Porting a Convolutional Neural Network to Hardware

International Conference for Pattern Recognition 2020

D. -Od. G. Sotiropoulos, Dr. M.Sc, G. - P. K. Economou,
Computer Engineering and Informatics Department,
University of Patras, Greece

Stereoscopic Vision

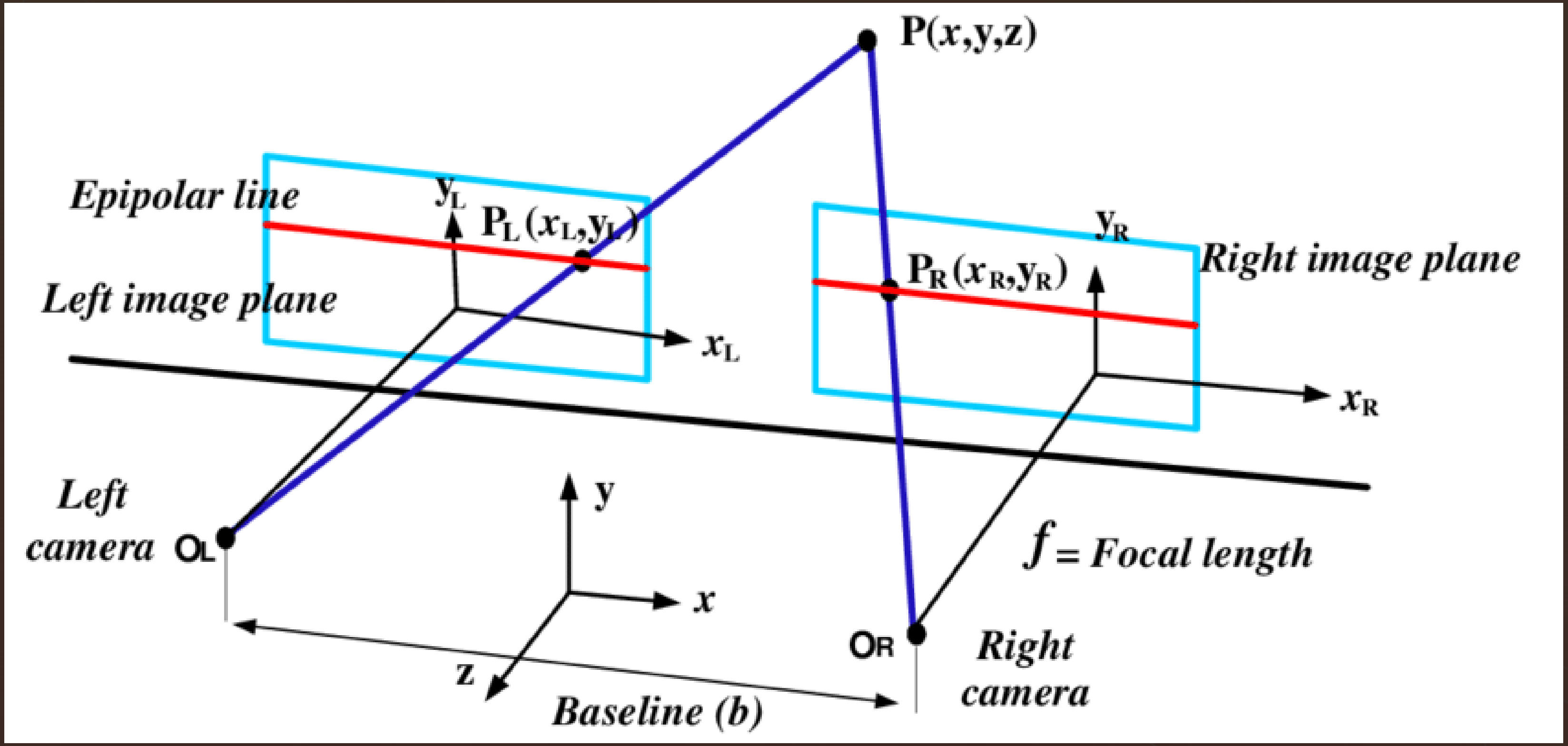
- **In Humans**
  - The distance between the eyes is constant and known.
  - The eyes can focus on a single point in the scene.
    - The point of focus is where the imaginary lines from the eyes to the scene intersect.
      Humans unconciously focus the intersection on a surface.
    - The angle of the eyes is known.
  - Calculation of the distance between the eyes is possible through trigonometry.

Stereoscopic Vision

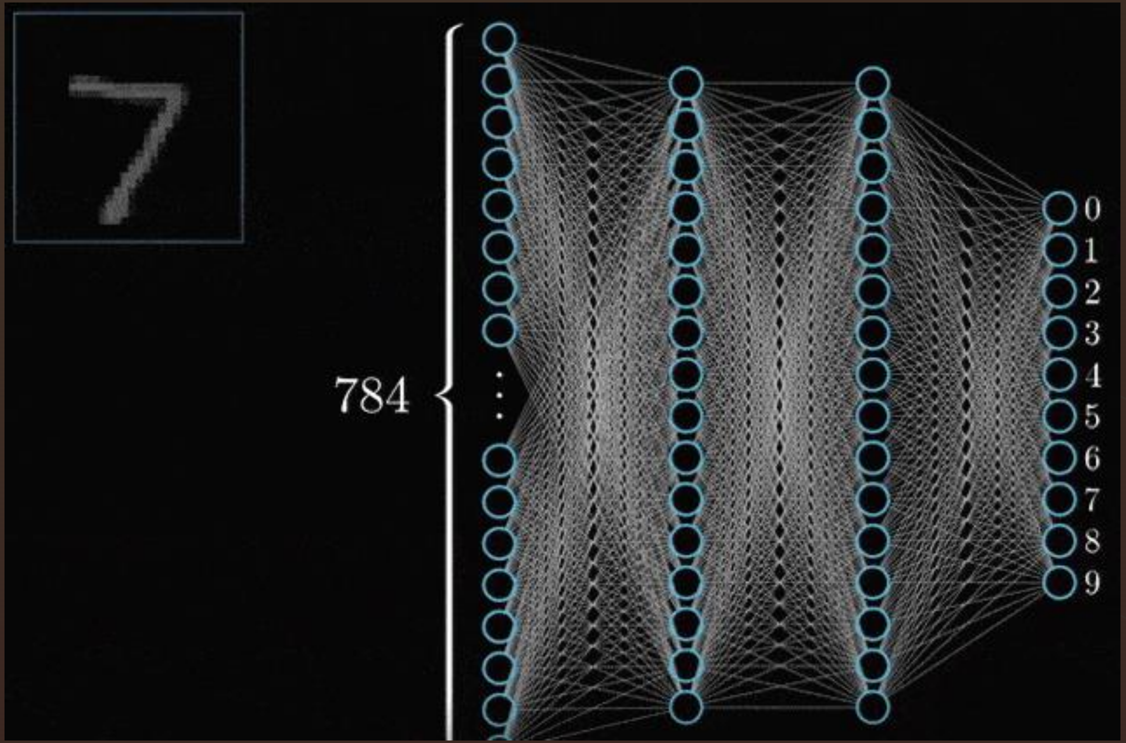
- In Machines
  - The distance between the cameras is constant and known.
  - *Focusing on a point in the scene requires complex recursive processing that is directly correlated to the distance of the point.*
  - Instead: The cameras are aimed in parallel to the scene.
  - Approximation of the distance is possible through pixel *disparity*

Disparity: The distance in pixels along the epipolar line between a pixel from the left image to a pixel from the right image that both illustrate the same point of the scene

Stereoscopic Vision



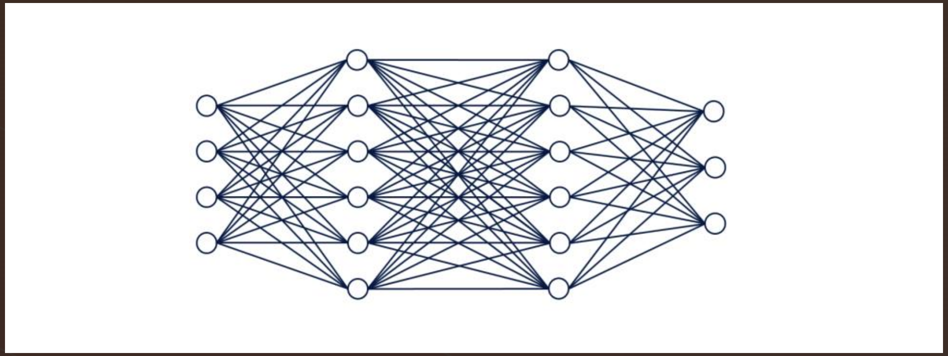
Artificial Neural Networks (ANNs)



- Most often used to classify data into classes.
- Fine tuned through training to produce expected result.
- Can extract abstract patterns of input.

Artificial Neural Networks (ANNs)

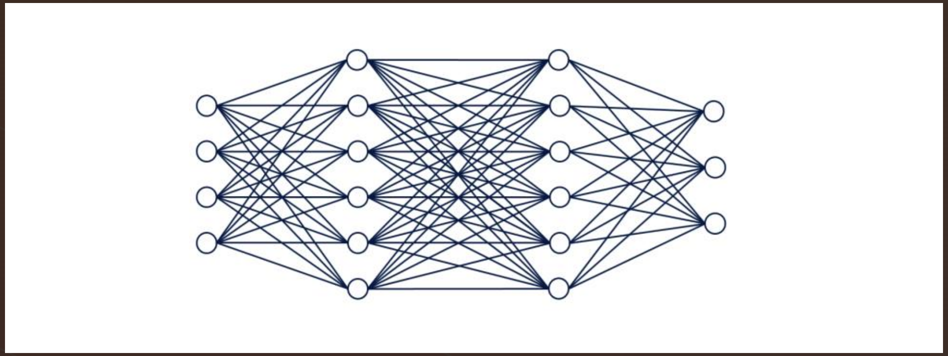
- Neurons are ordered in layers
- Each Neuron is fed as input ALL of the output of the Neurons of the previous layer
- A neuron is "activated" if it's value reaches a certain "theshold" to pass its output to the next layer.



Artificial Neural Networks (ANNs)

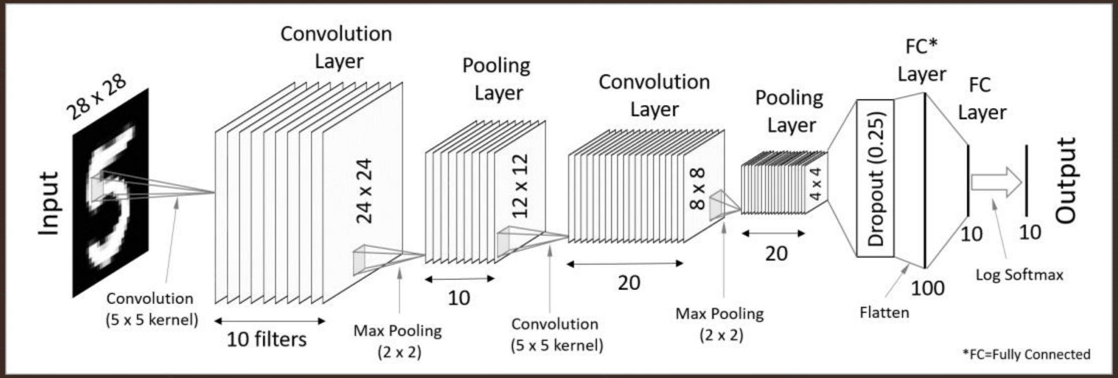
Artificial Neuron Process

- A sum of products
  - Each input is multiplied by a corresponding "weight", a native parameter of the neuron for that particular input.
- *k-th* Neuron
  Function: $(w_{k0}x_0+w_{k1}x_1+\ldots+w_{kn}x_n+b_k)$
  - x_i : Input *i* of neuron
  - w_{ki} : Weight *i* of neuron *k*
  - b_k : Bias of neuron *k*



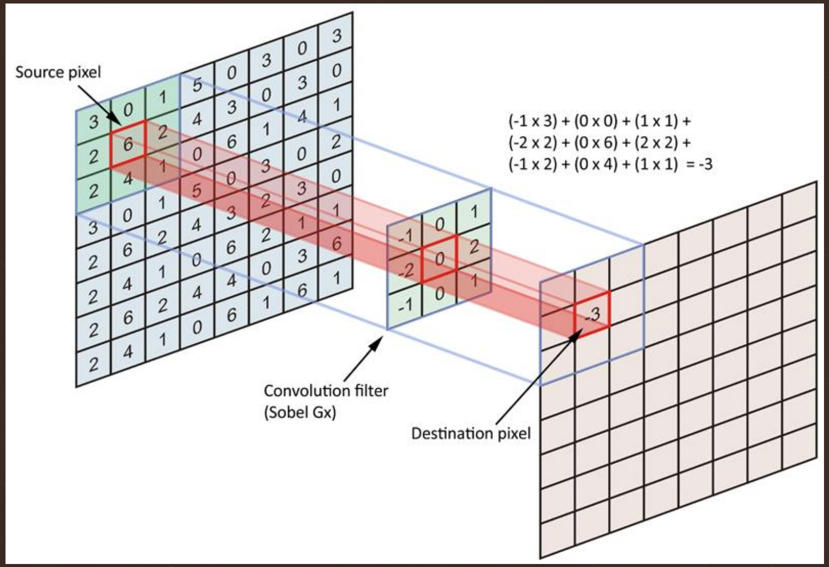
Convolutional Neural Networks

- A type of ANN that processes images (matrices) instead of scalars.
- Performs convolution instead of simple weight multiplication.
- Computationally expensive



Convolution

- A sum of products
- Convolutional filter applied on all possible positions of a larger input image.
- Each input pixel of the input image is multiplied by the corresponding parameter of the applied filter. The intermediate products are summed to produce the resulting output pixel.
- Result: a pixel corresponding to the location of input image where the center pixel of the filter is applied to.


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter (Sobel Gx)
Destination pixel

Convolutional Neuron vs Classic Artificial Neuron

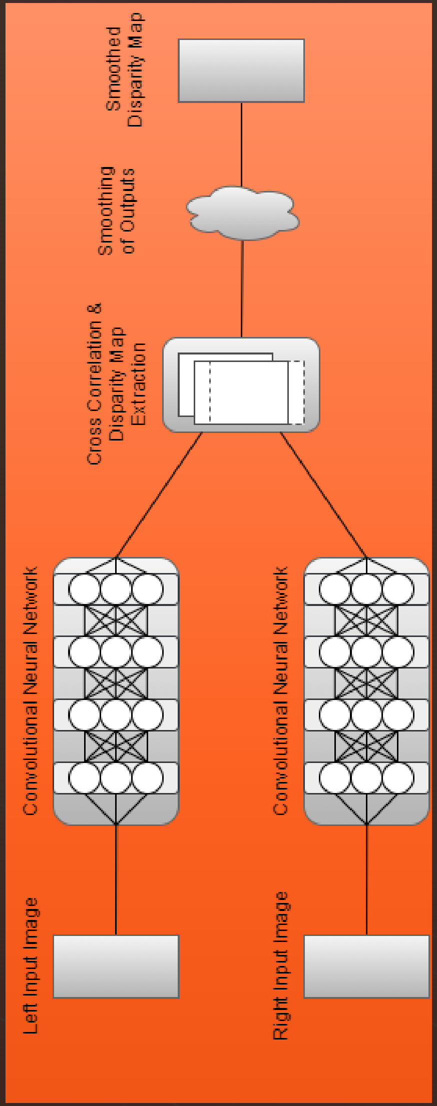
- Convolutional Neuron
  - Convolutional Neuron Function: $(W_{k0} * X_0 + W_{k1} * X_1 + \ldots + W_{kn} * X_n + b_k)$
  - $*$: Shifting Dot Product (aka Convolution)
  - X_i: i-th input image (matrix)
  - W_{ki}: i-th weight array of neuron *k*(aka convolutional filter)
  - b: bias of neuron *k (scalar)*

- Classic Artificial Neuron
  - Artificial Neuron Function: $(w_{k0}x_0 + w_{k1}x_1 + \ldots + w_{kn}x_n + b_k)$
  - Inputs x_i are scalars.
  - Weights w_i are scalars.
  - Bias b is a scalar.

The convolutional neuron's function can be described as a sum of sums of products.

Content CNN Model Achitecture

- Siamese Architecture: 2 identical CNN paths
- Cross Correlation: Sliding dot product layer.
- Disparity Map Extraction
- Smoothing

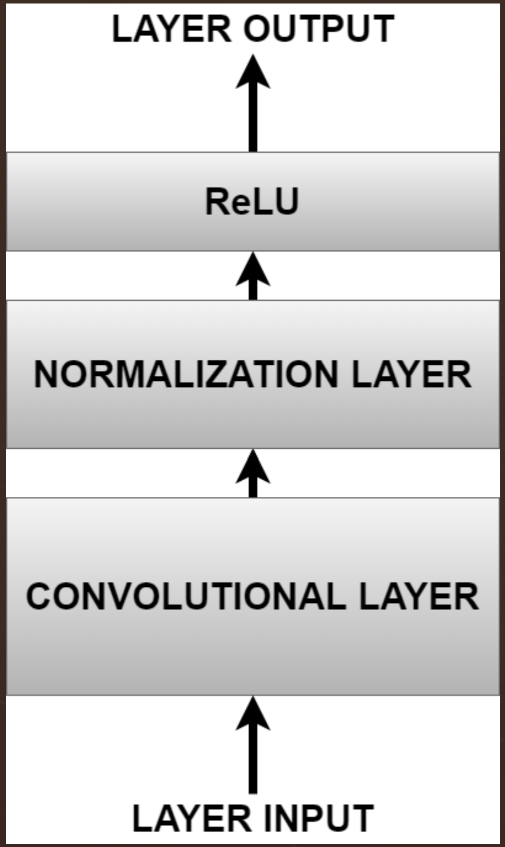


Smoothed Disparity Map
Smoothing of Outputs
Cross Correlation & Disparity Map Extraction
Convolutional Neural Network
Convolutional Neural Network
Left Input Image
Right Input Image

Original paper: W. Luo, A. Schwing, and R. Urtasun, “Efficient deep learning for stereo matching,” 2016.

Content CNN Specifications: CNN Part

- 2 Input Images of size 1242x375px (grayscale)
- 9 Fully Connected Layers
- Convolutional Sub-Layer
  - 64 Neurons
  - 3x3 Convolutional Filters
- Batch Normalization Layer
- Rectified Linear Unit Layer



Spatial Batch Normalization

$$y = \frac{x - mean(x)}{std(x)}\gamma + \beta$$

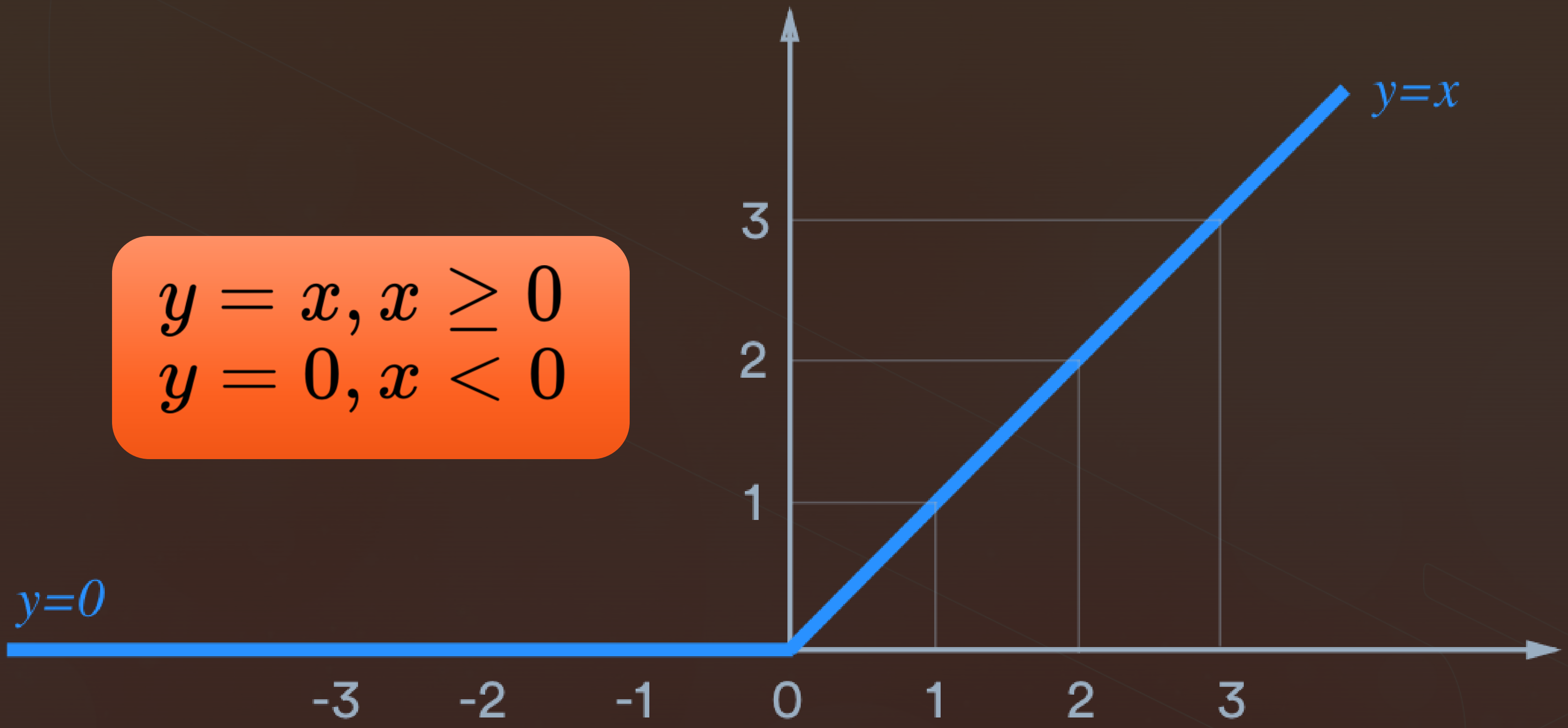
- *y*: normalized output
- *x*: input
- *mean(x)*: minibatch mean value
- *std(x)*: minibatch standard deviation value
- *γ*: gamma trainable parameter
- *β*: beta trainable parameter

Fast normalization tecnhnique utilizing statistical parameters (mean(x), std(x)) extracted from the training set.

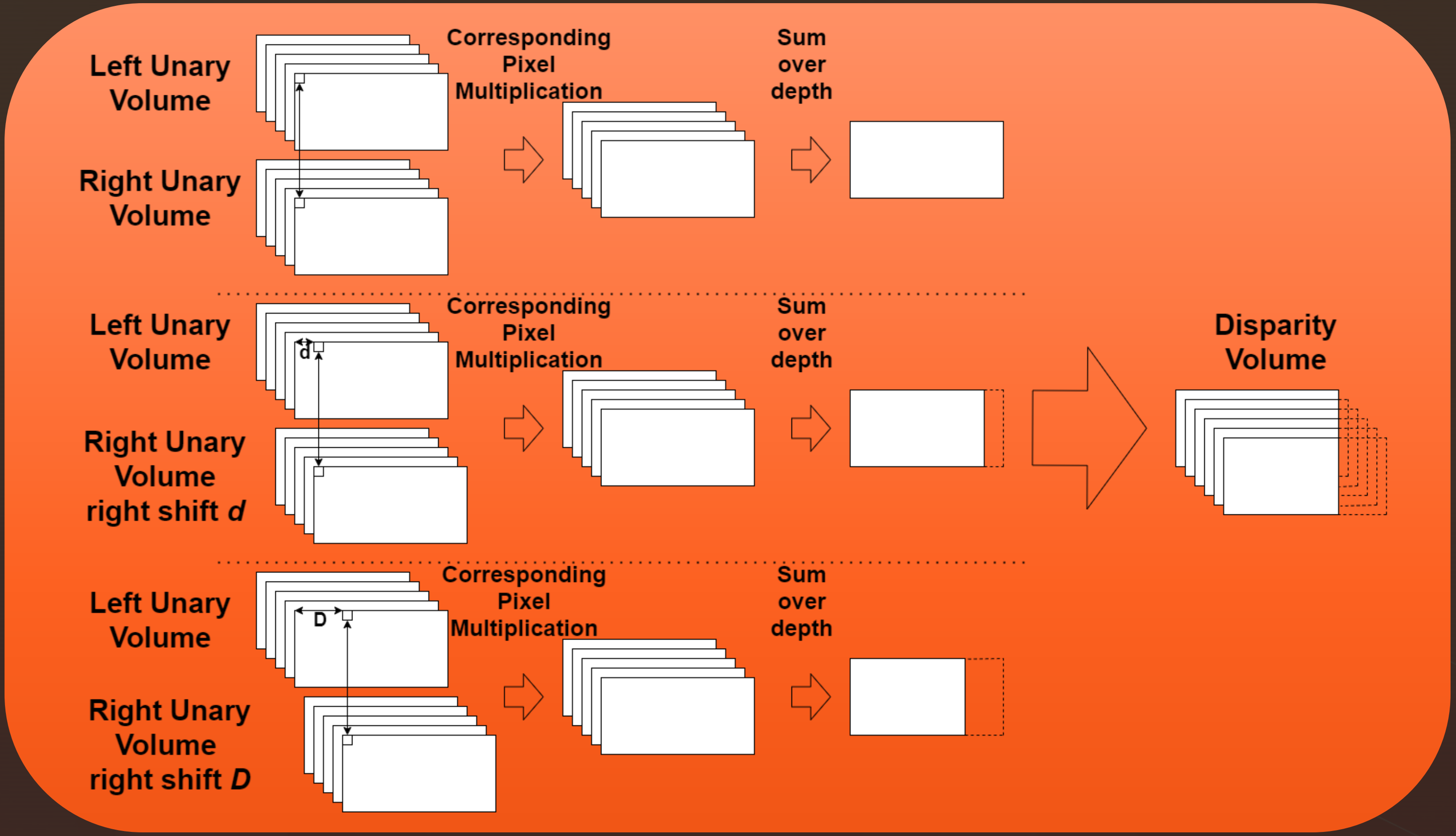
S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

Rectified Linear Unit

$$y = x, x \geq 0$$
$$y = 0, x < 0$$



Cross Correlation


Left Unary Volume
Right Unary Volume
Corresponding Pixel Multiplication
Sum over depth
Left Unary Volume
d
Right Unary Volume right shift *d*
Corresponding Pixel Multiplication
Sum over depth
Disparity Volume
Left Unary Volume
D
Right Unary Volume right shift *D*
Corresponding Pixel Multiplication
Sum over depth

Calculates the dot product of the CNN output tensors over the depth dimension for *D* shifts of one tensor over the other where *D* is the maximum disparity value.

Disparity Map Extraction

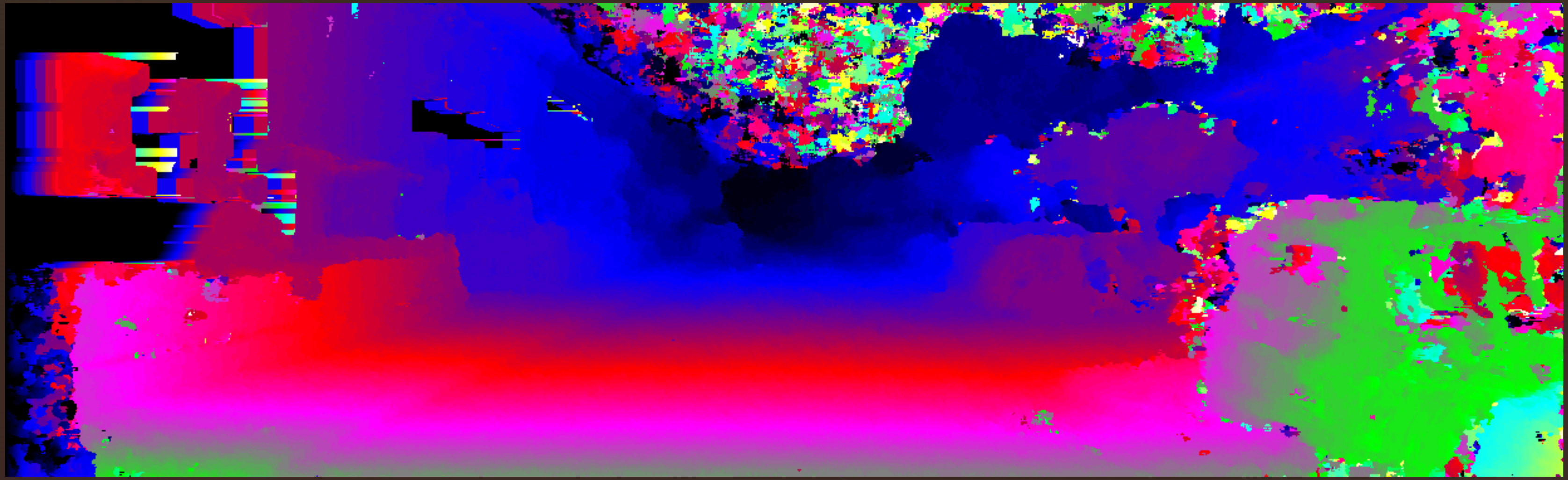
- Disparity Volume
  - The resulting tensor from cross correlation is a 3d reconstruction of the scene.
  - Pixel values depict the propability that material is present in the corresponding 3d quordinates.

- Extracting a disparity map
  - Essentially moving 3d space into 2d by assigning the 3rd dimension as color (pixel value).
  - Assign the corresponding pixel of the disparity map the index of the pixel with the maximum value of the corresponding depth vector.

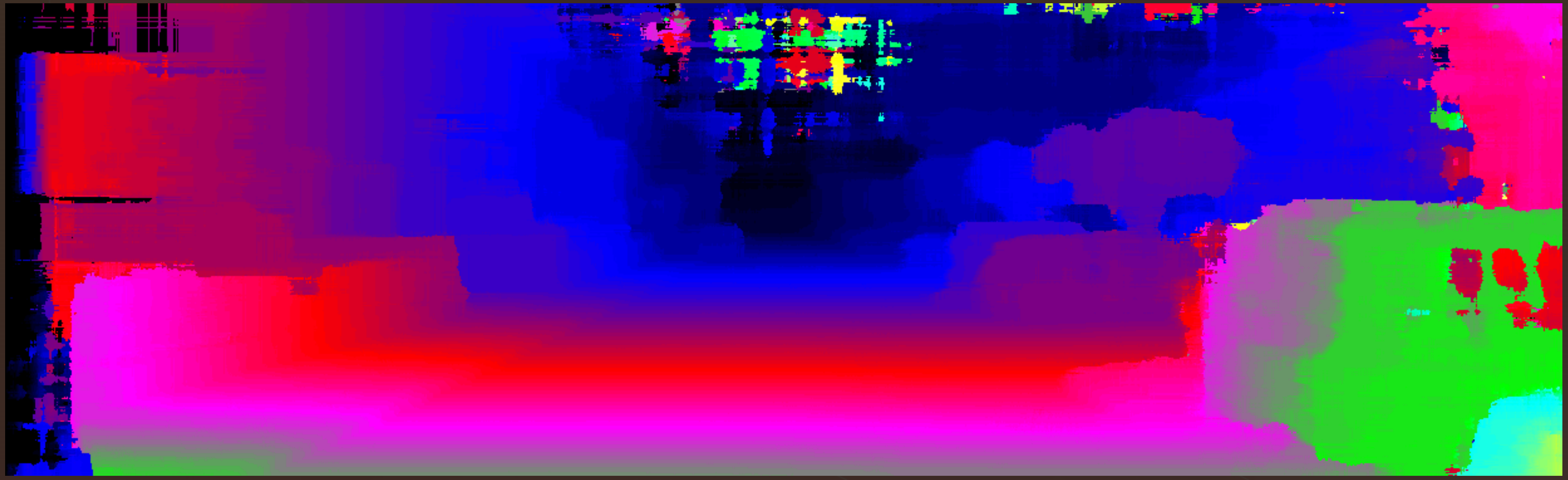
Demonstration of Content-CNN Execution

Original Input

Demonstration of Content-CNN Execution

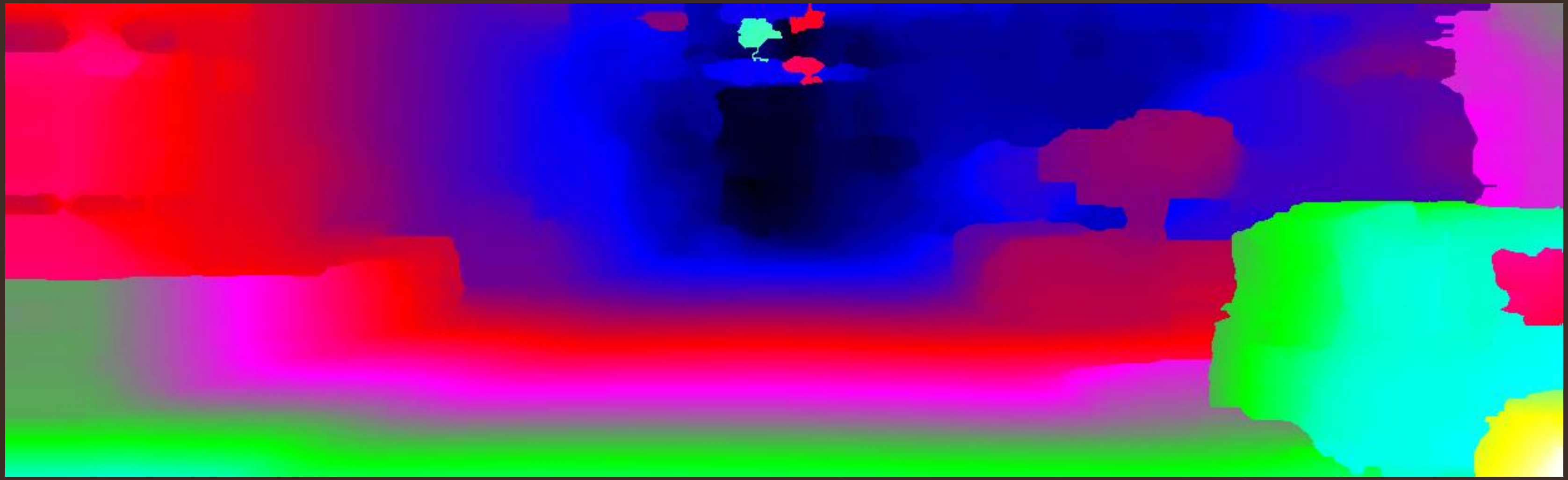

Raw Content-CNN Output

Demonstration of Content-CNN Execution



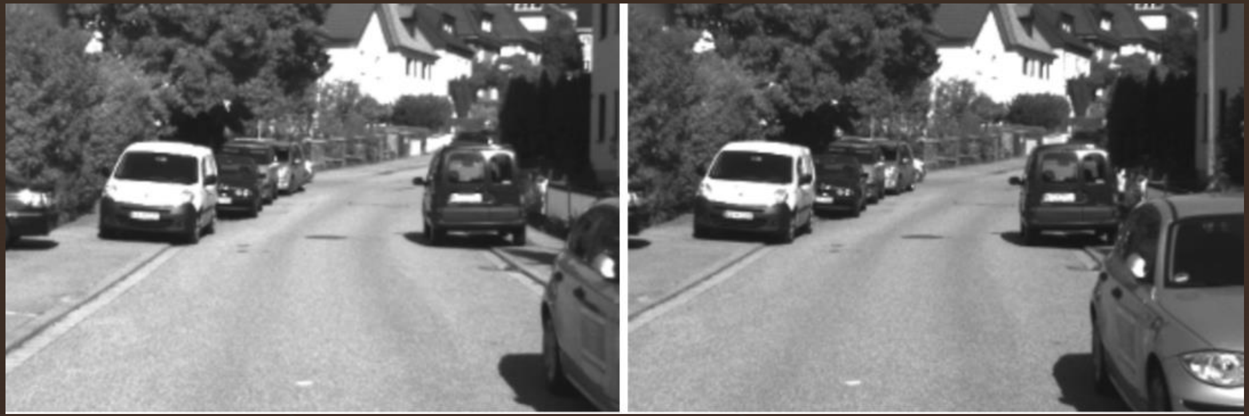
NYU and SGBM Smoothing

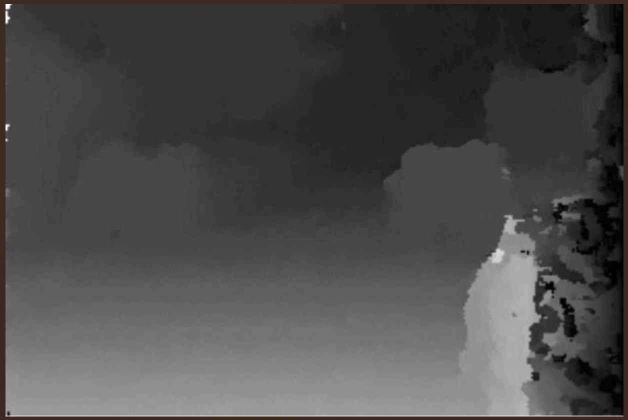
Demonstration of Content-CNN Execution



Original Paper Complex Smoothing

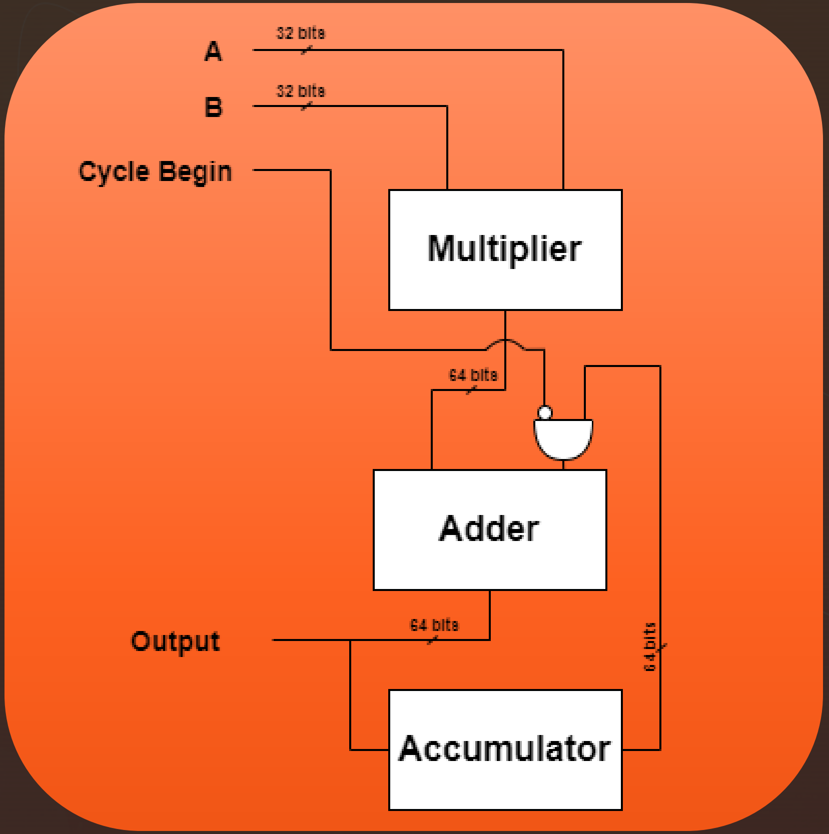
Our matLab Reconstruction Results





Input and output of our serial matLab reconstruction of Content-CNN

Convolutional Kernel



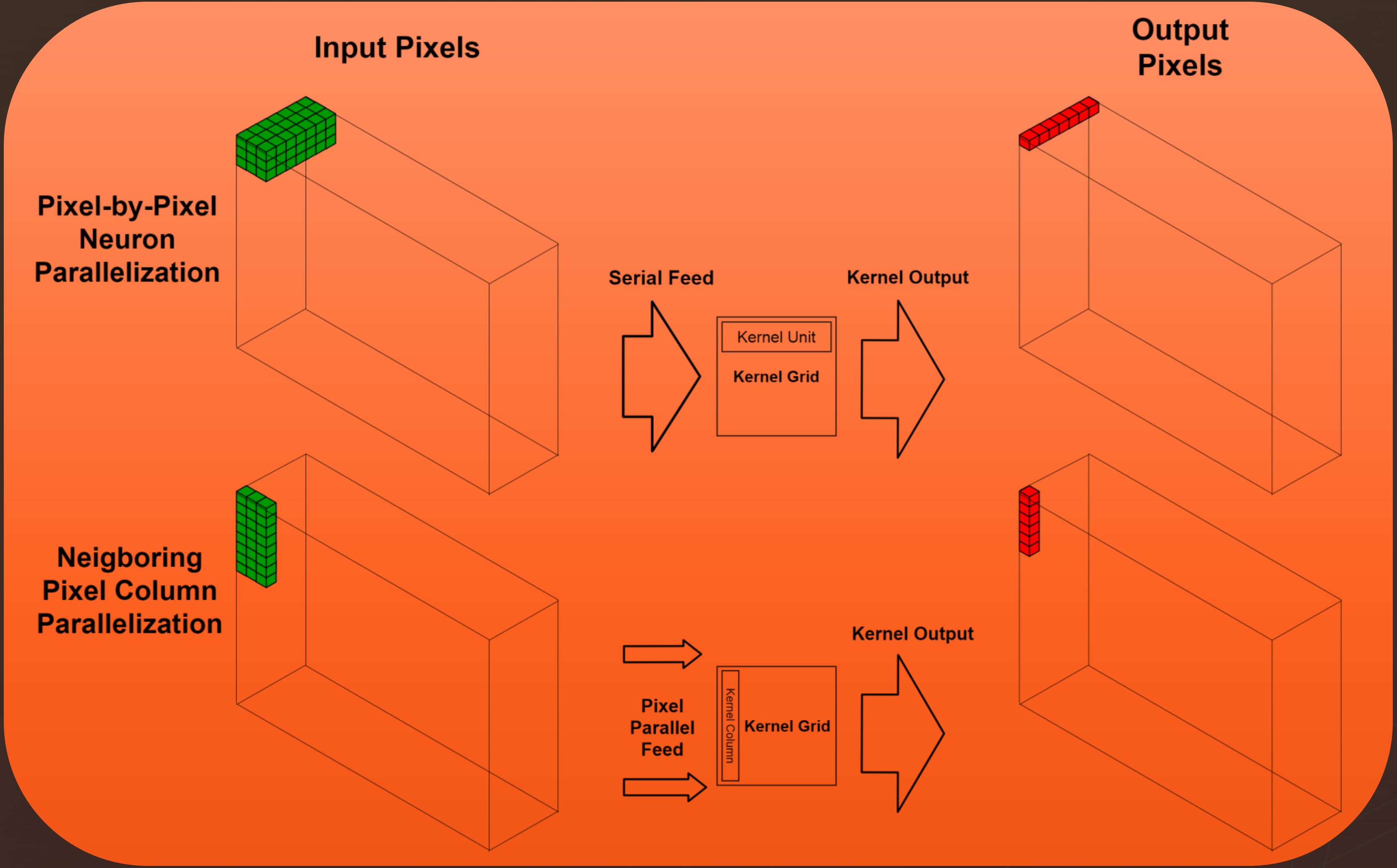
- Serial Implementation
  - Performs multiplication of a pixel by a parameter.
  - Adds the product to the accumulating result.
  - Stores the accumulating result in a buffer (Accumulator)

Double Parallelization Strategy

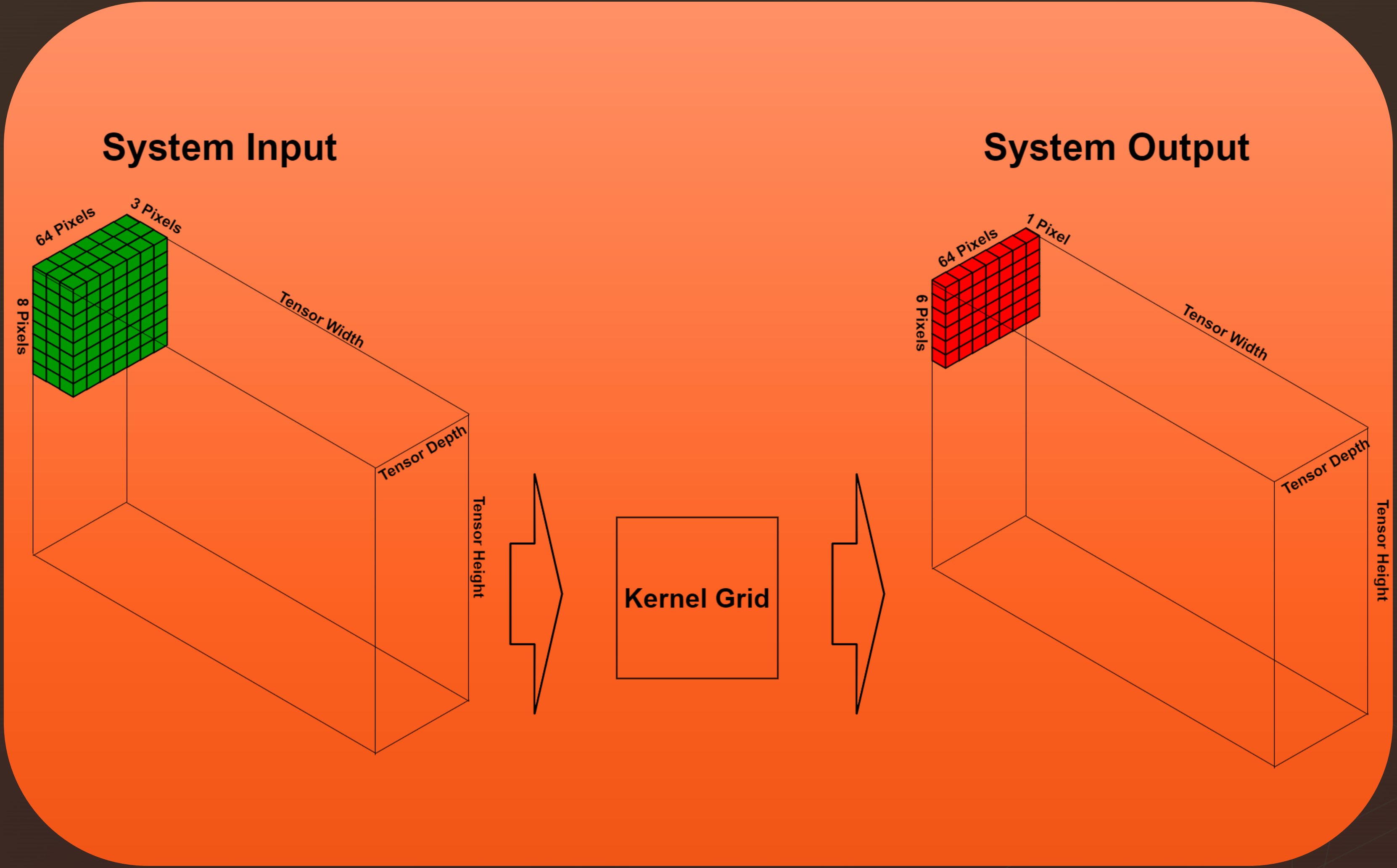
- Parallelization of Layer Output: Depth Vector Output
  - Serial processing of convolution *(convolutional kernel)*
  - Serial processing of neuron output *(pixel by pixel processing)*

- Parallelization of Neigbouring Pixels: Height Vector Output
  - Multiple systems of depth vector processing

Double Parallelization Strategy



Double Parallelization Strategy

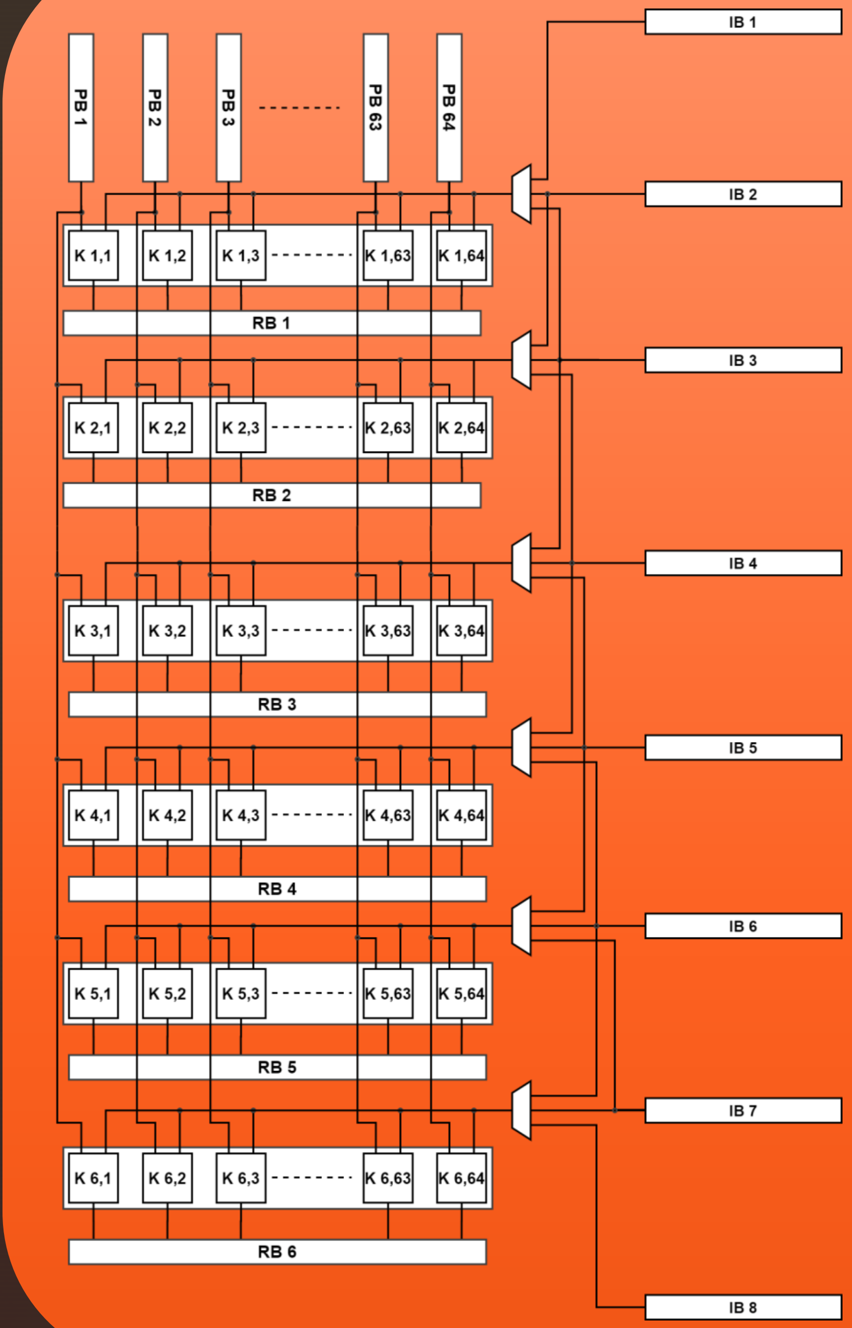


Kernel Grid

How to organize a set of kernels in order to realize the parallelization strategies

- Depth Vector:
  - Common pixel inputs
  - Individual parameters (each kernel performs the function of a different neuron)
- Row of 64 Kernels

- Height Vector:
  - Different pixel inputs (each kernels processes differen t pixels of the same neuron output)
  - Common parameters
- Column of *k* Kernels



Kernel Grid

- Rows of 64 Kernels
- Columns of 6 Kernels

Normalization & Cross-correlation Kernel Support

- Batch Normalization:

$$y = \frac{x - mean(x)}{std(x)}\gamma + \beta$$

- *Mean(x), std(x), γ, β* constant during execution.
- The function can be reduced to a first degree polynomial

$$y = \alpha x + \lambda$$

$$\lambda = \frac{\gamma \cdot mean(x)}{std(x)}$$

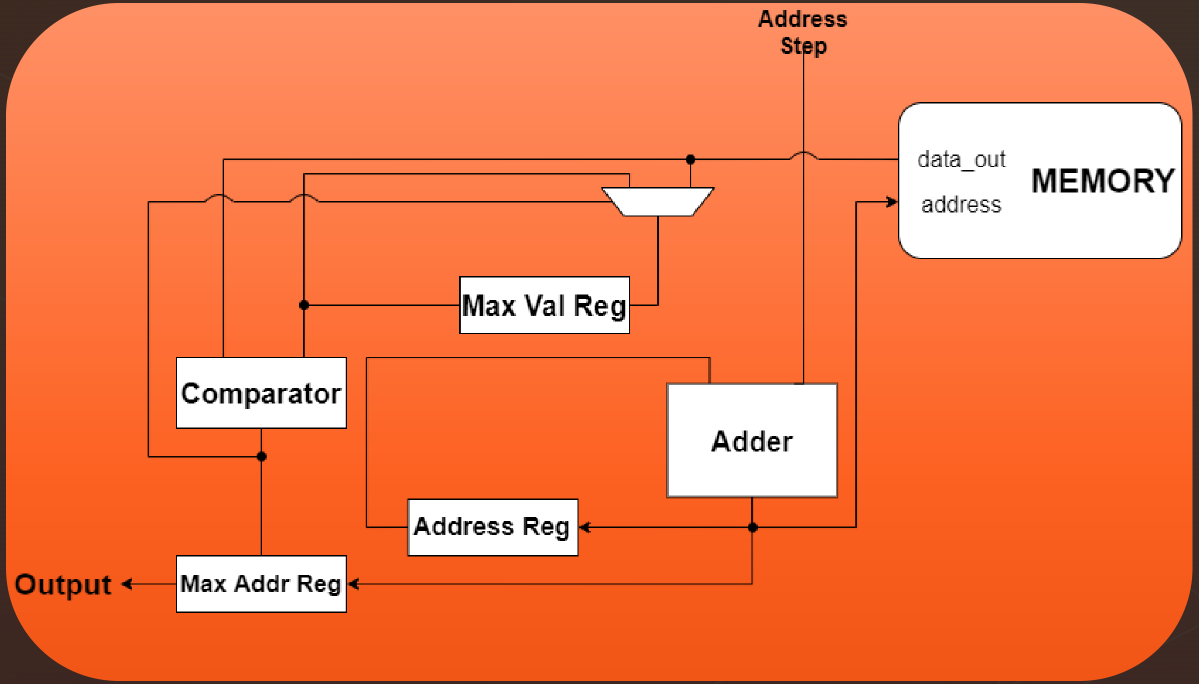
$$\alpha = \frac{\gamma}{std(x)}$$

A sum of products processable by the kernel

- Cross Correlation
- A sum of products, also processable by the kernel

ReLU & Disparity Map Extraction in Hardware

- ReLU
  - Can be easily implemented with multiplexers.
- Disparity Map Extraction
  - Simple specialized circuit



I/O Management

- Input Pixel Buffer System
- Input Parameter Buffer System
- Output Results Buffer System

Results Buffer

- Parallel Load
- Serial Store to memory
- Exploits long processing cycle to fully nest the memory store cycle into the processing cycle.

Parameter Buffer System

- Cyclic Shift Buffer System
  - All neuron specific parameters loaded in the buffer
  - Feedback of used parameters to the tail of the buffer to be used in next processing cycle
  - Auxiliary buffer

Horizontal and Vertical Overlap of Convolution by a 3x3 Filter

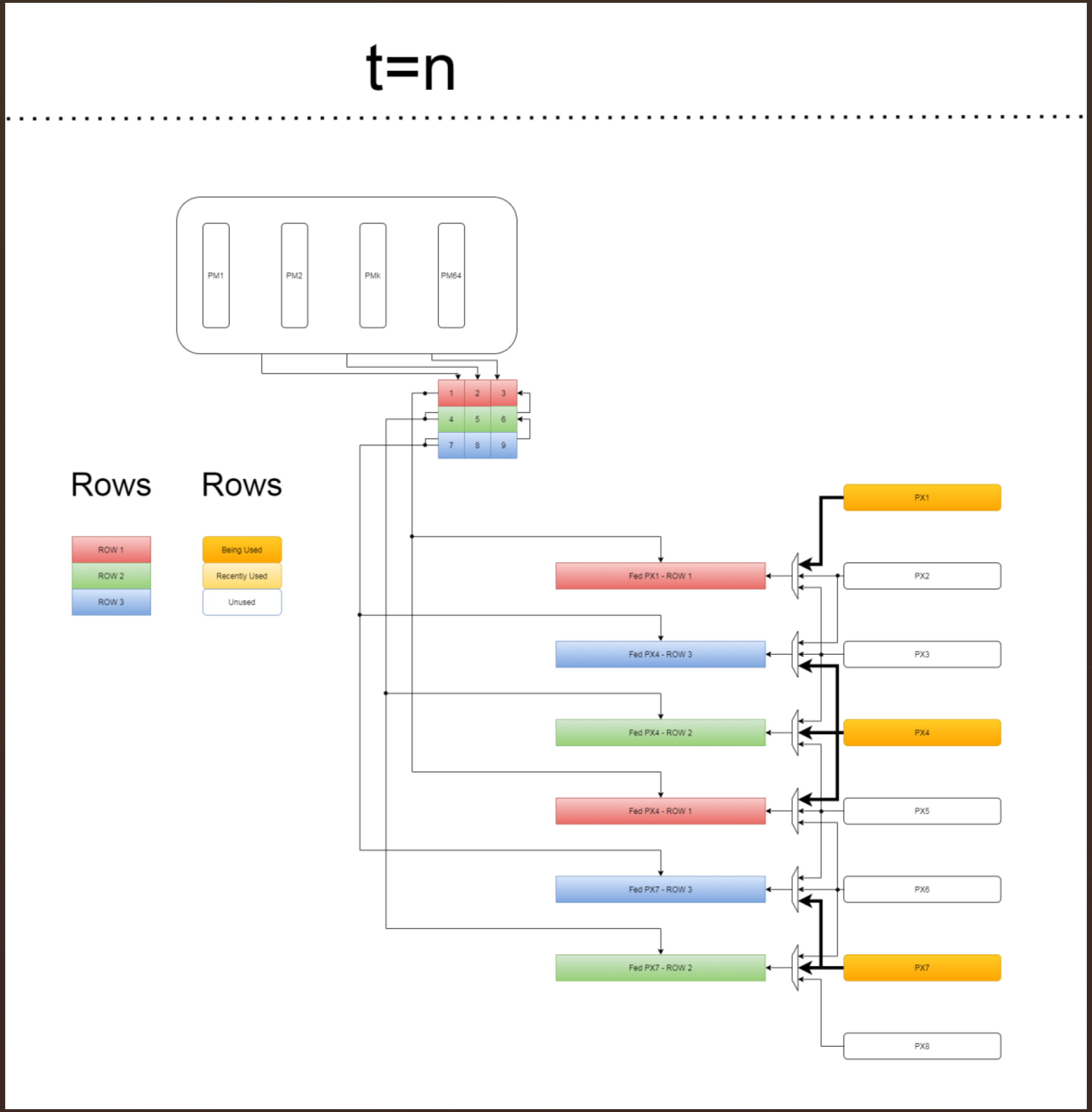
- Horizontal Overlap
  - Input pixel columns corresponding to the 2nd and 3rd columns of the applied filter constitute the 1st and 2nd input pixel columns of the convolution of horizontal to the right neigbouring output pixel

- Vertical Overlap
  - Input pixel rows corresponding to the 2nd and 3rd rows of the applied filter constitute the 1st and 2d input pixel rows of the convolution of vertical to below neigbouring output pixel

Horizontal and Vertical Overlap of Convolution by a 3x3 Filter

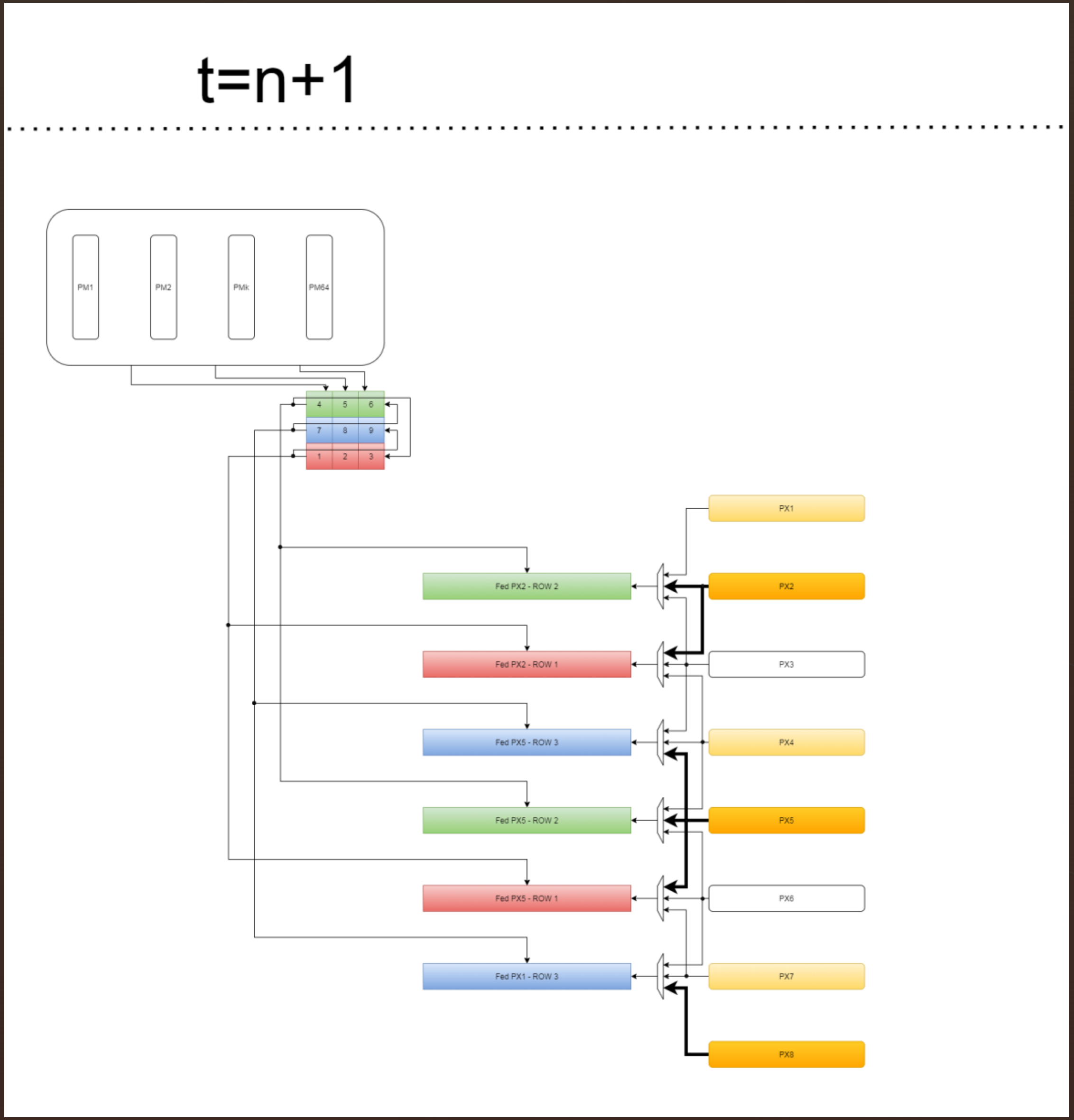
- Horizontal overlap exploitation
  - Cyclic buffer:
  - Pixels in the head that will **not** be used in the future are ejected from the buffer while new input pixels are read from memory and store to the tail
  - Pixels in the head that will be used in the future are fed back to the tail of the buffer

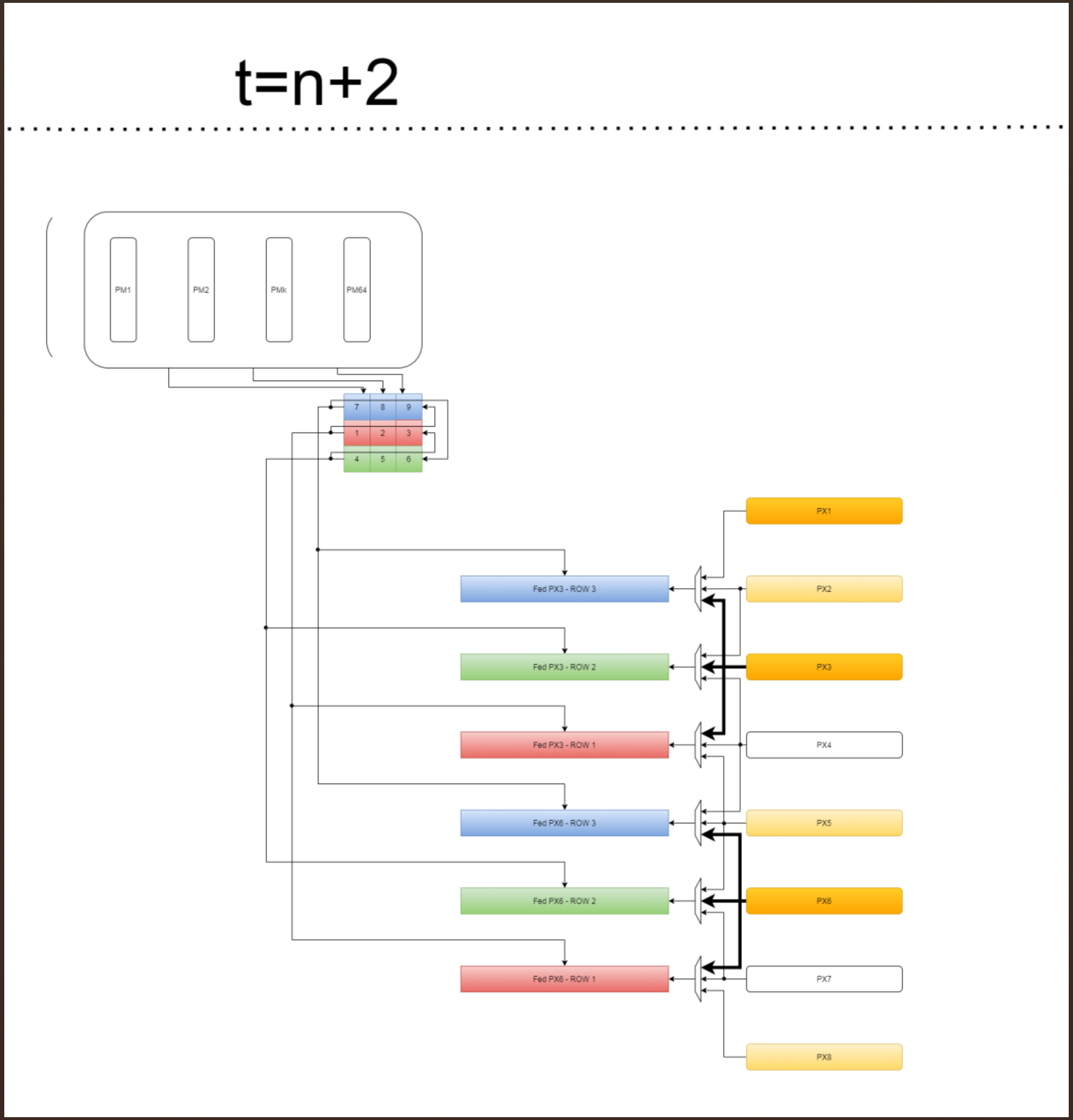
- Vertical overlap exploitation
  - Line buffer:
  - Each buffer holds pixels from a single input image row
  - Smart feed of pixels from a line buffer to multiple kernel units (kernel grid rows)



Pixel Buffer Behaviour over 3 phases of convolution cycle

Pixel Buffer Behaviour over 3 phases of convolution cycle





Pixel Buffer Behaviour over 3 phases of convolution cycle

Conclusion

- Total of $5.5 \cdot 10^{12}\ \mathit{calculations}$
- Fit in $4.5 \cdot 10^{9}\ \mathit{clock\ cycles}\ (cc)$ for a single kernel unit
- For 6 kernel units

  $750 \cdot 10^{6}\ \mathit{clock\ cycles}\ (cc)$
- Real time execution with a clock of **800MHz**