Analog Crossbar Arrays – Future Neuromorphic Workhorses for Neural Networks

Tutorial

Roger Dangel, PhD
Neuromorphic Devices and Systems Group
IBM Zurich GmbH

June 24, 2019
IBM Zurich: Neuromorphic Devices & Systems Group

Group leader
Bert Offrein

Research Staff Members (PhDs)

Post-Docs

Engineers

Pre-Docs
Introduction: ▪ Experiment: Human brain against computer
   ▪ Conceptional comparison: Brain vs. computer
   ▪ What makes the human brain so outstanding? – Can we mimic it?

Part 1: ▪ Neuromorphic computing – what is it?
   ▪ Neuromorphic tasks in AI
   ▪ Anatomy of computational “heavy” workloads – the actual problem
   ▪ Computational challenge: Matrix-vector multiplications
   ▪ Current and future AI acceleration hardware

Part 2: ▪ From brain-like to Deep Neural Networks (DNNs)
   ▪ Training of DNN with backpropagation algorithm - Mathematical background
   ▪ Status of today’s Deep Neural Network processing

Part 3: ▪ Analog electrical crossbar array vs. DNN
   ▪ Synaptic weight processing operations
   ▪ Targeted device properties for analog electrical crossbar arrays

Part 4: ▪ Memristive devices for synaptic weight implementation
   ▪ Examples: Resistive Random Access Memory (ReRAM)
     Phase Change Memory (PCM)
     Ferroelectric Tunneling Junctions (FTJ)

Summary
Keywords of this Tutorial

- Neuromorphic computing
- Human brain
- (Non) von-Neumann architecture
- Analog vs. digital processing
- Deep Neural Networks
- Matrix-vector multiplication
- Backpropagation algorithm
- CPU / GPU / FPGA / ASIC
- Accelerators
- Training / Inference
- Memristive devices
- Non-volatile memory
- Crossbar arrays
- Synaptic weights
- Multiply & accumulate
- PCM / ReRAM / FTJ
- Deep Neural Networks
- Human brain
- (Non) von-Neumann architecture
- Analog vs. digital processing
- Matrix-vector multiplication
- Backpropagation algorithm
- CPU / GPU / FPGA / ASIC
- Accelerators
- Training / Inference
- Memristive devices
- Non-volatile memory
- Crossbar arrays
- Synaptic weights
- Multiply & accumulate
- PCM / ReRAM / FTJ
Experiment: "Human Brain against Computer"

Task 1: Mathematics

Brain
\[ \sqrt{2} = ? \]

Computer
\[ \sqrt{2} = ? \]
Experiment: “Human Brain against Computer”

Task 1: Mathematics

\[
\sqrt{2} \approx \frac{9}{6} \quad \frac{357}{240} \quad 1.4142
\]

Roger Dangel: Result obtained in \(\approx 10\) min:

\[
\sqrt{2} \approx 1.4142
\]

Computer: Result obtained in \(< 1\) sec:

\[1.414213562373095048001688724209698078569671875376948073176679737990732478462107038850387534327641572735013846230912297024924248360585073721264412149709993583141322265927505592755799505011527820605714701095599716059727453459666201472857416864088919660955232923046430871432145083367626036709525140708968725339654633180829646020615258352395054745750287759961729835675220373218770113543765034084984716983689997099904816593954027759315654247623068452837519221668570443111566948722732328596248613649771542183342042856866014382470714358648715545951472267292612405059617316380941082106604523610269654756...\]
Experiment: “Human Brain against Computer”

Task 2: Image recognition

What does the image show?

Image to be recognized

What does the image show?
Experiment: “Human Brain against Computer”

**Task 1: Mathematics**

$$\sqrt{2} \approx 1.4142$$

Roger Dangel: Result obtained in \approx 10 min:

Computer: Result obtained in $<< 1$ sec:

1.414213562373095048801688724209698078569
67187537694807317667973799073247846210703869707
53437641572735013846230912297024924836055850737564401
214970999358314313222665927555975579995050115273206371420160
95599718050702745345988920147285174186098919860655232023044030977421450
839762003627995251407899687253390546331804826646206162583323956474576028759651
73893567620315331867011543746634084894716036889976999048150306440277930316454278023068
4029369186215667846311199668713013015669872373258608268612494877141283429202558
65601468272071435854874155458541472257229251245656661731636941362196045268162596547595.

**Task 2: Image recognition**

Roger Dangel: Result obtained in $< 1$ sec:

Group of ring-tailed lemurs eating fruits

Computer: not able to solve this task (yet)
Experiment: “Human Brain against Computer”

Task 1: Mathematics

\[ \sqrt{2} \approx 1.4142 \]

Roger Dangel: Result obtained in \( \approx 10 \) min:

Brain

Task 2: Image recognition

Roger Dangel: Result obtained in < 1 sec:

Group of ring-tailed lemurs eating fruits

Computer: not able to solve this task (yet)
What makes the Human Brain so Outstanding?

- **Power efficiency:** human brain $\approx 20$ Watts / supercomputers $\approx$ up to MWatts
- Brain recognizes patterns and images / can deduce facts from raw (noisy) data

**Brain at neural network level**

- Human brain: $\approx 100$ billions nerve cells (= neurons)
- Each neuron receives signals from 1’000 – 10’000 other neurons via synapses $\rightarrow$ massive connectivity
- Signals transmitted by synapses are adjustable: $\rightarrow$ “synaptic weight”

**Brain at neuron level**

- Signaling between neurons: Spikes, spike trains
- Neuron activation: “Integrate and Fire”
- Learning: Adjustment of the synaptic weights
  Spike Timing Dependent Plasticity:
  “Neurons that fire together wire together”

http://biomedicalengineering.yolasite.com/neurons.php

http://www.sciencephoto.com/dennis-kunkel-microscopy-collection

Conceptional Comparison: “Human Brain vs. Computer”

Human brain  Different (complementary) abilities  Today’s Computer

- Nerve cells (neurons) are processing units
- Analog operation
- Distributed processor and memory
- Massively, massively parallel processing
- Slow information processing
- Redundancy and fault-tolerance properties
- ....

- Transistors are processing units
- Digital operation
- Centralized processor and memory
- (Mostly) serial processing
- Very fast information processing
- Reliable and precise
- ....

Question: Can we mimic the human brain to exploit its superiority in certain applications?
Outlook

Part 1:
- Neuromorphic computing – what is it?
- Neuromorphic tasks in AI
- Anatomy of computational “heavy” workloads – the actual problem
- Computational challenge: Matrix-vector multiplications
- Current and future AI acceleration hardware
Neuromorphic Computing – What is it?

Ethymological:  
“neuro”  $\Leftrightarrow$ related to nerves or nervous system  
“morphic”  $\Leftrightarrow$ having form or structure of...

Definition:  
Neuromorphic computing is a brain-inspired signal processing technology that tries to mimic the neuro-biological architecture of the brain and its functions.  
As interdisciplinary technology, it involves
- biological,
- physical,
- mathematical,
- computer science,
- and electronic engineering concepts  
  to design and realize new artificial neural network systems.

Neuromorphic Tasks in AI

Neuromorphic challenges in AI are tasks which normally require human “intellect”, e.g.:

- Memorizing complex information
- Deducing facts from raw (unstructured) Data
- Making recommendations and decisions

in the presence of uncertainty and ambiguity

Elevator Control:

Deduce Facts from Data:
- floor-to/from button

Make Decisions:
- stop at floor(s)

Memorizing Information:
- remember pending call

Detect or Extract:

- Emotions
- Anomalies
- Image Classes
- Segments

Act:

- Recommendations
- Autonomous Decisions

R. Dangel - Analog Crossbar Arrays – Future Neuromorphic Workhorses for Neural Networks
Anatomy of “Heavy” Computational Workloads – The Actual Problem

Scientific Workloads

- PDEs
  - \[ \frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_2}{\partial x_2^2} + \frac{\partial^2 u_1}{\partial x_1 \partial x_2} + \frac{\partial^2 u_2}{\partial x_2 \partial x_1} + \theta_1 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_1}{\partial x_1^2} + \theta_2 \frac{\partial u_2}{\partial x_2} + \frac{\partial^2 u_2}{\partial x_2^2} + \theta_3 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2^2} + \theta_4 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2^2} + \theta_5 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2^2} + \theta_6 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2^2} + \theta_7 \frac{\partial u_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2^2} \]

- Backpropagation
  - \[ f_1 = 0 \]
  - \[ f_2 = 0 \]
  - \[ f_3 = 0 \]

AI / Machine Learning

- Time-Series Predictions
- Graph Analytics

Drug Discovery

Weather/Climate

Clustering Algorithms

Image Classification

(electro) Chemistry

Drug Discovery

Weather/Climate

Clustering Algorithms

Image Classification
Computational Challenge: Matrix-Vector Multiplications

Matrix-vector multiplications of the form

\[ Wx = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & \cdots & w_{0,N} \\ w_{1,0} & w_{1,1} & w_{1,2} & \cdots & w_{1,N} \\ w_{2,0} & w_{2,1} & w_{2,2} & \cdots & w_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{M,0} & w_{M,1} & w_{M,2} & \cdots & w_{M,N} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} w_{0,i} x_i \\ \sum_{i=0}^{N} w_{1,i} x_i \\ \sum_{i=0}^{N} w_{2,i} x_i \\ \vdots \\ \sum_{i=0}^{N} w_{M,i} x_i \end{bmatrix} \]

are common to the mentioned workloads and dominate the computation time and energy consumption.

Matrix-vector multiplications are “computationally expensive”!

Our mission: Develop dedicated hardware (→ Analog Crossbar Arrays) which enables efficient analog implementation of matrix-vector multiplications and therefore acceleration of Deep Neural Network Learning.
## Current and Future AI Acceleration Hardware

<table>
<thead>
<tr>
<th>Central Processing Unit (CPU)</th>
<th>Graphics Processing Unit (GPU)</th>
<th>Field-Programmable Gate-Array (FPGA)</th>
<th>Application-Specific Integrated Circuit (ASIC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs were originally designed for general computing workloads</td>
<td>GPUs were originally developed for manipulation of images which relies on similar mathematical basis than neural networks</td>
<td>FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects</td>
<td>ASICs are application-specifically designed hardware</td>
</tr>
<tr>
<td>GPUs operate on vectors of data in parallel</td>
<td>GPUs are effective at processing same set of operations in parallel (single instruction, multiple data (SIMD))</td>
<td>FPGA are reconfigurable, what makes evolution of hardware, framework and software easier</td>
<td>ASICs employ special strategies, e.g. optimized memory use or use of lower precision arithmetics</td>
</tr>
<tr>
<td>GPUs are effective at processing same set of operations in parallel</td>
<td>GPUs have well-defined instruction-set and fixed data width</td>
<td>FPGAs are effective at processing same or different set of operations in parallel (multiple instructions, multiple data (MIMD))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FPGAs do not have predefined instruction-set or fixed data width.</td>
<td></td>
</tr>
</tbody>
</table>

### Co-processors

- **CPUs**: Current Workhorse
- **GPUs**: Limited spread
- **FPGAs**: Under investigation

### Graphics Processing Unit (GPU)
- **CPUs** were originally designed for general computing workloads.
- **GPUs** were originally developed for manipulation of images which relies on similar mathematical basis than neural networks.
- **GPUs** operate on vectors of data in parallel.
- **GPUs** are effective at processing same set of operations in parallel (single instruction, multiple data (SIMD)).
- **GPUs** have well-defined instruction-set and fixed data width.

### Field-Programmable Gate-Array (FPGA)
- **FPGAs** contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects.
- **FPGA** are reconfigurable, what makes evolution of hardware, framework and software easier.
- **FPGAs** are effective at processing same or different set of operations in parallel (multiple instructions, multiple data (MIMD)).
- **FPGAs** do not have predefined instruction-set or fixed data width.

### Application-Specific Integrated Circuit (ASIC)
- **ASICs** are application-specifically designed hardware.
- **ASICs** employ special strategies, e.g. optimized memory use or use of lower precision arithmetics.

---

**Resistive Processing Unit (RPU)**

Based on Analog Crossbar Arrays
Part 2:  ▪ From brain-like to Deep Neural Networks (DNNs)
  ▪ Training of DNN with backpropagation algorithm – Mathematical background
  ▪ Status of today’s Deep Neural Network processing
From Brain to Brain-like Neural Network

- Omni-directional signal flow
- A-synchronous pulse signals
- Information encoded in signal timing

Difficult to implement efficiently on standard computer hardware

http://www.sciencephoto.com/dennis-kunkel-microscopy-collection
Artificial Neural Network

Brain-like neural network  Simplified model  Artificial Neural Network (ANN)

- ANNs are neuromorphic computing models, which mimic the brain in a simplified way.
- ANNs are composed of multiple nodes (= artificial neurons) which can be arranged in special configurations.
- The first developed, easiest and most common ANN is the: Feed-forward Deep Neural Network (DNN)

DNN better fit to standard hardware
- Feed-forward sequential processing
- Information encoded in signal amplitude
- Neuron activation: Weighted sum + Threshold
- Training with “Backpropagation algorithm”
Operation Phases of Deep Neural Network (DNN)

Phase 1: Training/Learning
- Very large number of known samples
- Forward propagation
- Answer with error

Phase 2: Inference/Use
- Smaller number of unknown samples
- Only forward propagation
- Correct answer

Computational speed and efficiency are extremely important because training of Deep Neural Networks can range from days to weeks (even with high-performance computers)!
DNN Training by Backpropagation Algorithm

Generic scheme for iterative error minimization by adjusting the synaptic weights

Use of next test sample \( \{x, y\} \)

\[ \sum \text{Weighted sum} \quad \text{Activation function} \]

Step 1: From input layer to hidden layer

Step 2: From hidden layer to output layer

\[ \sum \text{Weighted sum} \quad \text{Activation function} \]

Calculation of error function \( \varepsilon \):

\[ \varepsilon = \frac{1}{2} \sum_{i=1}^{M} [y_i - y_{\text{target},i}]^2 \]

Question 1:
How much differs the intermediate result vector \( \tilde{y} \) from the target vector \( y_{\text{target}} \)?

Question 2:
How do we have to adjust the synaptic weights to reduce the error?
DNN Training by Backpropagation Algorithm

Components:
- Layers of neurons
- Synaptic interconnections

Mathematical operations:
- : Signal vector
- $W_n$: Synaptic weight matrix $[W_n]$
- $\sigma$: Per-element neural (non-linear) activation function (sigmoid):

Neural net as chain of vector operations:

$W_1x = \begin{pmatrix} w_{1,0,0} & w_{1,0,1} & \cdots & w_{1,0,N} \\ w_{1,1,0} & w_{1,1,1} & \cdots & w_{1,1,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,M,0} & w_{1,M,1} & \cdots & w_{1,M,N} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{N} w_{1,0,i} x_i \\ \sum_{i=0}^{N} w_{1,1,i} x_i \\ \vdots \\ \sum_{i=0}^{N} w_{1,M,i} x_i \end{pmatrix}$
DNN Training by Backpropagation Algorithm

For many training case inputs $\mathbf{X}$ with target response $y_{\text{target}}$:

1. **Forward Propagate**: Input $\mathbf{X} \rightarrow$ Response $\mathbf{y}$
   - Store neuron activation patterns $\mathbf{x}_i$ for later use

2. **Determine output error $\mathcal{E}$**: 
   \[
   \mathcal{E} = \frac{1}{2} \sum_i \left[ y_i - y_{\text{target}_i} \right]^2
   \]

3. **Backward Propagate**: Which neuron inputs have strongest influence on $\mathcal{E}$?
   - Error gradient vectors $\delta_j$

4. **Adjust weights** that were active ($\propto \mathbf{x}$), proportionally to their influence on error $\mathcal{E}$ ($\propto \delta$):
   \[
   \Delta \mathbf{W} = -\eta \mathbf{x} \otimes \delta \quad \Delta w_{ij} = -\eta x_i \delta_j
   \]
Status of Today’s Deep Neural Network Processing

- Processing dominated by large matrix operations
  - Forward propagation: $W$
  - Backward propagation: $W^T$
  - Weight update: $\Delta W$
  - Scale $\propto N^2$
  - Neurons/layer

- Large training datasets: Thousands of training cases

- Inefficient on standard Von-Neumann architecture systems:
  - (Mostly) serial processing
  - Low computation to IO ratio
  - Memory bottleneck

Need for faster and more efficient DNN processing

Borrow some concepts from the brain:
- Analog signal processing
- Fully parallel processing
- Tight integration of processing and memory

Analog Crossbar Arrays
Part 3:  ◀ Analog electrical crossbar array vs. DNN  
  ▶ Synaptic weight processing operations
Analog Electrical Crossbar Array

**Implementation**

- Conductance: \( G = \frac{1}{R} \)
- Current: \( I = \frac{V}{R} = G \cdot V \)

\[
I_m = \sum_{i=0}^{N} \frac{V_i}{R_{m,i}} = \sum_{i=0}^{N} G_{m,i} \cdot V_i
\]

“Multiply and Accumulate”

\[
\begin{bmatrix}
    I_0 \\
    I_1 \\
    I_2 \\
    \vdots \\
    I_M
\end{bmatrix} =
\begin{bmatrix}
    G_{0,0} & G_{0,1} & G_{0,2} & \cdots & G_{0,N} \\
    G_{1,0} & G_{1,1} & G_{1,2} & \cdots & G_{1,N} \\
    G_{2,0} & G_{2,1} & G_{2,2} & \cdots & G_{2,N} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    G_{M,0} & G_{M,1} & G_{M,2} & \cdots & G_{M,N}
\end{bmatrix} \cdot
\begin{bmatrix}
    V_0 \\
    V_1 \\
    V_2 \\
    \vdots \\
    V_N
\end{bmatrix} =
\begin{bmatrix}
    \sum_{i=0}^{N} G_{0,i} \cdot V_i \\
    \sum_{i=0}^{N} G_{1,i} \cdot V_i \\
    \sum_{i=0}^{N} G_{2,i} \cdot V_i \\
    \vdots \\
    \sum_{i=0}^{N} G_{M,i} \cdot V_i
\end{bmatrix}
\]

- Tunable resistor \( R_{m,i} \)
- Tunable conductance \( G_{m,i} \)

- Cross-point
- Synaptic weight

Feedforward DNN w/ fully connected neural layers

- Input neurons
- Hidden neurons
- Output nodes
- Synaptic weights

R. Dangel - Analog Crossbar Arrays – Future Neuromorphic Workhorses for Neural Networks
Synaptic Weight Processing Operations

**Forward propagation**

\[ \hat{x} \rightarrow [W] \]

**Backward propagation**

\[ [W]^T \delta \rightarrow \delta \]

**Synaptic weight update**

\[ \Delta w_{ij} = -\eta x_i \delta_j \]

**Challenge**

Update must be proportional to signals on rows (\(\propto x_i\)) and on columns (\(\propto \delta_j\))

- **Symmetric** increase and decrease of weight
- **Analog behavior**: > 100 levels preferred (ca. 8 bit)

**Input vector**

\(\hat{x}\)

**Weight matrix**

\([W]\)

**Transposed weight matrix**

\([W]^T\)
Part 4:
- Targeted device properties for analog electrical crossbar arrays
- Memristive devices for synaptic weight implementation
- Examples: Resistive Random Access Memory (ReRAM)
  Phase Change Memory (PCM)
  Ferroelectric Tunneling Junctions (FTJ)
Our Dream-Device:
- CMOS compatibility
- Low voltage operation
- Small device footprint
- Very short (re-)set time
- Long retention time (< -> NVM)
- Low drift
- High dynamic range
- Large resistance range
  (high-resistance \( \rightarrow \) low power)
- Reproducibility, low variability
- (Some) linearity & symmetry

Operation:
- Current \( I < \mu A \)
- Voltage \( V < 1V \)

“Programming” Resistance:
- Representative, generic characteristic

Programming Scheme:
- Pulse Encoding (Incremental)

Which one works (best)?
\( \rightarrow \) linear & symmetric

Targeted Device Properties for Analog Electrical Crossbar Arrays

Memristive Devices for Synaptic Weight Implementation

ReRAM

 PCM

 FTJ

Others:
- FeRAM (Ferro-Electric RAM)
- MRAM (Magnetic RAM)
- ECRAM (Electro-Chemical RAM)

Conductance $G = 1/R$

Potentiation
Depression

Potentiation
Depression

Potentiation
Depression
Resistive Random Access Memory (ReRAM)

- **ReRAM** (also called RRAM) is one type of memristive non-volatile memory that works by changing the resistance across a dielectric solid-state material. Sufficiently high voltage $V_{\text{forming}}$ makes insulating dielectric material conductive.

- Filament-like or homogeneous current conduction path(s) induced by defects (oxygen-vacancies).

- Switching between **Low Resistance State (LRS)** and **High Resistance State (HRS)** by applying suitable voltages $-V_{\text{reset}}$ and $+V_{\text{set}}$.

- The oxygen vacancies act as charge carriers, meaning that the depleted area has a much lower resistance.

**ReRAM phases:**
- **FORMING**: creation of conducting filament in dielectric material between electrodes
- **RESET** (LRS $\rightarrow$ HRS): partial dissolution of filament
- **SET** (HRS $\rightarrow$ LRS): recreation of filament
- **STORAGE**: retain last resistance
**Challenge:**

With only one (or a few) localized conductive filaments, switching would be quite abrupt (between 2 resistance states: LRS and HRS).

However, for use of ReRAM in analog crossbar arrays, **gradual tuning of resistance with many intermediate states is required**.

Use of specifically engineered oxides with suitable oxygen intercalation(*) properties as electrodes.

**Volumetric changes of conductive filament(s)** (i.e., in lateral dimension)

---

Intercalation: In chemistry, **intercalation** is the reversible inclusion or insertion of molecules (or ions) into materials... (Wikipedia)
Phase Change Memory (PCM)

- PCM (also called PCRAM) is another memristive non-volatile memory
- PCM shows **amorphous** and **crystalline** phase
- Rapid and repeated switching between two phases possible
- Switching typically induced by optical or electrical heating
- Physical properties vary significantly between phases: crystalline phase → **Low Resistance State (LRS)**
  amorphous phase → **High Resistance State (HRS)**
- Ratio of electrical resistances $R_{\text{LRS}} : R_{\text{HRS}} = 1 : 100$ to $1 : 1000$
- Many phase change materials are **chalcogenides**, most studied and utilized: $\text{Ge}_2\text{Sb}_2\text{Te}_5$ (GST)

**Two-level-cell PCM**
- only two states
- commercially available as Storage Class Memory (SCM)

**Multi-level-cell PCM**
- many intermediate states
- under development for emerging analog crossbar arrays

---

Ferroelectric materials are dielectrics that exhibit a macroscopic electrical polarization $P$, even in absence of an external electric field ($E = 0 \rightarrow P = \pm P_{\text{remanent}}$).

By applying an electric field, the macroscopic polarization state of ferroelectric material can be gradually tuned (→ ferroel. hysteresis curve) because of polarization switching of individual domains in the material from $\uparrow$ to $\downarrow$ or vice versa.

### Ferroelectric material with boundary conditions:

- **Ferroelectric Tunneling Junction** is based on a few nm thick ferroelectric barrier layer sandwiched between two different electrodes (typically metal / semiconductor).

  Gradual polarization state tuning possible by applying suitable positive or negative voltage pulses across FTJ.

---

**Material example: BaTiO$_3$**
- Cubic phase of BaTiO$_3$
- Perovskite crystal
- Off-center-position of Ti$^{4+}$
  - Ferroelectric behavior

---

Ferroelectric Tunneling Junction (FTJ)

**Gradual polarization state tuning** can be achieved by applying suitable positive or negative voltage pulses across FTJ.

Many intermediate polarization states can be induced by nucleation and growth of domains with opposite polarization.

Electrical current through FTJ varies with macroscopic polarization state because of the different tunnel widths for the two opposite polarizations states in individual domains.

Electrical resistance of FTJ can be tuned by polarization state. → “Tunneling Electro-Resistance” (TER) with up to $10^4 \times$ variation. FTJ retains last resistance value when power is turned off.

**Dependence of tunneling current and resistance from polarization**

(A) Both situations A & B are for one domain only

(B)

![Diagram showing the dependence of tunneling current and resistance from polarization.](image)

- **Electrode**: Metal electrode
- **Ferroelectric layer**: Domain with opposite polarization
- **Semiconducting electrode**: Electron depletion

**Motion of electron**

- **Smaller tunneling width** → Low resistance state
- **Larger tunneling width** → High resistance state

![Electron](image)
Summary

For the learning (“training”) and use (“inference”) of Artificial Neural Networks, digital (co-)processors (CPUs, GPUs, FPGAs and ASICs) in computer systems based on Von-Neumann architecture are used almost exclusively today. One promising alternative to these energy-hungry digital logic based computer systems is Analog Neuromorphic Computing, where computationally time-consuming and therefore expensive operations are performed by specialized accelerators comprising analog elements with the promise to improve the performance and power efficiency by factors of 1000 to 10,000.

In general, suitable compute elements are programmable analog devices with non-volatile memory capabilities that can be arranged in crossbar arrays to perform various mathematical operations. The main requirements for such emerging “non Von-Neumann” architectures are vector-matrix multiplications and the ability to provide the transposed matrix for learning as well as means to store analog synaptic weights. This mitigates the huge communication overhead for the operands in traditional systems, i.e. avoids the time and energy consuming massive data shuffling between processor and memory.