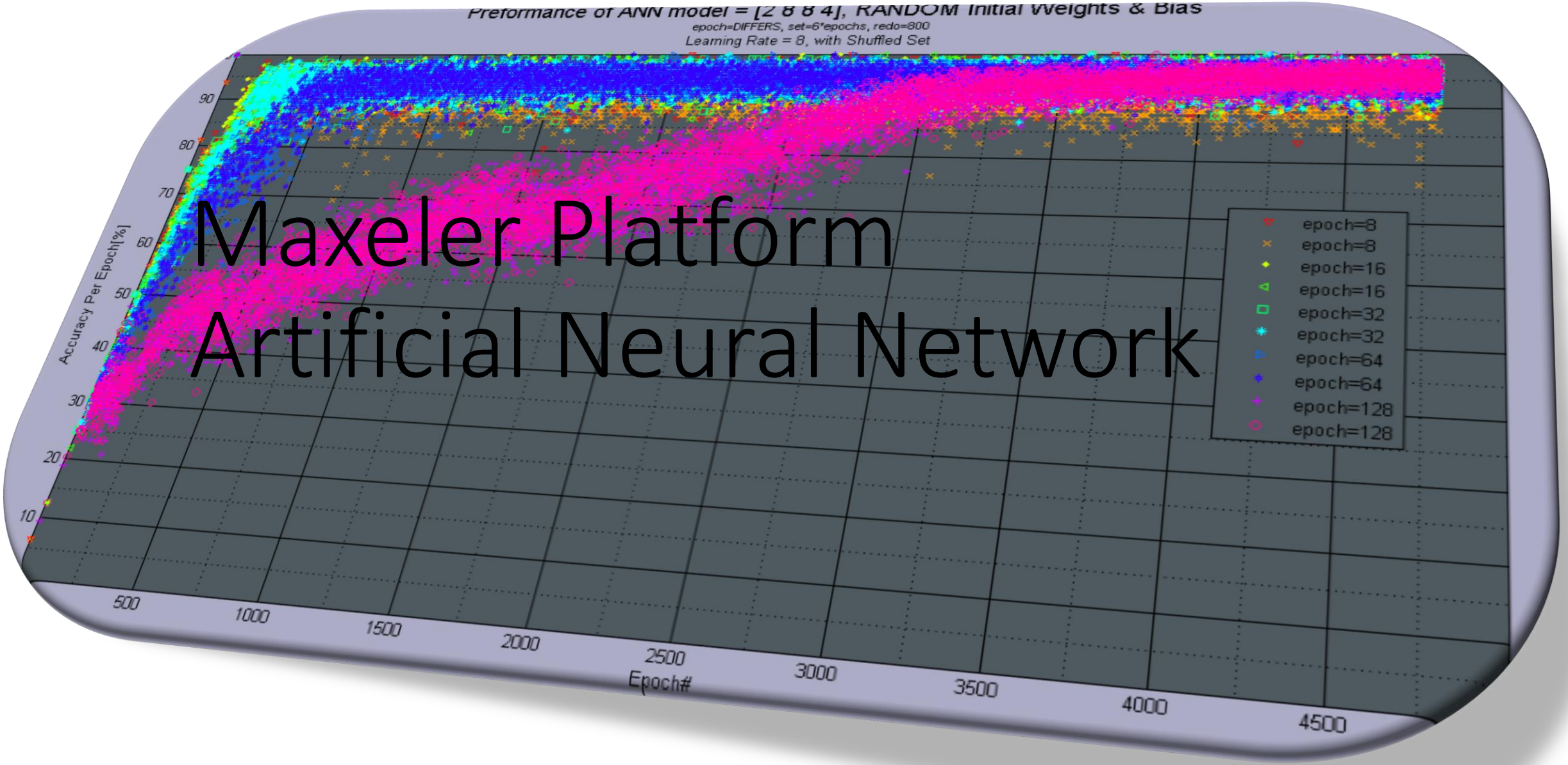


Maxeler Platform
Artificial Neural Network

1. Revision History

main change	by	date
creation	dogev	15/9/2016

2. Presentation Outline

- Motivation
- Goal
- Design Overview
- Problems Encountered
- Hardware Architecture
- Software Architecture
- Results
- Degrees of Freedom
- What's Next
- FAQs
- Questions

Motivation: Optimal Implementation of ANN

- ANN are becoming more and more popular in un-linear solution approximation
- New Available Massive Parallelism Technology
- Reaching far faster training time via Parallel Hardware… like a human brain
- Dynamically Self Organizing Optimal Topology
- Cascading Multiple ANN

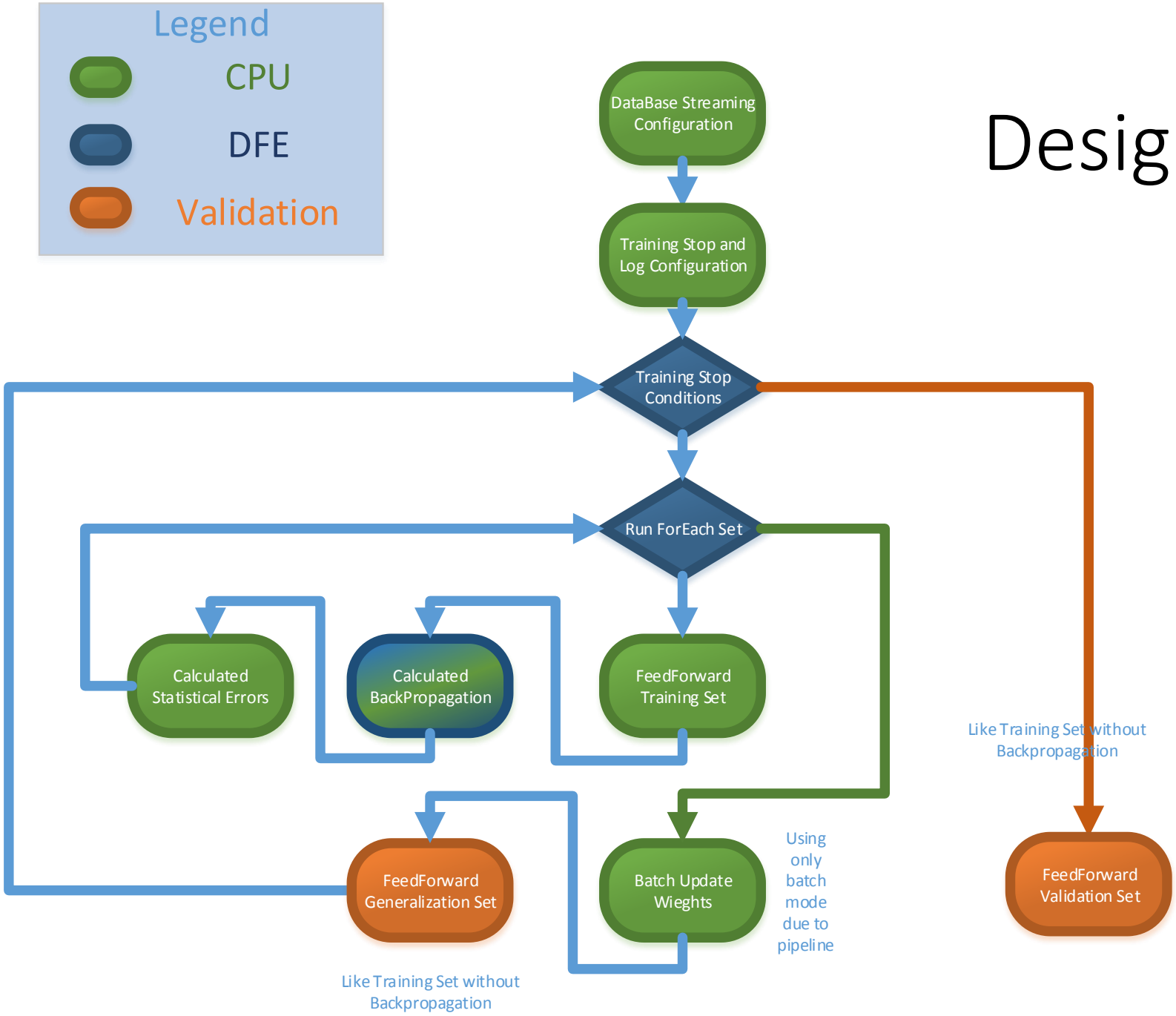
Goal

- Parallel Data Flow Engines ANN in Hardware
- Creating a Generic Building Block for AI architectures
  - Supporting Pre\Post data analysis or feature extractions
  - different activation functions
- Generic AI API
  - Loading and Exporting AI parameters for different tasks
  - Controlling Database partition and Epoch sizes
- CPU Controller oversees training and outputting
  - C++ Host Code calls the ANN
- Comprehensive Statistical Learning Report

Design Overview: Tasks Division

- CPU
  - API
  - Database management
  - DFE configuration for ANN
  - Batch Update Weights
  - Calculated Statistical Errors
- Maxeler DFE
  - Feedforward Parallel Data Stream Engine
  - Backpropagation

Design Overview: Top

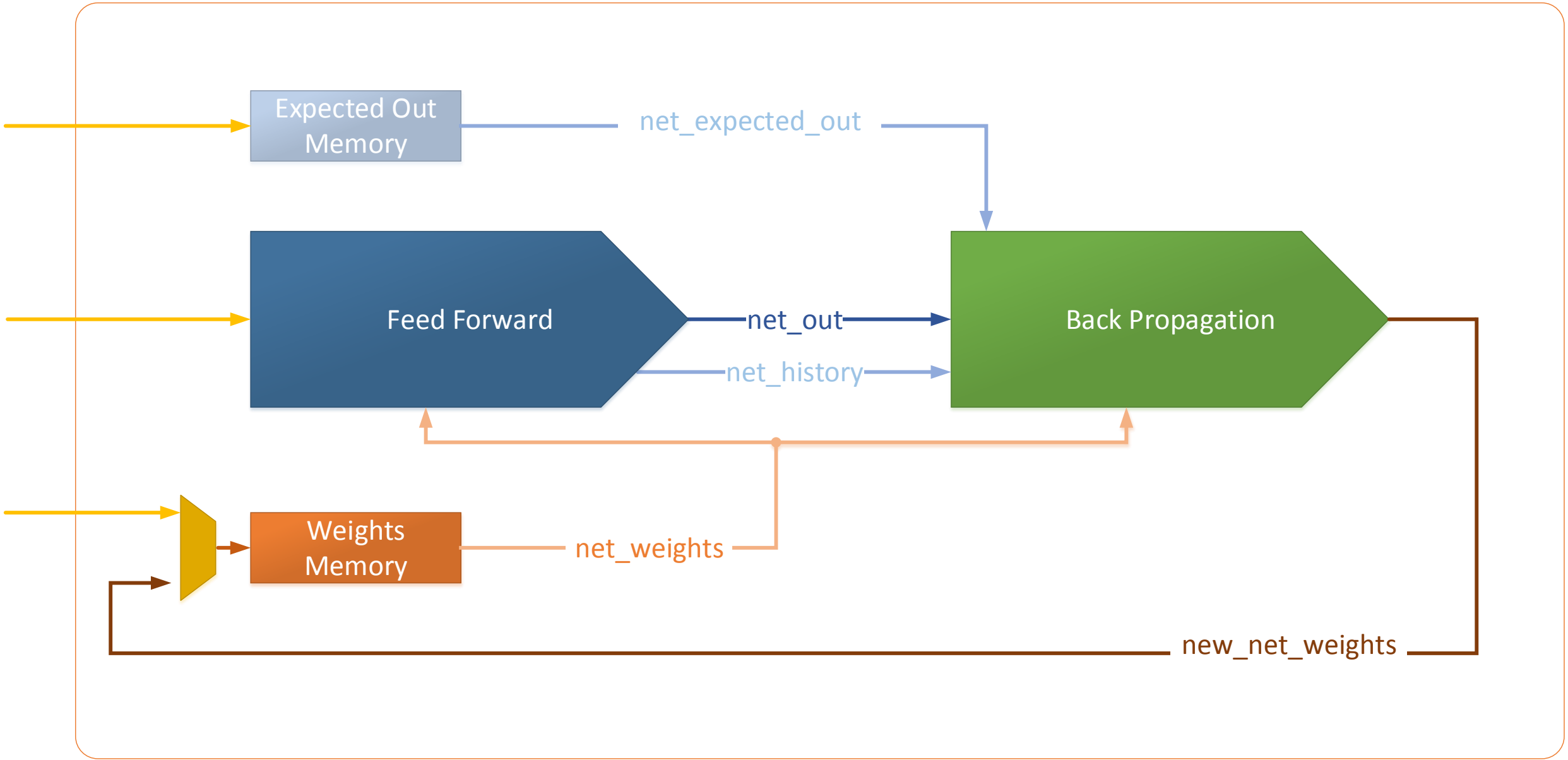


Problems Encountered

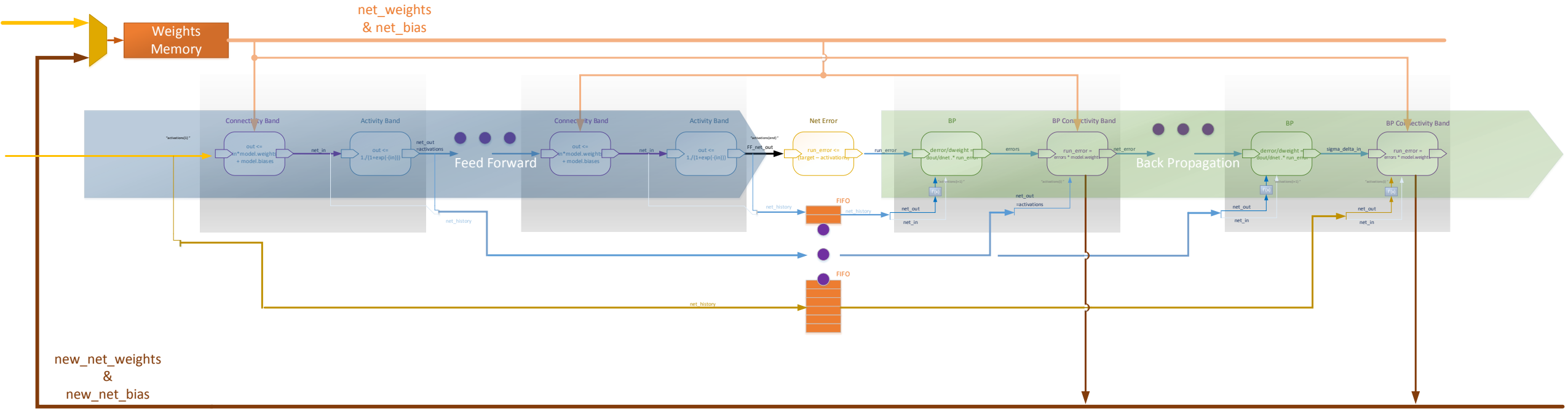
- Lacking Theoretical background
- Unquantifiable Maxeler Hardware bit sizes
- Avoiding data dependency loops in a stream DFE
- Optimal Runtime Back Propagation while streaming new data
  - Mathematical analysis and theorem for new BP
  - Pipeline and Hardware optimal utilization
- Unknown Syntax issues
- No Google
- Kernel.Math(exp) float mismatch in DFE vs Sim

Hardware Architecture

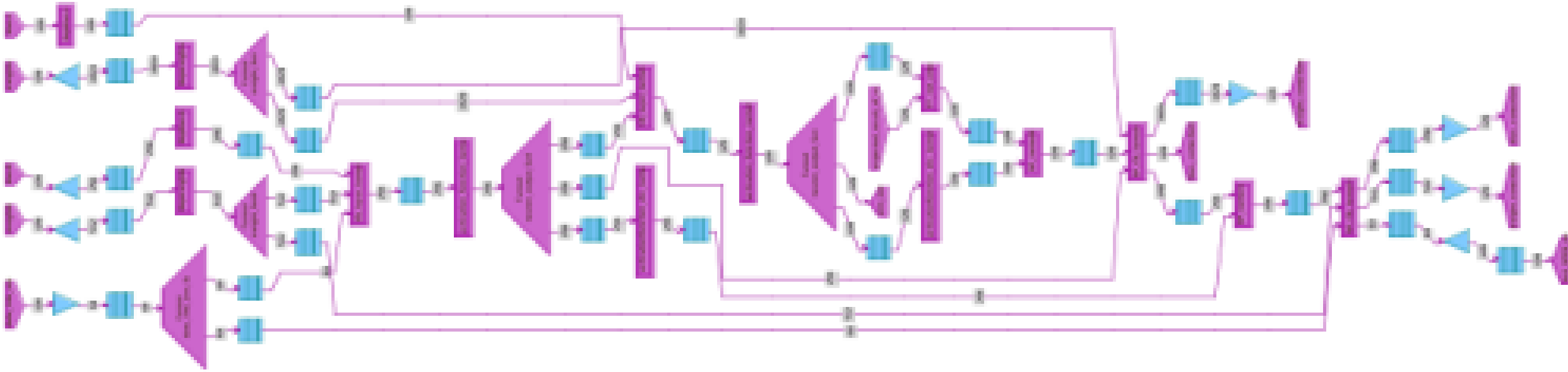
Top Flow (PIPE BASED ON REGS)



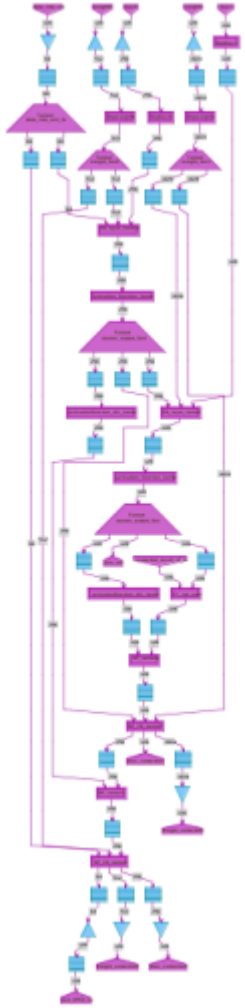
Hardware Architecture: inner

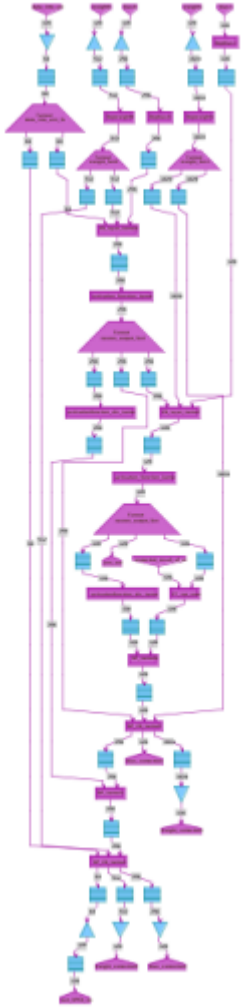


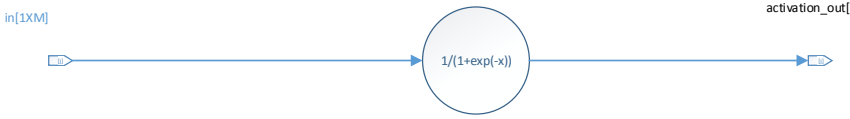
Hardware Architecture: Manager

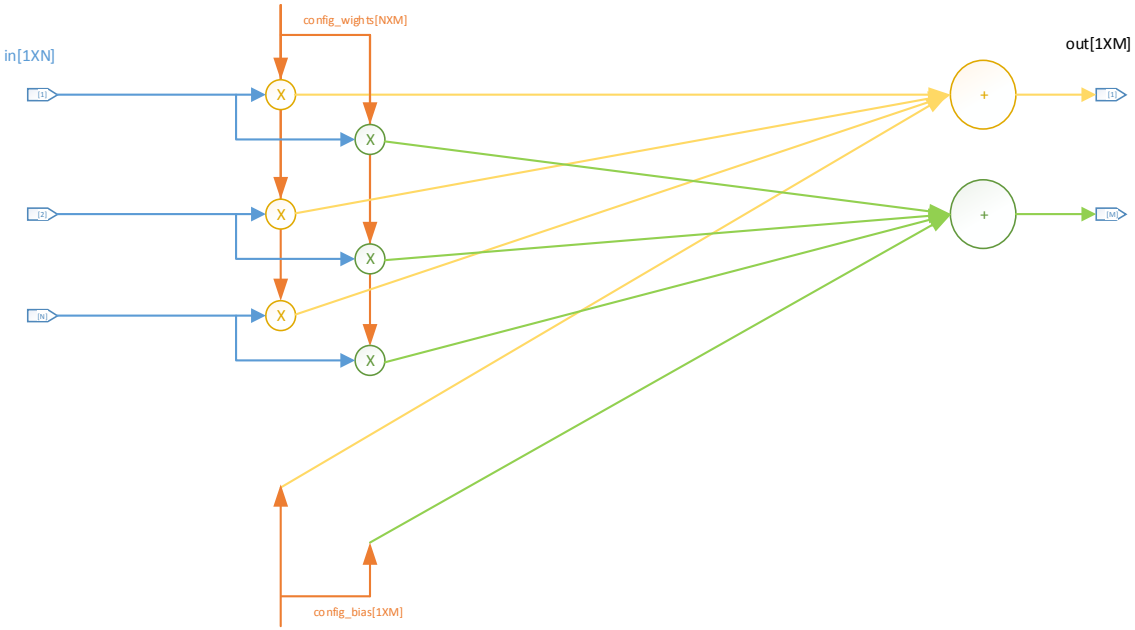


Hardware Architecture: FF Kernels









out[j] = 1./(1+exp(-(in[j])))

AKA in matrix operations:

out[1] out[M] = 1./(1+exp(-(in[1] in[M])))

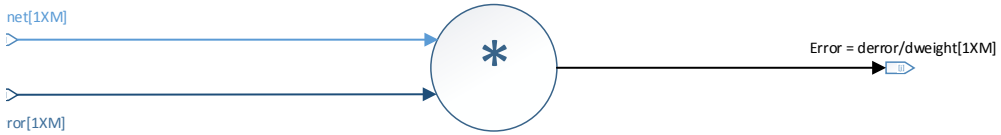
out[j] = ((in[1]*w[1,j])+...+(in[N]*w[N,j])) + bias[j]

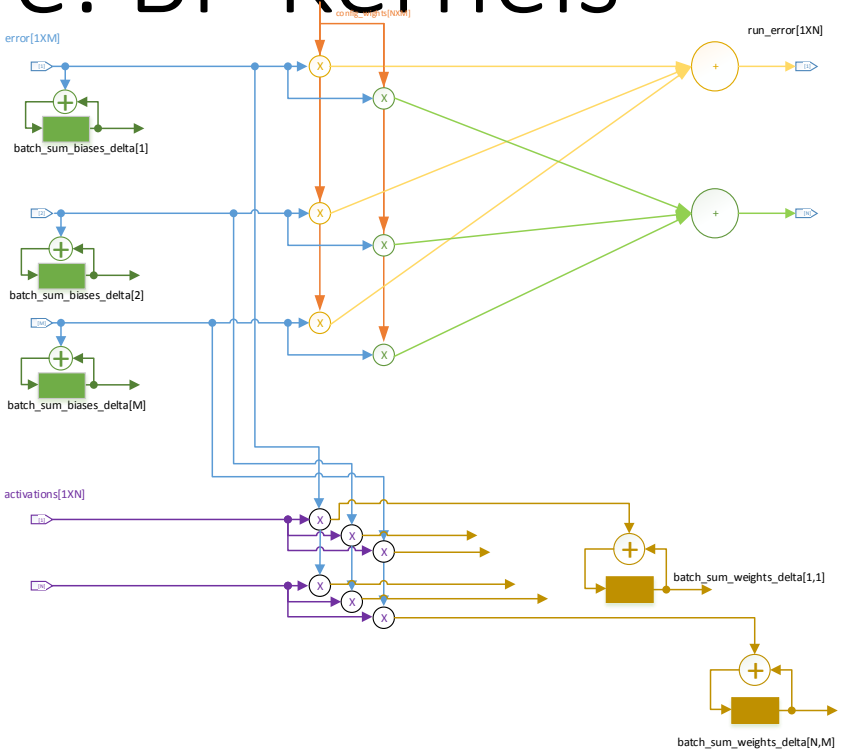
AKA in matrix operations:

out = in * model.weights + model.bias

out[1] out[M] = in[1] in[N] * w[1,1] w[1,M] w[N,1] w[N,M] + bias[1] bias[M]

Hardware Architecture: BP Kernels





out[j] = errors[j] = derror/dweight = dout/dnet .* run_error
Per epoch: batch_sum_biases_delta[j] += errors[j]

AKA in matrix operations:

out[1]	out[M]	=	f'[1]	f'[M]	.*	run error [1]	run error [M]

run_error[j] = ((errors[1]*w[j,1])+...+(errors[M]*w[j,N]))

AKA in matrix operations:

run_ error [1]		run_ error [N]	=	error [1]	error [M]	*	w[1,1]		w[N,1]
							w[1,M]		w[N,M]

run_error = errors * model.weights'

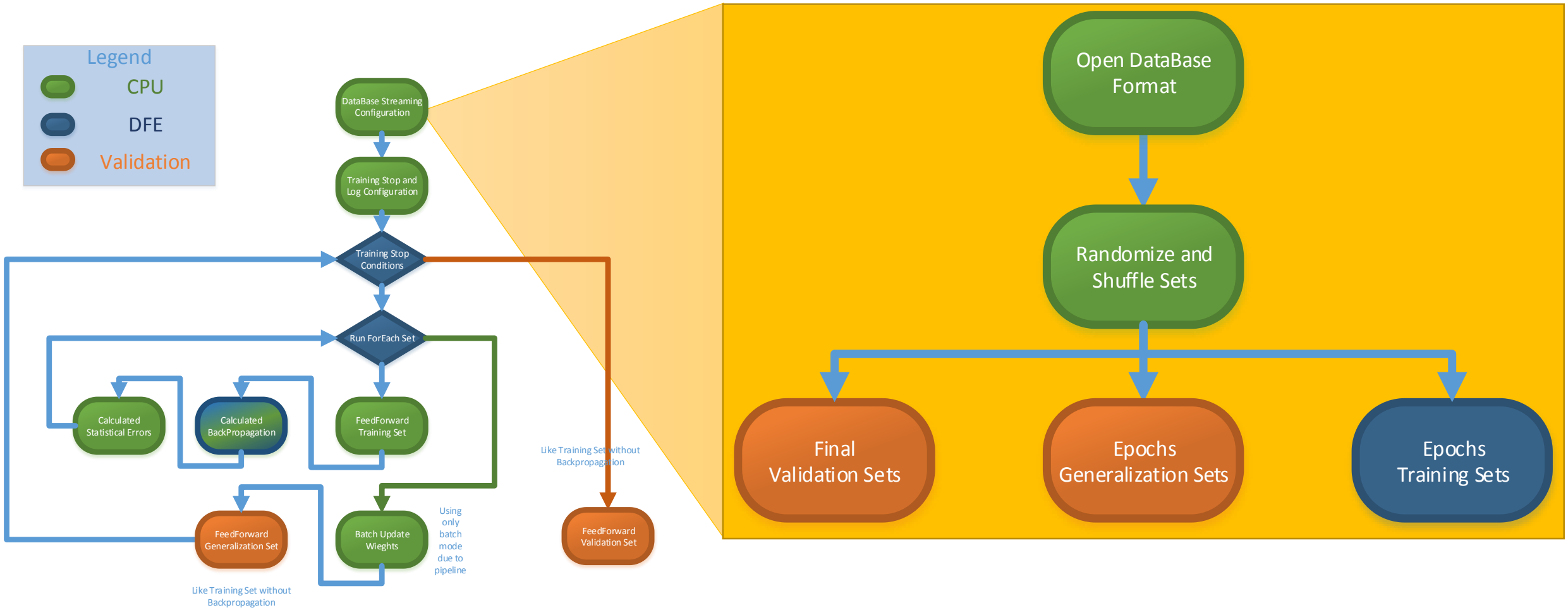
weights_delta[i,j] = (activations[i] * errors[j])

AKA in matrix operations:

w[1,1]	w[1,M]	=	activati ons [1]	*	error [1]	error [M]
w[N,1]	w[N,M]		activati ons [N]			

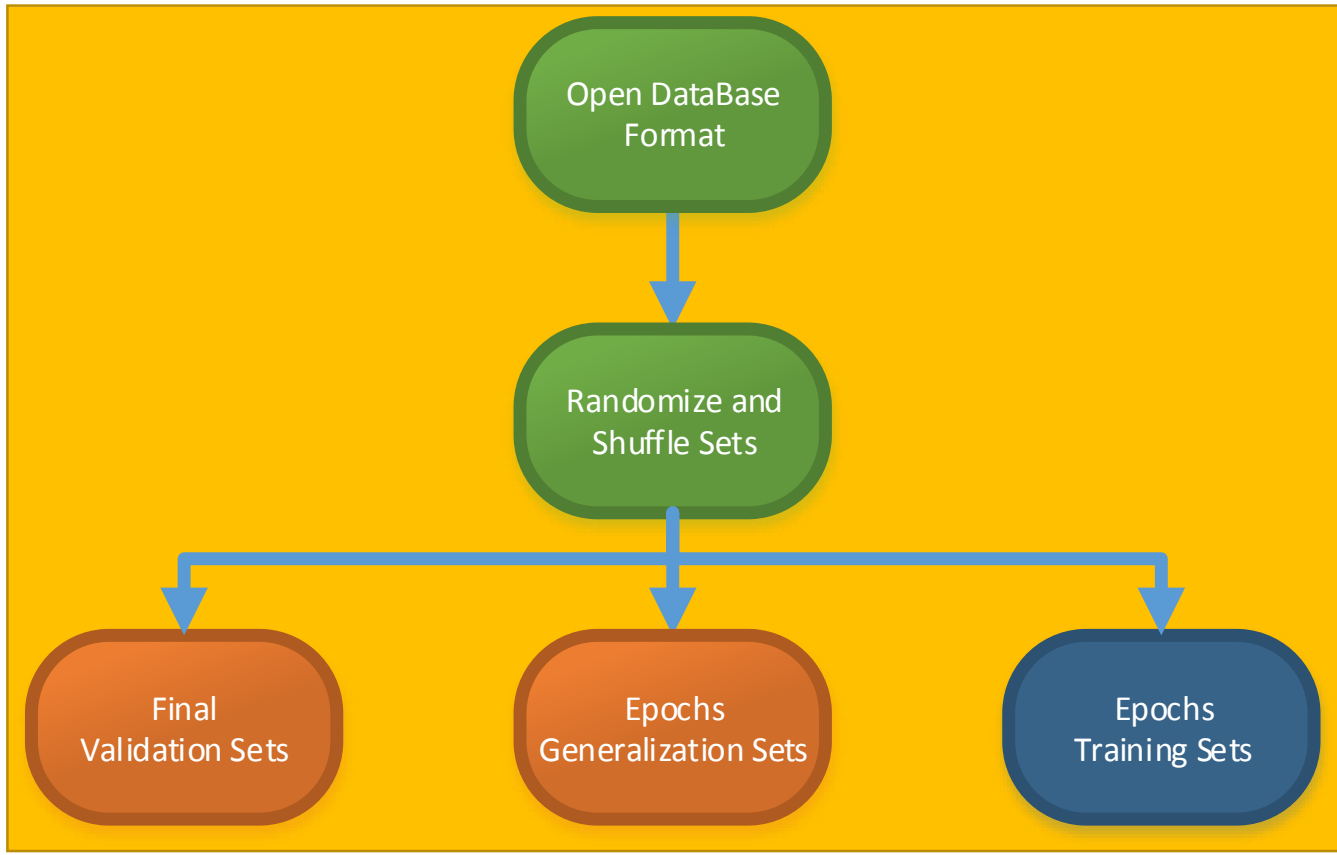
weights_delta = activations' * errors

Software Architecture: Input Database

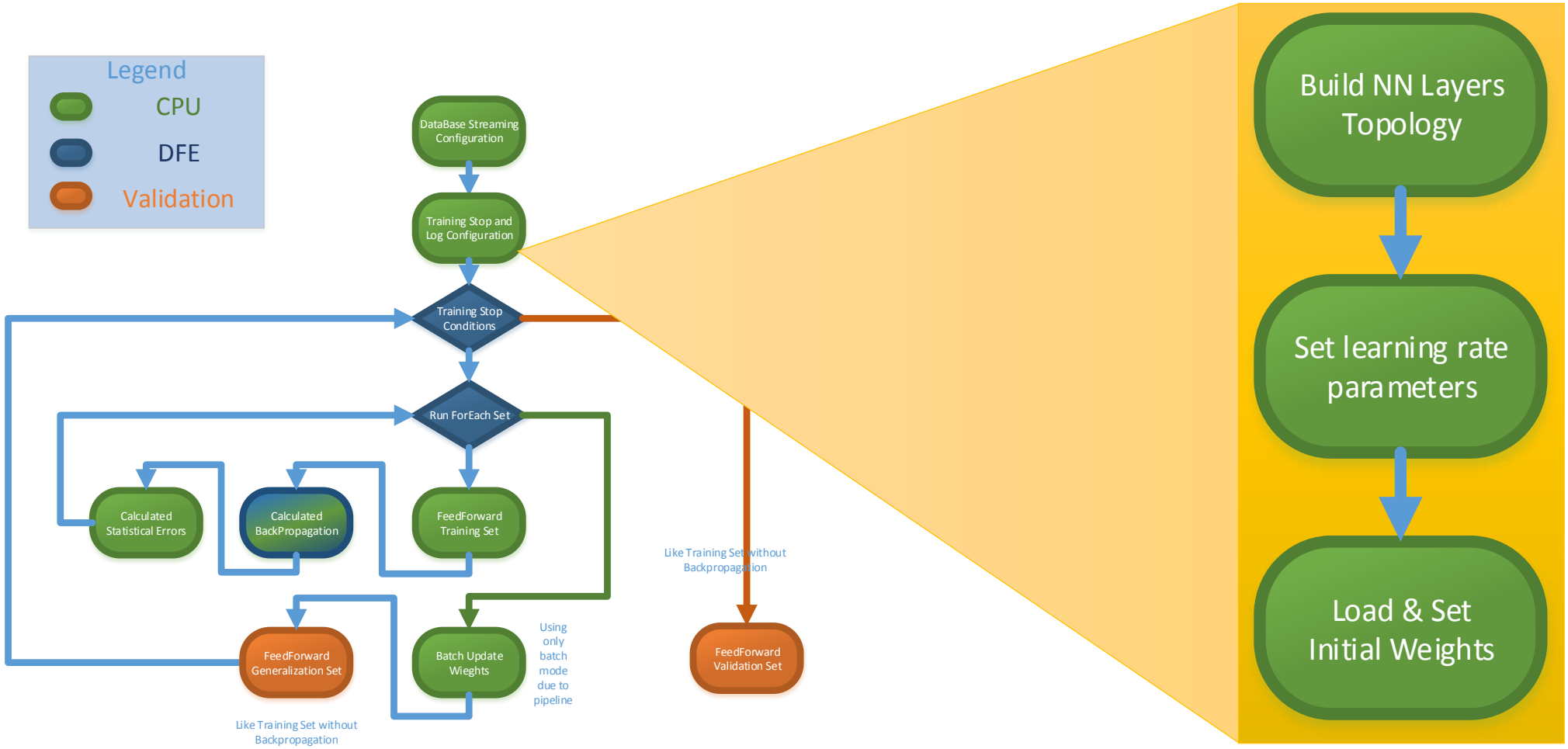


Software Architecture: Input Database

- Open csv database
- Randomly shuffle the database
- Split database per epoch to:
  - Training – learning set
  - Generalization – statistical analysis
- Validation – final performance test

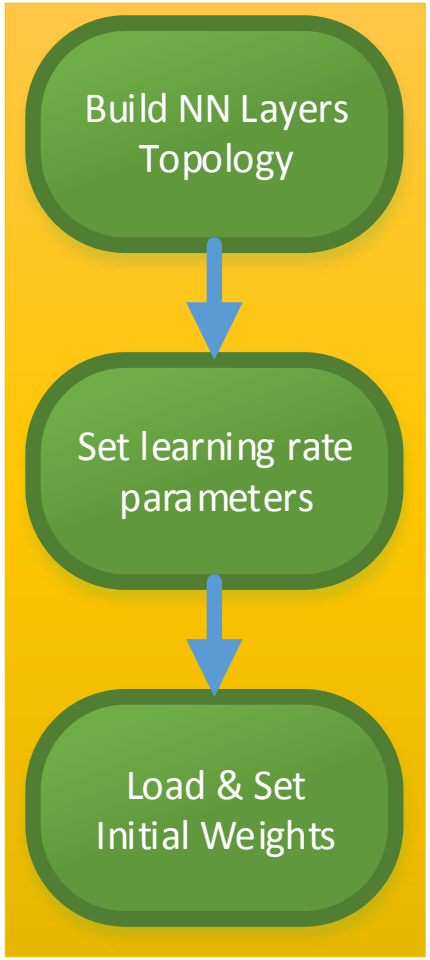


Software Architecture: Input Algorithm Params

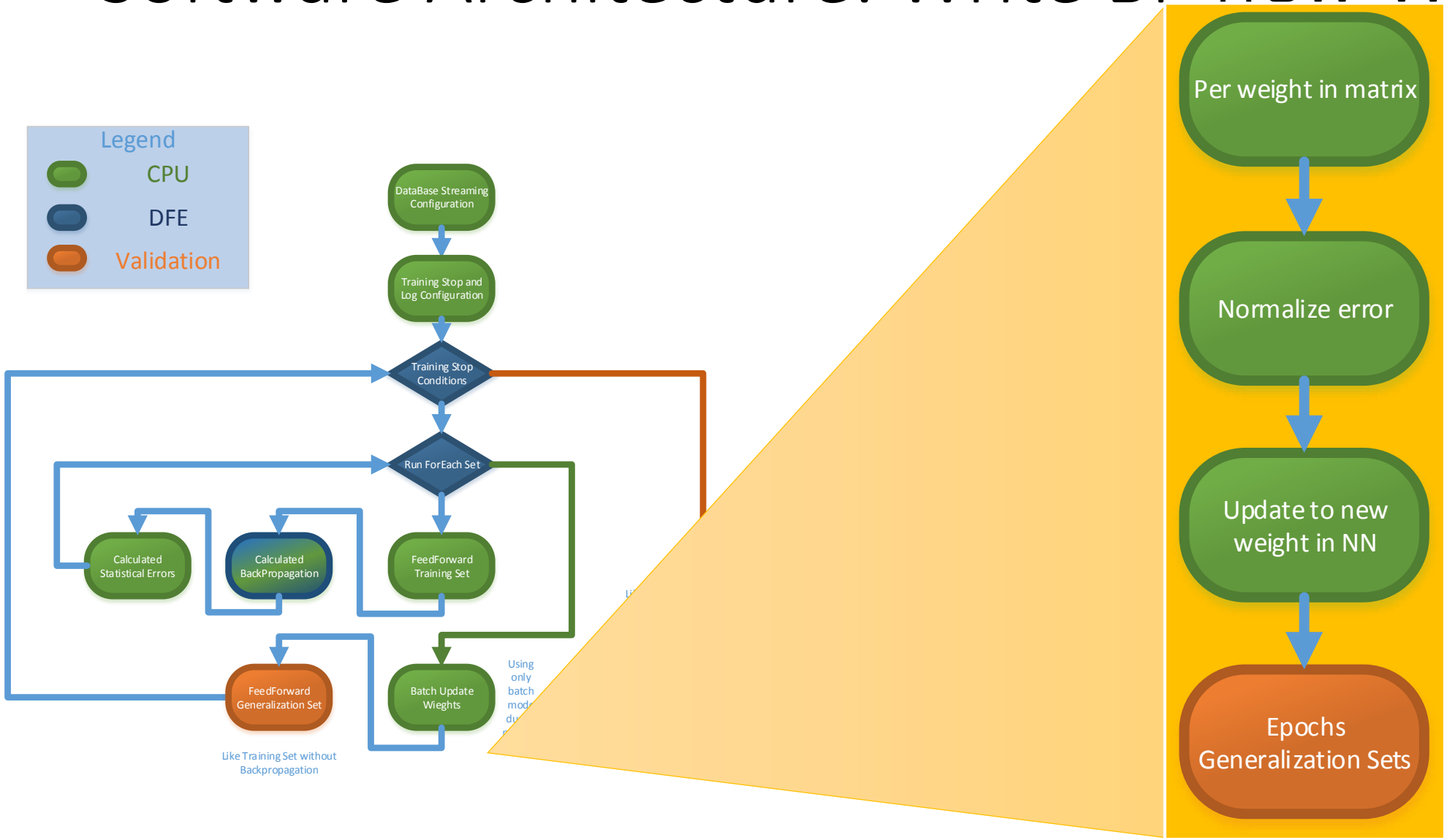


Software Architecture: Input Algorithm Params

- Calling the DFEs with the desired working an learning parameters
  - Number of Input \ Output nodes
  - number of hidden layers
  - Float size
  - Learning rate α
- Loading initial weights if present from csv
  - Else, use random

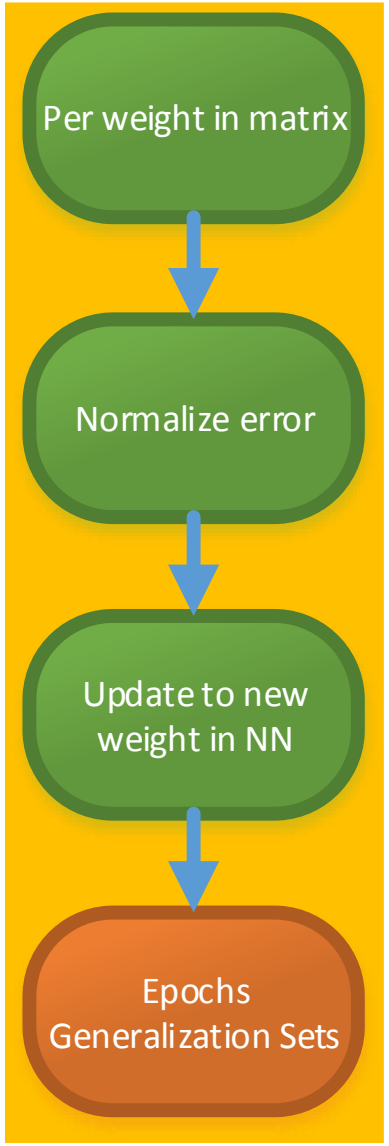


Software Architecture: Write BP new Weights

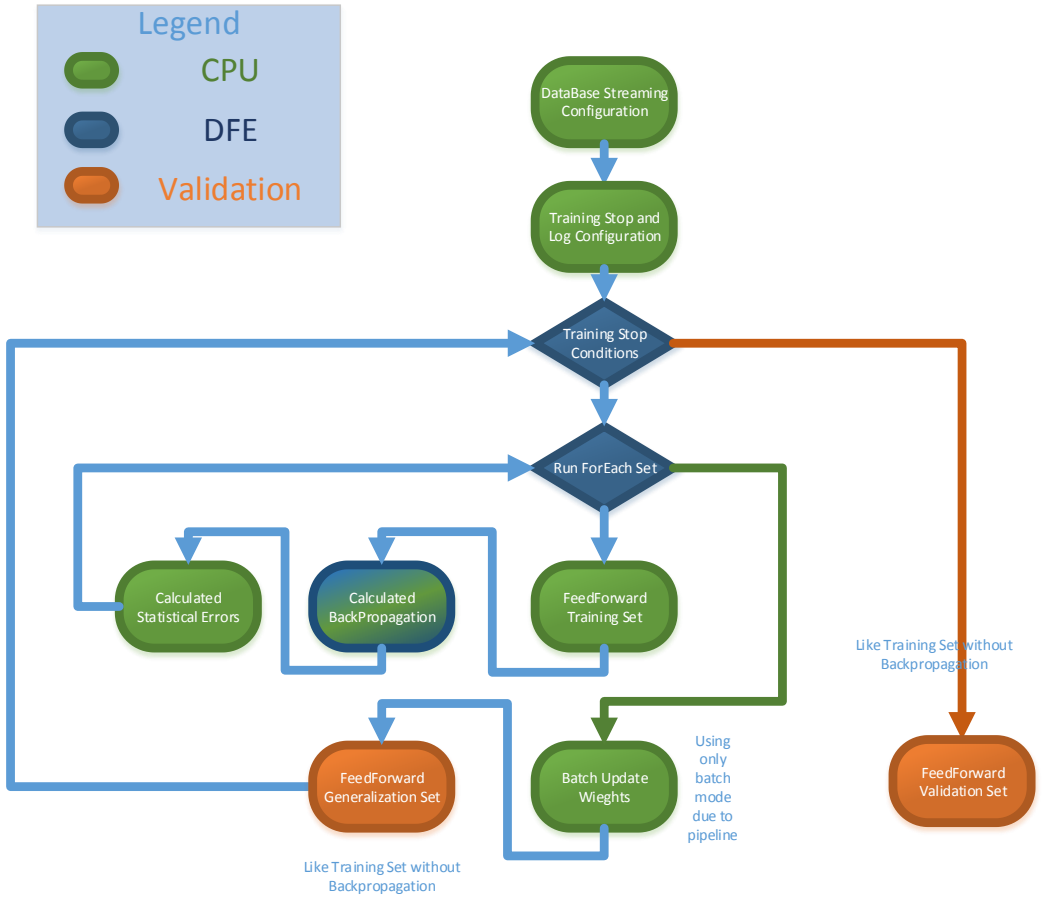


Software Architecture: Write BP new Weights

- Per Batch\Epoch:
- Gets weights corrections from the DFE Back Propagation algorithm
- Normalize the average correction over entire batch
  - In order to make sure each data entry has the same effect over weight calculation
- The memory can only be written by the CPU in our flow-control
- Run a generalization set for statistical analysis

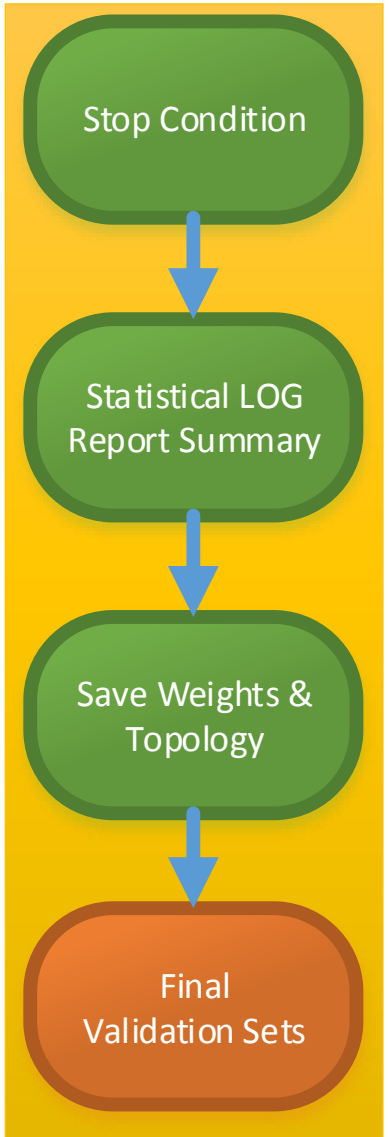


Software Architecture: Save Topology & LOGs

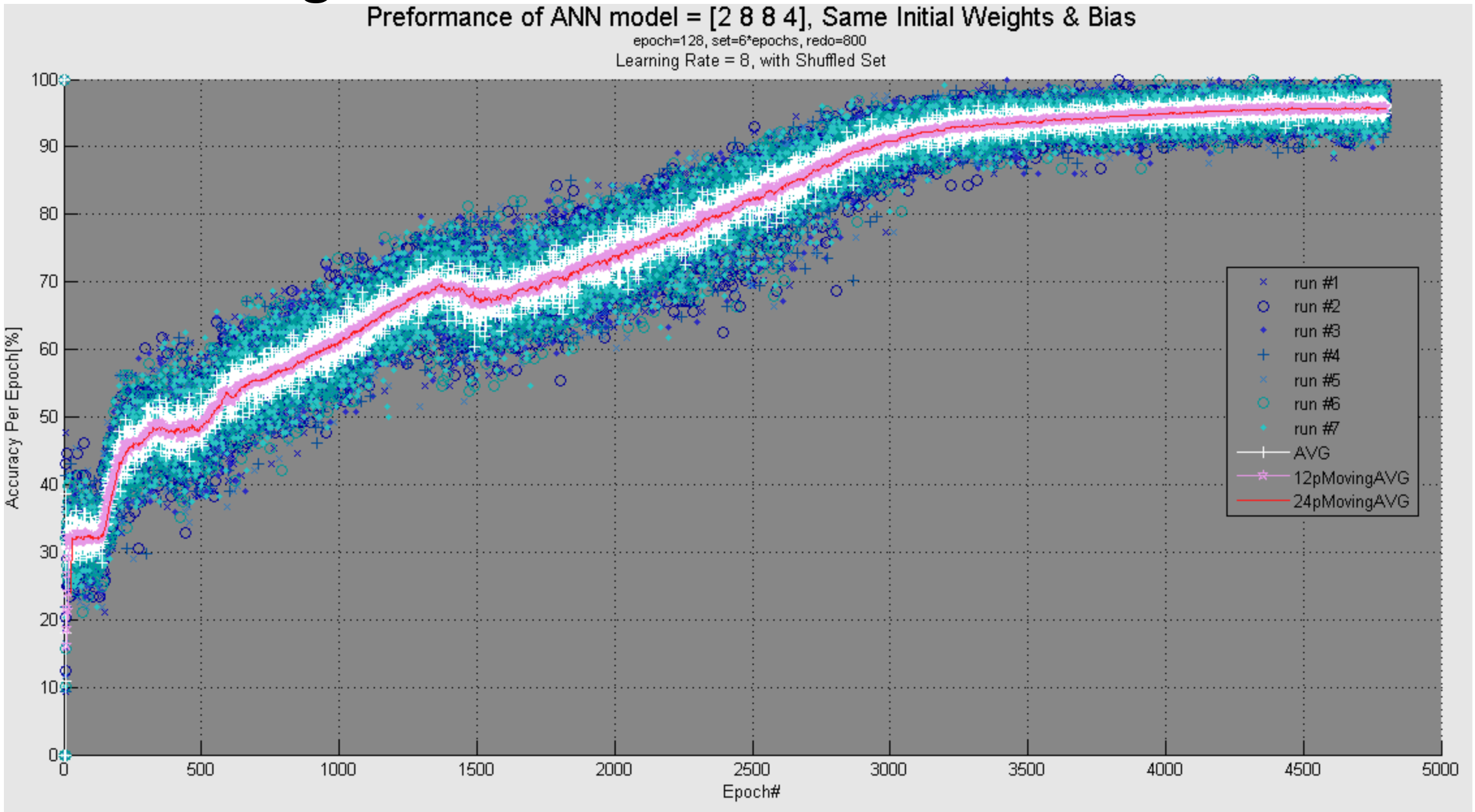


Software Architecture: Save Topology & LOGs

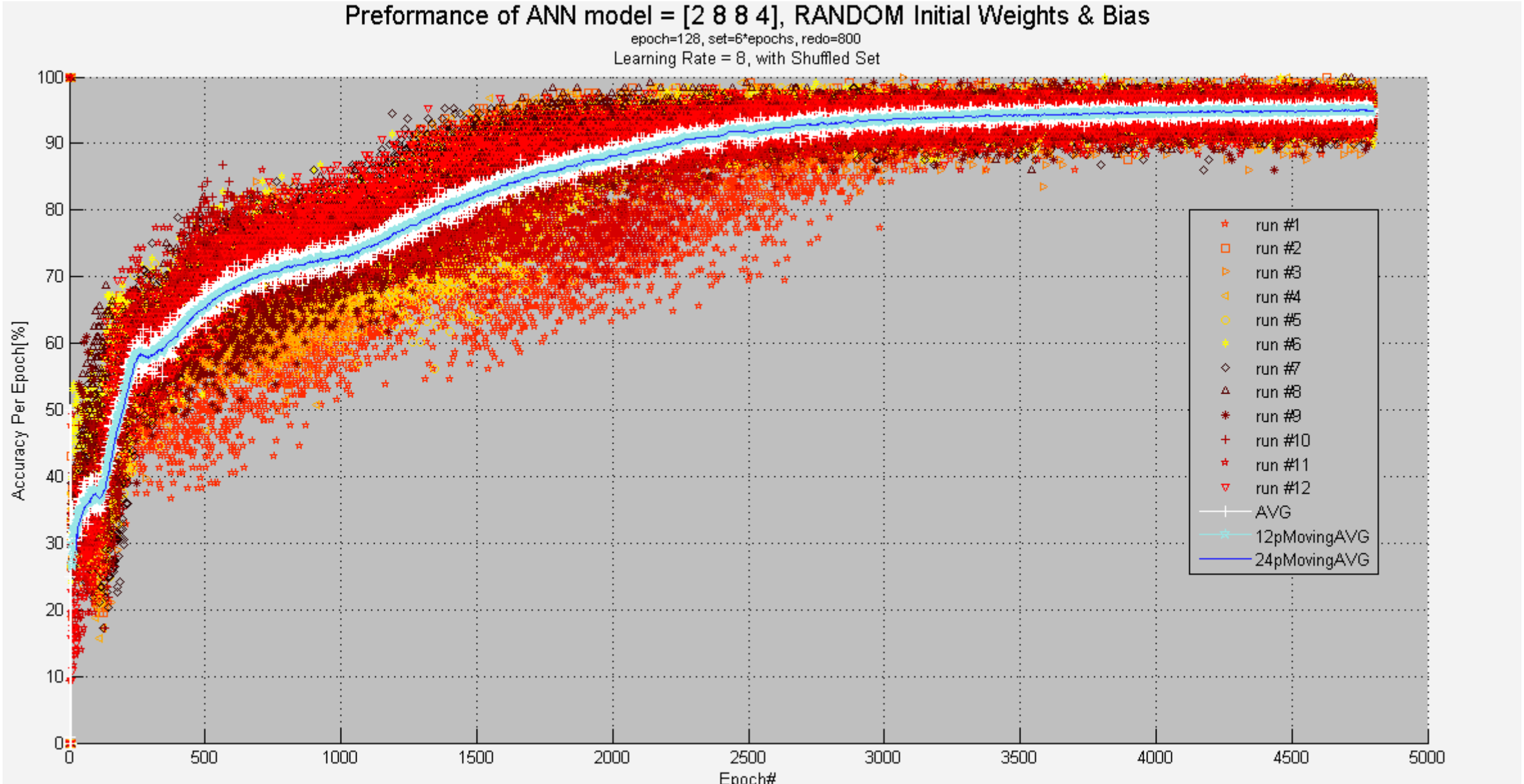
- check (performance or number of iterations)
- After stop condition is met
- Export report and final NN layout
- Run final validation set to estimate NN performance



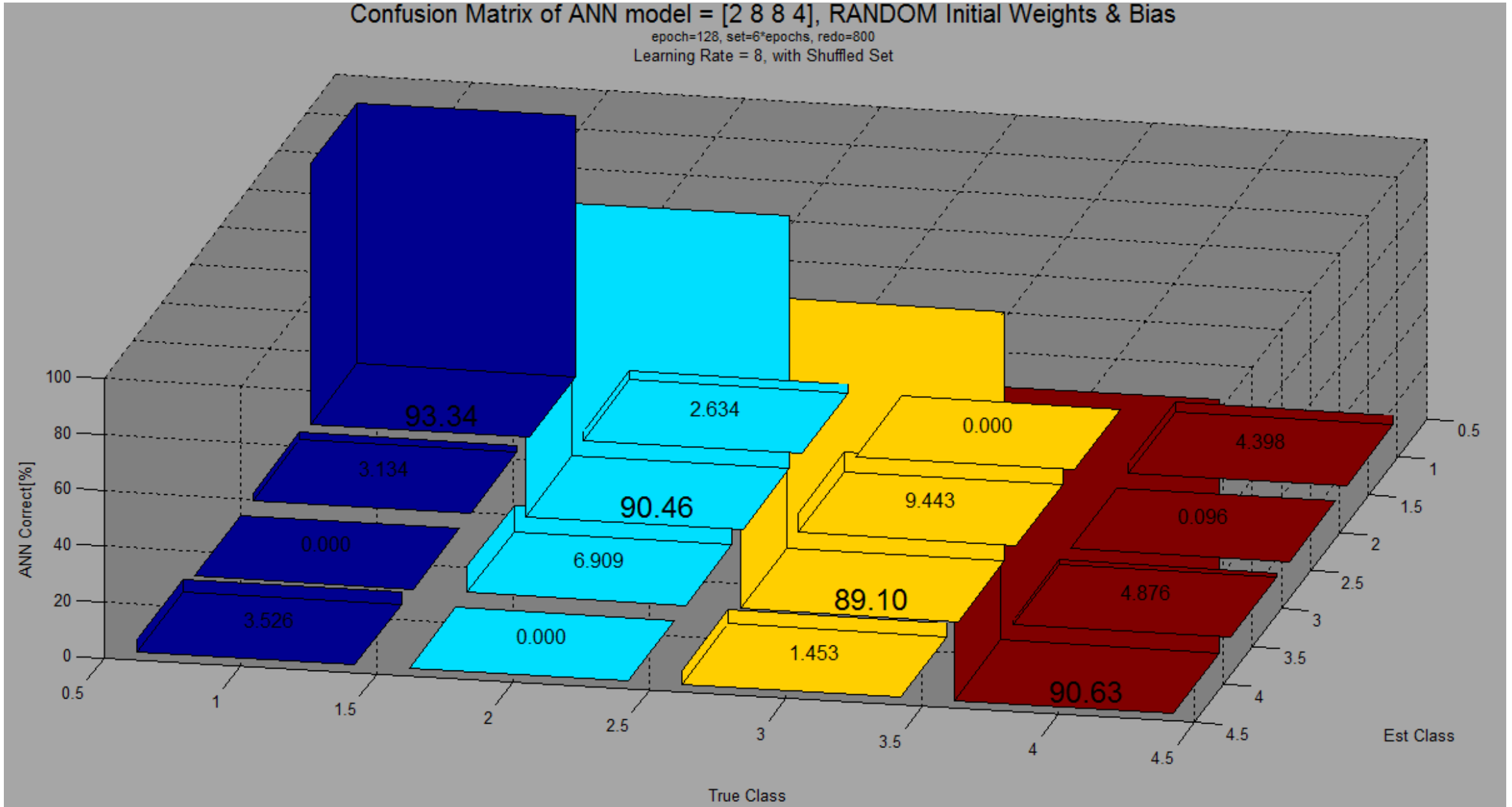
Results Against Golden Modules



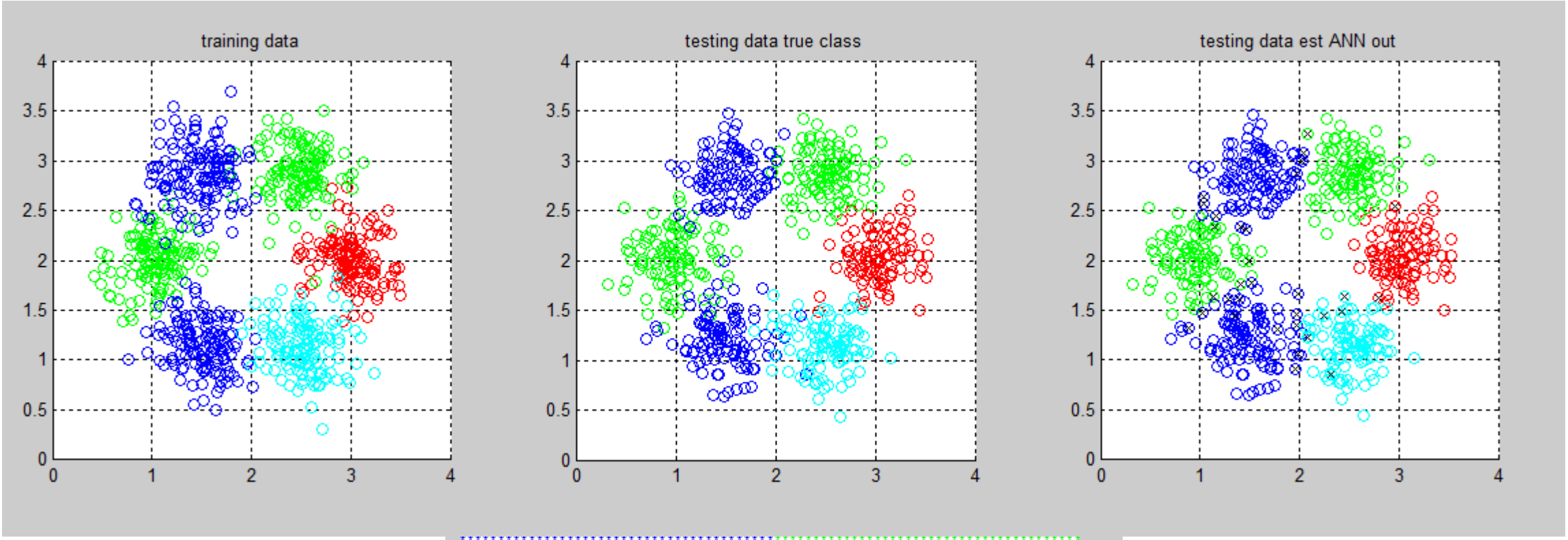
Results Against Golden Modules

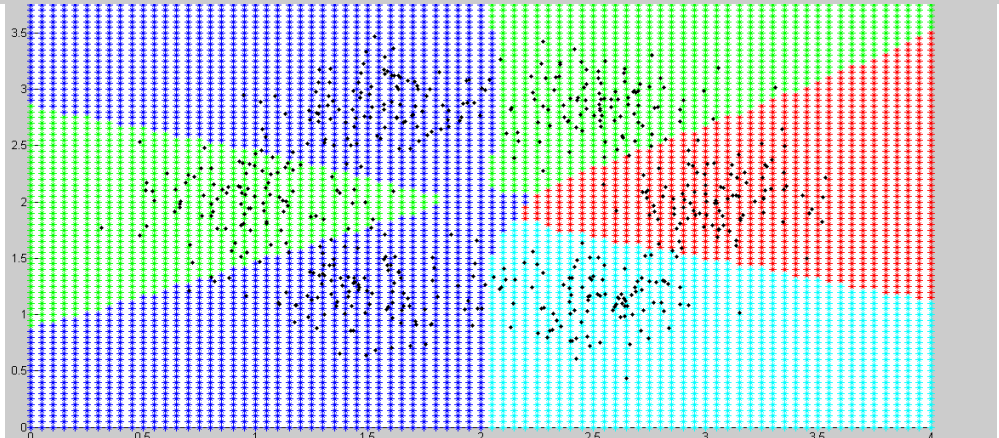


Results Against Golden Modules

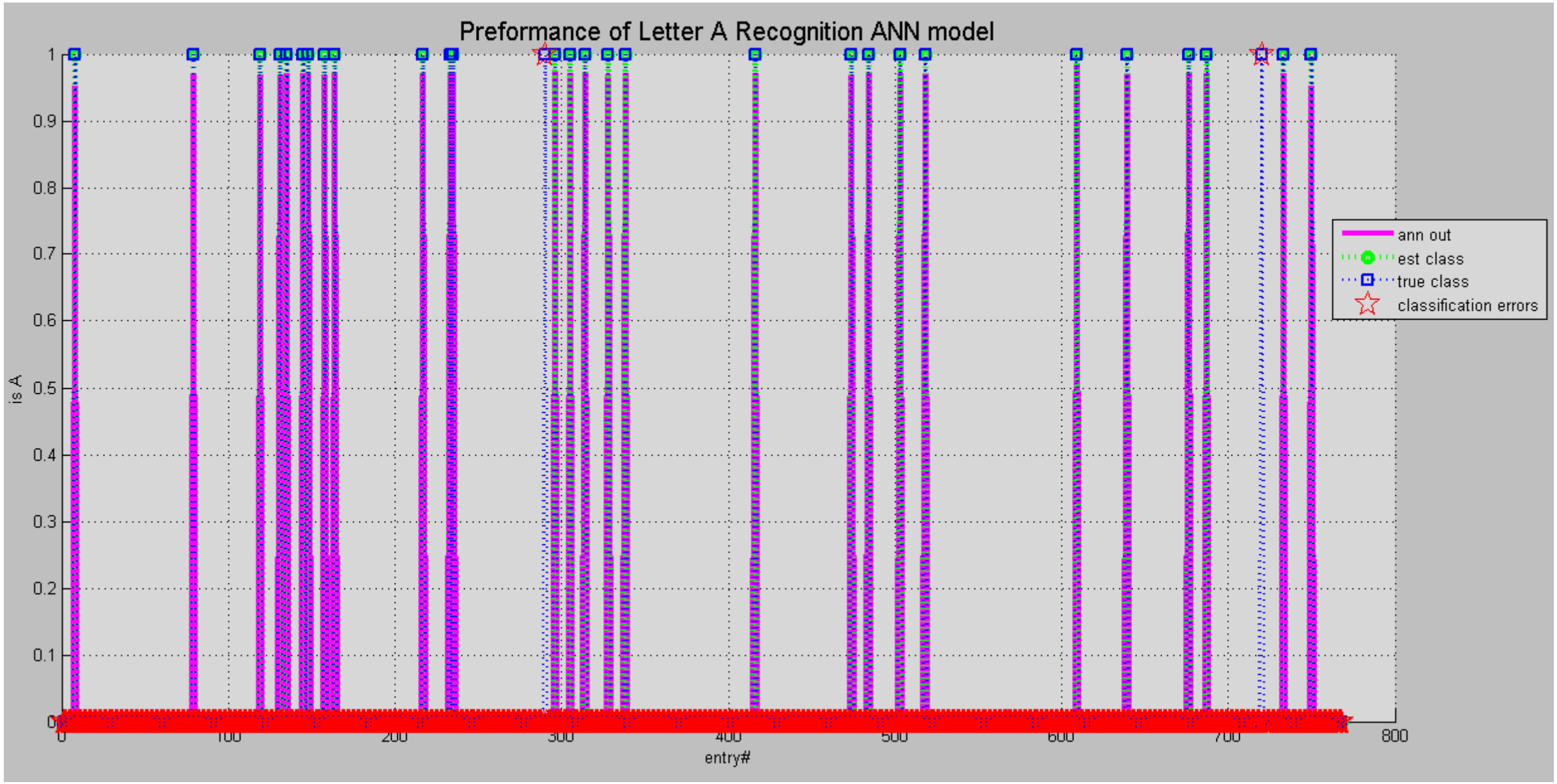


Results Against Golden Modules Classifier





Results MINST Letter A Recognition



Resource Utilization per topology

- ANN = [2 8 4] RESOURCE USAGE
  - Logic utilization: 195599 / 297600 (65.73%)
  - LUTs: 140452 / 297600 (47.19%)
  - Primary FFs: 187653 / 297600 (63.06%)
  - Multipliers (25x18): 444 / 2016 (22.02%)
  - DSP blocks: 444 / 2016 (22.02%)
  - Block memory (BRAM18): 377 / 2128 (17.72%)
- ANN = [2 8 8 4] RESOURCE USAGE
  - Logic utilization: 388378 / 297600 (130.50%)
  - LUTs: 281913 / 297600 (94.73%)
  - Primary FFs: 377632 / 297600 (126.89%)
  - Multipliers (25x18): 932 / 2016 (46.23%)
  - DSP blocks: 932 / 2016 (46.23%)
  - Block memory (BRAM18): 718 / 2128 (33.74%)

Timing Compare

- A fully connected ANN had been made into a working DFE!
- Timing performances vs GPUs is now tested…
- So the final result it yet to be posted

Degrees of Freedom

- Number of layers , inputs and outputs
- Database partition and Epoch size
- Learning rate
- Activation Function
- Initial Weights

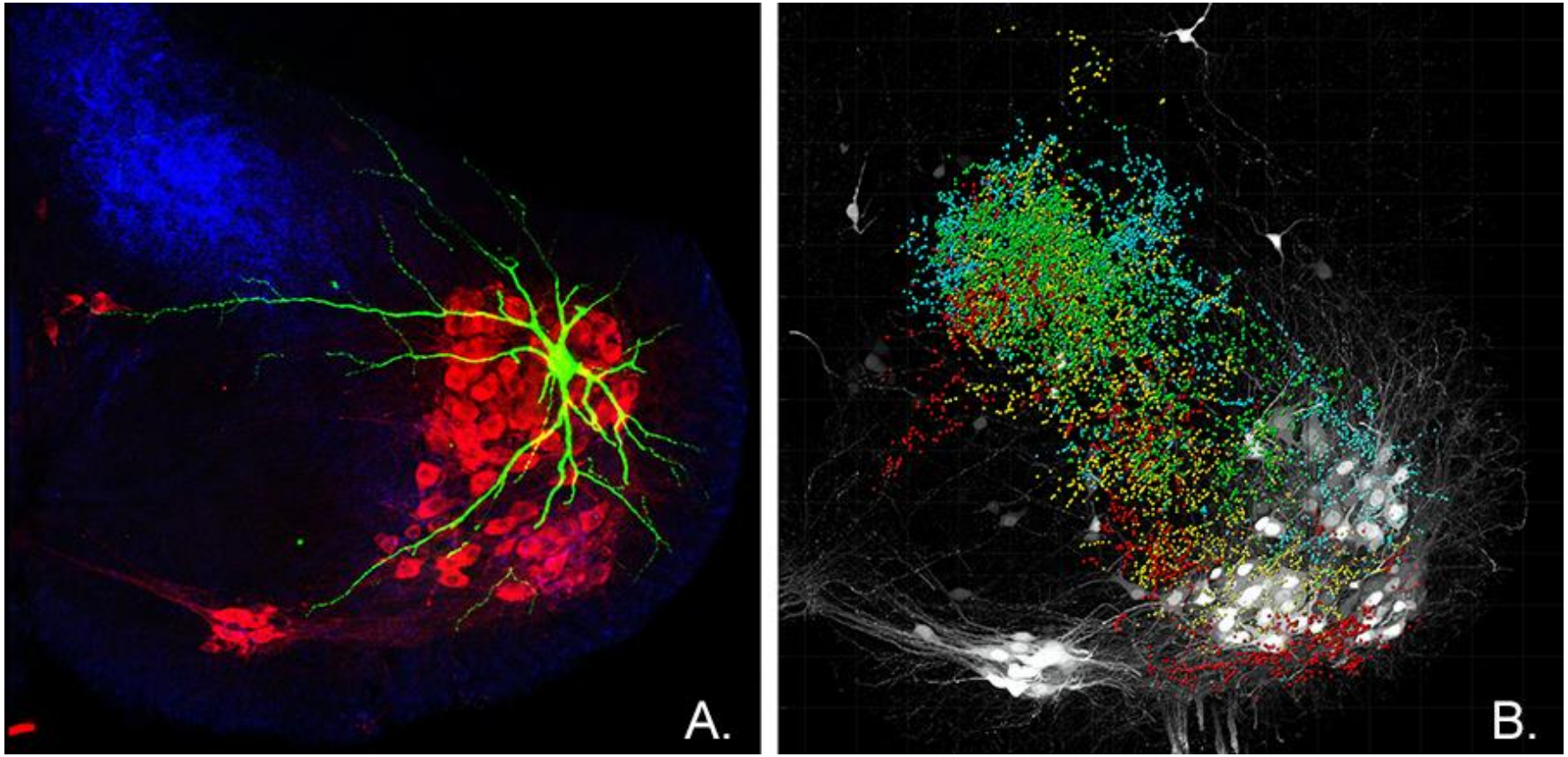
What’s Next

- Push our building block to the wide use APP
- Non Blocking Handler
- S.O.O – Self Organizing Optimization
- MaxRinging a lot of ANNs
- Python Generated envelop

Thank You!!!

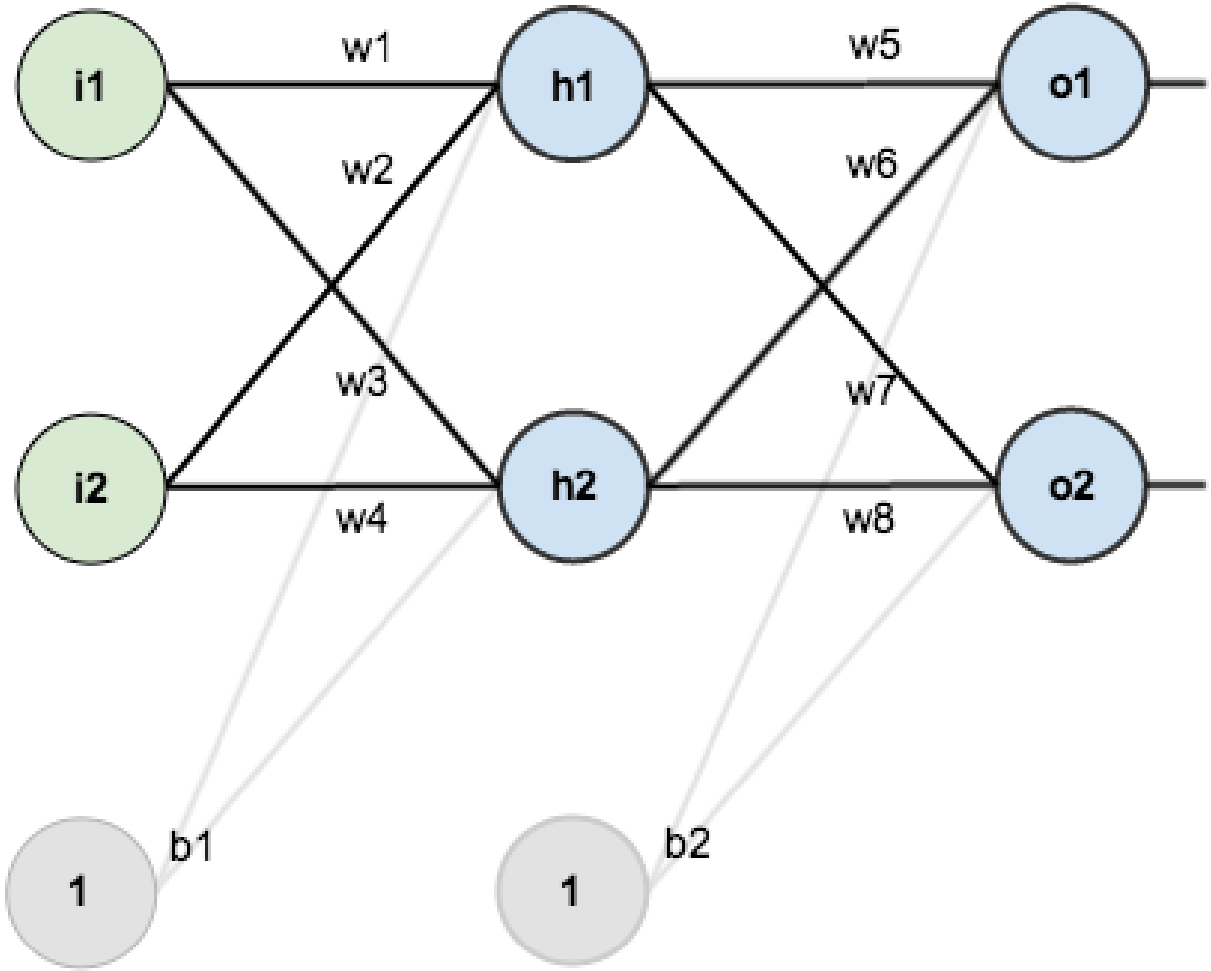
FAQ: What is Artificial Neural Network

- What kind of AI
- Advantages
- Basic structure



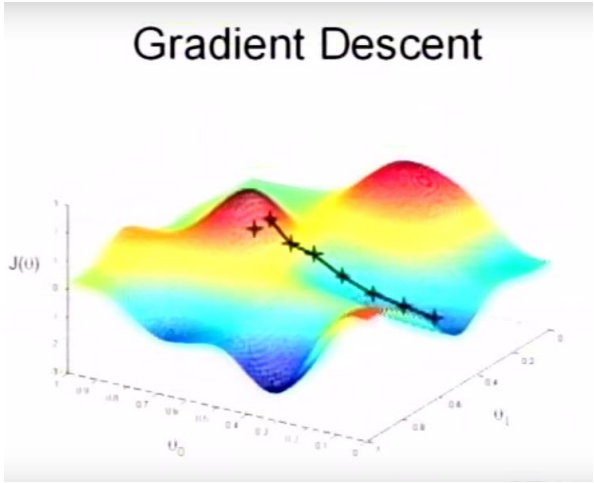
FAQ: Basic Artificial Neural Network

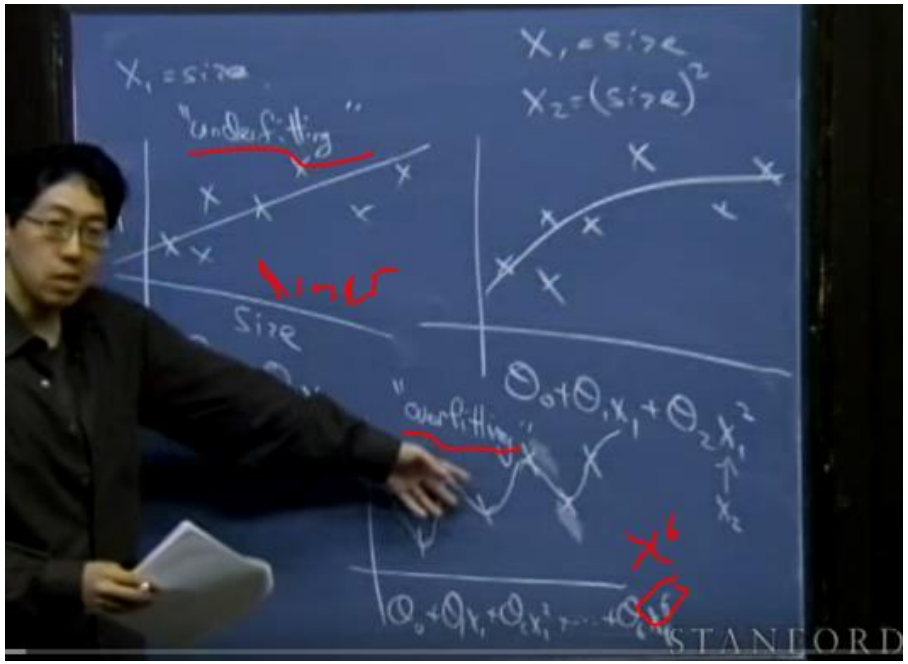
- Topology and connection
- Basic data flow

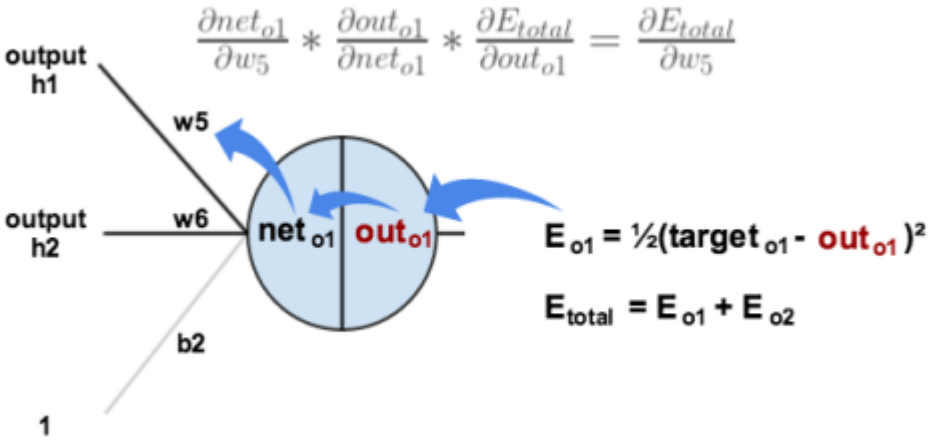


FAQ: Basic Learning Process

- BP
- Gradient Decent

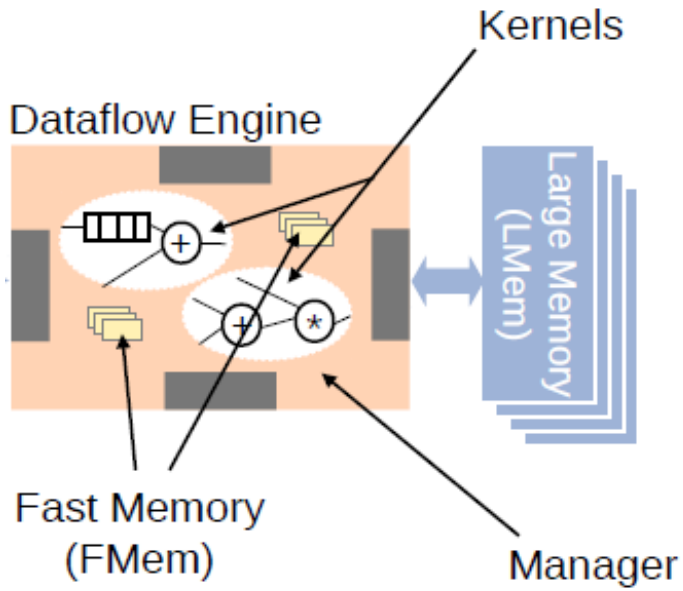


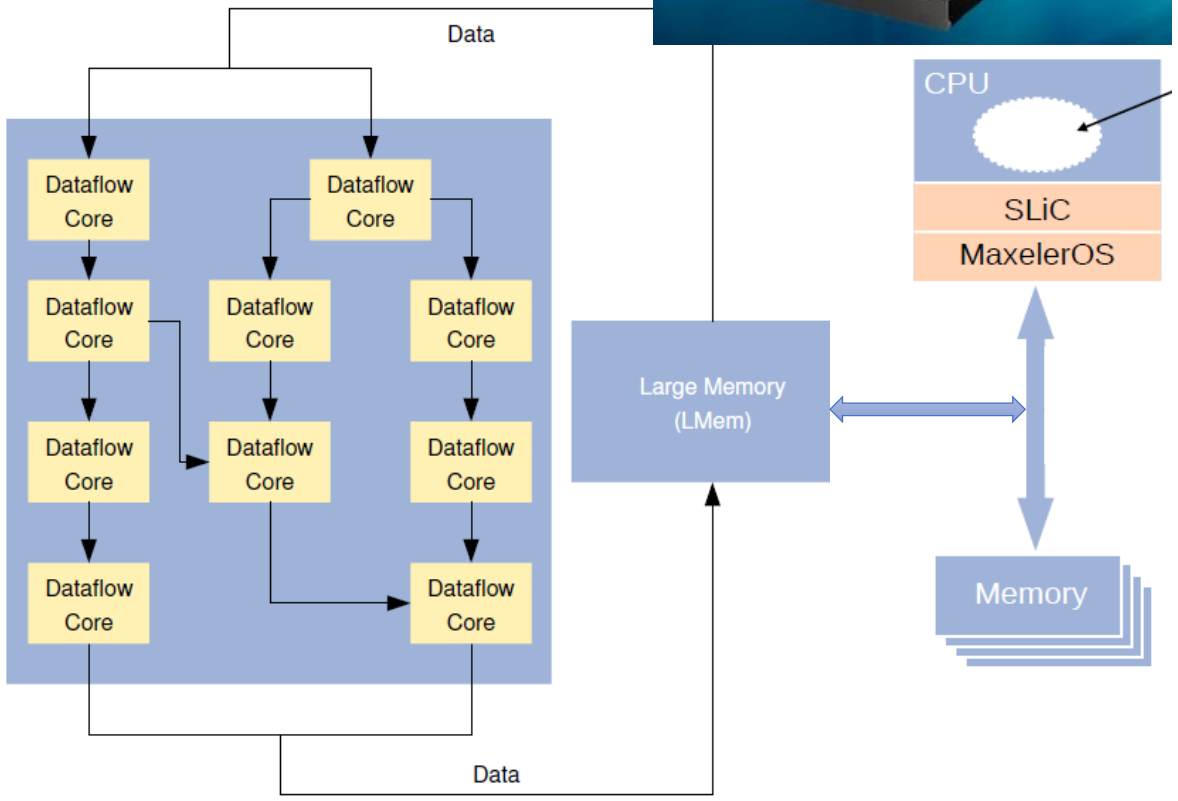




FAQ: What is the Maxeler Platform and DFE

- Data stream flow engines computing
- The different parts of the platform
- Design Flow
- Advantages





Questions