ETH zürich

ML on MCU: Satellite Imagery

Kai Berszin, Alessandro Weber

Dr. Michele Magno, Prof. Dr. Luca Benini

15. January 2024, Zurich

Satellite Imagery

Fig. 1: Sliding window in use during Satellite operation[1]

- Nadir Tracking in LEO (500km)
  - Ground speed ~7km/s
  - Orbit time 90 min
  - Avg. min. 2 overpasses / day
- i.e. CARBONITE-2 (5.2^2 km, 30fps)
- => 240m «new» per picture

Sliding Window
- Proof of concept
- No anchor boxes dataset
- Easy to implement

[1] ESA's Earth Observation Programmes: an introduction, Youtube, 2017. [Online]. Available: https://www.youtube.com/watch?v=N0G_3FWOOPA

Motivation

- Detection of Planes and Vessels using Machine Learning
- Comparison of two microcontrollers
  - MCU compareable to COTS Satellite OBC
  - Optimized Neural Network Accelerator
- Realtime eligibility

- Why should we use microcontrollers?
  - Limited power (mWs)
  - Onboard processing (on the edge)
  - Avoid unnecessary downlink

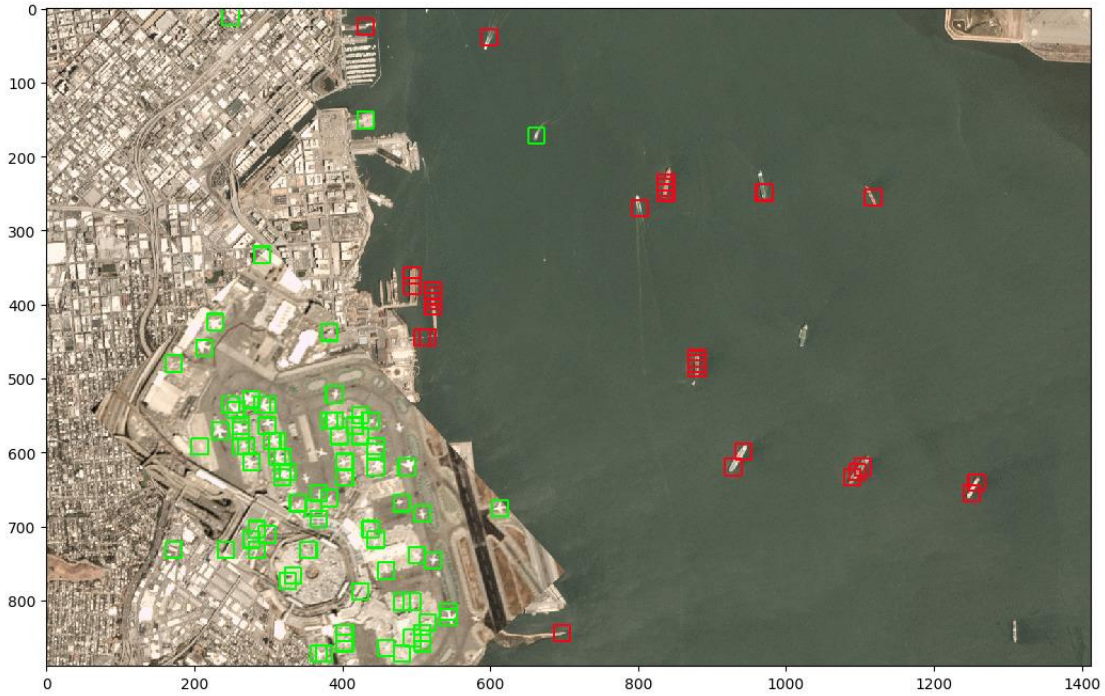


Fig. 2: Augmented pictures of two scenes from datasets [1] and [2]

STM32 / MAX78000

	STM32L475	MAX78000
Flash Memory	1 MB	512 KB
SRAM	128 KB	128 KB
CPU	Arm® Cortex®-M4 80MHz	Arm® Cortex®-M4 100MHz
Neural Network Accelerator	❌	✅ RISC-V Co-Processor @60MHz

Fig. 3: STM32 https://estore.st.com/en/b-l475e-iot01a2-cpn.html

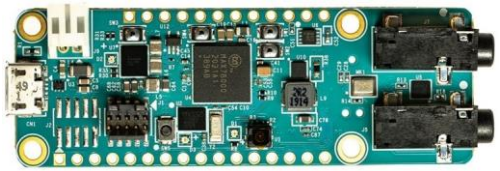


Fig 4: MAX78000
https://www.analog.com/en/products/max78000.html#product-overview

Dataset

- Merge of 2 datasets
- Resized all data to 20x20px

Plane

20x20 px [1]

Ship

80x80 px [2]

None

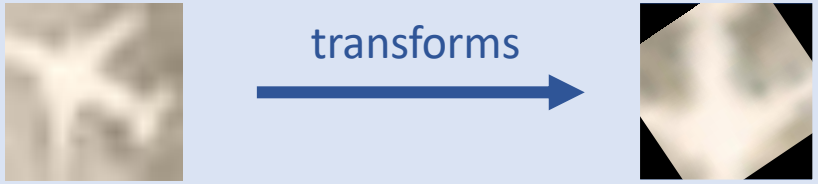
20x20 / 80x80 px [1]

Data Set & Augmentation

- 36’000 Images
  - 8000 Planes[1]
  - 1000 Ships[2]
  - 27000 None[1],[2]
- Class weights
- Generated more augmentated data
- Split into:
  - 80% Training Set
  - 10% Validation Set
  - 10% Test Set

- Horizontal flip
- Vertical flip
- Random rotation
- Random crop

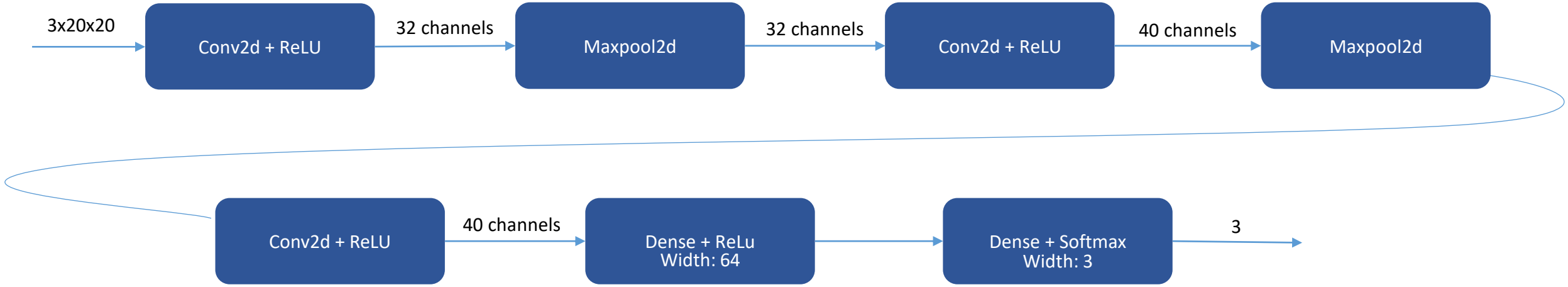
```
train_transform = transforms.Compose([
    transforms.ToPILImage(mode="RGB"),
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.RandomRotation(360),
    transforms.RandomResizedCrop((20,20), scale=(0.8, 1.0)),
    transforms.Resize((20,20)),
    transforms.ToTensor(),
    ai8x.normalize(args=args)
])
```



[1] Robert Hammell: Planes in Satellite Imagery, Kaeggle, 2023, Available: https://www.kaggle.com/dsv/4804225
[2] Robert Hammell: Ships in Satellite Imagery, Kaeggle, 2018, Available: https://www.kaggle.com/dsv/61115

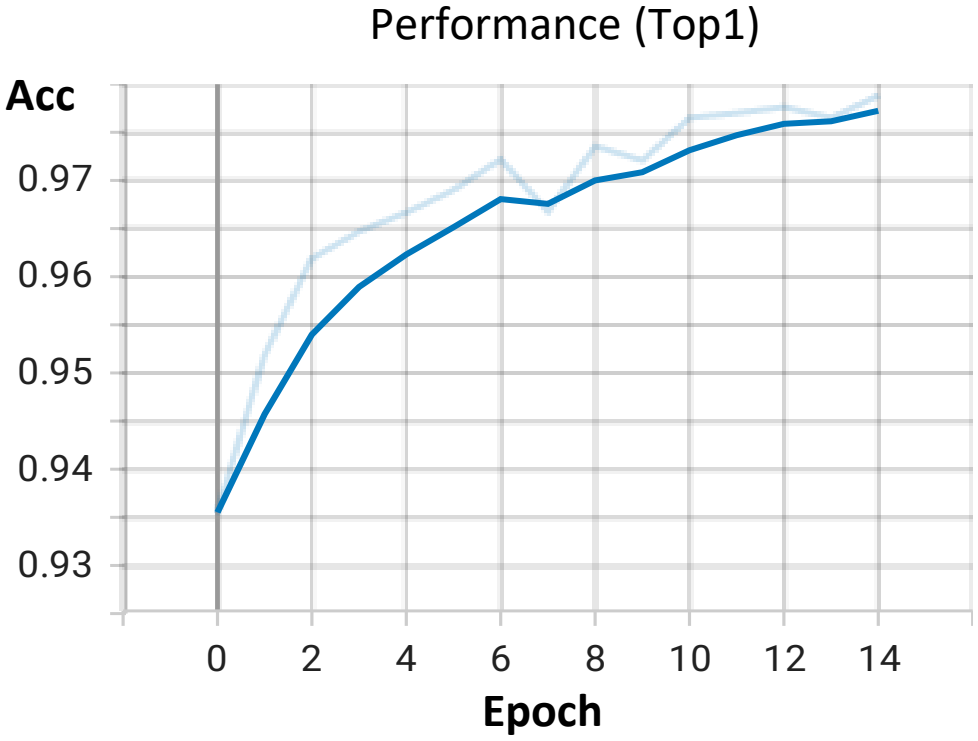
CNN Network – STM32

- 60’395 parameters
- 3 Convolutions (kernel_size: 3x3), 2 Maxpool layers (pool_size: 2)



STM32 Training

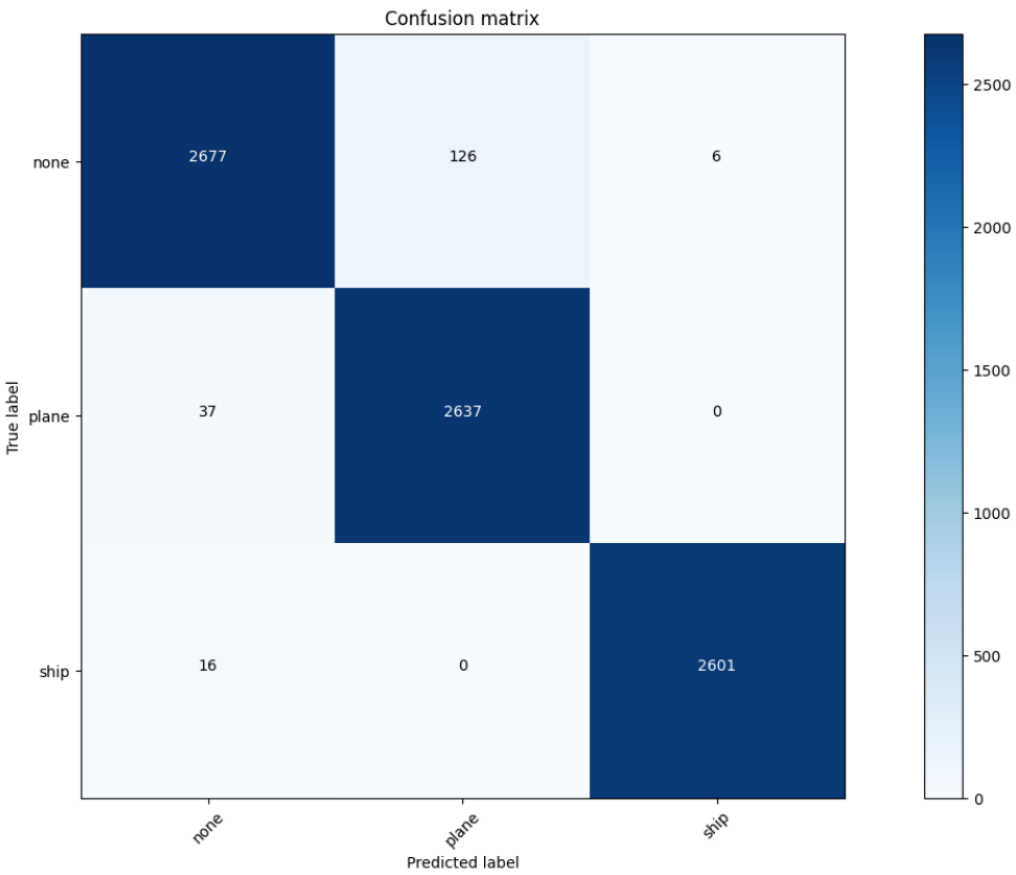
Training:
- 15 Epochs
- 32 Batch Size
- Adam Optimizer



97.7%

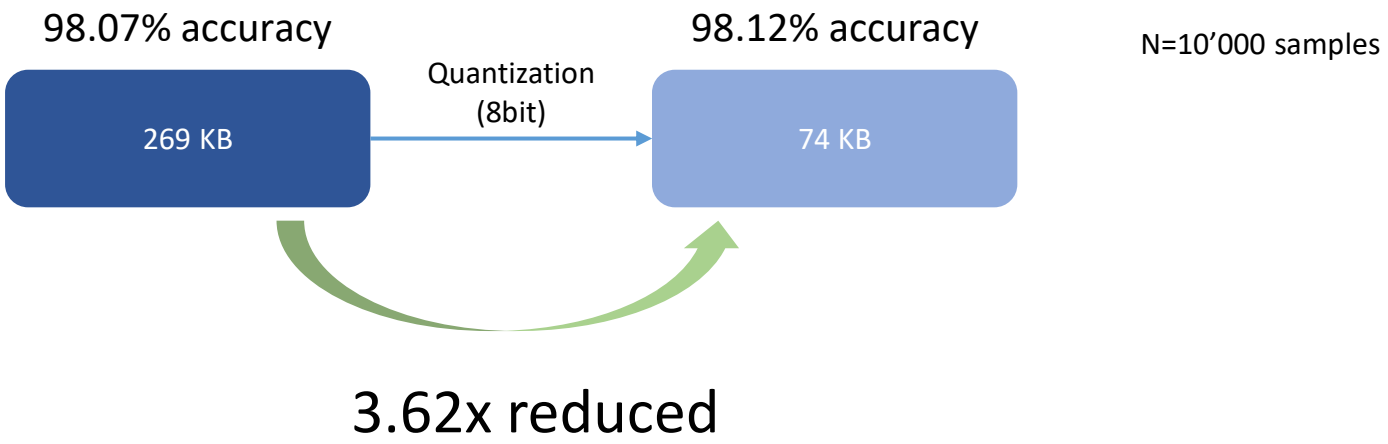
Confusion Matrix on Test Set

- 2700 samples per class:



STM32

- CMSIS-NN Library
- Post-Quantized:



STM32

Operations (macc):

Hardware: 1’500’166 macc
- Layer 0: 290’336
- Layer 1: 935’720
- Layer 2: 230’440
- Layer 3: 41’024
- Layer 4: 195

Estimated Latency:

Startup	1	
Layer 0	1’487’384	18.6ms
Layer 1	2’203’820	27.5ms
Layer 2	492’114	6.2ms
Layer 3	126’662	1.6ms
Layer 4	2’717	0.02ms
Total	4’328’305 cycles	

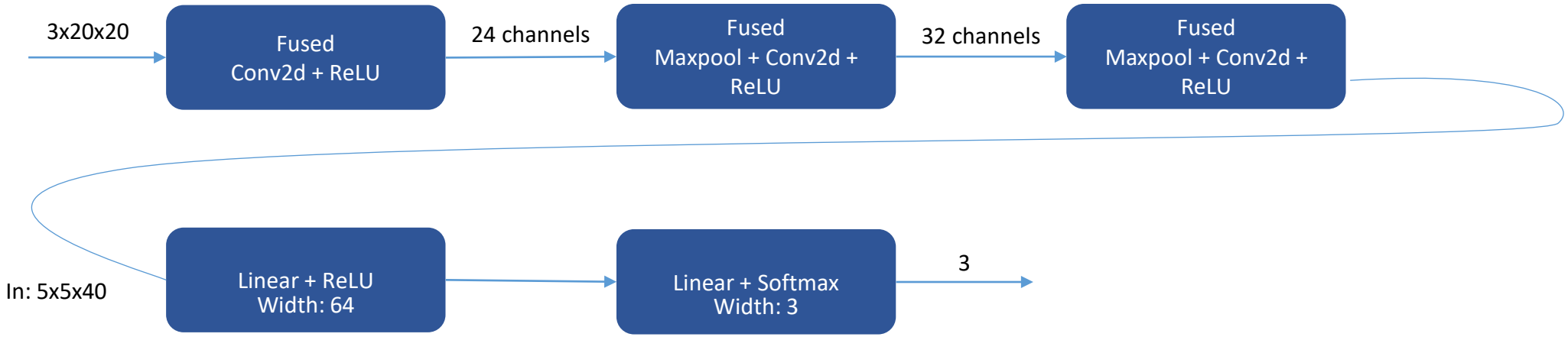
- Cylces/macc: 2.89
- MOps per sec: 27.68
- **Inference Time: 67ms (reality)**

Resource Usage:

Weight memory: 60,968 bytes out of ~1’000’000 bytes total (6.1%)

CNN Network – MAX78000

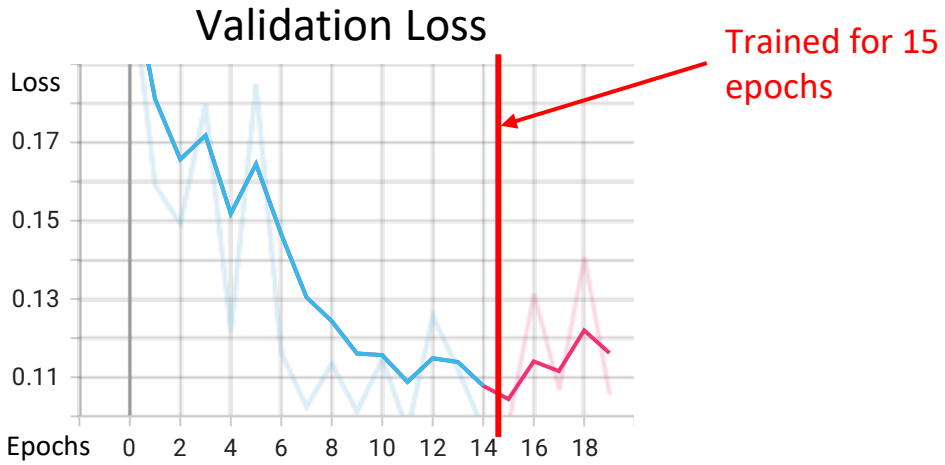
- Use of fused layers
- Same number of convolutions and maxpools as STM32 model

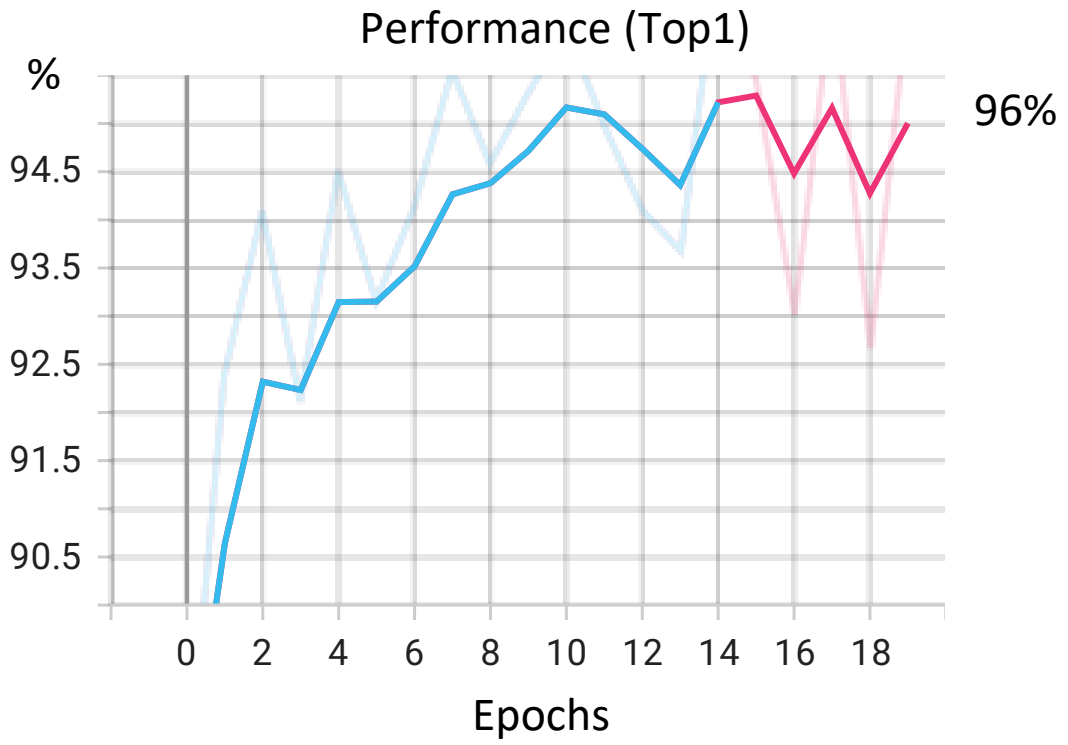


MAX78000 Training

Quantization aware training:

- 15 Epochs
- 32 Batch Size
- Adam Optimizer
- Start @epoch 5





MAX78000

Operations:

Hardware: 1,329,192 ops (1,302,592 macc; 26,600 comp)
- Layer 0: 268,800 ops (259,200 macc; 9,600 comp)
- Layer 1: 704,000 ops (691,200 macc; 12,800 comp)
- Layer 2: 292,200 ops (288,000 macc; 4,200 comp)
- Layer 3: 64,000 ops (64,000 macc; 0 comp)
- Layer 4: 192 ops (192 macc; 0 comp)

Estimated Latency:

Startup	1
Layer 0	10,652
Layer 1	3,832
Layer 2	1,197
Layer 3	1,651
Layer 4	5
Total	17,338 cycles

- Operations per clock cycle: 76
- MOps per sec: 3765.42
- **Inference Time: 353us**

Resource Usage:

Weight memory: 83,272 bytes out of 442,368 bytes total (18.8%)
Bias memory: 67 bytes out of 2,048 bytes total (3.3%)

Comparison

	STM32	MAX78000
Latency	4,328,305 cycles	17’338 cycles
Operations/cycle	0.35	76
Inference Time	67ms	353us

~249x fewer cycles
~190x faster

Power Estimate

	STM32	MAX78000
	11mA * 3V[1] X 67ms	4.02 pJ/MAC [2] X 1.3M MAC
Total Energy	2211 uJ	5.23 uJ
Avg. Power	33mW (67ms)	14.83mW (353us)

~440x less energy
~2x less avg. power

[1] CubeIDE power calculator
[2] CNN Active Energy in MAX78000 Datasheet

Realtime

Inference

- Inference time 67ms / 353us
- Allows for 15 / 2833 Fps

Overhead: UART

- 115’200 baud ~ 14’400 bytes/s
- 1200 bytes / picture

=> 12 pictures/s | 83ms / picture

=> UART adds overhead of 123% / 23’500%

SOLUTION:
DMA
Different communication (SPI, USB, …, I2C)

Conclusion

- Neural Network Accelerator speeds up inference nearly 200 times
  - MAX78000 needs less layers for the same operations (fused layers)
  - 249x fewer cycles
- Uses around 400x less power

Future Work

- Image preprocessing OR
- Yolo network detection instead of sliding window -> more energy efficent
- Different communication protocol
- (Better python script)

Demo

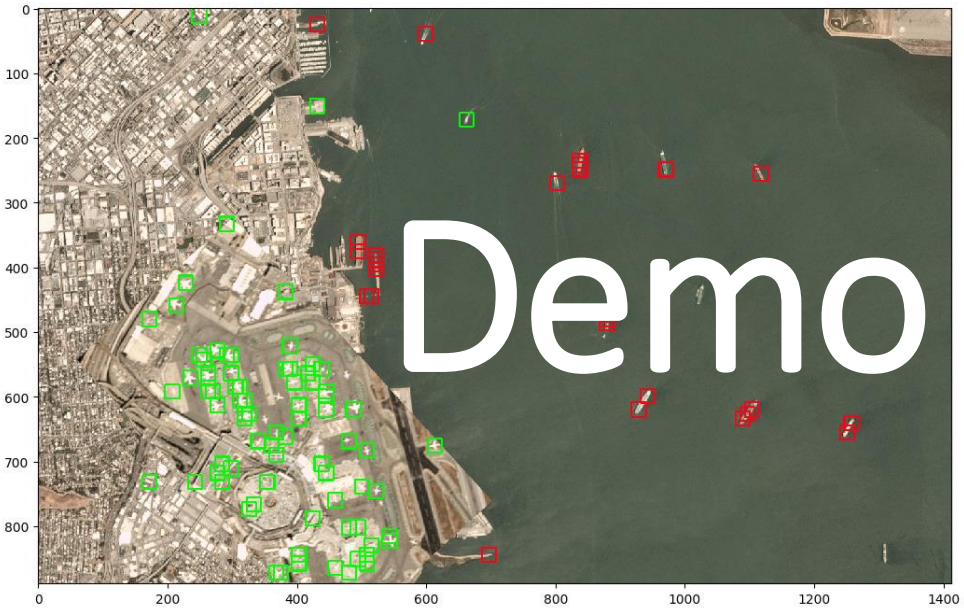


Fig. 5: Augmented pictures of two scenes from datasets [1] and [2]

Additional Slides

Num of Parameters

W_c = Number of weights of the Conv Layer.
B_c = Number of biases of the Conv Layer.
P_c = Number of parameters of the Conv Layer.
K = Size (width) of kernels used in the Conv Layer.
N = Number of kernels.
C = Number of channels of the input image.

$$W_c = K^2 \times C \times N$$
$$B_c = N$$
$$P_c = W_c + B_c$$

In a Conv Layer, the depth of every kernel is always equal to the number of channels in the input image. So every kernel has $K^2 \times C$ parameters, and there are N such kernels. That's how we come up with the above formula.

https://learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/#:~:text=In%20a%20CNN%2C%20each%20layer,of%20all%20weights%20and%20biases.&text=%3D%20Number%20of%20weights%20of%20the,biases%20of%20the%20Conv%20Layer

Network YAML

```yaml
arch: planeshipnet
dataset: planeships

# Define layer parameters in order of the layer sequence
layers:
  - pad: 1
    activate: ReLU
    out_offset: 0x2000
    processors: 0x0000.0000.0000.0007
    data_format: HWC
    op: conv2d

  - pad: 1
    max_pool: 2
    pool_stride: 2
    activate: ReLU
    out_offset: 0
    in_channels: 24
    processors: 0xffff.ff00.0000.0000  #24 processors (6x4) (f = 1111.)
    op: conv2d
    kernel_size: 3x3

  - pad: 1
    max_pool: 2
    pool_stride: 2
    activate: ReLU
    out_offset: 0x2000
    processors: 0xffff.ffff.0000.0000  #32 processors (8x4)
    in_channels: 32
    op: conv2d
    kernel_size: 3x3

  - op: mlp
    out_offset: 0
    flatten: true
    output_width: 8
    processors: 0x0000.00ff.ffff.ffff  #40 processors (10x4)

  - op: mlp
    out_offset: 0x2000
    output_width: 32
    processors: 0xffff.ffff.ffff.ffff #64 processors (16x4)
```

STM32 Network Layers

Inference time per node

c_id	m_id	type	dur (ms)	%	counters	name
0	0	NL (0x107)	0.063	0.1%	[5,045]	ai_node_0
1	2	Conv2dPool (0x109)	18.592	34.4%	[1,487,384]	ai_node_1
2	3	Pad (0x116)	0.071	0.1%	[5,648]	ai_node_2
3	4	Conv2dPool (0x109)	27.548	50.9%	[2,203,820]	ai_node_3
4	5	Pad (0x116)	0.032	0.1%	[2,534]	ai_node_4
5	5	Conv2D (0x103)	6.151	11.4%	[492,114]	ai_node_5
6	7	Dense (0x104)	1.583	2.9%	[126,662]	ai_node_6
7	8	Dense (0x104)	0.034	0.1%	[2,717]	ai_node_7
8	9	NL (0x107)	0.023	0.0%	[1,831]	ai_node_8
9	10	NL (0x107)	0.007	0.0%	[550]	ai_node_9
total			54.104		[4,328,305]	

Difference due to timer overhead