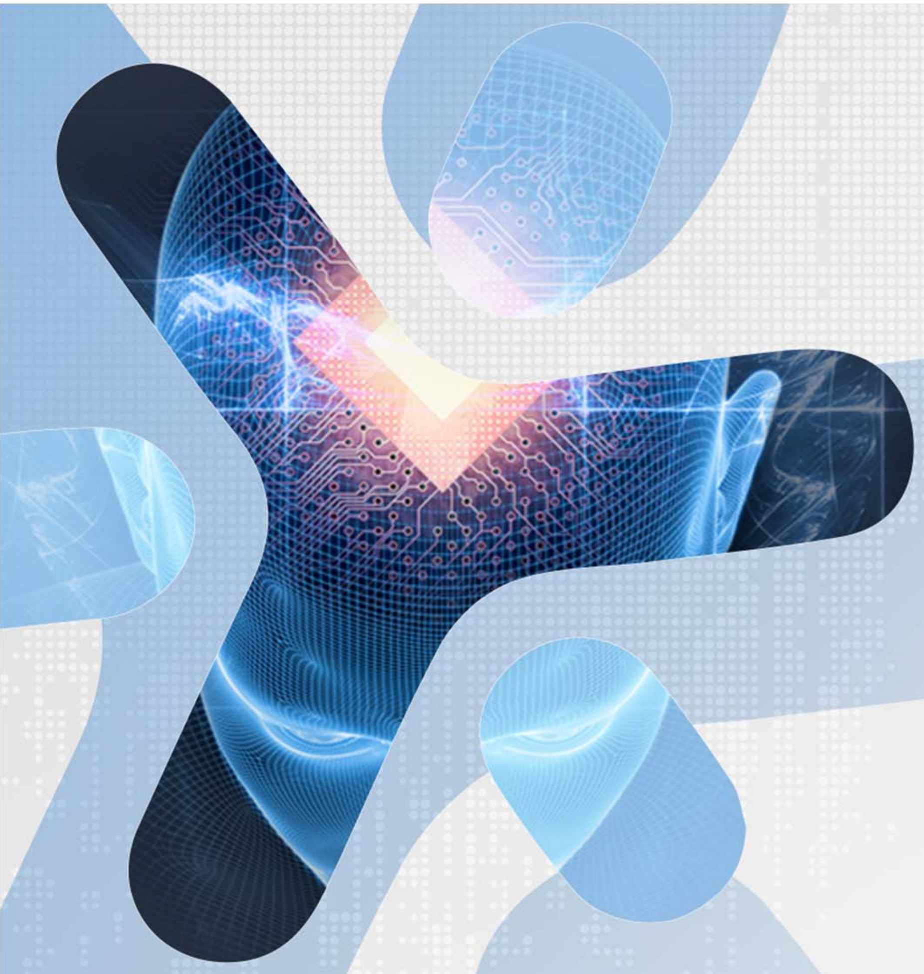


Akida: A Low-Power Neuromorphic SoC for Event-Based Computation

Kristofor D. Carlson
Senior Research Scientist

brainchip

This presentation discusses ongoing R&D work and the information in this presentation is subject to change.

Machine Learning at the Edge

- Edge computing: compute where data is generated
- Applications list:
  - Smart home
  - Industrial manufacturing
  - Environment
  - Medical/health care
  - Transportation
  - Energy management
- Benefits:
  - Latency reduction
  - Power reduction
  - Comm. reduction
  - Increased security
  - Increased privacy

Challenges: DNNs at the Edge

- DNNs are a popular solution for:
  - Speech recognition
  - Image classification
  - Anomaly detection
  - Facial detection/recognition

- DNNs are great at creating multi-level representations using automated feature extraction

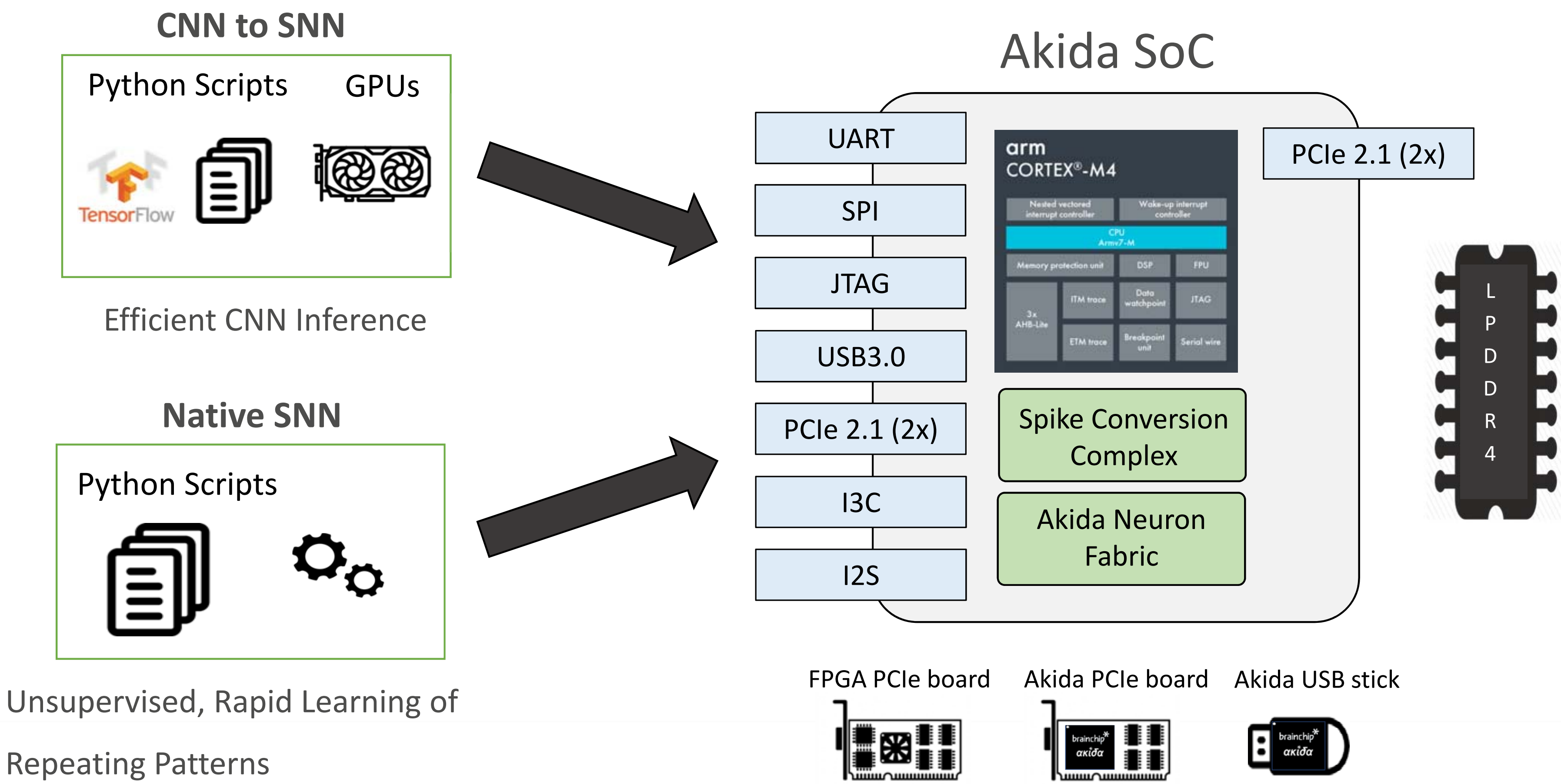
- **Challenges:**
  - Training requirements
  - Memory requirements
  - Not built with event-based computing in mind
  - Power requirements
  - Computational requirements
  - Adaptability

Design Goals

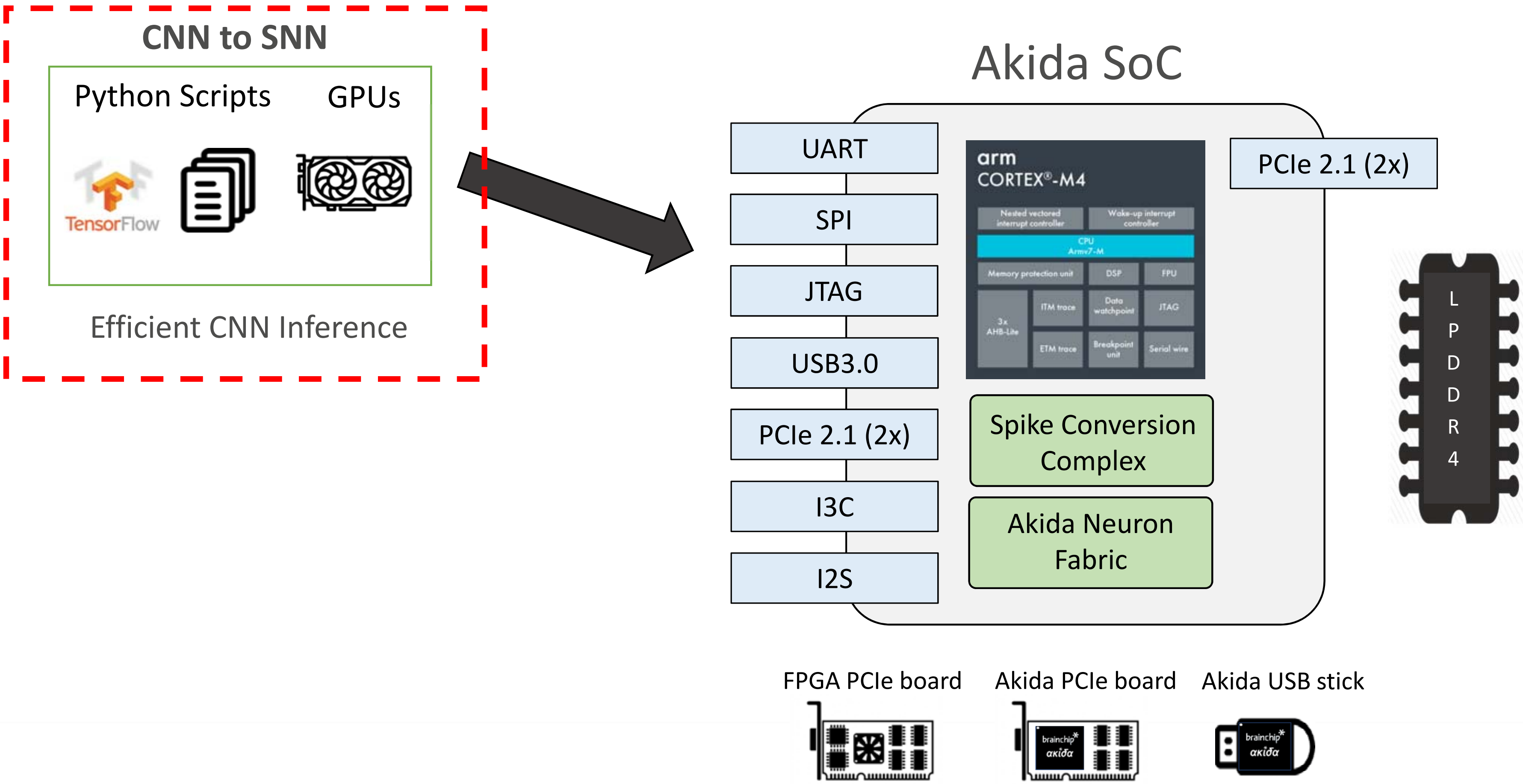
- Use a single **low-power hardware platform** to run:
  - Conventional DNN inference algorithms
  - Native SNNs and event-based algorithms
  - **On-chip** unsupervised learning algorithms
  - **Entire network** runs on Akida, host only passes data and receives results

- Additional goals:
  - Use well-known ML ecosystems for development
  - Implement a small set of useful computational operations

Akida System on a Chip



Akida System on a Chip



Akida SoC

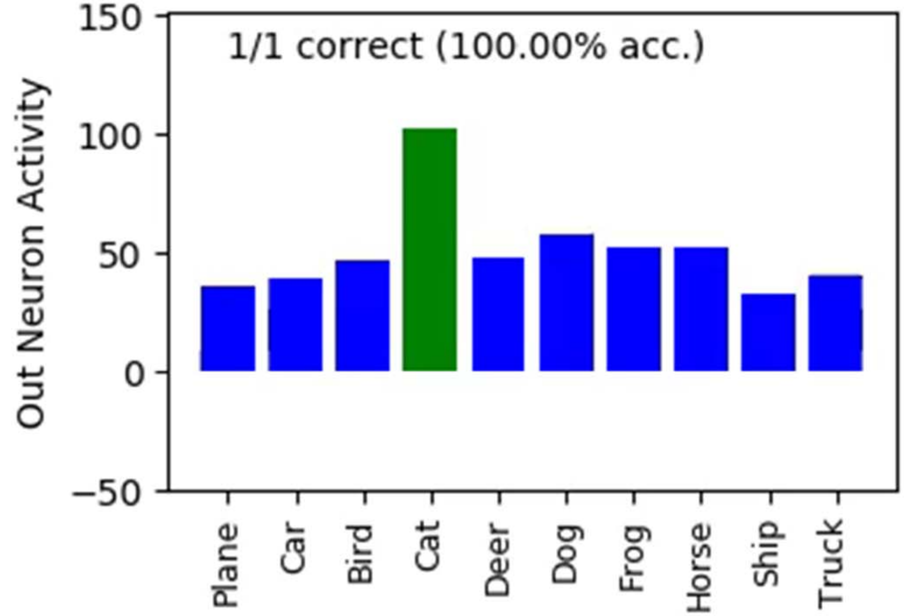
Low-Power CNN Inference

- **Applications**: Object classification, object detection, and speech recognition
- **Training**: Supervised with labelled dataset with activation and weight quantization
- **DNN algorithm focus**: Feed-forward CNNs
- **Purpose**: CNN inference on small, low-power devices
- **Solution**: Use event-based convolutions/separable convolutions

Supported Layer Types
Convolutional Input Layer
Generic Event-Encoding Layer
Standard Convolutional Layer
Separable Convolutional Layer
Fully-Connected Layer
Max Pooling Layer
Global Average Pooling Layer

Akida Execution Engine Running an Event-based CNN

CIFAR10 Classification by Akida



CIFAR10 Data Set Results

- Training Images:
  - 60,000 Color Images
  - 32 X 32 Resolution
  - 10 Classes
- Testing Images:
  - 10,000 Images
- Network:
  - VGG-like
  - Ternary weights
  - Binary activations
- Results:
  - Accuracy: **Top-1**: ~92%

Layer #	Type/Stride	Filter Shape	Input Size
1	Conv/s1	3×3×3×128	32×32×3
2	Conv/s1 MP	3×3×128×128	32×32×128
3	Conv/s1	3×3×128×256	16×16×128
4	Conv/s1 MP	3×3×256×256	16×16×256
5	Conv/s1	3×3×256×512	8×8×256
6	Conv/s1 MP	3×3×512×512	8×8×512
7	Fully Conn	8192×1024	4×4×512
8	Fully Conn	1024×1024	1024
9	Fully Conn	1024×10	10

Total Parameters: 14,018,560

Equivalent Synapses: 655,239,168 ← Benefit of using convolution

Dataset source: Alex Krizhevsky, computer Science dept. University of Toronto

ImageNet Data Set Results

- Training Images:
  - ~1.2M Color Images
  - 224 X 224 Resolution
  - 1,000 Classes
- Testing Images:
  - ~50,000 Images
- Network:
  - Adapted MobileNet V1*
  - Akida-Compatible:
    - 4-bit weights/activations
    - Single 2-bit weights layer
- Results:
  - Our Accuracy: **Top-1**: ~65.6%

*Howard, Andrew G., et al. *arXiv preprint arXiv:1704.04861 (2017).*

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Total Parameters: 4,208,224

Equivalent Synapses: 473,256,424

Benefit of convolution

Akida SoC

Low-Power CNN Inference: Convolutional Ops

Supported	Kernel Size	Stride	Type
Image convolution (RGB888 or grayscale)	Conv: 3x3, 5x5, 7x7 Max Pool: 2x2	1,2,3 2	Same, valid
Standard convolution	1x1, 3x3, 5x5, 7x7	1	Same
Point wise convolution	1×1	1	Same
Depth-wise convolution	3×3, 5x5, 7x7	1	Same
Max pooling	2×2, 3x3	1, 2, 3 (only for 3x3)	N/A
Global average pooling	$W_{Input} \times H_{Input}$	N/A	N/A

Weight (signed)	Activation	No MACs	MACs
1 – 2-bit	1 – 2-bit	✓	✗
>2-bit	1 – 2-bit	✓	✗
1 – 2-bit	>2-bit	✓	✗
4-bit	4-bit	✗	✓
		Lower power	***Higher Power***

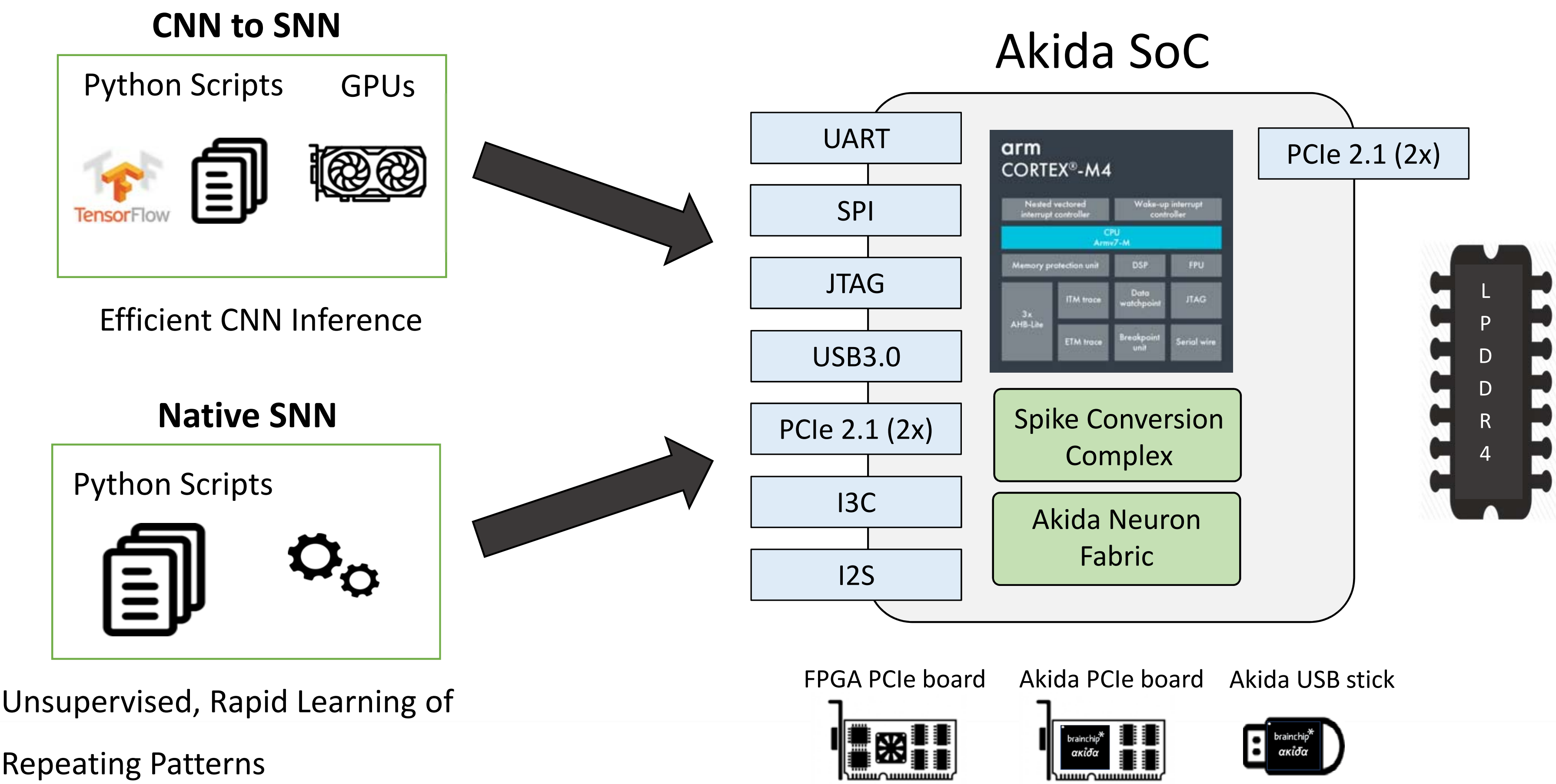
How Can We Compute Convolutions Efficiently?

- Reduce the number of required operations
  - Event-based convolutional and fully-connected layers
  - Separable convolutions yield fewer computations*

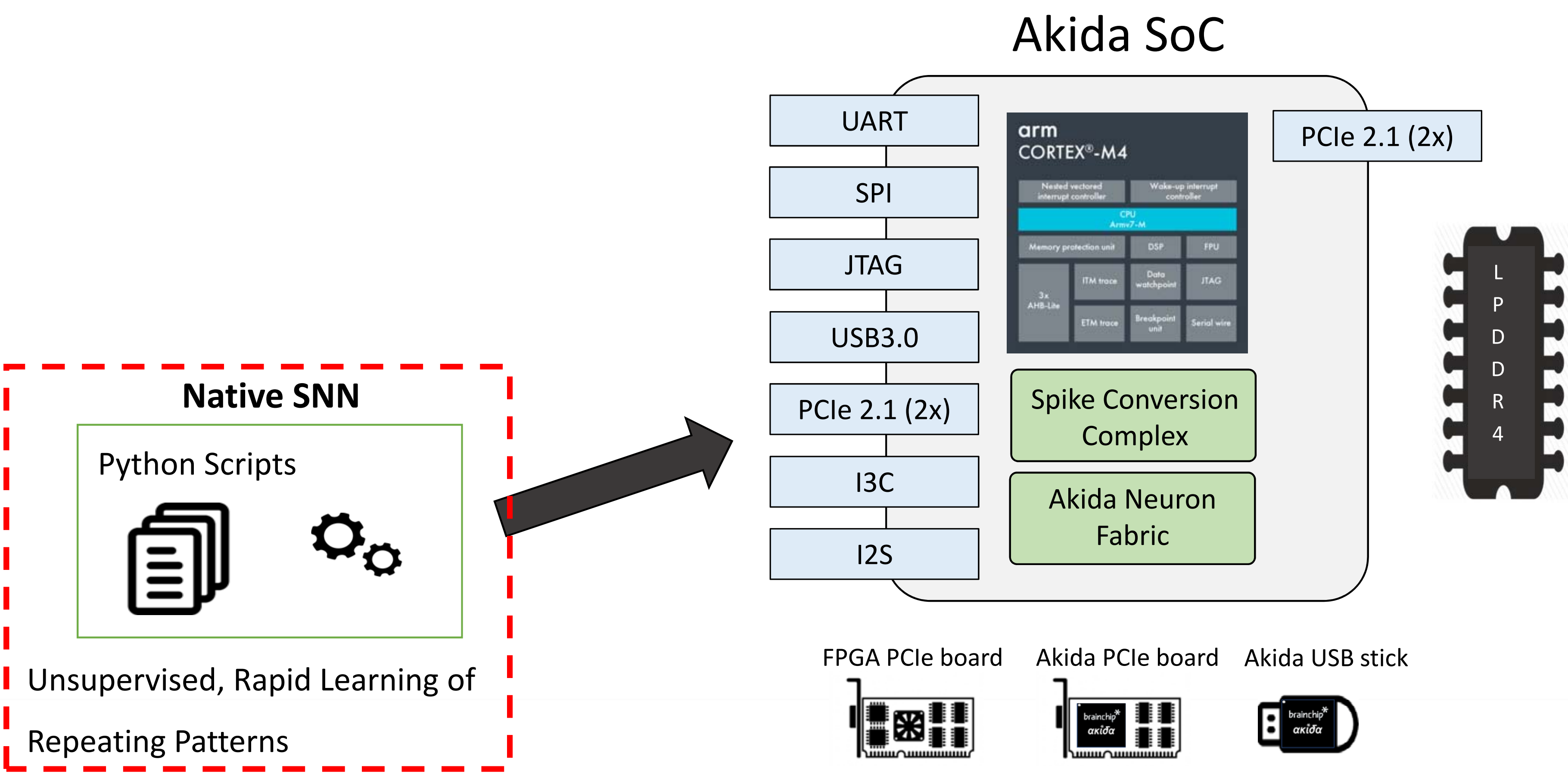
- Reduce cost of each operation
  - Especially important for SNNs (cost of each event matters!)

- Reduce memory requirements
  - Separable convolutions give you fewer parameters
  - Low-precision weights (1-4 bit)

Akida System on a Chip



Akida System on a Chip



Akida SoC

Low-Power Unsupervised Learning for Pattern Recognition

- Proprietary learning rule:
  - Is an STDP-inspired learning rule

- Includes:
  - Homeostatic mechanisms
  - Competition mechanisms
  - Extremely simple and efficient implementation

- Excels at finding and learning input patterns embedded in noise with very few repetitions (on average 3–5)

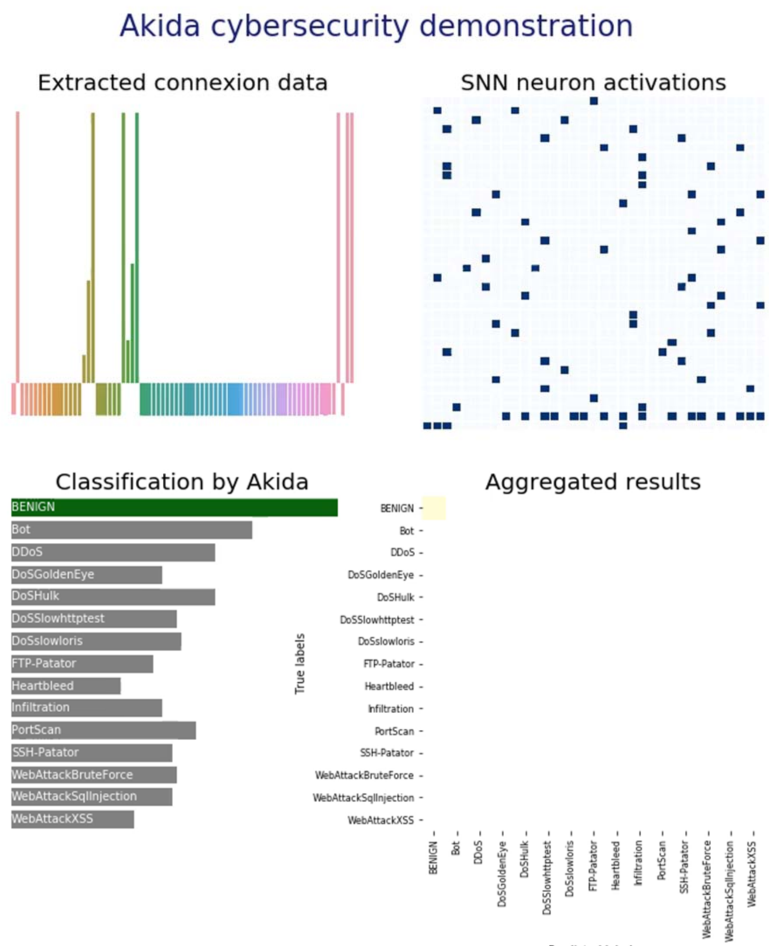
Akida SoC

On-Chip Learning Application Using DVS Camera

- Akida trained on hand-gestures using and event-based camera
- Pre-trained on 3 hand-gestures
- Learns class 5 in real-time
- Uses our proprietary unsupervised learning rule

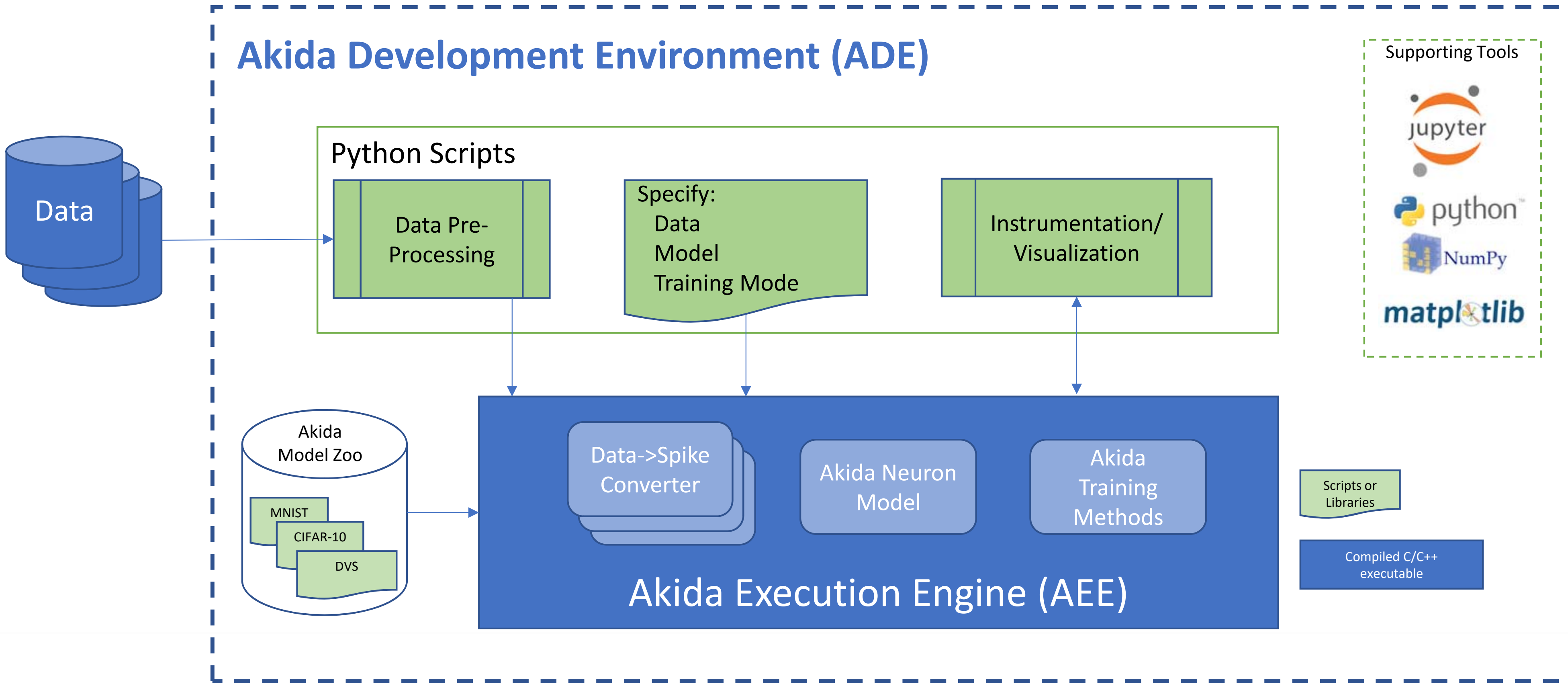
Akida SoC

On-Chip Learning Cybersecurity Application

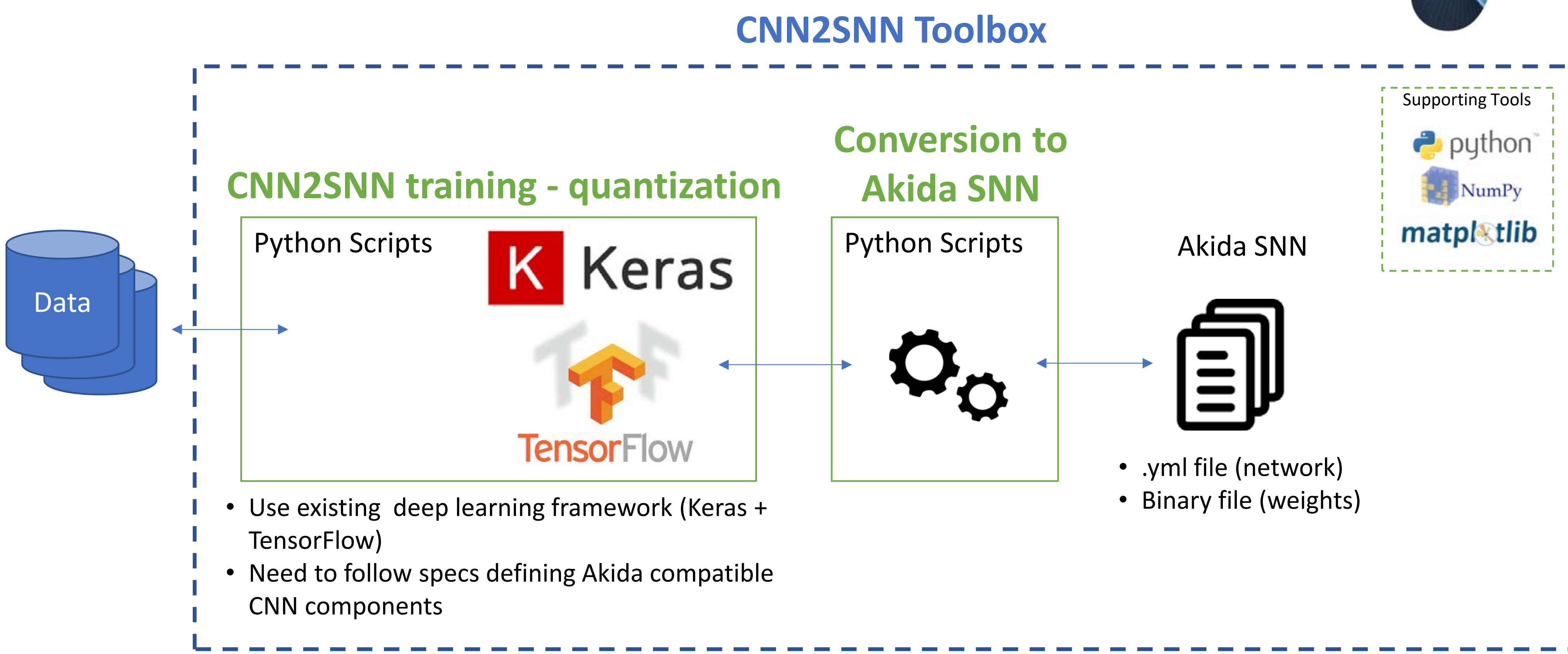


* Akida incorporates data-to-spike converters to process tabularized data from the PCAP file
* Trained on the CIC-IDS-2017 dataset, 2.5M lines corresponding to captured internet traffic with 14 categories of attack, using native on-chip learning
* 97.4% accuracy F1 score = 0.97 using 10% of never-seen samples as test set
* Training time: ~20 minutes/epoch (used 1 epoch)

Akida Development Environment



Akida Development Environment

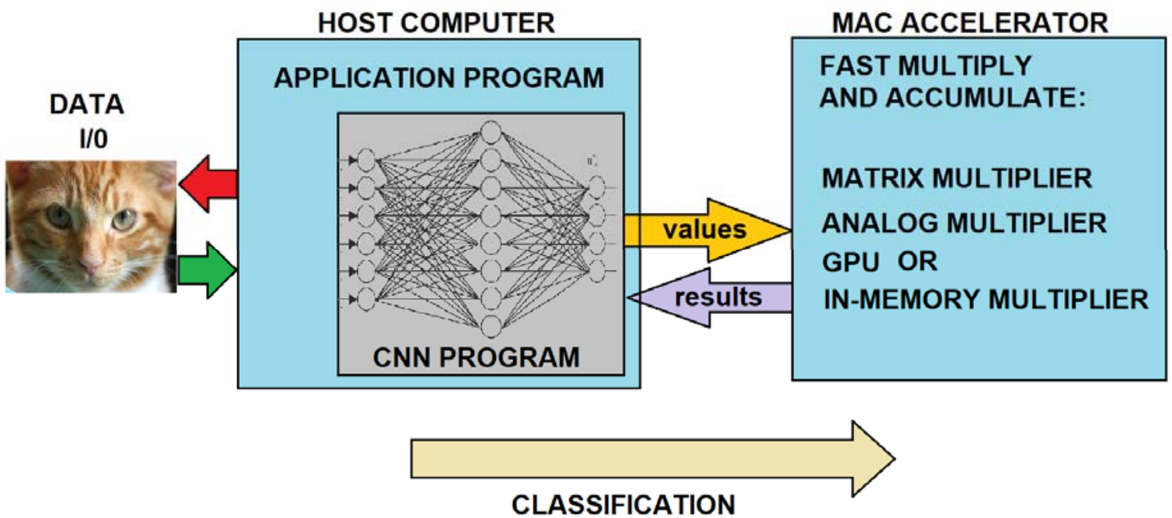


Thank you

For more details contact:

Peter AJ van der Made, CTO: pmade@brainchip.com

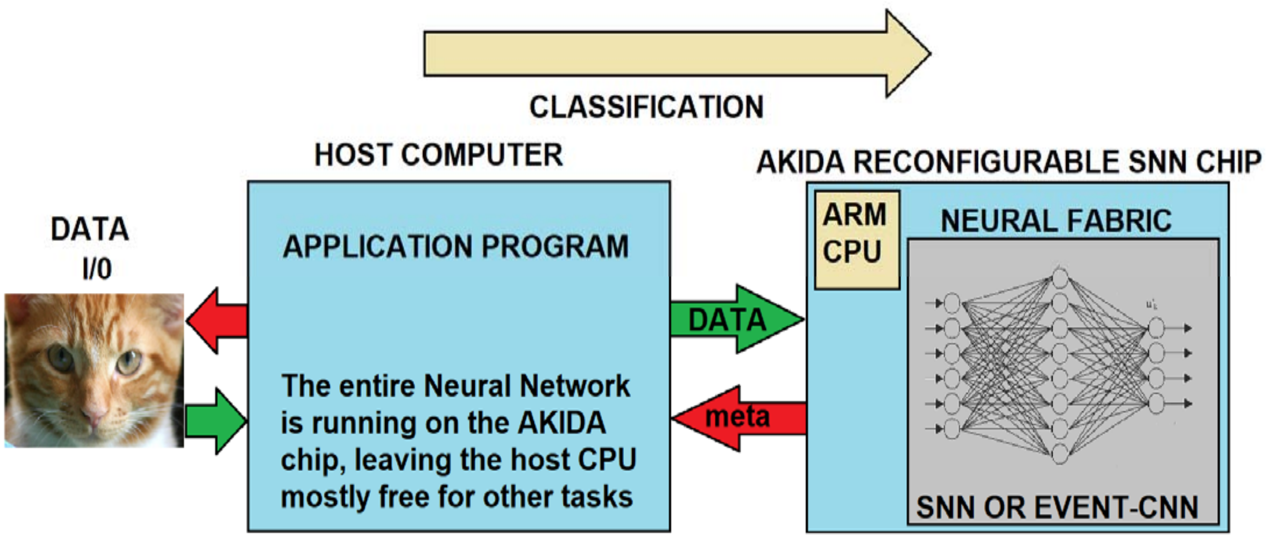
MAC Accelerator architecture



Advantages and Disadvantages

- The classification speed is highly dependent on the host computer
- Benchmarks of frames per second have little relevance to the MAC accelerator speed itself
- Power figures are also highly dependent on the host computer, which runs all of the neural network functions
- No autonomous learning functions, limited to CNN only
- This architecture is highly suited to run any kind of dedicated Artificial Neural Network trained with Deep Learning

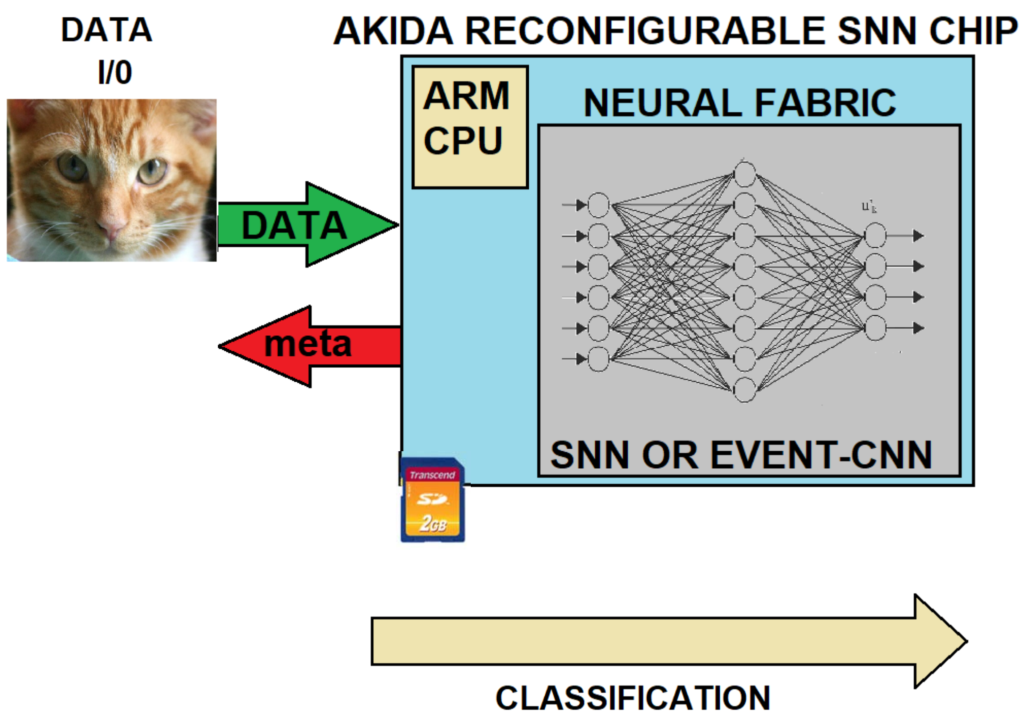
Akida with host architecture



Advantages and Disadvantages

- The entire network runs on the chip, the host only needs to pass preprocessed data to the network, and processed metadata is returned (classification etc)
- This frees up the host memory and cycles for other tasks
- Power and speed benchmarks are clearly defined for the Akida chip
- On-chip learning, optional convolutional, pooling and fully connected layers
- Limited to running native learning SNN, autonomous learning CNN, and feed-forward CNN trained with DL.

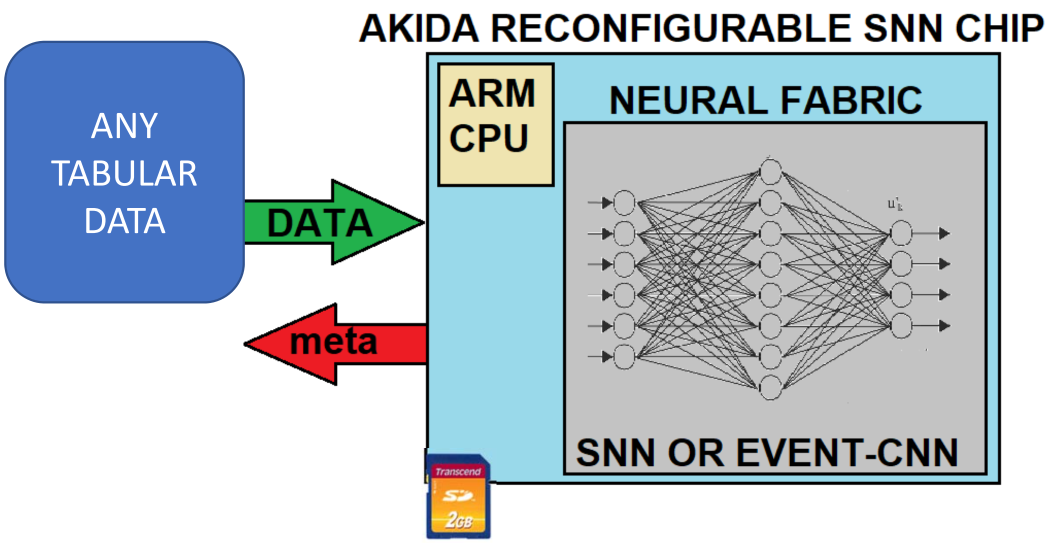
Akida without host architecture, ECNN



Advantages and Disadvantages

- The entire event-based CNN network runs on the chip, the network configuration is contained in non-volatile memory, no host is needed
- The on-chip CPU can be used for pre-processing of data
- Power and speed benchmarks are clearly defined for the Akida chip
- On-chip learning, optional convolutional, pooling and fully connected layers
- Limited to running native learning SNN, autonomous learning CNN, and feed-forward CNN trained with DL.

Akida without host architecture, SNN



Advantages and Disadvantages

- The entire network runs on the chip, no host is needed. The chip is initialized by the on-chip ARM processor, Meta-data output
- The on-chip ARM processor can be used for pre-processing of data
- Power and speed benchmarks are clearly defined for the Akida chip
- Network configuration, layers, connections are stored in non-volatile memory
- On-chip autonomous learning