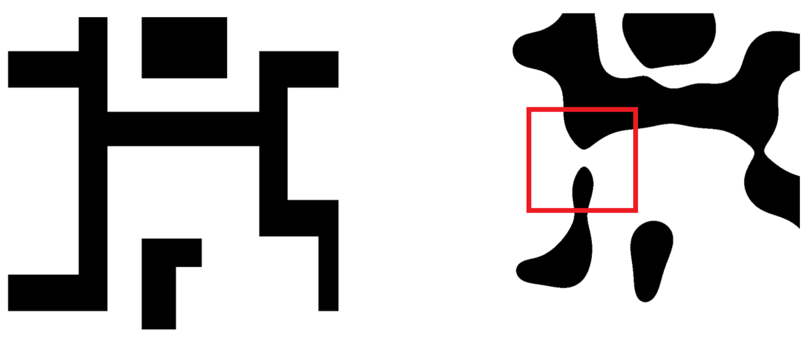
Efficient Layout Hotspot Detection via Binarized Residual Neural Network

Yiyang Jiang[1], Fan Yang[1*], Hengliang Zhu[1], Bei Yu[3], Dian Zhou[2], Xuan Zeng[1*]

[1]State Key Lab of ASIC & System, Microelectronics Department, Fudan University
[2]University of Texas at Dallas
[3]Chinese University of Hong Kong

Background



- What you see ≠ what you get
- RETs: OPC, SRAF, MPL
- Still exists hotspots: low fidelity patterns
- Lithography simulation: time consuming

Pattern Matching-based Hotspot Detection

- Characterize the hotspots as explicit patterns and identify the hotspots by matching these patterns
- [Yu+,ICCAD'14] [Nosato+,JM3'14] [Kahng+,SPIE'06] [Su+,TCAD'15] [Wen+,TCAD'14] [Yang+,TCAD'17]
- Fast but hard to detect unseen patterns

Machine Learning based Hotspot Detection

- Build implicit models by learning from existing training data
  - SVM, Bayesian, Decision-tree, Boosting, NN, ...
- [Ding+,ASPDAC'11] [Yu+,DAC'13] [Matsunawa+,SPIE'15] [Zhang+,ICCAD'16] [Wen+,TCAD'14]
- Possible to detect the unseen hotspots but may cause false alarm issues

Deep Learning based Hotspot Detection

- Belongs to ML-based hotspot detection but different from conventional ML models:
  - Feature Crafting v.s. Feature Learning
  - Stronger scalability
- [Yang+,DAC'17]
- Drawback: not storage and computational efficient

Preliminaries

Problem Formulation

Definition: Accuracy

The ratio of correctly predicted hotspots among the set of actual hotspots.

$$Accuracy = \frac{\#TP}{\#TP + \#FN}$$

Definition: False Alarm

The number of incorrectly predicted non-hotspots.

$$False\ Alarm = \#FP$$

Definition: ODST

Abbreviation of Overall Detection and Simulation Time. The sum of the lithography simulation time for layout patterns detected as hotspots (including real hotspots and false alarms) and the learning model evaluation time.

$$ODST = (\#FP + \#TP)\ t_{ls} + (\#TN + \#FN + \#FP + \#TP)\ t_{ev}$$

where t_{ls} is lithography simulation time per instance and t_{ev} is the model evaluation time per instance.

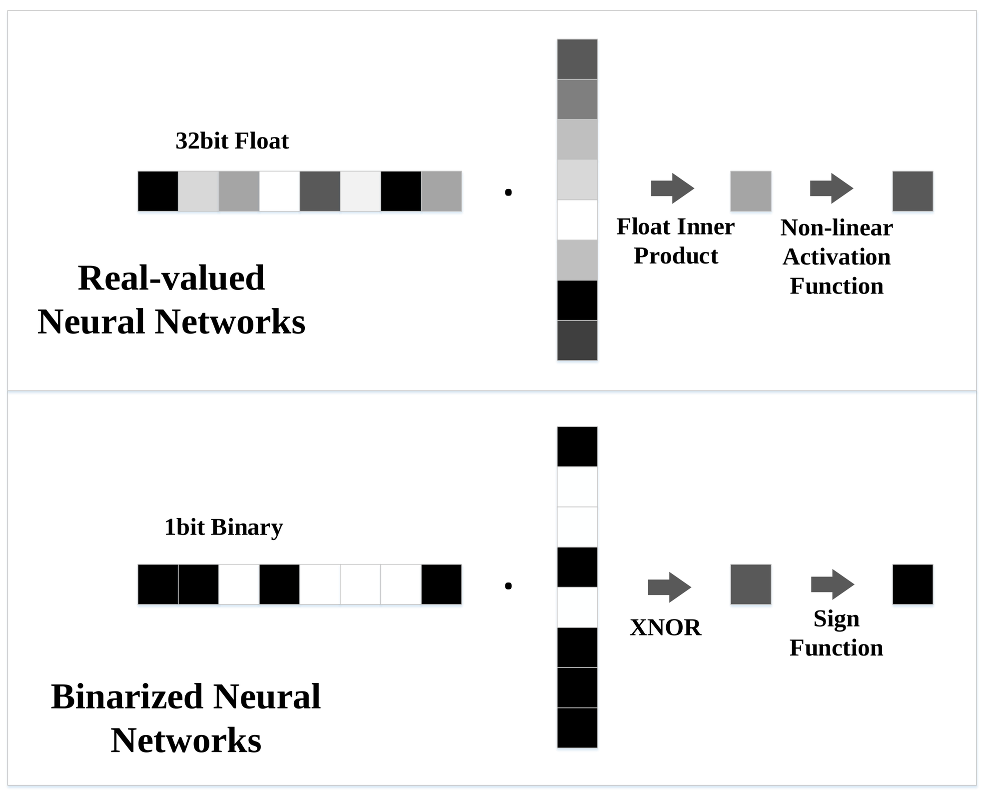
Problem1: Hotspot Detection

Given a dataset that contains hotspot and non-hotspot instances, train a classifier that can maximize the $accuracy$ and minimize the $false\ alarm$.

Parameter Quantization

- Problem with deep neural networks:
  - Enormous computational and storage consumption
- To alleviate this problem:
  - Parameter Quantization
  - 32-bit floating-point weights not necessary: quantized to fixed-point of 8-bit, 3-bit, 1-bit…
  - [Arora+,ICML'14] [Hwang+,SiPS'14] [Soudry+,ANIPS'14] [Rastegari+,ECCV'16]

Binarized Neural Networks



- Binarized neural network (BNN):
  - Extremely quantized to 1 bit
  - Inherently suitable for hardware implementation
- Layout patterns are binary images
  - BNN might be suitable for that

Binarization Approach

Definition

Let W be the kernel which is an n-element vector and X be the vector of the corresponding block in the input tensor, $n = w_k \times h_k$. Let W_B, X_B be the binarized kernel and input vector and α_W, α_X be the corresponding scaling factors. Here $W, X \in \mathbb{R}^n$, $W_B, X_B \in \{-1, +1\}^n$ and $\alpha_W, \alpha_X \in \mathbb{R}^+$.

Problem2: Binarization

Given the kernel and input vector W, X, find best $W_B, X_B, \alpha_W, \alpha_X$ that minimizes the binarization loss L_i.
$L_i(W_B, X_B, \alpha_W, \alpha_X) = \|W \odot X - \alpha_W W_B \odot \alpha_X X_B\|^2$
where $\odot$ means inner product.

- Solving the minimization problem:

$$W_B^* = sign(W), X_B^* = sign(X)$$
$$\alpha_W^* = \frac{1}{n}\|W\|_{l1},\ \alpha_X^* = \frac{1}{n}\|X\|_{l1}$$

- The estimated weight and corresponding input vector $\widetilde{W}, \widetilde{X}$ are:

$$\widetilde{W} = \frac{1}{n} sign(W)\|W\|_{l1}$$
$$\widetilde{X} = \frac{1}{n} sign(X)\|X\|_{l1}$$

Training BNN

- Gradient for $sign$ function [Hubara, 2016]

$$\frac{\partial sign(x)}{\partial x} = \mathbf{1}_{\|W\|<1}$$

- Back propagation through the Binarized Layer

$$\frac{\partial l}{\partial W} = \frac{\partial l}{\partial \widetilde{W}} \frac{\partial \widetilde{W}}{\partial W} = \frac{\partial l}{\partial \widetilde{W}} \frac{\partial(\frac{1}{n}\|W\|_{l1} sign(W))}{\partial W} = \frac{\partial l}{\partial \widetilde{W}} (\frac{1}{n} + \alpha_W^* \mathbf{1}_{\|W\|<1})$$

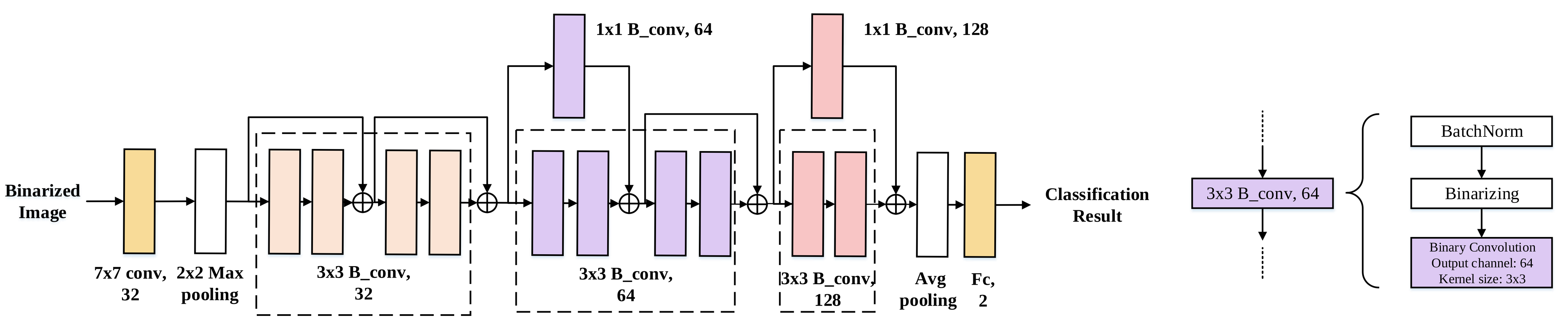
Algorithm 1 Training a BNN

Input: $(\mathcal{T}_0, Y)$: a minibatch of input tensors and labels;
$l(Y, Y_{out})$: loss function;
$\mathcal{W}^t$: current real-valued weight;
L: number of layers;
n: kernel size of layers;
η^t: current learning rate;
Output: $\mathcal{W}^{t+1}$: updated real-valued weight; η^{t+1}: updated learning rate.
1. Forward Process:
for $k = 1$ to L **do**
  $\mathcal{B}_k^t =$ **BinarizeWeight**$(\mathcal{W}_k^t)$
  $\mathcal{T}_{k+1} =$ **BinarizeInput**(**BatchNorm**$(\mathcal{T}_k)) \circledast \mathcal{B}_k^t$
end for
$Y_{out} = \mathcal{T}_{L+1}$
2. Backward Process:
for $k = L$ to 1 **do**
  $\frac{\partial l}{\partial \mathcal{T}_k} =$ **BinaryBackward**$(\frac{\partial l}{\partial \mathcal{T}_{k+1}}, \mathcal{T}_k)$
  $\frac{\partial l}{\partial \mathcal{B}_k^t} =$ **BinaryBackward**$(\frac{\partial l}{\partial \mathcal{T}_{k+1}}, \mathcal{B}_k^t)$
  $\frac{\partial l}{\partial \mathcal{W}_k^t} = \frac{1}{n_l}(1 + \|\mathcal{W}_k^t\|_{l1} \mathbf{1}_{\|\mathcal{W}_k^t\|<1}) \frac{\partial l}{\partial \mathcal{B}_k^t}$
end for
3. Update Parameters:
$\mathcal{W}^{t+1}, \eta^{t+1} =$ **Update**$(\mathcal{W}^t, \frac{\partial l}{\partial \mathcal{W}^t}, \eta^t)$
return $\mathcal{W}^{t+1}, \eta^{t+1}$

- Data preprocessing
  - Down-sampled to 128×128
- Training hyper-parameters
  - Batch size:128
  - Learning rate: Initial 0.15, exponentially decay each time loss plateaus
  - Optimizer: NAdam optimizer [Dozat, 2016]
  - Initializer: Xavier initializer [Glorot, 2010]

Network Architecture

Residual block-based architecture **Typical BNN block structure**



Experimental Results

- Benchmark: ICCAD 2012 Contest

Benchmark	# Train HS	# Train NHS	# Test HS	# Test NHS
ICCAD	1204	17096	2524	13503

- Experimental Results on ICCAD 2012 Contest

Method	Accuracy (%)	False Alarm #	Runtime (s)
SPIE'15	84.2	2919	2672
ICCAD'16	97.7	4497	1052
DAC'17	98.2	3413	482
Ours	**99.2**	**2787**	**60**

- Performance Comparisons with Previous Hotspot Detectors
  - Accuracy improved from 84.2% to 99.2%
  - Least False Alarms: 2787
  - Lowest Runtime: 60s, 8x faster

Conclusions

- Propose a BNN-based architecture to speed up the neural networks in hotspot detection
- Our architecture outperforms previous hotspot detectors and achieves an 8× speedup over the best deep learning-based hotspot detector.