DeepPicarMicro: Applying TinyML to Autonomous Cyber Physical Systems

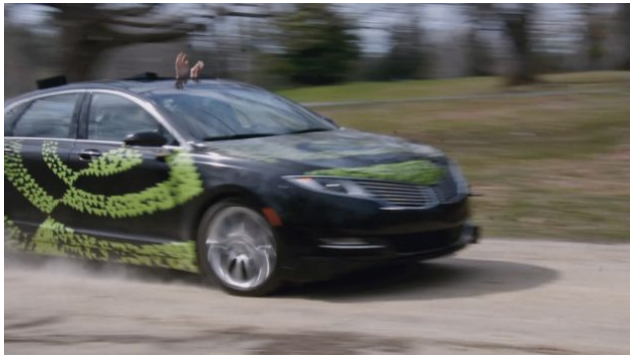
Michael Bechtel, QiTao Weng, Heechul Yun
University of Kansas, US

KU THE UNIVERSITY OF KANSAS

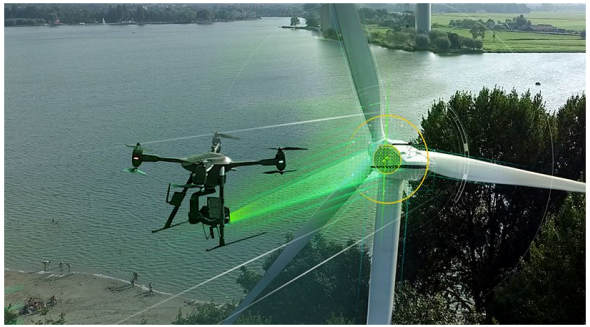
KU-CSL COMPUTER SYSTEMS LAB

Cyber Physical Systems (CPS)

- Deployed in many different areas of our daily lives.
  - Automotive, avionics, healthcare, etc.
- **CPS are becoming more intelligent.**
  - Many now employ machine learning algorithms.



Autonomous Vehicles



Autonomous UAVs

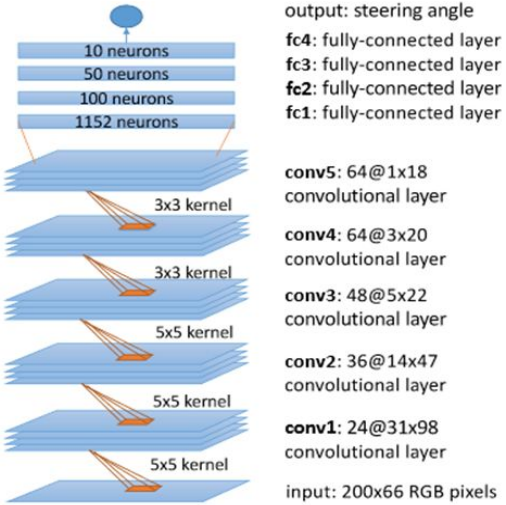
Smart Robots

KU THE UNIVERSITY OF KANSAS

DAVE-2[1]

- 2016 project by NVIDIA
- **Used a DNN called PilotNet**
- Could drive on public roads

PilotNet architecture

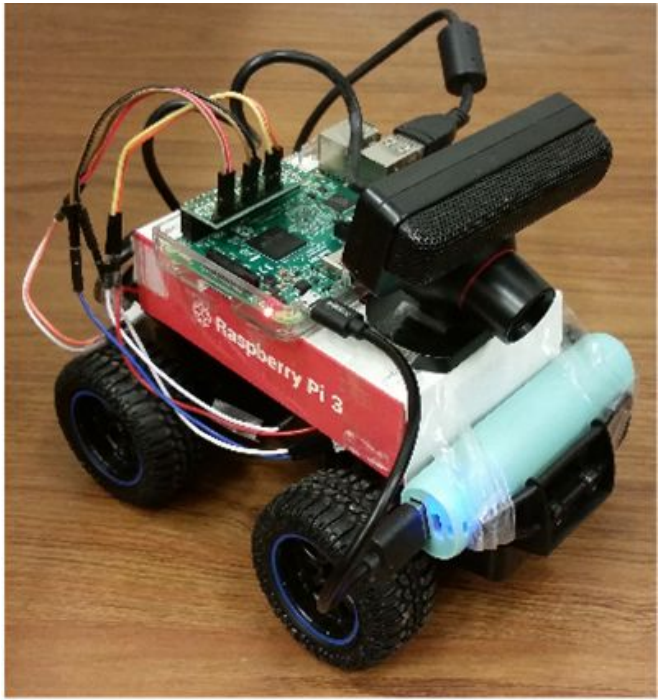


[1] Mariusz Bojarski, et al. End to End Learning for Self-Driving Cars. *arXiV,* 2016.

DeepPicar[1]

- Low cost, small scale replication of DAVE-2
- Uses the exact same PilotNet DNN
- Runs on a Raspberry Pi 3/4 in real-time

- **Q: Can we run PilotNet on a MCU platform?**



KU THE UNIVERSITY OF KANSAS

[1] Michael Bechtel, Elise McEllhiney, Minje Kim, Heechul Yun. DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018.

Microcontroller Unit (MCU)

- Widely used in many embedded/CPS applications
- Integrates computing logic, storage, memory into a single chip
- Inexpensive, power efficient, and highly deterministic
- But highly *resource constrained.*

Part	Raspberry Pi 4	Raspberry Pi Pico
CPU	BCM2837 4x Cortex-A72@1.5GHz	RP2040 2x Cortex-M0+@133MHz
Memory	48BK(I)/32KB(D) L1 cache 1MB L2 (16-way) L2 cache 4GB LPDDR4	264KB SRAM
Storage	8GB+ micro-SD	2MB Flash
Power	3A	<100mA

- **Challenge: How to run complex DNNs on an MCU?**

KU THE UNIVERSITY OF KANSAS

TinyML

- Refers to frameworks and methods to execute DNNs locally on MCUs
  - Premise: better reliability, energy efficiency, and privacy than connecting to cloud servers
- Significant interests exist in both industry and academia
  - Big potential in many industries: agriculture, medical devices, industrial systems…
- Several ML frameworks are specifically designed to target MCUs
  - **TensorFlow Lite Micro (TFLMicro)**, CMSIS-NN, uTensor, etc.

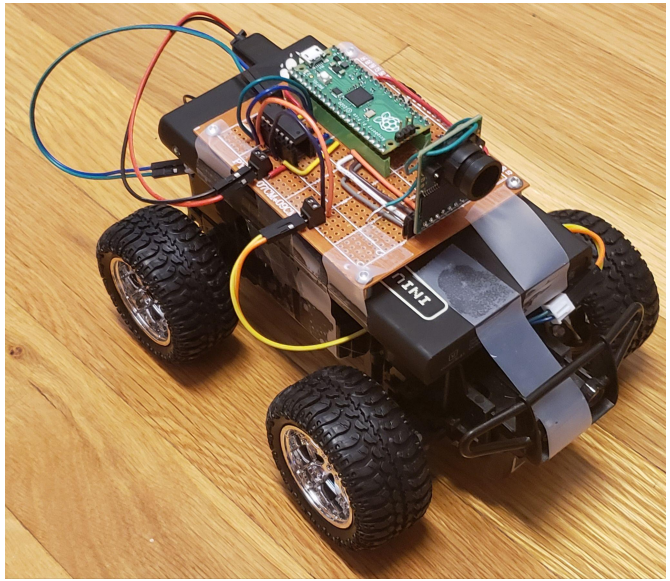
- **Our work focuses on evaluating the potential of TinyML in CPS**

Outline

- Background
- **DeepPicarMicro**
- Neural Architecture Search (NAS)
- Evaluation
- Conclusion

DeepPicarMicro

- An MCU-based self-driving car testbed
- Uses a Raspberry Pi Pico
  - 2 Cortex-M0+ cores @133MHz
  - 264KB SRAM, 2MB Flash
- Employs CNN-based end-to-end control
- Uses TFLMicro framework



KU THE UNIVERSITY OF KANSAS

PilotNet Architecture

- 9-layers
  - Five convolutional
  - Four fully-connected
- ~250K weights
  - Model size of ~1MB (fp32, tflite)
- ~27M MACs
  - Multiply-accumulate operations

Layer	Input size	Output size	Weights	MACs
Conv1	66x200x3	31x98x24	1.8K	5.5M
Conv2	31x98x24	14x47x36	21.6K	14.2M
Conv3	14x47x36	5x22x48	43.2K	4.8M
Conv4	5x22x48	3x20x64	27.7K	1.7M
Conv5	3x20x64	1x18x64	36.9K	663.6K
FC1	1152	100	115.3K	115.2K
FC2	100	50	5.1K	5K
FC3	50	10	510	500
FC4	10	1	11	51
Total			252.2K	26.9M

- **Non-trivial network to run on an MCU**
- We apply various optimizations to run it on a Pico MCU

Quantization

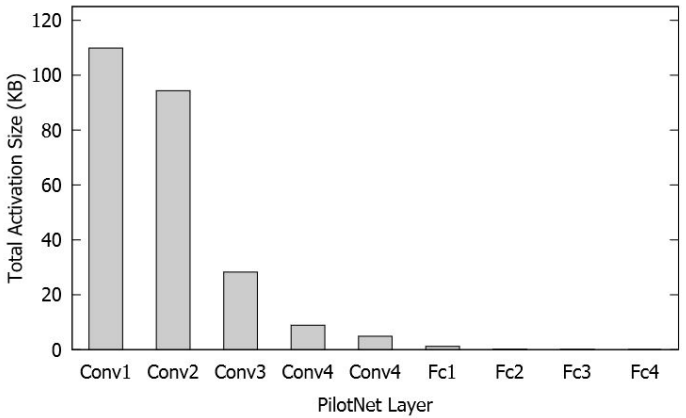
- Standard optimization technique for mobile/embedded
  - Reduces model size (ex: 32-bit → 8-bit weight: 4X reduction)
  - Can also improve performance due to greater parallelism
  - Typically with modest impact to accuracy
- We applied **quantization-aware training**[1]
  - Minimize accuracy loss caused by quantization
- Impact of quantization to PilotNet (32-bit to 8-bit)
  - Model size: ~1MB → ~250KB (~4X reduction) << Pico MUC's Flash (2MB)
  - Accuracy: 87.6% ⇒ 86.9% (<1% accuracy loss)

KU THE UNIVERSITY OF KANSAS

[1] https://www.tensorflow.org/model_optimization/guide/quantization/training

Memory (SRAM) Constraints

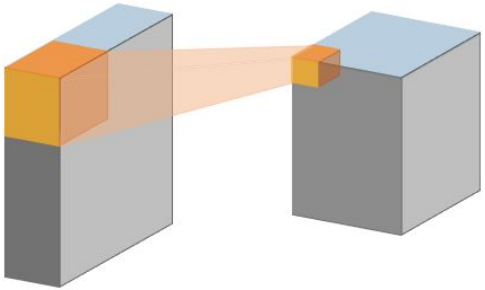
- TFLMicro needs input and output activation buffers in SRAM for processing DNN layers
- The largest layer determines the maximum SRAM demand
  - Layer size = input + output activation buffers
  - PilotNet's largest layer (~110KB) < Pico MCU's SRAM (264KB)



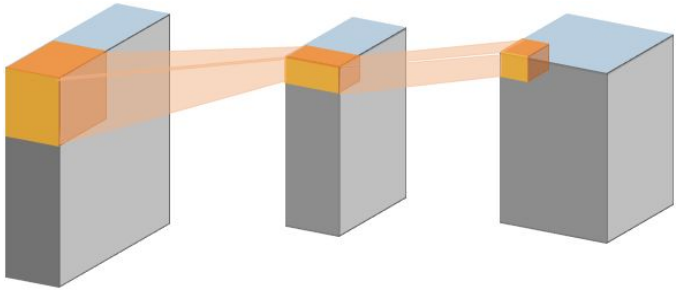
- **PilotNet can fit and achieve high accuracy on a Pico MCU**
- **But its inference latency (> 3 second) is still too high for real-time**

Depthwise Separable Layers[1]

- Split standard 2D convolutional layers into two layers
  - Depthwise Convolution: # of filters = # of input channels
  - Pointwise convolution: Convolution with a 1x1 kernel size



Standard Convolution



Depthwise Separable

- Reduces per-layer MACs by a factor of K^2
  - K = layer kernel size

KU THE UNIVERSITY OF KANSAS

[1] Francois Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Impact of Depthwise Separable Layers on PilotNet

- Use Depthwise Separable layers in PilotNet
  - Instead of the original convolutional layers

	Conv2D	Depthwise
Weights	252.2K	133.7K
MACs	26.9M	2.1M
Val. Loss	0.027	0.032
Accuracy (%)	86.9	85.7
Latency (ms)	3025	525

Reduces MACs by ~12.6X

- **Performance is still unsatisfactory (>500ms per inference on Pico MCU)**
- **Q: Can we further optimize PilotNet to run in real-time on MCUs?**

Outline

- Background
- DeepPicarMicro
- **Neural Architecture Search (NAS)**
- Evaluation
- Conclusion

Neural Architecture Search (NAS)

- Technique used to find optimal DNN architectures
  - Systematically explore different network architectures
- Search Space
  - The set of all DNN layouts to search
  - Typically defined by varying DNN parameters that affect model MACs
  - Ex: Layer width, input resolution, etc.
- Search objective
  - Many NAS maximize accuracy subject to resource constraints
  - Common constraints: latency, SRAM/flash sizes

NAS on PilotNet

- Inspired by the state-of-the-art *MCUNet*[1] paper's approach
- Search space
  - Width multiplier = {0.2, 0.4, 0.7, 0.8, 0.9, 1.0}
  - Depth = All unique combinations of 3 to 9 layers
  - Input resolution = 68x68x1 (height/width/depth)
  - Total space = 720 distinct layouts
- Evaluate all possible layouts in the search space
  - Must check whether the constraints are met
  - **Ex: 133 ms control deadline**

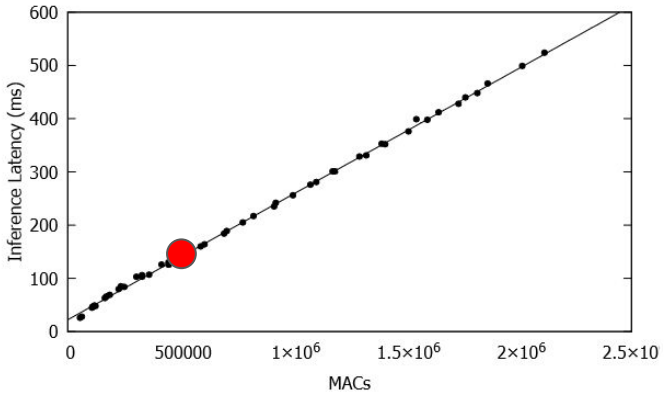
KU THE UNIVERSITY OF KANSAS

[1] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, Song Han. MCUNet: Tiny Deep Learning on IoT Devices. *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Latency Prediction

- Problem: How to know a model's latency?
  - Executing a large number of models on a real MCU is time consuming
- **A model's MACs correspond to its inference latency**
  - Due to MCUs' simpler architectures, this relationship is highly linear
- To find this relationship on the Pico, we run 50 different DNNs
  - MACs range from ~54K to ~2.1M

~470,000 MACs → 133 ms



Search Algorithm

- Start with all model layouts in search space
- **Only search over layouts with MACs <= 470K**
  - Out of 720, we search over 349 model layouts
- Train a DNN model for each layout we search
- **We measure both model validation loss and latency**
  - In order to find the optimal model

Performance Prediction

- Accuracy alone is not sufficient to predict CPS performance
- **Latency** is also a critical property in many CPS
  - Lower accuracy & latency can be better than higher accuracy & latency
  - Higher latency → longer reaction time → worse CPS performance
- Need to consider both accuracy and latency
- Proposed joint optimization on ***validation loss and latency***
  - Calculate a heuristic score for each model:

$$Score = norm(ValLoss) + norm(Latency)$$

- **Choose the model with the lowest heuristic score**

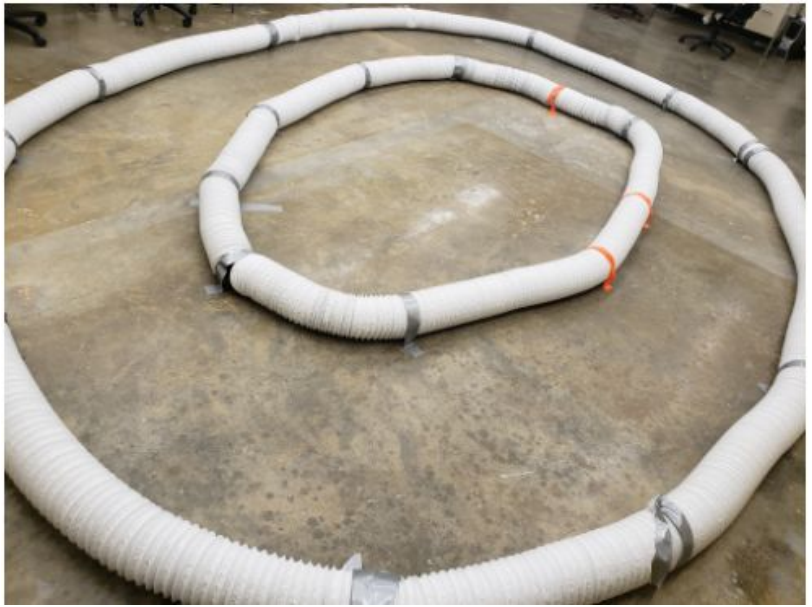
Outline

- Background
- DeepPicarMicro
- Neural Architecture Search (NAS)
- **Evaluation**
- Conclusion

Real Track Setup

- We constructed and used a simple real-world track
- Train models on 10,000 samples

- **We evaluate 16 models**
- Run all models around the track 10 times
  - Measure number of laps without crashes



KU THE UNIVERSITY OF KANSAS

Performance in Real Track

Model #	Latency (ms)	Val. Loss	Accuracy (%)	Score	Laps w/o Crash
1	37	0.031	84.9	0.04	7
2	58	0.031	85.0	0.27	9
3	85	0.032	85.2	0.50	7
4	73	0.042	82.1	0.56	7
5	39	0.060	75.3	0.58	0
6	107	0.033	83.8	0.75	4
7	110	0.032	85.1	0.84	5
8	52	0.073	67.7	0.92	0
9	97	0.050	78.2	0.96	0
10	96	0.052	76.5	0.98	2
11	37	0.084	65.6	1.00	0
12	87	0.057	75.0	1.03	1
13	122	0.044	80.4	1.16	2
14	85	0.072	69.8	1.23	0
15	111	0.066	72.8	1.56	0
16	111	0.083	62.7	1.71	0

- **Both accuracy and latency affect performance**
- The heuristic score was effective at predicting performance
  - The predictions for some models were inaccurate (e.g. Model #5)

KU THE UNIVERSITY OF KANSAS

Udacity Simulator Setup

- Perform similar tests in a simulator environment
  - Better evaluate how accuracy and latency affect performance
- Train models on ~14.4K samples
  - **All models can run on the Pico MCU**
- In total, we test 5 models
  - Validation losses from 0.26 to 0.45
  - Add synthetic delays from 0 to 100 ms
- Perform 5 runs with each model
  - Run for 300 seconds or till crash
  - Measure average runtime

KU THE UNIVERSITY OF KANSAS

Performance in Udacity Simulator

Average runtime (seconds)

Latency (ms)	Validation Loss				
	0.026	0.030	0.035	0.040	0.045
0	300	300	92	81	29
20	300	300	84	79	30
40	300	300	81	78	27
60	300	300	79	76	28
80	93	59	55	60	29
100	43	56	39	45	28

Heuristic scores

Latency (ms)	Validation Loss				
	0.026	0.030	0.035	0.040	0.045
0	0.00	0.12	0.28	0.46	0.64
20	0.20	0.32	0.48	0.66	0.84
40	0.40	0.52	0.68	0.86	1.04
60	0.60	0.72	0.88	1.06	1.24
80	0.80	0.92	1.08	1.26	1.44
100	1.00	1.12	1.28	1.46	1.64

- **Both validation loss and latency affect performance**
- Heuristic scores are again decent at predicting performance
  - There is still much room for improvement

Outline

- Background
- DeepPicarMicro
- Neural Architecture Search (NAS)
- Evaluation
- **Conclusion**

Conclusion

- We present DeepPicarMicro, an MCU-based autonomous RC car testbed
  - Existing DNN architectures can be consolidated and run on MCUs
  - For PilotNet, performance is too poor for real-time applicability
- We employ a NAS approach to optimize the PilotNet architecture
  - We use a Joint Optimization strategy on validation loss and latency
- We evaluate MCU fitting models in real-world and simulated environments
  - Models can successfully navigate both tracks
  - Both latency and accuracy are important factors for performance

Source code available at:
https://github.com/CSL-KU/DeepPicarMicro

KU THE UNIVERSITY OF KANSAS

Thank you!

Disclaimer:

This research is supported by NSF grant CNS-1815959, CPS-2038923 and NSA Science of Security initiative contract no. H98230-18-D-000.

KU THE UNIVERSITY OF KANSAS