Imbalance Aware Lithography Hotspot Detection: A Deep Learning Approach

Haoyu Yang\textsuperscript{1}, Luyang Luo\textsuperscript{1}, Jing Su\textsuperscript{2}, Chenxi Lin\textsuperscript{2}, Bei Yu\textsuperscript{1}

\textsuperscript{1}The Chinese University of Hong Kong
\textsuperscript{2}ASML Brion Inc.

Mar. 1, 2017
Outline

Introduction

Network Architecture

Imbalance Aware Learning

Experimental Results
Outline

Introduction

Network Architecture

Imbalance Aware Learning

Experimental Results
Moore’s Law to Extreme Scaling
1983年
1.2μm

1994年
0.35μm

1997年
0.25μm

1999年
0.18μm

2001年
0.13μm

2004年
90nm

2007年
65nm

2010年
45nm

2013年
32nm

2016年
22nm

SPIE.
CONNECTING MINDS.
ADVANCING LIGHT.
Lithography Hotspot Detection

- What you see \( \neq \) what you get
- Even w. RET: OPC, SRAF, MPL
- Still hotspot: low fidelity patterns
- Simulations: extremely CPU intensive

![Graph showing the ratio of lithography simulation time (normalized by 40nm node) vs technology node with a significant time reduction for 20nm technology.](image)
Layout Verification Hierarchy

- **Sampling**: scan and rule check each region
- **Hotspot Detection**: verify the sampled regions and report potential hotspots
- **Lithography Simulation**: final verification on the reported hotspots
Pattern Matching based Hotspot Detection

- Fast and accurate
- [Yu+, ICCAD’14] [Nosato+, JM3’14] [Su+, TCAD’15]
- Fuzzy pattern matching [Wen+, TCAD’14]
- Hard to detect non-seen pattern
Pattern Matching based Hotspot Detection

- Fast and accurate
- [Yu+, ICCAD’14] [Nosato+, JM3’14] [Su+, TCAD’15]
- Fuzzy pattern matching [Wen+, TCAD’14]
- Hard to detect non-seen pattern
Machine Learning based Hotspot Detection

- Extract layout features
- Hotspot detection model
- Classification

- Predict new patterns
  - Decision-tree, ANN, SVM, Boosting...

- [Drmanac+, DAC’09]
- [Ding+, TCAD’12]
- [Yu+, JM3’15]
- [Matsunawa+, SPIE’15]
- [Yu+, TCAD’15]
- [Zhang+, ICCAD’16]
Machine Learning based Hotspot Detection

- Predict new patterns
- Decision-tree, ANN, SVM, Boosting ...
  - [Drmanac+, DAC’09] [Ding+, TCAD’12] [Yu+, JM3’15] [Matsunawa+, SPIE’15]
  - [Yu+, TCAD’15] [Zhang+, ICCAD’16]
- Crafted features are not satisfactory
- Hard to handle ultra-large datasets.
Why Deep Learning?

▶ Feature Crafting v.s. Feature Learning
Although prior knowledge is considered during manually feature design, information loss is inevitable. Feature learned from mass dataset is more reliable.

▶ Scalability
With shrinking down circuit feature size, mask layout becomes more complicated. Deep learning has the potential to handle ultra-large-scale instances while traditional machine learning may suffer from performance degradation.

▶ Mature Libraries
Caffe [Jia+, ACM MMM’14] and Tensorflow [Martin+, TR’15]
Deep Learning has been widely applied in object recognition tasks. Nature of mask layout impedes the availability of existing frameworks.

- **Imbalanced Dataset**
  Lithographic hotspots are always the minority.

- **Larger Image Size**
  Effective clip region (> 1000 × 1000 pixels) is much larger than the image size in traditional computer vision problems.

- **Sensitive to Scaling**
  Scaling of mask layout patterns modifies its attributes.
Deep Learning based Hostpot Detection Flow

- Training Data Set
- Validation
- Testing Data Set
- Upsampling
- Random Mirroring
- Training
- Trained Model
- Accuracy
- False Alarm
- Model Testing
Outline

Introduction

Network Architecture

Imbalance Aware Learning

Experimental Results
CNN Architecture Overview

- Convolution Layer
- Rectified Linear Unit (ReLU)
- Pooling Layer
- Fully Connected Layer
Convolution Layer

**Convolution Operation:**

\[
I \otimes K(x, y) = \sum_{i=1}^{c} \sum_{j=1}^{m} \sum_{k=1}^{m} I(i, x - j, y - k)K(j, k)
\]
Convolution Layer (cont.)

Effect of different convolution kernel sizes:

- (a) $7 \times 7$
- (b) $5 \times 5$
- (c) $3 \times 3$

<table>
<thead>
<tr>
<th>Kernel Size</th>
<th>Padding</th>
<th>Test Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>$7 \times 7$</td>
<td>3</td>
<td>87.50%</td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>2</td>
<td>93.75%</td>
</tr>
<tr>
<td>$3 \times 3$</td>
<td>1</td>
<td>96.25%</td>
</tr>
</tbody>
</table>

*Stop after 5000 iterations.
Rectified Linear Unit

- Alleviate overfitting with sparse feature map
- Avoid gradient vanishing problem

<table>
<thead>
<tr>
<th>Activation Function</th>
<th>Expression</th>
<th>Validation Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU</td>
<td>max{x, 0}</td>
<td>0.16</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>\frac{1}{1+\exp(-x)}</td>
<td>87.0</td>
</tr>
<tr>
<td>TanH</td>
<td>\exp(2x)−1 \exp(2x)+1</td>
<td>0.32</td>
</tr>
<tr>
<td>BNLL</td>
<td>log(1 + \exp(x))</td>
<td>87.0</td>
</tr>
<tr>
<td>WOAF</td>
<td>NULL</td>
<td>87.0</td>
</tr>
</tbody>
</table>
Pooling Layer

- Extracts the local region statistical attributes in the feature map

(a) max pooling

(b) avg pooling
Pooling Layer (cont.)

- Translation invariant (✗)
- Dimension reduction

Effect of pooling methods:

<table>
<thead>
<tr>
<th>Pooling Method</th>
<th>Kernel</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>2 × 2</td>
<td>96.25%</td>
</tr>
<tr>
<td>Ave</td>
<td>2 × 2</td>
<td>96.25%</td>
</tr>
<tr>
<td>Stochastic</td>
<td>2 × 2</td>
<td>90.00%</td>
</tr>
</tbody>
</table>
Fully Connected Layer

- Fully connected layer transforms high dimension feature maps into flattened vector.
Fully Connected Layer (cont.)

- A percentage of nodes are **dropped out** (i.e. set to zero)
- Avoid overfitting

**Effect of dropout ratio:**

<table>
<thead>
<tr>
<th>Dropout Ratio</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>0.5</td>
<td>95.00</td>
</tr>
<tr>
<td>1</td>
<td>90.00</td>
</tr>
</tbody>
</table>

Convolutional Hidden Layers:

- C5-3
- P5

Dimensions:

- 16x16x32
- 2048
- 512
- 19/34

Diagram showing the structure of the network with layers and dropout rates.
Fully Connected Layer (cont.)

- A percentage of nodes are dropped out (i.e. set to zero)
- Avoid overfitting

Effect of dropout ratio:

<table>
<thead>
<tr>
<th>Dropout Ratio</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>0.5</td>
<td>95.00</td>
</tr>
<tr>
<td>1</td>
<td>90.00</td>
</tr>
</tbody>
</table>

Convolutional Hidden Layers:

- C5-3
- P5
- 16x16x32
- 2048
- 512
Architecture Summary

- Total 21 layers with 13 convolution layers and 5 pooling layers.
- A ReLU is applied after each convolution layer.
## Architecture Summary

<table>
<thead>
<tr>
<th>Layer</th>
<th>Kernel Size</th>
<th>Stride</th>
<th>Padding</th>
<th>Output Vertexes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1-1</td>
<td>$2 \times 2 \times 4$</td>
<td>2</td>
<td>0</td>
<td>$512 \times 512 \times 4$</td>
</tr>
<tr>
<td>Pool1</td>
<td>$2 \times 2$</td>
<td>2</td>
<td>0</td>
<td>$256 \times 256 \times 4$</td>
</tr>
<tr>
<td>Conv2-1</td>
<td>$3 \times 3 \times 8$</td>
<td>1</td>
<td>1</td>
<td>$256 \times 256 \times 8$</td>
</tr>
<tr>
<td>Conv2-2</td>
<td>$3 \times 3 \times 8$</td>
<td>1</td>
<td>1</td>
<td>$256 \times 256 \times 8$</td>
</tr>
<tr>
<td>Conv2-3</td>
<td>$3 \times 3 \times 8$</td>
<td>1</td>
<td>1</td>
<td>$256 \times 256 \times 8$</td>
</tr>
<tr>
<td>Pool2</td>
<td>$2 \times 2$</td>
<td>2</td>
<td>0</td>
<td>$128 \times 128 \times 8$</td>
</tr>
<tr>
<td>Conv3-1</td>
<td>$3 \times 3 \times 16$</td>
<td>1</td>
<td>1</td>
<td>$128 \times 128 \times 16$</td>
</tr>
<tr>
<td>Conv3-2</td>
<td>$3 \times 3 \times 16$</td>
<td>1</td>
<td>1</td>
<td>$128 \times 128 \times 16$</td>
</tr>
<tr>
<td>Conv3-3</td>
<td>$3 \times 3 \times 16$</td>
<td>1</td>
<td>1</td>
<td>$128 \times 128 \times 16$</td>
</tr>
<tr>
<td>Pool3</td>
<td>$2 \times 2$</td>
<td>2</td>
<td>0</td>
<td>$64 \times 64 \times 16$</td>
</tr>
<tr>
<td>Conv4-1</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$64 \times 64 \times 32$</td>
</tr>
<tr>
<td>Conv4-2</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$64 \times 64 \times 32$</td>
</tr>
<tr>
<td>Conv4-3</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$64 \times 64 \times 32$</td>
</tr>
<tr>
<td>Pool4</td>
<td>$2 \times 2$</td>
<td>2</td>
<td>0</td>
<td>$32 \times 32 \times 32$</td>
</tr>
<tr>
<td>Conv5-1</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$32 \times 32 \times 32$</td>
</tr>
<tr>
<td>Conv5-2</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$32 \times 32 \times 32$</td>
</tr>
<tr>
<td>Conv5-3</td>
<td>$3 \times 3 \times 32$</td>
<td>1</td>
<td>1</td>
<td>$32 \times 32 \times 32$</td>
</tr>
<tr>
<td>Pool5</td>
<td>$2 \times 2$</td>
<td>2</td>
<td>0</td>
<td>$16 \times 16 \times 32$</td>
</tr>
<tr>
<td>FC1</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>2048</td>
</tr>
<tr>
<td>FC2</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>512</td>
</tr>
<tr>
<td>FC3</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>2</td>
</tr>
</tbody>
</table>
Outline

Introduction

Network Architecture

Imbalance Aware Learning

Experimental Results
Minority Upsampling

Layout datasets are highly imbalanced as after resolution enhancement techniques (RETs) the lithographic hotspots are always the minority.

![Bar chart showing percentage of non-hotspot and hotspot for ICCAD-1 to ICCAD-5]
Minority Upsampling

Layout datasets are highly imbalanced as after resolution enhancement techniques (RETs) the lithographic hotspots are always the minority.

- Multi-label learning [Zhang+, IJCAI’15]
- Majority downsampling [Ng+, TCYB’15]
- Pseudo instance generation [He+, IJCNN’08]

Artificially generated instances might not be available because of mask layout nature.
Minority Upsampling

Layout datasets are highly imbalanced as after resolution enhancement techniques (RETs) the lithographic hotspots are always the minority.

- Multi-label learning
  [Zhang+, IJCAI’15]
- Majority downsampling
  [Ng+, TCYB’15]
- Pseudo instance generation
  [He+, IJCNN’08]

Artificially generated instances might not be available because of mask layout nature.

- Naïve upsampling (√)
  1. Gradient descent
  2. Insufficient training samples
Random Mirror Flipping

- Before fed into neural network
- Each instance is taking one of 4 orientations
- Resolve insufficient data
Validation performance does not show further improvement when the upsampling factor increases beyond a certain value.
Learning Rate

$\gamma$: defines how fast the neuron weights are updated

$$w_i = w_i - \gamma \frac{\partial l}{\partial w_i}.$$
Momentum and Weight Decay

▶ Momentum
Physical meaning is involved into gradient descent.

\[ v = \mu v - \gamma \frac{\partial l}{\partial w_i}, \]
\[ w_i = w_i + v. \]

▶ Weight Decay
An alternative to achieve $L_2$ regularization on neuron weights.

\[ v = \mu v - \gamma \frac{\partial l}{\partial w_i} - \gamma \lambda w_i, \]
\[ w_i = w_i + v. \]
Momentum and Weight Decay (cont.)

- **Momentum Effects:**

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>Learning Rate</th>
<th>Validation Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.001</td>
<td>0.21</td>
</tr>
<tr>
<td>0.9</td>
<td>0.001</td>
<td>0.22</td>
</tr>
<tr>
<td>0.95</td>
<td>0.001</td>
<td>0.21</td>
</tr>
<tr>
<td>0.99</td>
<td>0.001</td>
<td><strong>0.16</strong></td>
</tr>
</tbody>
</table>

- **Weight Decay Effects:**

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Learning Rate</th>
<th>Momentum</th>
<th>Validation Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>0.001</td>
<td>0.99</td>
<td>0.95</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>0.001</td>
<td>0.99</td>
<td>1.19</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>0.001</td>
<td>0.99</td>
<td>0.37</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>0.001</td>
<td>0.99</td>
<td><strong>0.2</strong></td>
</tr>
</tbody>
</table>
Weight Initialization

The weight initialization procedure determines what initial values assigned to each neuron before the gradient descent update starts.

- Random Gaussian (X)
  Cannot guarantee input & output have similar variance.
Weight Initialization

The **weight initialization** procedure determines what initial values assigned to each neuron before the gradient descent update starts.

- **Random Gaussian (X)**
  Cannot guarantee input & output have similar variance.

- **Xavier [Xavier+, AISTATS’10]**
  Initialized weights are determined by input node number.

\[
\hat{V}(w_i) = \frac{1}{N}.
\]
Outline

Introduction

Network Architecture

Imbalance Aware Learning

Experimental Results
Experimental Setup

- Based on Caffe [Jia+, ACMMM’14]
- Evaluated on ICCAD-2012 CAD contest benchmark

Evaluation metrics:

**Accuracy**

The ratio between the number of correctly detected hotspot clips and the number of all hotspot clips.

**ODST**

The sum of all lithographic simulation time for false alarm† and the deep learning model testing time.

$$\text{ODST} = \text{Test Time} + 10s \times \# \text{ of False Alarm}$$

†False alarm: the number of non-hotspot clips that are reported as hotspots by detector.
Layer Visualization
Compare Accuracy with State-of-the-Art‡

‡JM3'16: CNN based; TCAD'15: SVM based; ICCAD'16: Boosting based.
Compare ODST with State-of-the-Art

- Improve the performance of ODST by at least $24.80\%$ on average.

JM3’16: CNN based; TCAD’15: SVM based; ICCAD’16: Boosting based.
Conclusion

We explore the feasibility of deep learning as an alternative approach for hotspot detection.

- Hotspot-detection-oriented hyper-parameter tuning
- Imbalance Issue: Upsampling & Random mirror flipping
- Outperform state-of-the-art solutions
Conclusion

We explore the feasibility of deep learning as an alternative approach for hotspot detection.

▶ Hotspot-detection-oriented hyper-parameter tuning
▶ Imbalance Issue: Upsampling & Random mirror flipping
▶ Outperform state-of-the-art solutions

Future Works

▶ Test on larger scale test cases
▶ Further simplify architecture to speedup
▶ Seek other VLSI layout applications (e.g., OPC, SRAF)
Thank You

Haoyu Yang (hyyang@cse.cuhk.edu.hk)
Luyang Luo (lyluo4@cse.cuhk.edu.hk)
Jing Su (jing.su@asml.com)
Chenxi Lin (chenxi.lin@asml.com)
Bei Yu (byu@cse.cuhk.edu.hk)