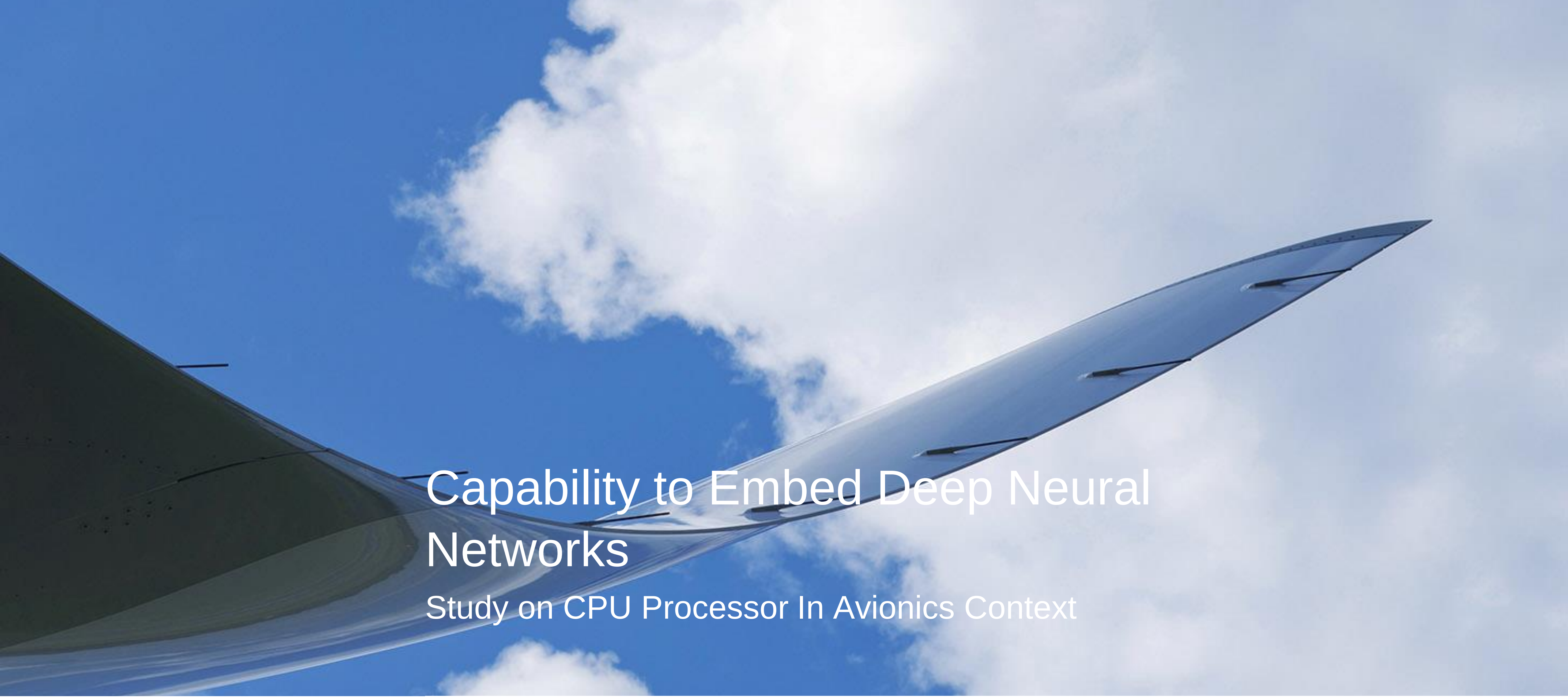


# Capability to Embed Deep Neural Networks

## Study on CPU Processor In Avionics Context

**Sergei CHICHIN**, Marc BRUNDLER, Dominique PORTES, Victor JEGU

30 January 2020

AIRBUS

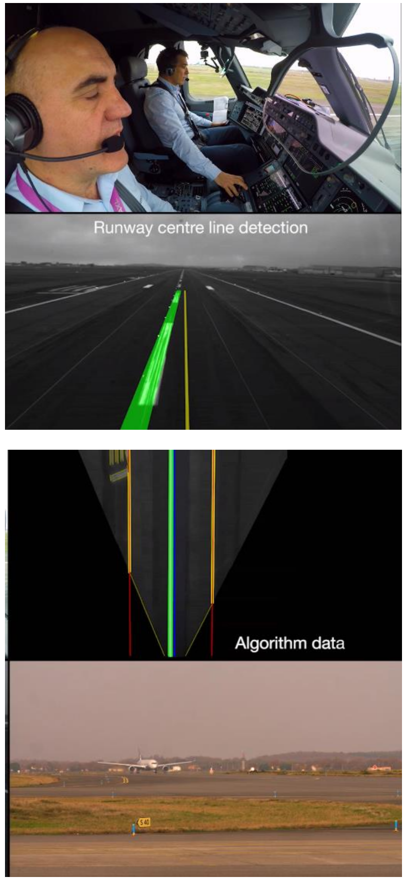
# Agenda

1. Industrial Problem
2. DNN on CPU Study
   a) Work Scope & Workflow
   b) DNN Operations & Implementation
   c) Experimental Results
3. Conclusions

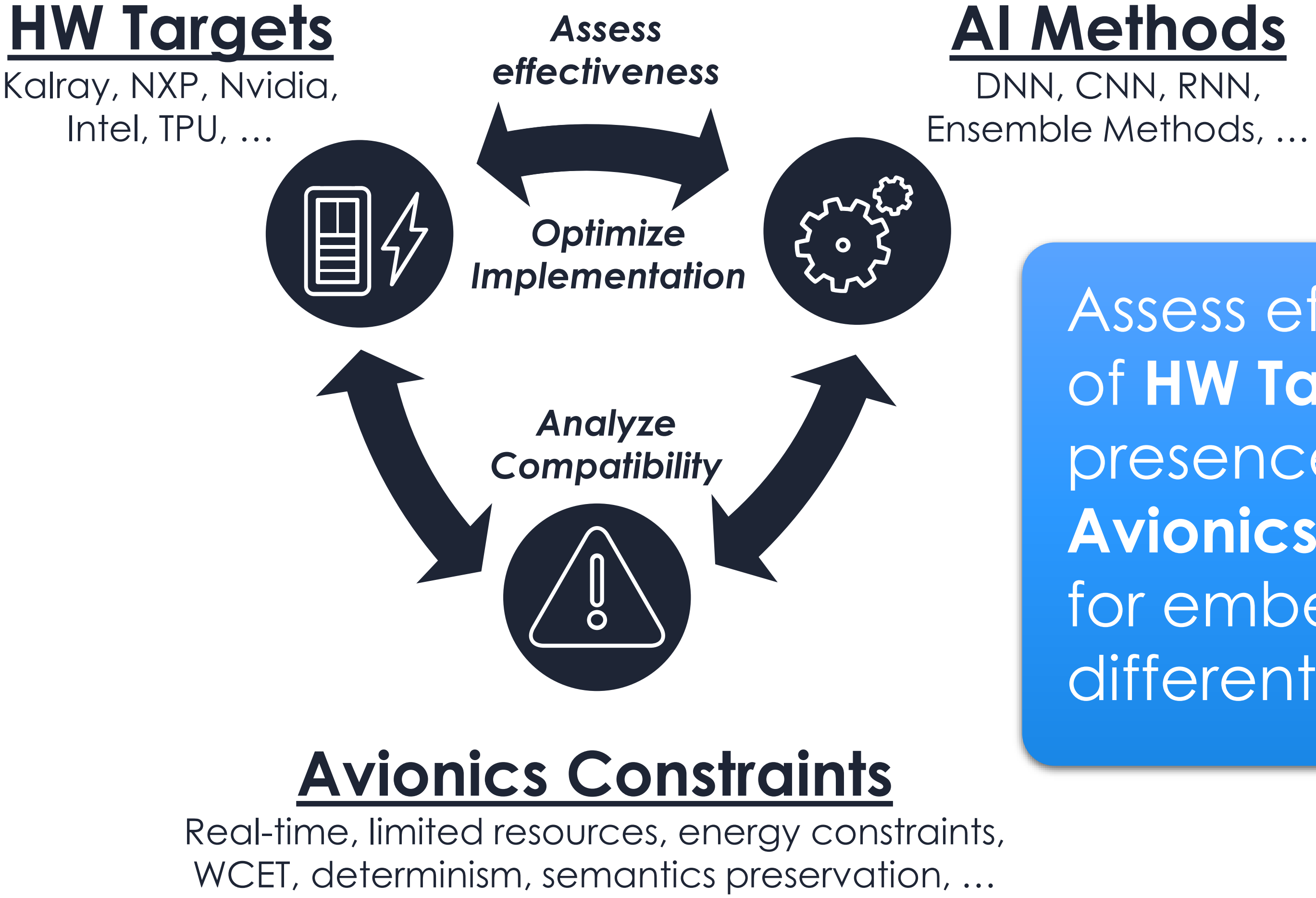
AIRBUS

# Industrial Problem

https://www.airbus.com/newsroom/stories/autonomy-aerial-mobility.html

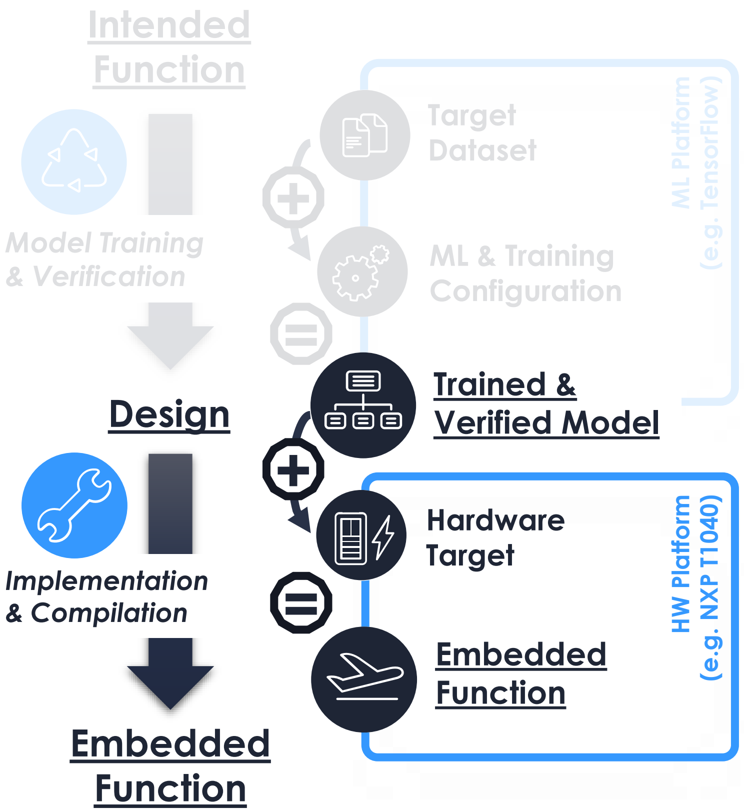



**ATTOL Test Flight**




Assess effectiveness of **HW Targets** in presence of strict **Avionics Constraints** for embedding different **AI Methods**

# Work Scope




## AI Method

Fully-connected Feedforward Neural Network
Fully trained model; focus on inference procedure

## HW Target

Multipurpose CPU processor
Monocore with limited cache

Study operational limits of **DNN** on **CPU monocore** in experimental setting

## Avionics Constraints

Real-time constraint: between 6 and 20 milliseconds
Semantics Preservation: model => code => executable
Deterministic execution: same input => same output
Worst Case Execution Time preliminary analysis

# Trained Model to Embedded Function Workflow

**TensorFlow Training & Export**

frozen .pb

**Model Implementation / Auto Code Generation**

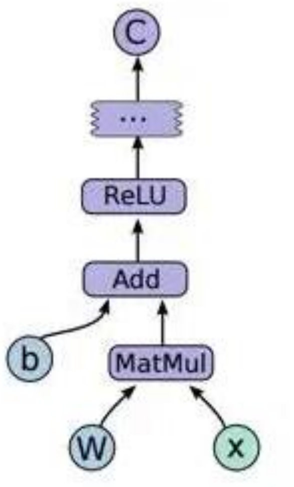
C code

**Code Compilation**

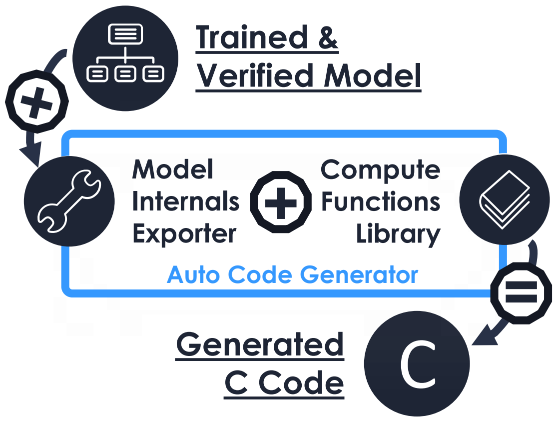
executable

**Embedded function on CPU**

**Bare metal mode**




**TF Computation Graph**



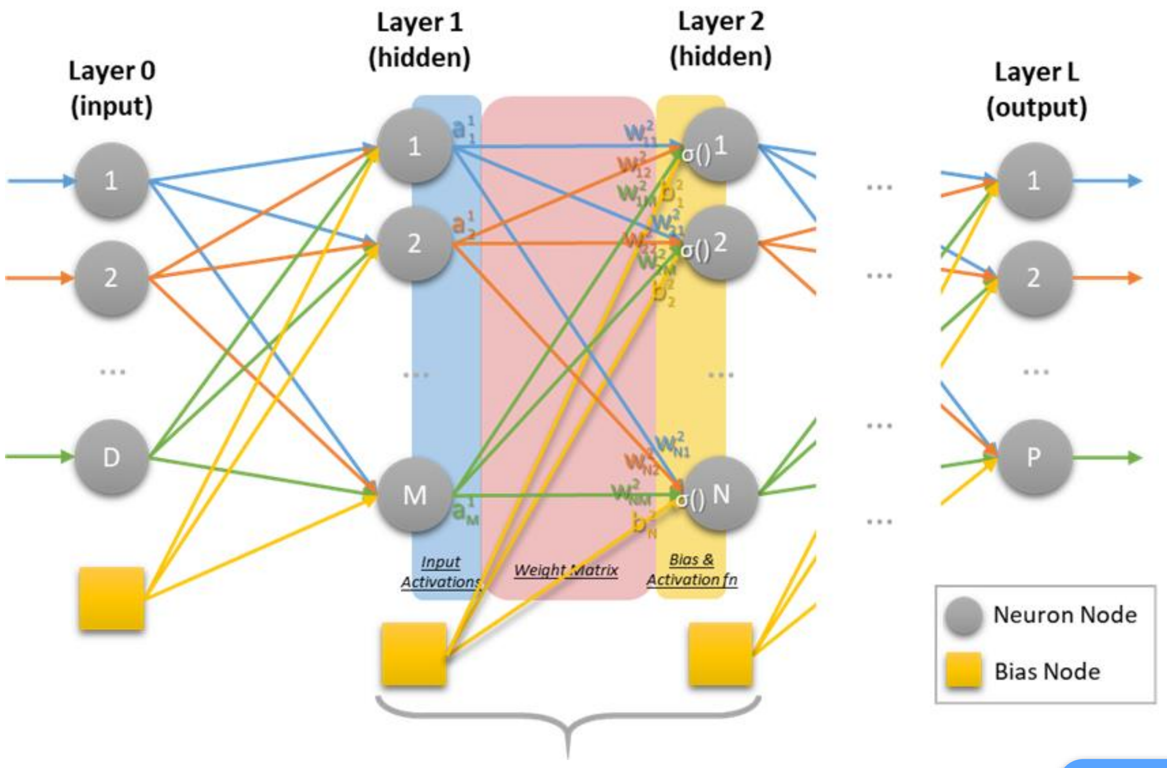

**Auto Code Generator Building Blocks**

**Compilator**

**CompCert** – qualified compiler

**GCC** with **O2** optimization

# Feedforward Deep Neural Network Operations




**Feedforward Neural Network Architecture**

Dense connectivity => memory-intensiveness
Computationally expensive

$$\begin{bmatrix} w_{11}^2 & w_{12}^2 & \dots & w_{1M}^2 \\ w_{21}^2 & w_{22}^2 & \dots & w_{2M}^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1}^2 & w_{N2}^2 & \dots & w_{NM}^2 \end{bmatrix} \begin{bmatrix} a_1^1 \\ a_2^1 \\ \vdots \\ a_M^1 \end{bmatrix} = \begin{bmatrix} w_{11}^2 a_1^1 + w_{12}^2 a_2^1 + \cdots + w_{1M}^2 a_M^1 \\ w_{21}^2 a_1^1 + w_{22}^2 a_2^1 + \cdots + w_{2M}^2 a_M^1 \\ \vdots \\ w_{N1}^2 a_1^1 + w_{N2}^2 a_2^1 + \cdots + w_{NM}^2 a_M^1 \end{bmatrix} = \begin{bmatrix} c_1^2 \\ c_2^2 \\ \vdots \\ c_N^2 \end{bmatrix}$$

**Weights Application**

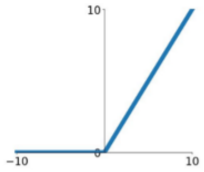
$$\begin{bmatrix} c_1^2 \\ c_2^2 \\ \vdots \\ c_N^2 \end{bmatrix} + \begin{bmatrix} b_1^2 \\ b_2^2 \\ \vdots \\ b_N^2 \end{bmatrix} = \begin{bmatrix} c_1^2 + b_1^2 \\ c_2^2 + b_2^2 \\ \vdots \\ c_N^2 + b_N^2 \end{bmatrix} = \begin{bmatrix} z_1^2 \\ z_2^2 \\ \vdots \\ z_N^2 \end{bmatrix}$$

**Bias Addition**

$$\sigma\left(\begin{bmatrix} z_1^2 \\ z_2^2 \\ \vdots \\ z_N^2 \end{bmatrix}\right) = \begin{bmatrix} \sigma(z_1^2) \\ \sigma(z_2^2) \\ \vdots \\ \sigma(z_N^2) \end{bmatrix} = \begin{bmatrix} a_1^2 \\ a_2^2 \\ \vdots \\ a_N^2 \end{bmatrix}$$

**Activation Application**

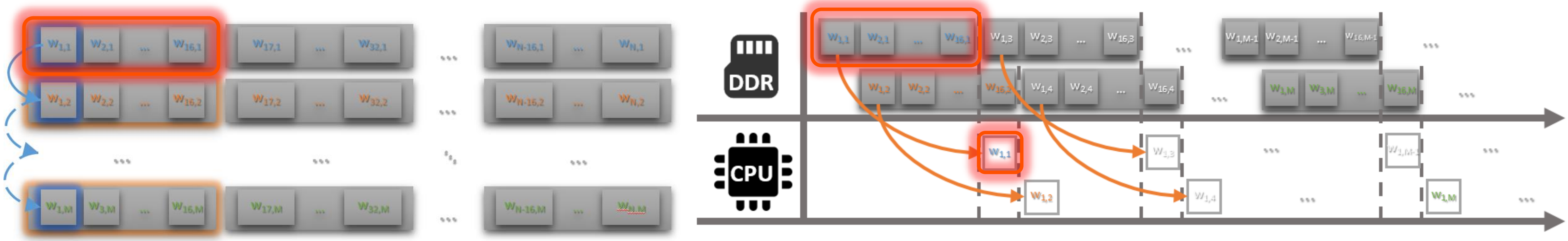
**ReLU**
$\max(0, x)$



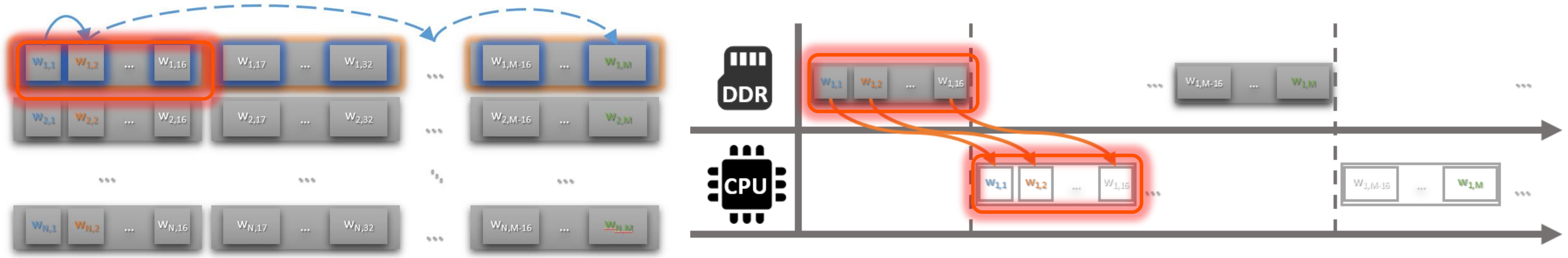

How to access **weights matrix** s.t. to realize **multiply & add** operations in the most efficient manner?

# DNN Implementation Optimization for CPU

## Initial Implementation

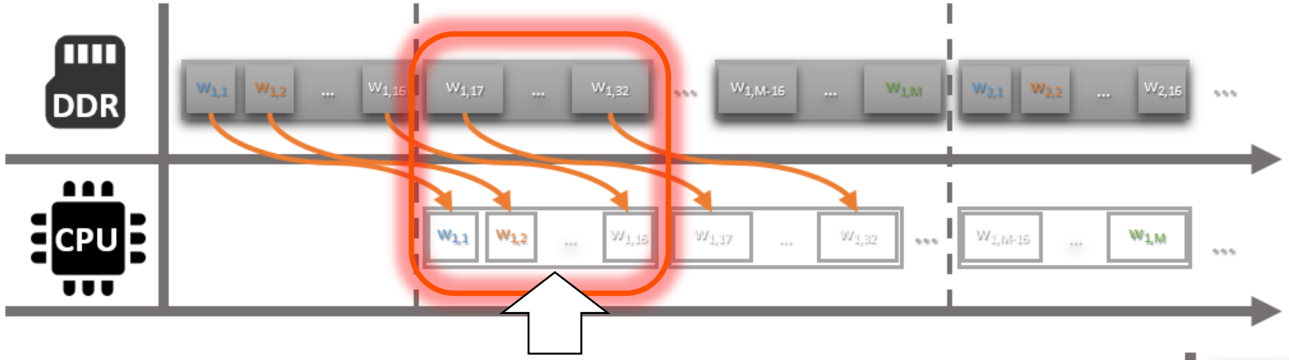



## Improved Memory Access



# DNN Implementation Optimization for CPU

## Improved Latency: Weights pre-fetching




**Multiply & add ops are sequenced**

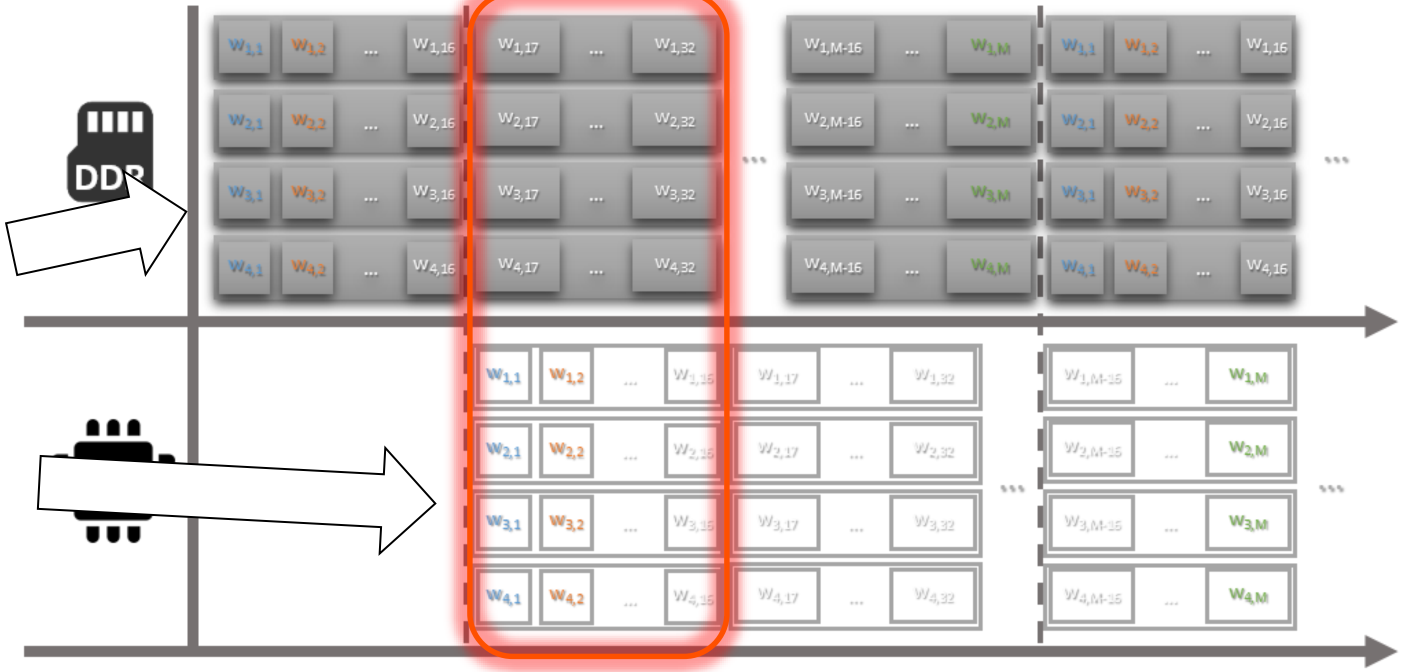
*5 lines maximum can be pre-fetched from DDR at a time*

**Pre-fetch weights for 4 neurons in parallel**

*7 FMADDS ops in parallel to fill FP unit to its max capacity*

**Realize 4 independent calculations**

## Multi-neuron Processing: 2D Pre-fetch

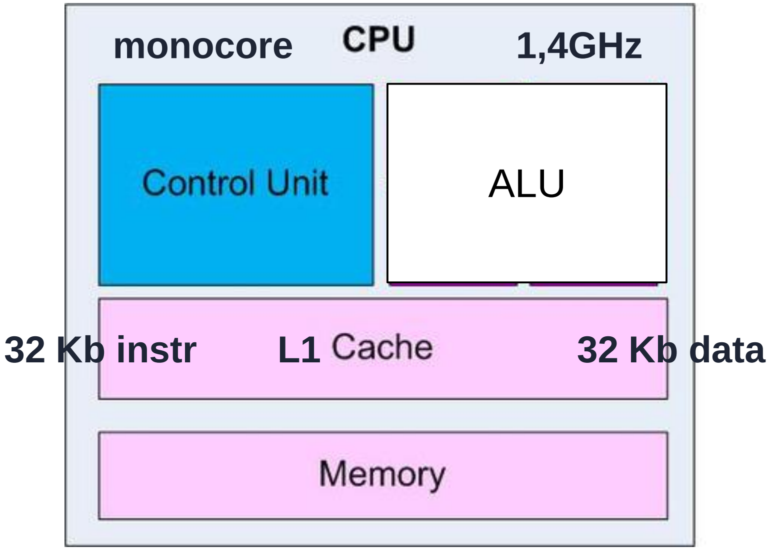





AIRBUS

# Experimental Study

## HW Target




Bare metal

## Performance Metrics

**Nb Clocks** & Execution Time
**Nb Instructions**: total & FP
Instructions per Clock (IPC)
Nb D1 Reloads
**Normalized Metrics**

## Impls & Compilers

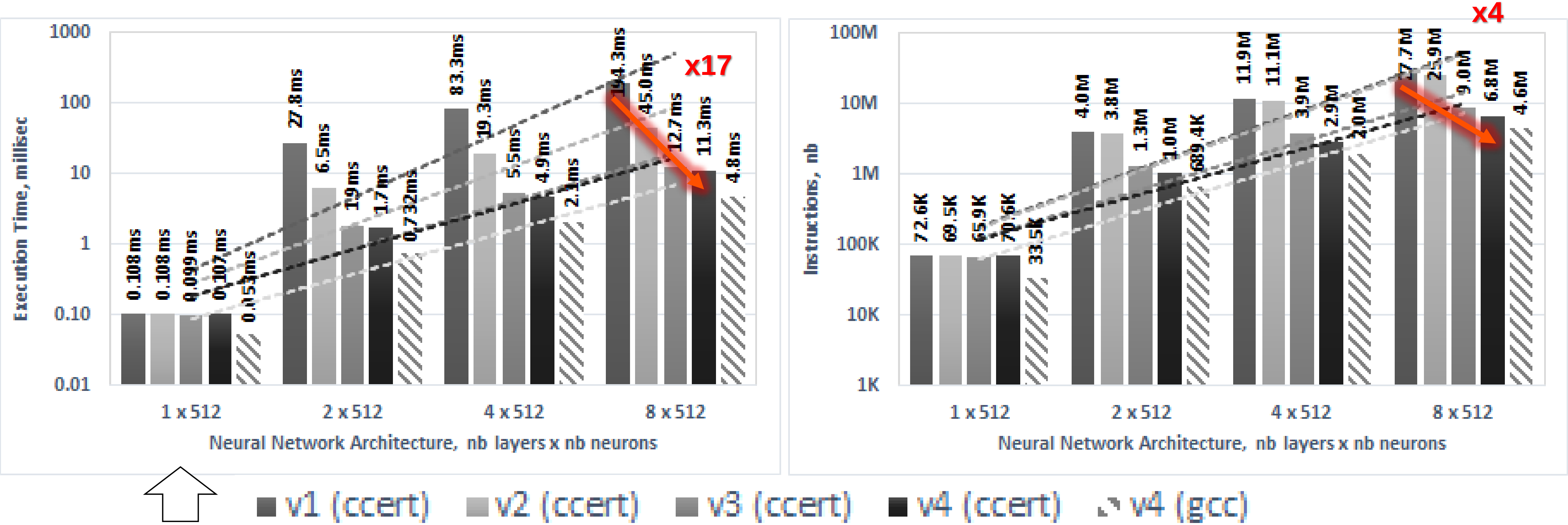
4 versions of DNN code gen
CompCert & GCC O2 (FMADDS)

## DNN Architectures

| | | Nb Hidden Layers | | | |
|---|---|---|---|---|---|
| | | 1 Layer | 2 Layers | 4 Layers | 8 Layers |
| Nb Neurons | 32 | 257 | 1,313 | 3,425 | 7,649 |
| | 64 | 513 | 4,673 | 12,993 | 29,633 |
| | 128 | 1,025 | 17,537 | 50,561 | 116,609 |
| | 256 | 2,049 | 64,841 | 199,425 | 462,593 |
| | 512 | 4,097 | 266,753 | 792,065 | 1,842,689 |

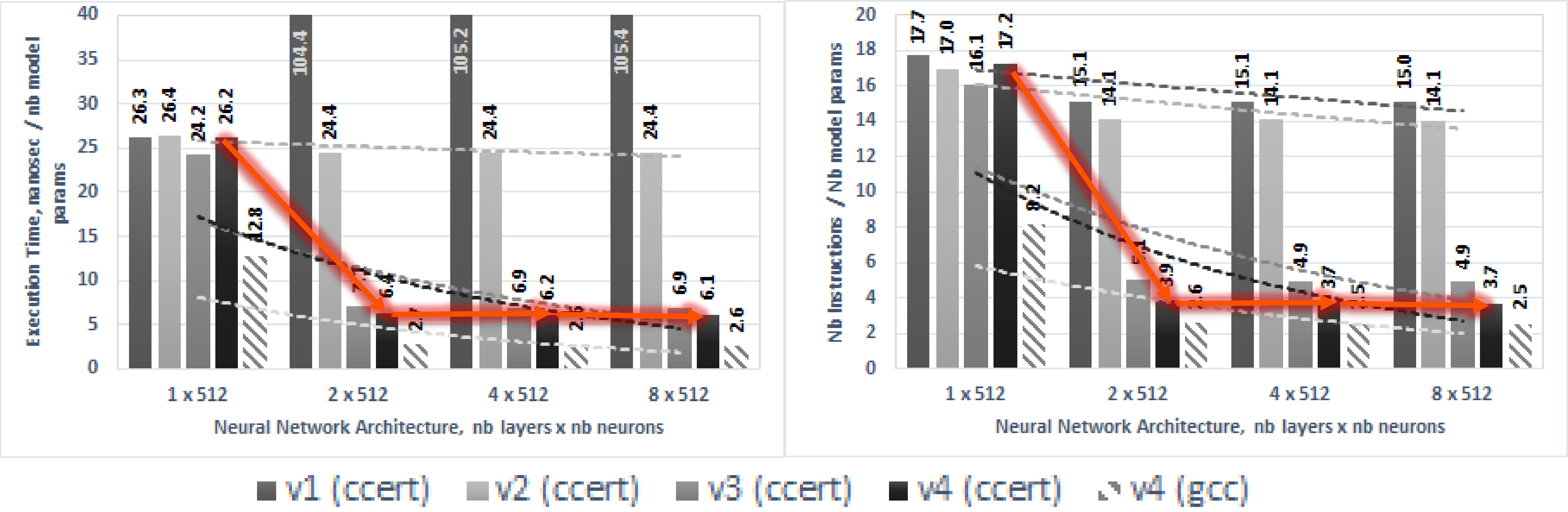
NB DNN Model Parameters

# Experimental Results: Exec Time & Nb Instructions




**1 hidden layer => Insignificant gain due to pre-fetch & parallelization**

# Experimental Results: Scalability





AIRBUS

# Conclusions

1. **Industrial Problem Expressed** => Capability to Embed AI Methods given Avionics Constraints
2. **DNN on CPU monocore study**
   - capable of executing DNN in real time (in general): 18M model params => prediction in 11 milliseconds
   - great scalability of implementation => quasi-linear exec. time in nb model params
   - DNN => same control flow regardless input data => offers temporal stability by construction
   - Nb instructions independent from input vector (branchless implementation)
3. **Future work**
   - Study other HW targets & AI methods
   - Commercial frameworks & certification
   - Numerical precision, quantization & WCET

# Thank you