Axiado CNN Accelerator

Sindhuja Yadiki, Salome Devkule, Raviteja Godugu, Arasch U Lagies

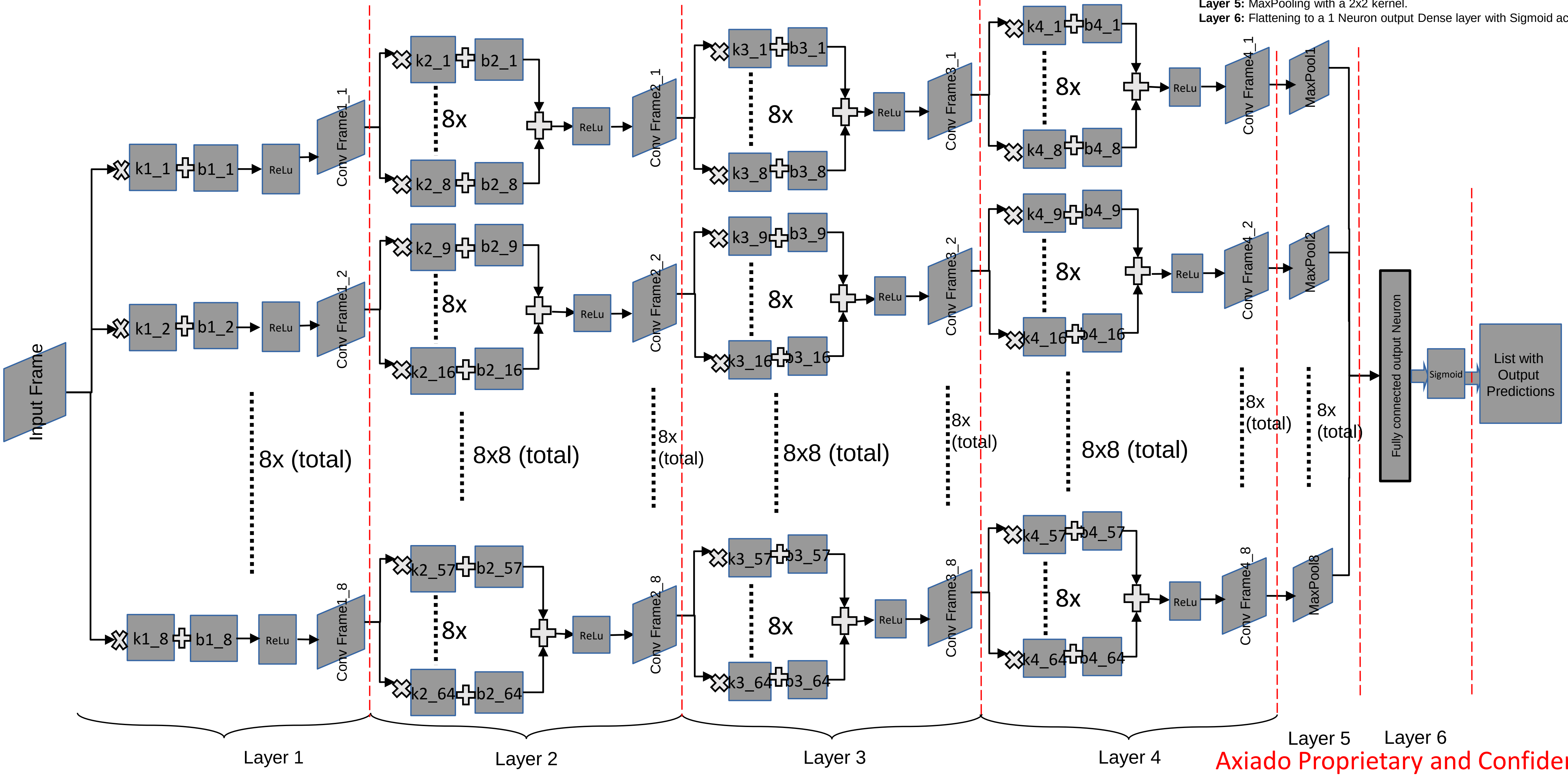
7/22/2020

Four Convolution Layer CNN Accelerator

- The Axiado CNN Accelerator has following specifications (see block diagram on next slide):
  - Software-based configuration of **up to 4 Convolution layers**.
  - Software-based configuration of **up to 8 output frames per Convolution layer**.
  - **3x3** kernels per Convolution layer. In later revision planned choice between 3x3 and 5x5 kernels.
  - One MaxPooling layer with (2x2) filter size after all of the Convolution layers.
  - One fully connected output layer with **1 neuron** for classification between **2** classes (e.g. malicious or benign).
  - Predefined **ReLu** (rectified linear unit) activation functions for each Convolution layer.
  - Predefined **Sigmoid** activation function for the output fully connected layer.

Four Conv. Layer CNN Accelerator

Model:
Layer 1: Conv2D, max 8 filter with (3x3) filter-size, stride=1, no padding, followed by ReLu.
Layer 2: Conv2D, max 8x8 filters with (3x3) filter-size, stride=1, no padding, followed by ReLu. 8 output frames.
Layer 3: Conv2D, max 8x8 filters with (3x3) filter-size, stride=1, no padding, followed by ReLu. 8 output frames.
Layer 4: Conv2D, max 8x8 filters with (3x3) filter-size, stride=1, no padding, followed by ReLu. 8 output frames.
Layer 5: MaxPooling with a 2x2 kernel.
Layer 6: Flattening to a 1 Neuron output Dense layer with Sigmoid activation.



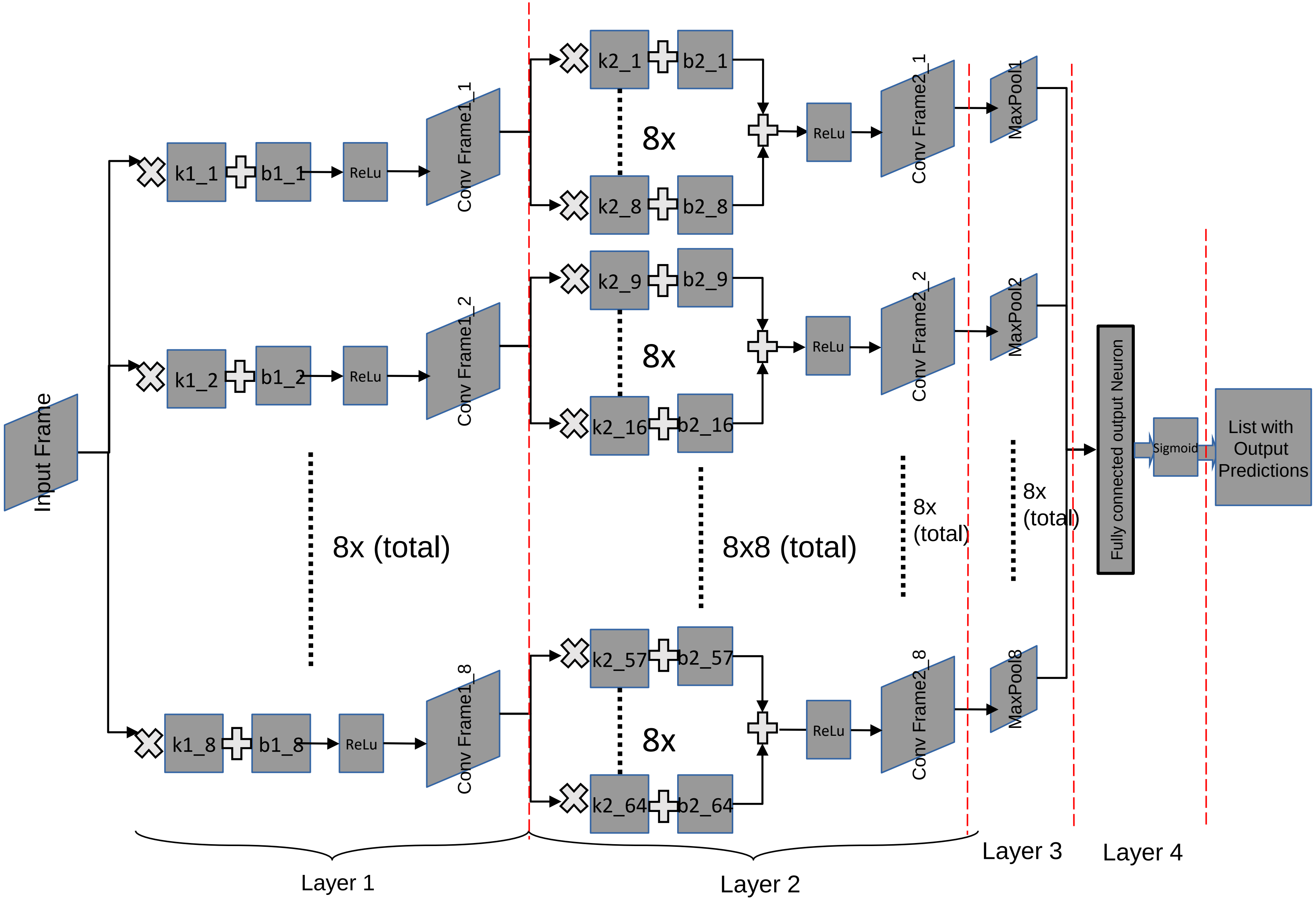
Axiado Proprietary and Confidential

Two Convolution Layer CNN Accelerator Example

- The accelerator block diagram on the next slide is reduced to 2 layers for simplified calculations of the required weight and parameter scaling.

Two Conv Layer CNN Model

Model:
Layer 1: Conv2D, max 8 filter with
(3x3) filter-size, stride=1, no padding, followed by ReLu.
Layer 2: Conv2D, max 8x8 filters with (3x3) filter-size,
stride=1, no padding, followed by ReLu. 8 output frames.
Layer 3: MaxPooling with a 2x2 kernel.
Layer 4: Flattening to a 1 Neuron output Dense layer
with Sigmoid activation.



Axiado Proprietary and Confidential

Two Layer CNN

- A floating point convolution between a frame *f* and a kernel *k₁* (*k₂*) of dimensions (*R, C)* can be expressed as
  - Convolution Layer #1:

$$z_1[m,n] = \sigma\left[(k_1 * f)[m,n]\right] = \sigma\left[\sum_{\mathrm{r}=0}^{\mathrm{R}-1}\sum_{\mathrm{c}=0}^{\mathrm{C}-1} h_1\,[\mathrm{r},c]\cdot f\left[m+\mathrm{r}-int\left(\frac{R}{2}\right), n+\mathrm{c}-int\left(\frac{C}{2}\right)\right] + b_1[m,n]\right]$$

  - Convolution Layer #2:

$$z_2[p,q] = \sigma\left[(h_2 * z_1)[p,q]\right] = \sigma\left[\sum_{\mathrm{r}=0}^{\mathrm{R}-1}\sum_{\mathrm{c}=0}^{\mathrm{C}-1} h_2\,[\mathrm{r},c]\cdot z_1\left[p+\mathrm{r}-int\left(\frac{R}{2}\right), q+\mathrm{c}-int\left(\frac{C}{2}\right)\right] + b_2[p,q]\right]$$

Notes:

1. *int* symbolizes conversion of the calculation in braces from float to integer.
2. These formulas are for the case that padding is applied. With no padding the dot product between kernel and frame-segment cannot start at (0,0).

Axiado Proprietary and Confidential

Two Layer CNN Fix-Point Scaling for Accelerator Inference

- Scaling Parameters:

$$h'_1 = \alpha \cdot h_1, \qquad b'_1 = \alpha \cdot \beta \cdot b_2 \qquad f' = \beta \cdot f$$

$$h'_2 = \alpha \cdot h_2, \qquad b'_2 = \alpha^2 \cdot \beta \cdot b_2 \qquad \alpha, \beta = ScalingFactors$$

$$z'_1[m,n] \;=\; \sigma\left[\sum_r \sum_c h'_1[\mathrm{r},c] f'\left[m + \mathrm{r} - int\left(\frac{R}{2}\right), n + \mathrm{c} - int\left(\frac{C}{2}\right)\right] + b'_1[m,n]\right]$$

$$z'_1[m,n] \;=\; \sigma\left[\sum_r \sum_c \alpha \cdot h_1[\mathrm{r},c]\beta \cdot f\left[m + \mathrm{r} - int\left(\frac{R}{2}\right), n + \mathrm{c} - int\left(\frac{C}{2}\right)\right] + \alpha \cdot \beta \cdot b_1[m,n]\right]$$

$$z'_1[m,n] \;=\; \sigma(\alpha \cdot \beta) \cdot z_1[m,n]$$

$$z'_2[p,q] \;=\; \sigma\left[\sum_r \sum_c h'_2[\mathrm{r},c] \cdot z'_2\left[p + \mathrm{r} - int\left(\frac{R}{2}\right), q + \mathrm{c} - int\left(\frac{C}{2}\right)\right] + b'_2[p,q]\right]$$

$$z'_2[p,q] \;=\; \sigma\left\{\sum_i \sum_l \alpha \cdot h_2[\mathrm{r},c] \cdot \alpha \cdot \beta \cdot z_1\left[p + \mathrm{r} - int\left(\frac{R}{2}\right), q + \mathrm{c} - int\left(\frac{C}{2}\right)\right] + \alpha^2 \cdot \beta \cdot b_2[p,q]\right\}$$

$$z'_2[p,q] \;=\; \sigma(\alpha^2 \cdot \beta) \cdot \sigma\left\{\sum_i \sum_l h_2\,[\mathrm{r},c] \cdot z_1\left[p + \mathrm{r} - int\left(\frac{R}{2}\right), q + \mathrm{c} - int\left(\frac{C}{2}\right)\right] + b_2[p,q]\right\}$$

$$z'_2[p,q] \;=\; \sigma(\alpha^2 \cdot \beta) \cdot z_2[p,q] \Rightarrow \textit{for a 2-layer CNN the result needs to be multiplied by} \frac{\mathbf{1}}{(\boldsymbol{\sigma}(\boldsymbol{\alpha}^2 \cdot \boldsymbol{\beta}))}$$

Note:

- *int* symbolizes conversion of the calculation in braces from float to integer.
- Assuming for the hidden layers ReLu activation functions (σ).

Axiado Proprietary and Confidential

Model Training on Arbitrary Alphanumeric Data – Performed on Host Computer

1. One-time procedure to produce a “dictionary” from the training material:

Training data: e.g.

DATA:	LABEL:
Vkd s gah dso vas sdo a; sdfgv	malicious
Fa ao;a;; a aeufw sd ;s gf;	benign
;asogra ; as ;asdo ds; o	benign
Af ;a iogi ; a; asd ; d	malicious

Tokenize

Dictionary: e.g.

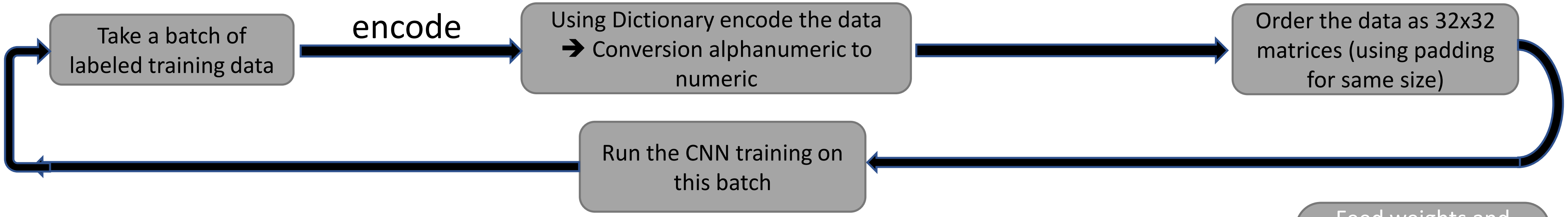
DATA Symbol:	Value:
Vkd	0
S	1
gah	2
dso	3
….	

LABEL Symbol:	Value:
malicious	0
benign	1

2. One-time procedure to design a CNN model:

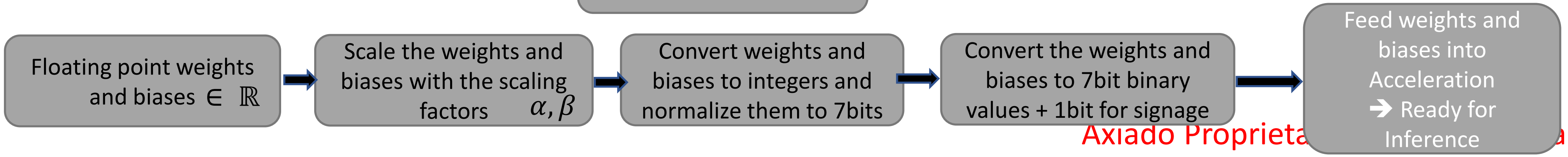
Design a CNN Model (can be the full capability of the CNN Accelerator)

3. Train the model Parameter (weights and biases):



Take a batch of labeled training data
encode
Using Dictionary encode the data ➔ Conversion alphanumeric to numeric
Order the data as 32x32 matrices (using padding for same size)
Run the CNN training on this batch

4. Prepare the trained parameters for the CNN Accelerator:

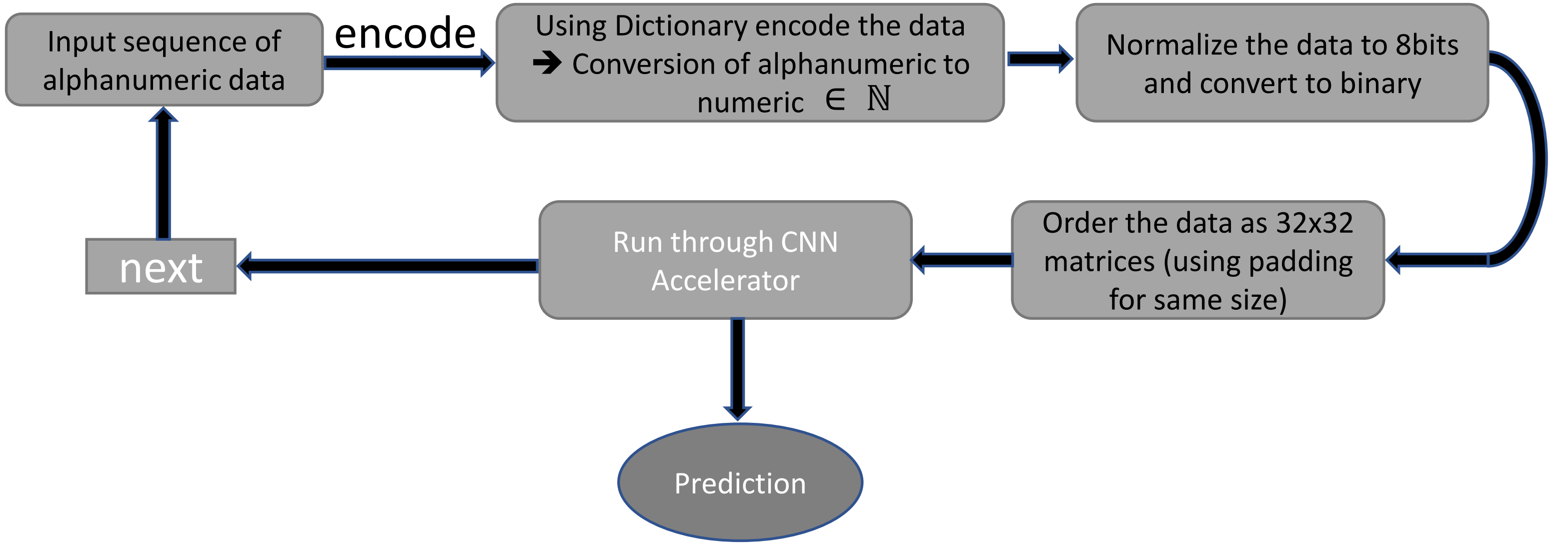


Floating point weights and biases ∈ ℝ
Scale the weights and biases with the scaling factors α, β
Convert weights and biases to integers and normalize them to 7bits
Convert the weights and biases to 7bit binary values + 1bit for signage
Feed weights and biases into Acceleration ➔ Ready for Inference

Axiado Proprieta al

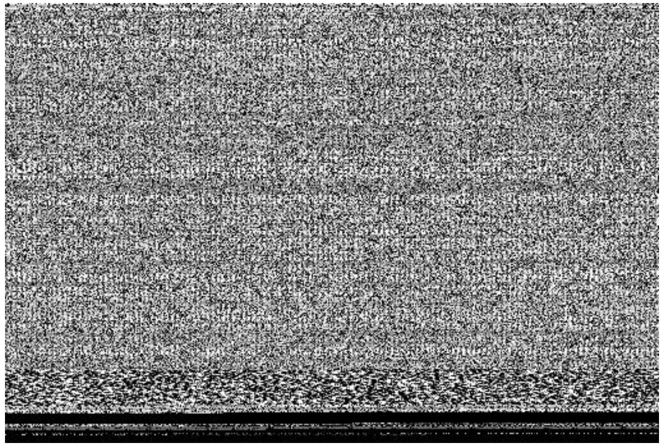
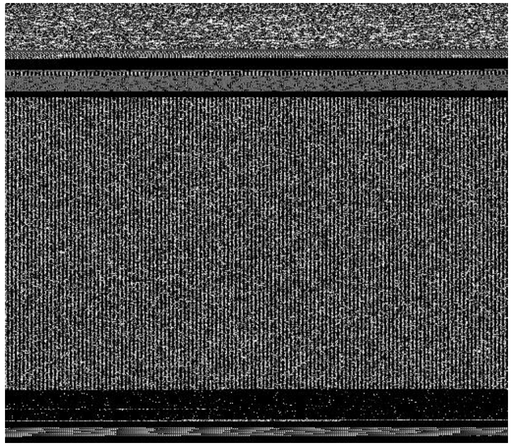
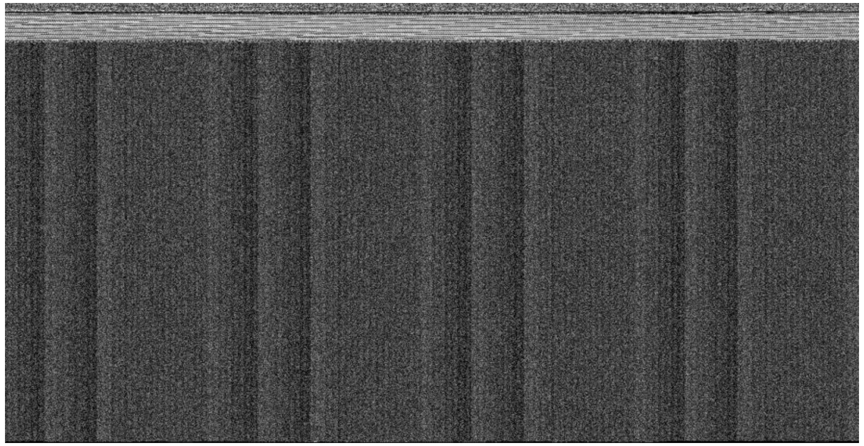
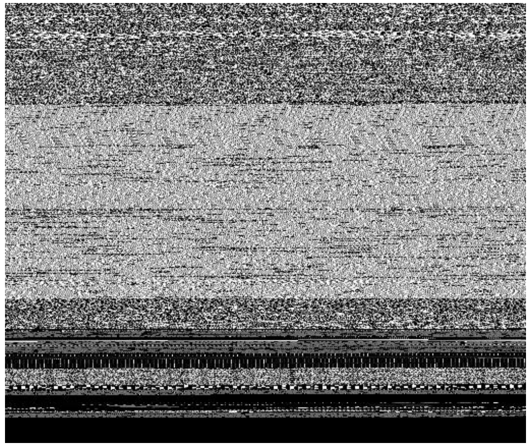
Inference Procedure with the CNN Accelerator

- After the training the resulting weights and biases are scaled, normalized to 7+1bits and converted to binaries.
- The binary weights and biases are loaded on the CNN Accelerator, which is then ready for inferencing.
- The block diagram on the right shows the procedure required for inferencing using alphanumeric input data.



Input sequence of alphanumeric data
encode
Using Dictionary encode the data ➔ Conversion of alphanumeric to numeric ∈ ℕ
Normalize the data to 8bits and convert to binary
Order the data as 32x32 matrices (using padding for same size)
Run through CNN Accelerator
next
Prediction

Application of CNN's on IP Packets

	Example 1	Example 2
Gatak		
Lollipop		

Appendix

Inference Engine Models

Memory Heat Map (MHM)

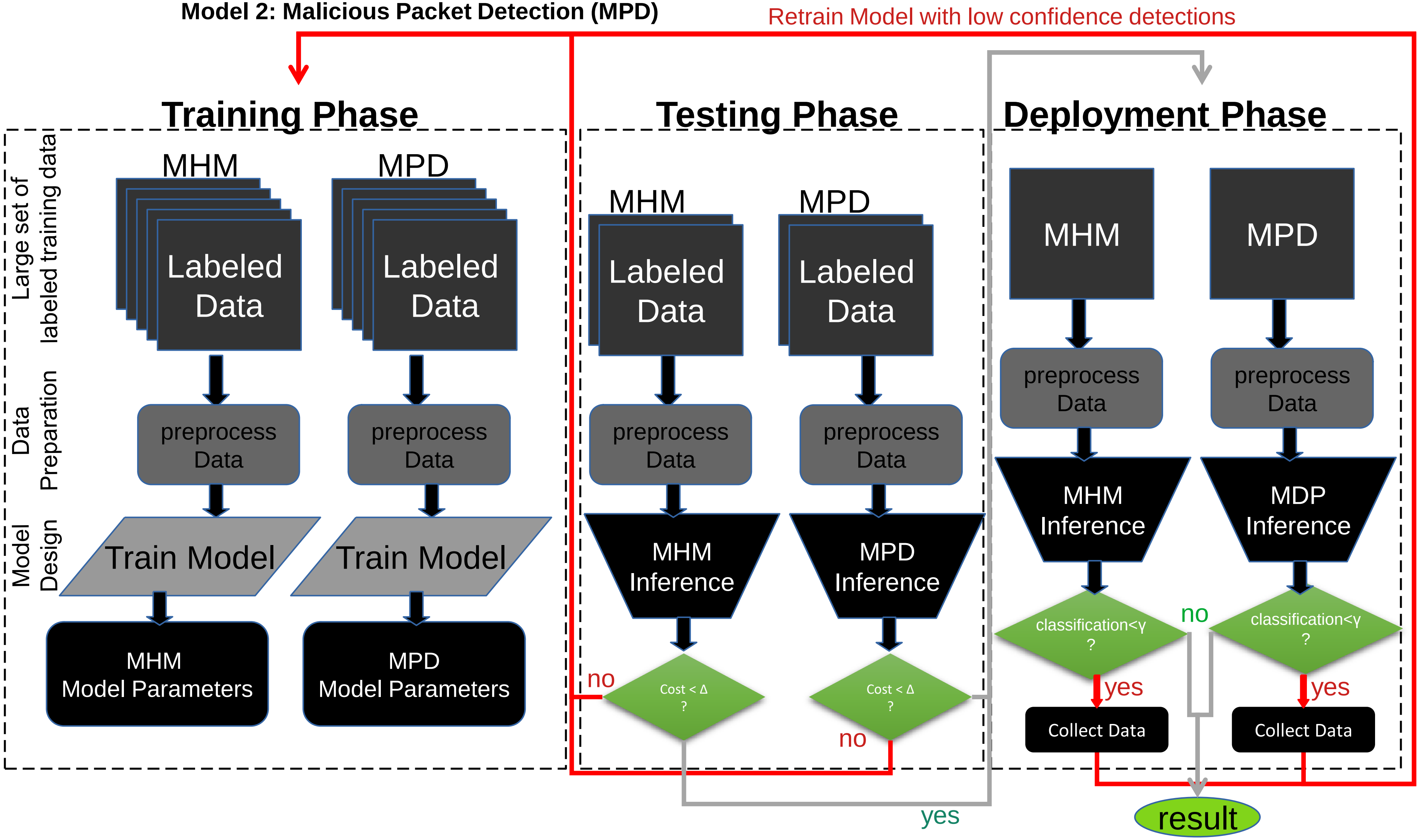
- Detection of system-wide anomalies by monitoring the behavior of memory accesses in the host system of the EDGE IQ.
- Conversion of the memory occupation into a “Memory Heat Map”.
- Train a Convolutional Neural Network on the MHM of the host system to detect anomalies.

Malicious Packet Detection (MPD)

- Detection of anomalies in data packets being sent in and out of the EDGE IQ.
- Train a Convolutional Neural Network to detect anomalies in network packets or other communication with the outside world, to detect malware or other attacks.

Inference Engine Preparation and Deployment

Model 1: Memory Heat Map (MHM)
Model 2: Malicious Packet Detection (MPD)



Axiado Proprietary and Confidential