# Advancing Micromobility safety by deploying compressed Lane Recognition CNN model on low-spec Microcontroller Unit

**Mr Chinmaya Kaundanya, Dr Suzanne Little**
**Dublin City university, Ireland**

**Insight** SFI RESEARCH CENTRE FOR DATA ANALYTICS

TINY ML

luna **Urban Mobility Platform**

## Introduction

- The rise of **Micromobility** as a primary mode of transportation across the globe has brought about the need for effective **rules and regulations** governing the usage of E-scooters and E-bikes
- **GPS and sensor** technologies struggle to provide high-level precision and adaptation for real-time lane recognition in variable road structures
- LASERS and LIDARS can be considered alternative solutions for lane recognition; however, they are often **computationally expensive** and not ideal for all situations
- The limited and **constrained** Micromobility **environment** makes it impossible to utilize efficient AI-based solutions which require high computational resources
- To address these constraints, it's necessary to introduce AI solutions into low-spec platforms such as FPGAs or Microcontrollers
- A custom **Lane Recognition** dataset has been prepared that focuses on different lane segments such as **road**, **bikelane** and **sidewalk** specifically tailored for E-scooter rides
- A light-weight Convolutional Neural Network architecture is used to train a Lane Recognition model followed by different model compression techniques to deploy on low-spec Microcontroller Unit
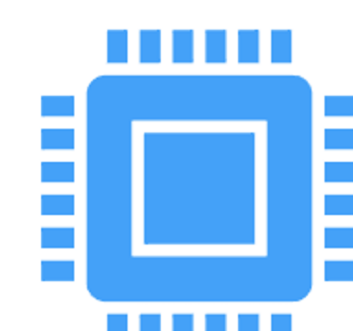


*Fig 1. The concept of Lane Recognition for Micromobilty users*

## Need of Compression
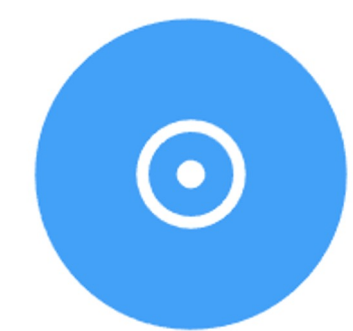
- Typically, the ANN models are **huge in size** (~100 Mbs-100 Gbs)
- Microcontrollers have **extremely low- Storage** capacity, **RAM**, **Processing** power
- Human brain works more efficiently with **1000 times less** neurons than in ANN models

## Platform

Processor: **32-Bit Arm Cortex-M7 @400MHz**
SRAM: **1 MB**
Flash Memory: **2 MB**
Power usage: **< 150-mA**
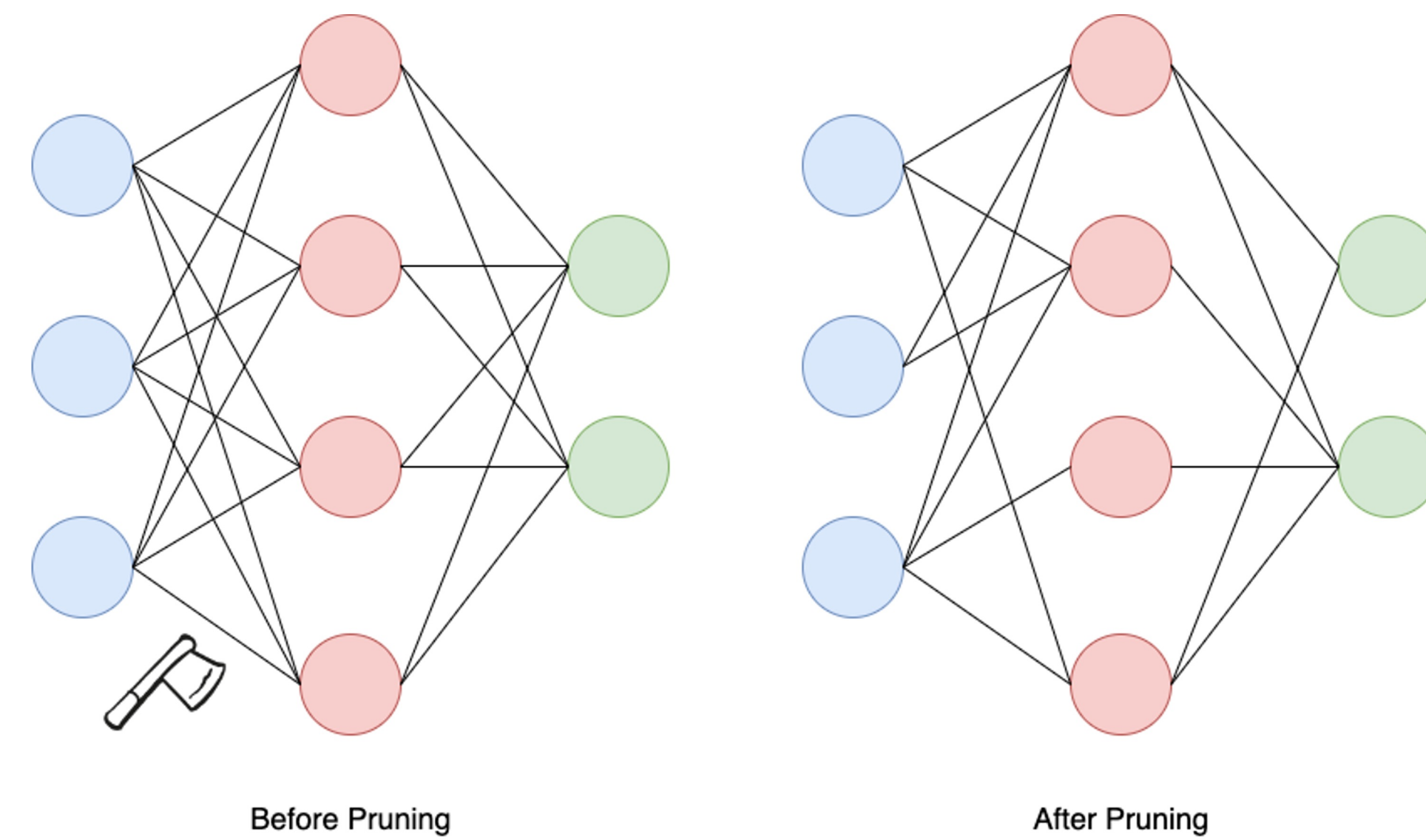Camera: **OV7725 (640X480)RGB**

## Weight Pruning



*Fig 2. The high-level idea of weight pruning in Neural Networks*

- A technique used to reduce model complexity and improve its efficiency by removing computational units in an Artificial Neural Network
- All computational units in the network are necessary **only during the training** for adjusting the weights to minimize the loss function
- A neural network is often **over-parameterised** for the inference
- The importance of a connection is determined by the **magnitude of the weight**, with **higher weights** indicating a **stronger influence** on the output of the network
- To reduce the size of the network, **connections** with **low weights** are removed, as they are deemed to have a minimal effect on the output of the network.
- Once weight pruning is complete, the final compressed ANN can be deployed on microcontrollers or other embedded devices

### Unstructured pruning

- A weight pruning technique which reduces model complexity by selectively removing **individual** weights
- Identify and remove less important weights based on a **specific criterion**
- Typically, the criterion is **low-magnitude weights**, as they have a smaller impact on the model's output

### Advantages

- **Compatibility:** The resulting models are sparse and tiny which are ultimately less reliant on GPUs but efficient for low-spec CPU platforms
- **Flexibility:** Selective removal of weights without being constrained by the model's architecture makes it adaptable to a wider range of network designs
- **Fine-Tuning:** Gradually pruning and retraining the model allows for recovery of the lost accuracy due to pruning

### Disadvantages

- **Not optimal:** Many existing hardware accelerators and deep learning frameworks are optimized for dense models, making it challenging to efficiently execute sparse models produced by unstructured pruning
- **Irregular sparsity patterns:** The presence of irregular sparsity patterns may not be efficiently exploited by specialized hardware or software, limiting potential speedup or memory savings
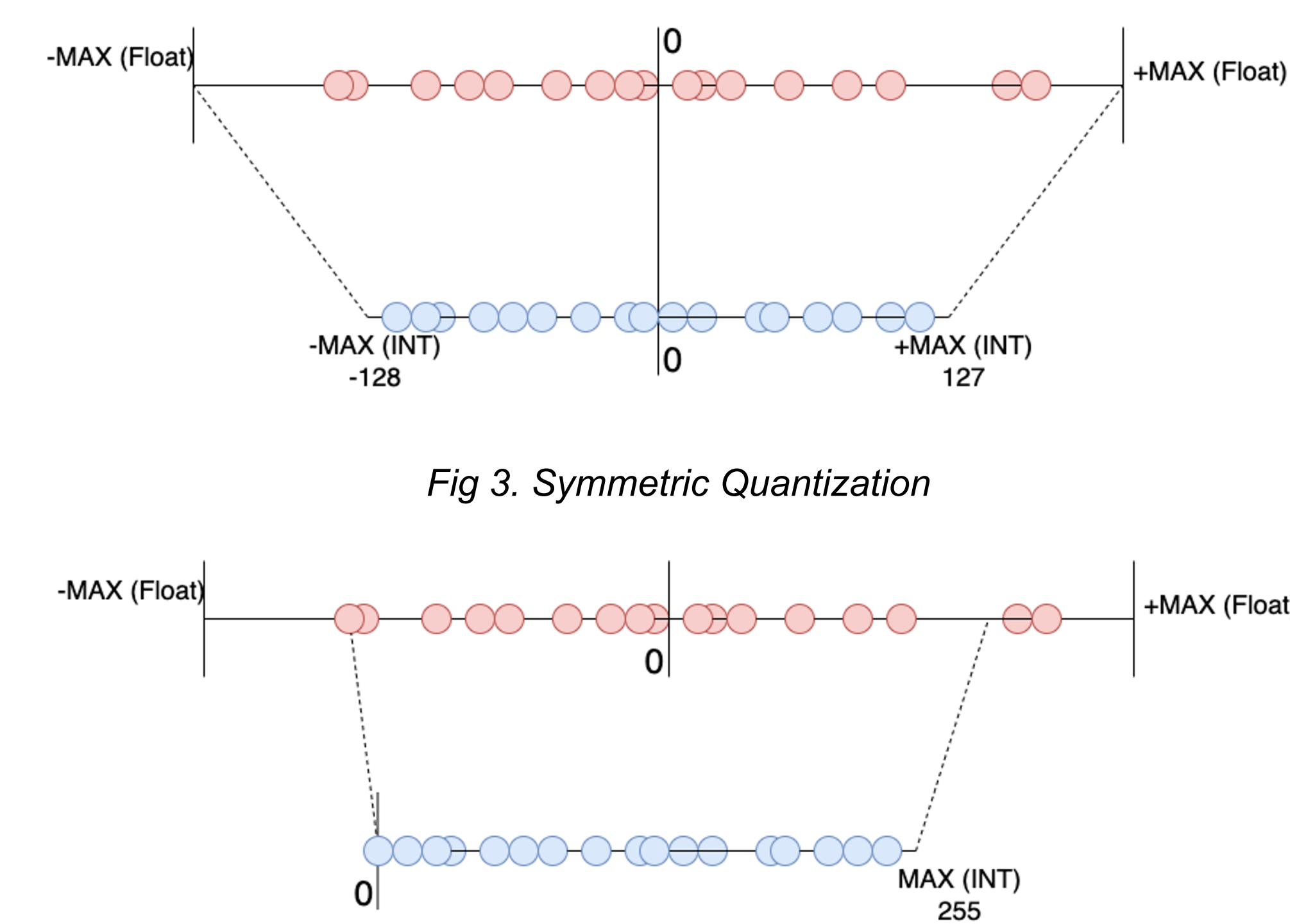
## Quantization



*Fig 3. Symmetric Quantization*



*Fig 4. Asymmetric Quantization*

### Affine Quantization

- Converting **high precision** (e.g., Float32) **weights** and **activations** of the network to the **low precision** (e.g., INT8) which leads to reduced memory footprint and faster computations
- The first step is **mapping** the **original** weights/activations to the desired **integer range** and the second step involves **rounding** them to the nearest integer value
- Quantization can be applied **during training** (quantization-aware training) or **after training** (post-training quantization), with the former generally yielding better performance

### Sparsity preserving Quantization Aware Training (QAT)

- QAT reduces the loss of accuracy when transitioning from floating-point to lower precision, **simulating** low-precision behaviour in the forward pass while maintaining the same backward pass
- The **quantization error** generated during the forward pass **accumulates** in the model's overall loss, which encourages the optimizer to modify parameters accordingly
- Sparsity Aware QAT is a special case of QAT which combines the advantages of **both weight pruning** (sparsity) and **quantization** to compress and optimize neural networks for efficient inference
- Experimental results indicated that the sparsity achieved during pruning is **not maintained** when using standard **QAT**; however, it is **preserved** when employing **Sparsity Aware QAT**

## Comparative Analysis

| Model | Test Accuracy | Test F1 Score | Test AUC | Model Size |
|---|---|---|---|---|
| Float32 Model | 74.08 | 64.21 | 91.07 | **1.4 MB** |
| Pruned Model | 76.06 | 65.46 | 93.18 | **782 KB** |
| Quantized int8 TFLite model | 74.28 | **66.18** | 92.73 | **218 KB** |

*Table 1. Comparison of performance and size between different models*
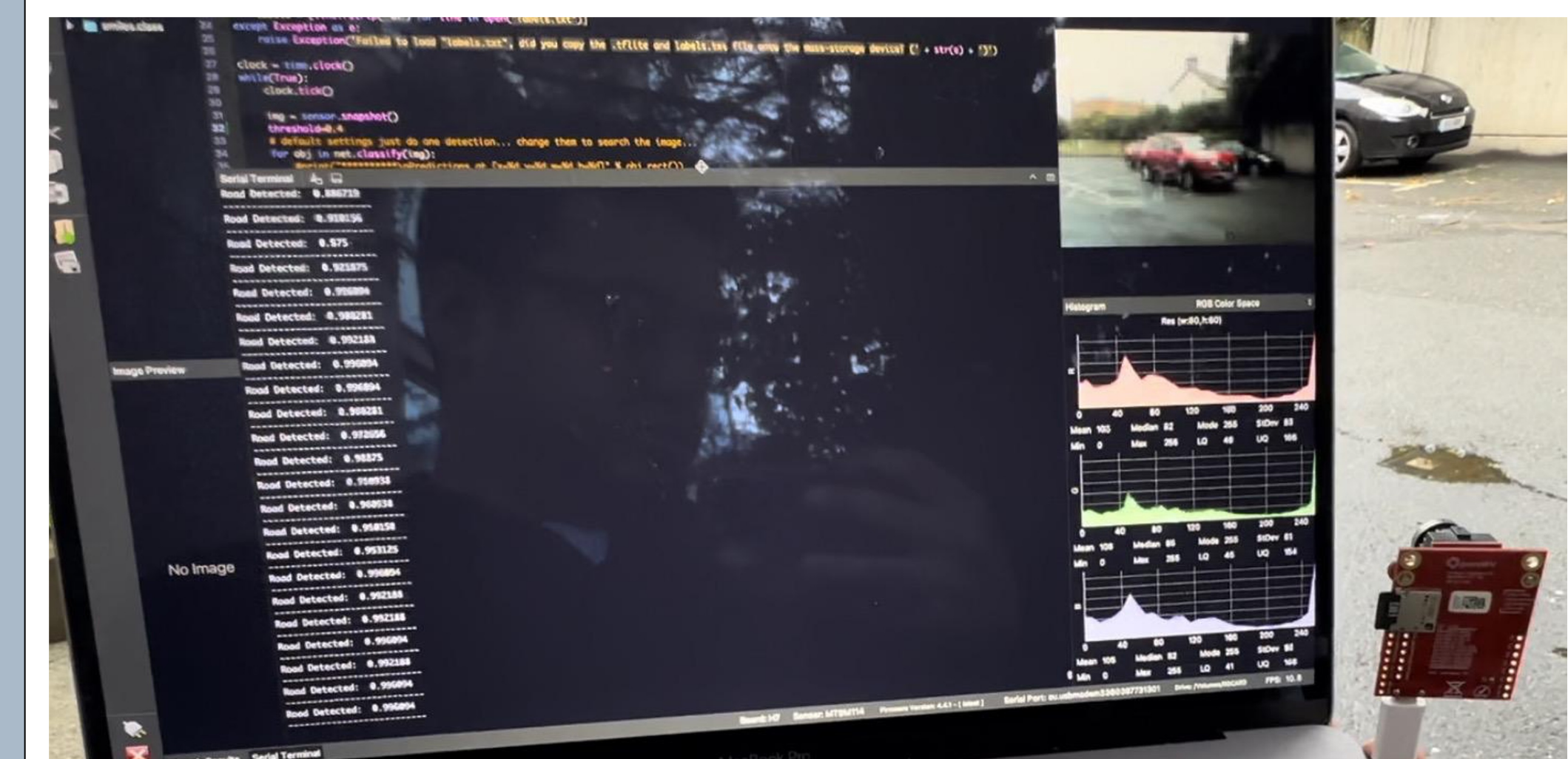
## Results



*Fig 5. 'Road Detected' after pointing the camera to the road area with 12.4 FPS and 17.09 M FLOPS*
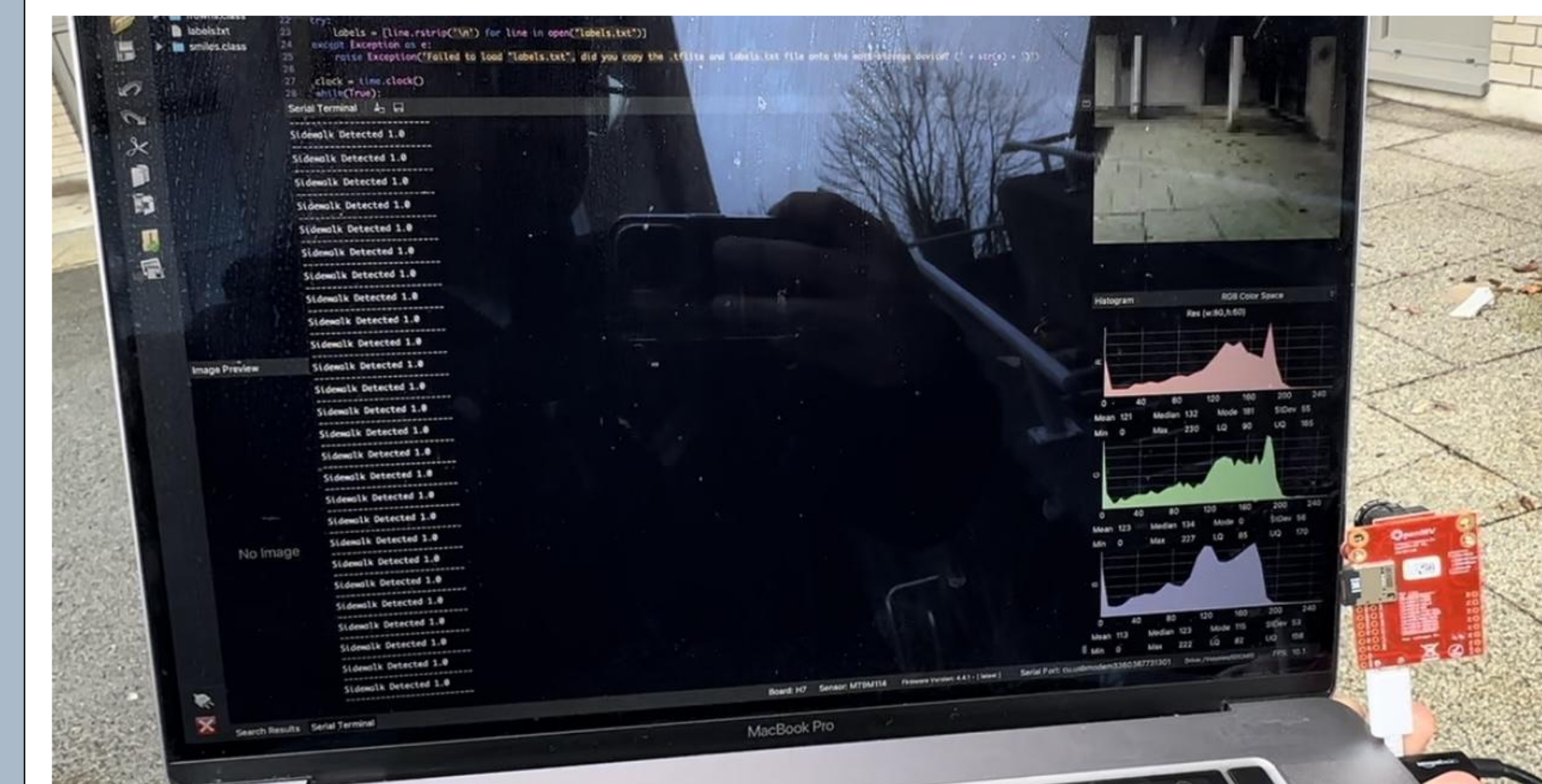


*Fig 6. 'Sidewalk Detected' after pointing the camera to the sidewalk area*

## Conclusions

- The growing popularity of Micromobility solutions necessitates the development of robust and efficient lane recognition systems
- Model compression is essential due to the computational constraints within these vehicles
- Weight pruning in CNNs presents a straightforward yet effective method for both reducing model size and its optimization
- Further compression can be achieved with Sparsity Aware QAT with a minimal trade-off between model size and performance
- Experimental results showed considerable model size reduction of **45.43%** from the **Float32** model to the pruned model and a **72.08%** reduction from the pruned to the **INT8** quantized model

## Acknowledgements

## References

1. Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
2. Zhu, Michael, and Suyog Gupta. "To prune, or not to prune: exploring the efficacy of pruning for model compression." *arXiv preprint arXiv:1710.01878* (2017).
3. Klauder, John R. "The benefits of affine quantization." *arXiv preprint arXiv:1912.08047* (2019).
4. Shomron, Gil, et al. "Post-training sparsity-aware quantization." *Advances in Neural Information Processing Systems* 34 (2021): 17737-17748.