Efficient and Reliable Deep Learning at Scale

Yiran Chen

Department of Electrical and Computer Engineering
Duke University Center for Computational Evolutionary Intelligence (CEI)
NSF IUCRC For Alternative Sustainable and Intelligence Computing (ASIC)
NSF AI Institute for Edge Computing Leveraging Next-generation Networks (Athena)
Evolution of Computing Power

Dr. Raymond Kurzweil

The Age of Intelligent Machines

Center of Computational Evolutionary Intelligence (CEI)
AI Systems – An Interdisciplinary Field

- Material & Device Innovation
- Interface & Controller Circuit Design
- Engine (Array + Controller)
- Core (Engine + Memory)
- System

Transistor/Device Number

- $10^0$
- $10^{10}$

Computer Architecture & Software Solution Stack

HW/SW Co-Design: Smarter Deployment
Hardware Platforms for DNNs

GPUs
- Fast, but high power consumption (~200W)
- Training DNNs in back-end GPU clusters

FPGAs
- Massively parallel + low-power (~25W) + reconfigurable
- Suitable for latency-sensitive real-time inference job

ASICs
- Fast + energy efficient
- Long development cycle

Novel architectures and emerging devices
- Highly integrated architecture/circuit/device co-design
Von Neumann Bottleneck

- Stalled single-thread performance
- Limited data throughput
- Constrained power efficiency
Memory Wall & Memory Centric Design

Energy consumption for 45 nm CMOS process [Han+, ISCA’16].

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy [pJ]</th>
<th>Relative Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit int ADD</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>32 bit float ADD</td>
<td>0.9</td>
<td>9</td>
</tr>
<tr>
<td>32 bit int MULT</td>
<td>3.1</td>
<td>31</td>
</tr>
<tr>
<td>32 bit float MULT</td>
<td>3.7</td>
<td>37</td>
</tr>
<tr>
<td>32 bit 32KB SRAM</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>32 bit DRAM</td>
<td>0.40</td>
<td>6400</td>
</tr>
</tbody>
</table>

DRAM-based Near Memory Processing System [ISCA’16 A. F.-Farahani, et. al]

- Near Memory Processing
  - Data movement overhead still exists.
- PIM with Conventional Memory
  - Large Area/Power Overhead
  - Only support simple classification algorithms with poor accuracy.

SRAM-based Processing-In-Memory System [JSSC 2017 vol. 52, no. 4, J. Zhang, et. al]
Outline

- Introduction
- ReRAM-based Deep Learning Accelerators
- Hardware Friendly Model Compression
- Distributed System
- Security & Privacy
- Neural Architecture Search for Automated Co-design Workflow
- Conclusions
eNVM – Computing Inspired by Brain

\[ \begin{bmatrix} x_1 & x_2 & \ldots & x_m \end{bmatrix} \]

Natural matrix operation

\[ y_1 = \sum x_i \cdot g_{il} \]

Synapse Network \[\xrightarrow{\text{↔}}\] Memristor Crossbar

High density

\[ I_1 = \sum_{i=1}^{M} g_{il} V_i \]

[J. Joshua Yang et al., Nature Nanotechnol. 3, 429 2008]
Neuro-Chip Development at Duke

August 2015
Spiking, 2520x2520 µm²

February 2016
Spiking, 2520x2520µm²

May 2016
Level, 2847x2471 µm²

October 2016
Spiking, RRAM, 700x700 µm²

February 2017
Spiking controller, 3152x3152 µm²
Measured Results

Vref: Read voltage of BL
- Vref ↑, BL current ↑
- Proportional tuning

Vth: Threshold of capacitor charging/discharging
- Vth ↓, Charging/discharging ↑
- Distorted tuning

[B. Yan, et al., 2019 Symposium on VLSI Technology]
Heterogeneous computing systems embedding with MIXED-signal memristor-based neuromorphic computing accelerators (NCAs).

**Training**

```c
bool RecallBSB(float *vec, float *wm)
{
    /* simulate the synapse network*/
    for(i=0;i<Bsb Size;++i) wx[i] += (wm[i*Bsb Size+j] * vec[j]);
    ...
    /* activation function */
    for(i=0;i<Bsb Size; ++i) wx[i] = ALPHA*wx[i] + LAMDA*vec[i];
    ...
    /* check convergence */
    for(i=0;i<Bsb Size; ++i) if(fabs f(vec[i]) != 1.0) return false;
    return true;
}
```

```c
bool RecallBSB(float *vec)
{
    /* inputs to NCA */
    Send(NCA.id, vec);
    /* outputs from NCA */
    return Receive(NCA.id);
}
```

**Source-to-source translation**

- send each input from register to input buffer associated with specific NCA
- launch the NCA
- put the output from output buffer of NCA to register, here is only one output
- send each input from register to input buffer associated with specific NCA
- launch the NCA
- put the output from output buffer of NCA to register, here is only one output

**RecallBSB**

- `LWR1, $(vec)`
- `MOVRN NCA.id, R1`
- `CONFIG NCA.id, #VAL`
- `MOVNR R1, NCA.id`
- `RET`

**Conventional Processing**

**Neuromorphic Computing Accelerators**

**Frontend:** NCA Software

**Backend:** NCA Hardware

**Normalized Values**

- Speedup
- Energy Efficiency

**[DAC 15, X. Liu et al.]**
Hybrid Parallelism - HyPar

- Hybrid Parallelism for Deep Learning Accelerator Array
  - Too many DLAs
  - Accelerator Wall
  - Instead of designing a better DLA for higher performance, why not use an array of DLAs?

16 Eyeriss DLAs. HyPar compared to model parallelism and data parallelism.

- To minimize total communication
- Data parallelism is suitable for CONV layer
- Model parallelism is suitable for FC layers

Hybrid parallelisms explore a new dimension for large scale DNN training.
Tensor Partitioning - AccPar

• **Tensor Partitioning for Heterogeneous Deep Learning Accelerators**

  • In addition to the model and data parallelism, we introduce a finer granularity – Tensor Partitioning.
Layer-wise partition

- To determine each layer configured by Type I, II or III,
- To minimize the total partitioning cost,
- With a dynamic programing approach.
- Backtrace to determine basic types for each layer.

- To extend the partitioning from two accelerators to multiple accelerators, we use hierarchical partition to recursively portion on two accelerator groups.

Performance on heterogeneous accelerators

- Test with the system configured with TPUv2 and v3
- Type-III partition further boosts performance.

AlexNet partitioning type configuration
Efficient PIM Architecture for Transformer

**ReTransformer**
- Intermediate result writing in MatMul layer.
- Multiple arithmetic operations in Multi-head self-attention block.
- Low crossbar utilization.

**Optimized MatMul:** remove data dependency.
**Hybrid Softmax:** ReRAM-based compare and select block.

---

[ICCAD 20, X. Yang et al.]
ReRAM-based Accelerators - GraphR

Accelerating Graph Processing
- Random accessing
- High memory bandwidth
- Graph Engine (GE) based on ReRAM for SpMV subgraph processing
- Stream-apply for sequential accessing
- GraphR: 8-8 Xbar, 32 Xbars/GE, 64 GEs.

[Image of a graph with vertices and edges labeled with colors and access types]

[GraphR: 8-8 Xbar, 32 Xbars/GE, 64 GEs.]

[HPCA 18, L. Song et al.]
In-situ Spare Graph Analytics Accelerator – GaaS-X

• Adapts the SpMV computation model to different graph algorithms.
• Use Crossbar architecture for MAC and organize nonvolatile memory (NVM) as Content Addressable Memories (CAM).
• Alleviate the overheads of redundant writes, sparse to dense conversions, and redundant computations on the invalid edges.

5375× and 252× average energy saving, 805.44× and 12.29× average speedup over CPU and GPU; 7.74× speedup over GraphR
Outline

- Introduction
- ReRAM-based Deep Learning Accelerators
- Hardware Friendly Model Compression
- Distributed System
- Security & Privacy
- Neural Architecture Search for Automated Co-design Workflow
- Conclusions
Weight Quantization

For Memristor Array

For IBM-TrueNorth Chip

Minimization target:

\[ \hat{E}(w) = E_D(w) + \lambda \cdot E_P(p) \]

For Memristor Array

[ASPDAC 17, Y. Wang, et al.]

Traditional floating-point precision

connectivity probabilities \( p \)

For IBM-TrueNorth Chip

[DAC 16, W. Wen et al.]

Binary/low integer precision

connectivity samples

LeNet accuracy recovery

- QR only: 96.98%
- DQ + QR: 98.66%

\[
\text{l1 norm: } |W_k| \\
\text{l2 norm: } \frac{1}{2} (W_k)^2 \\
\text{Proposed: } \text{sgn}(W_k - Q(W_k)) \cdot (W_k - Q(W_k))
\]
Mixed Precision Quantization

- Low-precision:
  - Less memory consumed
  - Reduce latency
  - Less energy cost
  - Less area cost
  - Lower accuracy

- High-precision:
  - More memory consumed
  - Increase latency
  - More energy cost
  - More area cost
  - Higher accuracy

- Relax the search space
- Lagrangian Multiplier
- Apply bi-level optimization

Our goal

Push the trade-off between accuracy and compression rate into the shaded region of the figure to achieve better compression efficiency.

Our approach ensures quantized model remain a similar accuracy while being compressed up to 30X.

Algorithm 1: Mixed Precision Search

Initialization;

while not converged do
  Update weights $w$ by descending $\nabla_w L_{train}(w, \alpha)$;
  if $L_{valid}(w, \alpha) - \theta \leq 0$ then
    $\lambda \leftarrow 0$;
  else
    $\lambda \leftarrow \text{inf}$;
  end
  Update probability $\alpha$ by descending $\nabla_{\alpha} (L_{valid}(w^*, \alpha) - \theta)$;
end

[AICAS 19, Y Huang et al.]
The Need of Structural Sparsity

- Non-structured sparsity may not bring much speedup on traditional platforms like GPUs
- Structured sparsity is more hardware-friendly
- Structured sparsity can be achieved by having all the parameters within a structured group become zero or nonzero simultaneously

[NeurIPS 16, W. Wei et. al.]
Structured Sparsity in DNN

- Removing filters and channels:

\[
E(W) = E_D(W) + \lambda_n \cdot \sum_{l=1}^{L} \left( \sum_{n_l=1}^{N_l} ||W_{n_l,i,j}^{(l)}||_g \right) + \lambda_c \cdot \sum_{l=1}^{L} \left( \sum_{c_l=1}^{C_l} ||W_{i,j,c_l}^{(l)}||_g \right)
\]

- Modifying filter shape:

\[
E(W) = E_D(W) + \lambda_s \cdot \sum_{l=1}^{L} \left( \sum_{c_l=1}^{C_l} \sum_{m_l=1}^{M_l} \sum_{k_l=1}^{K_l} ||W_{i,j,c_l,m_l,k_l}^{(l)}||_g \right)
\]

- This method can also be applied to FC/LSTM models

---

Table 1: Learning ISS sparsity from scratch in stacked LSTMs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dropout keep ratio</th>
<th>Perplexity (valid/test)</th>
<th>ISS # in (1st, 2nd) LSTM</th>
<th>Weight #</th>
<th>Total time (sec)</th>
<th>Speedup</th>
<th>Multi-add reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>0.35</td>
<td>(92.57, 78.55)</td>
<td>(1340, 1350)</td>
<td>0.68</td>
<td>157.0ms</td>
<td>1.00×</td>
<td>7.09×</td>
</tr>
<tr>
<td>ISS</td>
<td>0.60</td>
<td>(89.09, 78.65)</td>
<td>(273, 315)</td>
<td>21.8M</td>
<td>14.82ms</td>
<td>10.59×</td>
<td>7.49×</td>
</tr>
<tr>
<td>direct</td>
<td>0.50</td>
<td>(90.31, 85.66)</td>
<td>(373, 315)</td>
<td>21.8M</td>
<td>14.82ms</td>
<td>10.59×</td>
<td>7.48×</td>
</tr>
</tbody>
</table>

* Measured with 10 batch size and 30 unrolled steps.

* The reduction of multiplication-add operations in matrix multiplication. Defined as (original Multi-add)/(left Multi-add)

Figure 4: Intrinsic Sparse Structures learned by group Lasso regularization (zoom in for better view). Original weight matrices are plotted, where blue dots are nonzero weights and white ones refer zeros. For better visualization, original matrices are evenly down-sampled by 10 × 10.

[NIPS’16, W. Wei et. al.]

[ICASSP 19, J. Zhang, et. al.]
Compression for Memory Reduction

- Reduce the memory consumption for CNN Models.
  - PCA-based weight decomposition
  - Share basis kernels across input channels
  - Obtain a sparse coefficient matrix

- Reduce the memory footprint with streaming weighted feature map accumulation.

![Diagram](image_url)

- $K$ kernels ($K \ll d$)
- $c_t \times c_{t+1} \times K$ Sparse Coefficients

---

Memory Reduction with PENNI

<table>
<thead>
<tr>
<th>GFLOPs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>0.7</td>
<td>0.71</td>
</tr>
<tr>
<td>0.72</td>
<td>0.73</td>
</tr>
<tr>
<td>0.74</td>
<td>0.75</td>
</tr>
<tr>
<td>0.76</td>
<td>0.77</td>
</tr>
</tbody>
</table>

[ICML 2020, S. Li et. al.]
Outline

- Introduction
- ReRAM-based Deep Learning Accelerators
- Hardware Friendly Model Compression
- Distributed System
- Security & Privacy
- Neural Architecture Search for Automated Co-design Workflow
- Conclusions
Distributed Learning: TernGrad

Distributed Learning

Parameter Server

Worker 1
\[ w_{t+1} \leftarrow w_t - g_t \]

Worker 2
\[ w_{t+1} \leftarrow w_t - g_t \]

Worker N
\[ w_{t+1} \leftarrow w_t - g_t \]

Ternary Gradients

\[ \tilde{g}_t = \text{ternarize}(g_t) = s_t \cdot \text{sign}(g_t) \cdot b_t \]

\[ s_t \equiv \max(\text{abs}(g_t)) \]

\[ P(b_{tk} = 1 | g_t) = \frac{|g_{tk}|}{s_t} \]

\[ P(b_{tk} = 0 | g_t) = 1 - \frac{|g_{tk}|}{s_t} \]

Communication Bottleneck!
Distributed Mobile Training

- **Transmission Reduction**
  - quantization:
    - TernGrad
    - 1-Bit Quantization

- **Computation Reduction**

Lookup table for mini-batch size and computing time

<table>
<thead>
<tr>
<th>Minibatch Size</th>
<th>Node1</th>
<th>Node2</th>
<th>Node3</th>
<th>Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>200 ms</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>300 ms</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>55</td>
<td>75</td>
<td>400 ms</td>
</tr>
</tbody>
</table>

[ICCAD 17, J. Mao et al.]
Distributed Mobile Testing

Convolutional Layer

Fully-Connected Layers:

Average Time (μs)

Memory Usage (KB)

Computing intensive

Memory-intensive

Best Paper Award [DATE 17, J. Mao et al.]
Mobile Video Style Transfer System - MVStylizer

- Optical-flow-based frame interpolation.
- Edge-cloud federated learning scheme is applied to continuously improve the performance.

[Image of a diagram showing the system's workflow and key processes such as optical-flow-based frame interpolation, edge-cloud federated learning, and style transfer.]

---

Center of Computational Evolutionary Intelligence (CEI)
Outline

● Introduction
● ReRAM-based Deep Learning Accelerators
● Hardware Friendly Model Compression
● Distributed System
● Security & Privacy
● Neural Architecture Search for Automated Co-design Workflow
● Conclusions
Privacy-Respecting Data Crowdsourcing Framework

- Learn a feature extractor that can
  - Hide the privacy information from the intermediate representations (privacy)
  - Maximally retaining the original information embedded in the raw data without depending on known primary learning tasks (utility)

[Image of a graph showing the relationship between privacy and utility for different feature extractors (Noisy, DP, Encoder, Hybrid, TIPRDC) for 'gray hair' and 'smiling' attributes in the CelebA dataset.]

Goal 1: Hide privacy information
$y = E_p(E_q(x))$
$L(C) = CE(y, u)$

Goal 2: Retain original information
\[ I^{	ext{eq}} = E_q[-sp(-E_{wp}(x; z, u))] \]
\[ I^{	ext{eq}} = E_q[vf(E_{wp}(x; z, u))] \]
\[ I^{	ext{eq}} = I - I_u \]

[KDD’20, A. Li, et al.]
Heterogenous and Efficient Federated Learning

- **LotteryFL**: improve communication efficiency and achieve personalization under data heterogeneity simultaneously
  - **Personalization under Non-IID**: seek device-specific “Lottery Ticket” subnetworks
  - **Communication-efficient**: communicate only the parameters of the subnetworks

- **MaskFL**: Only communicate the binary mask during the training to further cut the communication and computation cost.
  - Learns a heterogeneous and structured sparse binary mask
  - The binary mask will generate a personalized and structured sparse model

[SEC’21, SenSys’21, A. Li, et al.]
Secure and Robust Federated Learning

- **Soteria**: a defense approach against model inversion attack in FL
  - Reveal the essential cause of leaking private information from the communicated local updates in FL from the perspective of data representations
  - Develop an effective defense against model inversion attack by perturbing data representations

- **FL-WBC**: a defense approach against model poisoning in FL
  - Reveal the reason for the long-lasting effect of a model poisoning attack on the global model.
  - Develop an effective defense against model poisoning attack by perturbing the part of the local training gradients where the attack effect resides in
Outline

- Introduction
- ReRAM-based Deep Learning Accelerators
- Hardware Friendly Model Compression
- Security & Privacy
- Distributed System
- Neural Architecture Search for Automated Co-design Workflow
- Conclusions
Towards Automated Co-design with NAS

- Neural Architecture Search (NAS) is a subfield of AutoML, which aims to automatically explore **efficient** but high-performance DNN models in various tasks.
- NAS has a high search cost (~2000 GPU hours\(^1\)).
- The hardware design space could be integrated into the NAS framework.

---

Search Space

**DAG-based Search Space**

- Use Directed Acyclic Graphs (DAGs) as prototype to construct search space is a novel approach. This facilitates topology-aware neural architecture search to explore efficient DNN models.

**Search space:**
- A complete DAG with 8/6 nodes is used as the topological search space of CNN/RNN architectures.

**Possible operations for CNN:**
- 1x1 convolution or 3x3 Depthwise separable convolution.
- (New) Concat operation combines the results from different layers.

**Possible operations for RNN:**
- ReLU/tanh/sigmoid/identity activation function for each fully-connected layer within the RNN cell.

[AAAI 20, T. Zhang et al.]
Iterative Shrinking

- We do not care about the optimal neural architecture in the whole search space. We only need the architecture search to make concrete progress after each evaluation iteration.
Search Strategy

Differentiable Approach

- Graph-level information is not preserved in traditional gradient-based (differentiable) NAS methods.

- Topological space is discrete, causing inaccuracy and/or reduced representation capability in the mapped space during architecture search.

Cheng et al. “NASGEM: Neural Architecture Search via Graph Embedding Method”
Differentiable Approach

NASGEM can achieve better efficiency score than random graph priors in about 0.15 GPU day (500 iterations).

62% parameter reduction & 20% MAC reduction
Automated Network Depth Discovery

- The process is human designed and heuristic, thus time-consuming and sub-optimal.
- Depth automation can save design time and find better models.

Growth policy
- Option 1: Grow after a shallow net converges
- **Option 2: Grow every $K=3$ epochs**

Initializer of new layers
- Option 1: Network Morphism
- Option 2: Adam pre-training
- Option 3: Uniform noise
- **Option 4: Gaussian noise**

[KDD 20, W. Wen et al.]
Auto Grow

- AutoGrow (with K=3) adapts to 5 datasets and 4 network architectures without any human tuning.
- Significantly improves the accuracy of plain (non-residual) nets.
- Scales efficiently to large-scale problems and outperform human.

Dataset: ImageNet

Human baselines

AutoGrow K=3

AutoGrow K=50

[KDD 20, W. Wen et al.]
Hardware/Software Co-design in AI system

- Algorithm Exploration & Characterization
  - More Computing Units
  - Resource Utilization
  - Parallelism
  - Model Reduction
  - Memory Efficiency
  - Higher Bandwidth

- Hardware Constrains
  - Computation
  - Memory Access
  - Performance Modelling
  - Hardware Implementation

- Accelerator Configurations
Conclusions

- Non-conventional hardware architectures are becoming critical for AI applications.
- Co-design workflow can further push the frontier of the system performance and efficiency.
- Automated co-design workflow will become a necessary component in future AI system designs.
<table>
<thead>
<tr>
<th>Year</th>
<th>Conference</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>NeurIPS</td>
<td>W. Wen  et al.</td>
<td>Learning Structured Sparsity In Deep Neural Networks</td>
</tr>
<tr>
<td>2016</td>
<td>DAC</td>
<td>W. Wen  et al.</td>
<td>A New Learning Method For Inference Accuracy, Core Occupation, And Performance Co-Optimization On Truenorth Chip</td>
</tr>
<tr>
<td>2017</td>
<td>HPCA</td>
<td>L. Song  et al.</td>
<td>Pipelay: A Pipelined Reram-Based Accelerator For Deep Learning</td>
</tr>
<tr>
<td>2017</td>
<td>ASPDAC</td>
<td>Y. Wang  et al.</td>
<td>Classification accuracy improvement for neuromorphic computing systems with one-level precision synapses</td>
</tr>
<tr>
<td>2017</td>
<td>NeurIPS</td>
<td>W. Wen  et al.</td>
<td>TernGrad: Ternary Gradients To Reduce Communication In Distributed Deep Learning (Oral)</td>
</tr>
<tr>
<td>2017</td>
<td>DATE</td>
<td>J. Mao  et al.</td>
<td>Modnn: Local Distributed Mobile Computing System For Deep Neural Network</td>
</tr>
<tr>
<td>2017</td>
<td>ICCAD</td>
<td>J. Mao  et al.</td>
<td>Mednn: A Distributed Mobile System With Enhanced Partition And Deployment For Large-Scale Dnns</td>
</tr>
<tr>
<td>2017</td>
<td>ICCAD</td>
<td>J. Mao  et al.</td>
<td>Adalearner: An Adaptive Distributed Mobile Learning System For Neural Networks</td>
</tr>
<tr>
<td>2018</td>
<td>HPCA</td>
<td>L. Song  et al.</td>
<td>GraphR: Accelerating Graph Processing Using Reram</td>
</tr>
<tr>
<td>2018</td>
<td>ASPDAC</td>
<td>F. Chen  et al.</td>
<td>ReGAN: A Pipelined Reram-Based Accelerator For Generative Adversarial Networks</td>
</tr>
<tr>
<td>2018</td>
<td>ICLR</td>
<td>W. Wen  et al.</td>
<td>Learning Intrinsic Sparse Structures Within Long Short-Term Memory</td>
</tr>
<tr>
<td>2019</td>
<td>HPCA</td>
<td>L. Song  et al.</td>
<td>Hypar: Towards Hybrid Parallelism For Deep Learning Accelerator Array</td>
</tr>
<tr>
<td>2019</td>
<td>CVPR</td>
<td>N. Inkawhich et al.</td>
<td>Feature space perturbations yield more transferable adversarial examples</td>
</tr>
<tr>
<td>2019</td>
<td>DAC</td>
<td>J. Mao  et al.</td>
<td>Mobieye: An Efficient Cloud-Based Video Detection System For Real-Time Mobile Applications</td>
</tr>
<tr>
<td>2019</td>
<td>ICASSP</td>
<td>J. Zhang et al.</td>
<td>Learning Efficient Sparse Structure In Speech Recognition</td>
</tr>
<tr>
<td>2019</td>
<td>AICAS</td>
<td>Y. Huang et al.</td>
<td>Exploration Of Automatic Mixed-Precision Search For Deep Neural Networks</td>
</tr>
<tr>
<td>2019</td>
<td>VLSI Symp.</td>
<td>B. Yan  et al.</td>
<td>RRAM-Based Spiking Nonvolatile Computing-In-Memory Processing Engine with Precision-Configurable in Situ Nonlinear Activation</td>
</tr>
<tr>
<td>Year</td>
<td>Conference</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>2020</td>
<td>MobiHoc</td>
<td>A. Li et al.</td>
<td>MVStylizer: An Efficient Edge-Assisted Video Photorealistic Style Transfer System for Mobile Phones</td>
</tr>
<tr>
<td>2020</td>
<td>ICLR</td>
<td>N. Inkawhich et al.</td>
<td>Transferable Perturbations of Deep Feature Distributions</td>
</tr>
<tr>
<td>2020</td>
<td>HPCA</td>
<td>L. Song et al.</td>
<td>ACCPAR: Tensor Partitioning with Co-optimization of the Communication and Computation for Training Acceleration in a Heterogeneous Deep Learning Accelerator Array</td>
</tr>
<tr>
<td>2020</td>
<td>ISCA</td>
<td>N. Challapalle et al.</td>
<td>GaaS-X: Graph Analytics Accelerator Supporting Sparse Data Representation using Crossbar Architectures</td>
</tr>
<tr>
<td>2020</td>
<td>KDD</td>
<td>A. Li et al.</td>
<td>Privacy-Preserving Representation Learning on Graphs: A Mutual Information Perspective</td>
</tr>
<tr>
<td>2020</td>
<td>ICML</td>
<td>S. Li et al.</td>
<td>PENNI: Pruned Kernel Sharing for Efficient CNN Inference</td>
</tr>
<tr>
<td>2020</td>
<td>ICCAD</td>
<td>X. Yang et al.</td>
<td>ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration</td>
</tr>
<tr>
<td>2020</td>
<td>NeurIPS</td>
<td>N. Inkawhich et al.</td>
<td>Perturbing Across the Feature Hierarchy to Improve Standard and Strict Blackbox Attack Transferability</td>
</tr>
<tr>
<td>2021</td>
<td>SEC</td>
<td>A. Li et al.</td>
<td>LotteryFL: Empower Edge Intelligence with Personalized and Communication-Efficient Federated Learning</td>
</tr>
<tr>
<td>2021</td>
<td>SenSys</td>
<td>A. Li et al.</td>
<td>FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking</td>
</tr>
<tr>
<td>2021</td>
<td>CVPR</td>
<td>J. Sun et al.</td>
<td>Soteria: Provable Defense against Privacy Leakage in Federated Learning from Representation Perspective</td>
</tr>
<tr>
<td>2021</td>
<td>NeurIPS</td>
<td>J. Sun et al.</td>
<td>FL-WBC: Enhancing Robustness against Model Poisoning Attacks in Federated Learning from a Client Perspective</td>
</tr>
</tbody>
</table>
Q&A