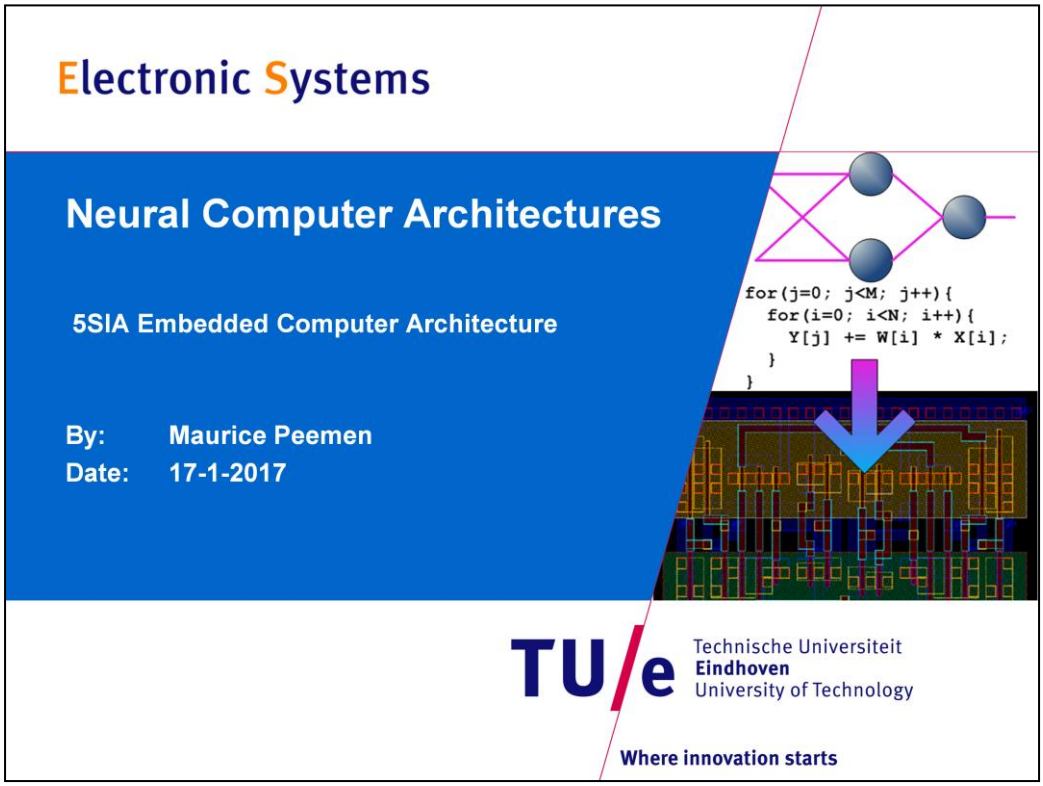
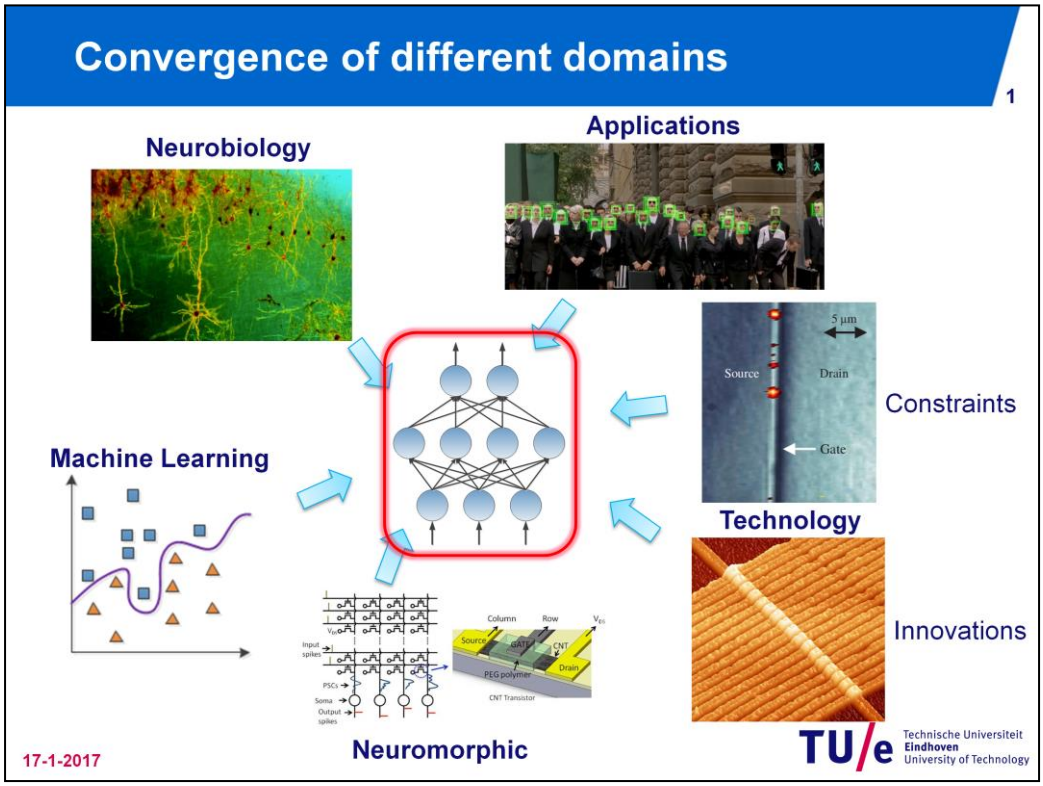
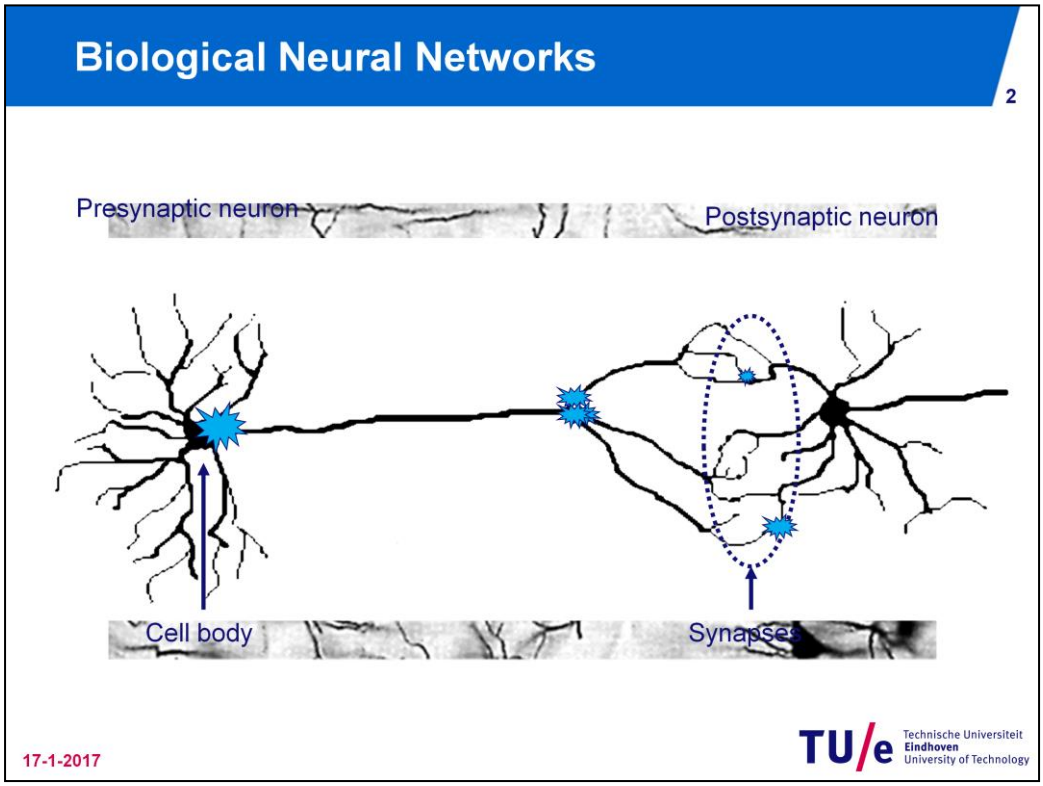



An overview lecture for EE/Embedded Systems students. During your future career as engineer it is useful to know about neural architectures. Neural architectures used in many domains that relate to computation.



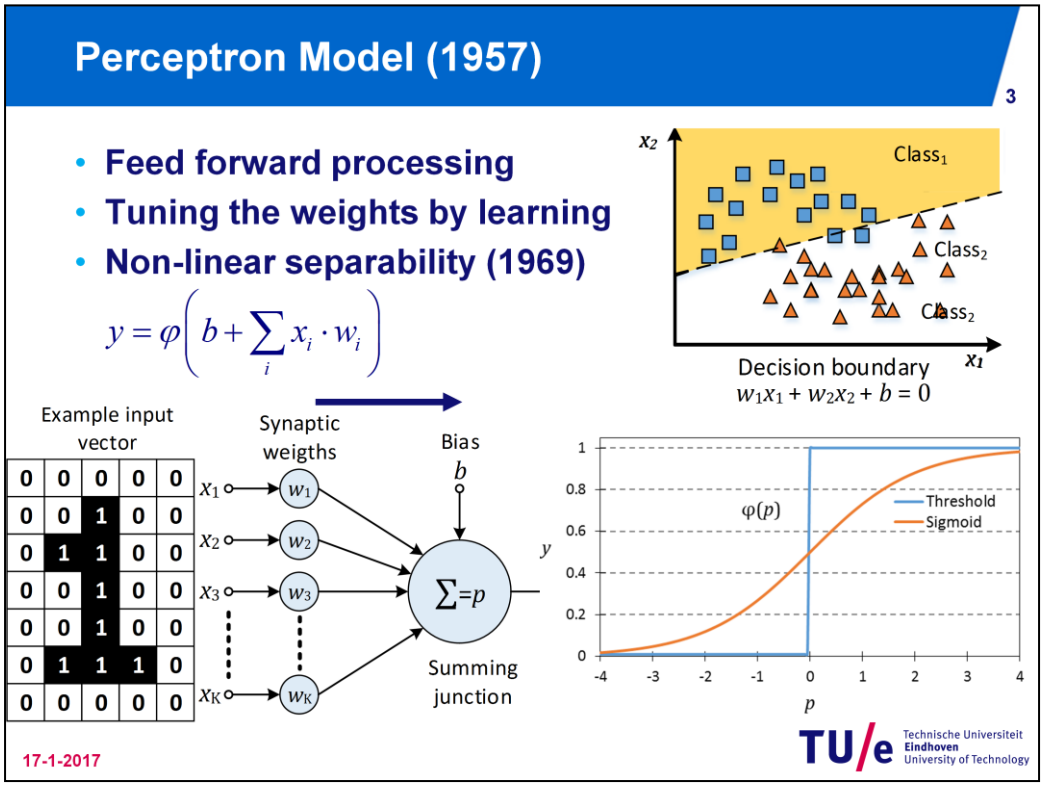

Interesting things happen in different domains; Machine Learning, Neurobiology, Computer Vision, Physics, Computer architecture, Neuromorphic). This makes Neural Networks more relevant for the Embedded Systems Community

First Introduce the neural network model, if we don't know what it is, we don't know why it is used.




Inspired by the Biological Neurons in the Brain these Neural network models are developed.

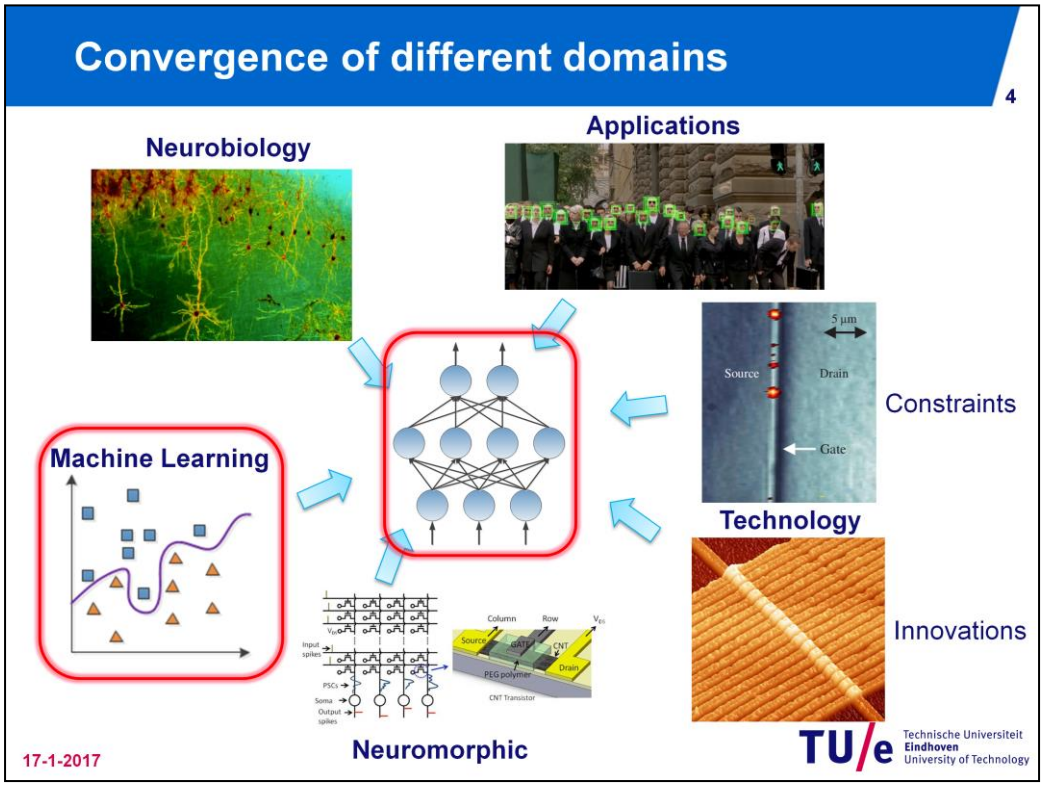
Later on more information about Neurobiology. For now enough to know that neurons have a connection with an efficiency that is used to send over signals. When enough signals arrive at the postsynaptic neuron, it fires an new spike.




Quick recap of the perceptron model, can separate input data to classes, and learn separation between classes.

Problem can not solve problems that require non-linear separation. For example an XOR function, in practice many problems require non-linear separation.

If we want the pattern on the input to give a high output value te one values on the input should be multiplied with positive weights. If we want that a different pattern has a low output we should set all weights connected to other pixels to a negative value. In this situation a pattern should match the input. We could use the bias input with a big negative value to force the output to zero if a different pattern that ‘one’ on the input is pressed.

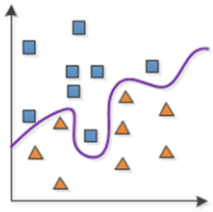



From single perceptron's it is possible to build more powerful classifiers that can solve problems that are non-linear. That is something which is interesting for the Machine Learning community. The desirable functionality of learning a behavior to a machine is very useful, the techniques are closely related to optimization theory.

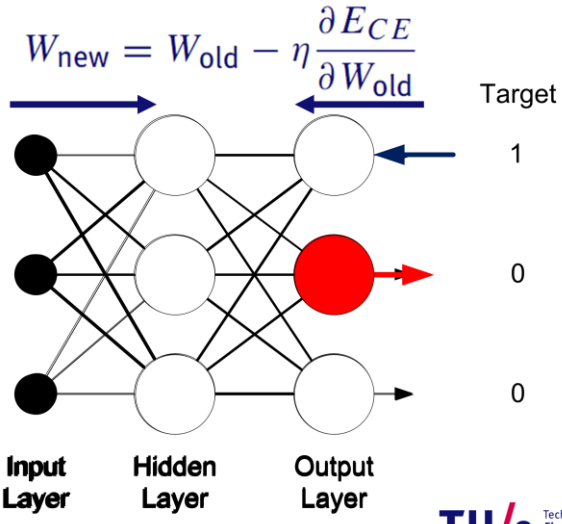
## Multi Layer Perceptron (1979)

5

- **Training is done by error back-propagation**



| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |




17-1-2017

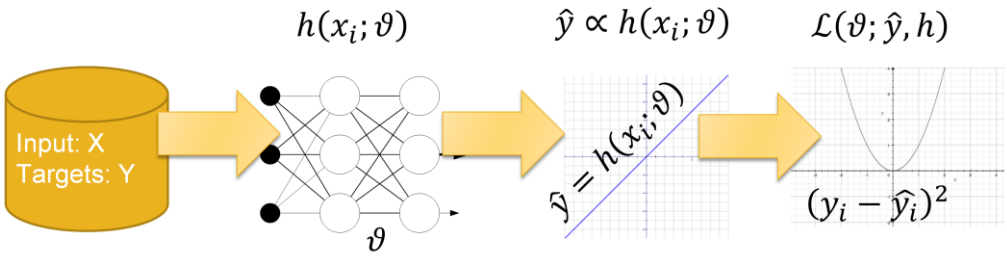
TU/e Technische Universiteit Eindhoven University of Technology

Single perceptrons can be connected to form a Multi Layer Perceptron (MLP) also called Aritificial Neural Network (ANN). Because the different representations that can be build in the hidden (middle) layer and the non-linear activation function, this network can separate non-linear problems. Training is done by stochastic gradient decent this involves updating the weights in the negative direction of the error gradient. This process is repeated for a big set of input patterns until the error converges to a low value. The gradient computation and weight updates can be implemented efficient by the error back-propagation algorithm.

# Optimization Through Gradient Descent (1)



- Collect annotated data
- Define model and initialize randomly
- Predict based on current model
  - In neural network jargon “forward propagation”
- Evaluate predictions






TU/e Technische Universiteit Eindhoven University of Technology

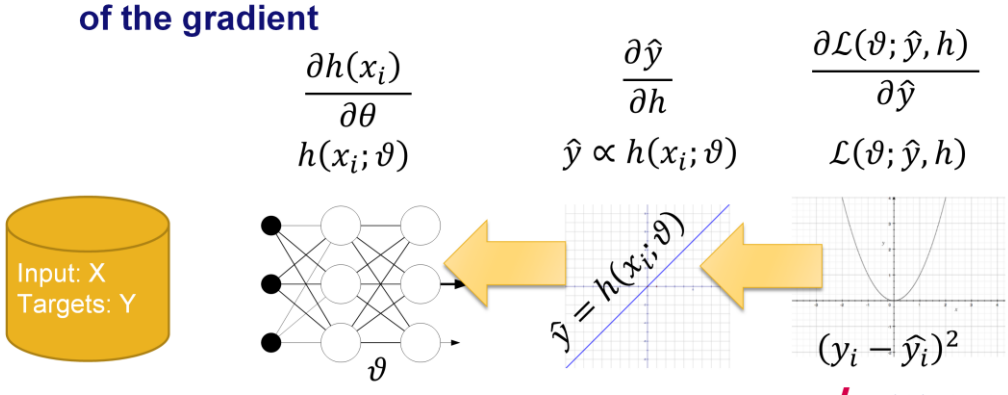
# Optimization Through Gradient Descent (2)



- **Use the derivative of the error to optimize:**

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta \mathcal{L}$$

- **The most important component is the gradient**
- **Then update the parameters in the negative direction of the gradient**



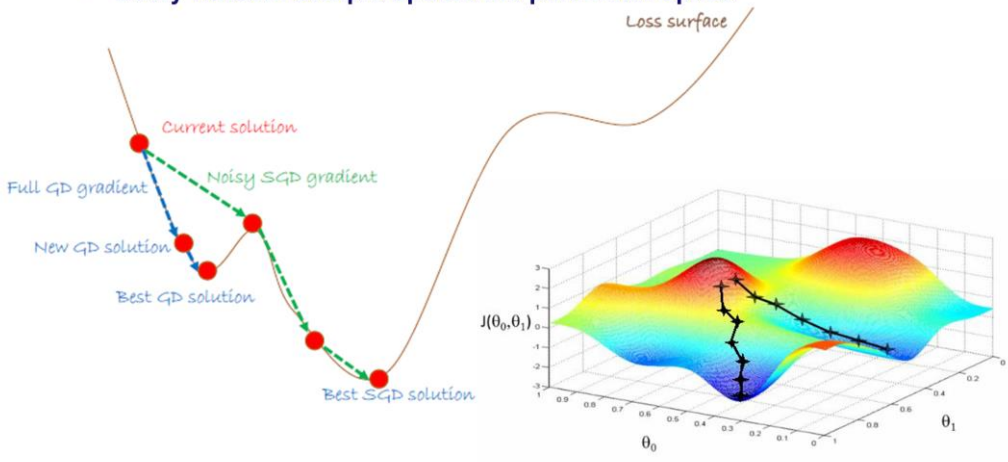



TU/e Technische Universiteit Eindhoven University of Technology

# Optimization Through Gradient Descent (3)



- Follow the path towards local minima in the parameter space
- Stochastic Gradient Descent
  - Every random sample update the parameter space



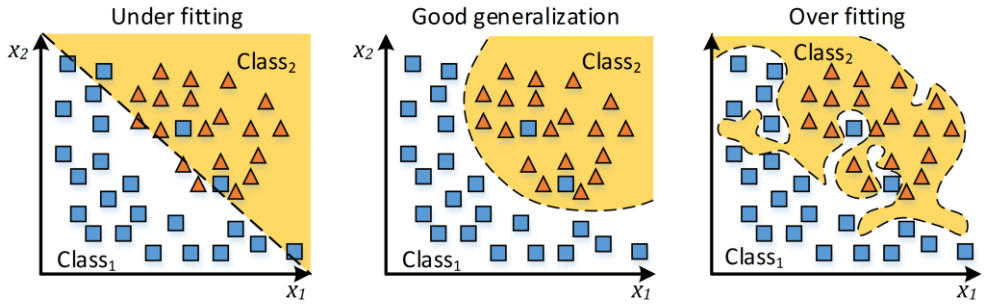



TU/e Technische Universiteit Eindhoven University of Technology

# Generalization



- Take an abstract representation
- Add details
- Add to many details
- A common problem in Machine Learning problems





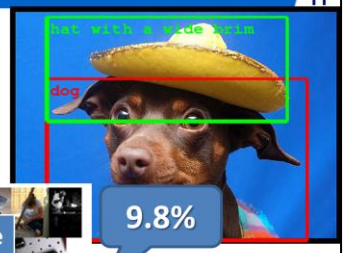

TU/e Technische Universiteit Eindhoven University of Technology

# Data is Everywhere

# Large Scale Visual Recognition Challenge



- IMAGENET 2012 Classification
  - Many training samples: 1,281,167
  - Large number of classes: 1000

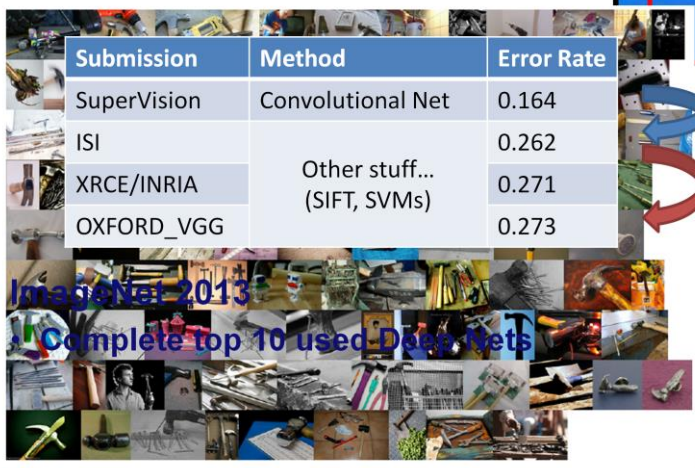



| Submission | Method | Error Rate |
|---|---|---|
| SuperVision | Convolutional Net | 0.164 |
| ISI | Other stuff… (SIFT, SVMs) | 0.262 |
| XRCE/INRIA | | 0.271 |
| OXFORD_VGG | | 0.273 |

9.8%

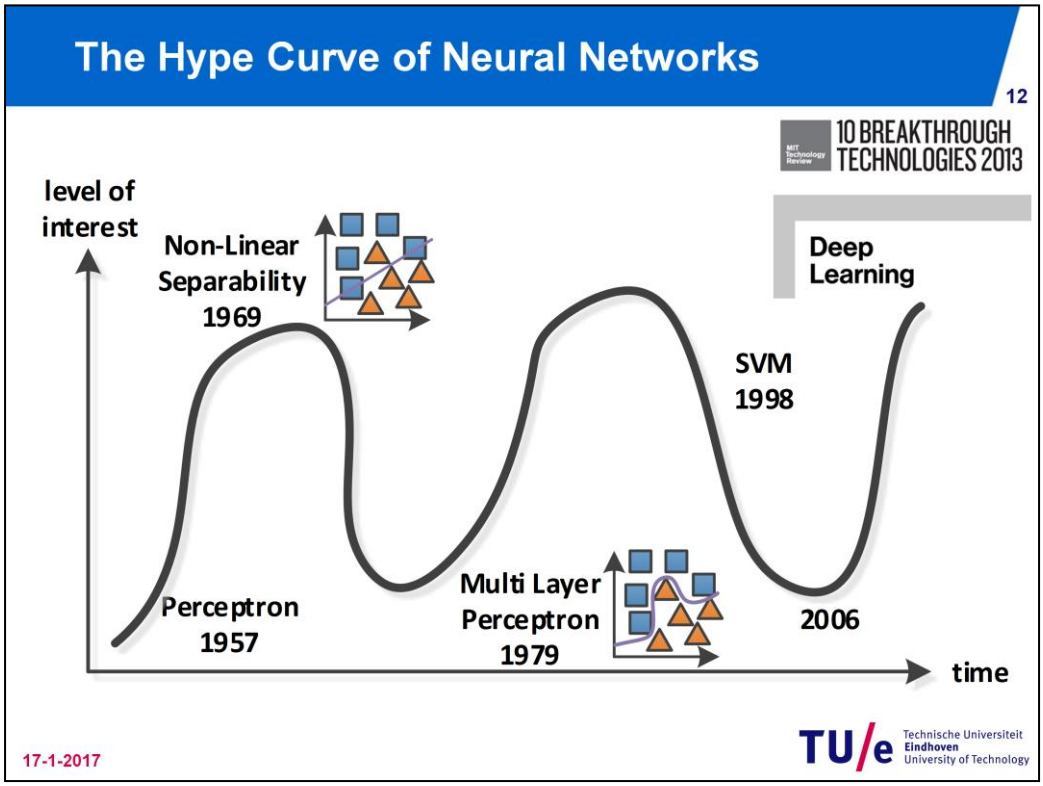
1.2%

- ImageNet 2013
  - Complete top 10 used Deep Nets





TU/e Technische Universiteit Eindhoven University of Technology




The idea of a learning perceptron introduced a hype, the famous XOR prove that it could only solve linearly separable classification problems removed much interest.

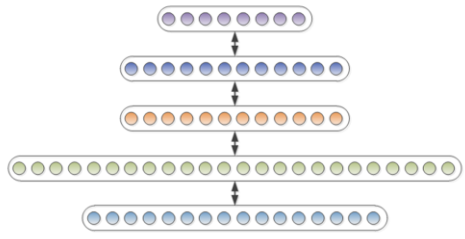
The MLP solution created a hype again, but overtraining and generalization was still a problem. Training required complex parameter tuning and Support Vector Machine showed to have better properties for generalization because they maximize class difference.

# Deep Big Neural Networks



- Deep Big Neural networks outperform SVM
- Complex models with over 10 billion parameters
- Unreasonably effective at classification tasks

- ≥ 5 layers
- 1000s of nodes
- connection constraints



Big Deep Network

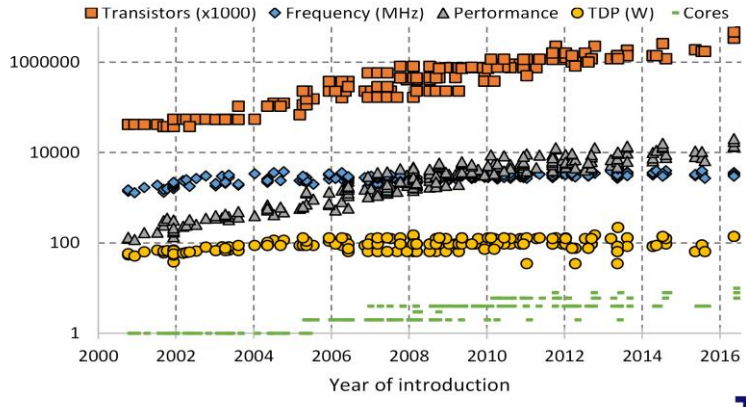


TU/e Technische Universiteit Eindhoven University of Technology

# The unreasonable effectiveness of deep networks



- Big data everywhere
  - Google, Facebook, Amazon
  - NSA, Government, Banks
- Moore's law

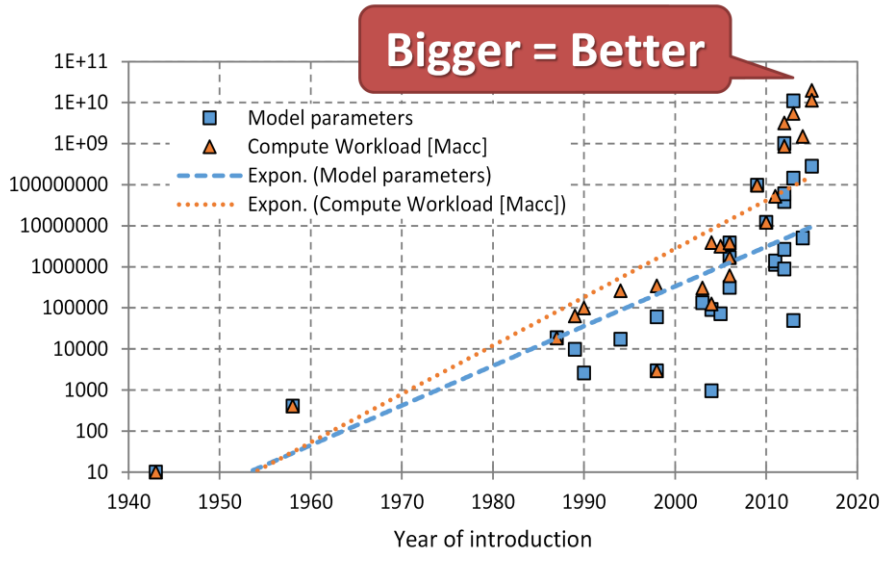





TU/e Technische Universiteit Eindhoven University of Technology

# Trends in Deep Learning





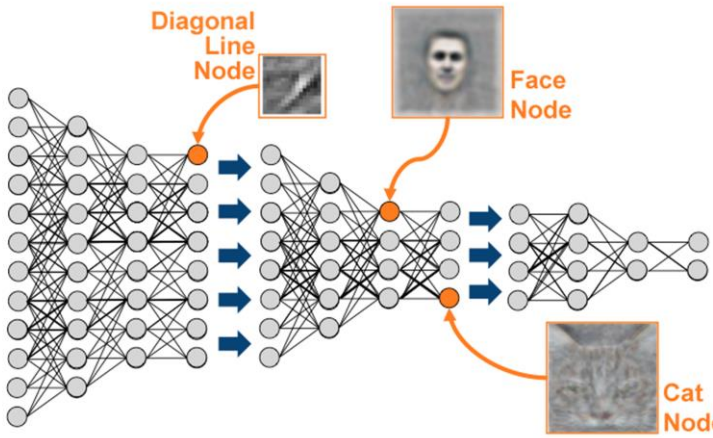



TU/e Technische Universiteit Eindhoven University of Technology

# Example of a large network trained by Google

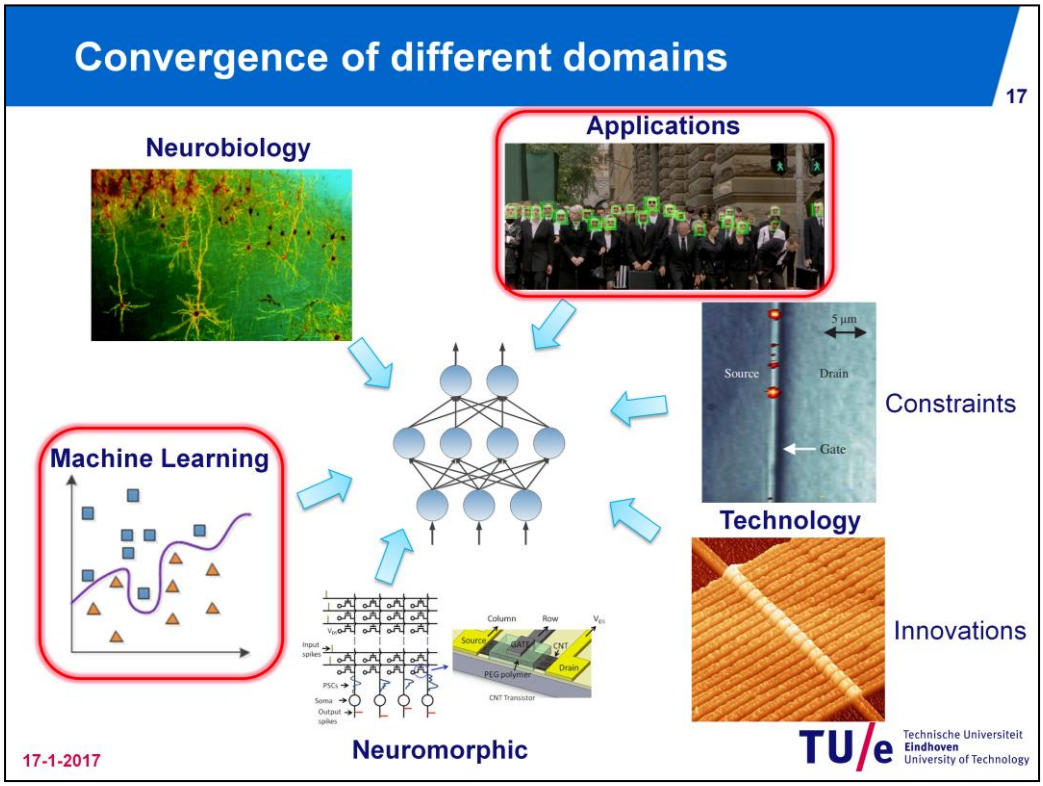


- 10 Billion free parameters
- Trained on many millions of YouTube videos

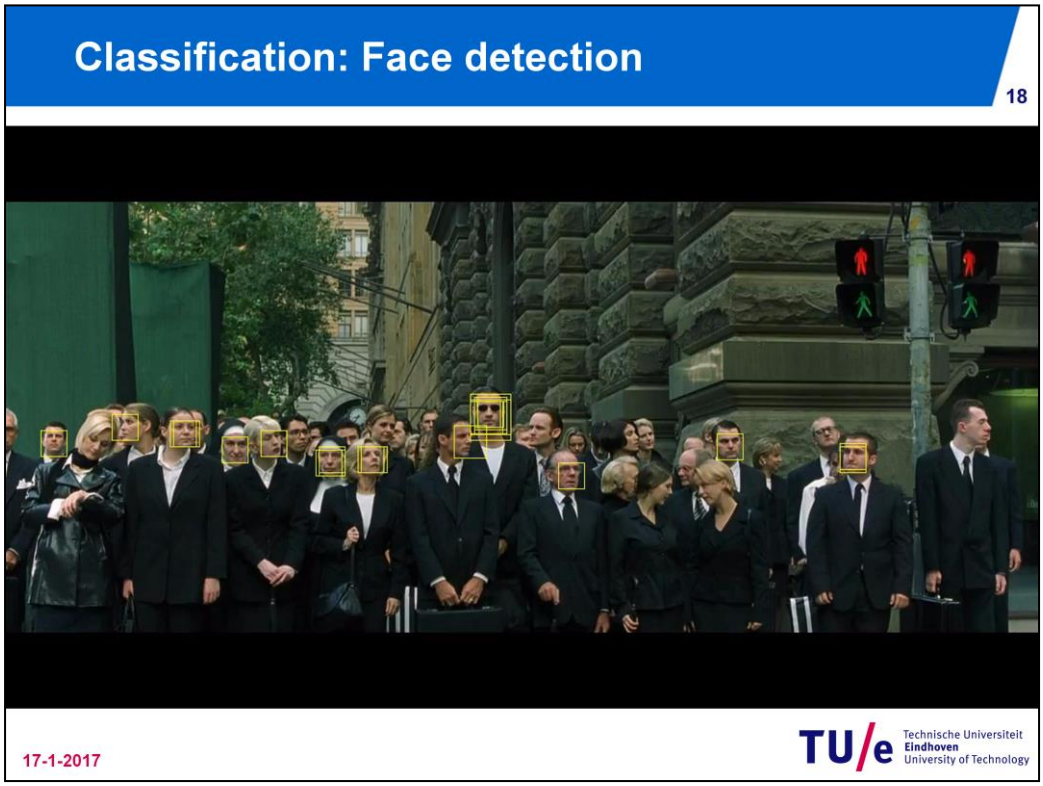





TU/e Technische Universiteit Eindhoven University of Technology




A system that can learn from example can also solve many problems an application designer encounters. Therefore many applications are driven by neural network based machine learning.

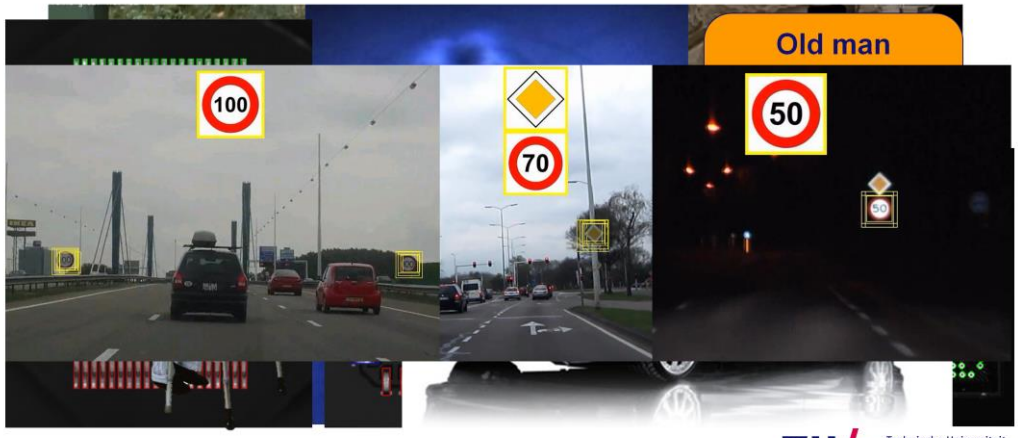



Read this reference for a good description of the CNN approach to face detection: **Garcia C.**, Delakis M., “Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 26(11), November 2004, p. 1408-1423.

# Intelligent Vision Applications



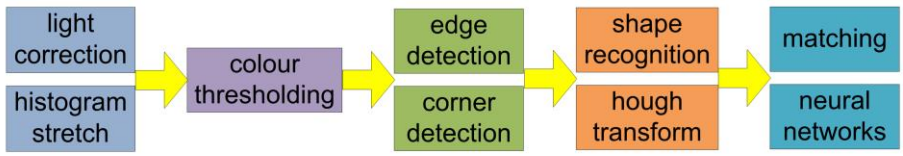
- Emerging field of research
- Applications in many domains
- Examples: Security, Industrial, Medical, Automotive



# Classical recognition systems are stupid

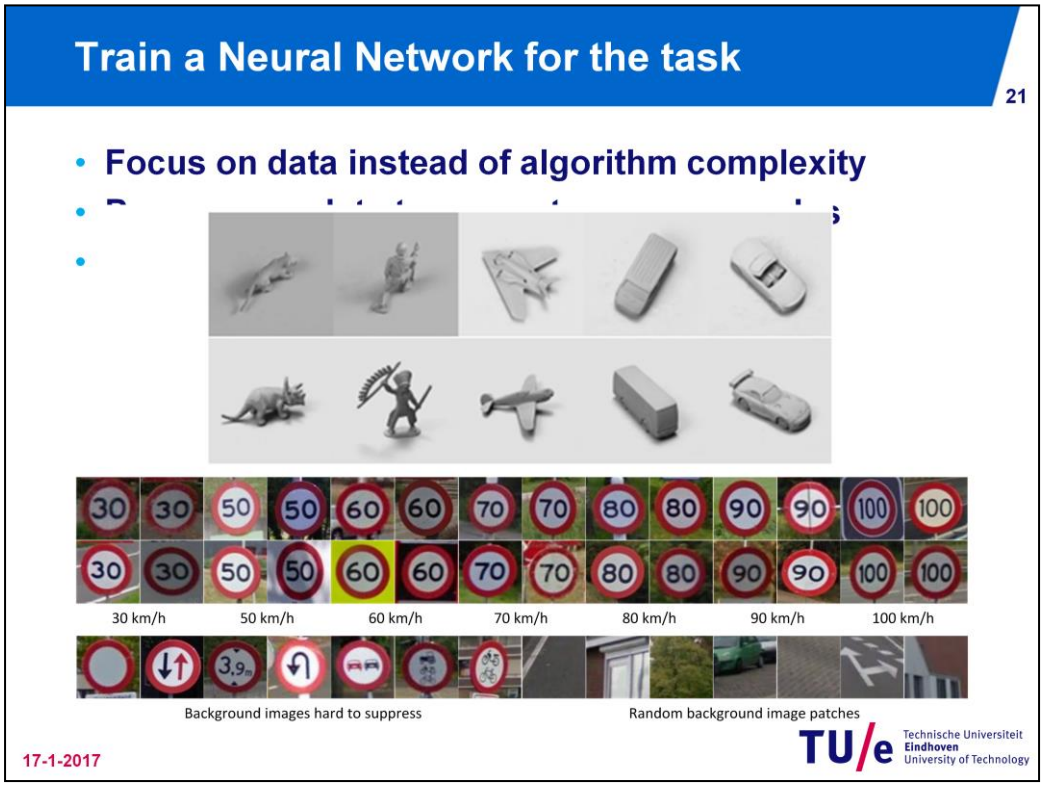


- Design is based on knowledge of the task
- Carefully tuned pipeline of algorithms
- Really complex for real world problems
- Design must be redone if the task changes





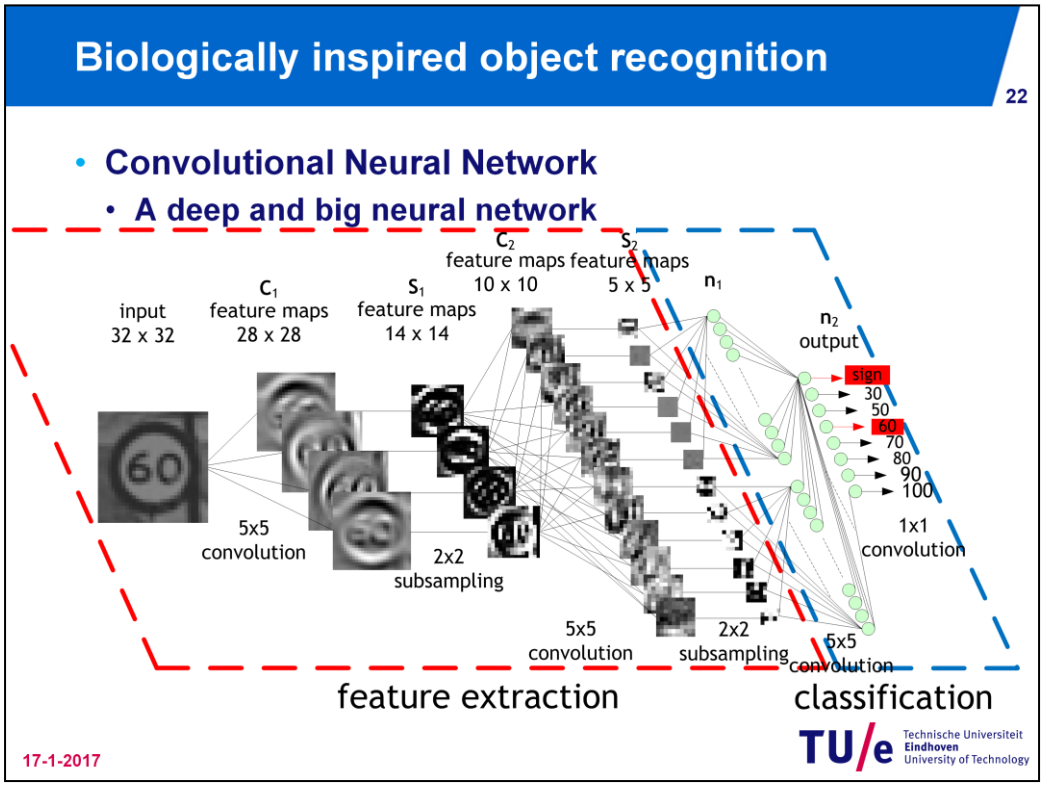

TU/e Technische Universiteit Eindhoven University of Technology



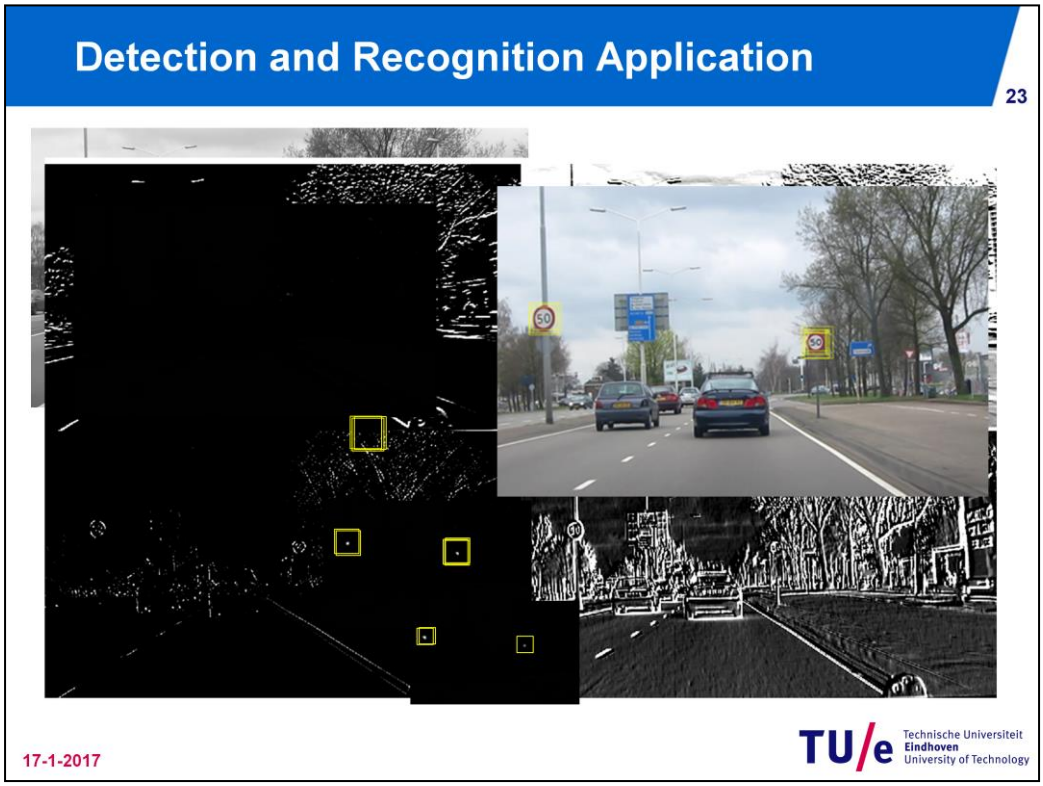

Focus on data instead of algorithm complexity

Pre-process data to generate more examples

Use a test set to verify generalization




Classify features with a hierarchy of trained simple detectors. Each stage simple features are combined into more complex features. If you want to know all details of this type of neural network read this reference (is a big paper but contains most of the details): Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: **Gradient-Based Learning Applied to Document Recognition**, *Proceedings of the IEEE, 86(11):2278-2324, November **1998**.*




For more information regarding the speed sign detection and recognition read our paper:

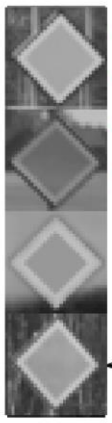
M.Peemen, B.Mesman and H.Corporaal, *Speed Sign Detection and Recognition by Convolutional Neural Networks*, In: Proceedings of the 8th International Automotive

Congress. pp. 162-170 (2011)

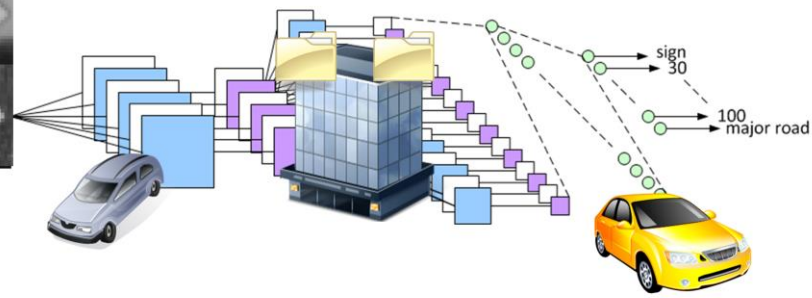
# Speed Sign Detection and Recognition

# Advantage of flexibility





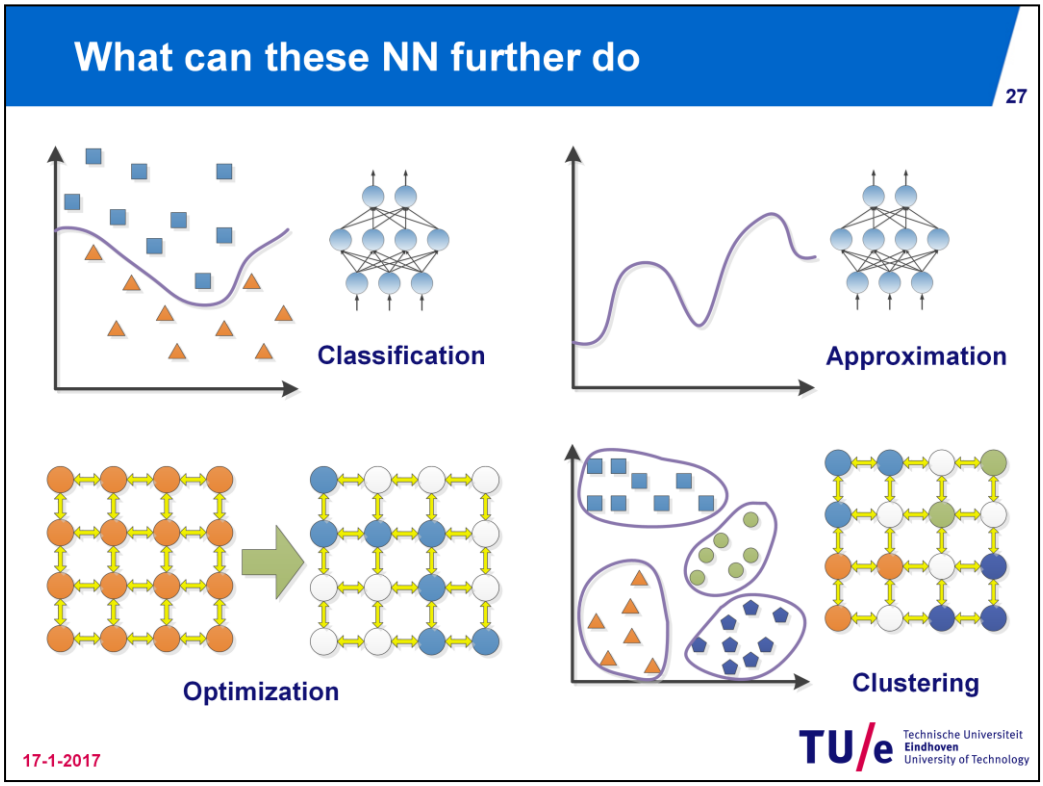
- Extend existing trained network
- Add new road signs and restart training
- New weight file is new functionality
- Send new weight file to users (100 KB)






TU/e Technische Universiteit Eindhoven University of Technology

# Major road detection

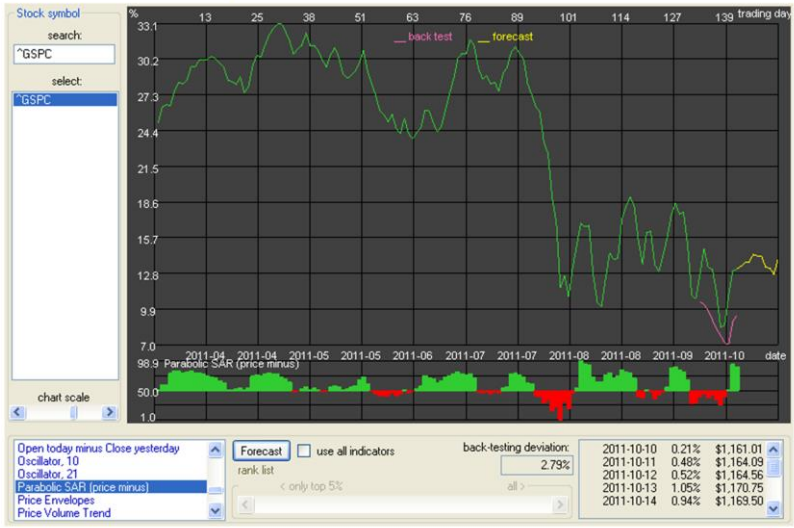


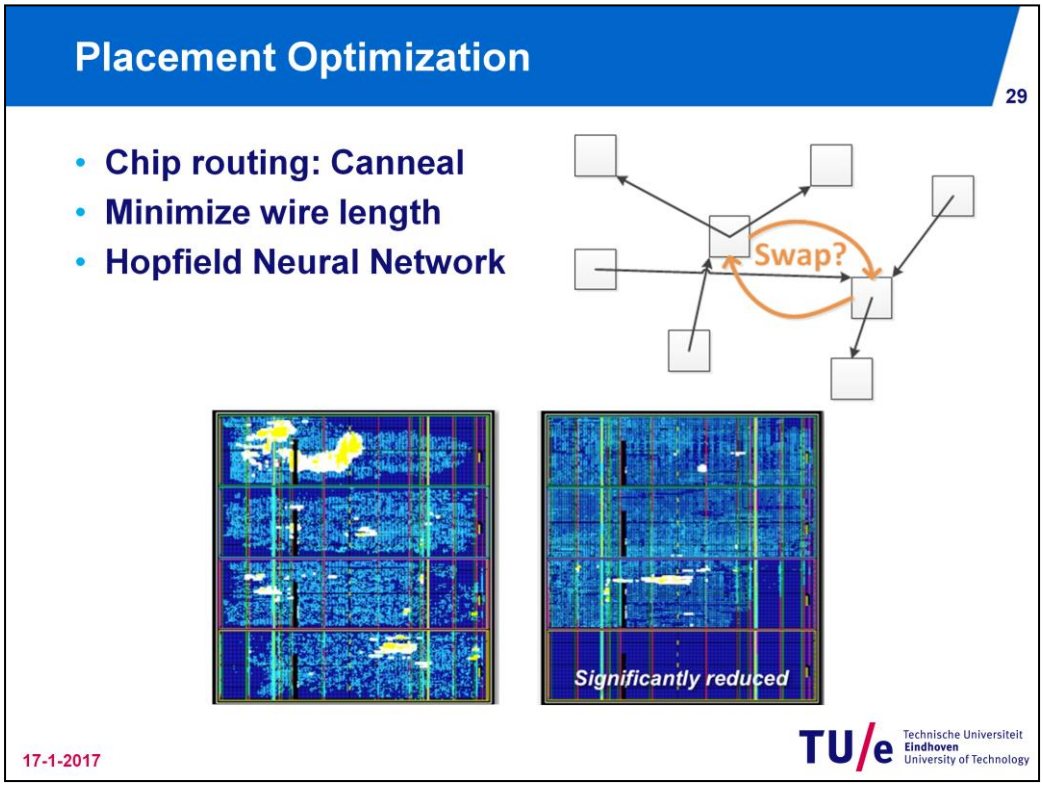

Four example application domains that ANN can solve very well

# Function Approximation



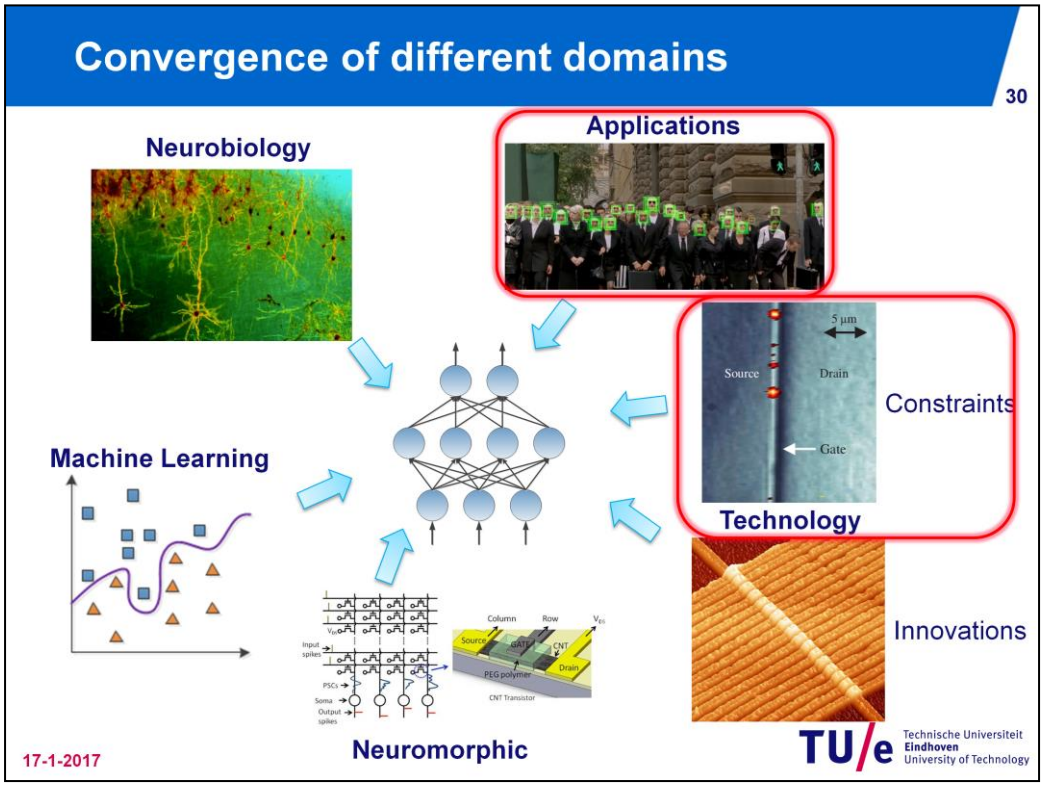
- **Stock market prediction: Black Scholes**






Read the paper on applications that can be solved with Neural networks:

BenchNN: T.Chen, Y.Chen, M.Duranton, Q. Guo, A. Hashmi, M.Lipasti, A.Nere, S.Qiu, M. Sebag, O.Temam. On the Broad Potential Application Scope of Hardware Neural Network Accelerators, IEEE International Symposium on Workload Characterization (IISWC), November 2012




Due to recent changes in the field of chip fabrication some constraints force this Tech branch to find solutions that can cope with the new constraints.

Neural nets can provide a few solutions to these new constraints.

## Technology Constraints



- Dark Silicon
- Defect tolerance



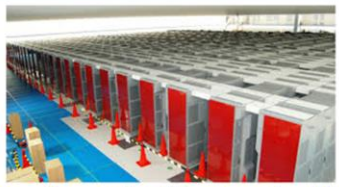
TU/e Technische Universiteit Eindhoven University of Technology

Two interesting constraints that motivate the industry to come op with solutions.

"What do you do when chips get too hot to take advantage of all of those transistors that Moore's Law provides? You turn them off, and end up with a lot of dark silicon — transistors that lie unused because of power limitations. As detailed in MIT Technology Review, Researchers at UC San Diego are fighting dark silicon with a new kind of processor for mobile phones that employs a hundred or so specialized cores. They achieve 11x improvement in energy efficiency by doing so."

# Energy Efficiency

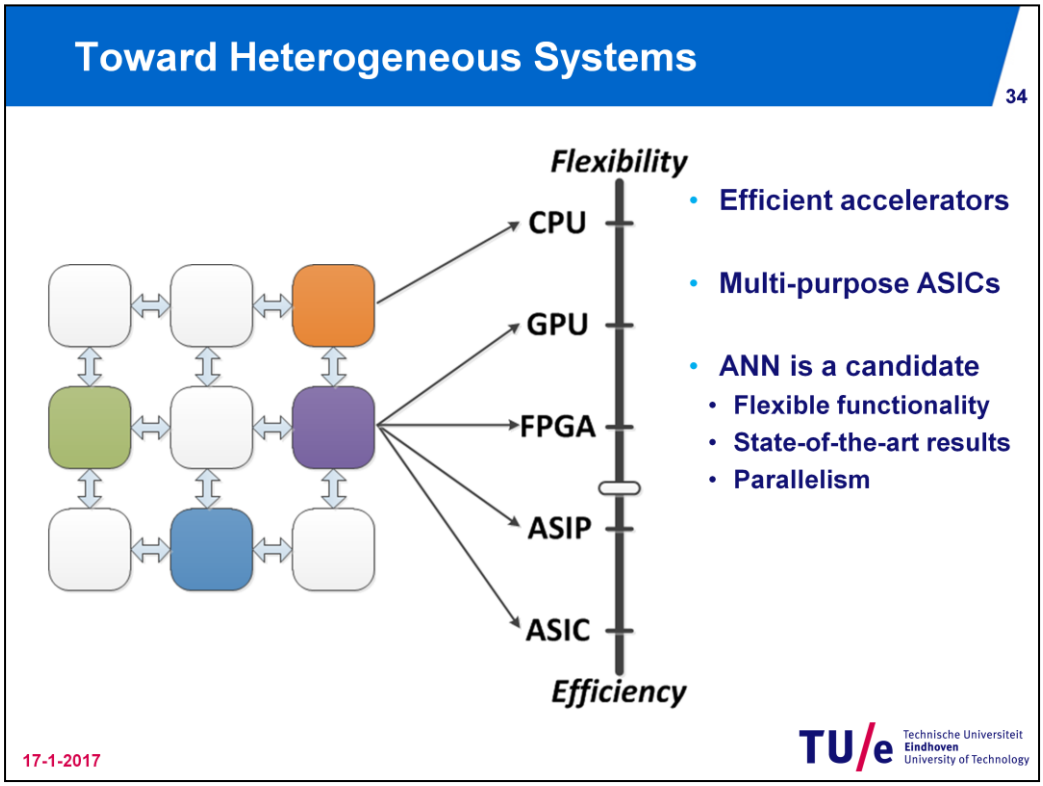




Super Computer (K computer, Fujitsu)
8.2 billion Megaflops => 9.9 million watts
~ 800 Megaflops / watt

Human Brain
2.2 billion Megaops => 20 watts
~ 110 Teraops / watt

iPad 2
170 Megaflops => 2.5 watts
~ 68 Megaflops / watt




As an efficient multi purpose accelerator Hardware Artificial Neural Networks could be used.

Functionality can be reprogrammed by updating the connections.

For various application fields these give state of the art results, as shown in previous slides.

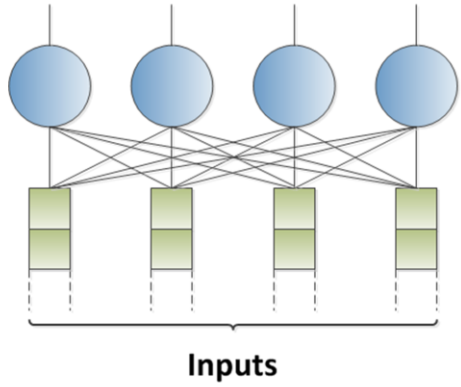
The fundamental operations contain a lot of parallelism.

## Developing ANN Accelerators



```
for i = 1:N
  Y[i] = Bias[i]
  for k = 1:K
    Y[i] += X[k] * W[i][k]
  Y[i] = Sigmoid(Y[i])
```

$$y_i = \varphi\left(b_i + \sum_k x_k \cdot w_{ik}\right)$$






TU/e Technische Universiteit Eindhoven University of Technology

How would we develop such an accelerator.

We have this mathematical description, and a graphical network. Let's look at the code that describes this network.

## Time-Multiplexed Accelerator

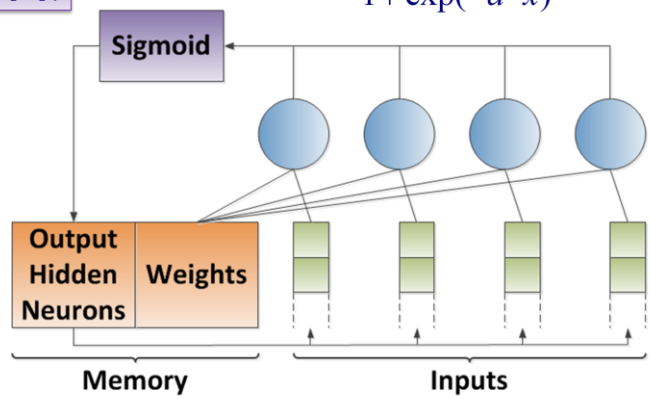


```
for i = 1:N
  Y[i] = Bias[i]
  for k = 1:K
    Y[i] += X[k] * W[i][k]
  Y[i] = Sigmoid(Y[i])
```

$$y_i = \varphi\left(b_i + \sum_k x_k \cdot w_{ik}\right)$$

$$\varphi(v) = \frac{1}{1+\exp(-a \cdot x)}$$

- Load Bias
  - X[1:N] = 1
  - W[i][1] = Bias[i]
- Perform MACC
- Sigmoid
  - Approximate





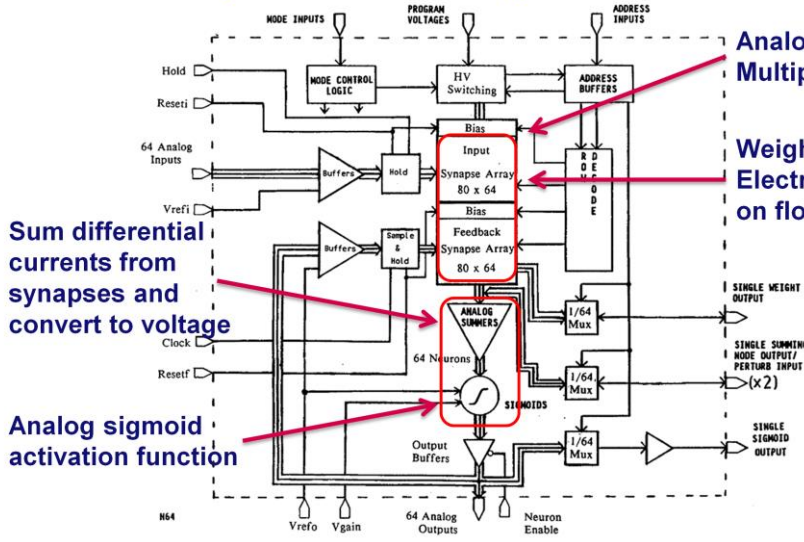

TU/e Technische Universiteit Eindhoven University of Technology

From a network towards hardware with memories, and computing elements.
How could you load bias values into this system?

## Analog Intel ETANN 1990



- **Electrically Trainable Analog Neural Network**




Analog Gilbert-Multiplier Circuits

Weights stored as Electrical Charge on floating gates

Sum differential currents from synapses and convert to voltage

Analog sigmoid activation function



TU/e Technische Universiteit Eindhoven University of Technology

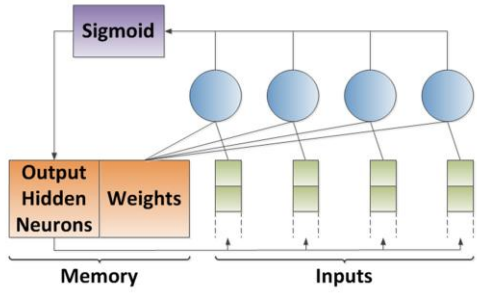
In the old days they tried to do this analog. Digital multipliers consume a lot of logic. Still this system needs sample & hold circuitry to process a net layer by layer.

## Digital Implementation

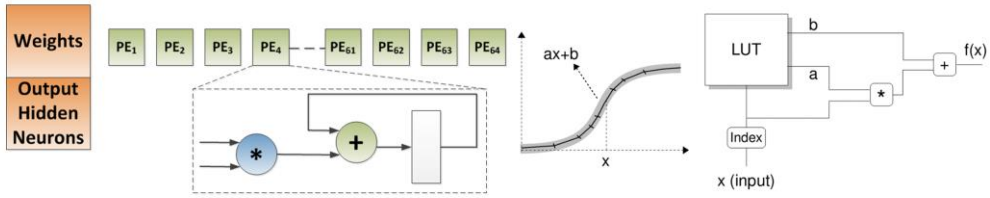


- **Sigmoid Function**
  - **Look Up Table**
  - **Use linear approximation**

$$\varphi(x) \approx b_i + a_i \cdot x$$



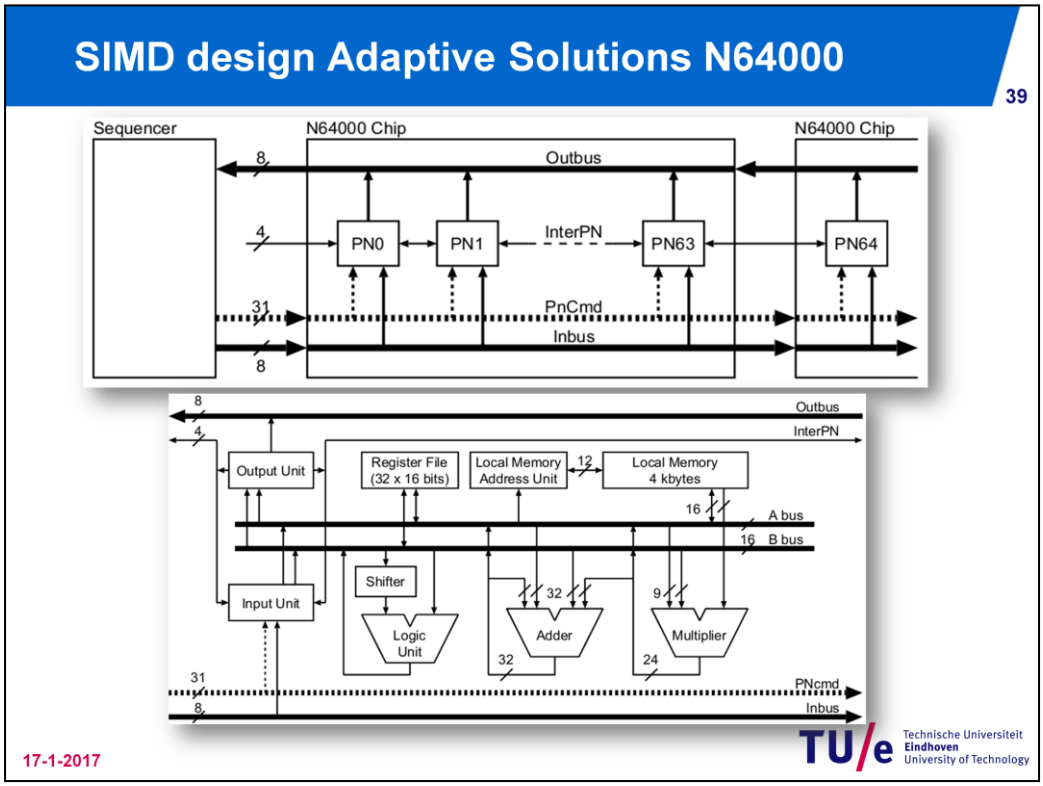

- **SIMD Multiply Accumulate**






TU/e Technische Universiteit Eindhoven University of Technology

Use a lot of MACC processing elements and a sigmoid approximation and two memories as basic elements of a digital neuro processor.




Commercial implementations of SIMD neuro processors exist! SIMD with an orthogonal instruction set is quite flexible there exist compilers to code these chips in languages such as C. But not the most efficient approach.

## Conversion to vector operations



$$y_i[n] = \varphi\left( b_i + \sum_k x_k[n] \cdot w_{ik} \right)$$

$$\mathbf{y}[n] = \varphi\left(\mathbf{b} + \mathbf{x}[n] \cdot \mathbf{W}\right)$$

$$\mathbf{Y} = \varphi\left(\mathbf{b} + \mathbf{X} \cdot \mathbf{W}\right)$$

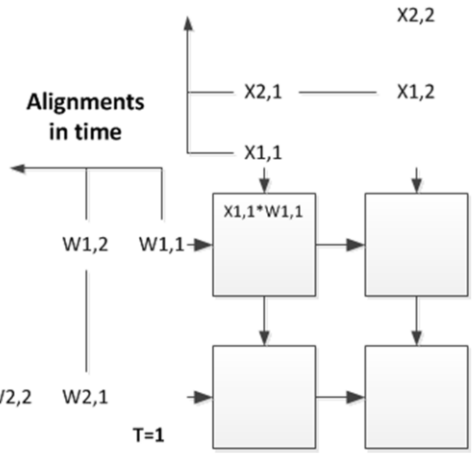
TU/e Technische Universiteit Eindhoven University of Technology

With multiple input patterns it is possible to perform the multiply accumulate operations into Matrix-Matrix products.

## Systolic Matrix Multiplication



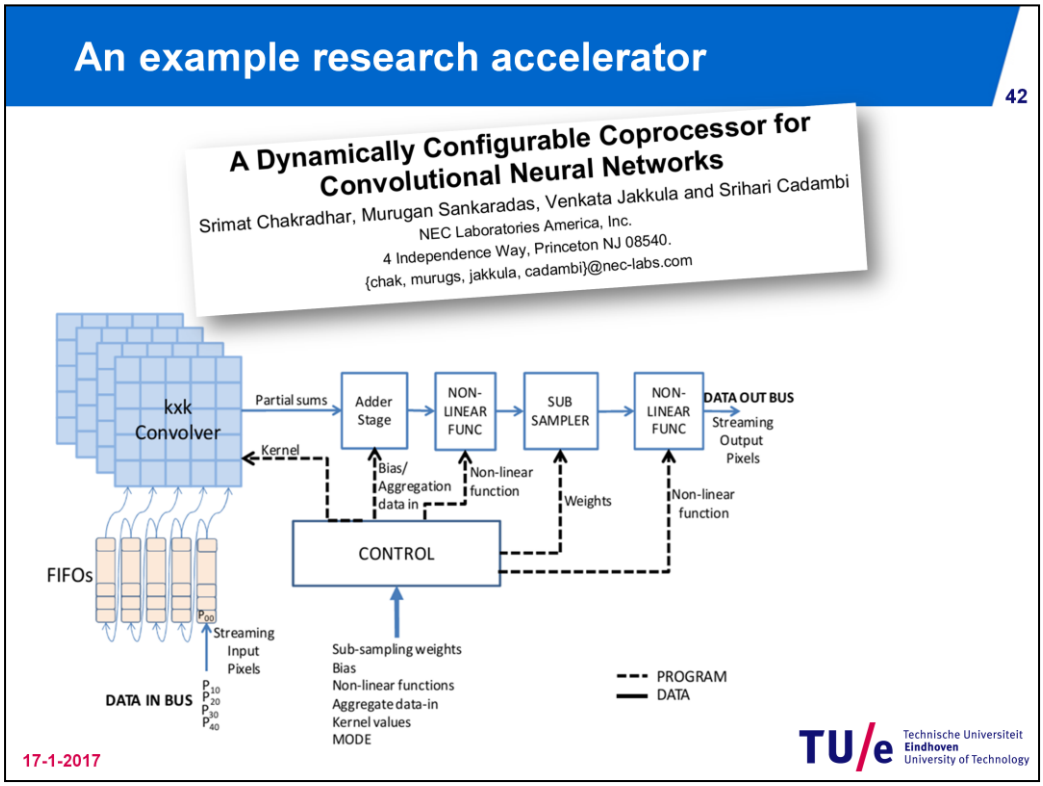
- **Siemens MA16**
  - **High efficiency**
  - **Low flexibility**






TU/e Technische Universiteit Eindhoven University of Technology

Could implement these in a systolic array. So it is possible to stream in your data with much less control. This approach is more efficient but less flexible. If your operations can only have these specialized functions and the designers overlooked some functionality, it is not easy to solve as a programmer. Development of compilers for these architectures is much more complex.



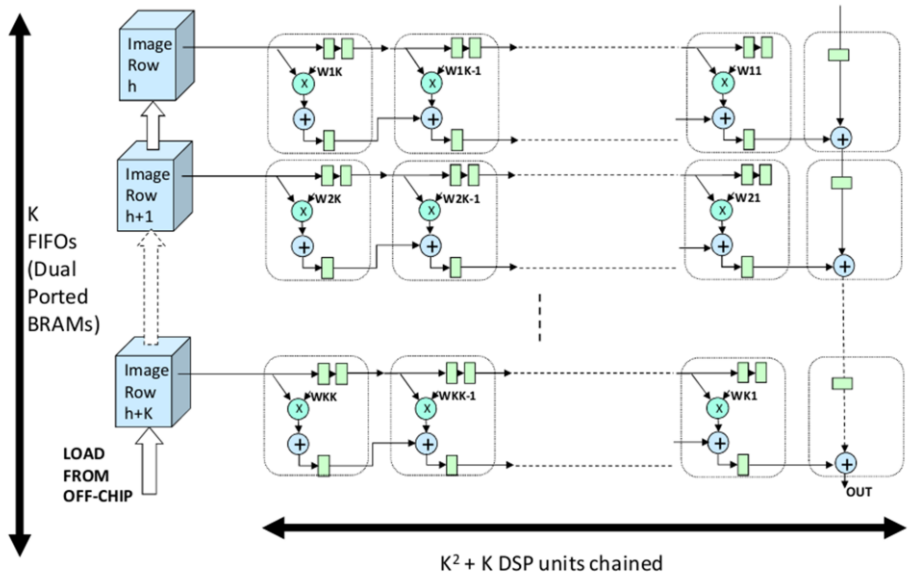

The systolic array used in this accelerator is discussed in another paper:

M.Sankaradas, V.Jakkula, S.Cadambi, S.Chakradhar, I.Durdanovic, E.Cosatto, H.P.Graf, *A Massively Parallel Coprocessor for Convolutional Neural Networks*, In Proc. 20th IEEE

International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2009, Boston, MA

# Systolic 2D Convolution





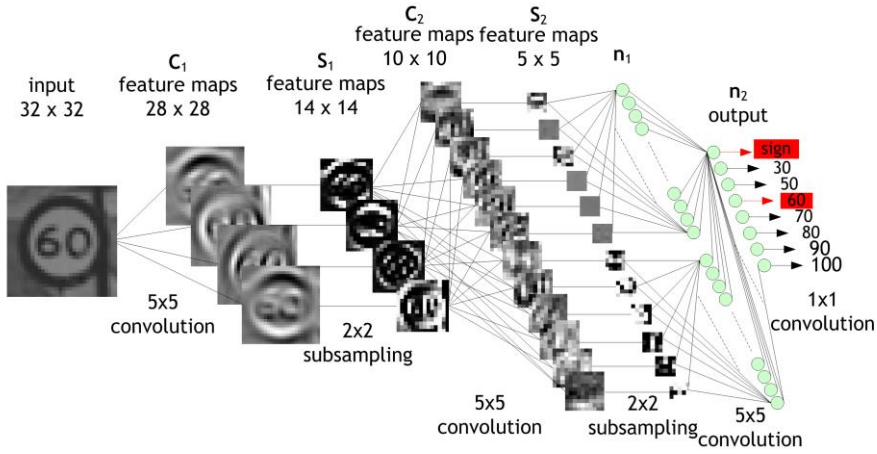



TU/e Technische Universiteit Eindhoven University of Technology

## Convolutional Neural Network



- **Data reuse**





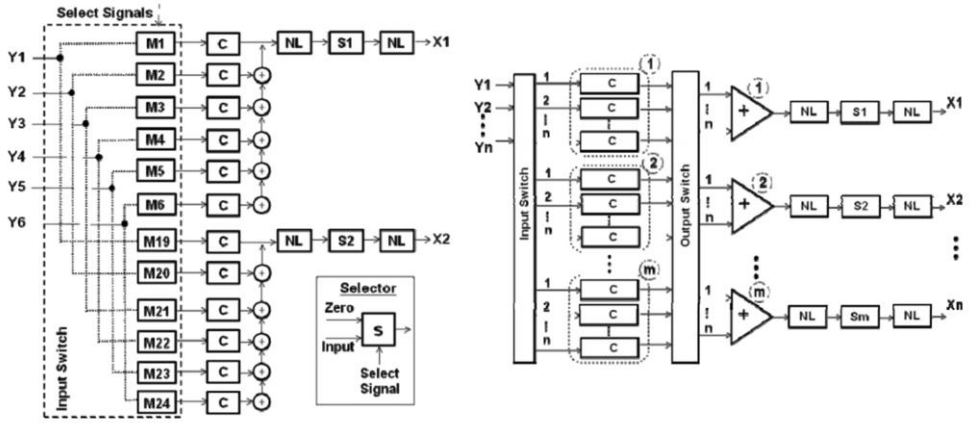

TU/e Technische Universiteit Eindhoven University of Technology

Recap of the intermediate images that need temporal storage.

## Reduce Memory Accesses



- Configurable Number of Input Maps
- Configurable Number of Output Maps






TU/e Technische Universiteit Eindhoven University of Technology

The parallel coprocessor connects the systolic arrays in a reconfigurable way to input pixels or output arrays. This minimizes the amount of stored intermediate image results.

## Is it worth the effort?

46

| CNN (640 x 480 pixels input image) | Multicore (Xeon @ 2.33 Ghz, 8 Cores, 16 GB) BLAS | GPU (C870 @ 1.35 Ghz, 1.5 GB RAM) PCIe | CNP (FPGA @200 MHz) | DC-CNN @ 120 Mhz 20 conv., 128-bit port width, PCI | | Speedup of DC-CNN | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Compute time | Transfer time | Over 2.3 GHz, 8-core | Over 1.35 GHz, 128-core GPU | Over CNP |
| Automotive Safety | 110 ms | 85 ms | - | 13 ms | 11 ms | 8.5x | 6.5x | - |
| Video Surveillance | 212 ms | 163 ms | - | 27 ms | 34 ms | 7.8x | 6.0x | - |
| Face Recognition | 217 ms | 167 ms | - | 42 ms | 11 ms | 5.2x | 4.0x | - |
| Mobile Robot Vision | 147 ms | 114 ms | 100 ms | 21 ms | 11 ms | 7.0x | 5.4x | 4.8x |
| Face Detection | 136 ms | 105 ms | - | 24 ms | 11 ms | 5.7x | 4.4x | - |

- **More important the energy efficiency**

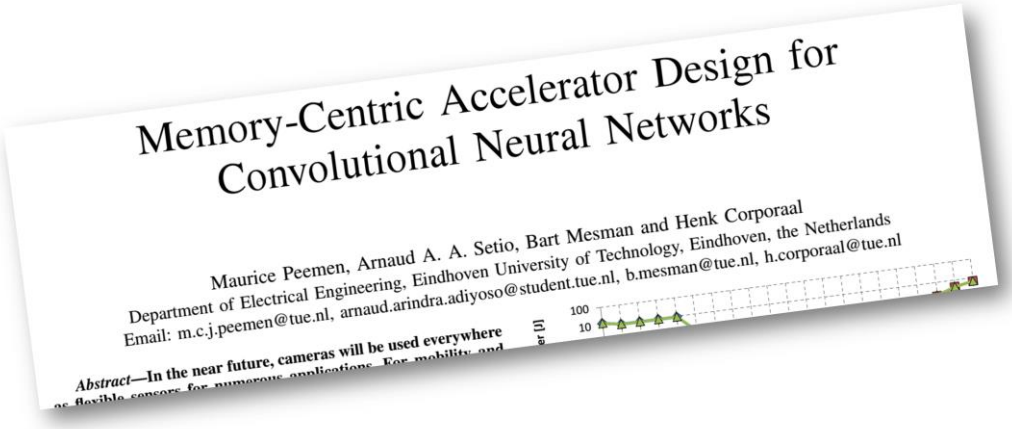
17-1-2017

TU/e Technische Universiteit Eindhoven University of Technology

5x faster and 10x better energy efficiency

# More Flexibility and Better Memory Behaviour?





Memory-Centric Accelerator Design for Convolutional Neural Networks

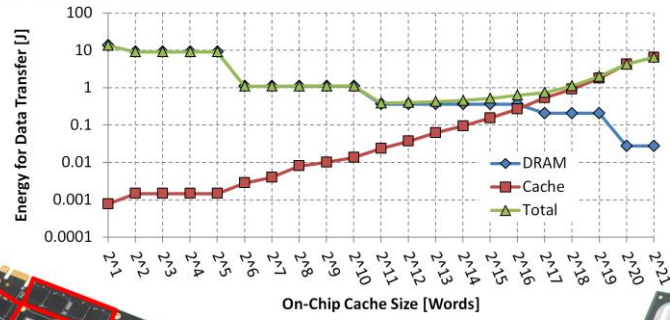
Maurice Peemen, Arnaud A. A. Setio, Bart Mesman and Henk Corporaal
Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, the Netherlands
Email: m.c.j.peemen@tue.nl, arnaud.arindra.adiyoso@student.tue.nl, b.mesman@tue.nl, h.corporaal@tue.nl

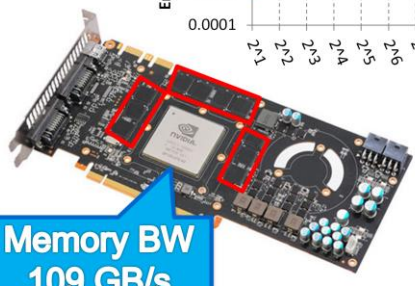
***Abstract*—In the near future, cameras will be used everywhere**

# The performance bottleneck



- Huge data transfer requirements (3.4 billion per layer)
- Exploit data reuse with local memories



Memory BW
109 GB/s

3 MB Cache

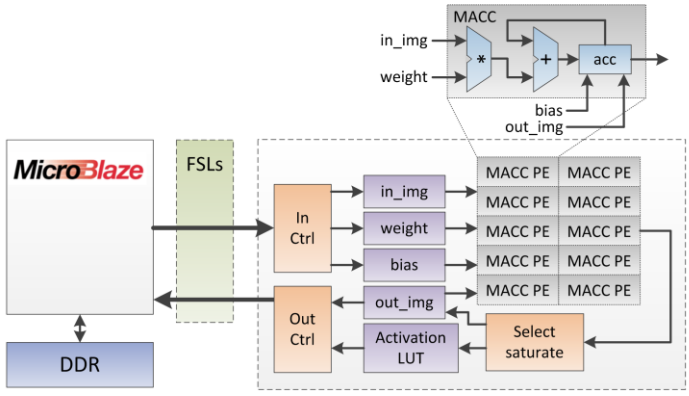


TU/e Technische Universiteit Eindhoven University of Technology

# Accelerator Template



- FPGA prototyping platform: Xilinx Virtex 6
- Designed with Vivado High Level Synthesis (HLS)

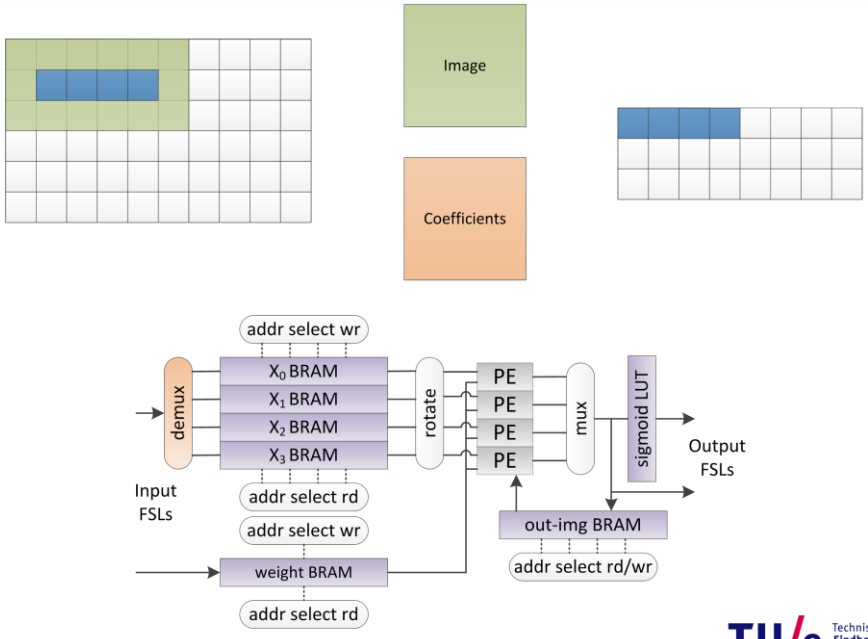





TU/e Technische Universiteit Eindhoven University of Technology

# Programmable Buffers



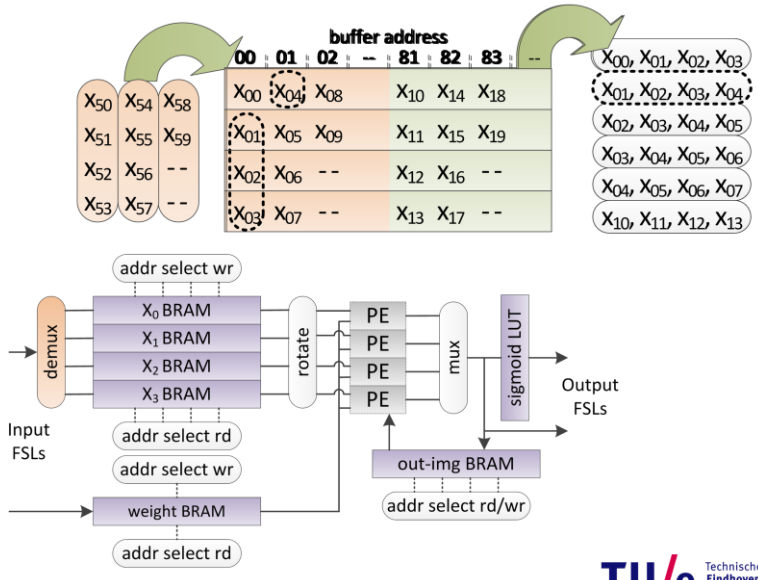





TU/e Technische Universiteit Eindhoven University of Technology

# Programmable Buffers











TU/e Technische Universiteit Eindhoven University of Technology

# What would be the best compute order?

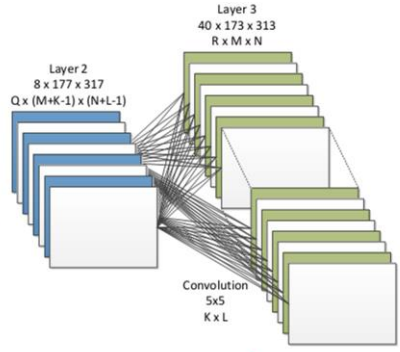


- Small memories have low energy per access
  - Area and Latency advantage
- Big memories can exploit more data reuse

```
for(r=0; r<R; r++){                  //output feature map
  for(q=0; q<Q; q++)                 //input feature map

    for(m=0; m<M; m++){              //slide over input
      for(n=0; n<N; n++){

        if(q==0){Y[r][m][n]=Bias[r];}
        for(k=0; k<K; k++){  //kernel operation
          for(l=0; l<L; l++){
            Y[r][m][n]+=W[r][q][k][l]*X[q][m+k][n+l]
          }
        }
        if(q==7){Y[r][m][n]=sigmoid(Y[r][m][n]);}
      }
    }
  }
}
```






TU/e Technische Universiteit Eindhoven University of Technology

# Improve by locality driven synthesis

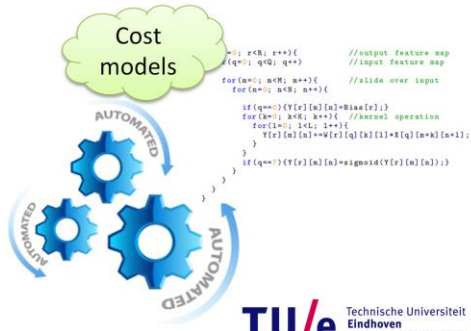


- Loop Transformations
  - Interchange
  - Tiling
- Reduce reuse distance
- A huge design space!
- Use a framework with:
  - Reuse detection
  - Model utilized reuse
  - Model required buffer size
  - Optimize for buffer size

```
for(r=0; r<R; r++){            //output feature map
  for(q=0; q<Q; q++)           //input feature map

    for(m=0; m<M; m++){        //slide over input
      for(n=0; n<N; n++){

        if(q==0){Y[r][m][n]=Bias[r];}
        for(k=0; k<K; k++){  //kernel operation
          for(l=0; l<L; l++){
            Y[r][m][n]+=W[r][q][k][l]*X[q][m+k][n+l];
          }
        }
        if(q==7){Y[r][m][n]=sigmoid(Y[r][m][n]);}
      }
    }
  }
}
```



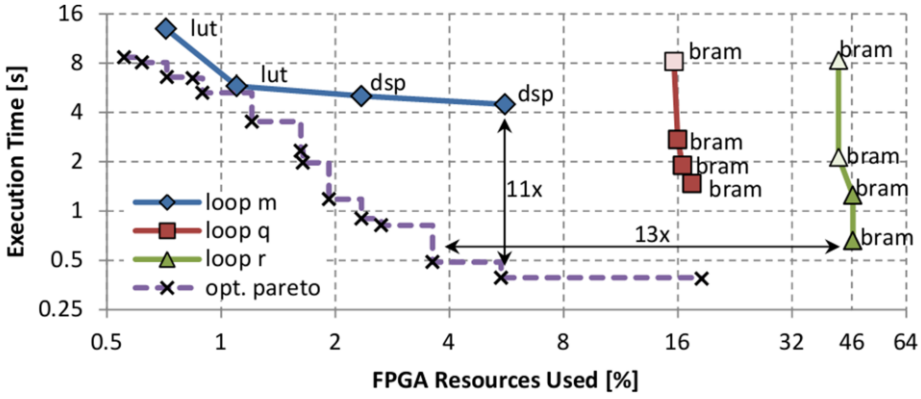



TU/e Technische Universiteit Eindhoven University of Technology

# Compared to manually optimized order



- Up to 13x resource reduction
- Up to 11x performance increase



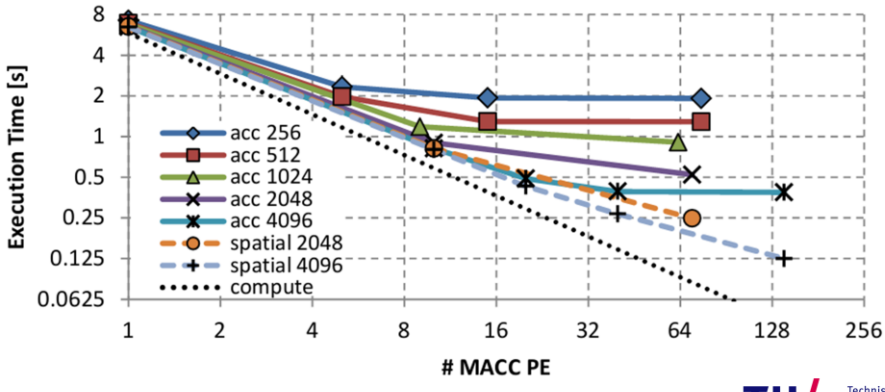



TU/e Technische Universiteit Eindhoven University of Technology

# Memory bandwidth requirements?



- Data layout transformation
- Bandwidth up to 150 MB/s
- Better than an optimized Intel implementation






TU/e Technische Universiteit Eindhoven University of Technology

# What do we achieve?



- Small but flexible accelerators
- Up to 13x smaller
- Up to 11x faster

- XPower Analyzer 4.5 Watt
- External RAM 0.5 Watt

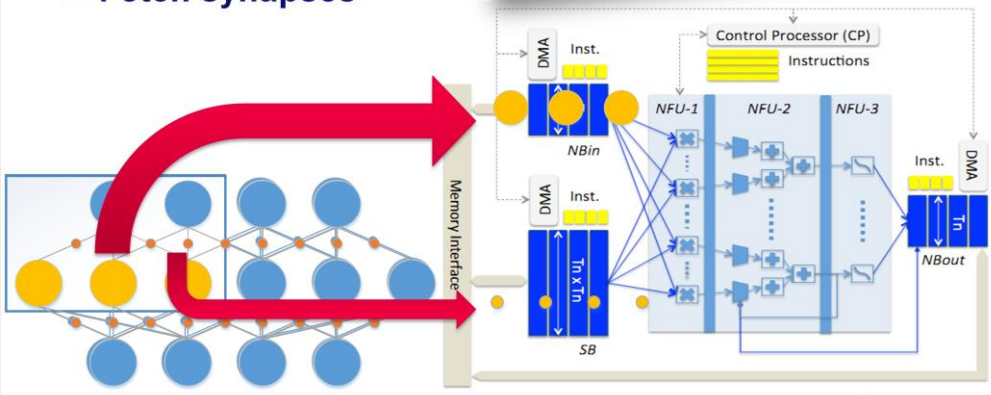
# State-of-the-art accelerator



- Split buffers for BW
- Partial layer
- Fetch input neurons
- Fetch synapses

**DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning**

Tianshi Chen, SKLCA, ICT, China
Zidong Du, SKLCA, ICT, China
Ninghui Sun, SKLCA, ICT, China
Jia Wang, SKLCA, ICT, China
Chengyong Wu, SKLCA, ICT, China
Yunji Chen, SKLCA, ICT, China
Olivier Temam, Inria, France



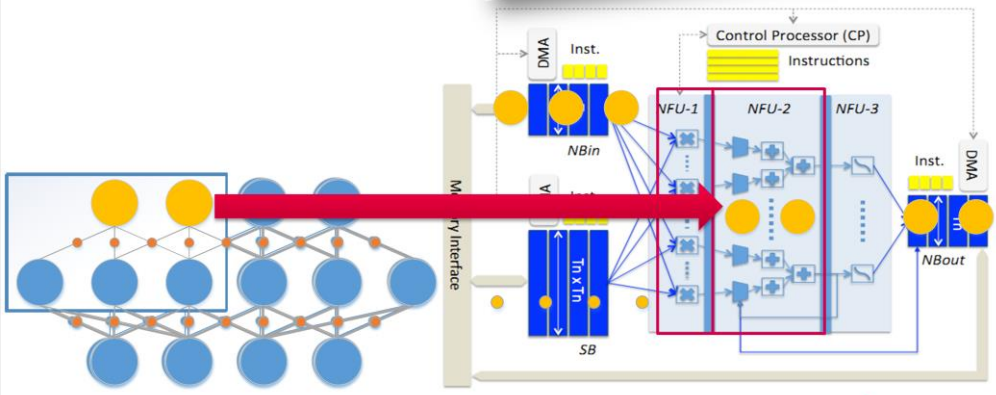
# State-of-the-art accelerator



- **Multiply**
- **Sum neurons inputs**
- **Backup partial sums**

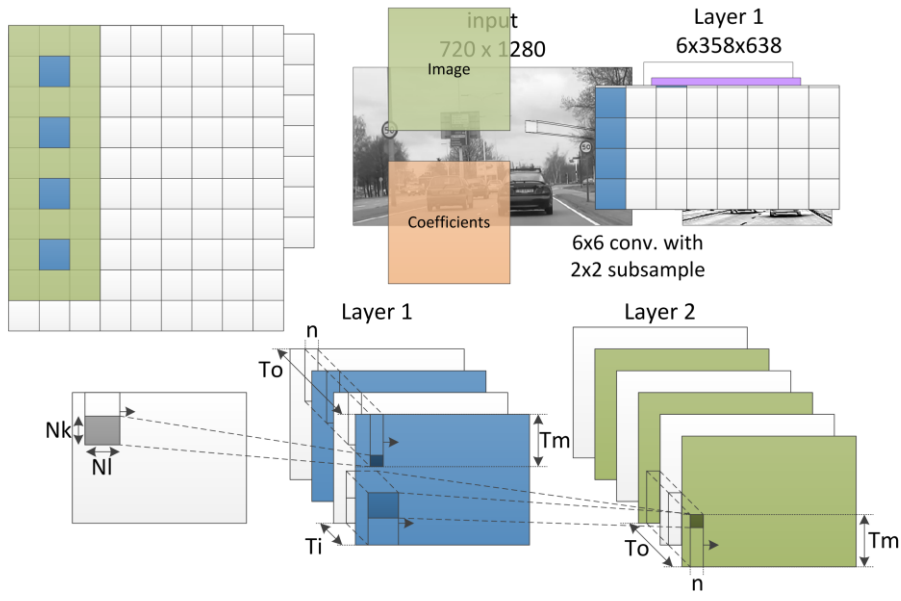
**DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning**

Tianshi Chen, SKLCA, ICT, China
Zidong Du, SKLCA, ICT, China
Ninghui Sun, SKLCA, ICT, China
Jia Wang, SKLCA, ICT, China
Chengyong Wu, SKLCA, ICT, China
Yunji Chen, SKLCA, ICT, China
Olivier Temam, Inria, France



# Convolutional Network Processing





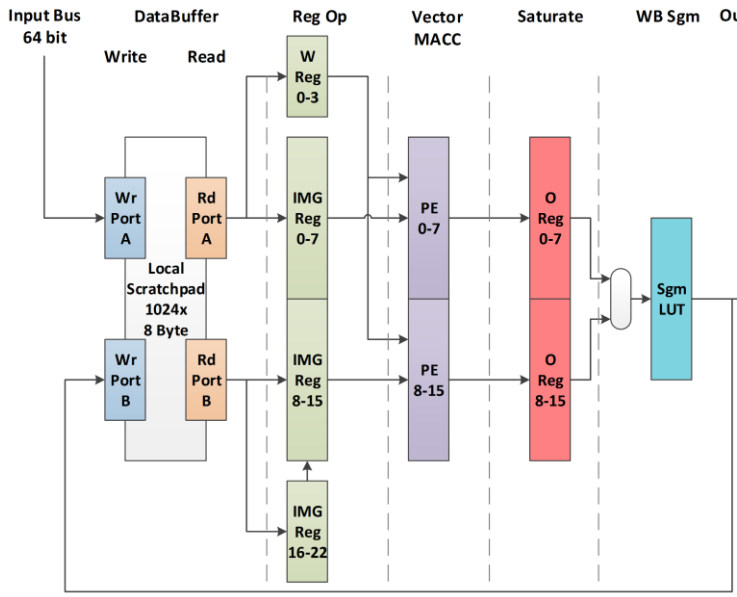

M. Peemen et al. "Inter-Tile Reuse Optimization Applied to Bandwidth Constrained Embedded Accelerators" DATE 2015



TU/e Technische Universiteit Eindhoven University of Technology

# The Neuro Vector Engine



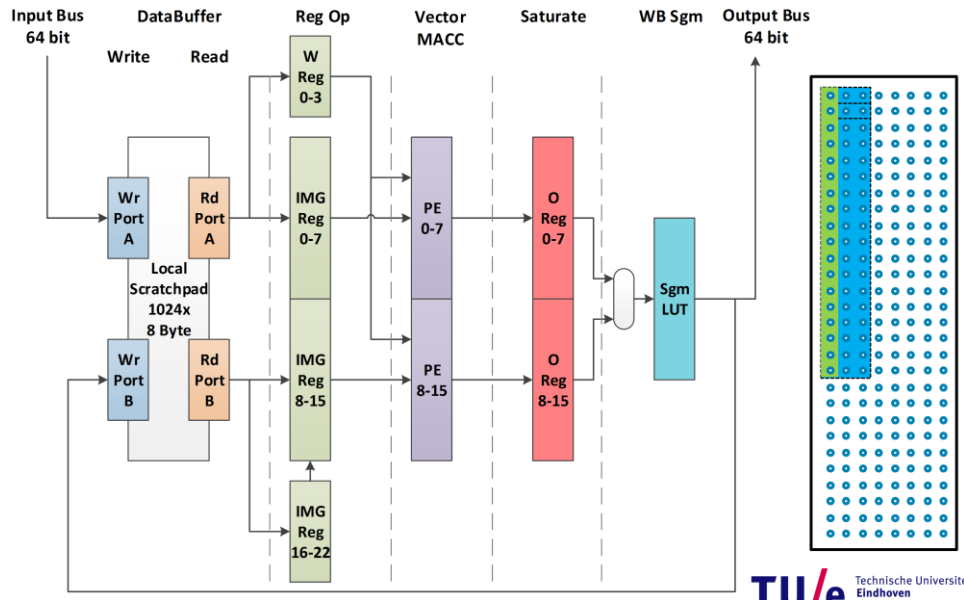



- Programmable
- SIMD operations
- Deep pipelining
- ILP
- Locality oriented

TU/e Technische Universiteit Eindhoven University of Technology

# NVE Operation








TU/e Technische Universiteit Eindhoven University of Technology

# VLIW Programming Model



| | Write Local |
|---|---|
| weights | Wr[b1 w0 w1 -], 0 |
| | Wr[w2 w3 w4 -], 1 |
| | Wr[w5 w6 w7 w8], 2 |
| prolog img | Wr[c0,i0-i7], 4 |
| | Wr[c0,i8-i15], 5 |
| | Wr[c0,i16-i17 - - - -], 6 |
| | Wr[c1,i0-i7], 7 |
| | Wr[c1,i8-i15], 8 |
| | Wr[c1,i16-i17 - - - -], 9 |
| | Wr[c2,i0-i7], 10 |
| | Wr[c2,i8-i15], 11 |
| | Wr[c2,i16-i17 - - - -], 12 |

- 3x3 Convolution filter
- Software Pipelining
- Steady state 10 cycles
  - 16 neighboring 3x3 convolutions
  - 144 Multiply Accumulate ops
- Code reuse with instruction buffer

| Write Local | Read Local | Reg Operation | EX VMAC | WB Sgm |
|---|---|---|---|---|
| 10 Cycles | Ld a 0, [b0 w0-1] | Set i3, Shift i0 i1, Shift W | MAC,w6, c2 i0 | |
| | Ld ab 4 5, [c0,i0-15] | Shift i0 i1, Shift W | MAC,w7, c2 i1 | |
| Wr[c3,i0-i7], 4 | Ld b 6, [c0,i16-17] | Set W0 | MAC,w8, c2 i2 | |
| | Ld a 1, [w2-4] | Set i0 i1, Shift W | Set, b0 | |
| | Ld ab 7 8, [c1,i0-15] | Set i3, Shift i0 i1, Shift W | MAC,w0, c0 i0 | |
| Wr[c3,i8-i15], 5 | Ld b 9, [c1,i16-17] | Shift i0 i1, Set W1 | MAC,w1, c0 i1 | |
| | Ld a 2, [w5-8] | Set i0 i1, Shift W | MAC,w2, c0 i2 | Sigm0, Wrback |
| | Ld ab 10 11, [c2,i0-15] | Set i3, Shift i0 i1, Shift W | MAC,w3, c1 i0 | Sigm1, Wrback |
| Wr[c3,i16-i19 - - - -], 6 | Ld b 12, [c2,i16-17] | Shift i0 i1, Set W2 | MAC,w4, c1 i1 | |
| | NOP | Set i0 i1, Shift W | MAC,w5, c1 i2 | |

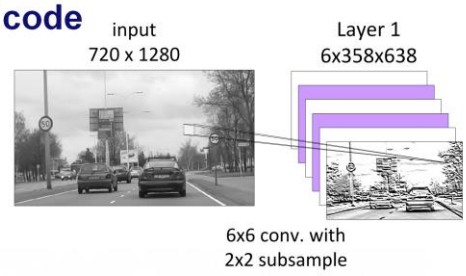


Technische Universiteit Eindhoven University of Technology

# Are You With Me?



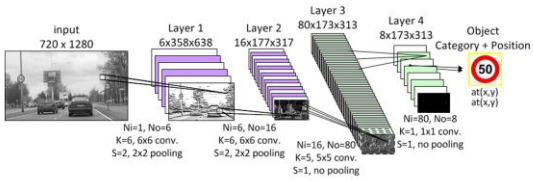
- 3x3 Convolution 20 lines of code
- Neural layer ~400 lines
- Expert programmer
  - 5 hours of coding per layer
  - Impossible for other users

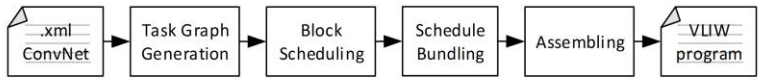


# Custom Compiler



- Abstract from the hardware




| | Write Local | Read Local | Reg Operation | EX VMAC | WB Sgm |
|---|---|---|---|---|---|
| 10 Cycles | | Ld a 0, [b0 w0-1] | Set i3, Shift i0 i1, Shift W | MAC,w6, c2 i0 | |
| | | Ld ab 4 5, [c0,i0-15] | Shift i0 i1, Shift W | MAC,w7, c2 i1 | |
| | Wr[c3,i0-i7], 4 | Ld b 6, [c0,i16-17] | Set W0 | MAC,w8, c2 i2 | |
| | | Ld a 1, [w2-4] | Set i0 i1, Shift W | Set, b0 | |
| | | Ld ab 7 8, [c1,i0-15] | Set i3, Shift i0 i1, Shift W | MAC,w0, c0 i0 | |
| | Wr[c3,i8-i15], 5 | Ld b 9, [c1,i16-17] | Shift i0 i1, Set W1 | MAC,w1, c0 i1 | |
| | | Ld a 2, [w5-8] | Set i0 i1, Shift W | MAC,w2, c0 i2 | Sigm0, Wrback |
| | | Ld ab 10 11, [c2,i0-15] | Set i3, Shift i0 i1, Shift W | MAC,w3, c1 i0 | Sigm1, Wrback |
| | Wr[c3,i16-i19 - - - -], 6 | Ld b 12, [c2,i16-17] | Shift i0 i1, Set W2 | MAC,w4, c1 i1 | |
| | | NOP | Set i0 i1, Shift W | MAC,w5, c1 i2 | |



Technische Universiteit Eindhoven University of Technology

# What do we gain?



- ~ 20x speedup vs ARM A9
- ~ 1.2x speedup vs embedded GPU
- Ultra-low power 100mW

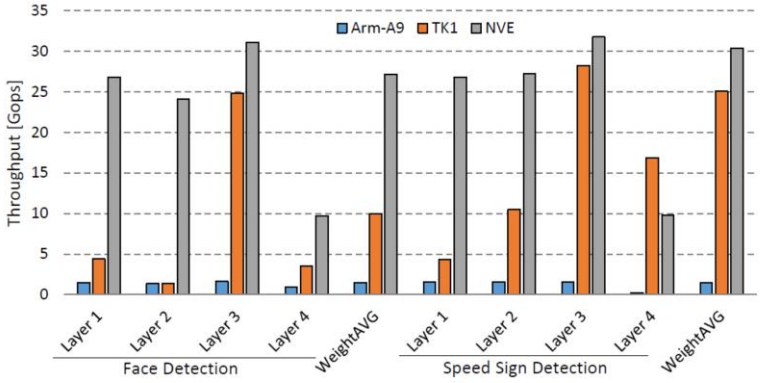
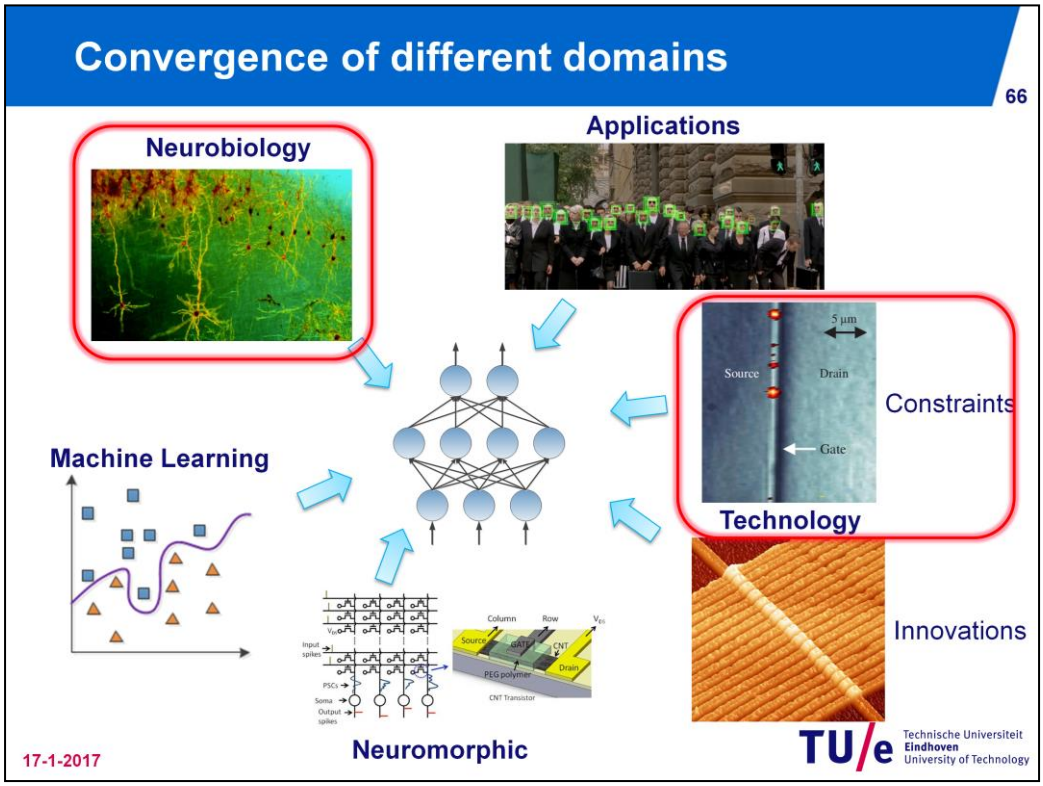



Fig. 8. Throughput comparison Arm-A9, NVidia Jetson TK1, and NVE



TU/e Technische Universiteit Eindhoven University of Technology

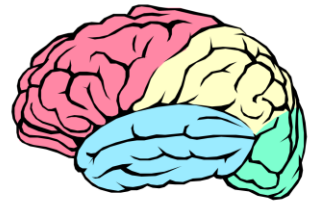



The tech. improvements also create new possibilities for the field of Neurobiology. Every year this domain can simulate bigger neural circuits.

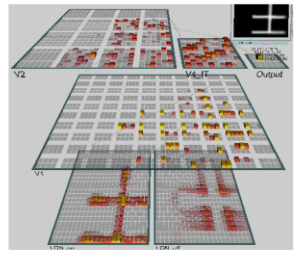
## Beyond ANNs: Biological NNs



- Understand the mind by simulating the brain
  - Model perception
  - Model memory
  - Etc.
- Understand brain diseases
  - Parkinson
  - Alzheimer
  - Etc.
- Software simulators
  - Emergent
  - NEURON



- $10^{11}$ Neurons
- $10^{15}$ Synapses
- 30-400 Hz






TU/e Technische Universiteit Eindhoven University of Technology

Why simulating the brain? Possible with software but this scales very bad. Only small neural circuits possible. Without the communication overhead the brain would require over 30 Peta Flops.

## Can computers do the same?



- **Blue Brain Project**
  - **IBM/EPFL**
  - **Molecular level**
  - **$10^4$ neurons**
  - **$10^3$ cores**

Blue Brain Project

- **Spinnaker**
  - **Integrate & fire**
  - **$10^9$ neurons**
  - **$10^4$ Arm9 cores**

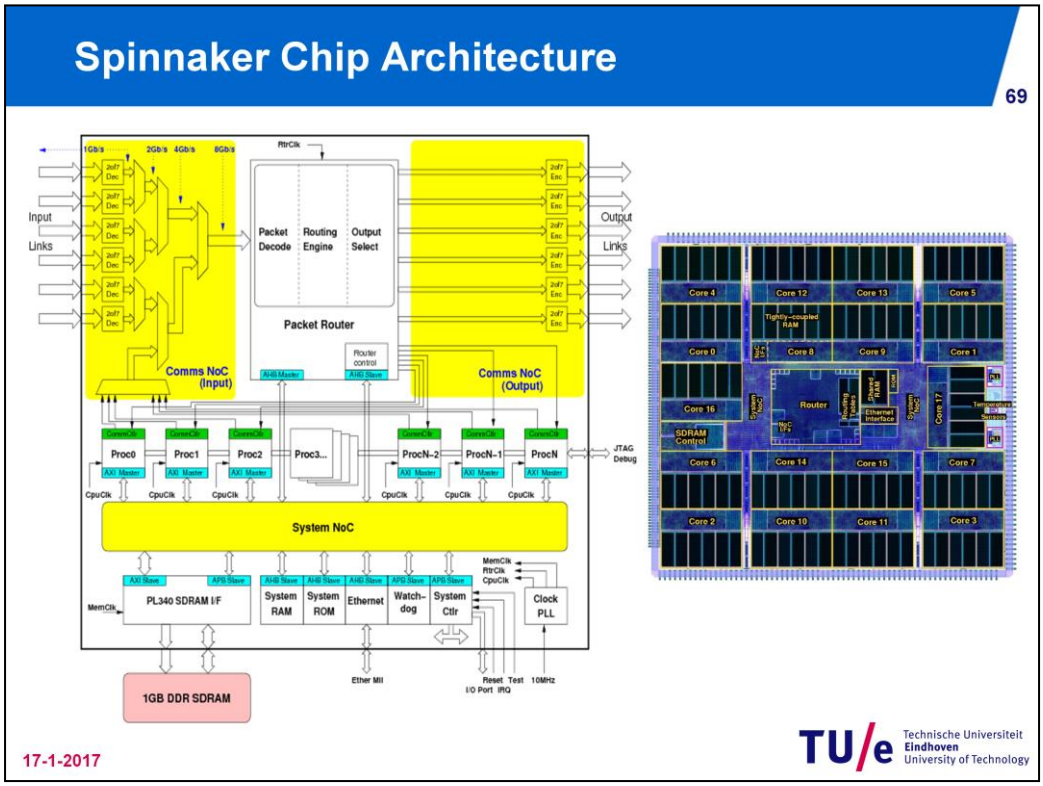
SpiNNaker
a universal Spiking Neural Network architecture



TU/e Technische Universiteit Eindhoven University of Technology

Blue brain project simulates small brain structures on the molecular level on a super computer.

Spinnaker builds a more energy efficient super computer out of many ARM cores. Compared to Blue Brain Spinnaker uses a more abstract Integrate & Fire neuron model.



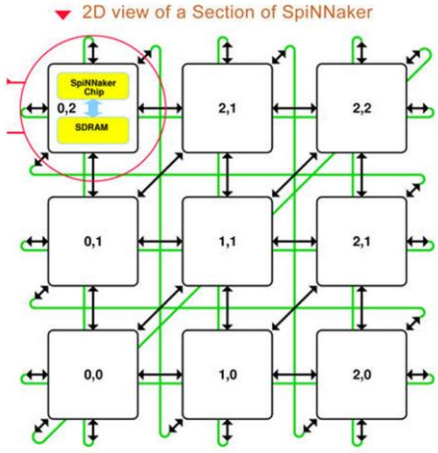

Take a look at the Spinnaker project:
http://apt.cs.man.ac.uk/projects/SpiNNaker/project/

18 Arm9 cores on a chip with a dedicated NoC and Packet router to go off chip.

## Spinnaker interconnect



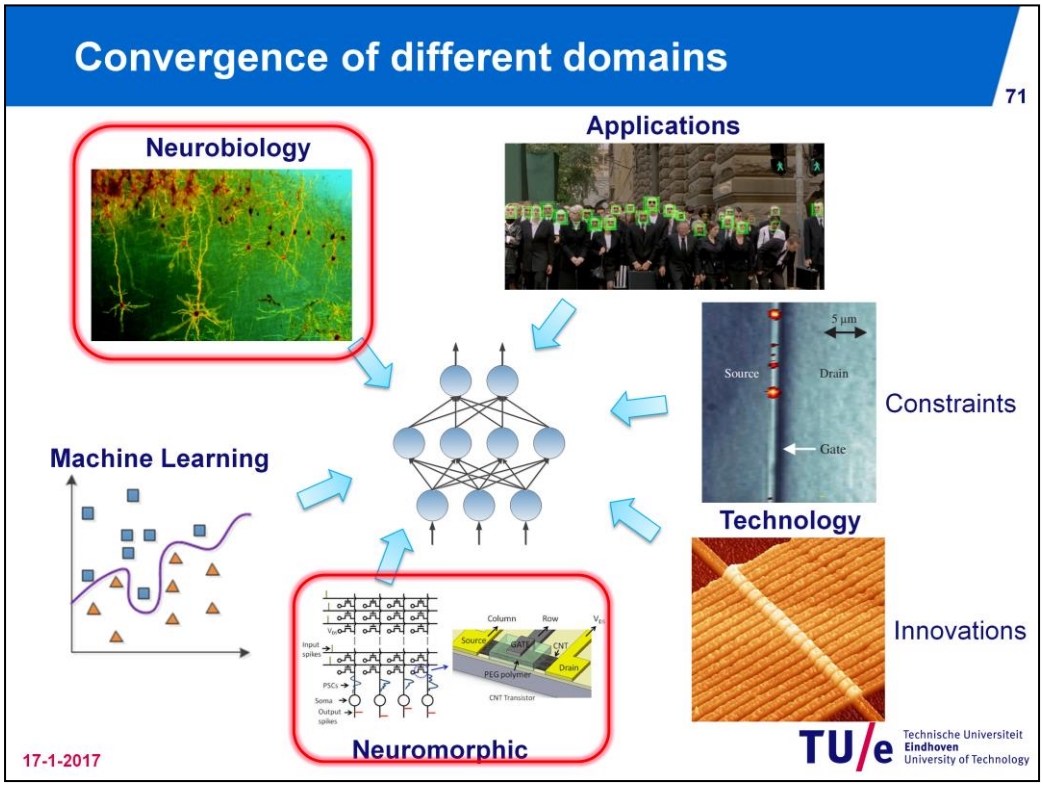
- Connection Hierarchy
- Group neurons to reduce inter-chip communication
- 128 MB SDRam
- Small Packets 40-70 bit
- Routing tables






TU/e Technische Universiteit Eindhoven University of Technology

Neurons that share a lot of interconnections are grouped on a chip with the local 128MB SDRAM. This minimizes the packet traffic over the off-chip interconnect.

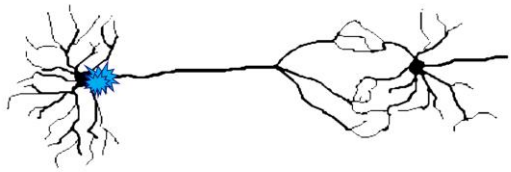



Spinnaker is still a multiprocessor network of general purpose cores. This is flexible but also less efficient compared to dedicated circuits

## Size



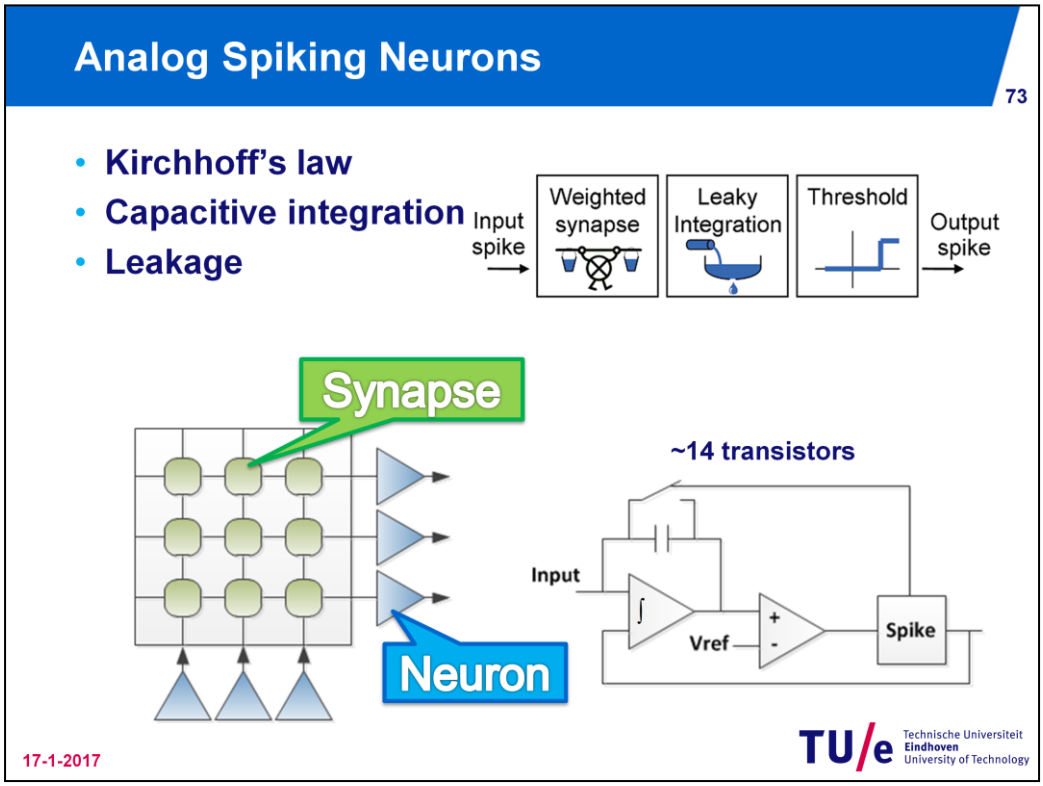
- Digital CMOS
  - Technology available
  - Implementation of useful accelerators
  - Not dense enough for largest bio-inspired networks
- Analog
  - Much more dense implementation
- Recall Biological Neuron





TU/e Technische Universiteit Eindhoven University of Technology

Biological Neuron communicates with spikes. Instead of only computing with the pike rates also the arrival time can trigger actions.

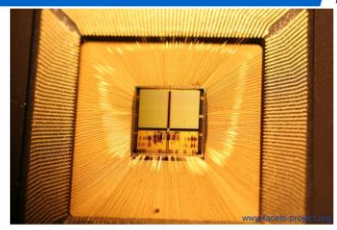


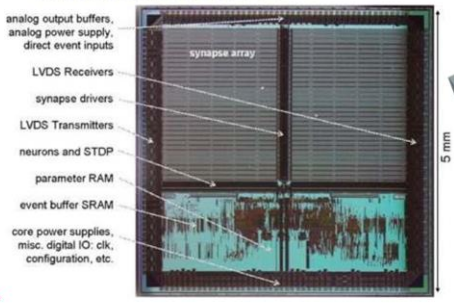

A model of a leaky Intergrate and Fire neuron. This neuron only requires ~14 transistors. Most area is now consumed by the synapses. Storing the weight in a capacitance consumes much area. Read about real implementations in: Antoine Joubert, Bilel Belhadj, Olivier Temam, Rodolphe Heliot: *Hardware Spiking Neurons Design: Analog or Digital?*, IEEE International Joint Conference on Neural Networks (IJCNN), June 2012.

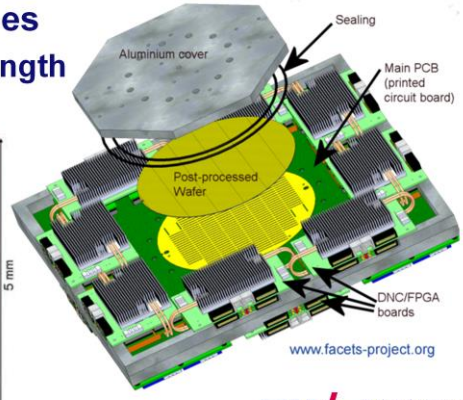
## Architecture Facets Project

74

- Facets
  - Integrate & Fire
  - 250000 neurons wafer
  - 60 million synapses
- Most area used for synapses
  - Storage of connection strength
  - Interconnect 2-D



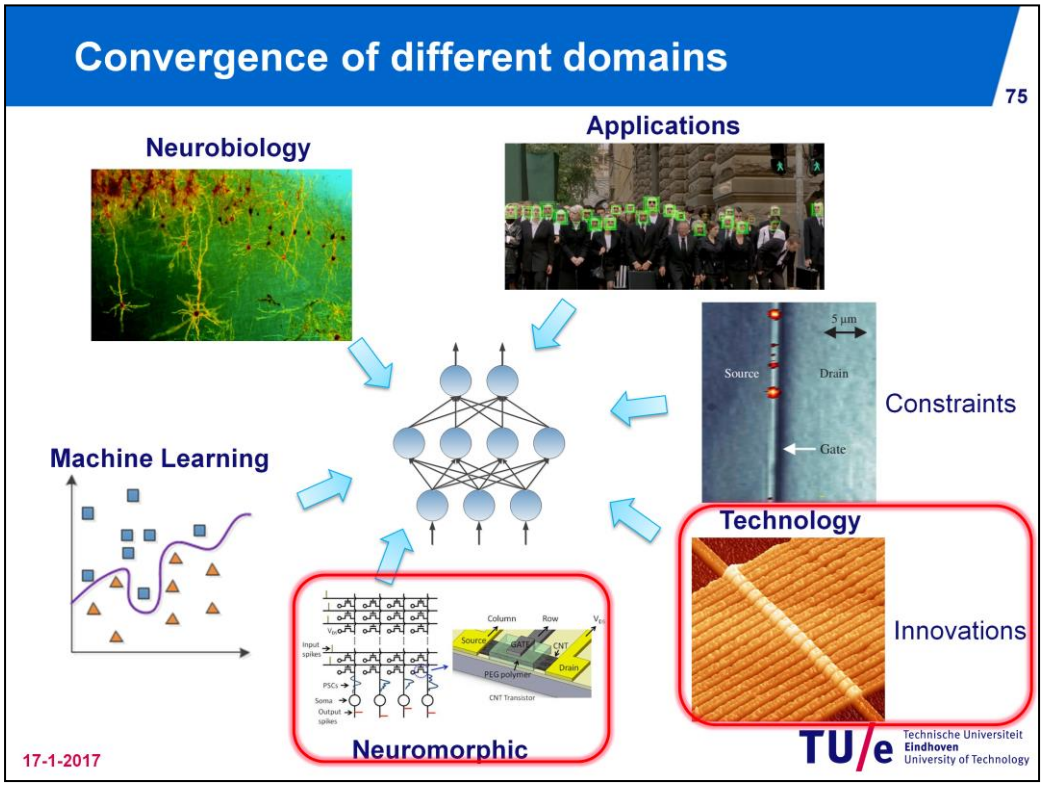



17-1-2017

TU/e Technische Universiteit Eindhoven University of Technology

Wafer scale integration of Integrate and Fire neuron models. See: http://facets.kip.uni-heidelberg.de/public/index.html




New technology innovations that open new possibilities for neural hardware.

## Synapses as Memristors Intel (2012)

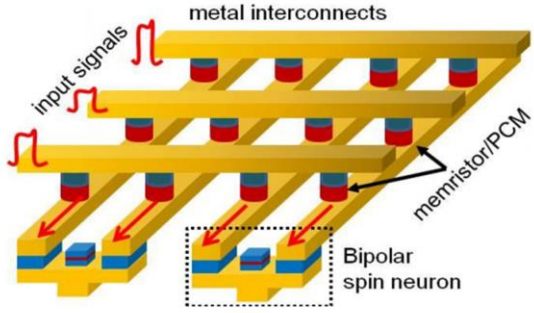


**Proposal For Neuromorphic Hardware Using Spin Devices**

[1]Mrigank Sharad, [2]Charles Augustine, [1]Georgios Panagopoulos, [1]Kaushik Roy
[1]Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Circuit Research Lab, Intel labs, Intel Corporation, Hillsboro, OR, US
msharad@.purdue.edu

**Abstract: We present a design-scheme for ultra-low power** Rest of the paper is organized as follows. Section 2 introduces

- **Memristor can be used as switch**
- **Also analog storage of memristance**





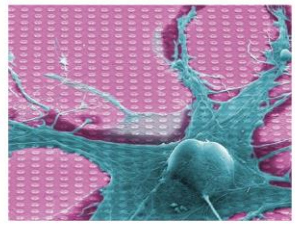

TU/e Technische Universiteit Eindhoven University of Technology

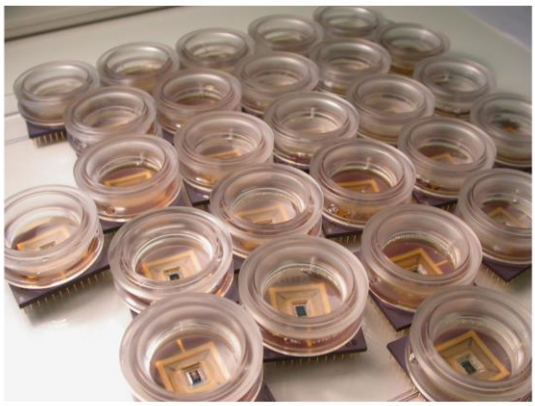
The memristior developed by HP (2008) looks very promising as a basic element for the implementation of synapses. Recently Intel has published an interesting paper about this technology with a crossbar synapse array. Read the paper for more information.

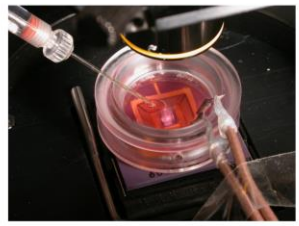
## Beyond Silicon



- Infineon NeuroChip (2003)
- Directly uses biological networks
- Difficult to connect to other devices



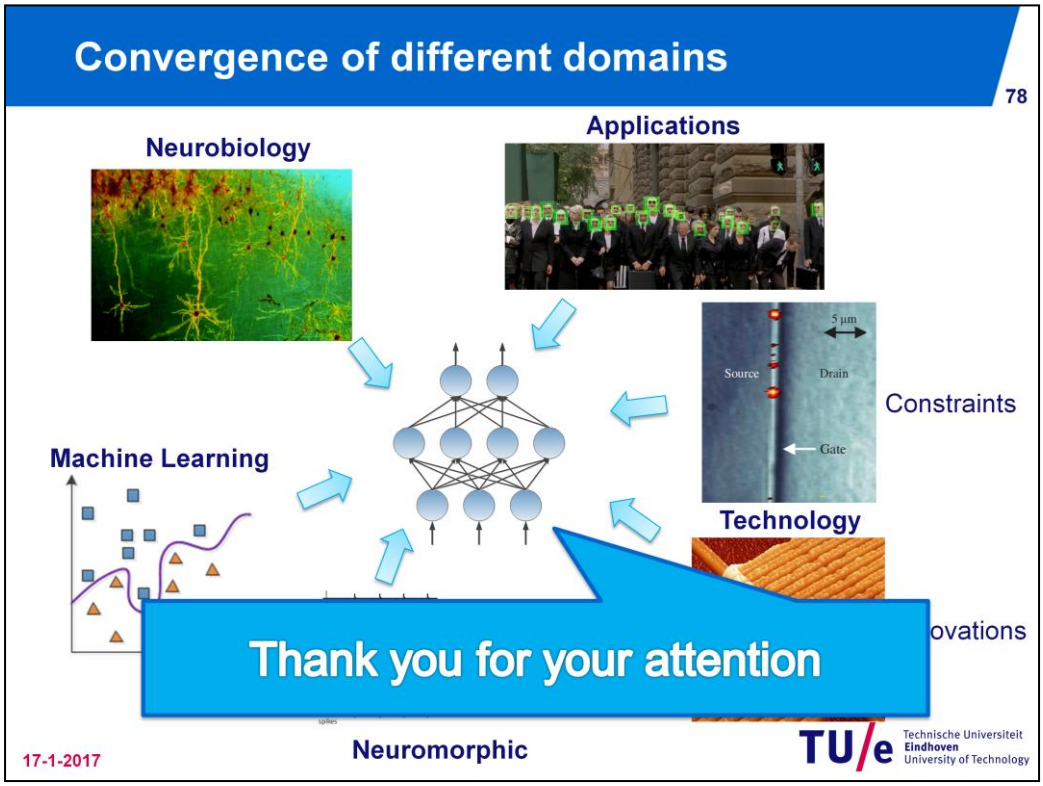






TU/e Technische Universiteit Eindhoven University of Technology

Growing organic chips, can be very cheap. But it is difficult to read out the signals form the living neurons. The neurons on these chips are used for experiments instead of a commercial product. This project was one of the first, many others have followed by now.




This was a broad overview of the field of neuro computing. It shows many promising concepts of neural architectures. For many domains this is only a short summary of the topic. For example Machine Learning has complete courses to understand the concepts. The chance is quite high that you will encounter neural networks in your EE/ES career. This is mainly due to the nice properties of neural networks; (learning, flexible, fault tolerant, and parallel).