tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

“Neural Architecture Search for Tiny Devices”

Swarnava Dey – Senior Scientist, TCS Research

April 10, 2023

TINY ML

www.tinyML.org

TINY ML TALKS webcast

Thank you, **tinyML Strategic Partners**, for committing to take tinyML to the next Level, together

AIZIP ANALOG DEVICES AHEAD OF WHAT'S POSSIBLE™ ARDUINO arm COMPASS MARKETING

EDGE IMPULSE GREENWAVES TECHNOLOGIES Grovety Inc. IBM imagimob

innatera Microsoft Nota AI NXP OctoML POLYN TECHNOLOGY

Qeexo Qualcomm RENESAS Schneider Electric SensiML SONY

SILICON LABS ST life.augmented synaptics SYNTIANT TDK InvenSense

Executive Strategic Partners

TINY ML TALKS webcast

EDGE IMPULSE

The Leading Development Platform for Edge ML

edgeimpulse.com

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

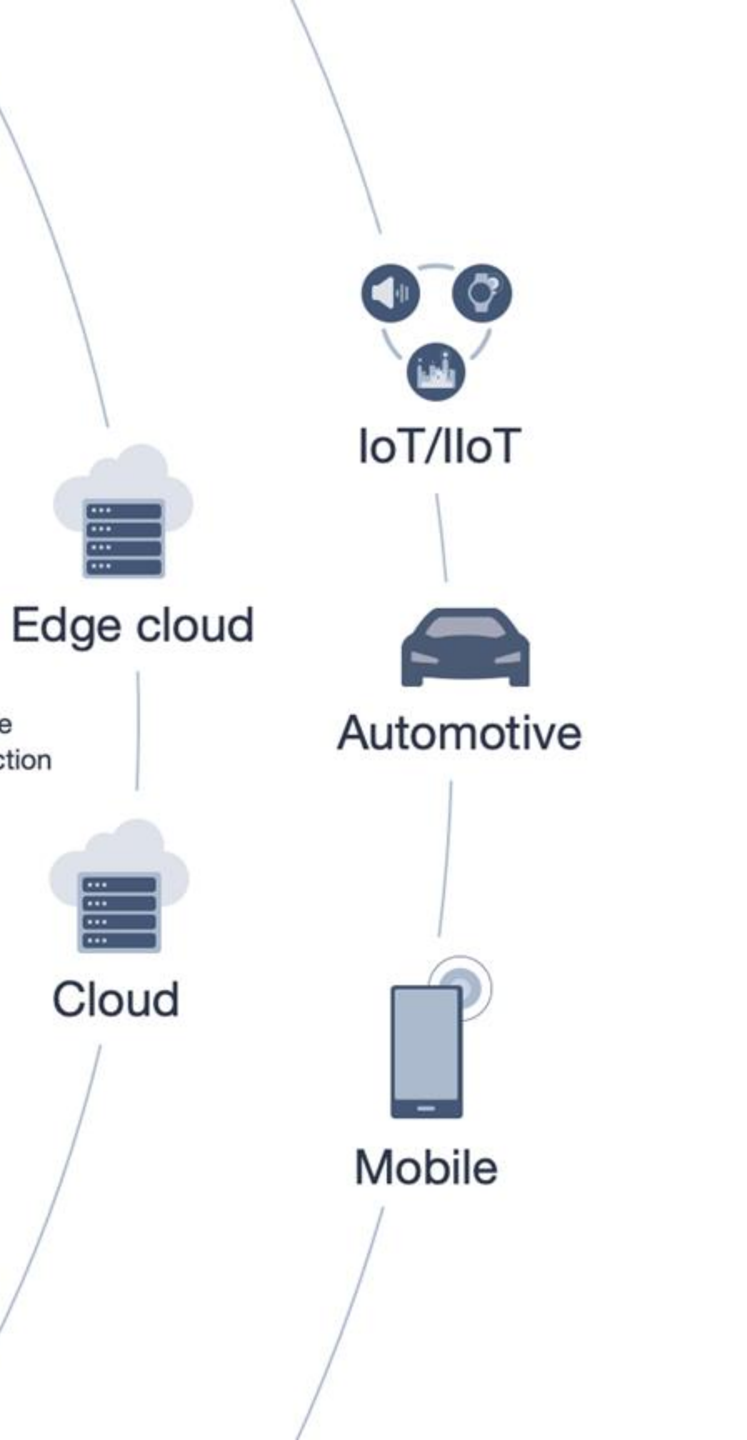
Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception
Object detection, speech recognition, contextual fusion

Reasoning
Scene understanding, language understanding, behavior prediction

Action
Reinforcement learning for decision making



Accelerate Your Edge Compute

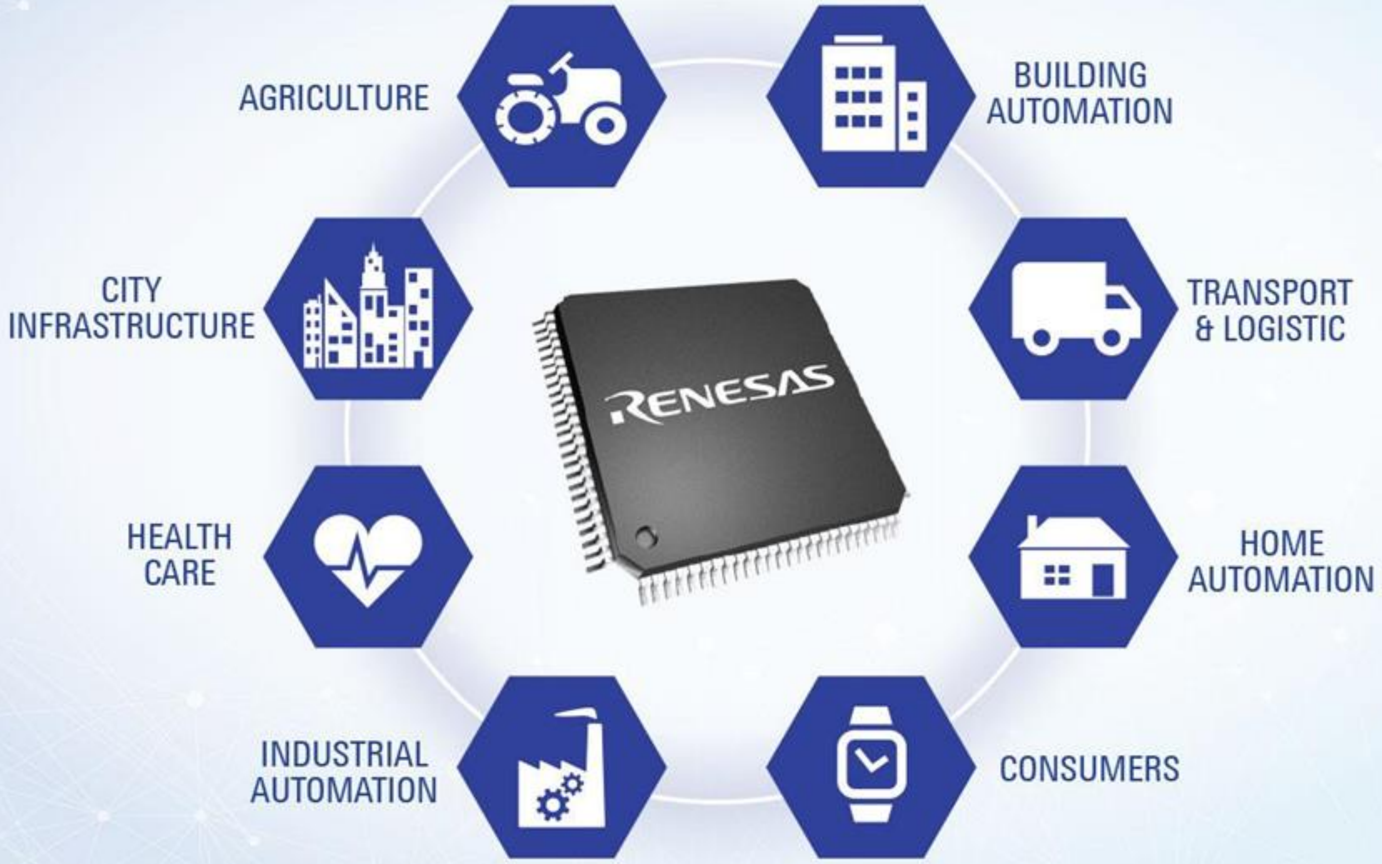
SYNTIANT

Making Edge AI A Reality

www.syntiant.com

Platinum Strategic Partners

Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.



renesas.com

RENESAS
BIG IDEAS FOR EVERY SPACE

AITRIOS

DEPLOY VISION AI AT THE EDGE AT SCALE

SONY

Gold Strategic Partners

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Where what if becomes what is.

Witness potential made possible at analog.com.

ARDUINO PRO™

Easily deploy your tinyML solutions with Arduino Pro

arduino.cc/pro

Made In Italy

arm AI

Powering tinyML Innovation

Arm AI Virtual Tech Talks

The latest in AI trends, technologies & best practices from Arm and our Ecosystem Partners.

Demos, code examples, workshops, panel sessions and much more!

Fortnightly Tuesday @ 4pm GMT/8am PT

Find out more:
www.arm.com/techtalks

innatera

NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE

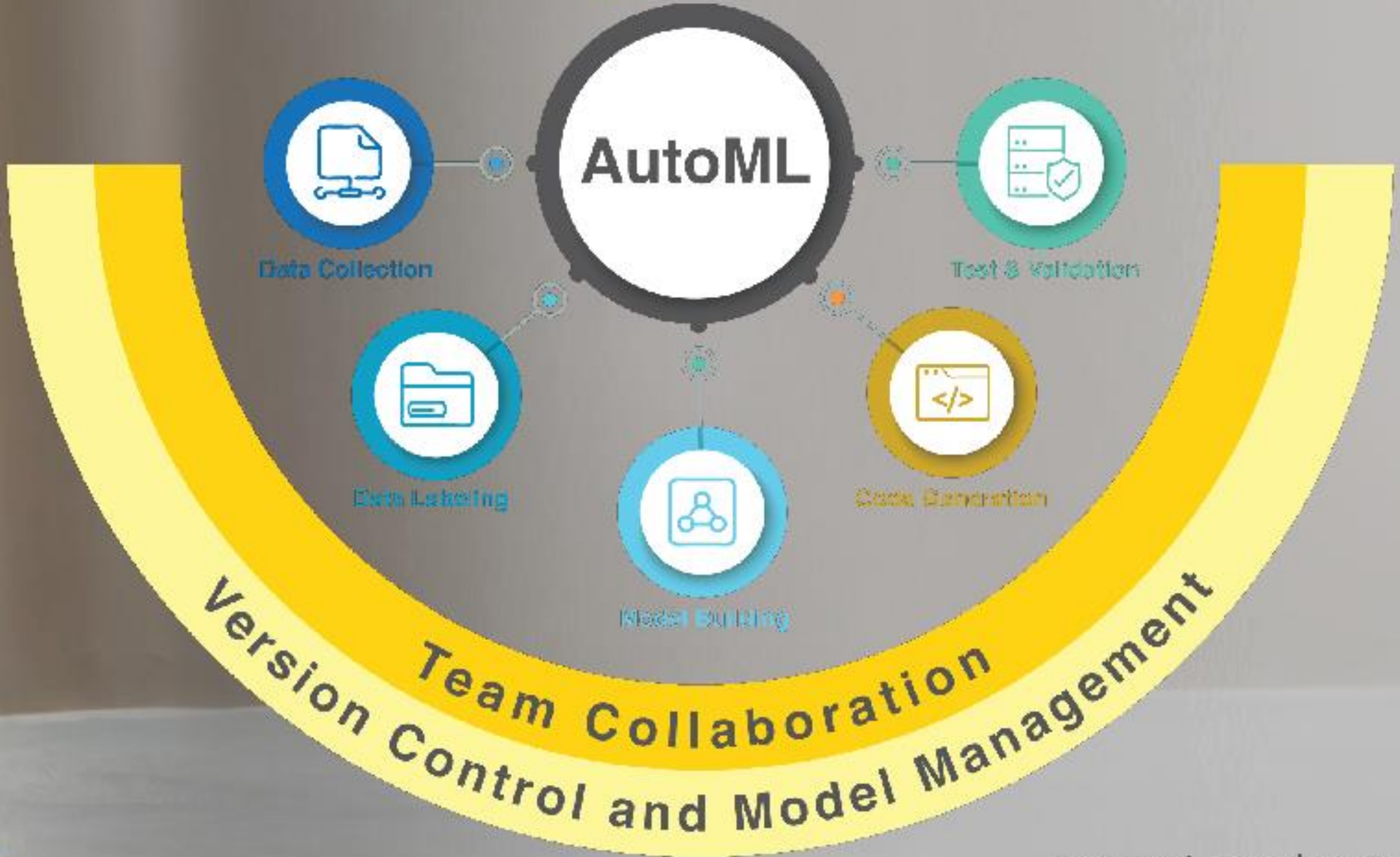
www.innatera.com

Microsoft

The Right Edge AI Tools Can Make or Break Your Next Smart IoT Product

SensiML

Analytics Toolkit Suite



sensiml.com/tinyML

STMicroelectronics provides extensive solutions to make tiny Machine Learning easy

life.augmented

www.st.com/ai

ENGINEERING EXCEPTIONAL EXPERIENCES

We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com

TINY ML TALKS webcast

Silver Strategic Partners

AIZIP

COMPASS MARKETING

GREENWAVES TECHNOLOGIES

Grovety Inc.

IBM

imagimob

Nota AI

NXP

OctoML

POLYN TECHNOLOGY NEUROMORPHIC ANALOG SIGNAL PROCESSING

Qeexo

Schneider Electric

SILICON LABS

TDK InvenSense

Join Growing tinyML Communities:

13.9k members in
47 Groups in 39 Countries

tinyML - Enabling ultra-low Power ML at the Edge

https://www.meetup.com/tinyML-Enabling-ultra-low-Power-ML-at-the-Edge/

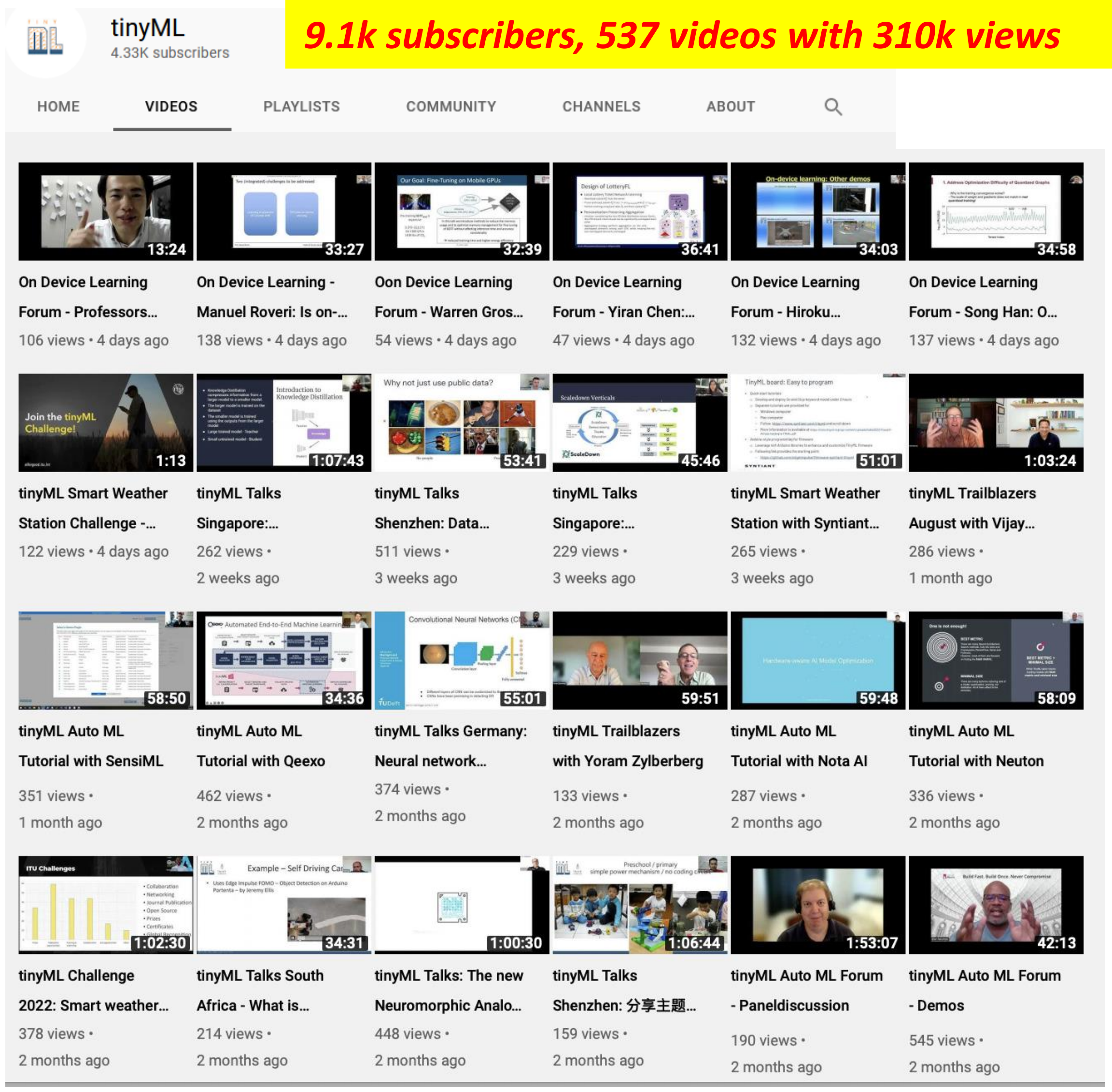
4k members
&
11.6k followers

The tinyML Community

https://www.linkedin.com/groups/13694488/

Subscribe to
tinyML YouTube Channel
for updates and notifications
(including this video)
www.youtube.com/tinyML

9.1k subscribers, 537 videos with 310k views


tinyML
4.33K subscribers

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

13:24 On Device Learning Forum - Professors... 106 views • 4 days ago	33:27 On Device Learning - Manuel Roveri: Is on-... 138 views • 4 days ago	32:39 Oon Device Learning Forum - Warren Gros... 54 views • 4 days ago	36:41 On Device Learning Forum - Yiran Chen:... 47 views • 4 days ago	34:03 On Device Learning Forum - Hiroku... 132 views • 4 days ago	34:58 On Device Learning Forum - Song Han: O... 137 views • 4 days ago
1:13 tinyML Smart Weather Station Challenge -... 122 views • 4 days ago	1:07:43 tinyML Talks Singapore:... 262 views • 2 weeks ago	53:41 tinyML Talks Shenzhen: Data... 511 views • 3 weeks ago	45:46 tinyML Talks Singapore:... 229 views • 3 weeks ago	51:01 tinyML Smart Weather Station with Syntiant... 265 views • 3 weeks ago	1:03:24 tinyML Trailblazers August with Vijay... 286 views • 1 month ago
58:50 tinyML Auto ML Tutorial with SensiML 351 views • 1 month ago	34:36 tinyML Auto ML Tutorial with Qeexo 462 views • 2 months ago	55:01 tinyML Talks Germany: Neural network... 374 views • 2 months ago	59:51 tinyML Trailblazers with Yoram Zylberberg 133 views • 2 months ago	59:48 tinyML Auto ML Tutorial with Nota AI 287 views • 2 months ago	58:09 tinyML Auto ML Tutorial with Neuton 336 views • 2 months ago
1:02:30 tinyML Challenge 2022: Smart weather... 378 views • 2 months ago	34:31 tinyML Talks South Africa - What is... 214 views • 2 months ago	1:00:30 tinyML Talks: The new Neuromorphic Analo... 448 views • 2 months ago	1:06:44 tinyML Talks Shenzhen: 分享主题... 159 views • 2 months ago	1:53:07 tinyML Auto ML Forum - Paneldiscussion 190 views • 2 months ago	42:13 tinyML Auto ML Forum - Demos 545 views • 2 months ago

EMEA 2023

https://www.tinyml.org/event/emea-2023

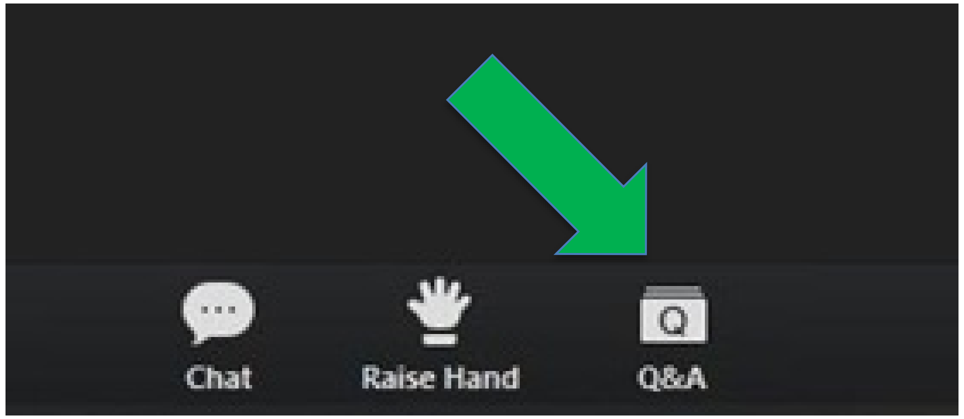
More sponsorships are available: sponsorships@tinyML.org

Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums youtube.com/tinyml

Please use the Q&A window for your questions



Swarnava Dey

Swarnava Dey is a Senior Scientist at TCS Research working on embedded vision systems. He is an M.Tech from IIT, Kharagpur, and currently pursuing PhD there in robustness, verifiability and explainability of Embedded Deep Neural Networks and Neuro Symbolic AI. He has 30+ granted patents, 25+ research papers, and is an author of Towards Data Science: https://medium.com/@qswadey. His publication details can be found at his Google Scholar page:https://scholar.google.co.in/citations?hl=en&user=aFplwjEAAAAJ

TATA CONSULTANCY SERVICES

Building on belief

Neural Architecture Search for Tiny Devices

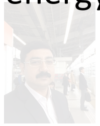
A Hardcore Technical Tutorial
In tinyML Talks, 10th April 2023

Swarnava Dey

Neural Architecture Search (~AutoML): Goals

- NAS research - Automatically generate better architectures than handcrafted models; benchmark accuracy on NAS Bench dataset & ImageNet

- NAS for TinyML - Automatically customize & optimize DNNs for multiple constraints - [accuracy, model size, SRAM usage (runtime memory), #MACs, latency, energy usage…]



Research

Inventing for impact

Neural Architecture Search (~AutoML): Goals

- NAS research - Automatically generate better architectures than handcrafted models; benchmark accuracy on NAS Bench dataset & ImageNet
- NAS for TinyML - Automatically customize & optimize DNNs for multiple constraints - [accuracy, model size, SRAM usage (runtime memory), #MACs, latency, energy usage…]

👂 NAS SOTA - Differential, one-shot NAS - DARTS & DARTS-based

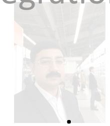
- ✔ DARTS works out of the box - https://github.com/quark0/darts
- ✔ Default implementation single-objective - accuracy
- ✔ Difficult to get ready version that allows integration of my preferred objectives

Neural Architecture Search (~AutoML): Goals

- NAS research - Automatically generate better architectures than handcrafted models; benchmark accuracy on NAS Bench dataset & ImageNet
- NAS for TinyML - Automatically customize & optimize DNNs for multiple constraints - [accuracy, model size, SRAM usage (runtime memory), #MACs, latency, energy usage…]

👂 NAS SOTA - Differential, one-shot NAS - DARTS & DARTS-based

- ✔ DARTS works out of the box - https://github.com/quark0/darts
- ✔ Default implementation single-objective - accuracy
- ✔ Difficult to get ready version that allows integration of my preferred objectives



👂 Heuristic / Sample based NAS are obsolete

- ✔ For searching tiny models the overhead is marginalized
- ✔ Easy to integrate preferred objectives

tcs Research

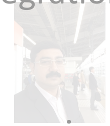
Inventing for impact

Neural Architecture Search (~AutoML): Goals

- NAS research - Automatically generate better architectures than handcrafted models; benchmark accuracy on NAS Bench dataset & ImageNet
- NAS for TinyML - Automatically customize & optimize DNNs for multiple constraints - [accuracy, model size, SRAM usage (runtime memory), #MACs, latency, energy usage…]

NAS SOTA - Differential, one-shot NAS - DARTS & DARTS-based

- ✔ DARTS works out of the box - https://github.com/quark0/darts
- ✔ Default implementation single-objective - accuracy
- ✔ Difficult to get ready version that allows integration of my preferred objectives



Heuristic / Sample based NAS are obsolete

- ✔ For searching tiny models the overhead is marginalized
- ✔ Easy to integrate preferred objectives

NAS for TinyML

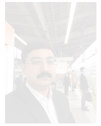
- ✔ Large many-layered networks, complex connections not required
- ✔ Accurate multi-objective conformance, platform API support - highly required
- ✔ MCUNet V1 & V2, µ-NAS, Micronets… have their merits & demerits. We may often need to tweak existing frameworks - understanding NAS helps

Today’s Agenda

- 📌 SOTA NAS research & TinyML
- 📌 A naive method to generate DNN
- 📌 Enhancing the method using Reinforcement Learning
- 📌 Other optimization techniques for sample-based NAS
- 📌 Gradient-based, one-shot NAS
- 📌 Take-home points

NAS Research & TinyML: Mainstream NAS Goals

🚩 Started with RL using RNN[1] , Q-Learning[2] to generate architecture

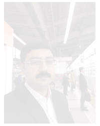


[1] Neural Architecture Search with Reinforcement Learning, Barret Zoph, Quoc V. Le, https://arxiv.org/abs/1611.01578

[2] Designing Neural Network Architectures using Reinforcement Learning, Bowen Baker *et al.*, https://arxiv.org/abs/1611.02167

NAS Research & TinyML: Mainstream NAS Goals

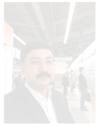
- Started with RL using RNN[1] , Q-Learning[2] to generate architecture
- Next two big innovations were cellular search space[3] and parameter sharing[4]



[1] **Neural Architecture Search with Reinforcement Learning,** Barret Zoph, Quoc V. Le, https://arxiv.org/abs/1611.01578
[2] **Designing Neural Network Architectures using Reinforcement Learning**, Bowen Baker *et al.*, https://arxiv.org/abs/1611.02167
[3] **Learning Transferable Architectures for Scalable Image Recognition,** Barret Zoph *et al.*, https://arxiv.org/abs/1707.07012
[4] **Efficient Neural Architecture Search via Parameter Sharing**, Hieu Pham *et al.*, https://arxiv.org/pdf/1802.03268.pdf

NAS Research & TinyML: Mainstream NAS Goals

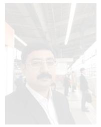
- Started with RL using RNN[1] , Q-Learning[2] to generate architecture
- Next two big innovations were cellular search space[3] and parameter sharing[4]
- Meanwhile Evolutionary[5], Bayesian NAS[6] pushed the NAS optimization SOTA



[1] **Neural Architecture Search with Reinforcement Learning,** Barret Zoph, Quoc V. Le, https://arxiv.org/abs/1611.01578
[2] **Designing Neural Network Architectures using Reinforcement Learning**, Bowen Baker *et al.*, https://arxiv.org/abs/1611.02167
[3] **Learning Transferable Architectures for Scalable Image Recognition,** Barret Zoph *et al.*, https://arxiv.org/abs/1707.07012
[4] **Efficient Neural Architecture Search via Parameter Sharing**, Hieu Pham *et al.*, https://arxiv.org/pdf/1802.03268.pdf
[5] **Regularized Evolution for Image Classifier Architecture Search**,Esteban Real *et al.*
[6] **Parallelised Bayesian Optimisation via Thompson Sampling,** Kirthevasan Kandasamy *et al.*

NAS Research & TinyML: Mainstream NAS Goals

- 🚩 Started with RL using RNN[1] , Q-Learning[2] to generate architecture
- 🚩 Next two big innovations were cellular search space[3] and parameter sharing[4]
- 🚩 Meanwhile Evolutionary[5], Bayesian NAS[6] pushed the NAS optimization SOTA
- 🚩 Next came the game changer DARTS[7]. It used earlier innovations.



[1] **Neural Architecture Search with Reinforcement Learning,** Barret Zoph, Quoc V. Le, https://arxiv.org/abs/1611.01578
[2] **Designing Neural Network Architectures using Reinforcement Learning**, Bowen Baker *et al.*, https://arxiv.org/abs/1611.02167
[3] **Learning Transferable Architectures for Scalable Image Recognition,** Barret Zoph *et al.*, https://arxiv.org/abs/1707.07012
[4] **Efficient Neural Architecture Search via Parameter Sharing**, Hieu Pham *et al.*, https://arxiv.org/pdf/1802.03268.pdf
[5] **Regularized Evolution for Image Classifier Architecture Search**,Esteban Real *et al.*
[6] **Parallelised Bayesian Optimisation via Thompson Sampling,** Kirthevasan Kandasamy *et al.*
[7] **DARTS: Differentiable Architecture Search**, Hanxiao Liu *et al.*, https://arxiv.org/abs/1806.09055

NAS Research & TinyML: Mainstream NAS Goals

- Started with RL using RNN[1] , Q-Learning[2] to generate architecture
- Next two big innovations were cellular search space[3] and parameter sharing[4]
- Meanwhile Evolutionary[6], Bayesian NAS[7] pushed the NAS optimization SOTA
- Next came the game changer DARTS[5]. It used earlier innovations.
- All paved the way for searching larger, more accurate nets much faster
- I had to drop many deserving papers due to space constraints
- This goal is not important for TinyML applications.
- However, let's remember **2-**MetaQNN**, 3-**NASNet**, 4-**ENAS**, 5-**DARTS**, 6-**AE**, & 7-**BO

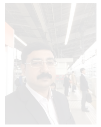
[1] **Neural Architecture Search with Reinforcement Learning,** Barret Zoph, Quoc V. Le, https://arxiv.org/abs/1611.01578
[2] **Designing Neural Network Architectures using Reinforcement Learning**, Bowen Baker *et al.*, https://arxiv.org/abs/1611.02167
[3] **Learning Transferable Architectures for Scalable Image Recognition,** Barret Zoph *et al.*, https://arxiv.org/abs/1707.07012
[4] **Efficient Neural Architecture Search via Parameter Sharing**, Hieu Pham *et al.*, https://arxiv.org/pdf/1802.03268.pdf
[5] **Regularized Evolution for Image Classifier Architecture Search**,Esteban Real *et al.*
[6] **Parallelised Bayesian Optimisation via Thompson Sampling,** Kirthevasan Kandasamy *et al.*
[7] **DARTS: Differentiable Architecture Search**, Hanxiao Liu *et al.*, https://arxiv.org/abs/1806.09055

tcs Research

Inventing for impact

NAS Research & TinyML: TinyML Goals

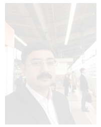
🚩 1000-layer DNN for MCUs ? - 20-30 layer DNN fits ; Global search space good enough; complex, unexpected interconnect[8] OK ; Training smaller models marginalizes many problems



[8] **SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures,** Cheng *et al.*

NAS Research & TinyML: TinyML Goals

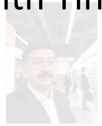
- 🚩 1000-layer DNN for MCUs ? - 20-30 layer DNN fits ; Global search space good enough; complex, unexpected interconnect[8] OK ; Training smaller models marginalizes many problems
- 🚩 For TinyML model size, peak RAM usage, #MAC are highly important
- 🚩 Evaluation, measurement of KPIs, reward engineering are important



[8] **SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures,** Cheng *et al.*

NAS Research & TinyML: TinyML Goals

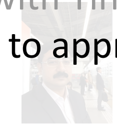
- 1000-layer DNN for MCUs ? - 20-30 layer DNN fits ; Global search space good enough; complex, unexpected interconnect[8] OK ; Training smaller models marginalizes many problems
- For TinyML model size, peak RAM usage, #MAC are highly important
- Evaluation, measurement of KPIs, reward engineering are important
- MCUNet v1[9] & v2[10] are top class works - based on one shot NAS[11], Search Space reduction highest FLOPs within SRAM constraint - closely tied with TinyEngine IR for HW params



[8] **SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures**, Cheng *et al.*
[9] **MCUNet: Tiny Deep Learning on IoT Devices**, Ji Lin *et al.*, https://arxiv.org/pdf/2007.10319
[10] **MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning**, Ji Lin *et al.*, https://arxiv.org/abs/2110.15352
[11] **Understanding and Simplifying One-Shot Architecture Search**, Bender *et al.*, https://proceedings.mlr.press/v80/bender 18a

NAS Research & TinyML: TinyML Goals

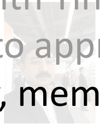
- 1000-layer DNN for MCUs ? - 20-30 layer DNN fits ; Global search space good enough; complex, unexpected interconnect[8] OK ; Training smaller models marginalizes many problems
- For TinyML model size, peak RAM usage, #MAC are highly important
- Evaluation, measurement of KPIs, reward engineering are important
- MCUNet v1[9] & v2[10] are top class works - based on one shot NAS[11], Search Space reduction highest FLOPs within SRAM constraint - closely tied with TinyEngine IR for HW params
- µ-NAS[12] based on AE & BO, use working sets to approximate PMU & #MAC for latency



[8] SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures, Cheng *et al.*
[9] MCUNet: Tiny Deep Learning on IoT Devices, Ji Lin *et al.*, https://arxiv.org/pdf/2007.10319
[10] MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning, Ji Lin *et al.*, https://arxiv.org/abs/2110.15352
[11] Understanding and Simplifying One-Shot Architecture Search, Bender *et al.*, https://proceedings.mlr.press/v80/bender 18a
[12] µNAS: Constrained Neural Architecture Search for Microcontrollers, Edgar Liberis *et al.*, https://arxiv.org/pdf/2010.14246

NAS Research & TinyML: TinyML Goals

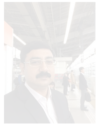
- 1000-layer DNN for MCUs ? - 20-30 layer DNN fits ; Global search space good enough; complex, unexpected interconnect[8] OK ; Training smaller models marginalizes many problems
- For TinyML model size, peak RAM usage, #MAC are highly important
- Evaluation, measurement of KPIs, reward engineering are important
- MCUNet v1[9] & v2[10] are top class works - based on one shot NAS[11], Search Space reduction highest FLOPs within SRAM constraint - closely tied with TinyEngine IR for HW params
- μ-NAS[12] based on AE & BO, use working sets to approximate PMU & #MAC for latency
- MicroNets[13] most promising: DARTS + latency, memory, energy - all approximated by #FLOPS, MO optimization DARTS Loss + 'some regularizer' - not clear: https://github.com/liyunsheng13/micronet
- Moreover, this CVPR '20 paper showed that #MAC not good proxy for latency



[10] MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning, Ji Lin *et al.*, https://arxiv.org/abs/2110.15352
[11] Understanding and Simplifying One-Shot Architecture Search, Bender *et al.*, https://proceedings.mlr.press/v80/bender 18a
[12] μNAS: Constrained Neural Architecture Search for Microcontrollers, Edgar Liberis *et al.*, https://arxiv.org/pdf/2010.14246
[13] MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers, Colby Banbury *et al.*, https://arxiv.org/abs/2010.11267
[14] Latency-Aware Differentiable Neural Architecture Search, Yuhui Xu *et al.*, https://arxiv.org/abs/2001.06392

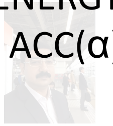
NAS Research & TinyML: Reward Engineering

- 🚩 Generally, for NAS reward is ACC - proven on benchmark datasets
- 🚩 TinyML objective: Pareto Optimal - one objective can’t be improved without making another worse
- 🚩 argmin $\alpha \in A \{ 1.0 - ACC(\alpha), SIZE(\alpha), PMU(\alpha), LAT(\alpha) \}$
- 🚩 Easiest method - treat all other objectives as constraints (comes from platform)



NAS Research & TinyML: Reward Engineering

- Generally, for NAS reward is ACC - proven on benchmark datasets
- TinyML objective: Pareto Optimal - one objective can’t be improved without making another worse
- argmin $\alpha \in A \{ 1.0 - ACC(\alpha), SIZE(\alpha), PMU(\alpha), LAT(\alpha) \}$
- Easiest method - treat all other objectives as constraints (comes from platform)
- DPP-Net[15] - ACC, #params, inference time, memory usage on actual hardware
- MONAS[16] - scalarization: $R = \alpha * ACC - (1-\alpha) * ENERGY$
- μ-NAS[12] - scalarization: $L^t(\alpha) = \max(\lambda^t_1 (1.0 - ACC(\alpha)), \lambda^t_2 PMU(\alpha), \lambda^t_3 SIZE(\alpha), \lambda^t_4 MACS(\alpha))$

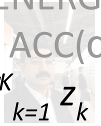


[15] DPP-Net: Device-aware Progressive Search for Pareto-optimal Neural Architectures, Jin-Dong Dong *et al.*, https://arxiv.org/abs/1806.08198

[16] MONAS: Multi-Objective Neural Architecture Search using Reinforcement Learning, Chi-Hung Hsu *et al.*, https://arxiv.org/abs/1806.10332

NAS Research & TinyML: Reward Engineering

- Generally, for NAS reward is ACC - proven on benchmark datasets
- TinyML objective: Pareto Optimal - one objective can’t be improved without making another worse
- argmin $\alpha \in A$ { $1.0 - ACC(\alpha)$, $SIZE(\alpha)$, $PMU(\alpha)$, $LAT(\alpha)$ }
- Easiest method - treat all other objectives as constraints (comes from platform)
- DPP-Net[15] - ACC, #params, inference time, memory usage on actual hardware
- MONAS[16] - scalarization: $R = \alpha * ACC - (1-\alpha) * ENERGY$
- µ-NAS[12] - scalarization: $L^t(\alpha) = \max(\lambda^t_1 (1.0 - ACC(\alpha)), \lambda^t_2 PMU(\alpha), \lambda^t_3 SIZE(\alpha), \lambda^t_4 MACS(\alpha))$
- MicroNets[13] - using regularizer with DARTS: $\Sigma^K_{k=1} z_k |\theta_k|$ - how ?

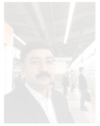


[15] DPP-Net: Device-aware Progressive Search for Pareto-optimal Neural Architectures, Jin-Dong Dong *et al.*, https://arxiv.org/abs/1806.08198

[16] MONAS: Multi-Objective Neural Architecture Search using Reinforcement Learning, Chi-Hung Hsu *et al.*, https://arxiv.org/abs/1806.10332

The Naive Neural Architecture Search: Model

- input images x $\in \mathbb{R}^{hxwxc}$ from an input space X , and a set of labels x $\in \mathbb{R}$ from an output space Y
- Mapping η: $X, Y, P \rightarrow \mathcal{M}(\theta)$
- $\mathcal{N}^m(\theta_{□}) \circ \ldots \mathcal{N}^2(\theta_2) \circ \mathcal{N}^1(\theta_1)$, where each layer implementation $\mathcal{N}^i$ is parametrized by θ_i, and implements the functionality f_i



- $y = f_{□}(f_{□-1}(\ldots(f_1(x))\ldots))$, where each f_i can be an affine transform, non-linear transform , etc.

```
model = nn.Sequential(nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size = 2, stride = 2)),
                      nn.Linear(400, 10),
                      torch.nn.Softmax(dim=10))
```

Naive NAS: Representing $f_n(f_{n-1}(\ldots(f_1(x))\ldots))$

1. Assign identifiers for each component of the architecture
2. Let the encoding of a single layer be: [**<layer type><input><output><kernel size><stride><padding>…]**
3. Let the architecture be a concatenation of all such layers in sequence:
   [[<layer 1><param 1><param 2>**…]**
   [<layer 2><param 1><param 2>**…]**
   [<layer n><param 1><param 2>**…]]**

Operation	**Params**	**Values**
Conv2D	Kernel size	{1,3,...}
	Out Channel	{16,32,...}
	In Channel	{16,32...}
	Stride	{1,2}
Linear	In	{10,64,..}
	Out	{10,64,..}
Softmax		

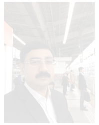
Operation	**Encoding**
Conv2D	1
Linear	2
MaxPool2D	3
BN	4
Softmax	5

tcs Research

Inventing for impact

Algorithm 1: Naive NAS - Random Architecture Search

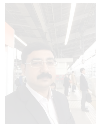
```
Initialize: Set max_layers, max_params, max_score, Layers,
Params, Values lists & empty architecture $\mathfrak{n}$

while $\rho$ < max_score do
  for each $l$ in Layers do
     for each $h$ in Params [ $l$ ] do
        2. $l$ [ $h$ ] $\leftarrow$ random_choice ( Values [ $h$ ] )
     3. Append $l$ to architecture $\mathfrak{n}$
  4. Convert string architecture $\mathfrak{n}$ to $\mathcal{M}(\theta)$
  5. Train and evaluate $\mathcal{M}(\theta)$ to get ACC, SIZ, RAM, MAC
  6. Convert ACC, SIZ, RAM, MAC into a weighted score $\rho$
```



```
for each $l$ in Layers do
  for each $h$ in Params
    random_apply($\mathfrak{n}$[$l$ [$h$ ] ], Values($h$))
```

Enhancing Naive NAS

- 👎 Problems of Naive NAS
  - ✖ Learns nothing, better architecture are NOT generated progressively
  - ✖ We may get a good architecture, but no guarantee when



Inventing for impact

Enhancing Naive NAS

👎 Problems of Naive NAS

- ✖ Learns nothing, better architecture are NOT generated progressively
- ✖ We may get a good architecture, but no guarantee when

❓ How to make the search intelligent?

- 🔧 Consider and incorporate the feedback generated from the evaluation of the network
- 🔧 Should strive for generating progressively better architectures

Enhancing Naive NAS

- 👎 Problems of Naive NAS
  - ✖ Learns nothing, better architecture are NOT generated progressively
  - ✖ We may get a good architecture, but no guarantee when

- ❓ How to make the search intelligent?
  - 🔧 Consider and incorporate the feedback generated from the evaluation of the network
  - 🔧 Should strive for generating progressively better architectures

- 👂 Sometimes however, it works surprisingly well*

[*] **Random Search and Reproducibility for Neural Architecture Search,** Liam Li, Ameet Talwalkar, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, PMLR, 2020

Enhancing Naive NAS

- 👎 Problems of Naive NAS
  - ✖ Learns nothing, better architecture are NOT generated progressively
  - ✖ We may get a good architecture, but no guarantee when

- ❓ How to make the search intelligent?
  - 🔧 Consider and incorporate the feedback generated from the evaluation of the network
  - 🔧 Should strive for generating progressively better architectures

- 👂 Sometimes however, it works surprisingly well*
- 👂 Making the search intelligent is just putting it into one of the existing formulae - not very difficult if we know the formula & understand our problem

[*] **Random Search and Reproducibility for Neural Architecture Search,** Liam Li, Ameet Talwalkar, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, PMLR, 2020

Reinforcement Learning Based NAS: Model

- We model the search (NAS ***agent***) as Markov Decision Process (MDP)
- Agent starts with an initial architecture $\mathfrak{n}$ - its initial ***state***
- ***States*** are associated with $\mathfrak{n}$, identified by a state vector
- From one ***state*** other ***states*** can be generated by modifying $\mathfrak{n}$

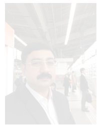
- Modifying $\mathfrak{n}$ is taking some ***action***
- There can be several choices of action from a current state
- Why are we doing all these? We want the agent to choose best ***actions***

tcs Research

Inventing for impact

RL NAS: How to Identify the Best Action?

- Agent aims to take actions, such that a ***reward*** value is maximized
- **Reward** represents the feedback that agent receives from ***environment***
- **Reward** can be the ***validation accuracy***



RL NAS: How to Identify the Best Action?

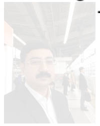
- Agent aims to take actions, such that a ***reward*** value is maximized
- **Reward** represents the feedback that agent receives from ***environment***
- **Reward** can be the ***validation accuracy***
- **Reward** can be ***immediate*** or in distant ***future*** for a state trajectory

$$R_t = \rho_{t_1} + \gamma \rho_{t_2} + \gamma^2 \rho_{t_3} + \ldots$$

RL NAS: Value of a State Transition

- Need to associate a ***value*** for transition from one state to other on an action
- Cumulative ***value*** of a state transition from **C□₁** to **C□₂,** on an ***action*** **α**

$$V(C_{t_1}, \alpha) = \rho^{\alpha}_{C_{t_1} C_{t_2}} + \gamma \mathbb{V}_{C_{t_2}}$$

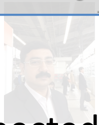


Inventing for impact

RL NAS: Value of a State Transition

- Need to associate a ***value*** for transition from one state to other on an action
- Cumulative ***value*** of a state transition from **C□$_1$** to **C□$_2$,** on an ***action* α**

$$V(C_{t_1}, \alpha) = \rho^{\alpha}_{C_{t_1} C_{t_2}} + \gamma \mathbb{V}_{C_{t_2}}$$



- A greedy strategy considers the maximum expected reward starting from **C□$_2$**

$$V(C_{t_1}, \alpha_{t_1}) = \rho^{\alpha_{t_1}}_{C_{t_1} C_{t_2}} + \gamma \max_{\alpha_{t_2}} V(C_{t_2}, \alpha_{t_2})$$

- ***V(◦)*** → ***action-value function*** and the individual ***V(C□, α)*** → ***Q-values***

tcs Research

Inventing for impact

Algorithm 2: RL NAS

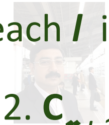
```
Initialize: Set max_score, Layers, Params, Values lists, empty
architecture $\mathfrak{n}$, action set $\mathcal{A}$ & state set $\mathcal{C}$

while $\rho$ < max_score do
 for each $l$ in Layers do
     for each $h$ in Params [ $l$ ] do
         2. $l$ [ $h$ ] $\leftarrow$ random_choice ( Values [ $h$ ] )
     3. Append $l$ to architecture $\mathfrak{n}$
     4. Update $\mathcal{A}$ with $\alpha$, $\mathcal{C}$ with $C_{\mathfrak{n}}$
  4. Convert string architecture $\mathfrak{n}$ to $\mathcal{M}(\theta)$
  5. Train and evaluate $\mathcal{M}(\theta)$ to get ACC, SIZ, RAM, MAC
  6. Convert ACC, SIZ, RAM, MAC into a weighted score $\rho$
  for each $C_{\mathfrak{n}}$ in $\mathcal{C}$ do
     for each $\alpha$ in $\mathcal{A}$ do
     7. Use $C_{\mathfrak{n}}$, $\alpha$, $\rho$, $V(C_{\mathfrak{n}}, \alpha)$ to train $\mathcal{M}(\theta)$
```

EXPLORE

EVALUATE

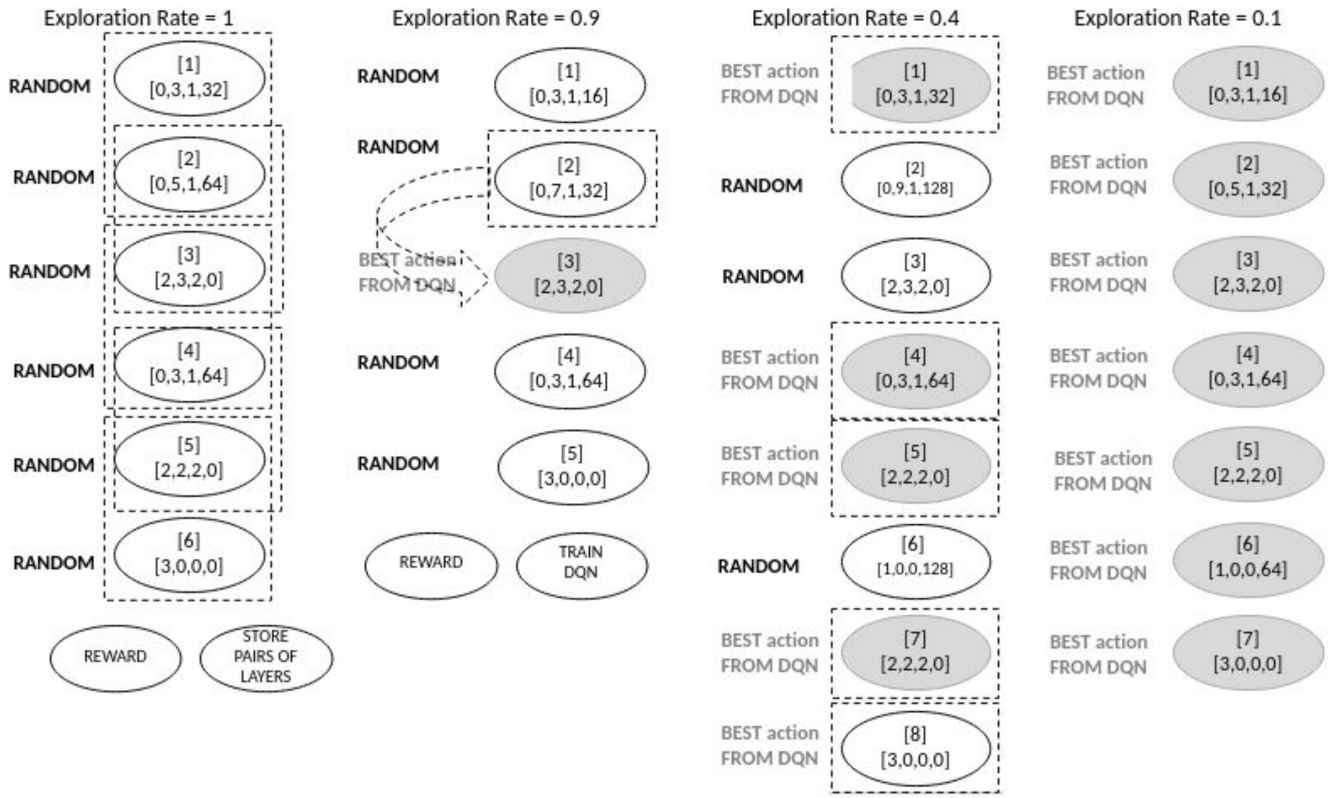
Algorithm 2: RL NAS

```
Initialize: Set max_layers, max_params, max_score, Layers, Params,
Values lists,  empty architecture $n$, & action set $\mathcal{A}$

while $\rho$ < max_score do

 If random_value < $\varepsilon$ do

    1. EXPLORE

 else do

          for each $l$ in Layers do

              2. $C_{n+1} \leftarrow \mathcal{M}(\theta)(C_n)$                  EXPLOIT

              3. $l \leftarrow C_{n+1}$

          4. Append $l$ to architecture $n$

 7. EVALUATE
```



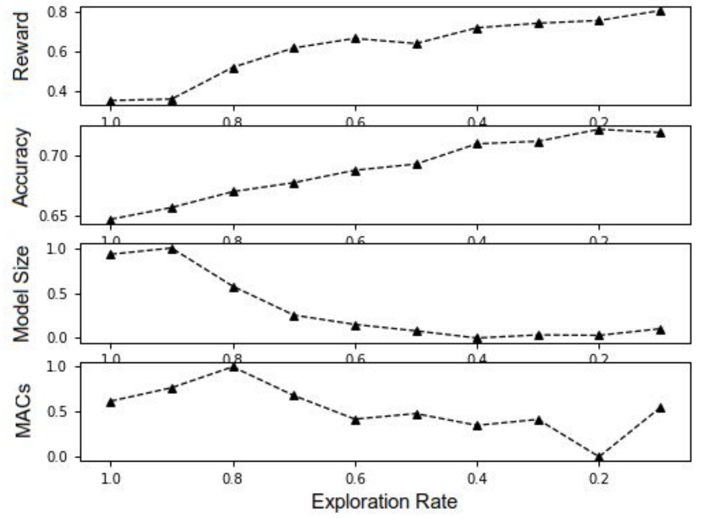
Case Study: RL-NAS

Generating Tiny Deep Neural Networks for ECG Classification on MCU, Shalini Mukhopadhyay, Swarnava Dey, Avik Ghose (TCS Research), Pragya Singh (IIIT-D), Pallab Dasgupta (IIT KGP) IEEE Percom 2023

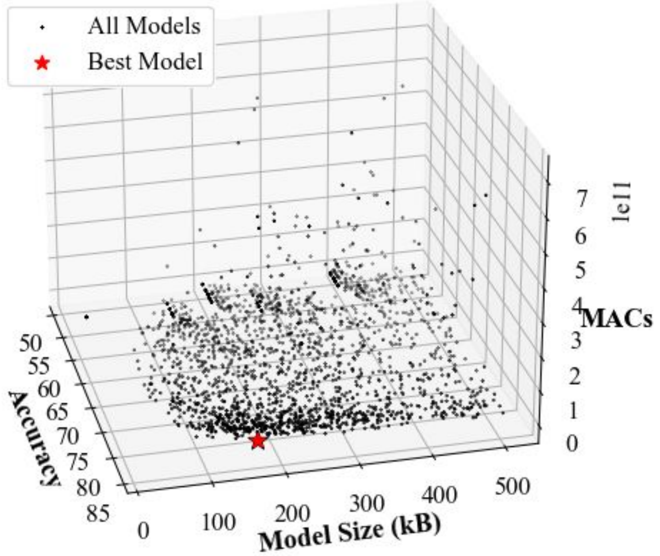

Exploration Rate = 1
RANDOM [1] [0,3,1,32]
RANDOM [2] [0,5,1,64]
RANDOM [3] [2,3,2,0]
RANDOM [4] [0,3,1,64]
RANDOM [5] [2,2,2,0]
RANDOM [6] [3,0,0,0]
REWARD
STORE PAIRS OF LAYERS
Exploration Rate = 0.9
RANDOM [1] [0,3,1,16]
RANDOM [2] [0,7,1,32]
BEST action FROM DQN [3] [2,3,2,0]
RANDOM [4] [0,3,1,64]
RANDOM [5] [3,0,0,0]
REWARD
TRAIN DQN
Exploration Rate = 0.4
BEST action FROM DQN [1] [0,3,1,32]
RANDOM [2] [0,9,1,128]
RANDOM [3] [2,3,2,0]
BEST action FROM DQN [4] [0,3,1,64]
BEST action FROM DQN [5] [2,2,2,0]
RANDOM [6] [1,0,0,128]
BEST action FROM DQN [7] [2,2,2,0]
BEST action FROM DQN [8] [3,0,0,0]
Exploration Rate = 0.1
BEST action FROM DQN [1] [0,3,1,16]
BEST action FROM DQN [2] [0,5,1,32]
BEST action FROM DQN [3] [2,3,2,0]
BEST action FROM DQN [4] [0,3,1,64]
BEST action FROM DQN [5] [2,2,2,0]
BEST action FROM DQN [6] [1,0,0,64]
BEST action FROM DQN [7] [3,0,0,0]

Case Study: RL-NAS

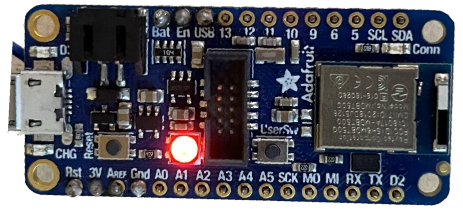
- RL based NAS (MetaQNN[2]) - ECG Atrial Fibrillation, Smoking episode, & Hand Gesture
- 3000 iterations, 90 hours (A100), 20-layer, 225 KB model with SOTA F1 score for ECG-AF
- MO - ACC, SIZE, SRAM, #MAC: $\mathbf{R = (W_a ACC + \sum_i W \exp(w_i P_i)) / \sum |W|}$



Multi-objective Convergence



Clear Front of good models



Adafruit Feather
NRF52840
RAM Size : 256 kB
Flash Size : 1 MB
Target MACs : 16 M

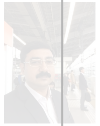
Search Parameters

Generating Tiny Deep Neural Networks for ECG Classification on Micro-controllers, Mukhopadhyay et al., IEEE Percom 2023

Algorithm 3: EVO NAS

```
Initialize: Set max_score, max_pop, Layers, Params, Values lists, empty architecture $\boldsymbol{\pi}$, action set $\mathcal{A}$ & state set $\mathcal{C}$

1. $POPULATION \leftarrow \{\phi\}$
for 1… max_score   do
  2. $\boldsymbol{\pi}, \mathcal{A} \leftarrow$ EXPLORE
  3. $\rho \leftarrow$ EVALUATE($\boldsymbol{\pi}$)
  4. $POPULATION$.append($\boldsymbol{\pi}, \mathcal{A}, \rho$)
mut_rate $\leftarrow$ 1.0
for 1… max_score   do
  5. parent $\leftarrow$ $POPULATION$.fittest()
  6. child $\leftarrow$ mutate(parent, mut_rate)
```

```
(contd.)
for 1… max_score   do
  7. parent $\leftarrow$ $POPULATION$.fittest()
  8. Child, $\mathcal{A} \leftarrow$ mutate(parent, mut_rate)
  9. $\rho \leftarrow$ EVALUATE(child)
  10. $POPULATION$.append(child, $\mathcal{A}, \rho$)
  11. $POPULATION$.remove_least_fits()
  12. Update mut_rate
13. final_architectures $\leftarrow$ $POPULATION$.fittest()
```

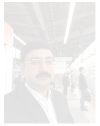


tcs Research

Inventing for impact

Points to Note

- ✋ The main approach is sampling architectures randomly, evaluating, learning to sample better architectures in an informed manner
- ✋ Sequentially building network
- ✋ Reward is completely decoupled from search
- ✋ Discrete choices for layers and params

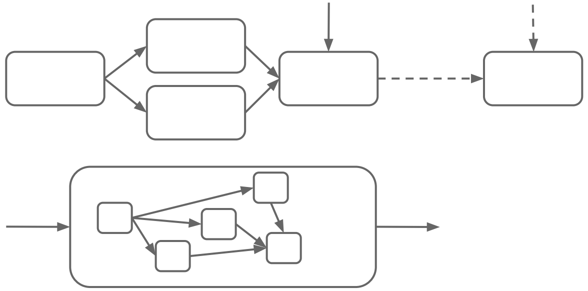


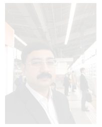
Inventing for impact

DARTS: Differentiable NAS

Considers a fixed network a number of ***cells***

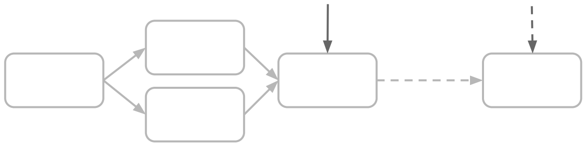
Cells have nodes, connectivity (ops) is searched



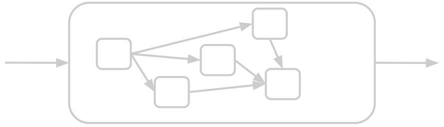


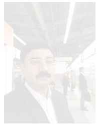
DARTS: Differentiable NAS

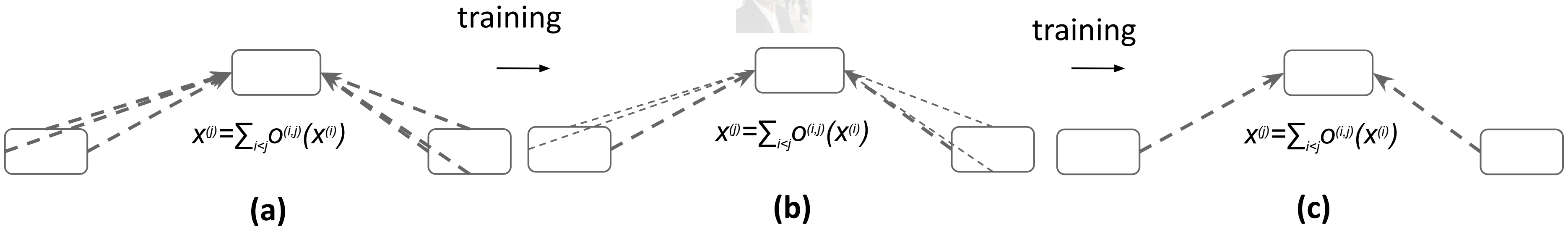
Considers a fixed network a number of ***cells***



Cells have nodes, connectivity (ops) is searched







training
training
$x^{(j)}=\sum_{i<j} o^{(i,j)}(x^{(i)})$
$x^{(j)}=\sum_{i<j} o^{(i,j)}(x^{(i)})$
$x^{(j)}=\sum_{i<j} o^{(i,j)}(x^{(i)})$
(a)
(b)
(c)

$$\bar{o}^{(i,j)}(x)=\sum_{o\in\mathcal{O}} \left(\left(\exp(\alpha^{(i,j)}_{o}) \right) / \left(\sum_{o'\in\mathcal{O}} \exp(\alpha^{(i,j)}_{o'}) \right) \right) * o(x)$$

Algorithm 4: DARTS – Differentiable Architecture Search

```
Set max_layers, max_params, Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge $(i, j)$
while not converged do
    1. Update architecture $\alpha$ by descending $\nabla \alpha L_{val}(w - \xi \nabla w\, L_{train}(w, \alpha), \alpha)$
    ($\xi = 0$ if using first-order approximation)
    2. Update weights $w$ by descending $\nabla w L_{train}(w, \alpha)$
3. Discretize for the final architecture based on the learned $\alpha$.
4. Re-train
```

$$min_{\alpha} L_{val}(\alpha, argmin_{w} L_{train}(\alpha,w)) \approx \nabla \alpha L_{val}(w - \xi \nabla w\, L_{train}(\alpha,w)$$

TCS Research

Inventing for impact

Take Home Points

✋ Gordon Moore is no more. Moore's law is supposed to reach saturation levels.
Still, it seems that the capacity of tiny devices will keep increasing in this decade

✋ Consequently, we have to search larger & complex networks for *Edge* devices

✋ Thus multi-objective, one-shot differential NAS is *the* research direction for TinyML

✋ Meanwhile, this tutorial showed that the existing multi-objective, sample-based NAS methods are <u>quite usable</u> for the next few years

Thank You for bearing with me! 😊

Questions 👂

TINY ML TALKS webcast

Copyright Notice

Copyright Notice

www.tinyml.org