
# Deep Neural Networks' Applications in EDA and Their Acceleration Techniques

Yiran Chen

*Electrical and Computing Engineering Department*

*Duke Center for Evolutionary Intelligence (CEI)*

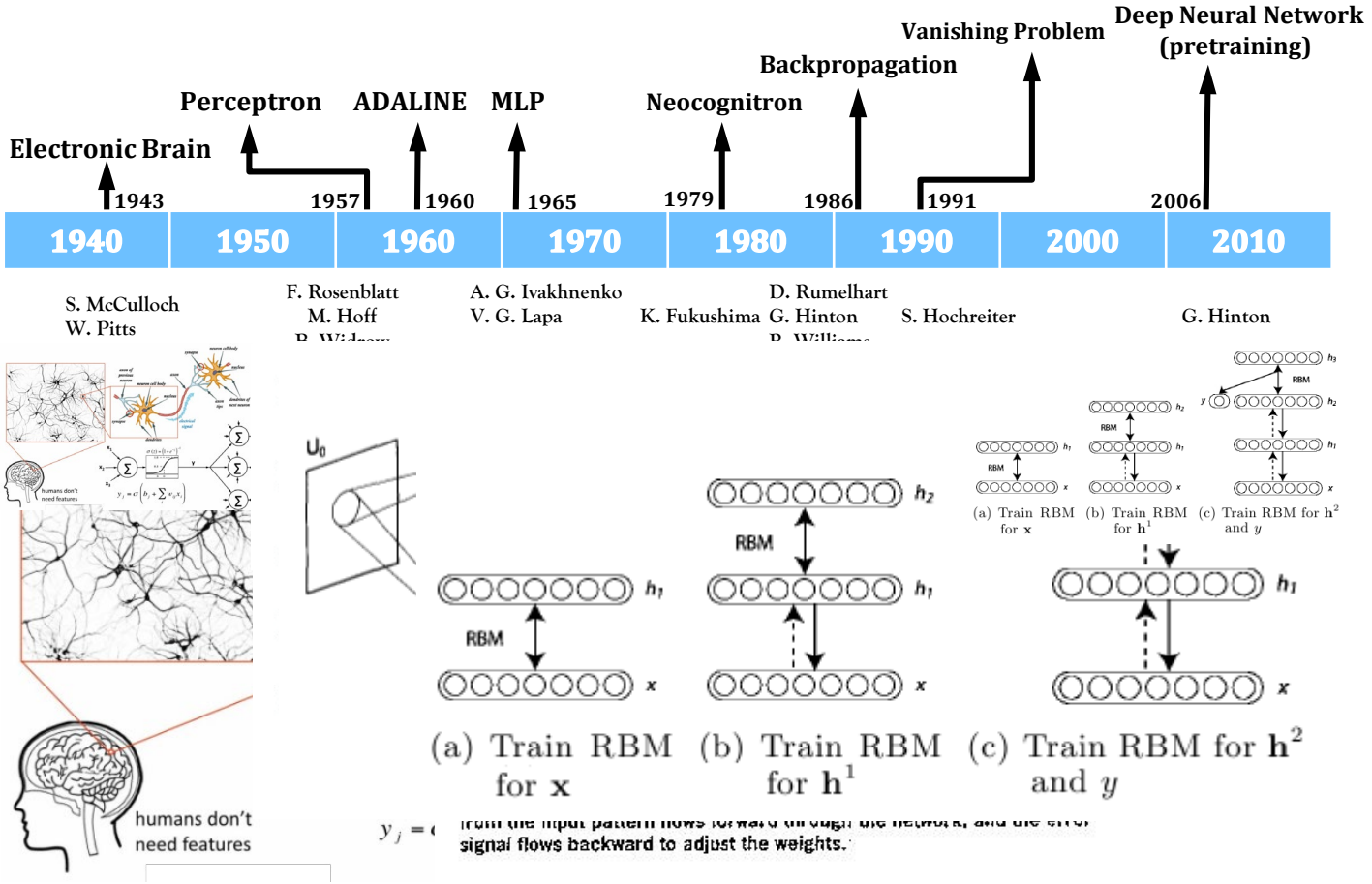
*NSF IUCRC For Alternative Sustainable and Intelligent Computing (ASIC)*

# Outline

- Introduction
- Learning Structured Sparsity in Deep Neural Networks
  - NIPS 2016
- TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning
  - NIPS 2017 (oral)
- ML Application Example in EDA: RouteNet
- Our prospective

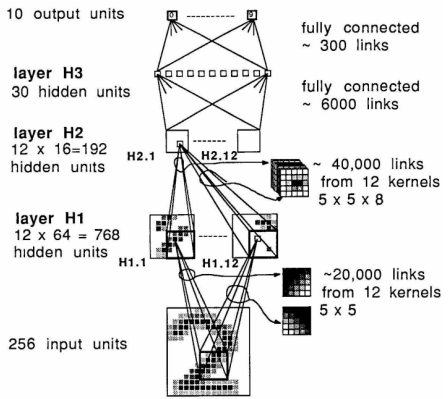
# Introduction

# Development of Neural Networks



# Rise and Decline of Neural Network

## Convolutional Network (1980s)

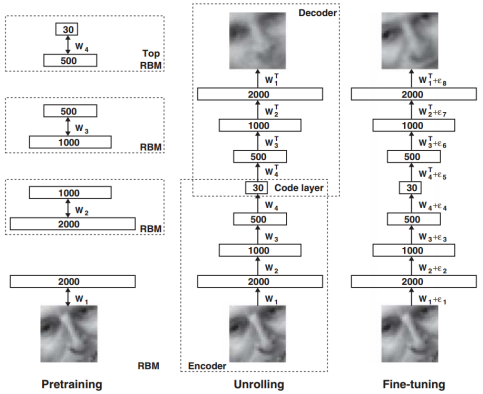




## Dark period (1990s)

- Serious problem: Vanishing gradient
- No benefits observed by adding more layers
- No high performance computing devices


## Renaissance (2006 ~ Present)





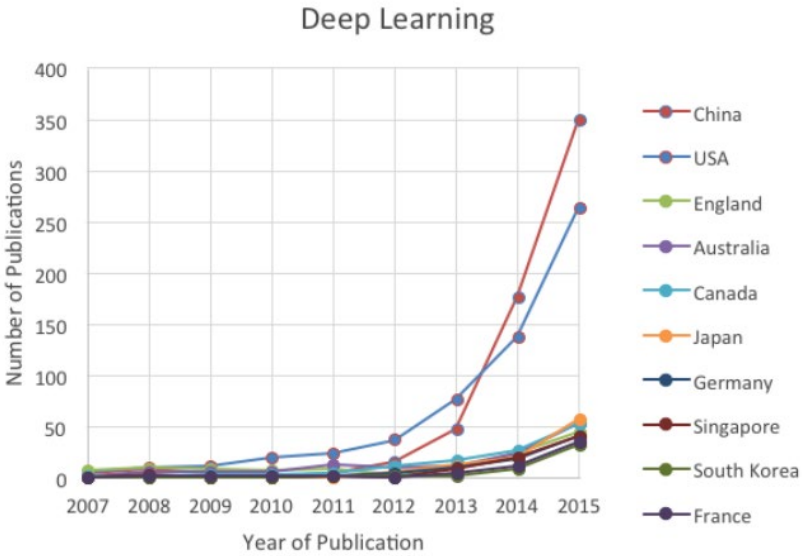
*Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition.1989.*

*J. Schmidhuber. Deep Learning in Neural Networks: An Overview. arxiv, 2014.*

*G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. Science, 2006.*

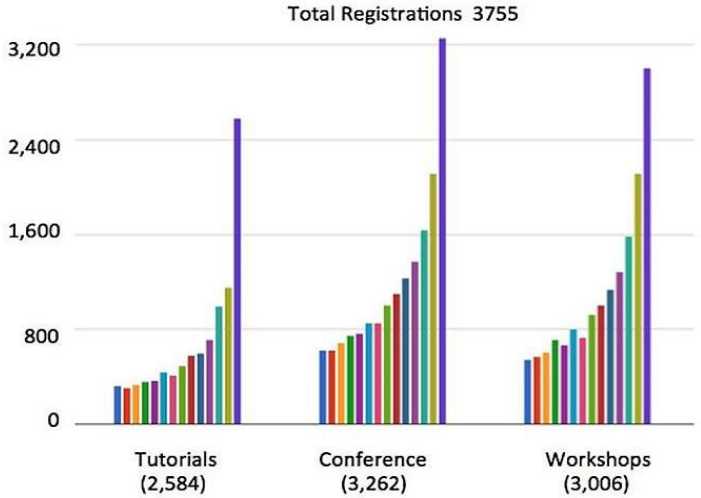
# Machine Learning in Academia

Journal articles mentioning "**deep learning**" or "**deep neural networks**"




*Source: Office of Science and Technology Policy/The White House*

NIPS registrations growth 2015: 3755, 2016: 6000+




NIPS does not organize sponsor presentations at the Conference or Workshops an
sponsors. Satellite meetings should be organized by sponsors immediatley before

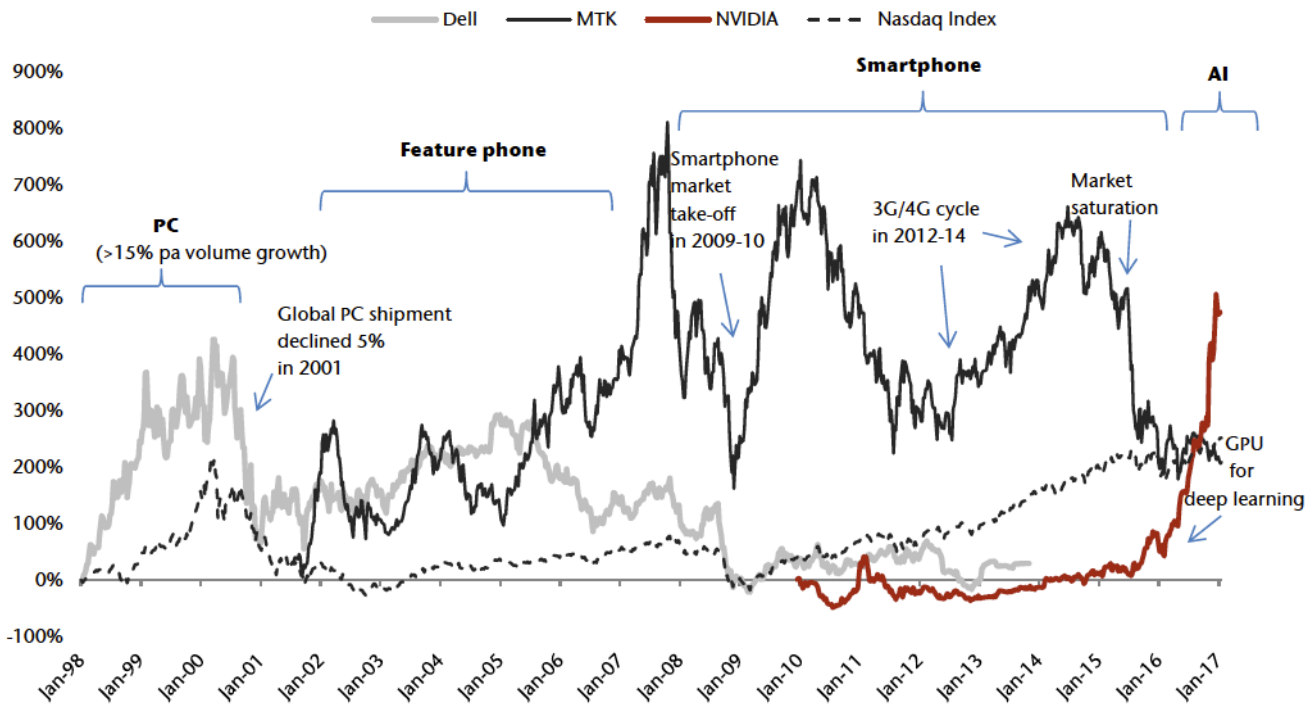
**2017 Levels of Sponsorship:**

**All Levels of Sponsorships are SOLD OUT.**

# Machine Learning in the Market

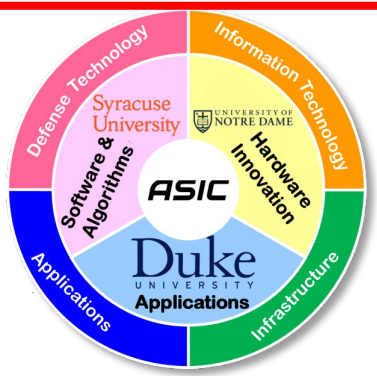
**Technology cycle** - from *PC*, to *smartphone*, to ***artificial intelligence?***

## *"Pure Play"* Share Price Performance




*Source: Bloomberg, Jefferies*

# NSF IUCRC ASIC Center




## What is ASIC?

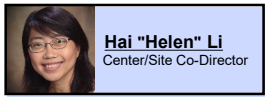
- The **A**lternative **S**ustainable and **I**ntelligent **C**omputing (**ASIC**) Center is a multi-site, multidisciplinary consortium that explores research frontiers in emerging computing platforms for cognitive applications
- The **ASIC** Center focuses on designing alternative computing platforms for cognitive applications, which perform inefficiently on conventional von Neumann architectures

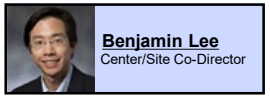
## Industry members:

THALES · SRC · ergomotion · 云知声 Unisound · Horizon Robotics · Kneron · QUANTIL · YH 永辉超市 YONGHUI SUPERSTORES · Lenovo · JD.COM · cadence · MAXBIOE

## Members include influential faculty across the three research sites:


**Yiran Chen**
Center/Site Director


**Hai "Helen" Li**
Center/Site Co-Director


**Benjamin Lee**
Center/Site Co-Director


**Robert Calderbank**

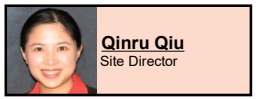

**Krish Chakrabarty**

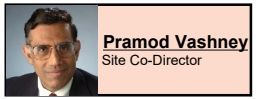

**Maria Gorlatova**


**Xin Li**


**Miroslav Pajic**

Duke UNIVERSITY


**Qinru Qiu**
Site Director


**Pramod Vashney**
Site Co-Director


**Chilukuri Mohan**
Site Co-Director


**Jian Tang**


**Amit Sanyal**


**Senem Velipasalar**


**Bei Yu**

Syracuse University


**Yiyu Shi**
Site Director


**Sharon Hu**
Site Co-Director


**Michael Niemier**
Site Co-Director


**Danny Chen**


**Patrick J. Flynn**


**Walter Scheirer**


**Chaoli Wang**

UNIVERSITY OF NOTRE DAME

# Facts of ASIC

**22** AFFILIATED FACULTY MEMBERS, INCLUDING 6 FEMALE PROFESSORS

**1** MEMBER OF NATIONAL ACADEMY OF ENGINEERING

**8** DIFFERENT DEPARTMENTS, INCLUDING ECE, CS, ACMS, BIO, MAE, MATH, PHYS AND SIS

**1** ACM FELLOWS AND 3 ACM DISTINGUISHED MEMBERS

**13** RECIPIENTS OF NSF CAREER AWARD

**10** IEEE FELLOWS

# Research of ASIC: DNN Acceleration

**RESEARCH TOPICS**

- Weight quantization
- Compact models
- Network compression
- Network pruning
- Distributed learning

## Representative Industrial Impacts

❑ Our 1-level quantization method (ASP-DAC'17 and DAC'16) is included in the latest SDK of IBM TrueNorth Chip, achieving 6X performance improvement and/or 2/3 hardware cost reduction.

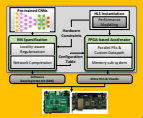
❑ Our structural pruning technique (NIPS'16)
- is supported by the library of Intel Nervana Neural Network Processors.
- is adopted by Intel NLP accelerator.
- is adopted by SF-Technology, achieving 2X performance improvement.

❑ Our TenGrad technique (NIPS'17) is supported by Facebook Caffe2 and HP parameter server product for distributed learning.

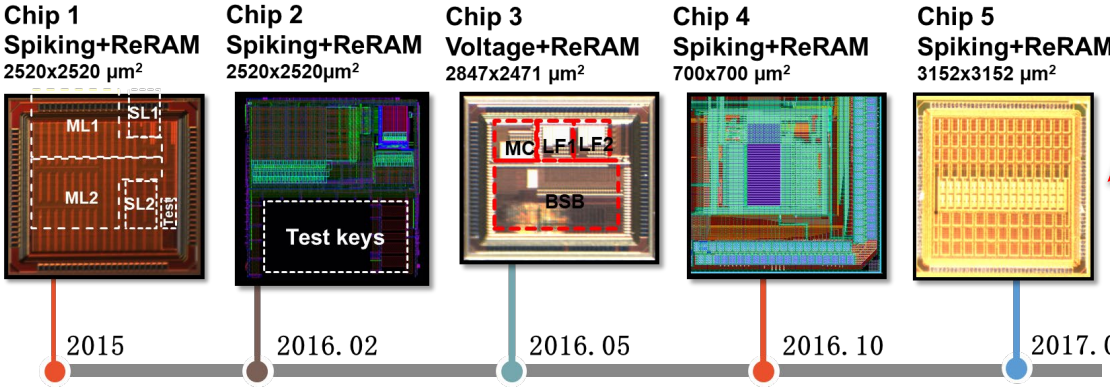
**Academic Recognitions:**

- ASP-DAC'17 Best Paper Award
- NIPS'17 Oral Presentation (40 out of 3240 submissions)
- DAC'16 Best Paper Nomination

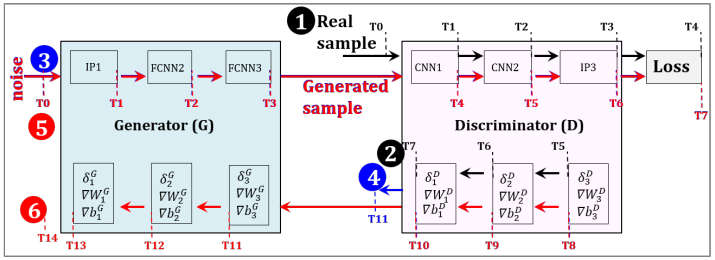
# Research of ASIC: AI Computing Platforms

| | | |
|---|---|---|
| CPU |  | Optimizing training and inference on CPU platforms for high cost efficiency. Achieved 3.1-7.3X speedup on Intel Atom, Xeon, and Xeon Phi (ICLR'17). |
| GPU |  | Systematic accelerations of deep learning on GPU and heterogeneous platforms. Received 3rd Place of 2018 IEEE Low-Power Image Recognition Challenge (track 2). |
| FPGA |  | Comprehensive technical portfolio of FPGA-based deep learning, including the first RNN acceleration work published in major FPGA conferences (FCCM 2015). |
| ASIC |  | Extensive experience on AI Chip design, including both architecture and circuit. Taping out chips in every 6-9 months. |
| Emerging |  | Access to emerging architectures and device technologies, such as IBM TrueNorth, Intel Loihi, and ReRAM, Spin Memory, etc. |

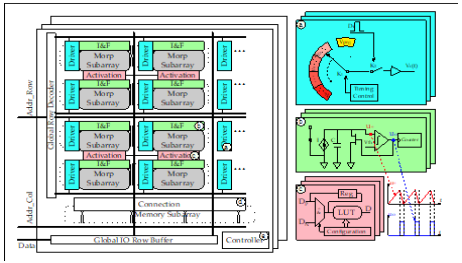
# Research of ASIC: AI Chips



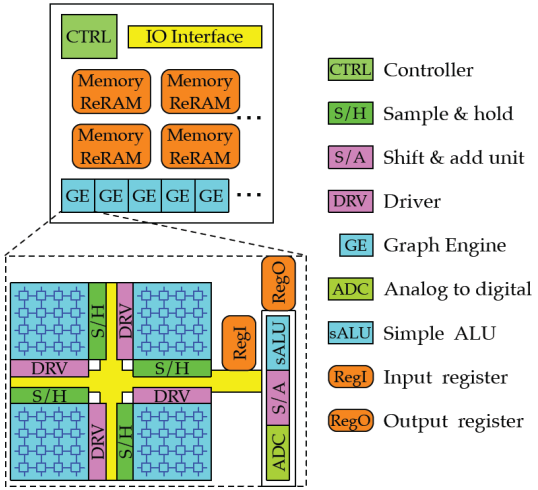

**The largest ReRAM-based Neuromorphic PIM Chip in the world!**




Generative Adversarial Network Acceleration (ASP-DAC'18)

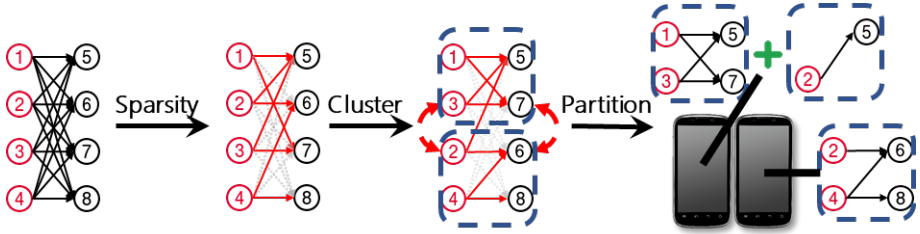


Pipeline and NoC Design (HPCA'17, DAC'15)

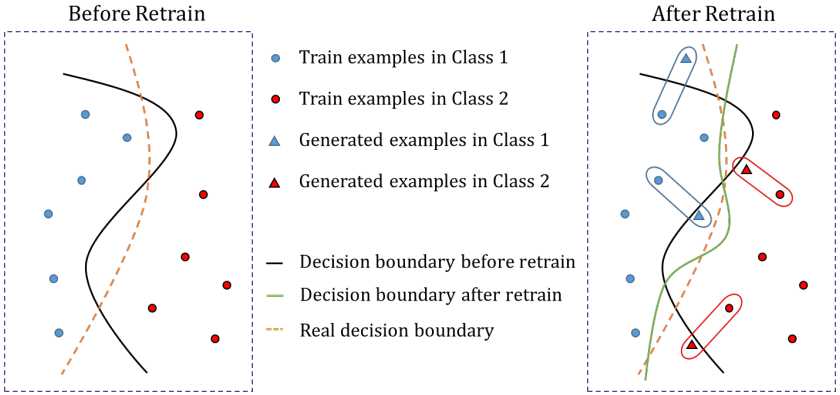



Graph Computing & Process-in-Memory (HPCA'18)

# Research of ASIC: Miscellaneous

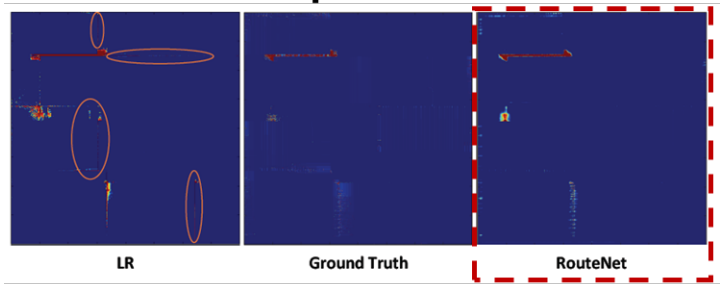



**Mobile DNN computing platforms (DATE'17 Best Paper)**




**Robustness and safety of DNN models**

## Hotspot Check

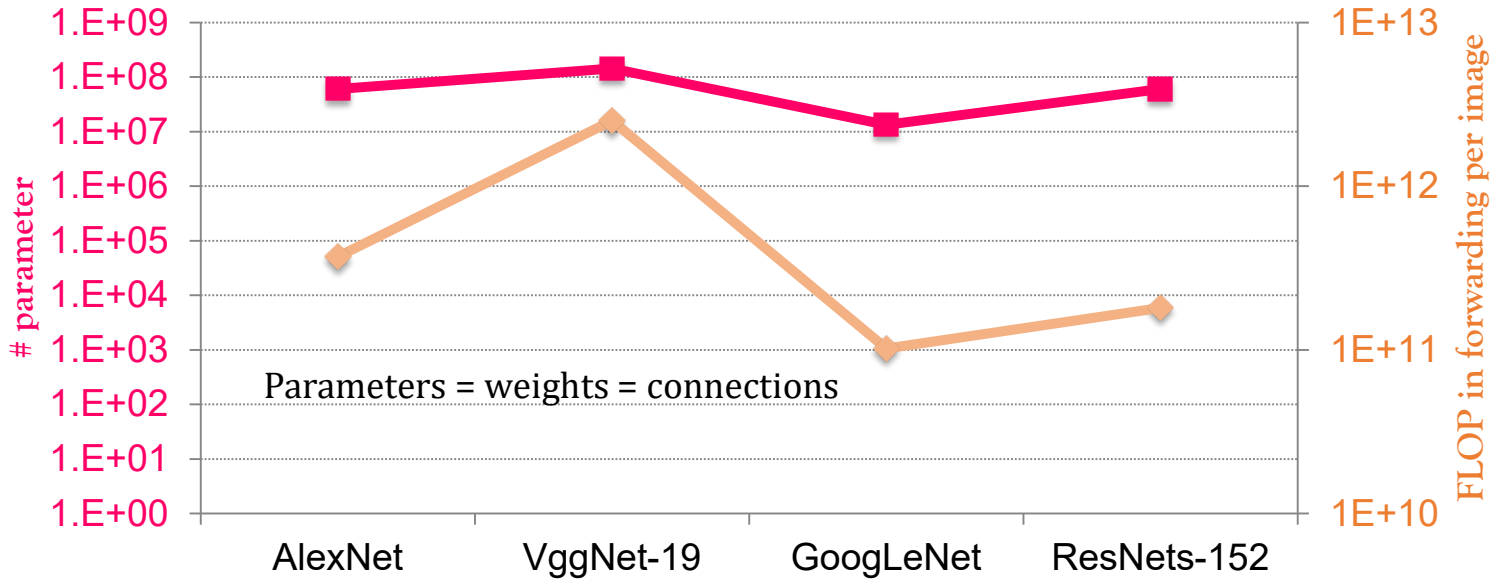



| Circuit Name | FPR (%) | TPR (%) | | | | |
|---|---|---|---|---|---|---|
| | | TR | GR | LR | SVM | RouteNet |
| des_perf | 0.54 | 17 | 56 | 54 | 42 | 74 |
| edit_dist | 1.00 | 25 | 36 | 38 | 28 | 64 |
| fft | 0.30 | 21 | 45 | 54 | 31 | 71 |
| matrix_mult_a | 0.21 | 13 | 30 | 34 | 12 | 49 |
| matrix_mult_b | 0.24 | 13 | 37 | 41 | 20 | 53 |
| Average | 0.46 | 18 | 41 | 44 | 27 | 62 |

**DNN-enabled Electronic Design Automation (EDA): Placement & route, High-level synthesis, Timing analysis, Data argumentation, …**

# Learning Structured Sparsity in Deep Neural Networks


# Complexity of Deep Neural Networks




Winners of ImageNet Challenge in recent years

Fewer parameters, fewer computation (FLOP: Floating Point Operation)

**How to reduce the number of parameters in DNN so as to reduce FLOP, meanwhile maintain the classification accuracy?**

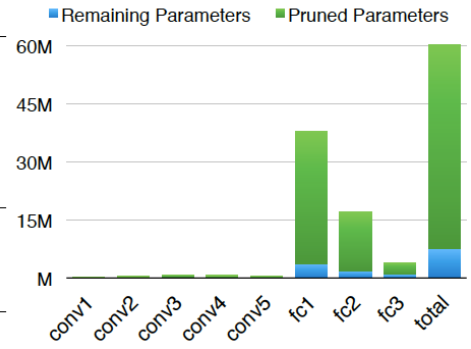
# Non-structurally Sparse DNNs

- State-of-the-art methods to reduce the number of parameters
  - Weight regularization (L1-norm)
  - Connection pruning

| Layer | conv1 | conv2 | conv3 | conv4 | conv5 |
|---|---|---|---|---|---|
| Sparsity% | 0.927 | 0.95 | 0.951 | 0.942 | 0.938 |
| Theoretical speedup | 2.61 | 7.14 | 16.12 | 12.42 | 10.77 |

Sparsity: the ratio of zeros remained

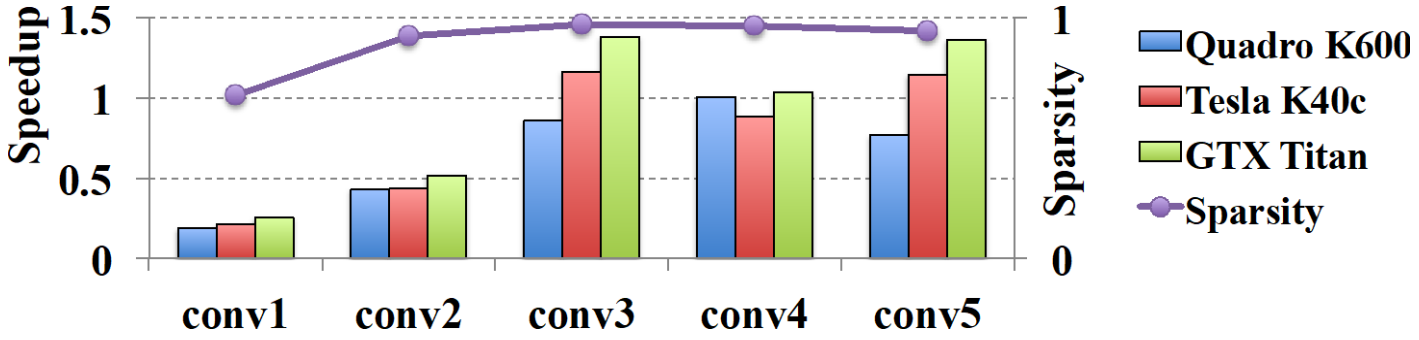
| Layer | Weights | FLOP | Act% | Weights% | FLOP% |
|---|---|---|---|---|---|
| conv1 | 35K | 211M | 88% | 84% | 84% |
| conv2 | 307K | 448M | 52% | 38% | 33% |
| conv3 | 885K | 299M | 37% | 35% | 18% |
| conv4 | 663K | 224M | 40% | 37% | 14% |
| conv5 | 442K | 150M | 34% | 37% | 14% |
| fc1 | 38M | 75M | 36% | 9% | 3% |
| fc2 | 17M | 34M | 40% | 9% | 3% |
| fc3 | 4M | 8M | 100% | 25% | 10% |
| Total | 61M | 1.5B | 54% | **11%** | **30%** |



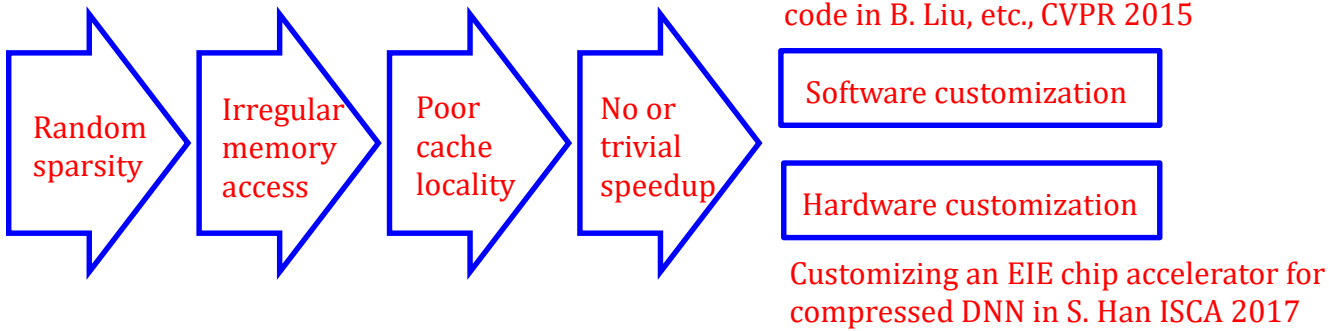

AlexNet, S. Han, *et al.*, NIPS 2015

AlexNet, B. Liu, *et al.*, CVPR 2015

# Theoretical Speedup ≠ Practical Speedup

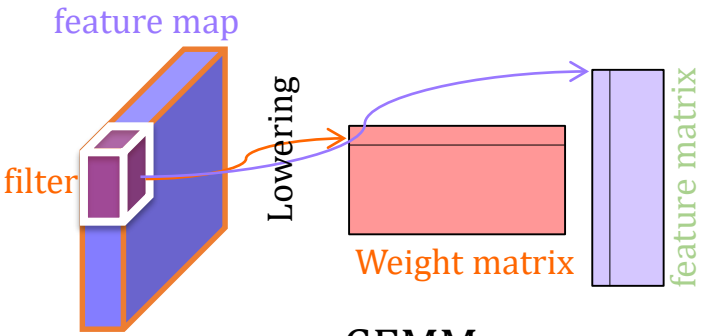



**Forwarding speedups of AlexNet on GPU platforms and the sparsity. Baseline is GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.**



# Theoretical Speedup ≈ Practical Speedup

**Example**: Removing rows/columns in GEMM (row/column-wise sparsity)




GEMM

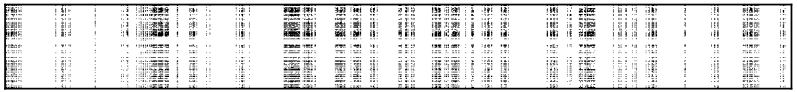
GEneral Matrix Matrix Multiplication

Non-structured sparsity

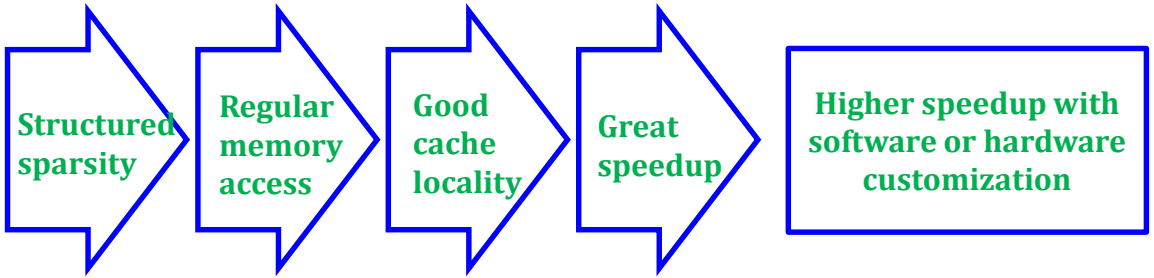
conv2_1: weight sparsity (col:8.7% row:19.5% elem:94.6%)

Structured sparsity

conv2_1: weight sparsity (col:75.2% row:21.9% elem:91.5%)



5.17X speedup



# Structured Sparsity Regularization

- Group Lasso regularization in ML model

$$\arg\min_{\mathbf{w}}\left\{E(\mathbf{w})\right\}=\arg\min_{\mathbf{w}}\left\{E_D(\mathbf{w})+\lambda_g\cdot R_g(\mathbf{w})\right\}$$

$$\Downarrow$$

$$\arg\min_{\mathbf{w}}\left\{E(\mathbf{w})\right\}=\arg\min_{\mathbf{w}}\left\{E_D(\mathbf{w})\right\}$$

$$s.t.\ R_g(\mathbf{w})\le\eta_g$$

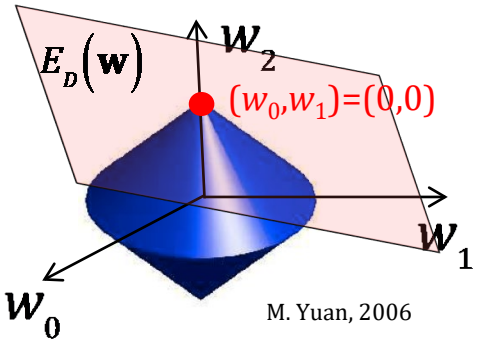
Example:

$$R_g\left(\underbrace{w_0,w_1}_{\text{group 1}},\underbrace{w_2}_{\text{group 2}}\right)=\sqrt{w_0^2+w_1^2}+\sqrt{w_2^2}\le\eta_g$$

Many groups will be zeros

$$R_g(\boldsymbol{w})=\sum_{g=1}^{G}||\boldsymbol{w}^{(g)}||_g,$$

$$||\boldsymbol{w}^{(g)}||_g=\sqrt{\sum_{i=1}^{|\boldsymbol{w}^{(g)}|}\left(w_i^{(g)}\right)^2}$$




M. Yuan, 2006

# SSL: Structured Sparsity Learning

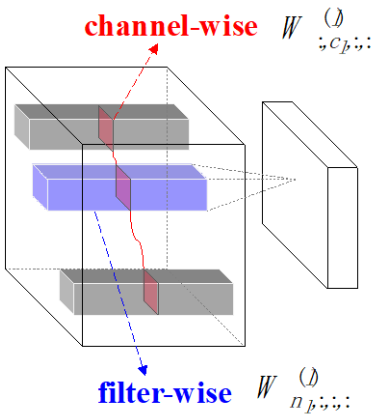
- Group Lasso regularization in DNNs:

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda \cdot R(\boldsymbol{W}) + \lambda_g \cdot \sum_{l=1}^{L} R_g\left(\boldsymbol{W}^{(l)}\right)$$

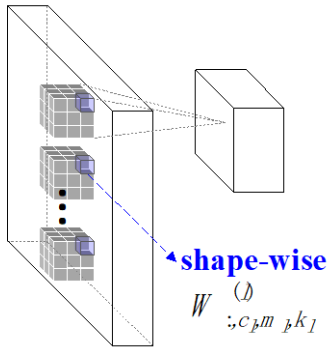
$$R_g(\boldsymbol{w}) = \sum_{g=1}^{G} ||\boldsymbol{w}^{(g)}||_g$$

**Learned structured sparsity is determined by the way of splitting groups**

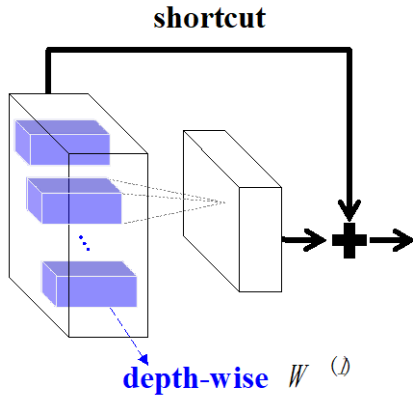
**Penalize unimportant filters and channels**




**Learn filter shapes**




**Learn the depth of layers**



# Penalizing unimportant filters & channels

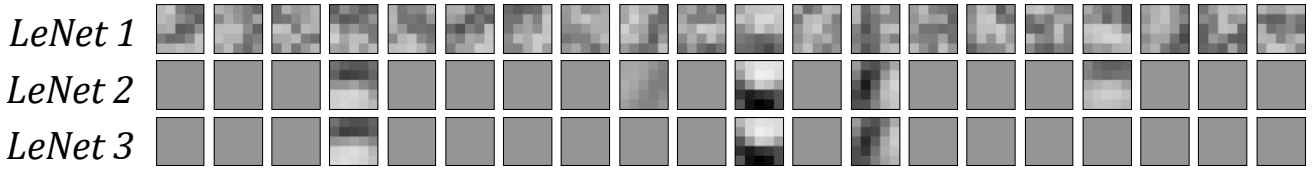
LeNet on MNIST

Table 1: Results after penalizing unimportant filters and channels in *LeNet*

| *LeNet* # | Error | Filter # § | Channel # § | FLOP § | Speedup § |
|---|---|---|---|---|---|
| 1 (*baseline*) | 0.9% | 20—50 | 1—20 | 100%—100% | 1.00×—1.00× |
| 2 | 0.8% | 5—19 | 1—4 | 25%—7.6% | 1.64×—5.23× |
| 3 | 1.0% | 3—12 | 1—3 | 15%—3.6% | 1.99×—7.44× |

§In the order of *conv1*—*conv2*

conv1 filters (gray level 128 represents zero)




Fewer but more natural patterns

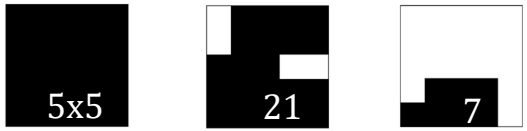
# Learning smaller filter shapes

Table 2: Results after learning filter shapes in *LeNet*

| *LeNet* # | Error | Filter size § | Channel # | FLOP | Speedup |
|---|---|---|---|---|---|
| 1 (*baseline*) | 0.9% | 25—500 | 1—20 | 100%—100% | 1.00×—1.00× |
| 4 | 0.8% | 21—41 | 1—2 | 8.4%—8.2% | 2.33×—6.93× |
| 5 | 1.0% | 7—14 | 1—1 | 1.4%—2.8% | 5.19×—10.82× |

§ The sizes of filters after removing zero shape fibers, in the order of *conv1—conv2*

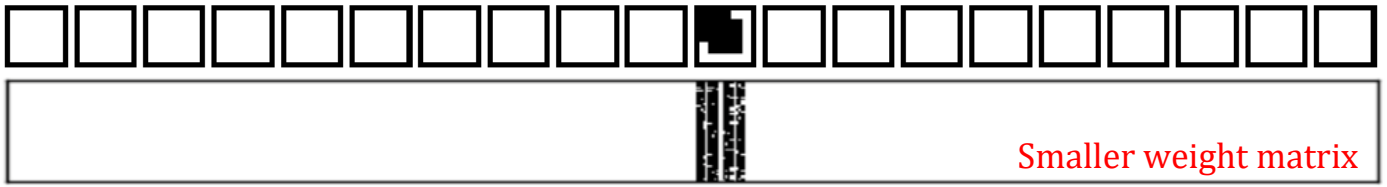
Learned shapes of conv1 filters:




*LeNet 1* *LeNet 4* *LeNet 5*

Learned shape of conv2 filters @ *LeNet 5*

3D 20x5x5 filters is regularized to 2D filters!




SSL can efficiently learn DNNs with smaller filters without accuracy loss

# AlexNet@ImageNet

Learning row-wise and column-wise sparsity:

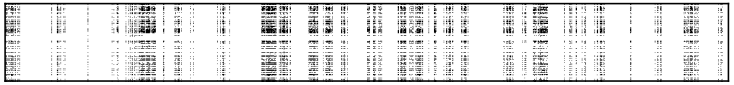
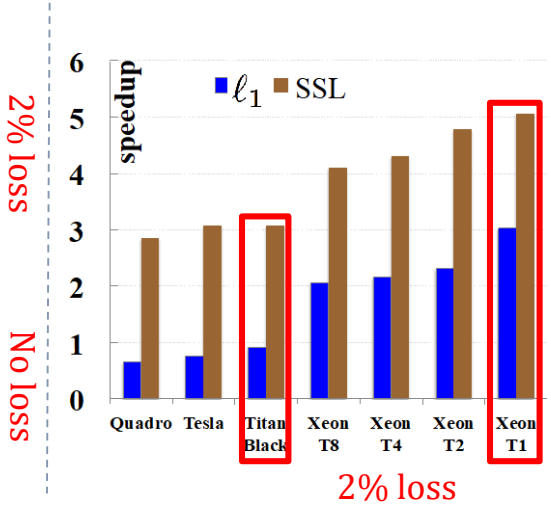


Table 4: Sparsity and speedup of *AlexNet* on ILSVRC 2012

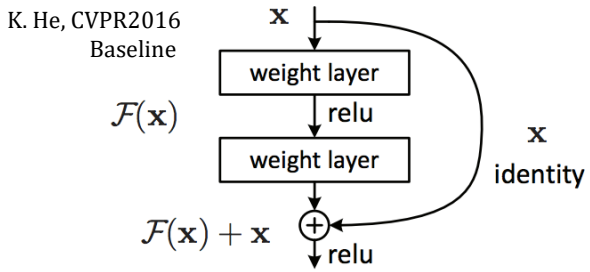
| # | Method | Top1 err. | Statistics | conv1 | conv2 | conv3 | conv4 | conv5 |
|---|---|---|---|---|---|---|---|---|
| 1 | $\ell_1$ | 44.67% | sparsity | 67.6% | 92.4% | 97.2% | 96.6% | 94.3% |
| | | | CPU × | 0.80 | 2.91 | 4.84 | 3.83 | 2.76 |
| | | | GPU × | 0.25 | 0.52 | 1.38 | 1.04 | 1.36 |
| 2 | SSL | 44.66% | column sparsity | 0.0% | 63.2% | 76.9% | 84.7% | 80.7% |
| | | | row sparsity | 9.4% | 12.9% | 40.6% | 46.9% | 0.0% |
| | | | CPU × | 1.05 | 3.37 | 6.27 | 9.73 | 4.93 |
| | | | GPU × | 1.00 | 2.37 | 4.94 | 4.03 | 3.05 |
| 3 | pruning[6] | 42.80% | sparsity | 16.0% | 62.0% | 65.0% | 63.0% | 63.0% |
| 4 | $\ell_1$ | 42.51% | sparsity | 14.7% | 76.2% | 85.3% | 81.5% | 76.3% |
| | | | CPU × | 0.34 | 0.99 | 1.30 | 1.10 | 0.93 |
| | | | GPU × | 0.08 | 0.17 | 0.42 | 0.30 | 0.32 |
| 5 | SSL | 42.53% | column sparsity | 0.00% | 20.9% | 39.7% | 39.7% | 24.6% |
| | | | CPU × | 1.00 | 1.27 | 1.64 | 1.68 | 1.32 |
| | | | GPU × | 1.00 | 1.25 | 1.63 | 1.72 | 1.36 |

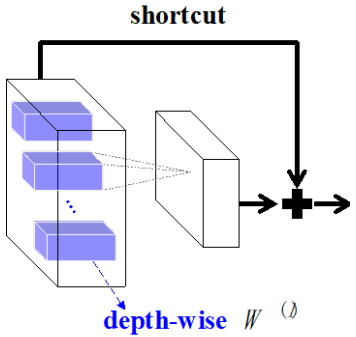



1. Non-structured sparsity method even slows down in some layers
2. layer-wise 5.1X /3.1X on CPU/GPU with 2% accuracy loss
3. layer-wise 1.4X on both CPU and GPU w/o accuracy loss
4. Higher speedups than non-structured speedups

# Regularizing the depth of DNNs

Experiments of ResNets on CIFAR-10

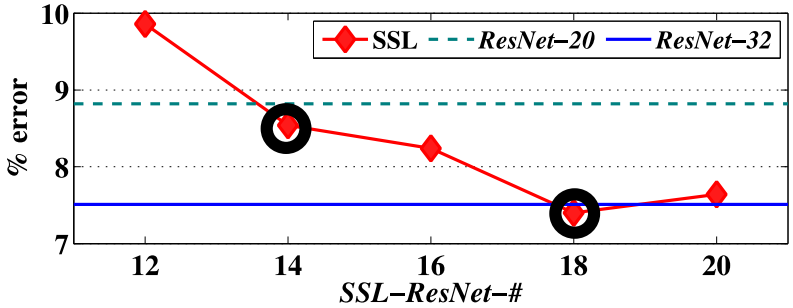


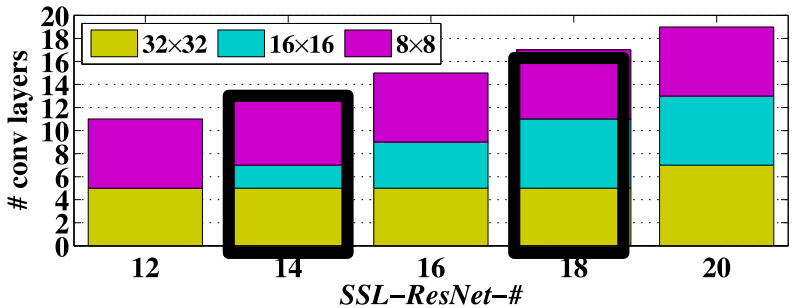




ResNet-20/32: baseline with 20/32 layers

SSL-ResNet-#: Ours with # layers after learning depth of ResNet-20

| | # layers | error | # layers | error |
|---|---|---|---|---|
| ResNet | 20 | 8.82% | 32 | 7.51% |
| SSL-ResNet | 14 | 8.54% | 18 | 7.40% |



# TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning

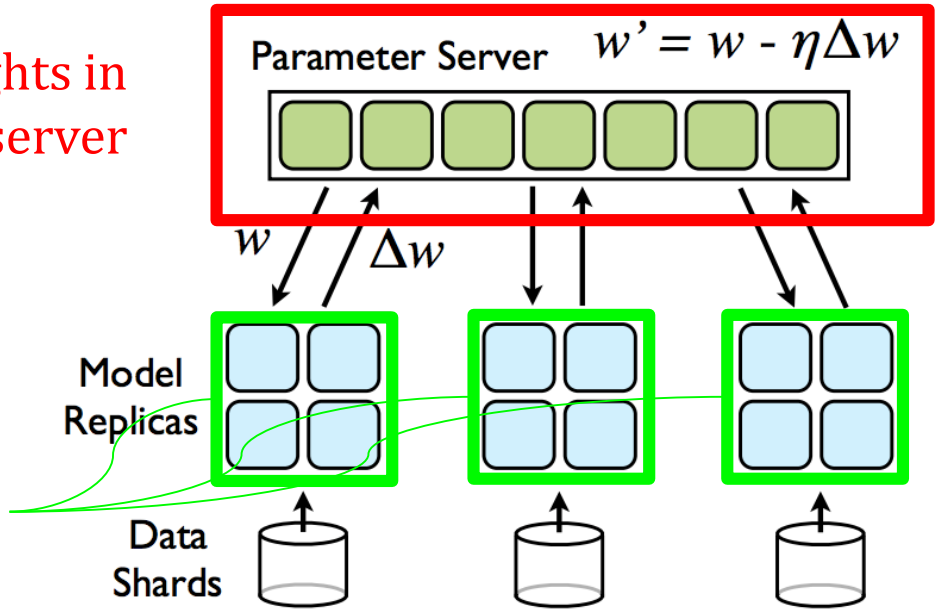
Duke UNIVERSITY

EI

# Distributed Deep Learning

When parallelism increase, communication is the bottleneck

Synchronize weights in parameter server



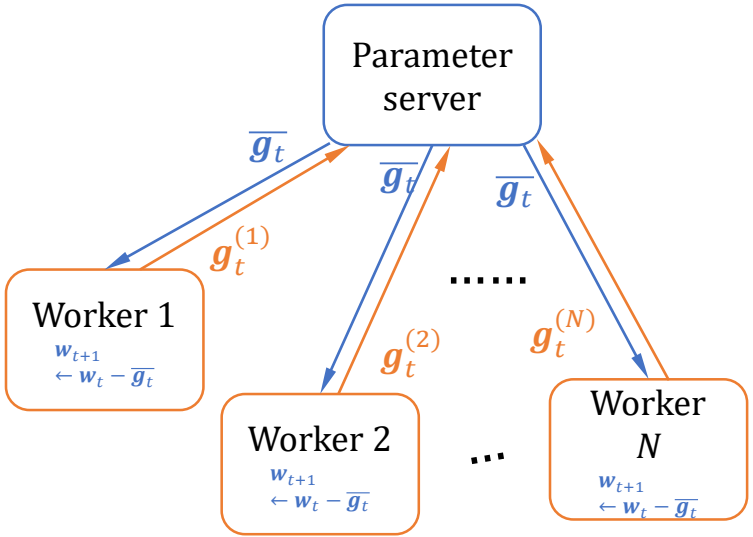

**Train the same model by different data in parallel**

DistBelief by Google

# TernGrad – distributed training with ternary gradients

Key ideas to reduce communication:

1. Randomly quantize gradients to only three levels (0, ±1)
2. The expectations of quantized gradients equal original values
3. Exchange quantized gradients instead of floating weights




**Algorithm 1** *TernGrad*: distributed SGD training using ternary gradients.

**Worker :** $i = 1, ..., N$

1 Input $\boldsymbol{z}_t^{(i)}$, a part of a mini-batch of training samples $\boldsymbol{z}_t$

2 Compute gradients $\boldsymbol{g}_t^{(i)}$ under $\boldsymbol{z}_t^{(i)}$

3 Ternarize gradients to $\tilde{\boldsymbol{g}}_t^{(i)} = ternarize(\boldsymbol{g}_t^{(i)})$

4 Push ternary $\tilde{\boldsymbol{g}}_t^{(i)}$ to the server

5 Pull averaged gradients $\overline{\boldsymbol{g}_t}$ from the server

6 Update parameters $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \cdot \overline{\boldsymbol{g}_t}$

**Server :**

7 Average ternary gradients $\overline{\boldsymbol{g}_t} = \sum_i \tilde{\boldsymbol{g}}_t^{(i)} / N$

$$\tilde{\boldsymbol{g}}_t = ternarize(\boldsymbol{g}_t) = s_t \cdot sign\,(\boldsymbol{g}_t) \circ \boldsymbol{b}_t$$

$$s_t \triangleq max\,(abs\,(\boldsymbol{g}_t))$$

$$\begin{cases} P(b_{tk} = 1 \mid \boldsymbol{g}_t) = |g_{tk}|/s_t \\ P(b_{tk} = 0 \mid \boldsymbol{g}_t) = 1 - |g_{tk}|/s_t \end{cases}$$

# TernGrad – Convergence

Mathematically guarantee the convergence of *TernGrad*

L. Bottou 1998

**Assumption 1.** *$C(\boldsymbol{w})$ has a single minimum $\boldsymbol{w}^*$ and gradient $-\nabla_{\boldsymbol{w}} C(\boldsymbol{w})$ always points to $\boldsymbol{w}^*$, i.e.,*

$$\forall \epsilon > 0, \inf_{||\boldsymbol{w}-\boldsymbol{w}^*||^2>\epsilon} (\boldsymbol{w}-\boldsymbol{w}^*)^T \nabla_{\boldsymbol{w}} C(\boldsymbol{w}) > 0. \quad (8)$$

Convexity is a subset of Assumption 1, and we can easily find non-convex functions satisfying it.

**Assumption 2.** *Learning rate $\gamma_t$ is positive and constrained as $\sum_{t=0}^{+\infty} \gamma_t^2 < +\infty$ and $\sum_{t=0}^{+\infty} \gamma_t = +\infty$, which ensures $\gamma_t$ decreases neither very fast nor very slow respectively.*

We assume the gradient is bounded as

**Assumption 3** (Gradient Bound)**.** *The gradient $\boldsymbol{g}$ is bounded as $\mathbf{E}\{max(abs(\boldsymbol{g})) \cdot ||\boldsymbol{g}||_1\} \leq A + B\,||\boldsymbol{w}-\boldsymbol{w}^*||^2$, where $A, B > 0$ and $||\cdot||_1$ is $\ell_1$ norm.*

**Theorem 1.** *When online learning systems update as $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t (s_t \cdot sign(\boldsymbol{g}_t) \circ \boldsymbol{b}_t)$ using stochastic ternary gradients, they converge **almost surely** toward minimum $\boldsymbol{w}^*$, i.e., $P(\lim_{t\to+\infty} \boldsymbol{w}_t = \boldsymbol{w}^*) = 1$.*

# TernGrad – Gradients Histograms

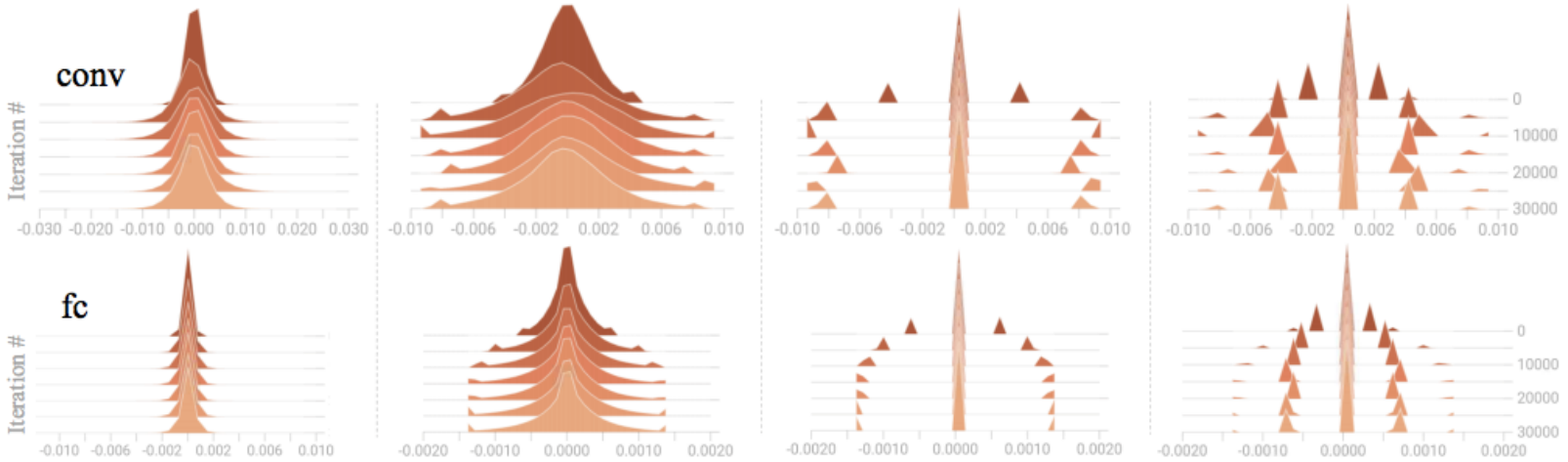
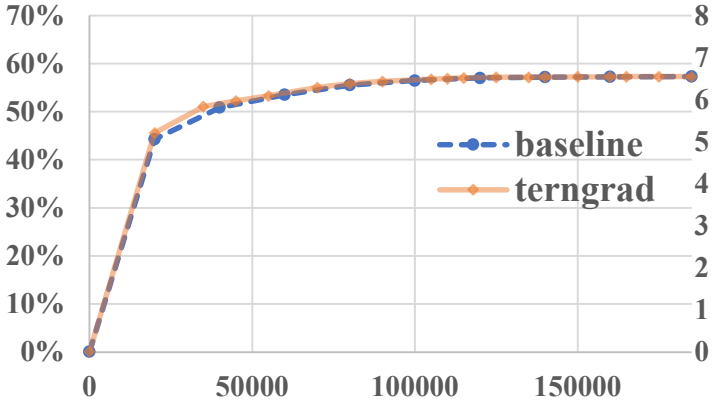



Figure 2: Histograms of (a) original floating gradients, (b) clipped gradients, (c) ternary gradients and (d) final averaged gradients. Visualization by TensorBoard. The DNN is *AlexNet* distributed on two workers, and vertical axis is the training iteration. Top row visualizes the third convolutional layer and bottom one visualizes the first fully-connected layer.

# TernGrad – AlexNet

(a) Top-1 accuracy vs iteration




(b) Training loss vs iteration

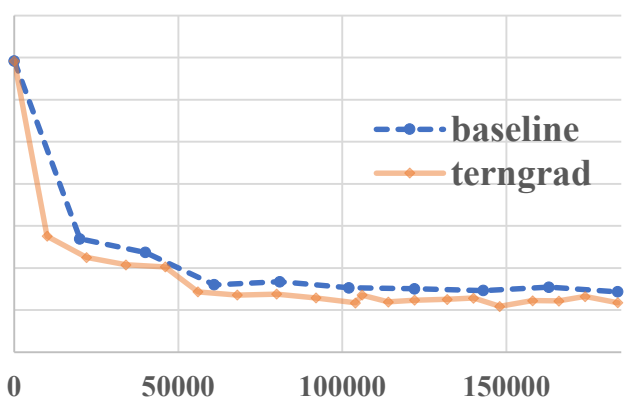



Table 2: Accuracy comparison for *AlexNet*.

| base LR | mini-batch size | workers | iterations | gradients | weight decay | DR† | top-1 | top-5 |
|---|---|---|---|---|---|---|---|---|
| 0.01 | 256 | 2 | 370K | floating | 0.0005 | 0.5 | 57.33% | 80.56% |
| | | | | *TernGrad* | 0.0005 | 0.2 | 57.61% | 80.47% |
| | | | | *TernGrad*-noclip ‡ | 0.0005 | 0.2 | 54.63% | 78.16% |
| 0.02 | 512 | 4 | 185K | floating | 0.0005 | 0.5 | 57.32% | 80.73% |
| | | | | *TernGrad* | 0.0005 | 0.2 | 57.28% | 80.23% |
| 0.04 | 1024 | 8 | 92.5K | floating | 0.0005 | 0.5 | **56.62%** | 80.28% |
| | | | | *TernGrad* | 0.0005 | 0.2 | **57.54%** | 80.25% |

† DR: dropout ratio, the ratio of dropped neurons. ‡ *TernGrad* without gradient clipping.

# TernGrad – GoogLeNet

Table 3: Accuracy comparison for *GoogLeNet*.

| base LR | mini-batch size | workers | iterations | gradients | weight decay | DR | top-5 |
|---|---|---|---|---|---|---|---|
| 0.04 | 128 | 2 | 600K | floating | 4e-5 | 0.2 | 88.30% |
| | | | | *TernGrad* | 1e-5 | 0.08 | 86.77% |
| 0.08 | 256 | 4 | 300K | floating | 4e-5 | 0.2 | 87.82% |
| | | | | *TernGrad* | 1e-5 | 0.08 | 85.96% |
| 0.10 | 512 | 8 | 300K | floating | 4e-5 | 0.2 | 89.00% |
| | | | | *TernGrad* | 2e-5 | 0.08 | 86.47% |

# TernGrad – Speedup

A performance model to evaluate the speed distributed training.

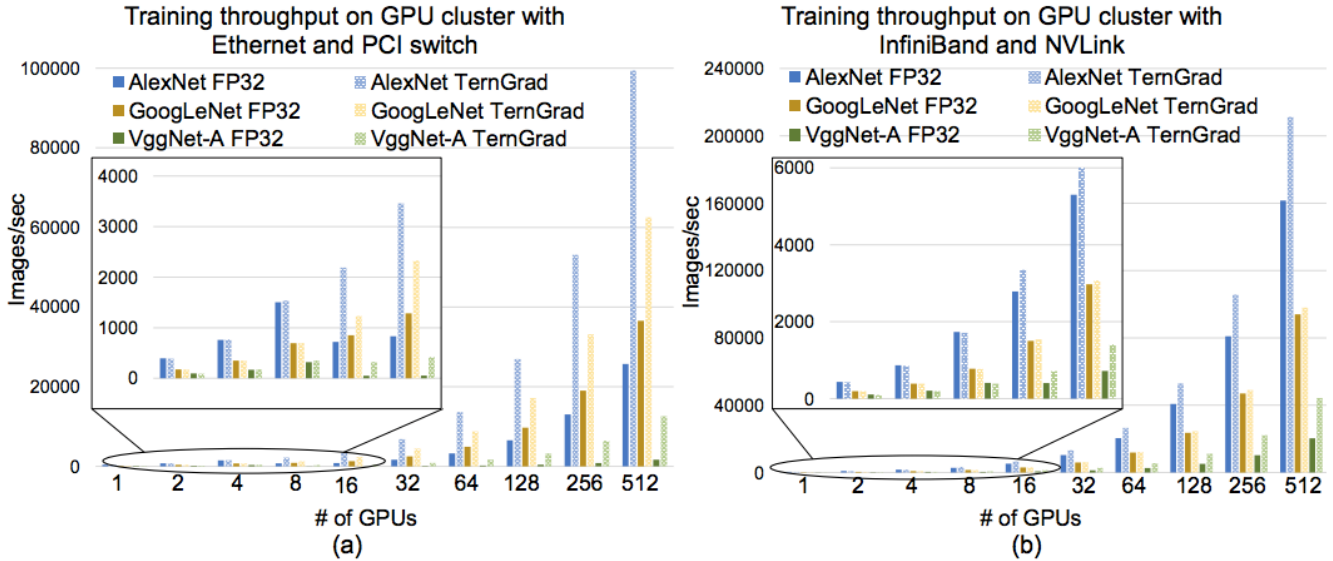
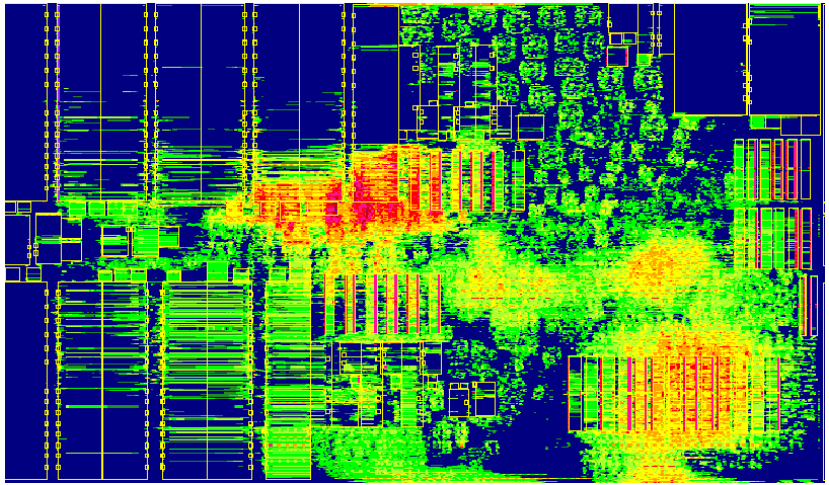



Figure 5: Training throughput on two different GPUs clusters: (a) 128-node GPU cluster with 1Gbps Ethernet, each node has 4 NVIDIA GTX 1080 GPUs and one PCI switch; (b) 128-node GPU cluster with 100 Gbps InfiniBand network connections, each node has 4 NVIDIA Tesla P100 GPUs connected via NVLink. Mini-batch size per GPU of *AlexNet*, *GoogLeNet* and *VggNet-A* is 128, 64 and 32, respectively

# RouteNet: Routability Prediction Using CNN

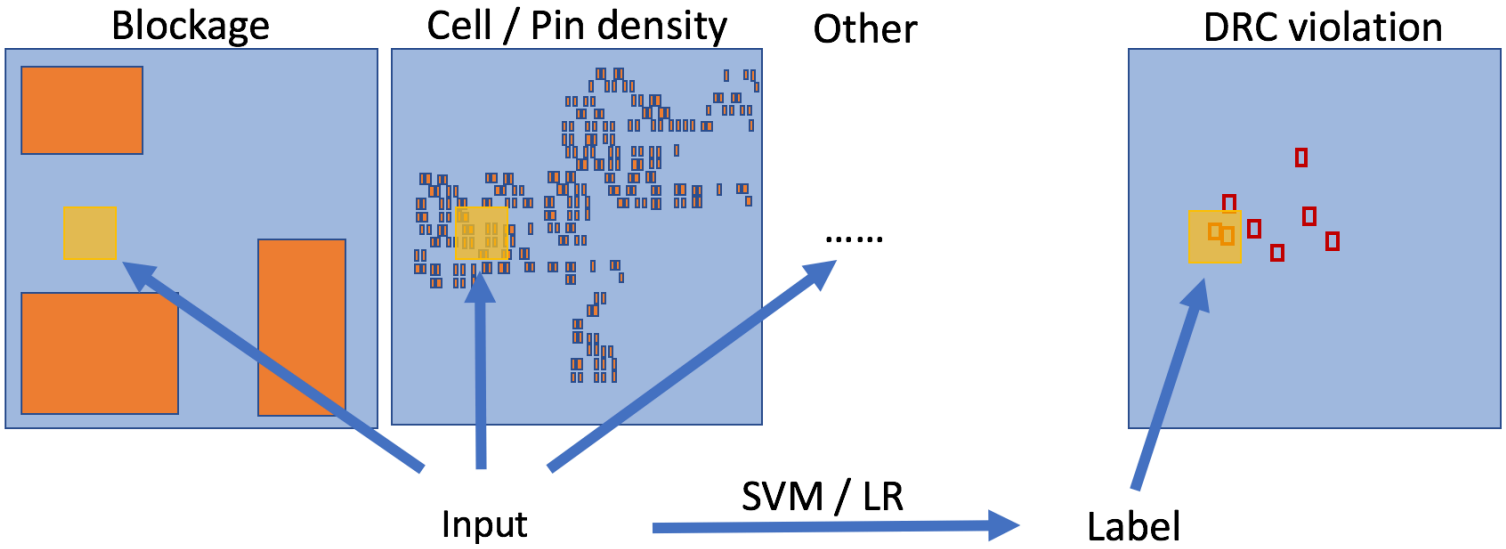
# Early Routability Prediction

- Routability: post-routing design rule violations
- Early prediction at placement stage
- Analytical techniques
  - Very fast
  - Not enough fidelity
- Trial routing
  - Acceptable fidelity
  - Not fast enough



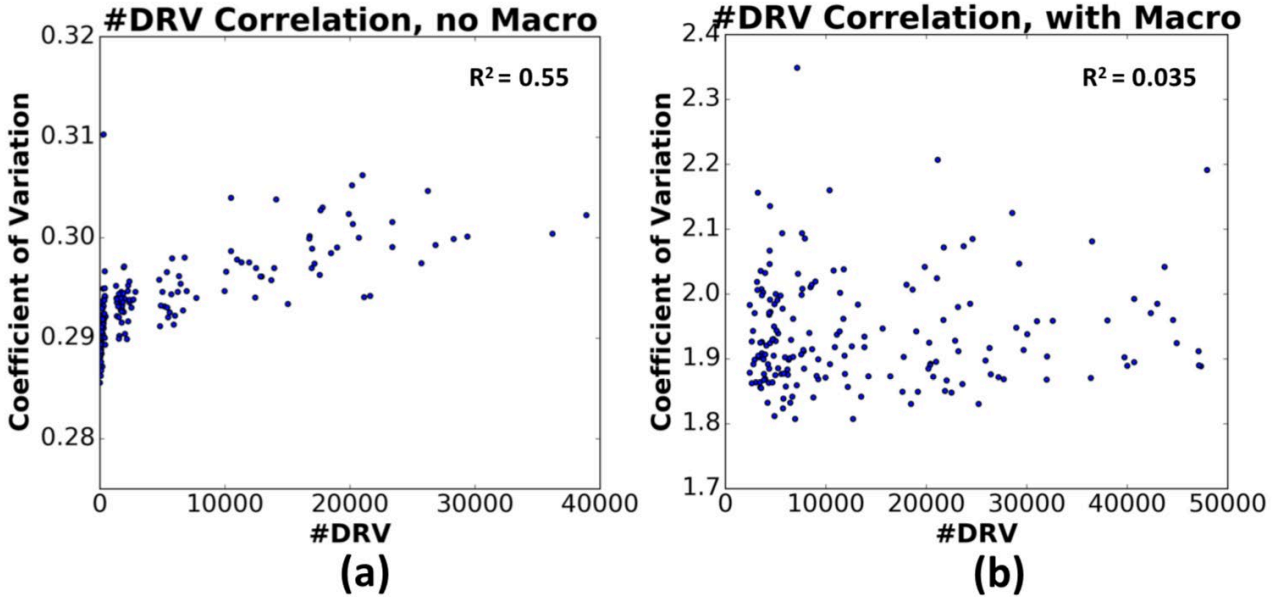
# Previous ML Approaches

- Learning on ***small*** *cropped regions*



# Challenges of Macros

- Layout is less homogeneous
- Correlation between pin density and #DRV becomes weak



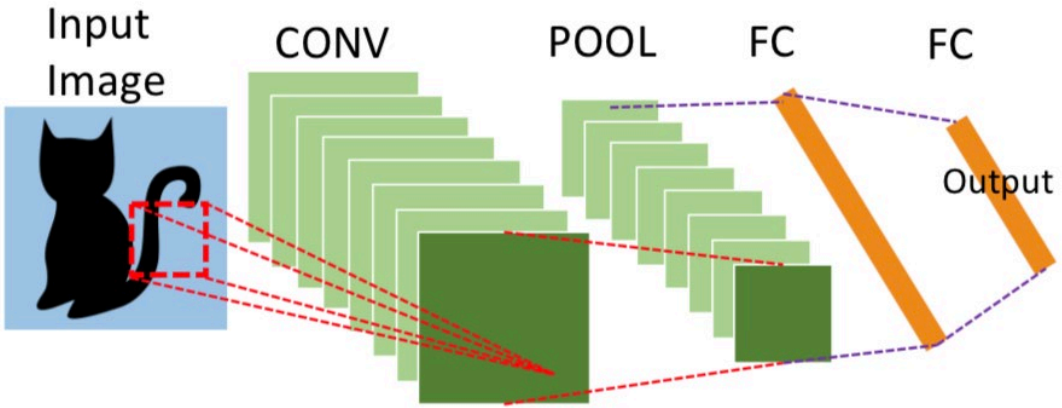

Each point corresponds to one placement

# Problem Formulations

- Predicting overall number of design rule violations (#DRV)
  - Given two placement solutions, tell which is more routable with high fidelity

- DRV hotspot detection
  - Given a relatively routable placement solution, pinpoint DRV hotspots such that mitigation measures are well targeted

# CNN for #DRV Prediction

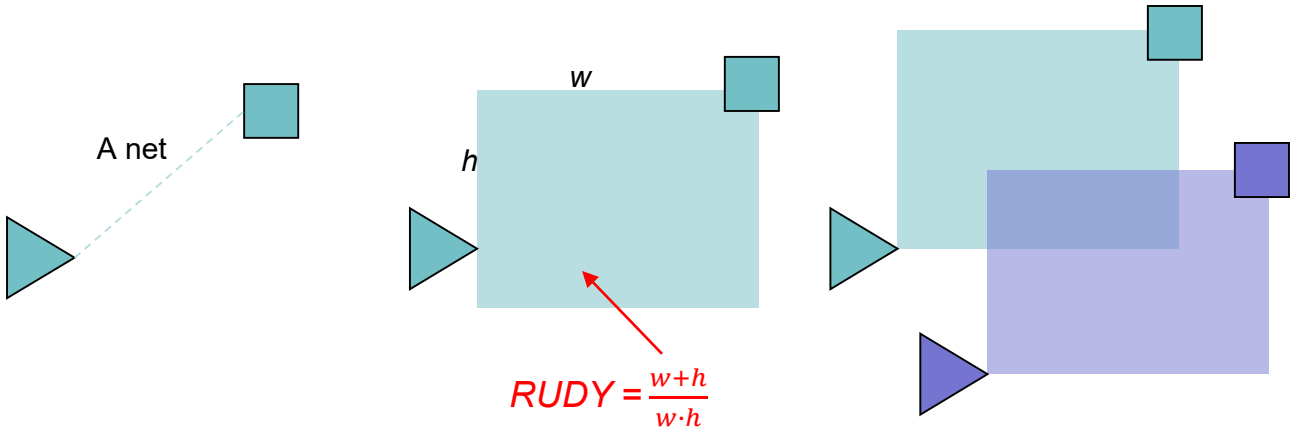



Convolutional (CONV) layers
Pooling (POOL) layers
Fully Connected (FC) layers
Widely used in image classification

- Given a cell placement, classify it among four routability levels, c0, c1, c2, c3
- c0 has the least #DRVs

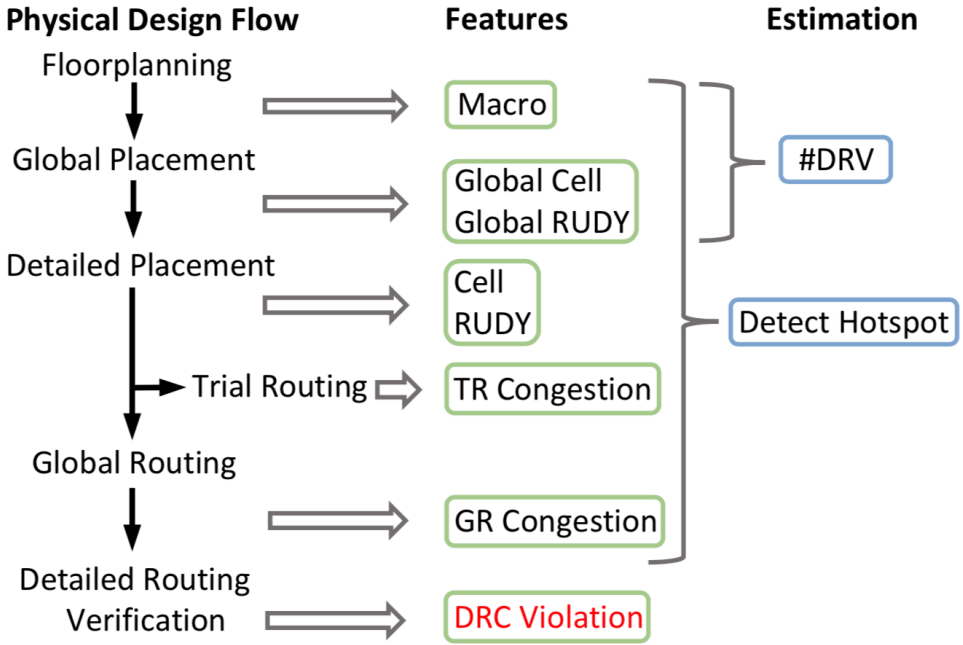
# An Important Feature

- RUDY (Rectangular Uniform wire DensitY) (P. Spinder et al.  DATE07)
- RUDY at a point is superposition of RUDYs of multiple nets



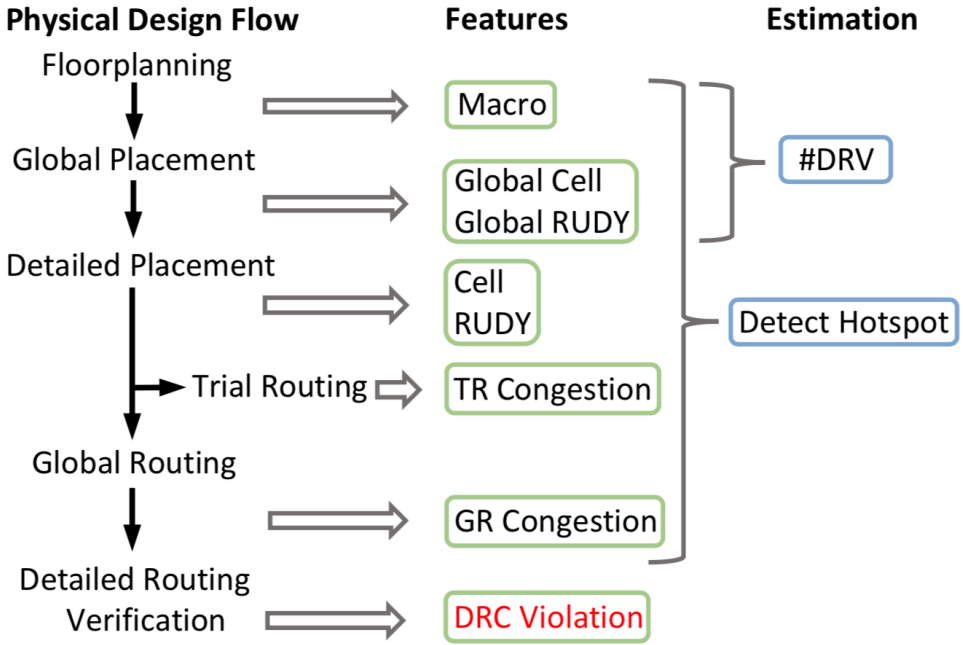
# Features for #DRV Prediction

- Macro:
  - region occupied by macros
  - density of macro pins in each layer
- Cell:
  - density of cells
  - density of cell pins
- Global Cell:
  - cell features at global placement
- Global RUDY:
  - RUDY features calculated by global placement results

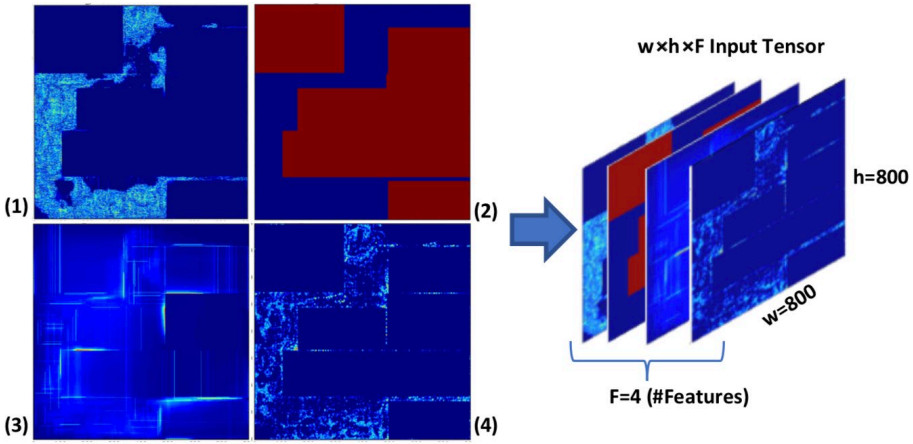


# Additional Features for Hotspot Detection

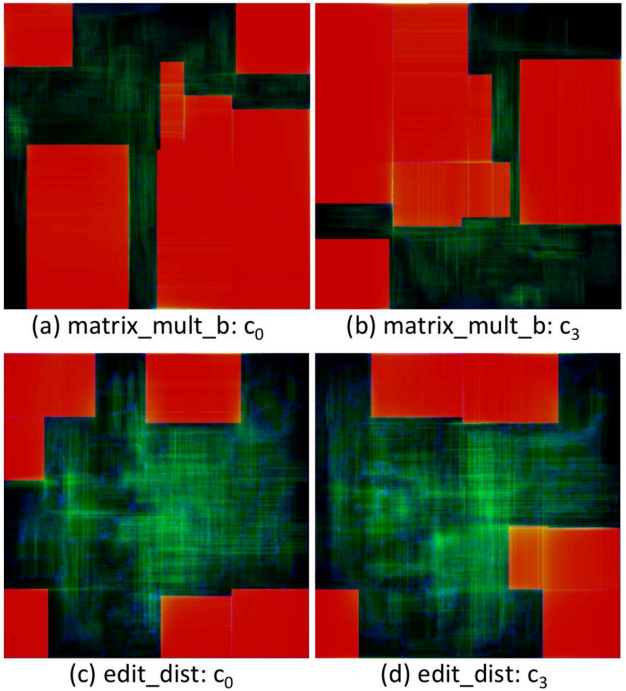
- RUDY
  - long-range RUDY
    - RUDY from long-range nets
  - short-range RUDY
    - DURY from short-range nets
  - RUDY pins
    - pins with density value equal to the RUDY value of its net
- Congestion
  - trial global routing congestion
  - global routing congestion
- DRC violation
  - prediction target / label



# Feature Illustration

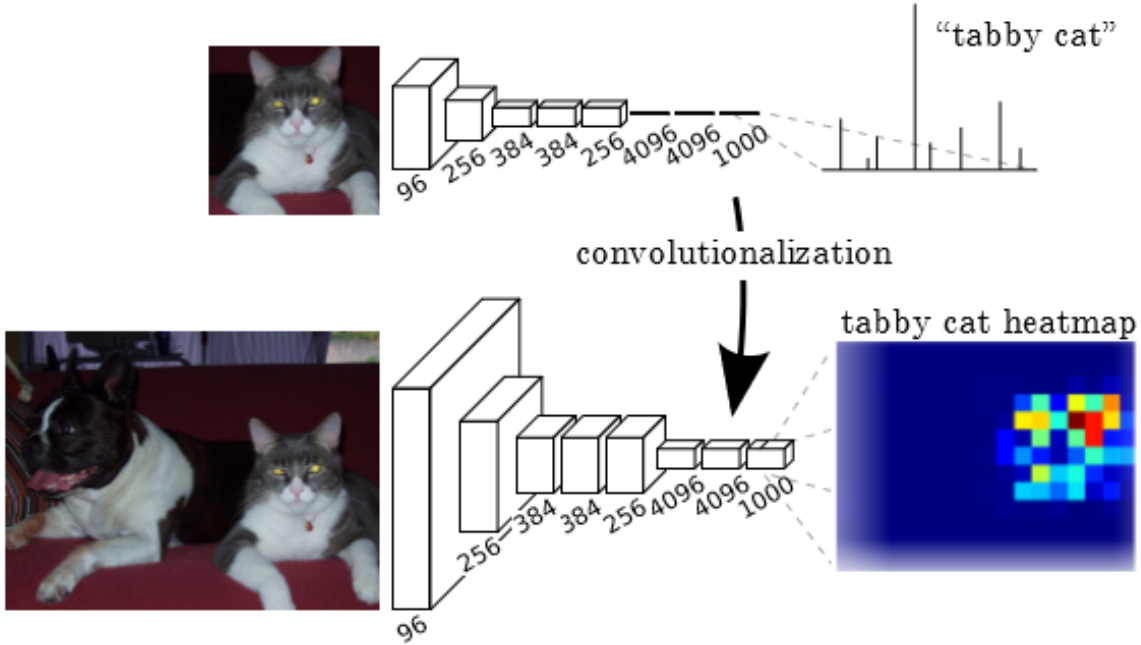



Input tensor constructed by stacking 2D features:
(1) Pin density, (2) macro (3) long-range RUDY, (4) RUDY pins




Input features for #DRV prediction.
Red: macro region
Green: global long-range RUDY
Blue: global RUDY pins

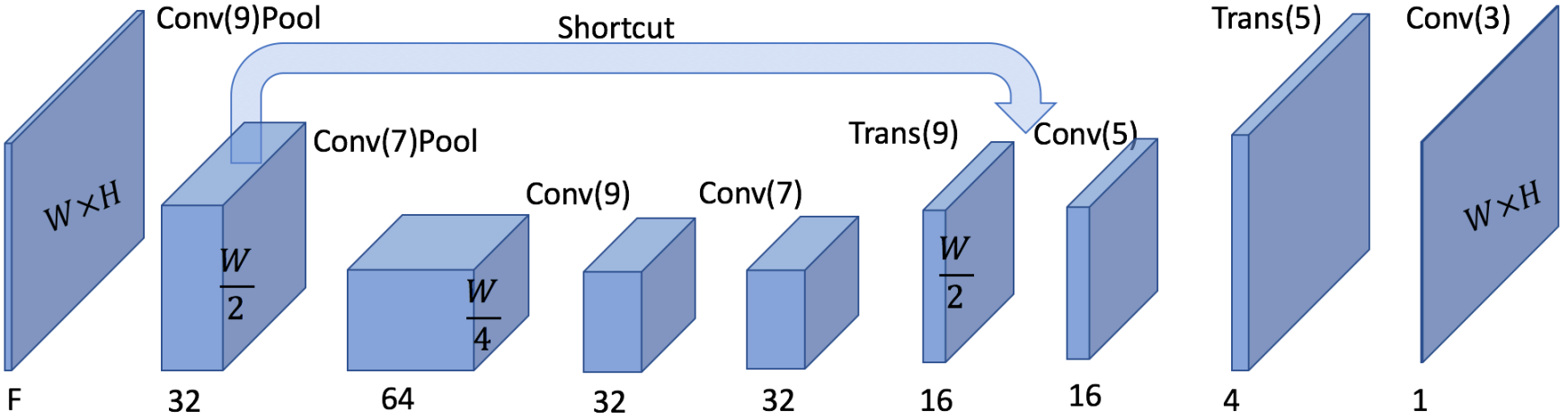
# Fully Convolutional Network (FCN) for Hotspot Detection




Eliminate FC layers
May use transposed-convolutional to up-sample
Used in image segmentation, object detection

Image from Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2017. Fully Convolutional Networks for Semantic Segmentation. (TPAMI)

# FCN Architecture for Hotspot Detection




Filter size indicated in ()

# Experiment Setup

- Five designs from ISPD 2015 placement contest
- ~300 different placements by placing macros in different ways
- Placement, routing and DRC are done by Cadence tool
- When a circuit is tested, the model trained with the other circuits
- SVM and Logistic Regression (LR) methods for comparison

| Circuit Name | #Macros | #Cells | #Nets | Width (μm) | #Placements |
|---|---|---|---|---|---|
| des_perf | 4 | 108288 | 110283 | 900 | 600 |
| edit_dist | 6 | 127413 | 131134 | 800 | 300 |
| fft | 6 | 30625 | 32088 | 800 | 300 |
| matrix_mult_a | 5 | 149650 | 154284 | 1500 | 300 |
| matrix_mult_b | 7 | 146435 | 151614 | 1500 | 300 |

# #DRV Prediction Fidelity

- How methods recognize placements with the lowest #DRV level ($c_0$)
- The quality of placements selected by each method
  - The best rank of top ten placements predicted to have least #DRV

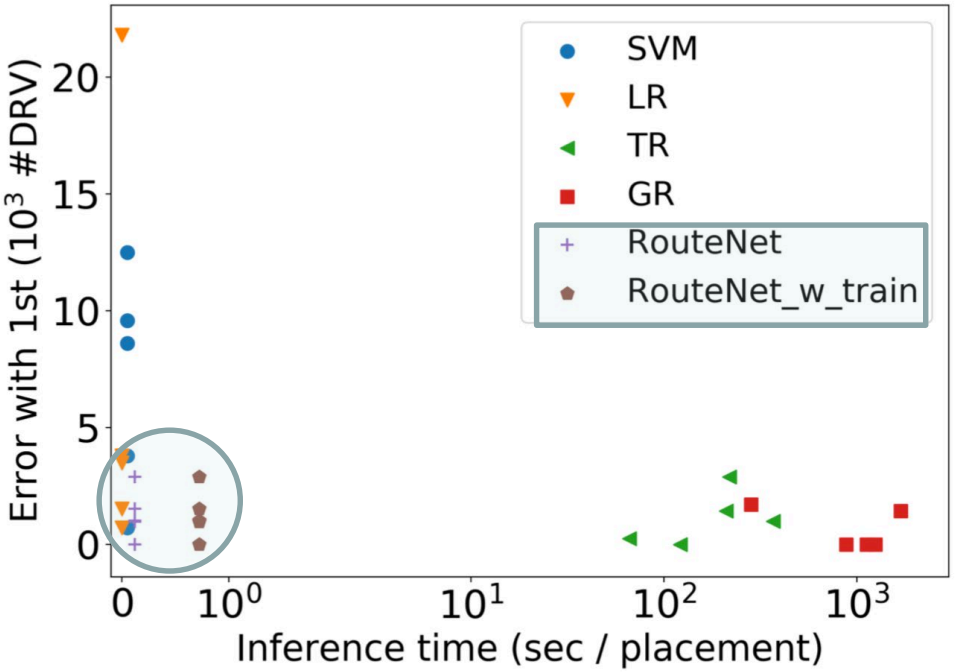
| Circuit Name | $c_0/c_1+c_2+c_3$ accuracy (%) | | | | | Best rank in top 10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SVM | LR | TR | GR | Route Net | SVM | LR | TR | GR | Route Net |
| des_perf | 63 | 74 | 80 | 77 | 80 | $87^{th}$ | $15^{th}$ | $2^{nd}$ | $1^{st}$ | $2^{nd}$ |
| edit_dist | 69 | 68 | 78 | 77 | 76 | $17^{th}$ | $17^{th}$ | $3^{rd}$ | $3^{rd}$ | $2^{nd}$ |
| fft | 66 | 62 | 73 | 70 | 75 | $6^{th}$ | $6^{th}$ | $2^{nd}$ | $33^{rd}$ | $1^{st}$ |
| matrix_mult_a | 66 | 65 | 78 | 74 | 72 | $30^{th}$ | $5^{th}$ | $1^{st}$ | $1^{st}$ | $5^{th}$ |
| matrix_mult_b | 63 | 62 | 76 | 73 | 76 | $22^{nd}$ | $93^{rd}$ | $4^{th}$ | $1^{st}$ | $4^{th}$ |
| Average | 65 | 66 | 77 | 74 | 76 | $32^{nd}$ | $27^{th}$ | $2^{nd}$ | $8^{th}$ | $3^{rd}$ |

Our method

TR: Trial Routing
GR: Global Routing

# #DRV Prediction Error and Runtime

- Y: gap between the ‘best in 10’ and the actually 1st-ranked placement with least #DRV
- X: inference time taken for each method
- RouteNet achieves low inference time and high accuracy at the same time



# DRV Hotspot Detection Evaluation

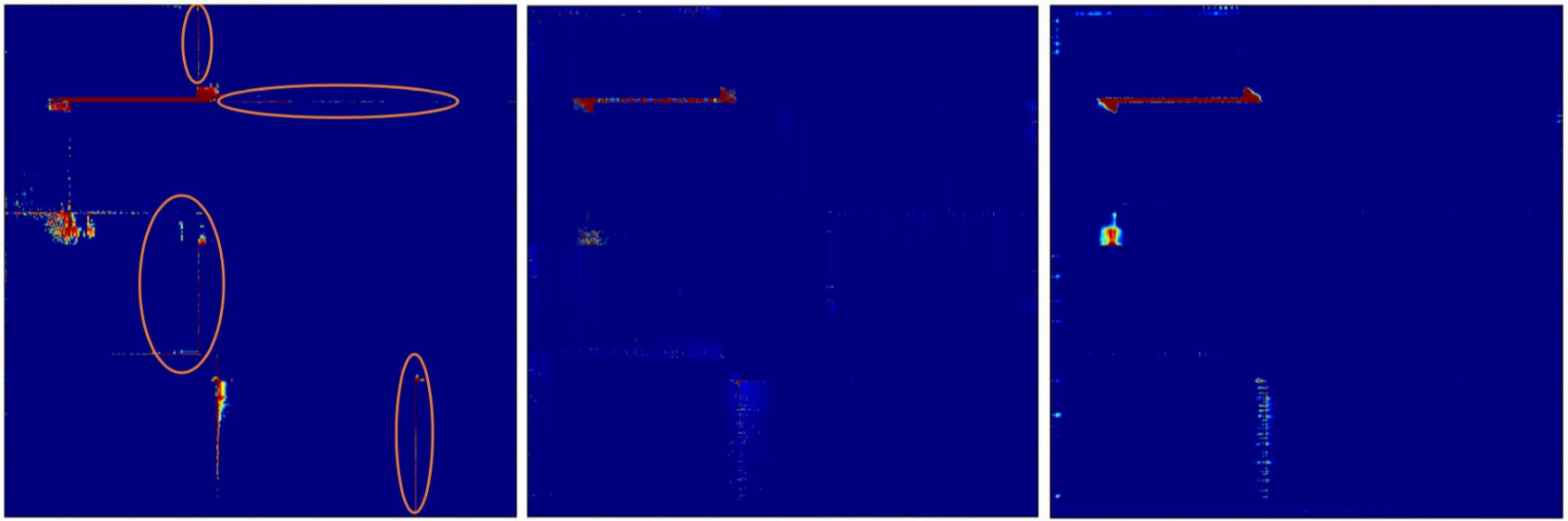
- Same decision threshold is used for all designs
- Slight different FPR, but all under 1%
- RouteNet is superior to all methods and improves global routing accuracy by 50%

| Circuit Name | FPR (%) | TPR (%) | | | | |
|---|---|---|---|---|---|---|
| | | TR | GR | LR | SVM | RouteNet |
| des_perf | 0.54 | 17 | 56 | 54 | 42 | 74 |
| edit_dist | 1.00 | 25 | 36 | 38 | 28 | 64 |
| fft | 0.30 | 21 | 45 | 54 | 31 | 71 |
| matrix_mult_a | 0.21 | 13 | 30 | 34 | 12 | 49 |
| matrix_mult_b | 0.24 | 13 | 37 | 41 | 20 | 53 |
| Average | 0.46 | 18 | 41 | 44 | 27 | 62 |

| | | Prediction Result: Positive | Prediction Result: Negative | Evaluation |
|---|---|---|---|---|
| Label | Positive | $TP$ | $FN$ | $TPR = \frac{TP}{TP + FN}$ |
| | Negative | $FP$ | $TN$ | $FPR = \frac{FP}{FP + TN}$ |

TPR (True Positive Rate)
FPR (False Positive Rate)

# DRC Hotspot Detection Demonstration

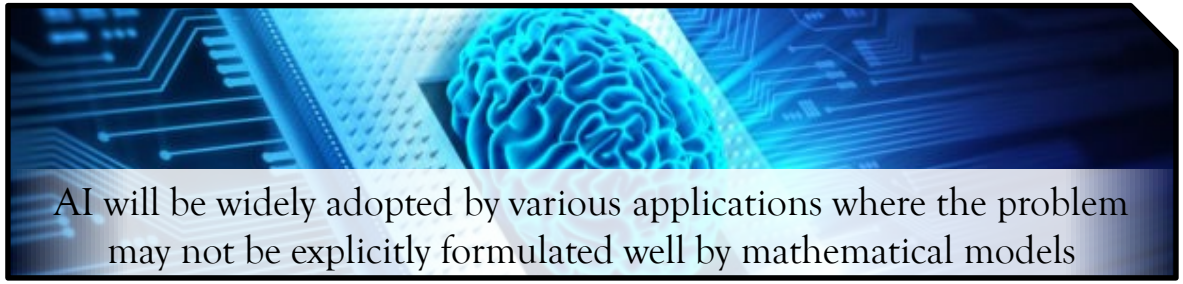


# Our Perspective

1

AI is going mainstream, showing potential on both the cloud and edge, however, is limited by infrastructure

2



AI will be widely adopted by various applications where the problem may not be explicitly formulated well by mathematical models

3

Future AI will be more user friendly, more automatic, more cost efficient