each column of the output feature
      Apply biase and activation function;

//Pooling layer
for each output feature
  for each row of the output feature
    for each column of the output feature
    {
      for each row of the kernel
        for each column of the kernel
          find the Pooling result
      Assign the pooling result to respective output cell
    }
```

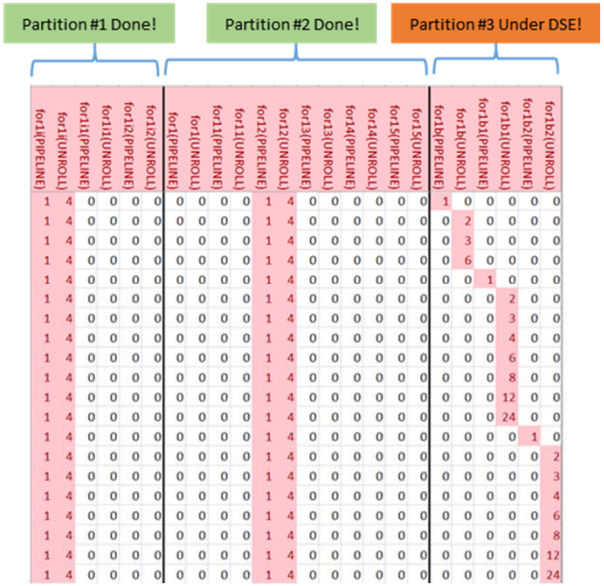
Row	Loop purpose	Nesting level	Loop iterations	Pipeline options	Unroll options	Possibilities
1	Layer Output Initialization	0	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16
2	Layer Output Initialization	1	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16
3	Layer Output Initialization	2	6	No, Yes	1, 2, 3, 6	8
4	Convolution Calculation	0	1	No, Yes	-	1
5	Convolution Calculation	1	6	No, Yes	1, 2, 3, 6	8
6	Convolution Calculation	2	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16
7	Convolution Calculation	3	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16
8	Convolution Calculation	4	5	No, Yes	1, 5	4
9	Convolution Calculation	5	5	No, Yes	1, 5	4
10	Bias and Activation Function	0	6	No, Yes	1, 2, 3, 6	8
11	Bias and Activation Function	1	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16
12	Bias and Activation Function	2	24	No, Yes	1, 2, 3, 4, 6, 8, 12, 24	16

$$(16\times16\times8\times1\times8\times16\times16\times4\times4\times8\times16\times16)$$

More than 137 billion possibilities

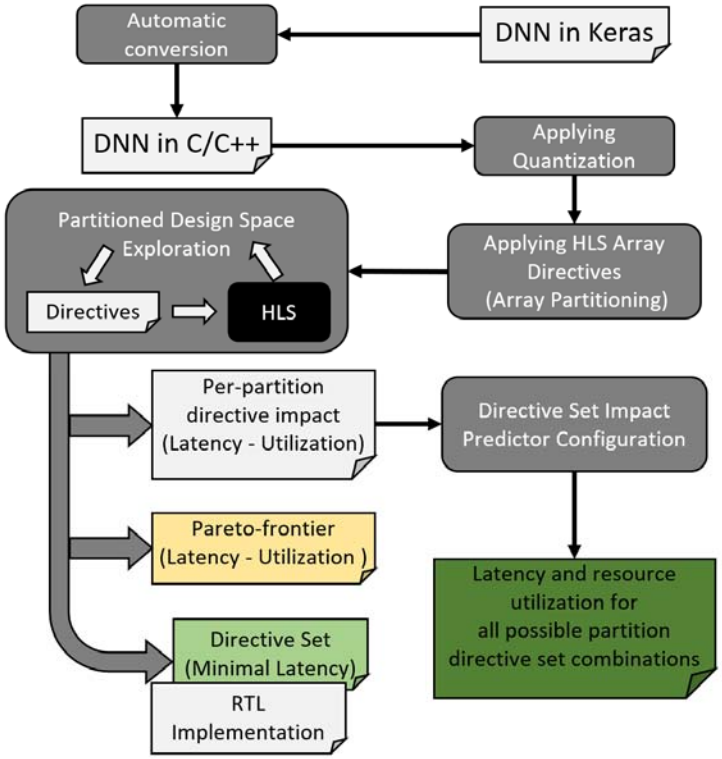




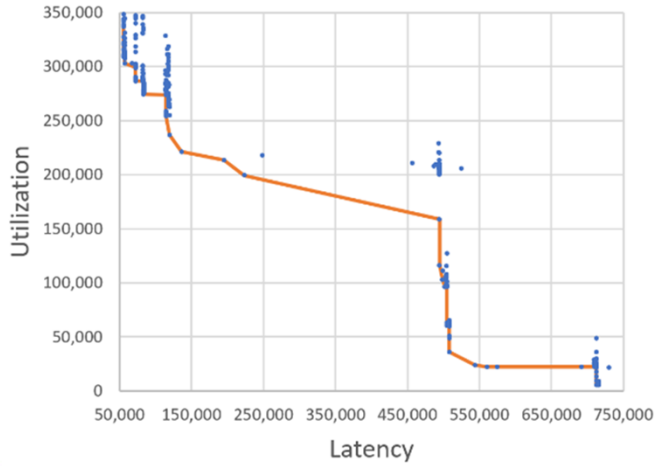


DeepMaker – Backend - Second stage (step 3)

Resource-aware DSE



Explored points and the Pareto-frontier



Pareto-frontier

- Helps designer choose a specific point in the explored space based on the required latency and available resources.

Directive Set (Minimal Latency)

- The point at which all the directives are applied, and the minimum latency is achieved.

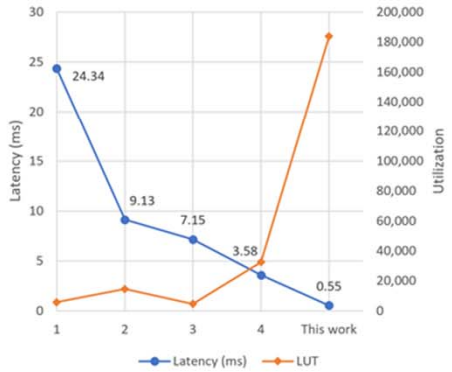
Directive set impact predictor

- Based on the impacts of the directives related to a specific partition
- Is needed to avoid excessive HLS runs, which is the most time-consuming part of the flow

DeepMaker – Backend - Results

Comparison with existing HLS based works and other HLS implementations

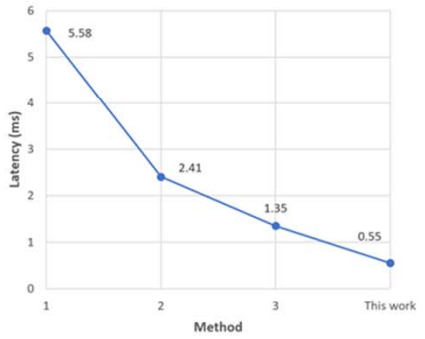
Row	Implementation	Latency (ms)	FF	LUT	Data type
1	No directives (1)	24.34	2,099	5,884	32-bit Floating
2	[13]	9.13	14,172	14,659	32-bit Floating
3	No directives (2)	7.15	1,219	4,651	16-bit Fixed
4	[14]	3.58	33,585	32,589	11-bit Fixed
This work	HLS DSE	0.55	157,927	183,746	16-bit Fixed



6.51x higher performance

Comparison with other implementations

Row	Title	Description	Latency (ms)
1	2018 - An FPGA Based Hardware Accelerator for Classification of Handwritten Digits	Cusom LeNet Implementaion	5.58
2	2016 - From high-level deep neural models to fpgas	DNNWeaver: Template based	2.41
3	2017 - Throughput-optimized fpga accelerator for deep convolutional neural networks	Computation/Memory analysis	1.35
This work	Proposed method		0.55

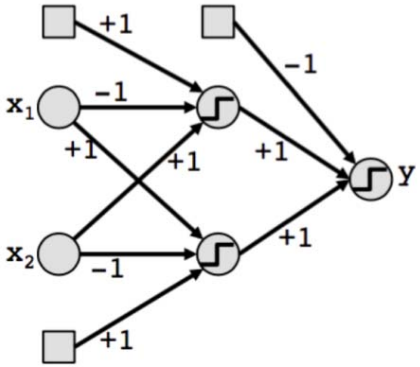


Automatic and easy to use for DNN designers without HW knowledge

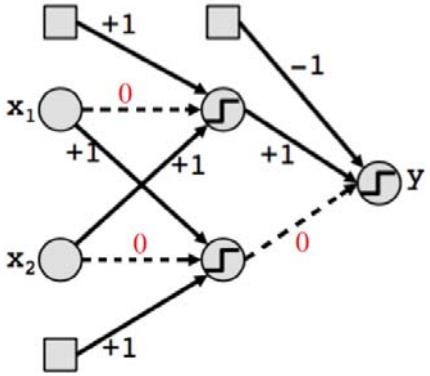
4.5x faster than 2 (DNNWeaver)

DeepMaker – Quantization – Ternary

- Representing the floating-point weights and/or activations with fewer bits
  - Reduces memory footprint
  - Improves computational efficiency
  - Essential for deploying deep models onto IoT devices
- Two Popular types of network quantization
  - **Binary quantization**: 1-bit values {-1, 1}
  - **Ternary quantization** : 2-bit values {-1, 0, 1}



Binary Quantization



Ternary Quantization

N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

DeepMaker – Quantization – Ternary (ToT-NET)

- Comparing different quantization methods:

Ternary neural network (TNN) vs. Binary neural network (BNN)

Criteria	TNN	BNN
Sparsity		
Accuracy		
Energy Consumption		
Memory Footprint		

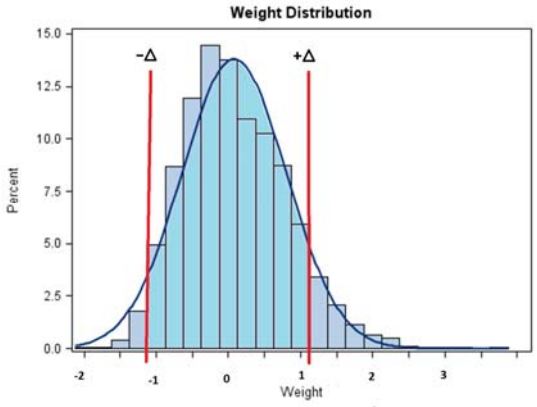
- **Goal:** Increasing the sparsity rate and energy efficiency of TNNs
  - Introduced a novel ternarized training to reduce the accuracy gap compared to floating-point training
  - Proposing a novel multiplier for ternary neural networks to improve network sparsity, energy efficiency, and silicon footprint
  - Proposing novel ternarized activation function to improve model accuracy

N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

DeepMaker – Quantization – Ternary (ToT-NET)

- We propose a novel TNN that ternarizes both weights and activations at training time
- To clip weights into the *[-1, 1]* interval, we use Eq. 1 [10] to obtain the suitable value of both activations and weights depending on their values

$$Wt = \begin{cases} +1 & if\ W \geq \Delta, \\ 0 & if\ |W| \leq \Delta\ , \\ -1 & if\ W \leq -\Delta. \end{cases} \quad \Delta = \frac{0.7}{n} \sum_{i=1}^{n} |X_i| \qquad \textbf{(1)}$$



- Where $\boldsymbol{\Delta}$ is a threshold parameter (the sparsity level), $\boldsymbol{n}$ is the total number of activations or weights in a filter, and $\boldsymbol{X}$ denotes either activations or weights
- Note that, $\boldsymbol{\Delta}$ changes dynamically for each filter and the activation during training
- We eliminate $|\boldsymbol{X}| < \boldsymbol{\Delta}$ values which are ineffective in the computation

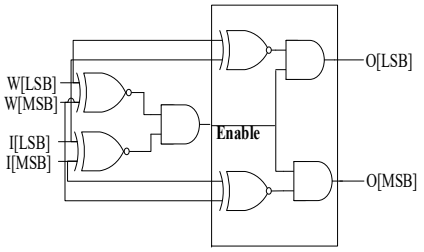
N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

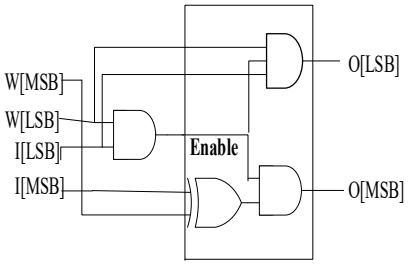
DeepMaker – Quantization – ToT-NET – Mult unit

- We use the standard sign and magnitude encoding to encode ternary values

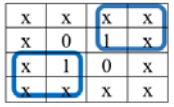
Sign and magnitude encoding

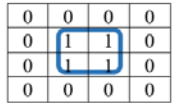
	alt. 1	alt. 2
-1	00	11
0	01 or 10	00 or 10
1	11	01



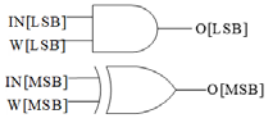


I_{MSB}	I_{LSB}	W_{MSB}	W_{LSB}	O_{MSB}	O_{LSB}
0	0	0	0	x	0
0	0	0	1	x	0
0	0	1	0	x	0
0	0	1	1	x	0
0	1	0	0	x	0
0	1	0	1	0	1
0	1	1	0	x	0
0	1	1	1	1	1
1	0	0	0	x	0
1	0	0	1	x	0
1	0	1	0	x	0
1	0	1	1	x	0
1	1	0	0	x	0
1	1	0	1	1	1
1	1	1	0	x	0
1	1	1	1	0	1





$O_{MSB} = W_{MSB} \oplus I_{MSB}$ $O_{LSB} = W_{LSB}\, I_{LSB}$



- TOT-Net increases sparsity due to adding zero activation

N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

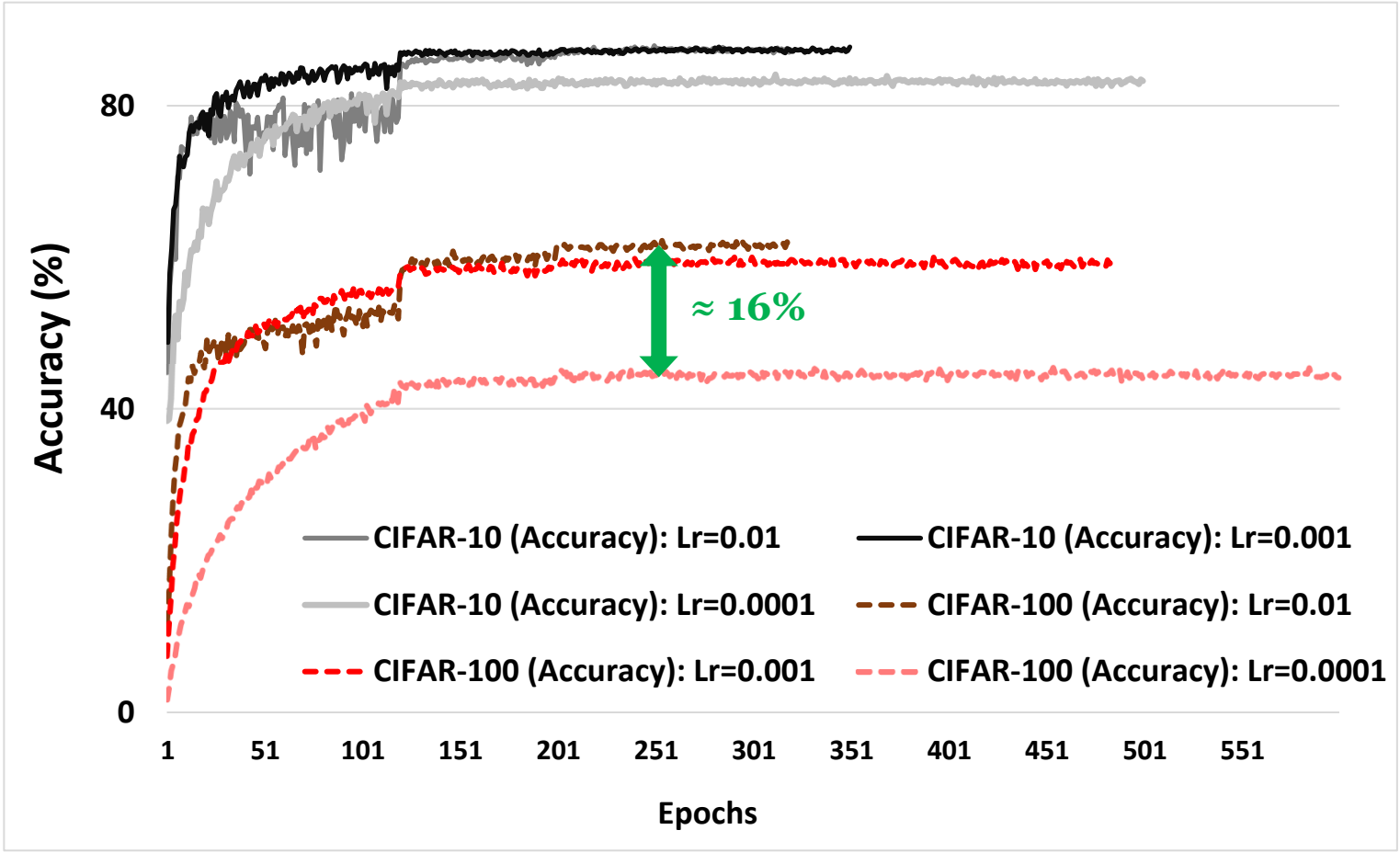
DeepMaker – Quantization – ToT-NET – Results

Method	Accuracy %, (Neural Network Architecture)		
	CIFAR-10	CIFAR-100	ImageNet §
Full-Precision [6]	89.67	64.32	80.2/56.6, (AlexNet) +
BC [9]	NA	NA	61/35.4, (AlexNet) +
BNN [4]	NA	NA	50.4/27.9, (AlexNet) +
XNOR-Net * [6]	85.74, (NIN)	54.10, (NIN)	62.52/37.47, (AlexNet)* +
TOT-Net	**87.53, (NIN)**	**61.61, (NIN)**	**68.20/42.99, (AlexNet) *+**

*** The experiments have been run by us (trained by 20 epochs).**
+ Presenting both top-5 and top-1 accuracy, respectively.

N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

DeepMaker – Quantization – ToT-NET – Learning rate



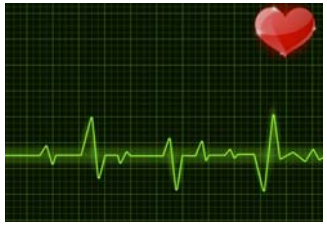
The impact of tweaking learning rate on the TOT-Net accuracy

N. Nazari, M. Loni, M. Salehi, M. Daneshtalab, M. Sjödin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," IEEE Conference on Digital System Design (DSD), 2019.

The DeepMaker Framework (cont.)

- Recurrent Neural Networks
  - Long Short Term Memory (LSTM) is the most popular and effective RNN
  - Sequence learning task

  - Health care application and medical diagnoses:
    - ElectroEncephaloGraphy (EEG),
    - Electrocardiogram (ECG) signal classification,
    - Electromyography (EMG) signal classification


The DeepMaker Framework (cont.), RNN/LSTM (cont.)

- Ternary Compression Method
  - **Ternarize** weights, inputs, and all internal parameters of LSTM cell ternarized to **{-1,0,1}** values
  - Considering the proposed Scaling Factor to reduce accuracy loss
  - In our micro-architecture the multiplication of activation function, and ternarizing function are merged and implemented by couple of XNOR gates
  - The experiments on the ECG and EMG signals show
    - reducing **Latency: ~110×-120×** and **Silicon for ~29×-33×** respectively.
    - Gain **Memory footprint: 17x** and **bandwidth 31×** compared to full precision signal classification.
    - **~0.88-2.04% accuracy loss** in comparison to the full-precision counterparts

- N. Nazari, A. Mirsalari, S. Sinaei, M. Salehi, M. Daneshtalab, "FaCT-LSTM: Fast and Compact Ternary Architecture for LSTM Recurrent Neural Networks" IEEE Design and Test (D&T), 2021.
- N. Nazari, S. A. Mirsalari, S. Sinaei, M. E. Salehi, M. Daneshtalab, "ELC-ECG: Efficient LSTM Cell for ECG Classification based on Quantized Architecture," in Proceedings of 28th IEEE International Symposium on Circuits and Systems (ISCAS), 2021, Spain.

Conclusion

- DeepMaker deals with
  - Optimizing DNN architectures with NAS considering the following criteria
    - Complexity
    - Accuracy
  - Currently looking for hardware-aware NAS
    - Complexity
    - Accuracy
    - Energy and resource utilization
    - Robustness
  - DeepHLS and DSE-HLS for finding the best possible directive sets
    - Using predictive models to speedup the process
  - Ternery NN, looking for optimal boundaries for -1, 0, 1
- Find papers in googlescholar (Masoud Daneshtalab) or my MDH homepage for preprints

Copyright Notice

www.tinyML.org