Research Project Proposal:
Input-Aware Dynamic Quantization in Deep Neural Networks

Mehmet Emre Akbulut
mehmetemre.akbulut@mail.polimi.it
Computer Science and Engineering Track

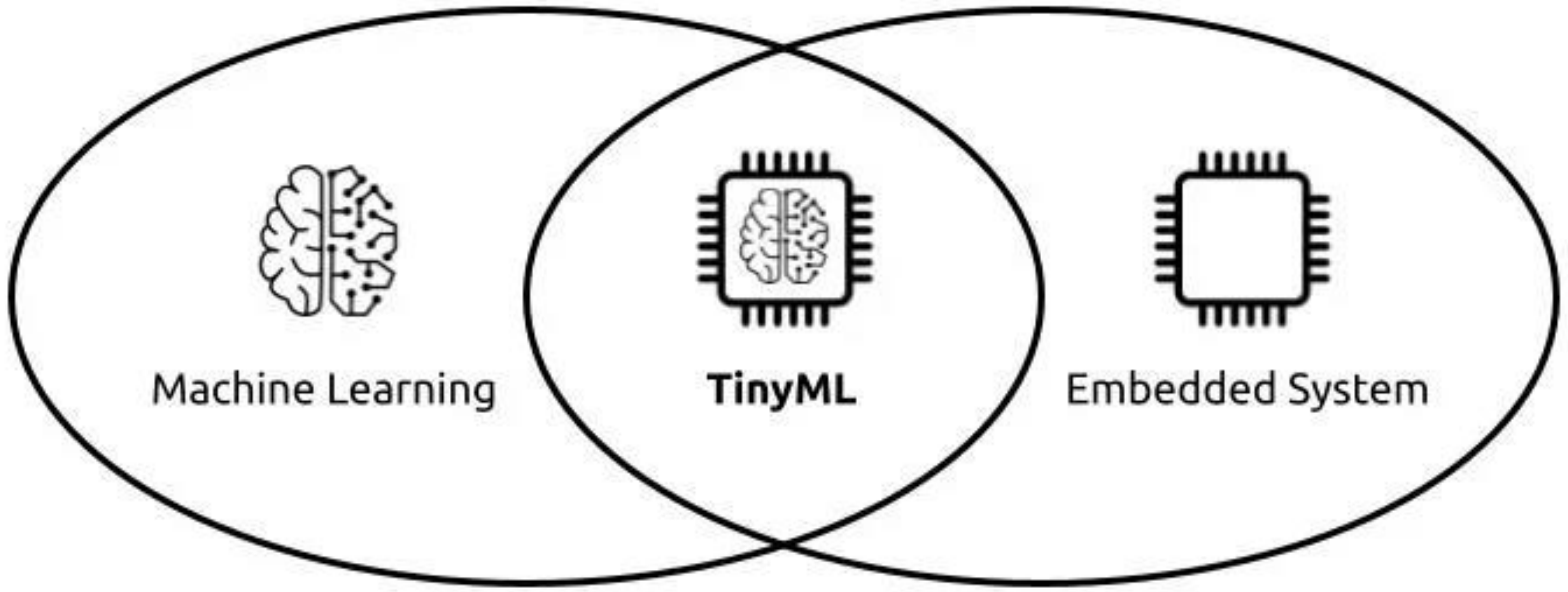
POLITECNICO
MILANO 1863

HONOURS PROGRAMME HP-SR
in Information Technology

Input-Aware Dynamic Quantization in Deep Neural Networks
Research Question

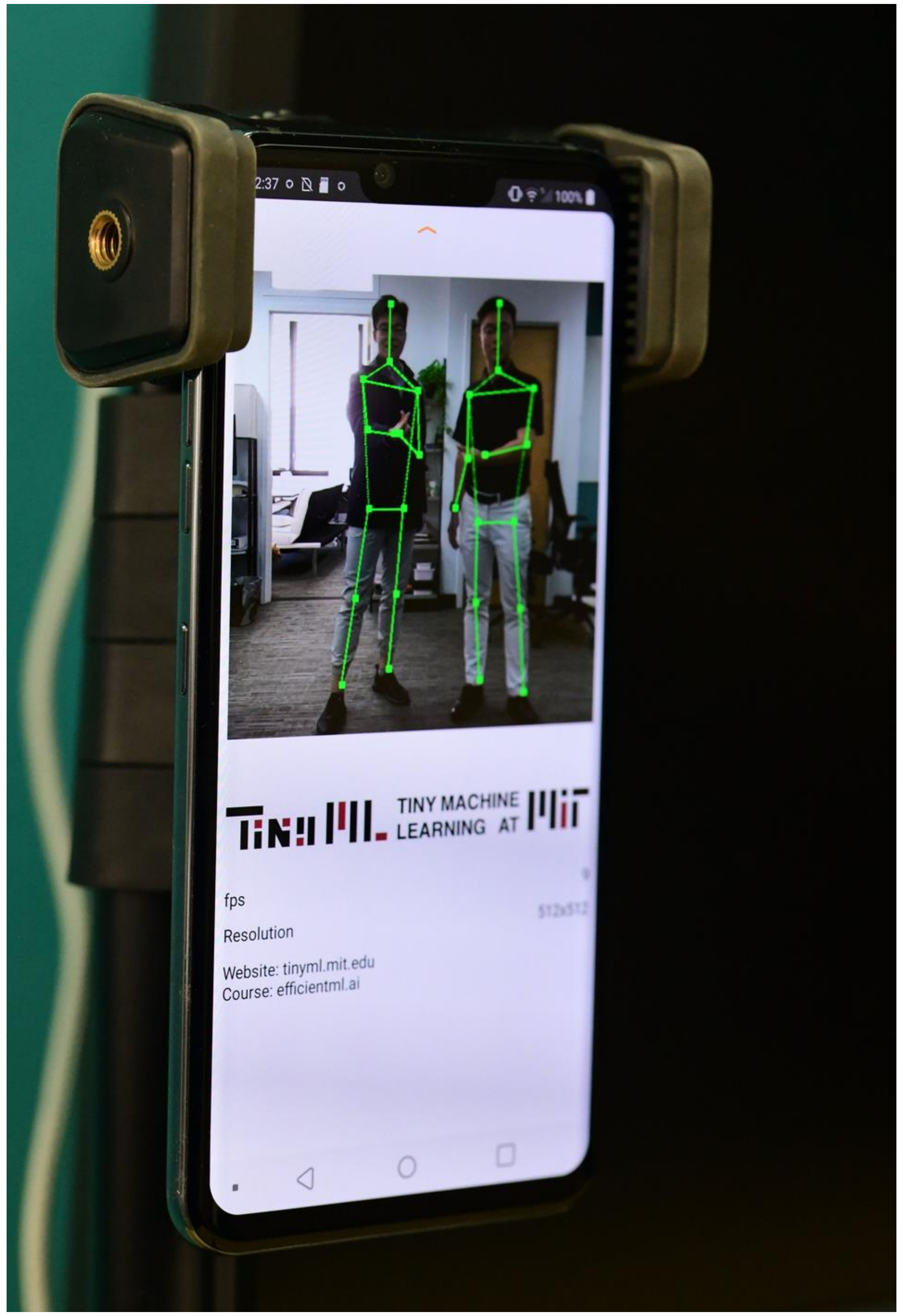
“How can we design and
implement an ***instance-aware dynamic quantization framework*** *that*
adapts bit precision considering ***given input*** *for devices*
with limited memory and computation power ***while maintaining the model***
accuracy*?”*

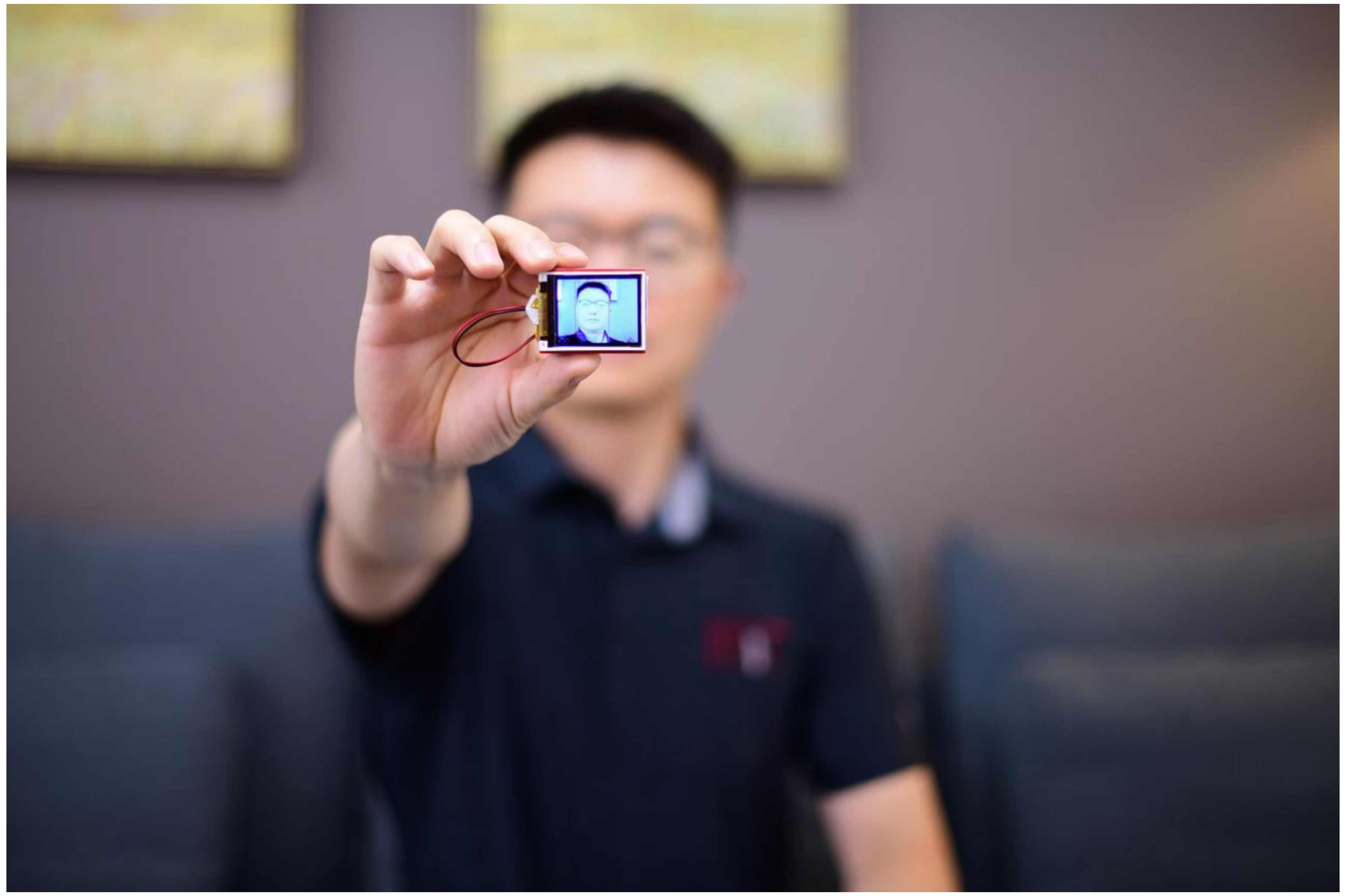
Main Research Areas - What is TinyML?

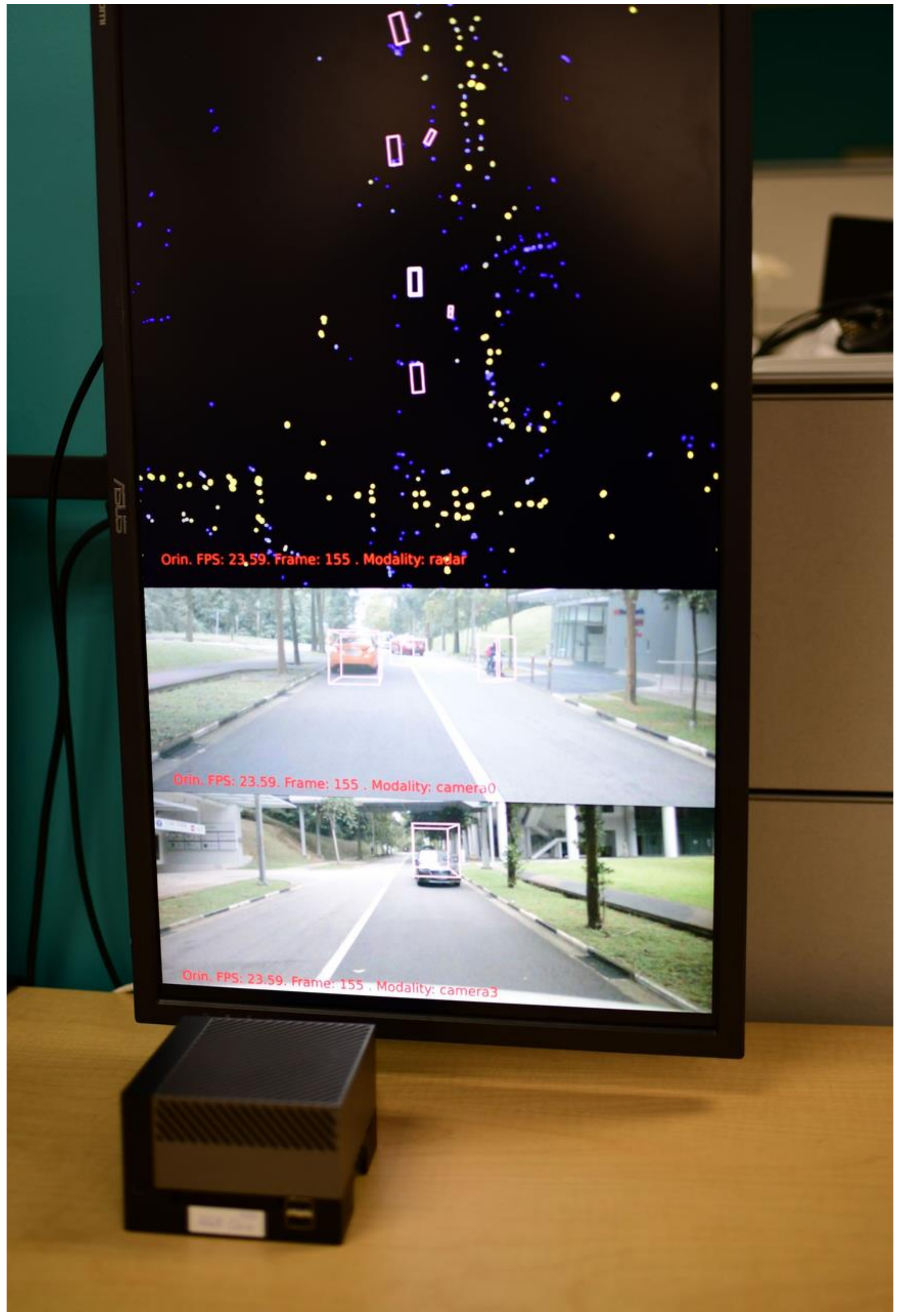


TinyML is a bridge between ML and Embedded Systems [1]

Tiny Machine Learning (TinyML) is a subset of Machine Learning that serves as a link between the ML domain and the embedded system ecosystem.







Credits: MIT HAN Lab [2]

TinyML: Advantages and Challenges

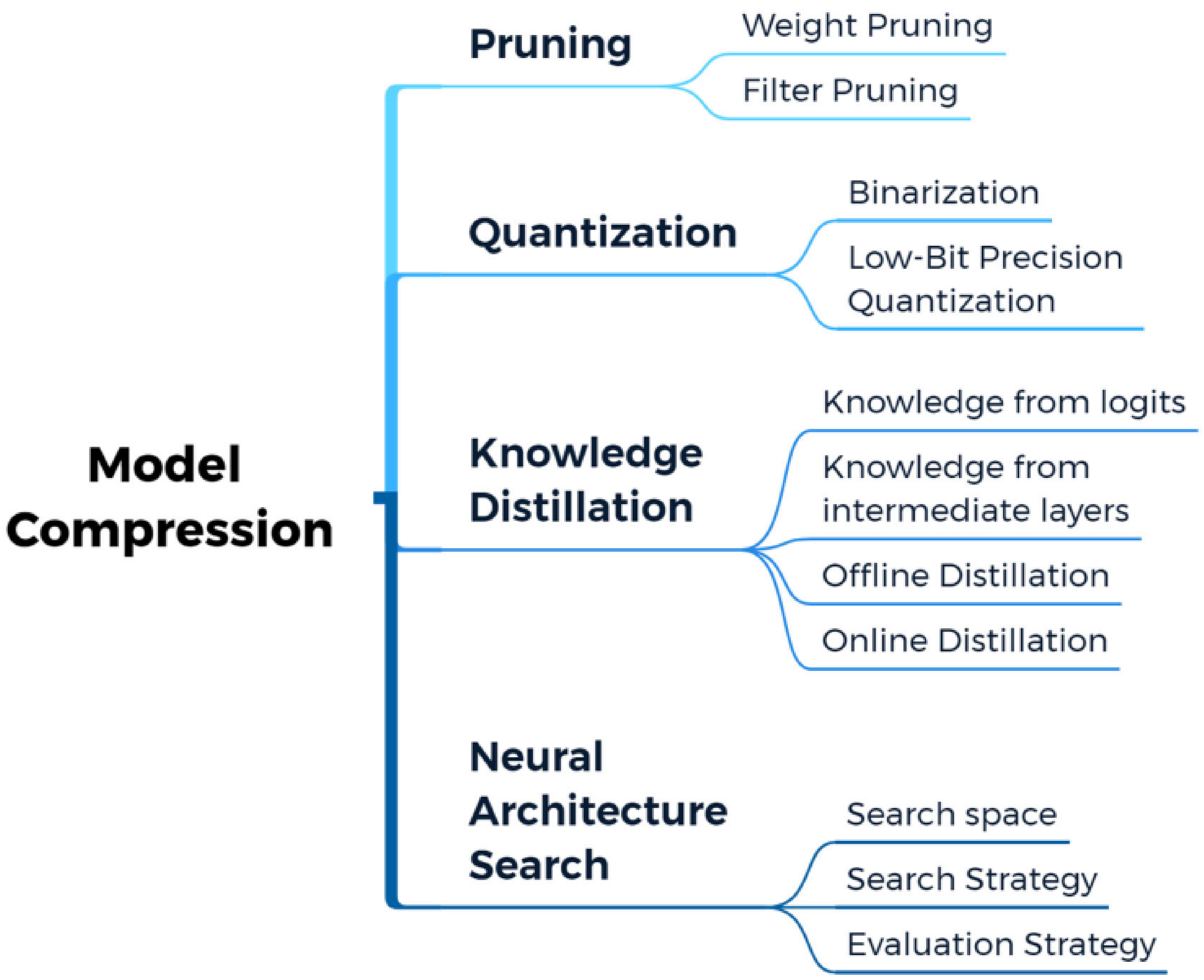
Advantages

- Enables lower memory consumption and computation overhead
- Reducing latency through on-device data processing
- Reducing networking costs
- Incremental learning
- Better Privacy

Challenges

- Resource-constrained edge devices: memory, **computation**, energy
- Hardware complexity and heterogenity
- Miscellaneous techniques: Hardware, Software and ML Algorithms

TinyML: Solutions



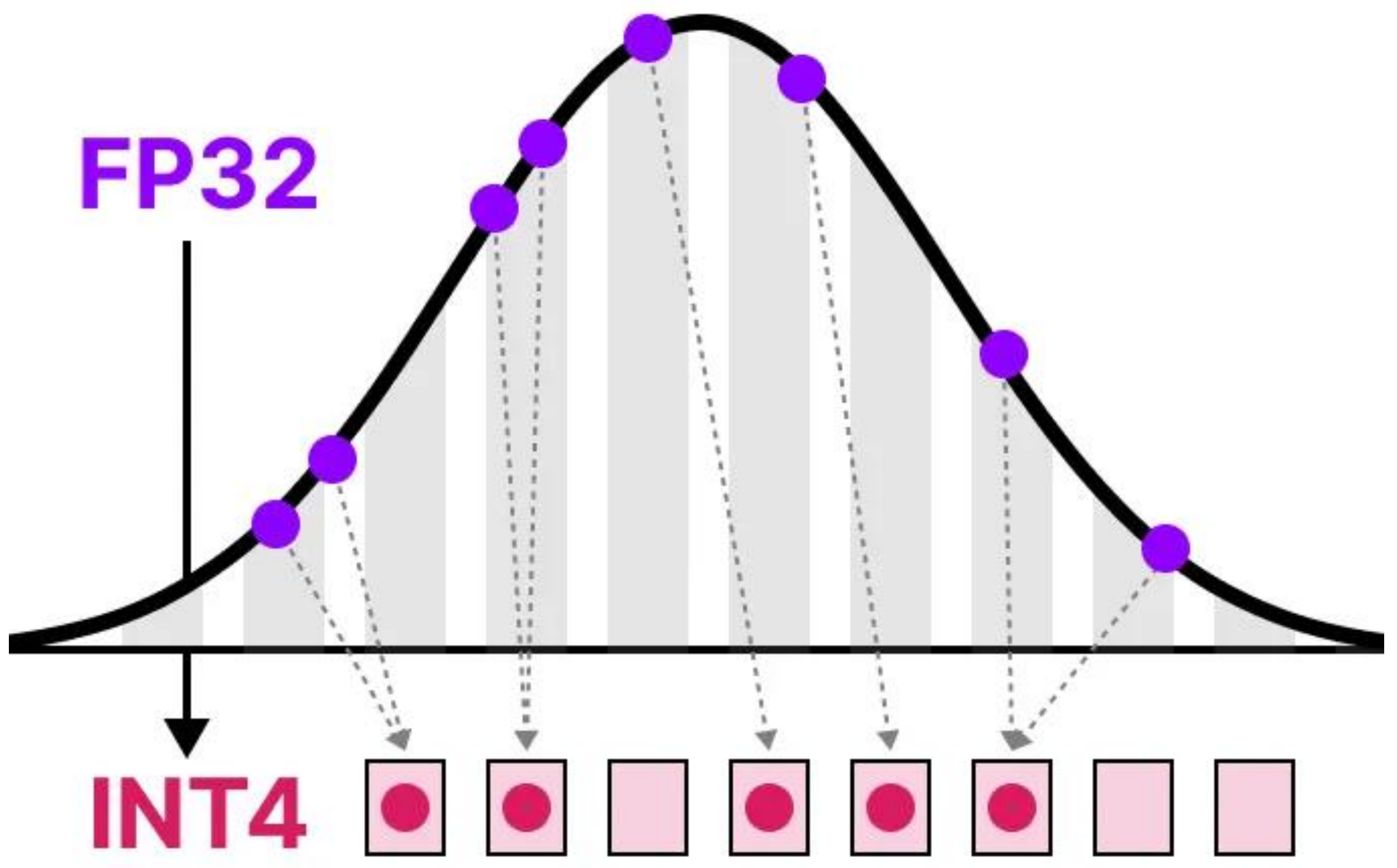
Model Compression Summary [3]

One of the main techniques used to compress and deploy these models on devices with limited resources is **low-precision quantization.**

Input-Aware Dynamic **Quantization** in Deep Neural Networks

- Lower precision of weights, biases, and activations.
- A 32-bit full precision model is compressed to a low-bit representation by employing bit widths from 8-bit to 1-bit.

- Lower memory consumption and fewer arithmetic operations with little loss in task performance
- Higher inference speed



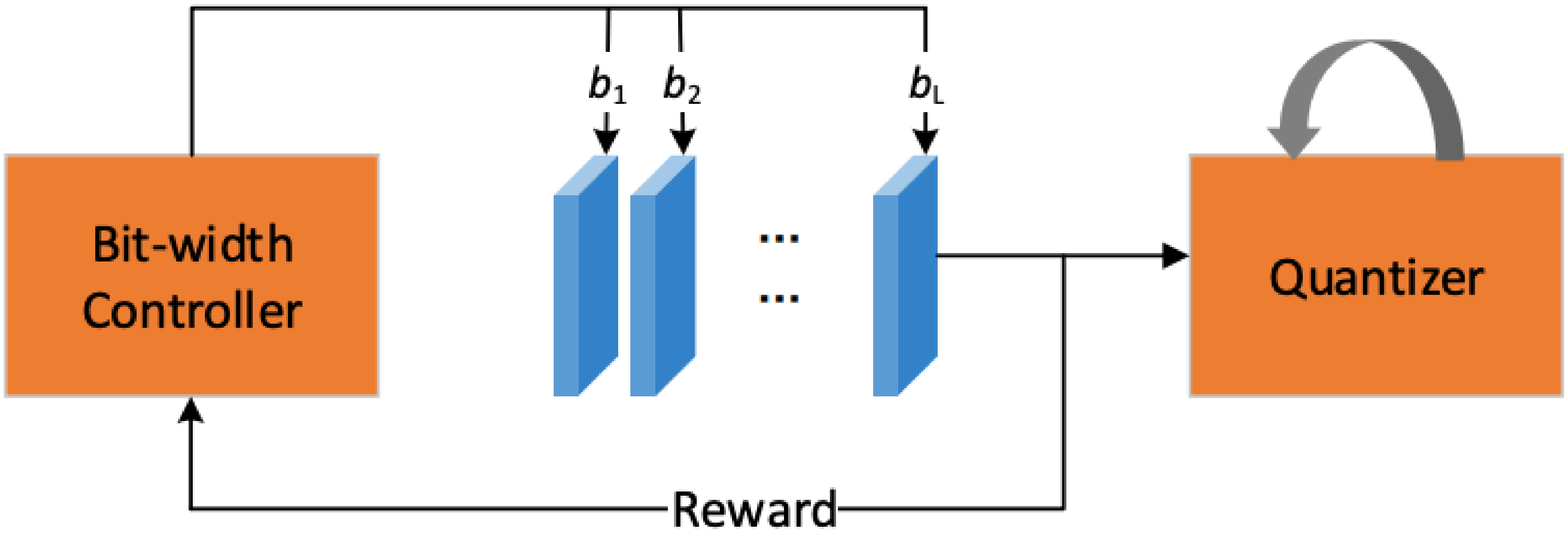
Quantization illustration [5]

Input-Aware Dynamic **Quantization** in Deep Neural Networks

- One of the important techniques is the **mixed-precision quantization of neural networks.**
- Different bit precision for different layers/blocks in the model
- Bit selection problem
- Current solutions in the literature use pre-defined, fixed bit widths for each layer, that can not be modified without retraining the model.

Input-Aware **Dynamic Quantization** in Deep Neural Networks

Dynamic quantization techniques aims to reduce memory and computation overhead at **run-time** through changing bit-widths during inference without retraining.

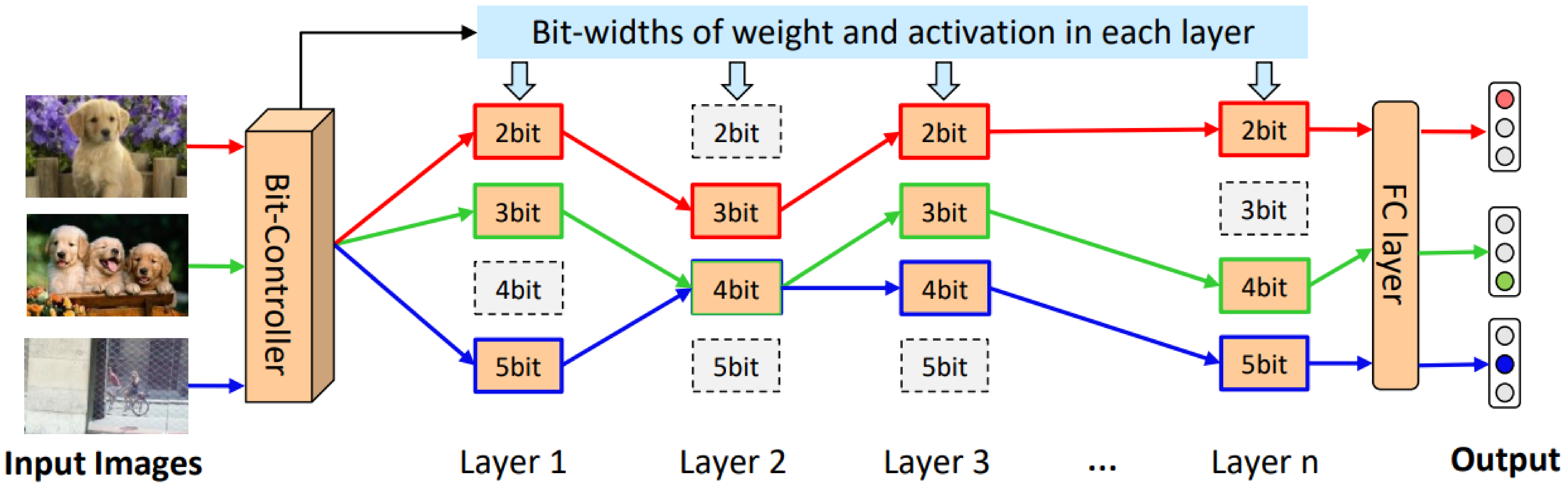


Dynamic Network Quantization Illustration [6]

Input-Aware Dynamic Quantization in Deep Neural Networks

Choosing optimal bit-widths for each layer by considering:

- Resource Availability
- Performance on the Task
- **Input of the Model**



Instance-aware dynamic quantization [7]

Main Related Works

- **AdaBits** were proposed to allow dynamically adjust bit precision of the model during inference, however **same precision for all layers** [9].
- **Bit-Mixer** focuses on choosing bit precisions for each layer on inference time considering the **resource availability and performance, not input** [10].
- Also, an **Instance-Aware DQNet** which consists of a **predictor bit controller network** is proposed. Mainly focusing on custom solutions with a specific Neural Net (ResNet) and not a generalized framework [11].

Further Ideas

- Different bit predictor network architectures integrated to model
- Better regularization metrics for input complexity when training the model
- Focusing on different patches of the input
- Layer statistics
- A framework without depending on specific neural net architecture

Research Plan

The goal of the research is to

- design
- implement
- deploy

a framework that enables **dynamic quantization concerning given input** in edge devices, by improving the current solutions in literature.

The nature of this research mainly lies between theory and application.

Steps and Goals

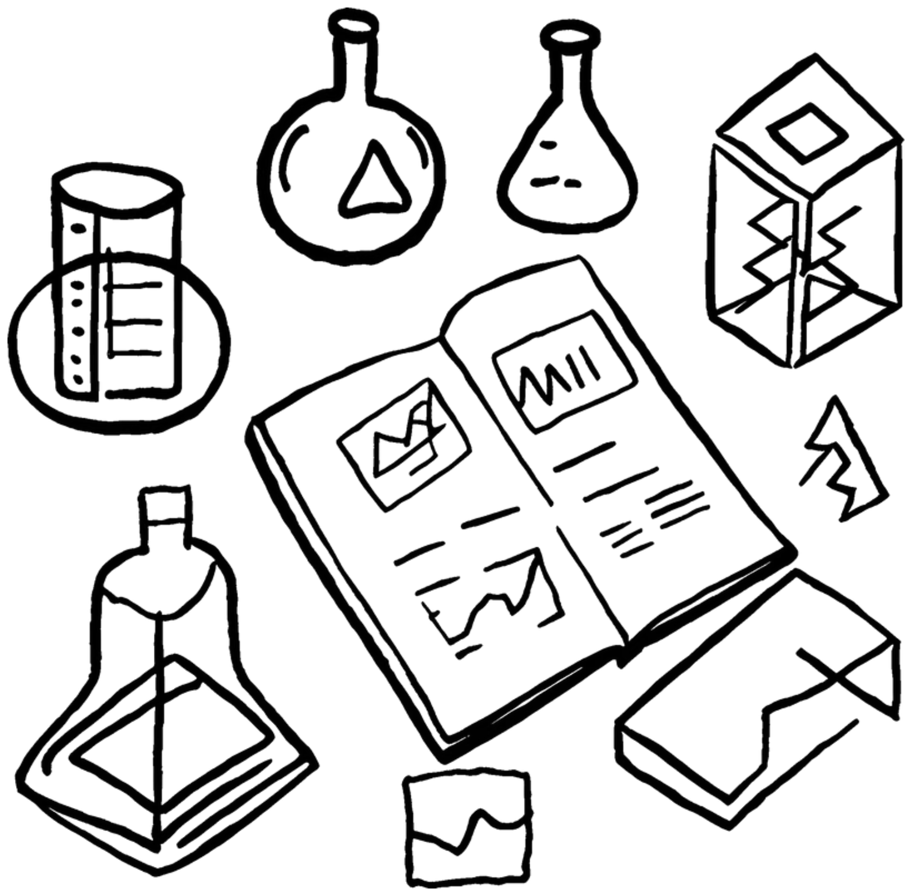
- Problem Formulation and SotA
- Literature Review
- Design
- Implementation
- Experiments
- Thesis Writing

Design / Implementation

1. Choosing the CNN architecture to be worked on (ResNet, MobileNet...).
2. Exploring efficient ways of **dynamic quantization to choose bit-widths of the layers based on the input** (bit predictor network integrated to model, possible input-aware architectures, patch complexity of input, layer statistics etc.) while **considering the challenges in the TinyML**.
3. Concretize different candidate solutions for the research question.
4. Firstly starting with a theoretical assumptions, then continuing synchronously with implementation.
5. A set of different solutions and versions are aimed to start experiments after the implementation and development of these candidate solutions.

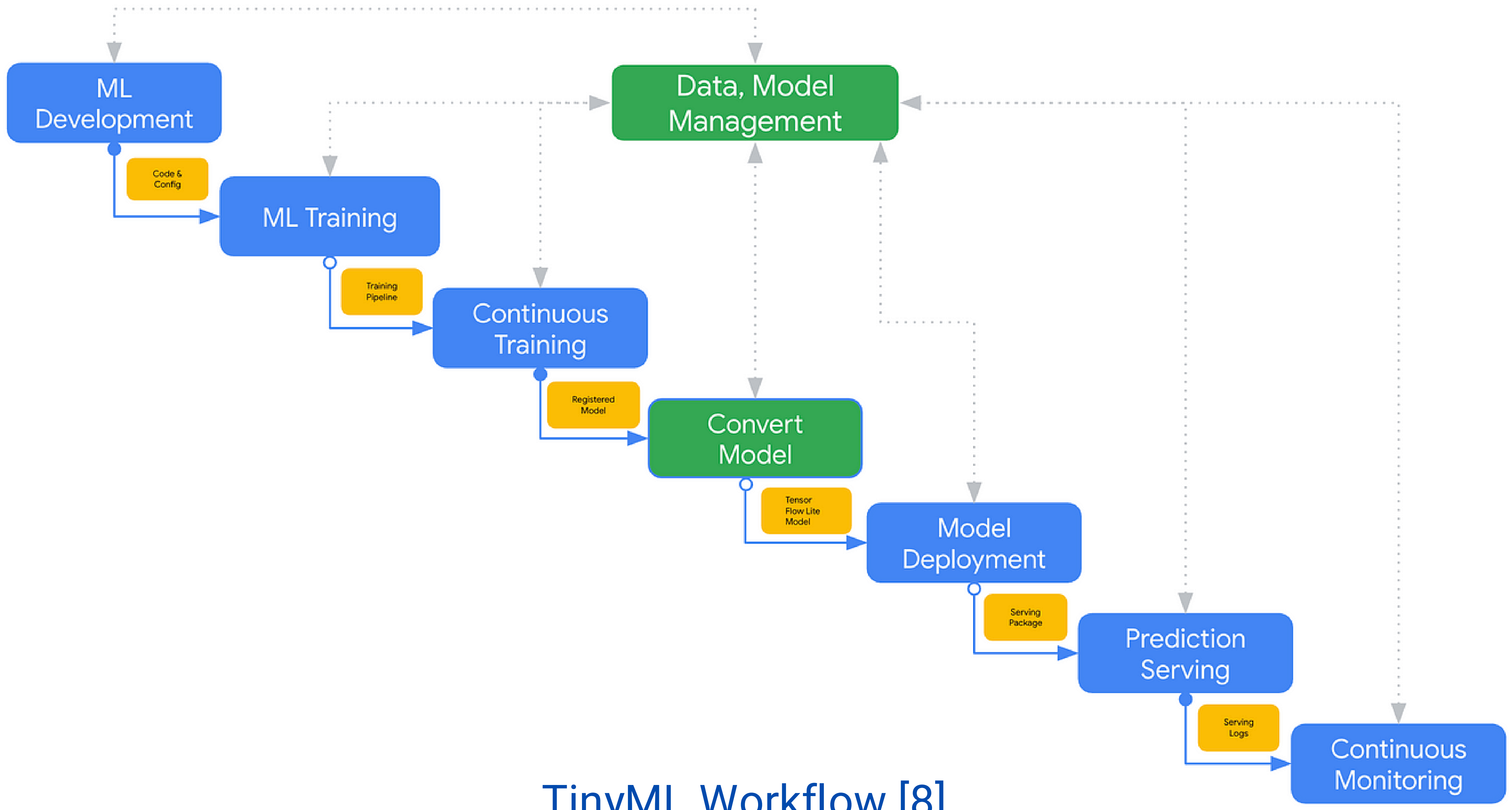
Experiments and Research Assessment

- Result collection and analysis, reimplementing solution with respect to their outcomes.
- In the thesis, the real concern is maintaining model success (accuracy, precision, AUC, etc) while reducing computation overhead.
    - Accuracy of classifier
    - FLOPs and MAC Operations
    - Inference Time
    - Memory Usage



Experiments and Research Assessment

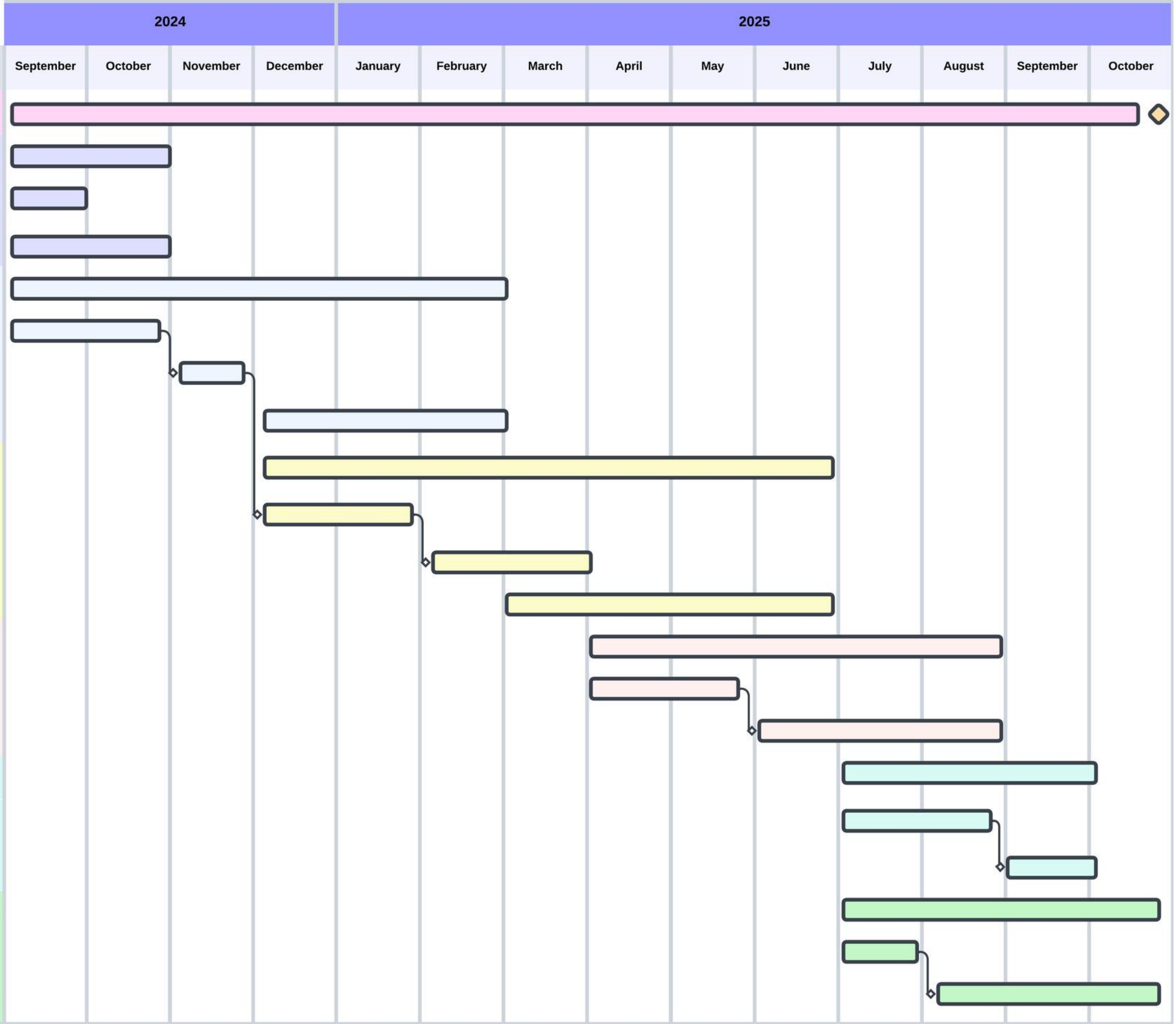
- Apart from this, a real deployment on a resource-constrained edge device will strengthen our assessment through **testing the developed solution in real-world scenarios**, which **differs from previous works.**



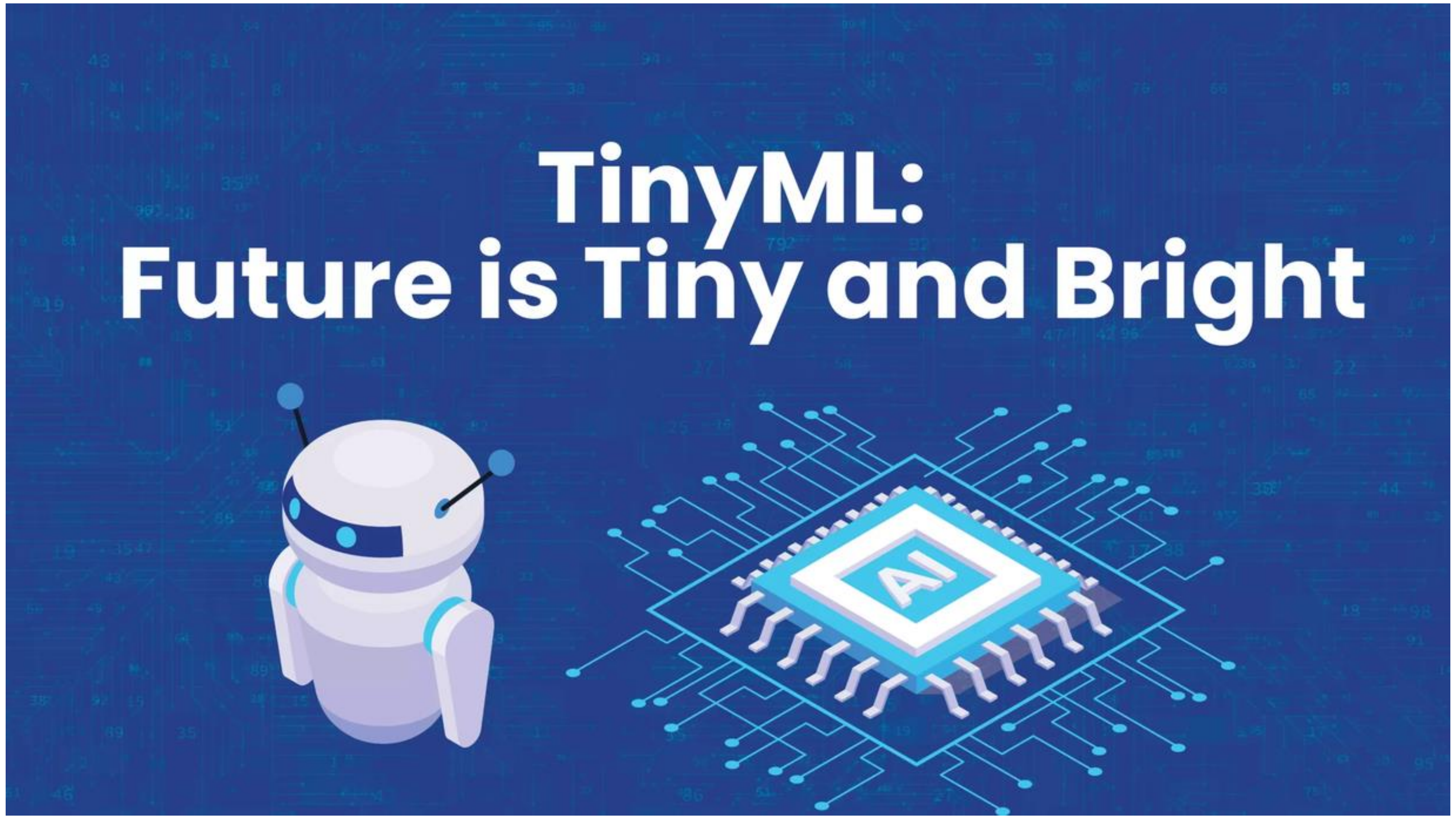
TinyML Workflow [8]

Research Plan

Tasks	Duration
Thesis	~12 Months
1 - Problem Analysis and SotA	~ 1-2 Months
1.1 - Defining research question, goals and constraints	1 Month
1.2 Defining SotA and what can be novel	2 Months
2 - Literature Review	~ 6 Months
2.1 - Analysis on current literature	~ 2 Months
2.2 - Working on possible novel research directions	~ 2 Months
2.3 - Staying up to date for new papers	~4 Months
3 - Design	~ 6-7 Months
3.1 - Working on proposed idea	2-3 Months
3.2 - Creating first version of solution	2-3 Months
3.2 - Continous design process parallel to implementation	4 Months
4 - Implementation	~ 6 Months
4.1 - Very first implementation of current solution	2 Months
4.2 - Adapting implementation to improved solution, fixing, re-designing etc.	4 Months
5 - Experiments	~2-3 Months
5.1 - Result collection and analysis, reimplementing solution w.r.t results	2 Months
5.2 - Evaluation and final fixes	1 Month
6 - Paper writing	~ 3-4 Months
6.1 - Arranging layout and structure	1 Month
6.2 - Writing	~ 2-3 Months



THANKS FOR YOUR ATTENTION!



References

1. Cavagnis L, https://leonardocavagnis.medium.com, 2024
2. MIT HAN LAB, https://www.youtube.com/@tinyML, 2024. [Accessed: October 2024].
3. C. -H. Wang, K. -Y. Huang, Y. Yao, J. -C. Chen, H. -H. Shuai and W. -H. Cheng, "Lightweight Deep Learning: An Overview," in IEEE Consumer Electronics Magazine, vol. 13, no. 4, pp. 51-64, July 2024, doi: 10.1109/MCE.2022.3181759.
4. Grootendorst M, https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization, 2024. [Accessed: October 2024]
5. Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., and Blankevoort, T. A white paper on neural network quantization. ArXiv abs/2106.08295 (2021).
6. Xu, Yuhui et al. “DNQ: Dynamic Network Quantization.” 2019 Data Compression Conference (DCC) (2018): 610-610.
7. Liu, Z., Wang, Y., Han, K., Ma, S., and Gao, W. Instance-aware dynamic neural network quantization. 2022, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022), 12424–12433.
8. Suzuki L, https://larissa-suzuki.medium.com/a-very-short-introduction-to-mlops-for-tinyml-part-1-40432708b974, 2021 [Accessed: October 2024]
9. Jin, Q., Yang, L., and Liao, Z. A. Adabits: Neural network quantization with adaptive bit-widths. 2020IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019), 2143–2153.
10. Bulat, A., and Tzimiropoulos, G. Bit-mixer: Mixed-precision networks with runtime bit-width selection. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021), 5168–5177.
11. Liu, Z., Wang, Y., Han, K., Ma, S., and Gao, W. Instance-aware dynamic neural network quantization. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022), 12424–12433.