Differentiable Computational Lithography Framework

Guojin Chen$^1,^2$, Hao Geng$^3$, Bei Yu$^1$, David Z. Pan$^2$

$^1$CUHK, $^2$UT Austin, $^3$ShanghaiTech

gjchen@utexas.edu

March 7, 2024
Outline

1 Background
   1.1 Differentiable Programming
   1.2 Lithography Simulation

2 Differentiable Lithography
   2.1 Lithography Modeling
   2.2 Implementation of Differentiable Lithography
   2.3 Composable Differentiable Lithography
Differentiable Programming
Remember derivatives and gradients?

**Derivative**

\[ f : \mathbb{R} \rightarrow \mathbb{R} \]
\[ f'(x) := \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]

**Gradient**

\[ Q(w) = \sum_{i=1}^{N} Q_i(w) \]
\[ w_{t+1} = w_t - \eta \sum_{i=1}^{d} \nabla_w Q_i(w) \]

(Ruder, 2017) [http://ruder.io/optimizing-gradient-descent/]
Automatic Differentiation: careful application of the chain rule to programs.

**Automatic Differentiation** = methods for automatically computing gradients of functions specified by a computer program.
Execute **differentiable code via automatic differentiation**.

**Differentiable programming**: Writing software composed of **differentiable and parameterized building blocks** that are executed via **automatic differentiation** and **optimized** in order to perform a specified task.

1. A **parameterized function** (method / model / building blocks) to be optimized;
2. Automatic differentiability of the function to be **optimized**.
3. A **loss** to measure performance;

**differentiable programming** = programming languages + **automatic differentiation**.

---

**Diagram**:
- **Data Preprocessing**
- **Computation graph rules**
- **Unit tests**
- **Feed input**
- **Output**
- **Loss**
- **\( \nabla \text{Loss} \)**
- **Weights + biases**
- **Backpropagation via automatic diff.**
- **Loop while loss is too high**
Yann LeCun
January 5, 2018 · 🌏

OK, Deep Learning has outlived its usefulness as a buzz-phrase. Deep Learning est mort. Vive Differentiable Programming!

Andrej Karpathy
@karpathy · Follow

Gradient descent can write code better than you. I'm sorry.

3:56 PM · Aug 4, 2017

❤️ 2.7K · Reply · Share
Differentiable programming: Software 2.0

Software 2.0 from Andrej Karpathy¹

AI is eating software from Jensen Huang²

¹Software 2.0: https://karpathy.medium.com/software-2-0-a64152b37c35
²AI: https://www.technologyreview.com/ai-is-going-to-eat-software/
Background of Lithography
Lithography

3

The scalar imaging equation under partially coherent illumination

\[ I(x_1, y_1) = J_I((x_1, y_1), (x_1, y_1)) \]

\[ = \int \int \int \int_{-\infty}^{\infty} J_C(x_0 - x'_0, y_0 - y'_0) O(x_0, y_0) O^*(x'_0, y'_0) \]

\[ H(x_1 - x_0, y_1 - y_0) H^*(x_1 - x'_0, y_1 - y'_0) \, dx_0 \, dy_0 \, dx'_0 \, dy'_0, \]  

(1)

where \( O \) is the object function, the field of the photomask in the lithography case, \( H \) is the projector transfer function, and \( J_C \) is the mutual intensity, a weight factor, of two points under extended source conditions.

Conclusion

The intensity at a point in the image plane is given by the propagation of the mutual intensity of all contributing points, that is, of all points that lay in the support of the projection system and the illuminator.
Abbe’s VS Hopkins’

• Abbe’s approach
  • illumination cross-coefficients (ICC)

\[
ICC(x, y; f, g) = \left| \int_{-\infty}^{\infty} H(f + f', g + g') \mathcal{F}(M)(f', g') \exp(-j2\pi(f'x + g'y)) \, df' \, dg' \right|^2.
\]

• Abbe’s approach

\[
I(x, y) = \int_{-\infty}^{\infty} I(f, g) ICC(x, y; f, g) \, df \, dg.
\]

• Hopkins’ approach
  • TCC

\[
I(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{T}(f', g'; f'', g'') \mathcal{F}(M)(f', g') \mathcal{F}(M)^*(f'', g'') \exp(-j2\pi((f' - f'')x + (g' - g'')y)) \, df' \, dg' \, df'' \, dg''.
\]
Abbe’s VS Hopkins’

<table>
<thead>
<tr>
<th>Visualization</th>
<th>Abbe</th>
<th>Hopkins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source J</td>
<td><img src="image1" alt="Abbe Diagram" /></td>
<td><img src="image2" alt="Hopkins Diagram" /></td>
</tr>
<tr>
<td>Mask M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projector H</td>
<td></td>
<td>TCC Kernel Approximation</td>
</tr>
<tr>
<td>Wafer Z</td>
<td></td>
<td>Mask M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Abbe</th>
<th>Hopkins</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^6)$</td>
<td>Can be accelerated using parallel computing, or compressive sensing.</td>
<td>$O(n^4)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Abbe</th>
<th>Hopkins</th>
</tr>
</thead>
</table>
What’s next?
Forward Lithography

Wafer → Projection Lens → Mask → Lens → Source
Computational Lithography

Wafer → Projection Lens → Mask → Lens → Source
Differentiability without surrogates

Inputs $x \in \mathbb{R}^d$ → Parameters $\psi \in \mathbb{R}^n$ → Outputs $F(x, \psi)$

Non-differentiable simulator (model)
• Use automatic differentiation tools to make the simulator directly differentiable.
How can we implement differentiable lithography?

- Complex lithography setups can be composed of a pipeline of a series of distinct modules \textit{i.e.}, source, lens, mask, aerial.

- One might need to differentiate through the whole end-to-end pipeline, which can be achieved by compositionality and the chain rule.

(a) Core components of forward lithography process. (b) The visualization of the differentiable lithography chain.
Differentiable Lithography

- **Differentiable analysis**
  - Unify analysis pipeline by simultaneously optimizing the free parameters of an analysis with respect to the desired physics objective.

- **Differentiable simulation**
  - Enable efficient simulation-based inference, reducing the number of events needed by orders of magnitude.
Differentiable Source Module

```python
class Source:
    # Source data is used for Abbe formulation:
    # Source data is used for Hopkins formulation: Mutual Intensity, TCC calculation.

    def __init__:
        self.
        ma: float = 1.35
        wavelength: float = 193.0
        maskpitch: float = 2000.0
        maskypitch: float = 2000.0
        sigma_out: float = 0.8
        sigma_in: float = 0.6
        smooth_data: float = 0.93
        source_type: str = "annular"
        shiftAngle: float = math.pi / 4,
        openAngle: float = math.pi / 16,
    
    self.ma = ma
    self.wavelength = wavelength
    self.maskpitch = maskpitch
    self.maskypitch = maskypitch
    self.sigma_out = sigma_out
    self.sigma_in = sigma_in
    self.smooth_data = smooth_data
    self.shiftAngle = shiftAngle
    self.openAngle = openAngle
    self.type = source_type

    # Process calculation
    ...
    self.update()
```

1. Init Parameters
2. Calculate source
   - Forward
   - Backward
   Calculate source gradient with respect to source value

https://github.com/TorchOPC/TorchLitho
Differentiable Mask Module

1. Init mask layout / params
2. Calculate mask spectrum / value
   - Forward
   - Backward
   - Calculate gradient with respect to mask params

```python
class Mask:
    def __init__(
        self,
        gds_path: str,
        layername: int = 11,
        pixels_per_um: int = 100,
        xmax=1024,
        ymax=1024,
        x_gridsize=1,
        y_gridsize=1,
    ):  
        self.x_range = [-xmax, xmax]  # nm
        self.y_range = [-ymax, ymax]
        self.x_gridsize = x_gridsize  # nm
        self.y_gridsize = y_gridsize
        self.mask_groups = []
        self.CD = CD
        self.gds_path = gds_path
        self.layername = layername
        self.pixels_per_um = pixels_per_um

        Process calculation
        self.openGDS()
        self.maskfft()
```

https://github.com/TorchOPC/TorchLitho
Differentiable Lens Module

1. Init projection parameters
2. Calculate PSF / TCC

Forward
Backward

Calculate gradient with respect to lens params

https://github.com/TorchOPC/TorchLitho
Adjoint Differentiable Chain

Source  Lens  Mask  Resist

$\theta_i$  $\theta_h$  $\theta_m$  $\theta_r$  $\theta_{net}$

Forward  Backward
Composable Differentiable Lithography

Adjoint Differentiable Chain

Source | Lens | Mask | Resist

$\theta_i$ | $\theta_h$ | $\theta_m$ | $\theta_r$ | $\theta_{net}$

Forward | Backward

Training Set

Pluggable Neural Network

Loss

Composable config

defaults:
- _self_
- source: default.yaml
- lens: default.yaml
- tcc: default.yaml
- mask: default.yaml
- aerial: default.yaml
Differentiable Lithography Applications

Differentiable Chain

$f_i \rightarrow f_h \rightarrow f_m \rightarrow f_r$

Source \hspace{0.5cm} Lens \hspace{0.5cm} Mask \hspace{0.5cm} Aerial

RETs Tasks

- Parameter tuning
- Mask Optimization
- SMO
- Resist modeling

Feedback Parameters
Multi-level optimization framework.

\[
P_n : \quad \theta_n^* = \arg\min_{\theta_n} C_n(\theta_n, U_n, L_n, D_n) \quad \triangleright \text{Level } n \text{ problem}
\]

\[
P_k : \quad \text{s.t. } \theta_k^* = \arg\min_{\theta_k} C_k(\theta_k, U_k, L_k, D_k) \quad \triangleright \text{Level } k \in \{2, \ldots, n-1\} \text{ problem}
\]

\[
P_1 : \quad \text{s.t. } \theta_1^* = \arg\min_{\theta_1} C_1(\theta_1, U_1, L_1, D_1) \quad \triangleright \text{Level } 1 \text{ problem}
\]

Source Optimization : optimize source parameters, fix others.

Mask Optimization : optimize mask parameters, fix others.

Source Mask Optimization : bi-level optimization for source and mask
### Results

The comparison of the proposed method and the SOTA method.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DAMO$^{21}$</th>
<th>TEMPO$^{34}$</th>
<th>DOINN$^{32}$</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mPA</td>
<td>mIOU</td>
<td>mPA</td>
<td>mIOU</td>
</tr>
<tr>
<td>Benchmark1$^{33}$</td>
<td>95.2</td>
<td>91.1</td>
<td>94.6</td>
<td>88.7</td>
</tr>
<tr>
<td>Benchmark2$^{35}$</td>
<td>98.97</td>
<td>97.31</td>
<td>98.24</td>
<td>96.55</td>
</tr>
<tr>
<td>Benchmark3$^{35}$</td>
<td>99.11</td>
<td>93.56</td>
<td>99.06</td>
<td>93.28</td>
</tr>
<tr>
<td>Benchmark4$^{33, 35}$</td>
<td>99.01</td>
<td>97.1</td>
<td>98.63</td>
<td>95.84</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>98.07</td>
<td>94.77</td>
<td>97.63</td>
<td>93.59</td>
</tr>
<tr>
<td><strong>Ratio</strong></td>
<td>0.99</td>
<td>0.96</td>
<td>0.98</td>
<td>0.94</td>
</tr>
</tbody>
</table>
THANK YOU!