SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL, AND
MANAGEMENT ENGINEERING ANTONIO RUBERTI

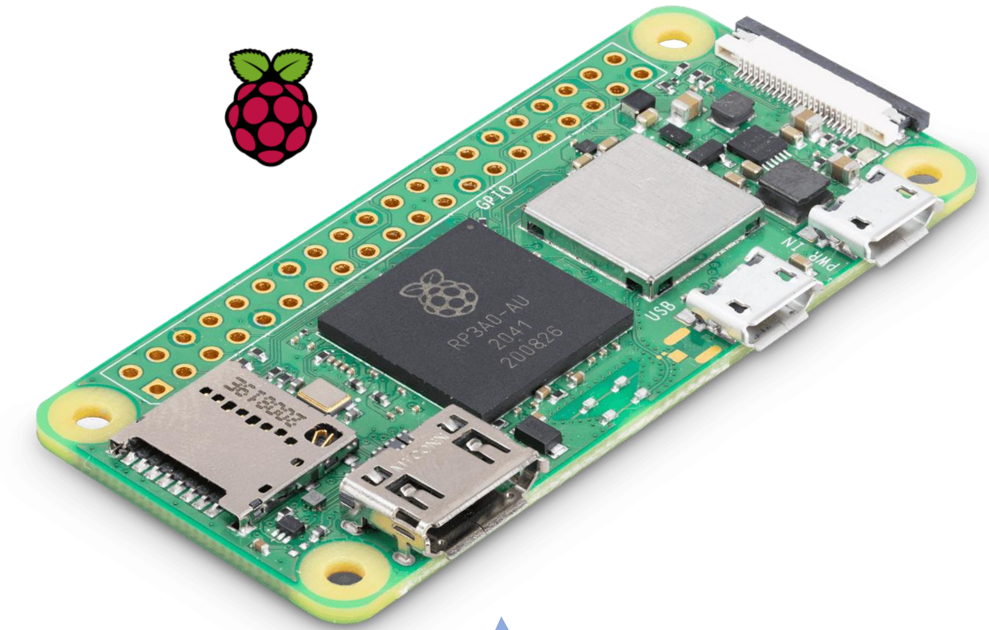
SAPIENZA
UNIVERSITÀ DI ROMA

# Computer Vision

## Real-Time Monocular Depth Estimation in low resources devices.

Damiano Imola – 2109063
Prof: Irene Amerini

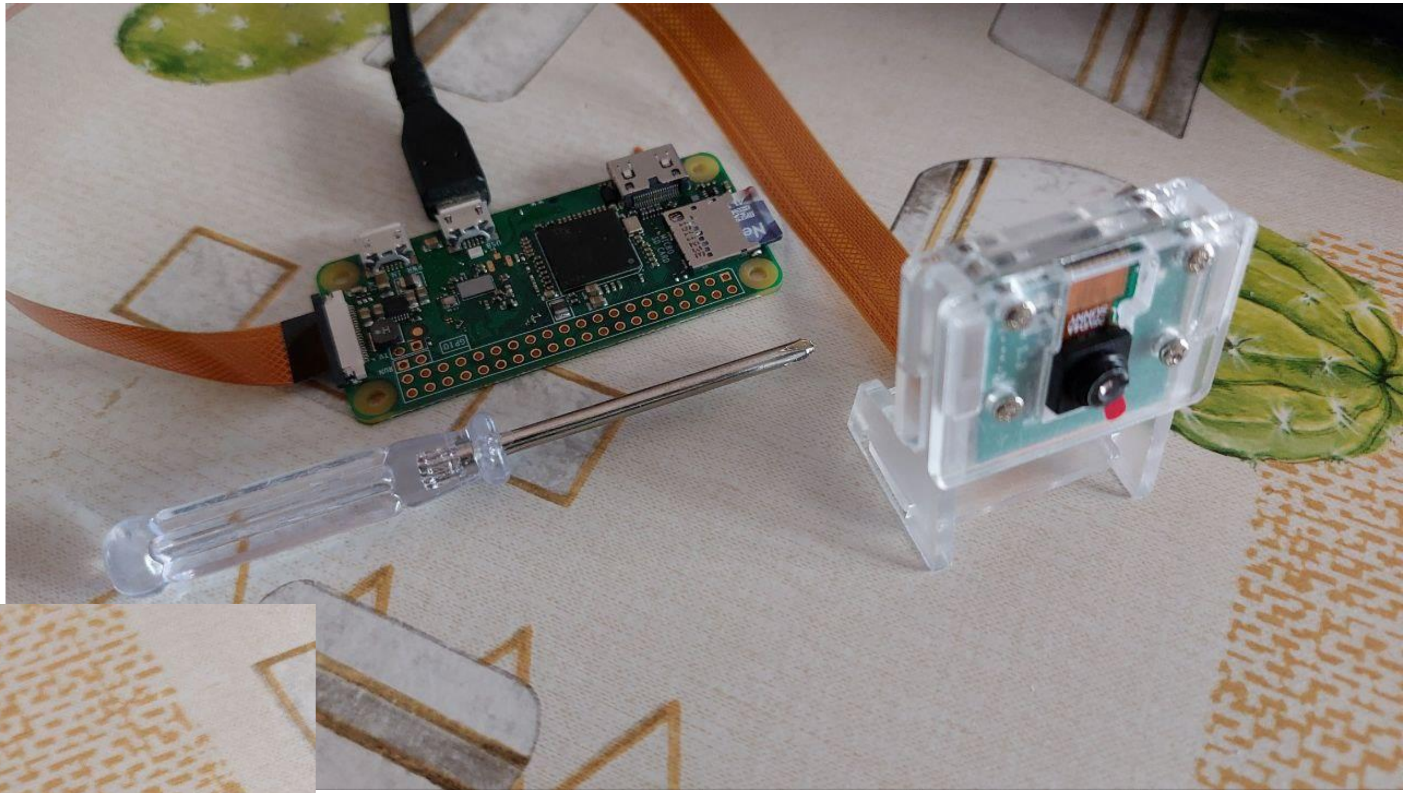
# The Fail




Raspberry Pi
Zero W

- **1GHz single-core CPU**
- **512MB RAM**
- Mini HDMI® port
- Micro USB OTG port
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector (v1.3 only)

# The Fail



Raspberry Pi Zero W camera stand

Original Raspberry Pi Zero W camera cover

# The Fail

```
(cv_venv) raspberry@raspberrypi:~/cv_project $ pip install tensorflow
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
ERROR: Could not find a version that satisfies the requirement tensorflow (from versions: none)
ERROR: No matching distribution found for tensorflow
```

```
(cv_venv) raspberry@raspberrypi:~/cv_project $ pip install onnxruntime
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
ERROR: Could not find a version that satisfies the requirement onnxruntime (from versions: none)
ERROR: No matching distribution found for onnxruntime
```

```
(cv_venv) raspberry@raspberrypi:~/cv_project $ pip install tflite-runtime
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
ERROR: Could not find a version that satisfies the requirement tflite-runtime (from versions: none)
ERROR: No matching distribution found for tflite-runtime
```

```
(cv_venv) raspberry@raspberrypi:~/cv_project $ pip install torch
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
ERROR: Could not find a version that satisfies the requirement torch (from versions: none)
ERROR: No matching distribution found for torch
```

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza Università di Roma

# Computer Vision

## MonoDeRT:
## A novel light-weight Real-Time architecture for Monocular Depth Estimation.

Damiano Imola – 2109063
Prof: Irene Amerini

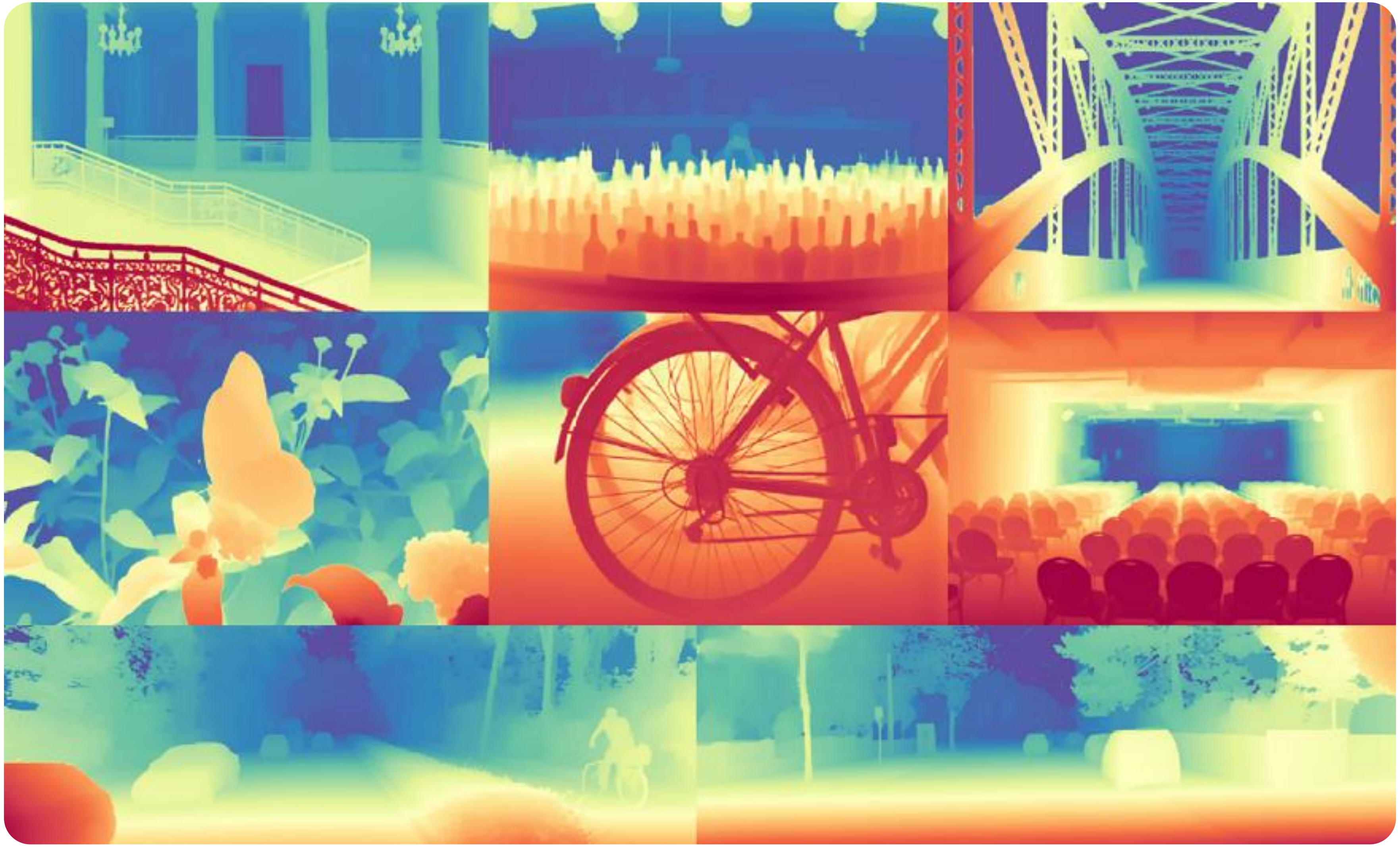
1. Introduction
2. Related works
3. Proposed methods
4. Dataset and metrics
5. Experimental results
6. Conclusion

# 1. Introduction

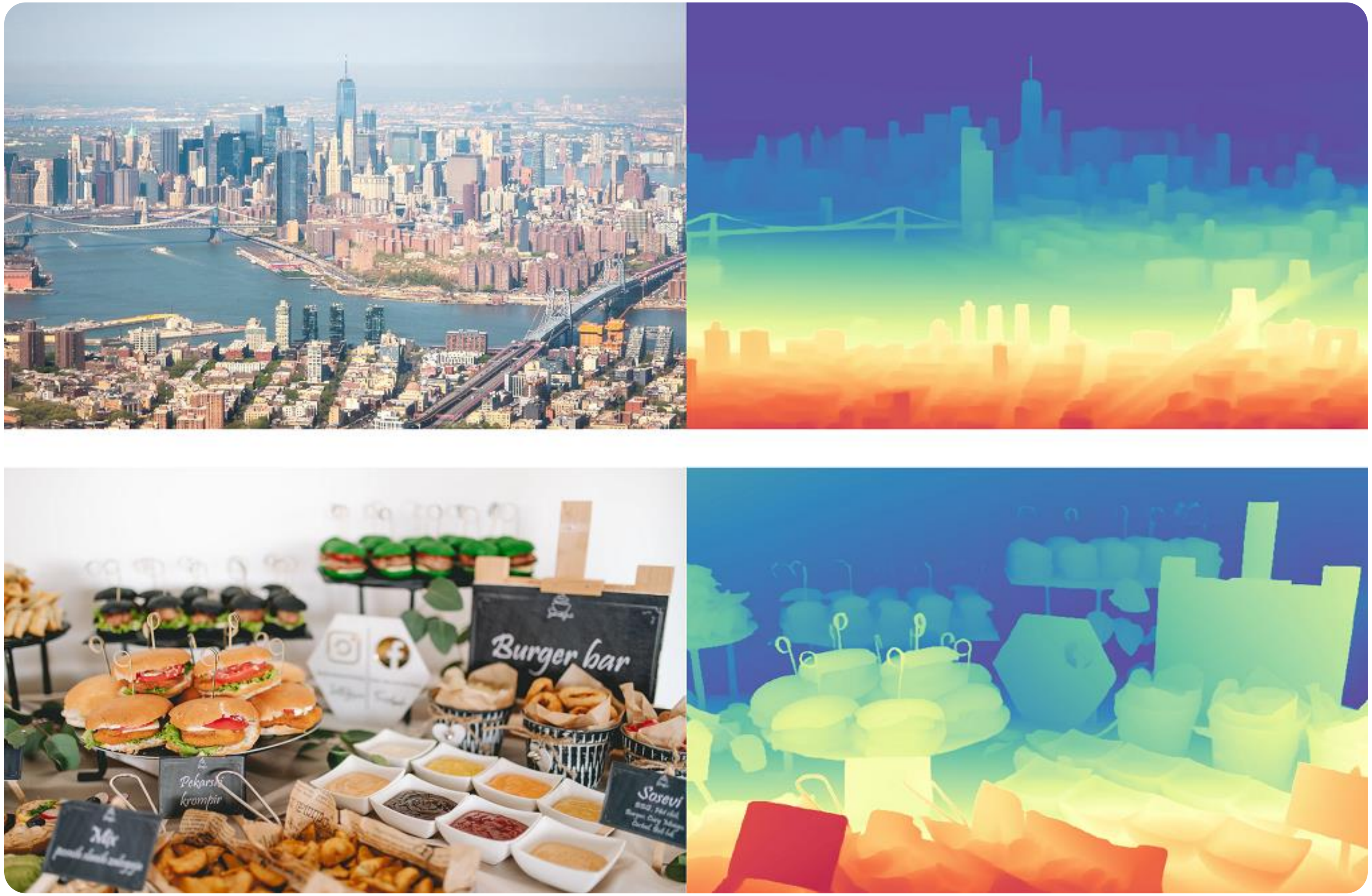
2. Related works
3. Proposed methods
4. Dataset and metrics
5. Experimental results
6. Conclusion

# Introduction – The ill-posed problem

# Introduction – The applications



# Introduction – Challenges (1)



# Introduction – Challenges (2)

# Introduction – Challenges (3)

1. Introduction
2. **Related works**
3. Proposed methods
4. Dataset and metrics
5. Experimental results
6. Conclusion

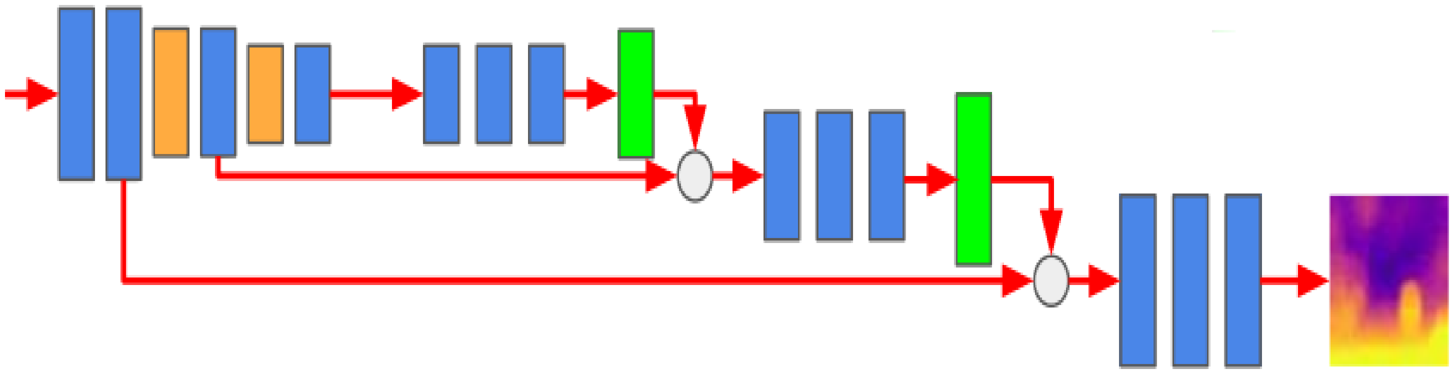
# Related works

- µPyD-Net (MCUs)

Size: 32x32 then super resolution.

1 FPS with 512KB Ram

- RT MonoDepth



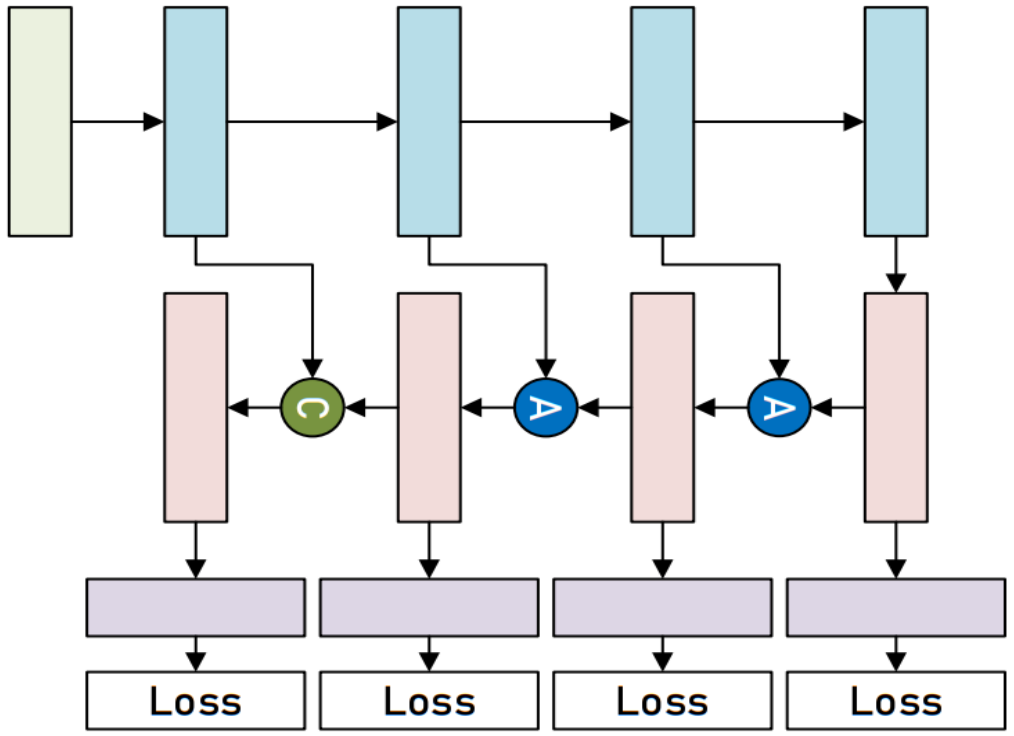
- MonoDepth2

# Related works

- µPyD-Net (MCUs)
- RT MonoDepth
- MonoDepth2

Additional "-S" variant.

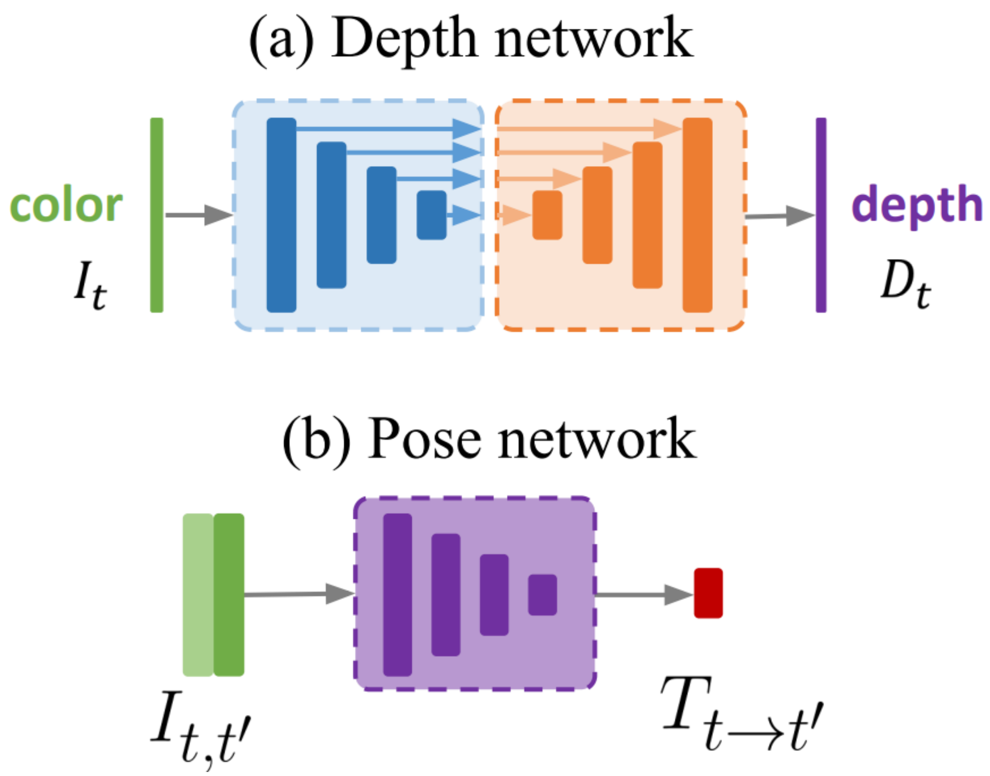
18.4&30.5 FPS on NVIDIA Jetson Nano



# Related works

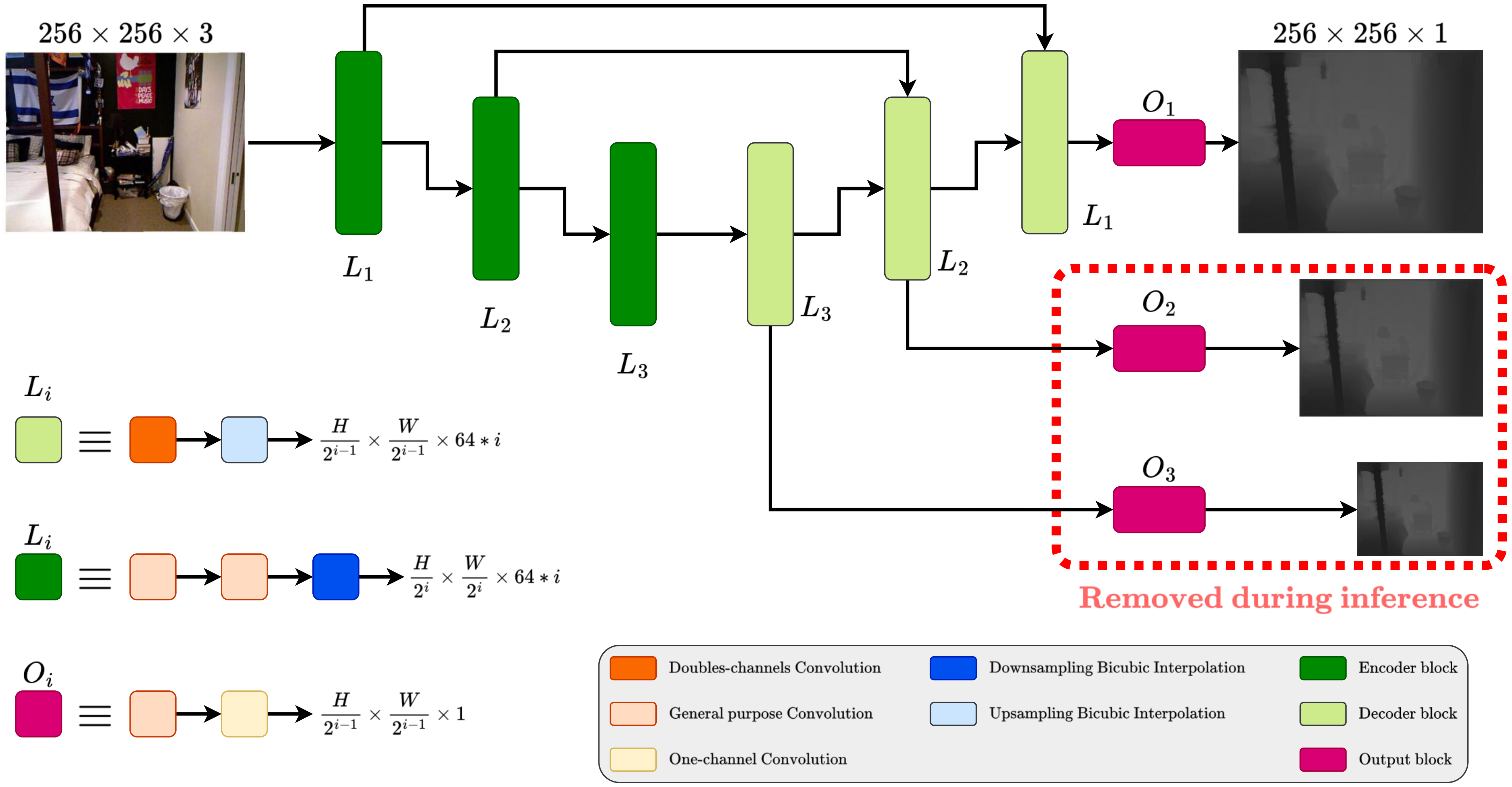
- µPyD-Net (MCUs)
- RT MonoDepth
- MonoDepth2

Leverages self-supervision
Developed new occlusion handling method and Edge-Aware Smoothness Loss.

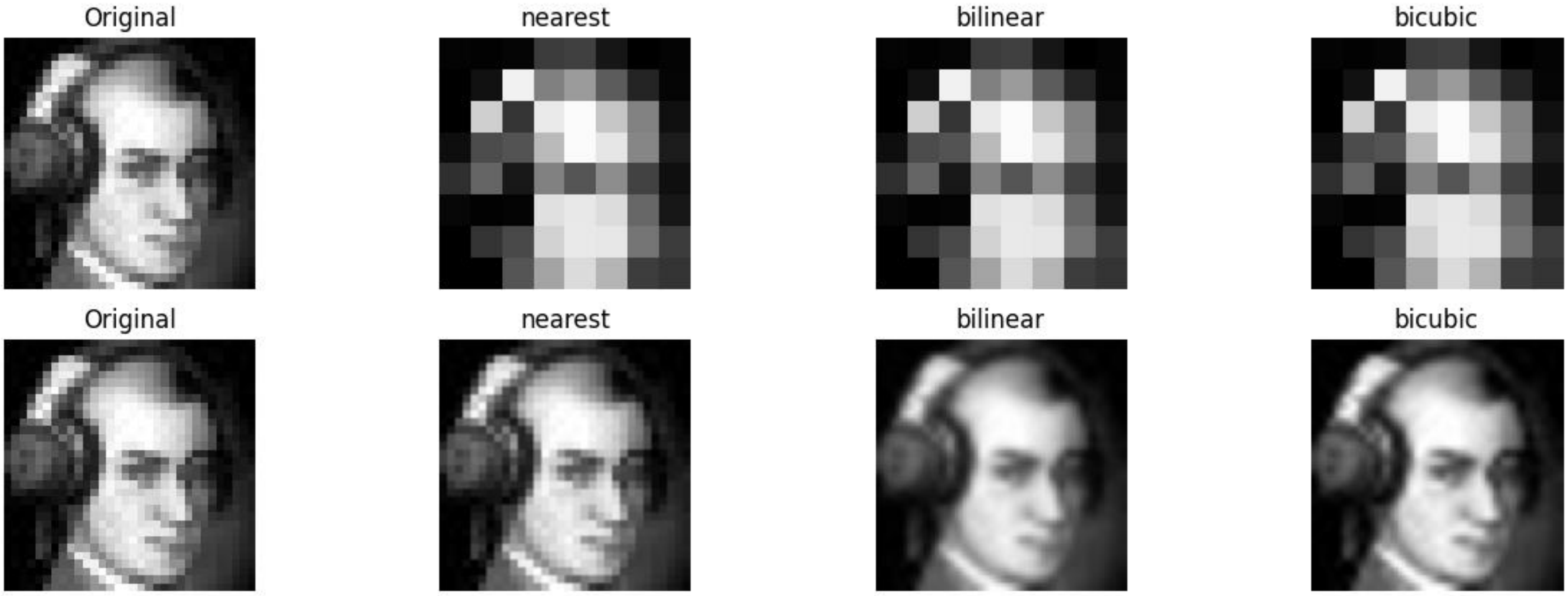


1. Introduction
2. Related works
3. **Proposed methods**
4. Dataset and metrics
5. Experimental results
6. Conclusion

# Proposed method: MonoDeRT

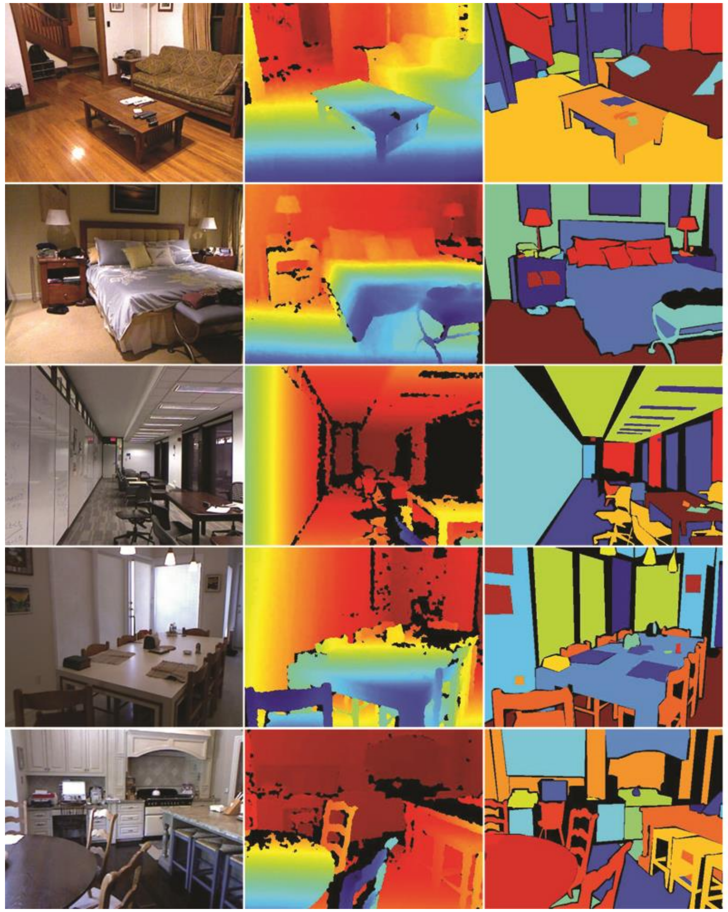


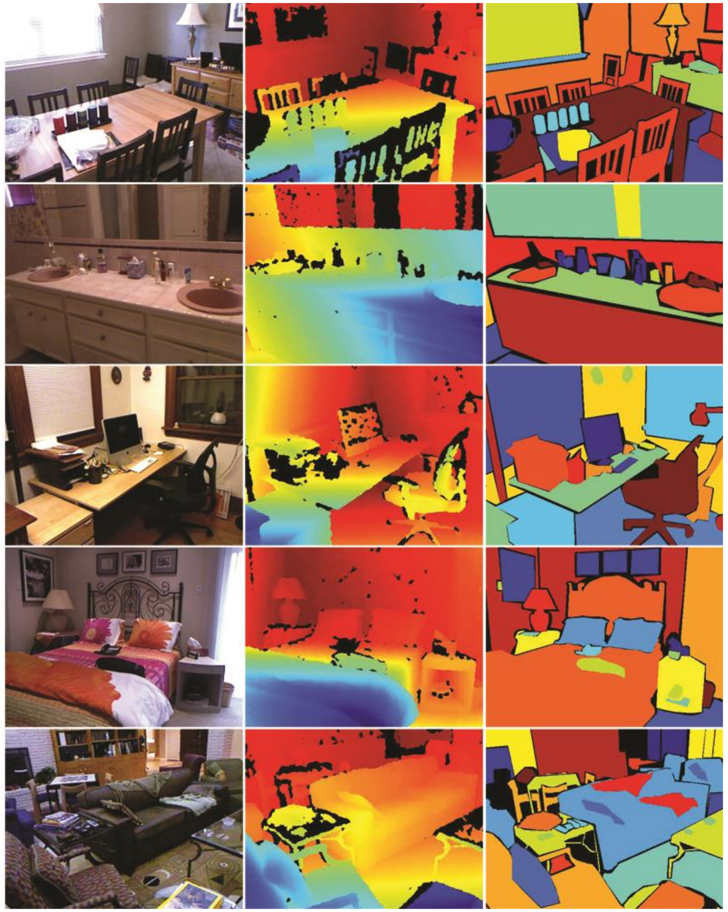
# Down-/Up-sampling with interpolation



1. Introduction
2. Related works
3. Proposed methods
4. **Dataset and metrics**
5. Experimental results
6. Conclusion

# Dataset: NYU Depth Dataset V2





**~408.000** total samples

Trained against **~50.000** samples

# Losses

- **SiLog** (Scale-Invariant Logarithmic error)
  Absolute scale of depth values can vary widely depending on the scene.

$$\mathrm{SiLog}(y, y^*) = \frac{1}{n}\sum_i d_i^2 + \frac{\lambda}{n}\left(\sum_i d_i\right)^2$$

$$d_i = \log(y_i) - log(y_i^*)$$

- BerHu (Reverse Huber)

- SSIM (Structural similarity index measure)

# Losses

- SiLog (Scale-Invariant Logarithmic error)

- **BerHu** (Reverse Huber)

Particularly effective in managing outliers, which are common in depth estimation tasks.

$$\mathrm{BerHu}(d) = \begin{cases} |d| & \text{if } |d| \leq c \\ \frac{d^2+c^2}{2c} & \text{if } |d| > c \end{cases}$$

- SSIM (Structural similarity index measure)

# Losses

- SiLog (Scale-Invariant Logarithmic error)
- BerHu (Reverse Huber)

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

- **SSIM** (Structural similarity index measure)

Measures the perceptual similarity between the predicted and ground truth depth maps. To make prediction aligned with human perception.

# Metrics

$$\mathrm{RMSE}(y,\hat{y}) = \sqrt{\frac{1}{n}\sum_i |y_i - \hat{y}_i|^2}$$

$$\mathrm{MAE}(y,\hat{y}) = \frac{1}{n}\sum_i |y_i - \hat{y}_i|$$

$$\mathrm{RMSE}_{\log}(y,\hat{y}) = \sqrt{\frac{1}{n}\sum_i |\log_{10}(y_i) - \log_{10}(\hat{y}_i)|^2}$$

$$\mathrm{MAE}_{\log}(y,\hat{y}) = \frac{1}{n}\sum_i |\log_{10}(y_i) - \log_{10}(\hat{y}_i)|$$

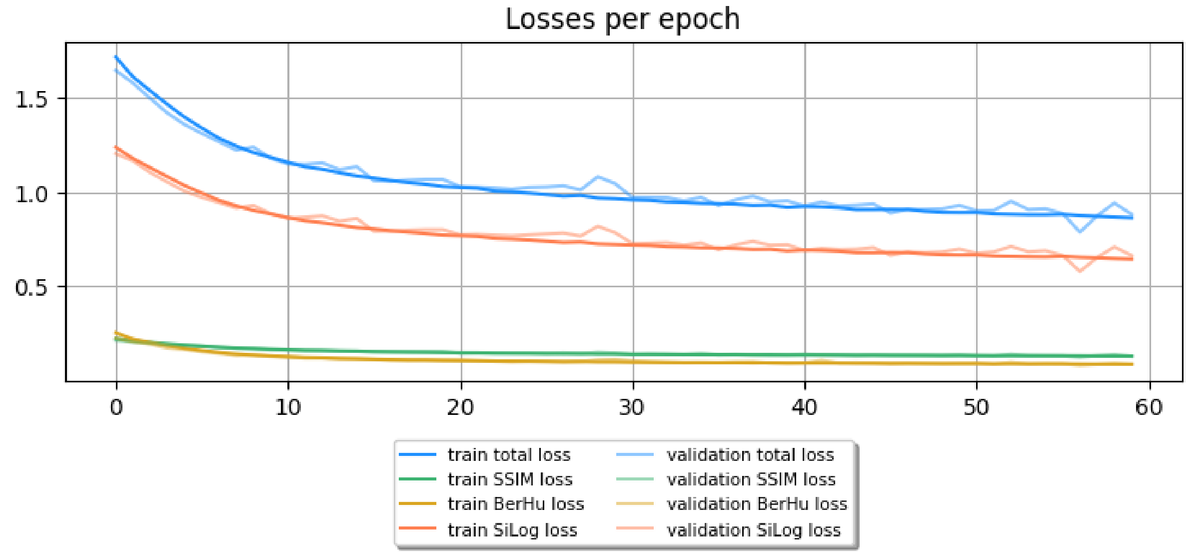
$$\mathrm{SqRel}(y,\hat{y}) = \frac{1}{n}\sum_i \frac{|y_i - \hat{y}_i|^2}{y_i}$$

$$\mathrm{AbsRel}(y,\hat{y}) = \frac{1}{n}\sum_i \frac{|y_i - \hat{y}_i|}{y_i}$$

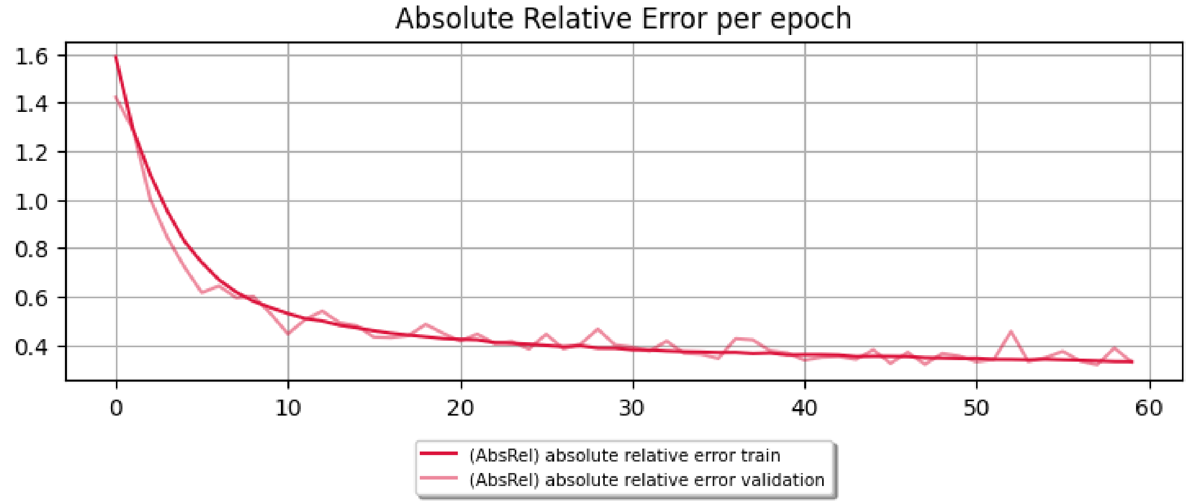
$$\delta_j(y,\hat{y}) = \%\text{ of } y_i \text{ s.t. } \max\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right) = \delta < thr, \quad i = 1,2,3 \quad thr = 1.25^i$$

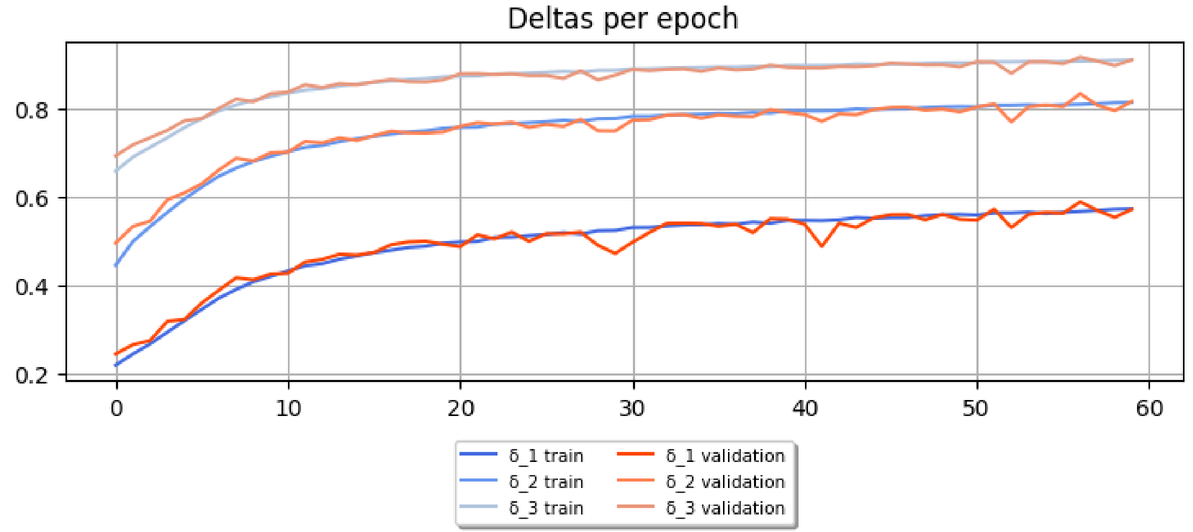
1. Introduction
2. Related works
3. Proposed methods
4. Dataset and metrics
5. **Experimental results**
6. Conclusion

# Training and validation metrics trend

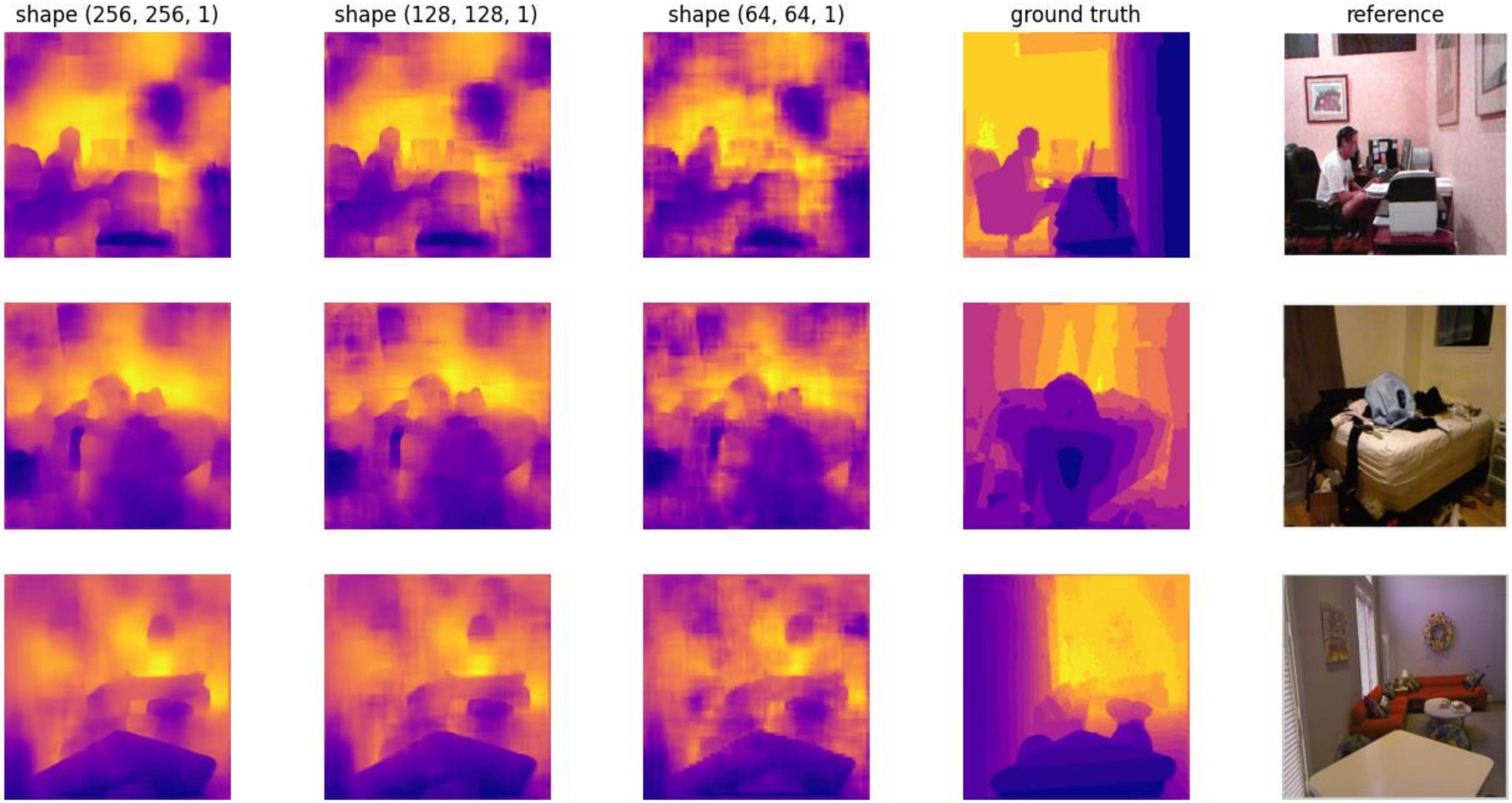




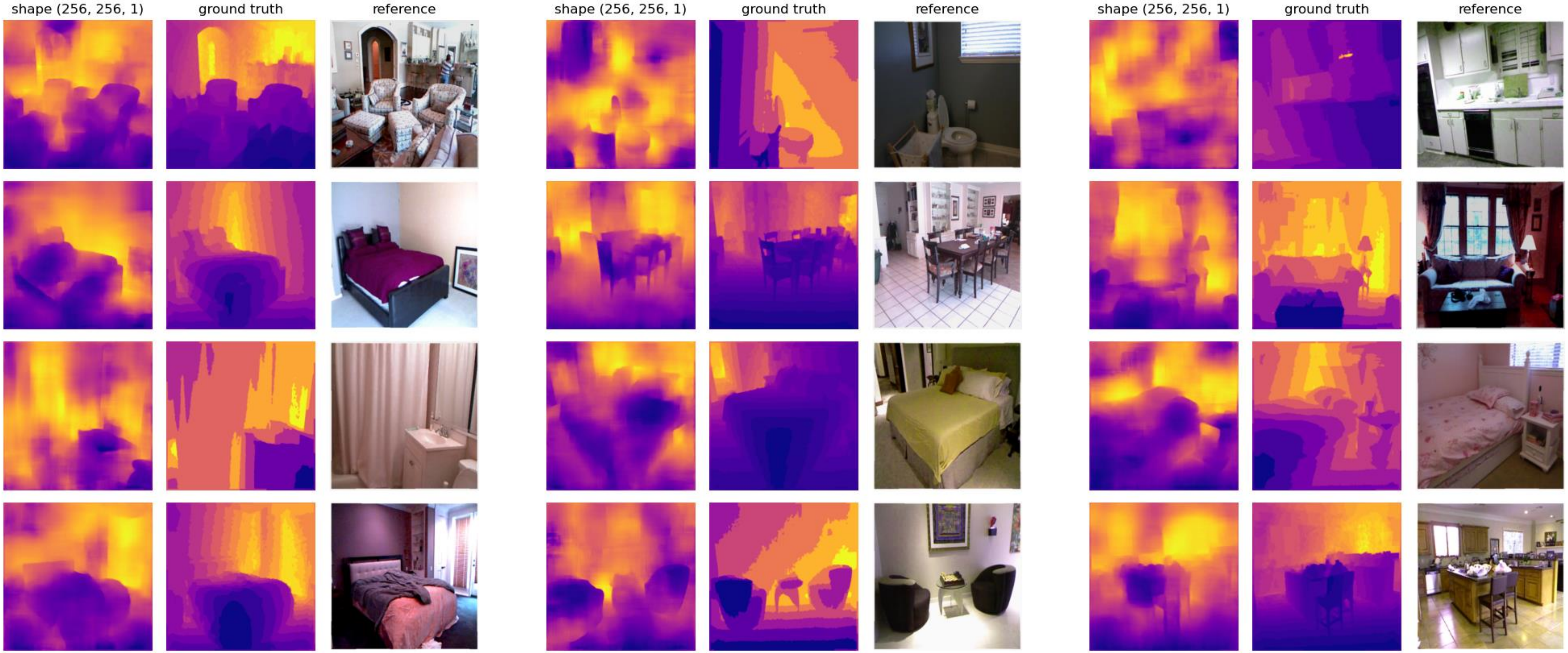


# Batch predictions – Train mode



# Batch predictions – Eval mode




FPS: 0

# The lightweight model

| Model | Checkpoint size | Model size | Quantized model size |
|---|---|---|---|
| MonoDeRT | **21.97 MB** | **7.31 MB** | **1.89 MB** |
| RT MonoDepth | 67.74 MB | 23.12 MB | / |
| Unet (4 layers) | 63.00 MB | 21.50 MB | / |

1. Introduction
2. Related works
3. Proposed methods
4. Dataset and metrics
5. Experimental results
6. **Conclusion**

# Conclusions

MonoDeRT is a lightweight pyramidal encoder decoder with residuals that performs prediction in real-time.

Due to its nature it's easy can be easily deployed in embedded devices.

Due to its simplicity improve the architecture is quite easy.

# Conclusions

MonoDeRT is a lightweight pyramidal encoder decoder with residuals that performs prediction in real-time.

Due to its nature it's easy can be easily deployed in embedded devices.

Due to its simplicity improve the architecture is quite easy.

# Conclusions

MonoDeRT is a lightweight pyramidal encoder decoder with residuals that performs prediction in real-time.

Due to its nature it's easy can be easily deployed in embedded devices.

Due to its simplicity improve the architecture is quite easy.

THANKS FOR

WATCHING