Machine Learning and Its Applications in IC Physical Design

David Z. Pan
University of Texas at Austin
http://users.ece.utexas.edu/~dpan/

Yibo Lin
Peking University
http://yibolin.com/

Outline

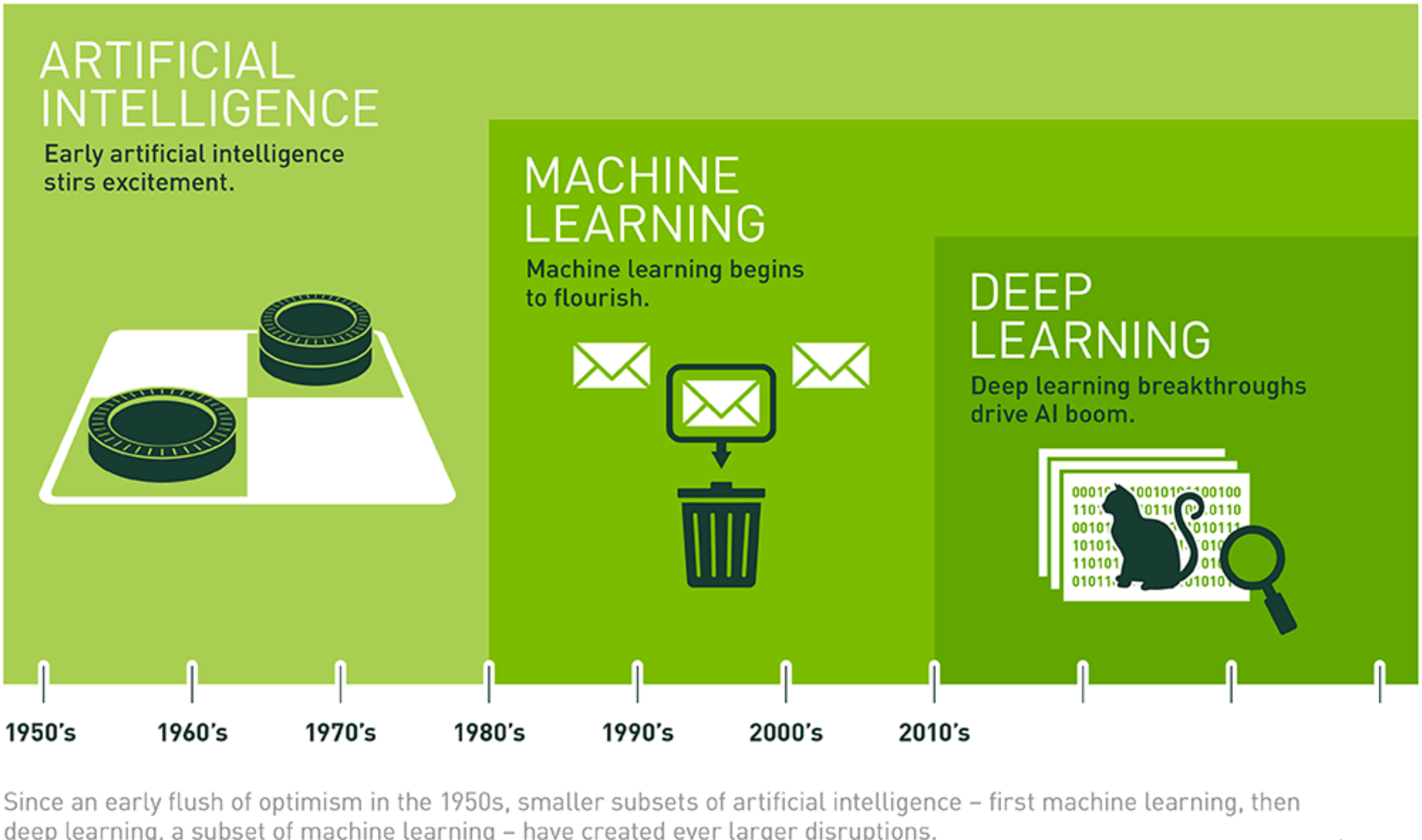
Part I: Introduction to Machine Learning

- What is Machine Learning
- Taxonomy of Machine Learning
- Machine Learning Techniques

Part II: Machine Learning for Physical Design

- Motivations
- Opportunities
- Challenges and Future Directions

A Bit History



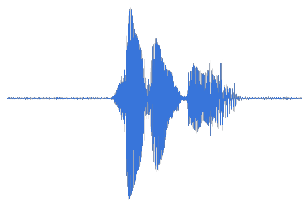
[Courtesy Nvidia]

Enormous Opportunities for Machine Learning

Enormous Opportunities for Machine Learning

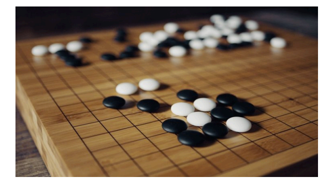
Machine Learning ≈ Looking for a Function

Speech recognition

$f($  $)$

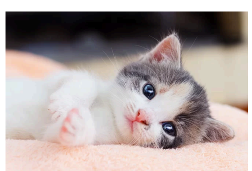
$= \text{“}How\ are\ you\text{”}$

Playing Go

$f($  $)$

$= \text{“}5-5\text{”}$ (next move)

Image recognition

$f($  $)$

$= \text{“}Cat\text{”}$

Dialogue system

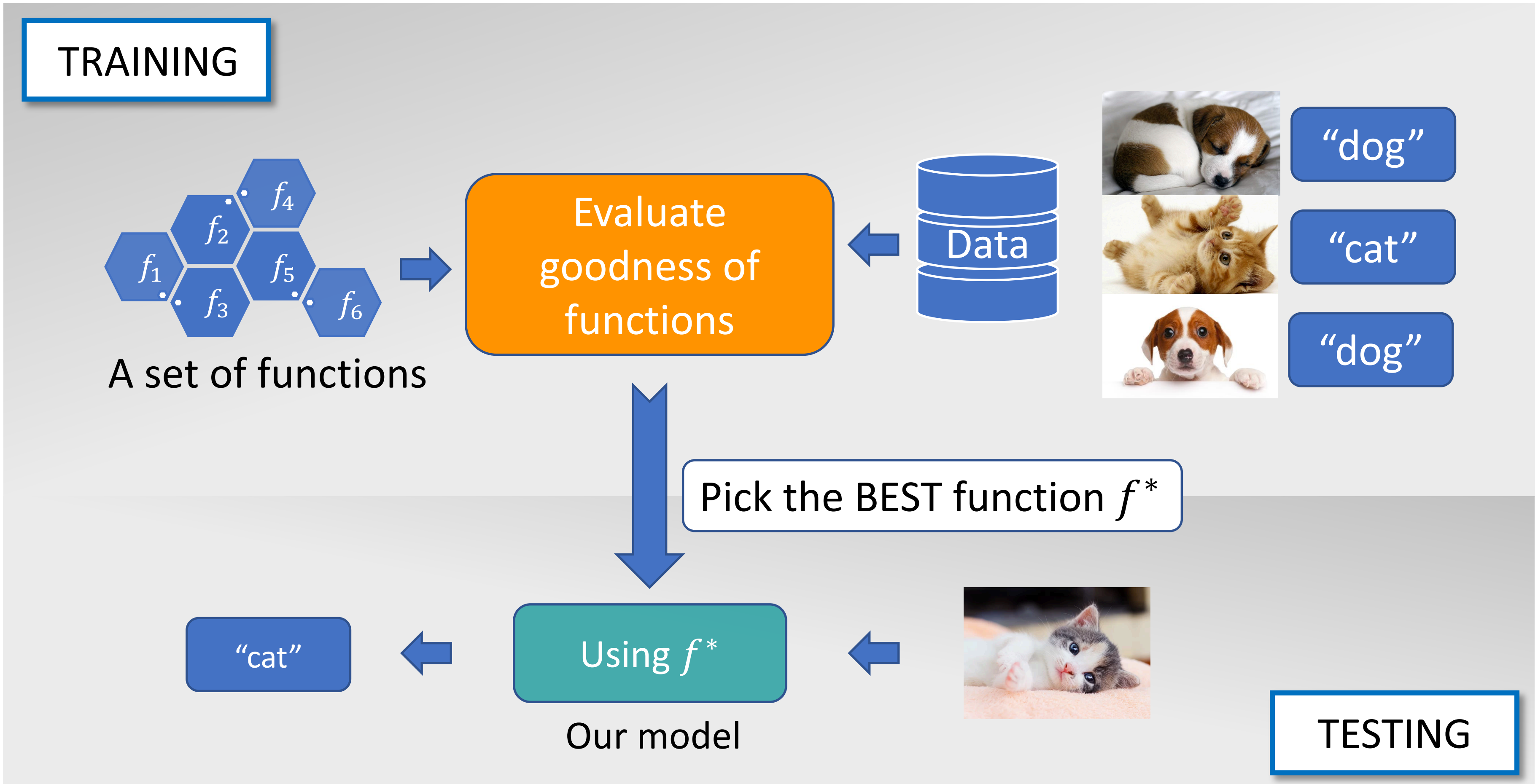
$f(\quad \text{“}Hi\text{”} \quad)$

(What the user said)

$= \text{“}Hello\text{”}$

(system response)

Machine Learning Framework



Taxonomy of Machine Learning

Unsupervised Learning	Supervised Learning	Semi-supervised Learning
Unsupervised Learning *only inputs x;* *no labels/outputs y*	Supervised Learning *training data with inputs x* *and labels/outputs y*	Semi-supervised Learning *subsets of data have* *labels/outputs y*

Scenarios

Based on [Boning@MIT2019]

Taxonomy of Machine Learning

Unsupervised Learning

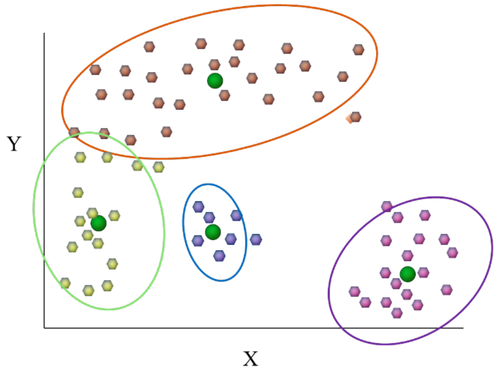
- Clustering
- Dimensionality reduction
- Kernel methods
- Density estimation

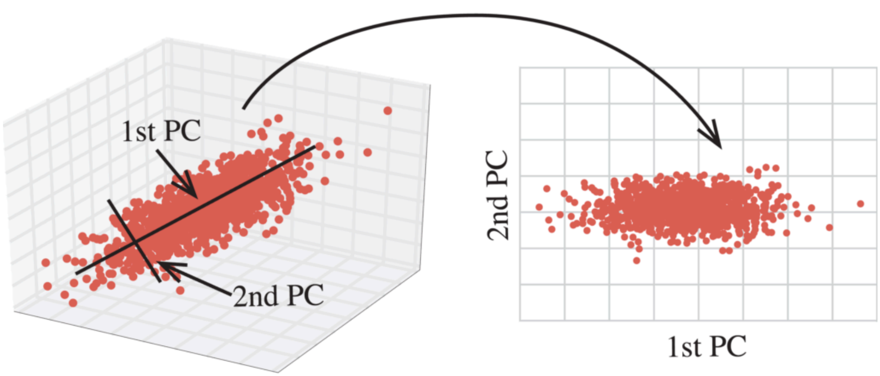
Supervised Learning

K-means

PCA

Semi-supervised Learning





Taxonomy of Machine Learning

Unsupervised Learning	Supervised Learning	Semi-supervised Learning
		• Sampling • Search and optimization • Active learning • Reinforcement learning • Online learning • Transfer learning • Adversarial learning

Taxonomy of Machine Learning

Unsupervised Learning

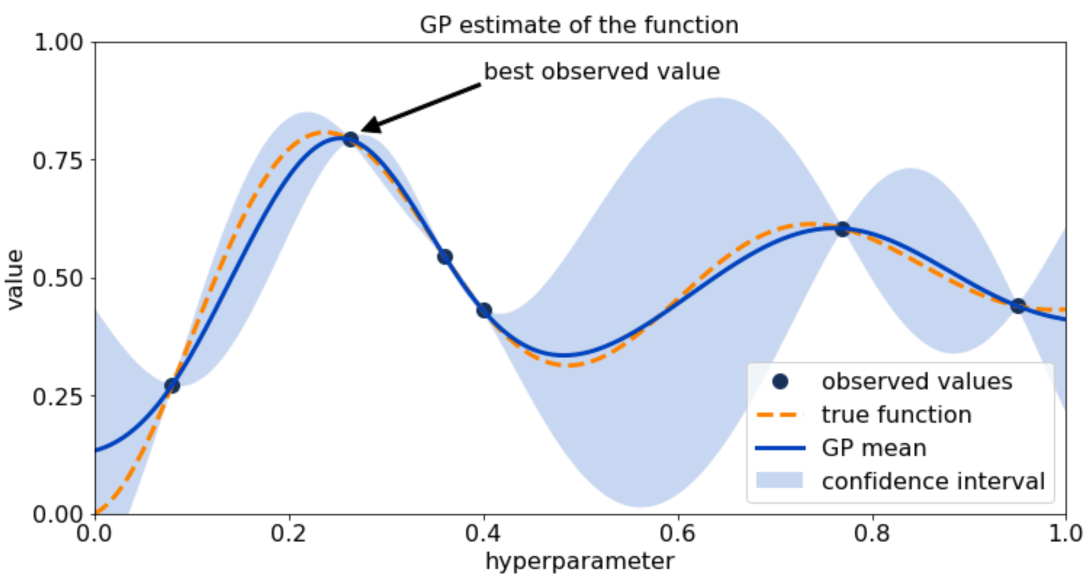
Supervised Learning

Semi-supervised Learning

- Sampling
- Search and optimization
- Active learning
- Reinforcement learning
- Online learning
- Transfer learning
- Adversarial learning

Genetic Algorithm

Bayesian Optimization



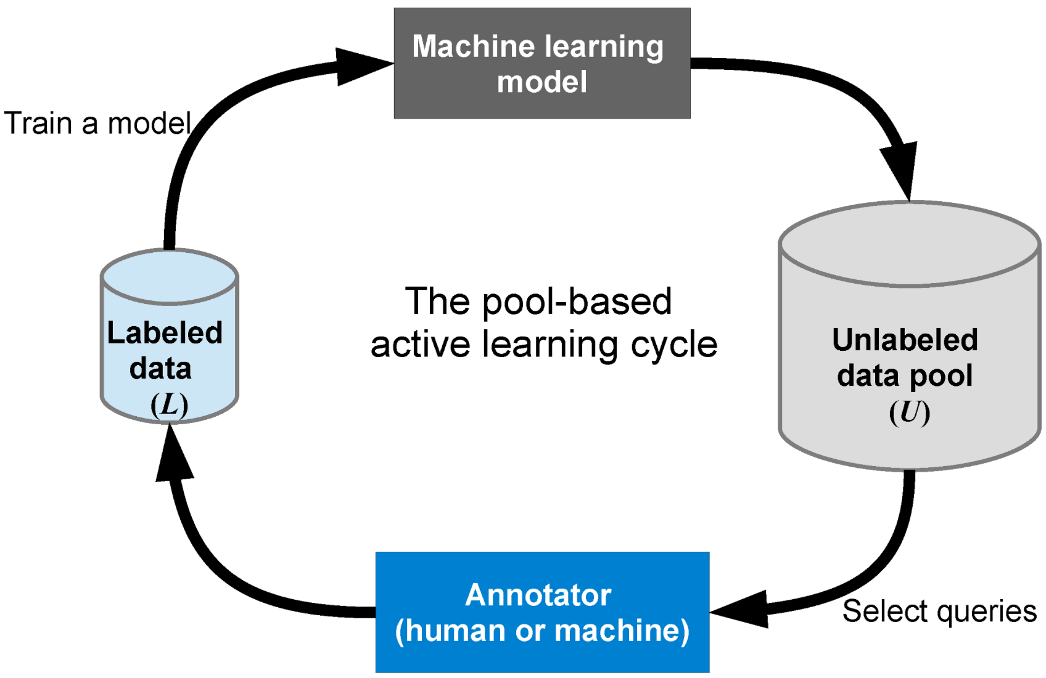
Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

- Sampling
- Search and optimization
- <u>Active learning</u>
- Reinforcement learning
- Online learning
- Transfer learning
- Adversarial learning



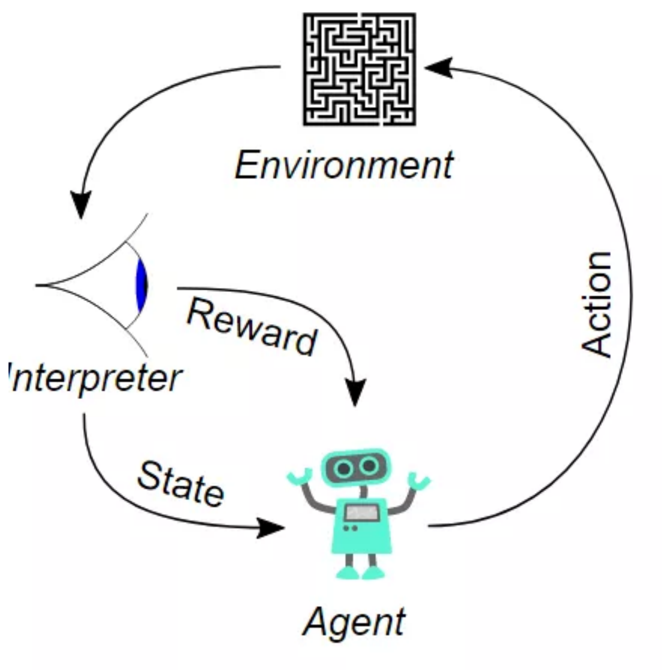
Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

- Sampling
- Search and optimization
- Active learning
- Reinforcement learning
- Online learning
- Transfer learning
- Adversarial learning



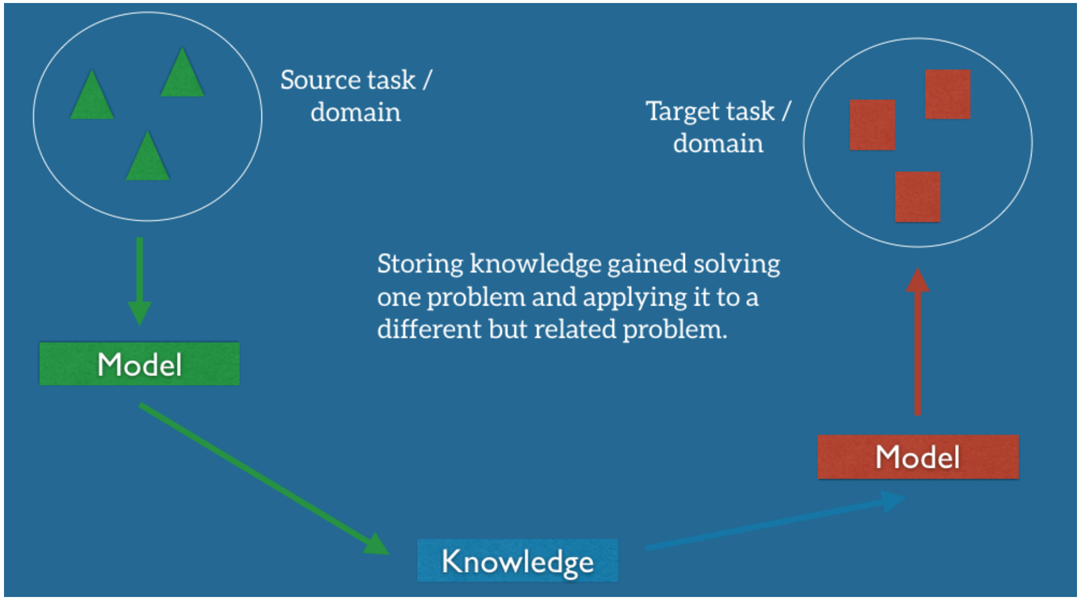
Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

- Sampling
- Search and optimization
- Active learning
- Reinforcement learning
- Online learning
- <u>Transfer learning</u>
- Adversarial learning



Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

Logistic
SVM

Gaussian process
Trees, random forest

Bayesian
MCMC

- Classification
- Regression (parametric)
- Instance-based (nonparametric)
- Dimensionality reduction
- Generative (density estimation)
- Discriminative

OLS, MARS, PCR, ridge…

Latent variable
Sparse coding
Matrix completion

NN

Deep
Convolutional
recurrent

Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

Logistic
SVM

Gaussian process
Trees, random forest

Bayesian
MCMC

- Classification
- **Regression (parametric)**
- **Instance-based (nonparametric)**
- Dimensionality reduction
- Generative (density estimation)

OLS, MARS, PCR, ridge…

Latent variable
Sparse coding
Matrix completion

Deep

Parametric methods
model complexity fixed (does not depend on number of training samples; may depend on hyperparameters and/or regularization)

Nonparametric methods
model complexity depends on training samples

Taxonomy of Machine Learning

Unsupervised Learning

Supervised Learning

Semi-supervised Learning

- Classification
- Regression
- Dimensionality reduction
- **Generative (density estimation)**
- **Discriminative**

Logistic

Gaussian pro

Trees, random

Bayesian
MCMC

RS, PCR, ridge…

nt variable
Sparse coding
Matrix completion

NN

Deep
Convolutional

Generative adversarial
combines generative and discriminative learners, e.g., GAN

Generative methods
learn $p(x, y)$ and/or $p(x|y)$ to reason about $p(y|x)$

Discriminative methods
learn $p(y|x)$ relationships directly

Before Going too Technical…

New Job Opportunities: AI Trainer

- Machines can learn by themselves
- Why do we need AI trainers/engineers?

- Pokemons can fight by themselves. Why do they need pokemon trainers?

Based on [Lee@NTU2017]

To be Serious, Tasks for AI Trainer

- Proper model
  - Define the set of functions
- Proper loss function
- Not always easy to find the best function
  - Optimality? E.g., Deep learning
- Require experienced engineers

Based on [Lee@NTU2017]

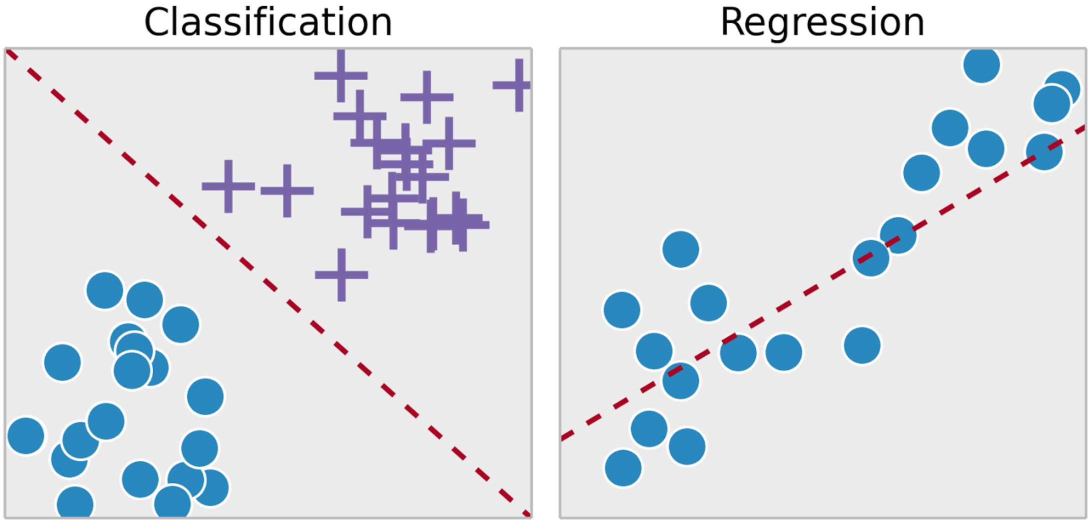
Taxonomy of Machine Learning

Unsupervised Learning	Supervised Learning	Semi-supervised Learning
• Clustering • Dimensionality reduction • Kernel methods • Density estimation	• Classification • Regression (parametric) • Instance-based (nonparametric) • Dimensionality reduction • Generative (density estimation) • Discriminative	• Sampling • Search and optimization • Active learning • Reinforcement learning • Online learning • Transfer learning • Adversarial learning

Based on [Boning@MIT2019]

Linear Models

- Linear regression
  - $f(x) = \langle w \cdot x \rangle + b = \sum_{i=1}^{N} w_i x_i + b$
- Logistic classification
  - $P(y = 1|x) = \frac{1}{1+e^{-(\sum_{i=1}^{N} w_i x_i + b)}}$



Linear Models – Training

- Ordinary Least Squares (OLS)
  - Minimizes the mean square error over the n data points
  - Direct solution is possible

$$\min_{w,b} \quad \frac{1}{n}\sum_{i=1}^{n}(w^T x^{(i)} + b - y^{(i)})^2$$

- Ridge Regression
  - Regularization by penalizing large weights

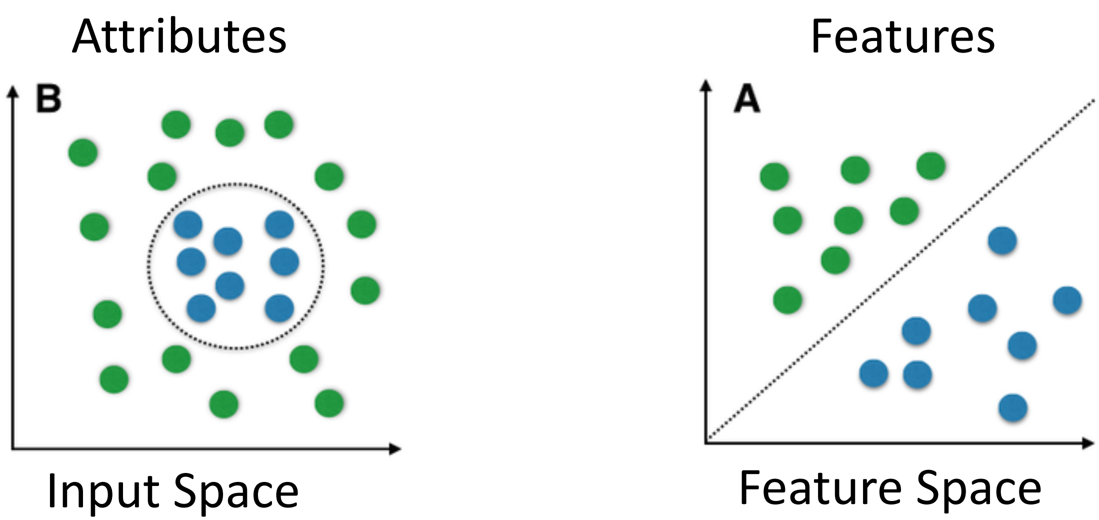
$$\min_{w,b} \quad (\frac{1}{n}\sum_{i=1}^{n}(w^T x^{(i)} + b - y^{(i)})^2) + \lambda \left\| w \right\|^2$$

- Stochastic gradient descent (SGD)
  - Pick a data point i (or a batch) at random for iterative model updates, same idea as neural networks

Feature Space

- Mapping from input space to feature space

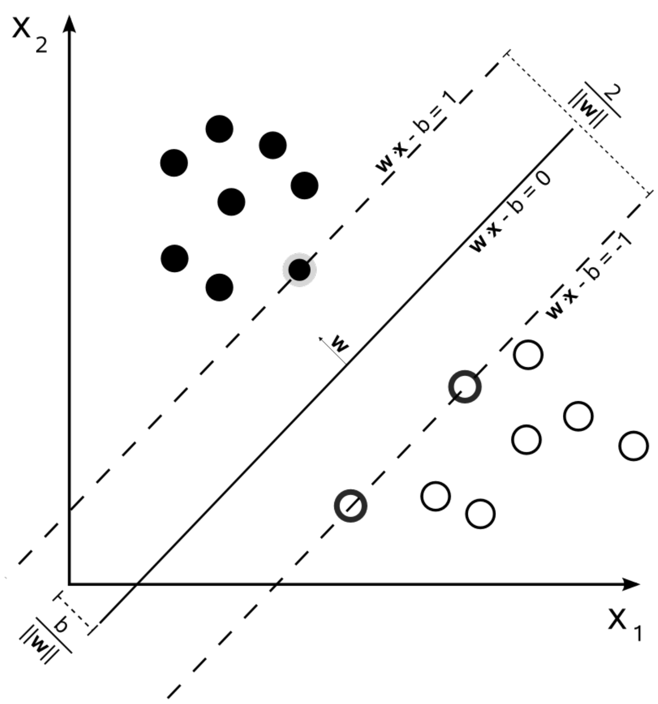
$$\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}] \mapsto \boldsymbol{\phi}(\boldsymbol{x}) = [\boldsymbol{\phi}(\boldsymbol{x_1}), \boldsymbol{\phi}(\boldsymbol{x_2}), \cdots, \boldsymbol{\phi}(\boldsymbol{x_N})]$$



- Nonlinear features under a linear model
  - $f(x) = \sum_{i=1}^{M} w_i \phi_i(x) + b$

Support Vector Machine

- A learning method that uses a hypothesis space of linear functions in a high dimensional feature space trained with an optimization based learning algorithm



x_i: feature vector

y_i: label, {-1, 1}

Consider $\hat{y}_i = w_i x_i - b$

Objective

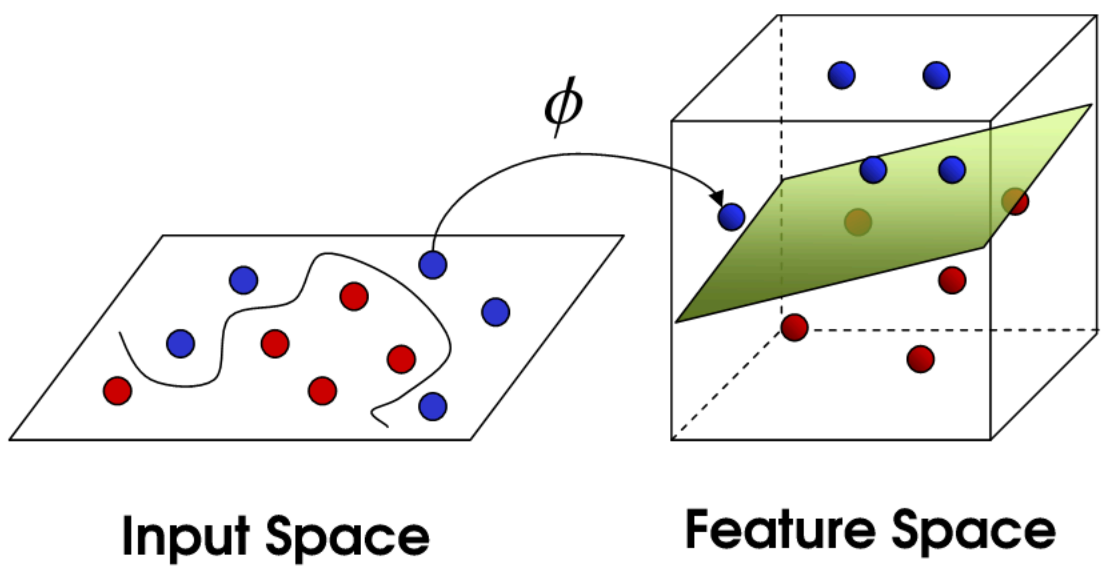
$$min. \frac{1}{2} \|w\|^2$$

Subject to

$$(w_i x_i - b) y_i \geq 1$$

Support Vector Machine

- Extend to a high-dimensional feature space through the (nonlinear) kernel
- Data can be linearly separable in feature space



Decision Trees for Classification

Partition the input space and fit very simple models to predict the output in each partition.

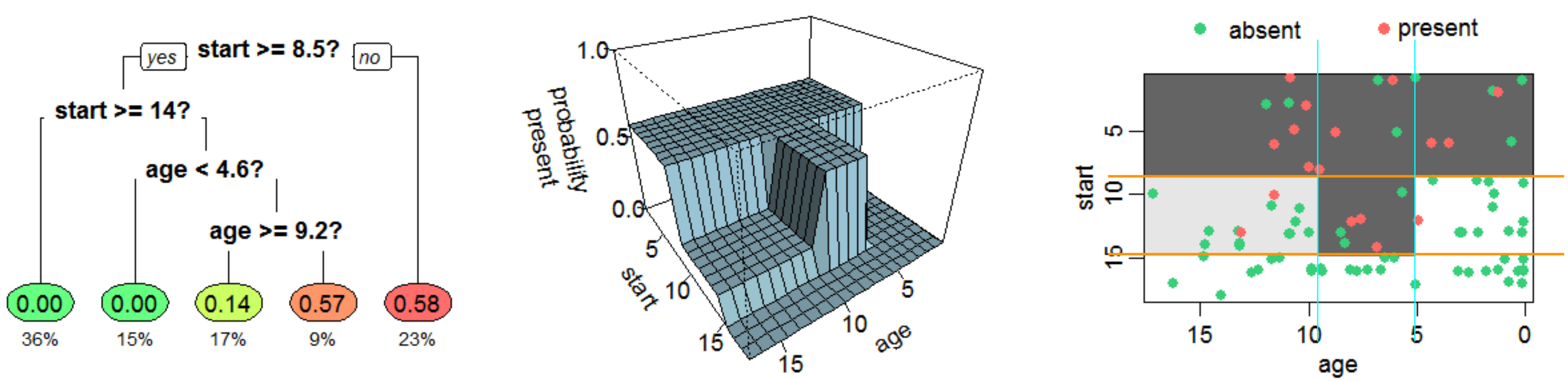
- Split nodes: select the “best” feature and value to split
- “Best”: usually defined in terms of some notion of “impurity” in the resulting partition of training data. Typical impurity measures include misclassification error, Gini index, or entropy

Probability of kyphosis after surgery [wikipedia]

Decision Trees for Classification

Partition the input space and fit very simple models to predict the output in each partition.

- Split nodes: select the “best” feature and value to split
- “Best”: usually defined in terms of some notion of “impurity” in the resulting partition of training data. Typical impurity measures include misclassification error, Gini index, or entropy



Probability of kyphosis after surgery [wikipedia]

K-Means Clustering

Unsupervised learning

- Only have feature x, no label y

Target: identify natural groups of data

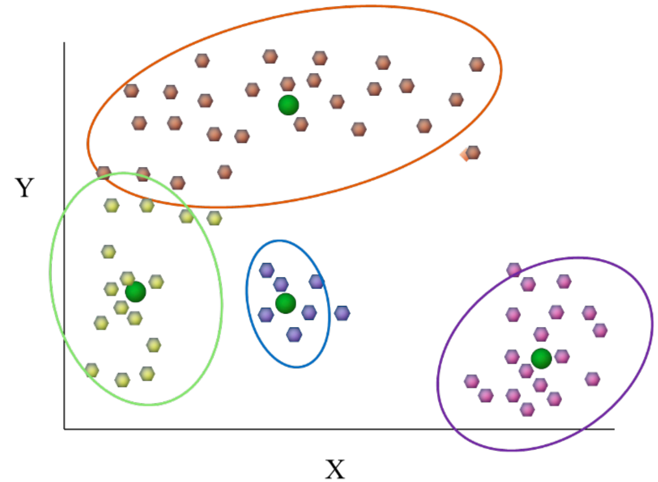
- $\min \sum_{i=1}^{K} \sum_{x \in S_i} \|x - \mu_i\|^2$

<u>Iterative algorithm</u>

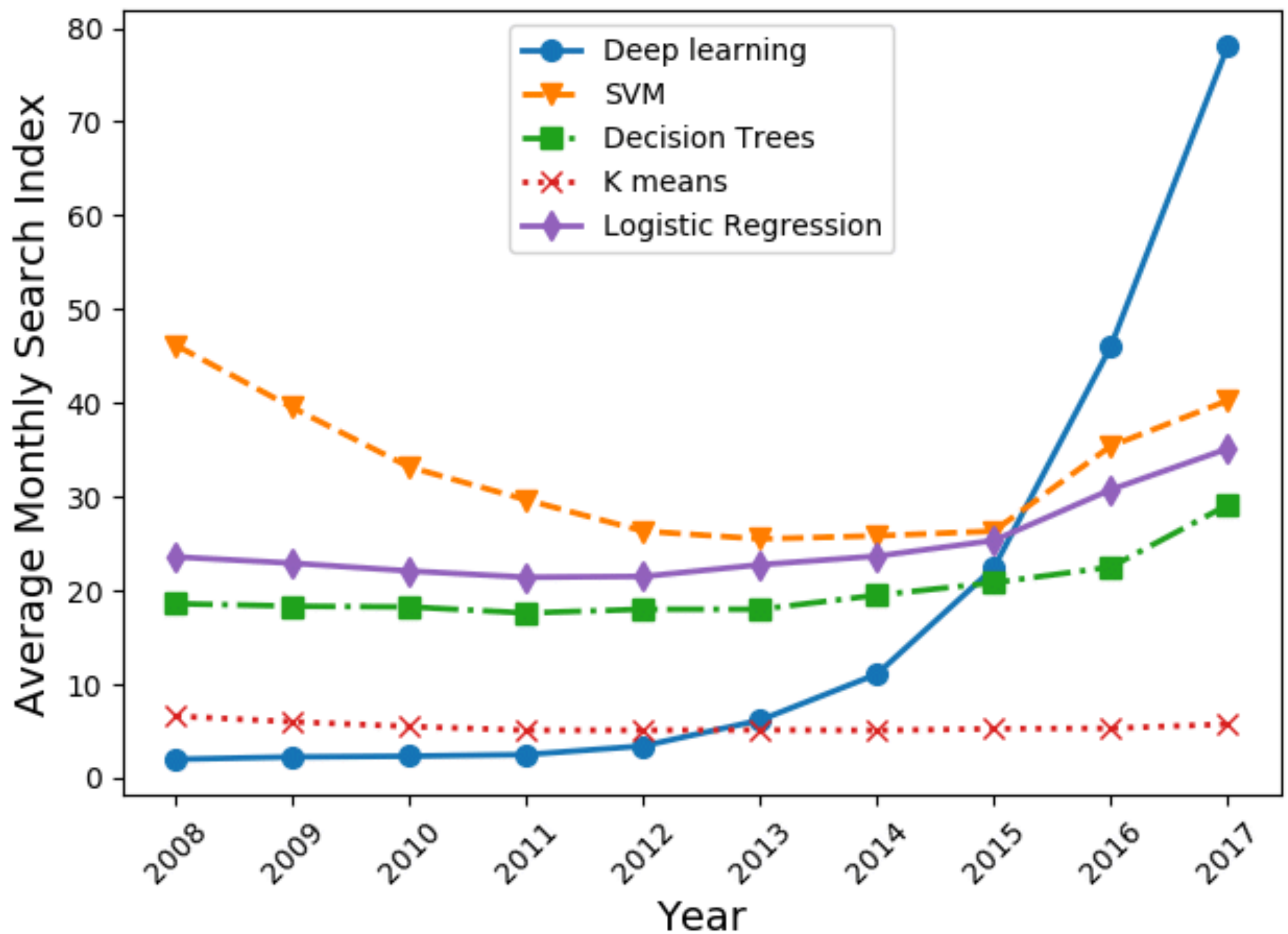
Randomly set cluster centers $m_1, \ldots m_k$;

For not converged

- Use these centers to assign each data point to the nearest cluster;
- Adjust centers m_i based on those memberships;



Deep Learning



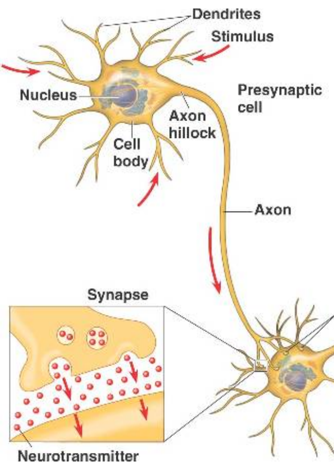
Deep Learning for IoT Big Data and Streaming Analytics: A Survey

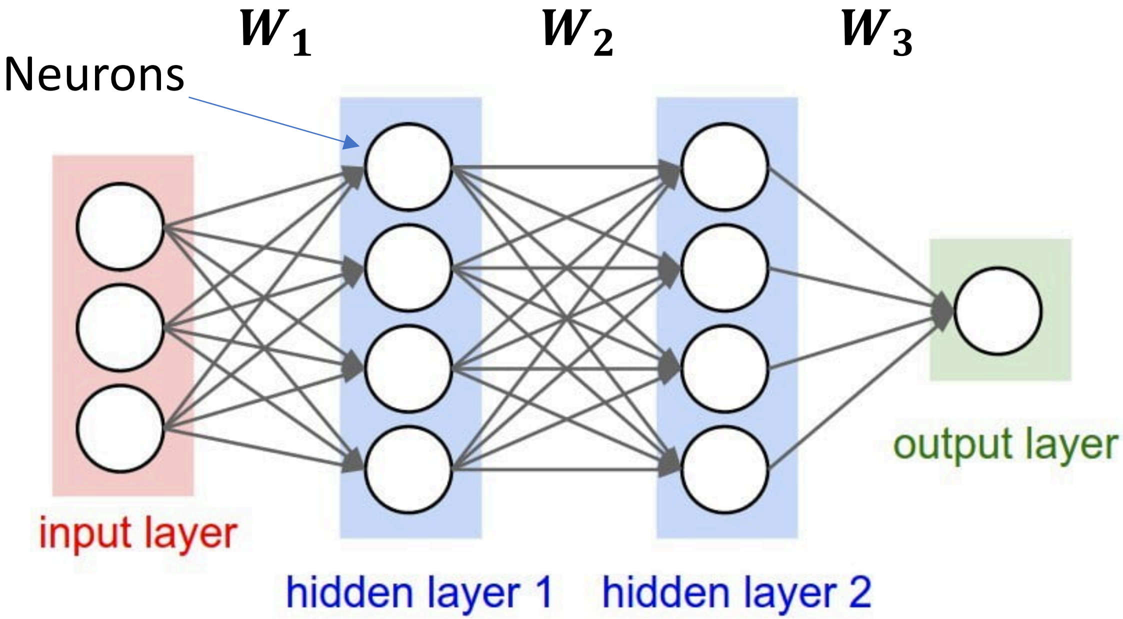
Neural Networks

- Multilayer perceptron (MLP)

$$f(x) = \sigma_3(W_3\sigma_2(W_2\sigma_1(W_1x)))$$

σ_i: activation function



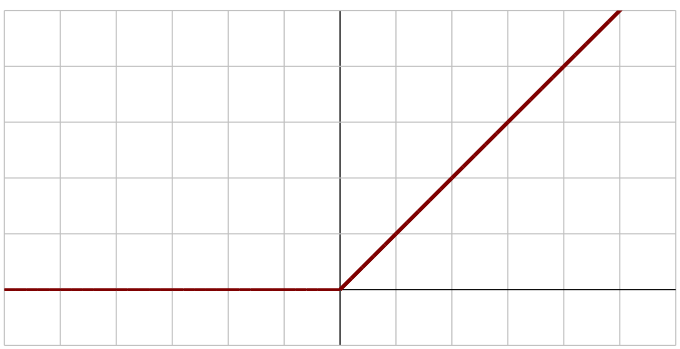


Neural Networks

- Multilayer perceptron (MLP)

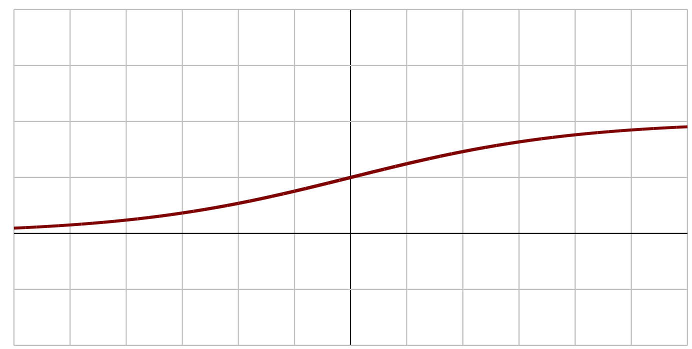
$$\boldsymbol{f(x) = \sigma_3(W_3\sigma_2(W_2\sigma_1(W_1x)))}$$

$\boldsymbol{\sigma_i}$: activation function



Rectified linear unit (ReLU)

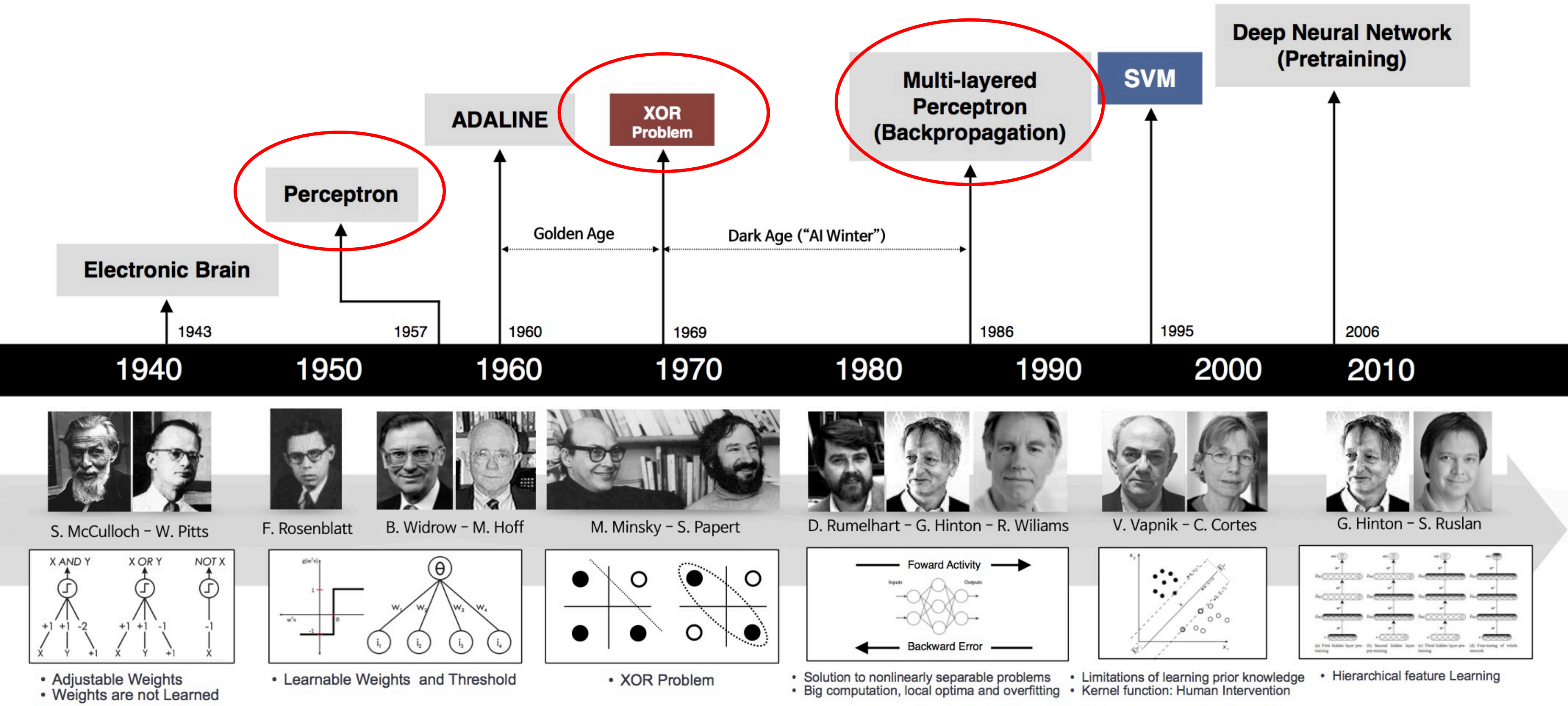
$$f(x) = \begin{cases} 0, for\ x < 0 \\ x, for\ x \geq 0 \end{cases}$$



Logistic (Sigmoid)

$$f(x) = \frac{1}{1 + e^{-x}}$$

A Bit History on Deep Learning

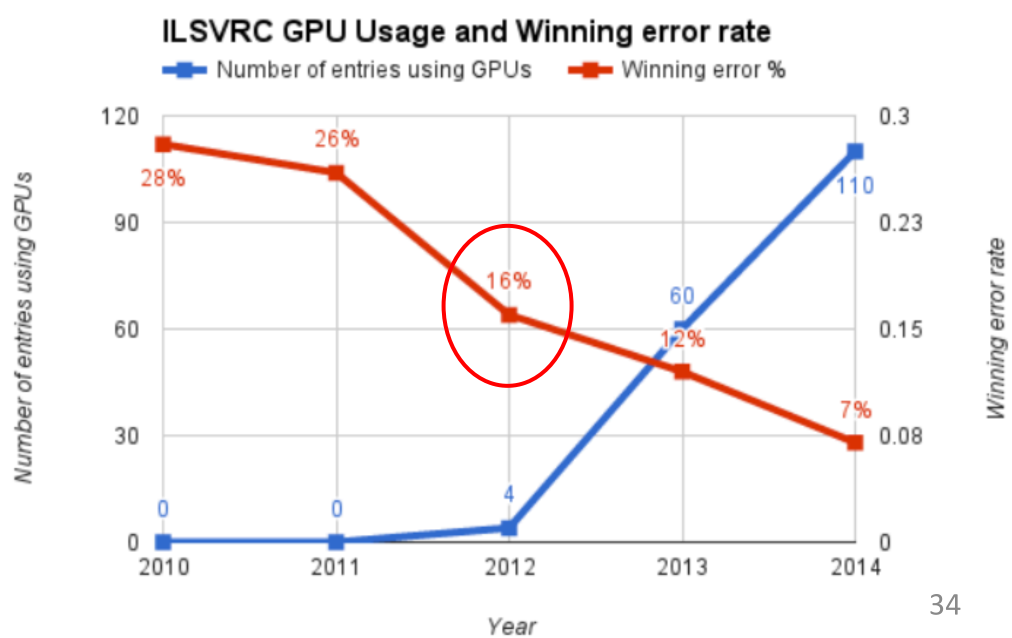


https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

A Bit History on Deep Learning

- "Neural networks" → "Deep learning" in 2006
  - Geoffrey Hinton
- Breakthrough in 2012
  - Large scale visual recognition challenge (LSVRC)
  - ImageNet

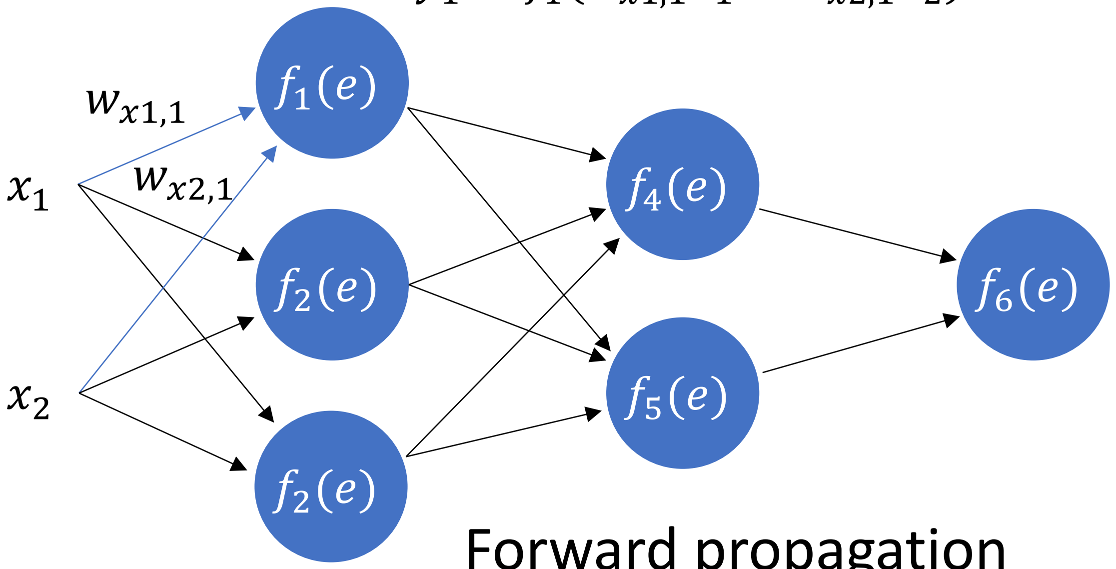
https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html



How to Train a Neural Network

- Objective: $min. \sum_i Loss(f(x_i), y_i)$
- Gradient descent
- Backpropagation

$$y_1 = f_1(w_{x1,1}x_1 + w_{x2,1}x_2)$$

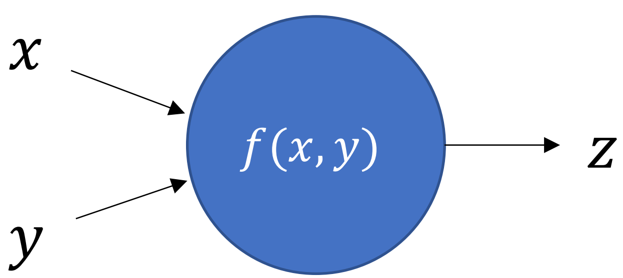


Forward propagation

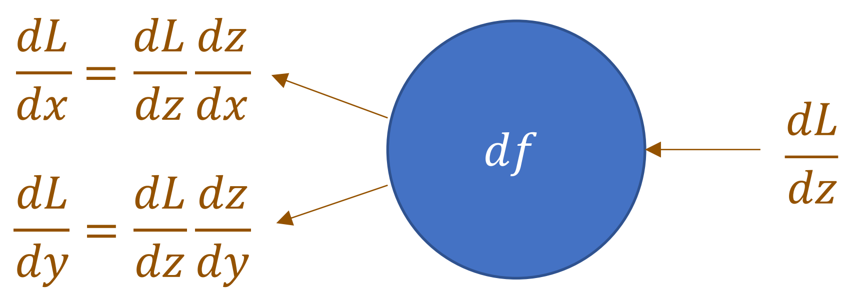
How to Train a Neural Network

- Gradient descent
  - Loss function: $L(z)$
- Backpropagation
  - Chain rule

Forward pass

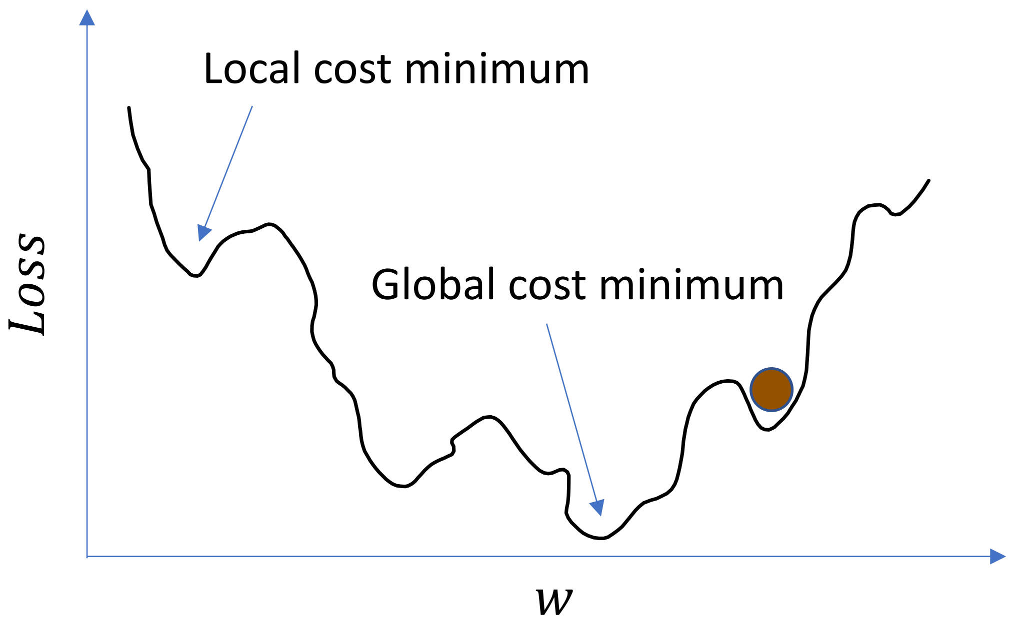


Backward pass



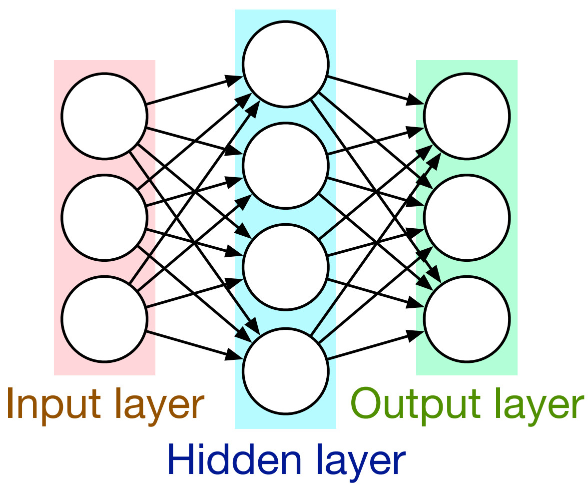
How to Train a Neural Network

- Gradient descent
- Backpropagation
- Non-linear optimization

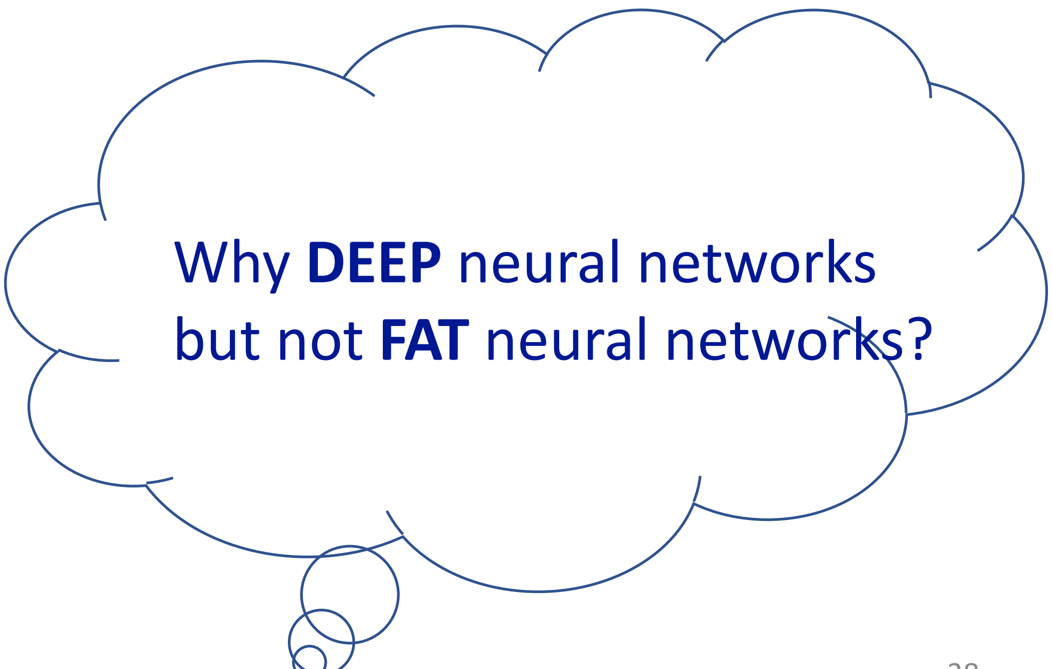


Why Deep? Universality Theorem

- Any function $f: R^N \rightarrow R^M$
- Can be realized by a neural network with one hidden layer
- (given enough hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

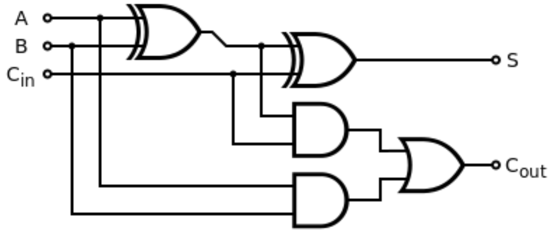


Why Deep? Analogy

Logic Circuits

- Consist of logic gates
- A two layers of logic gates can represent **any Boolean function**
- Use multiple layers of logic gates to build functions in a much simpler way

Less gates required



Neural Networks

- Consist of neurons
- A hidden layer network can represent **any continuous function**
- Use multiple layers to represent functions in a much simpler way

Less parameters → Less data?

More reason:
https://www.youtube.com/watch?v=XsC9byQkUH8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=13

Why Deep? NOT Fat

- Ocaam's razor: Entities should not be multiplied without necessity
  - The simplest solution is most likely the right one
  - Less parameters, better generalization
- Deep + Thin v.s. Short + Fat
  - Same # of parameters
  - Deep networks tend to generalize better
  - Require less amount of training data

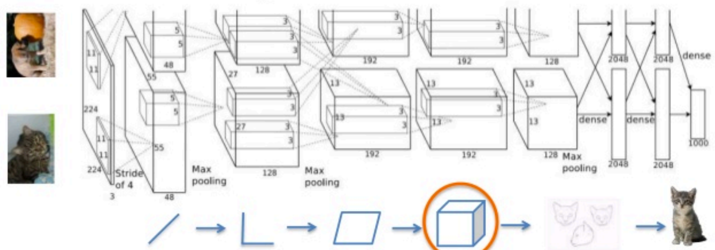
#Layers x #Neurons	Error Rate (%)	1 x #Neurons	Error Rate (%)
5x2k	17.2	1x3772	22.5
7x2k	17.1	1x4634	22.4

Seide, Frank, Gang Li, and Dong Yu. "Conversational speech transcription using context-dependent deep neural networks." *Twelfth annual conference of the international speech communication association*. 2011.

How Deep?

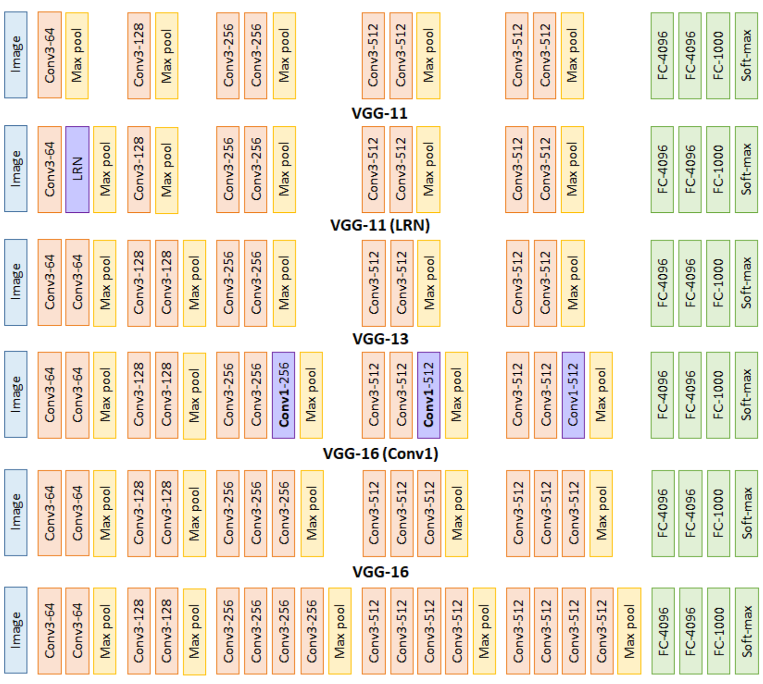
AlexNet (Krizhevsky et al. 2012)

The class with the highest likelihood is the one the DNN selects

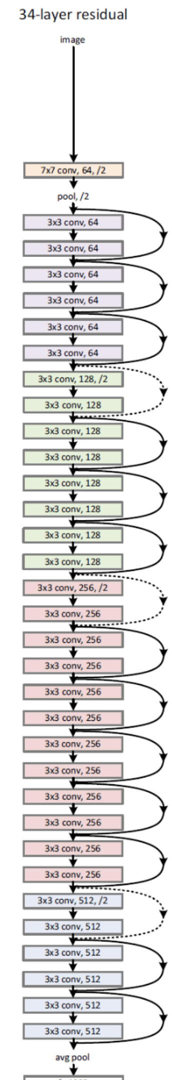


When AlexNet is processing an image, this is what is happening at each layer.

5-layers
Top-5 err.
16.4



Model	Number of Parameters (millions)	Top-5 Error Rate (%)
VGG-11	133	10.4
VGG-11 (LRN)	133	10.5
VGG-13	133	9.9
VGG-16 (Conv1)	134	9.4
VGG-16	138	8.8
VGG-19	144	9.0

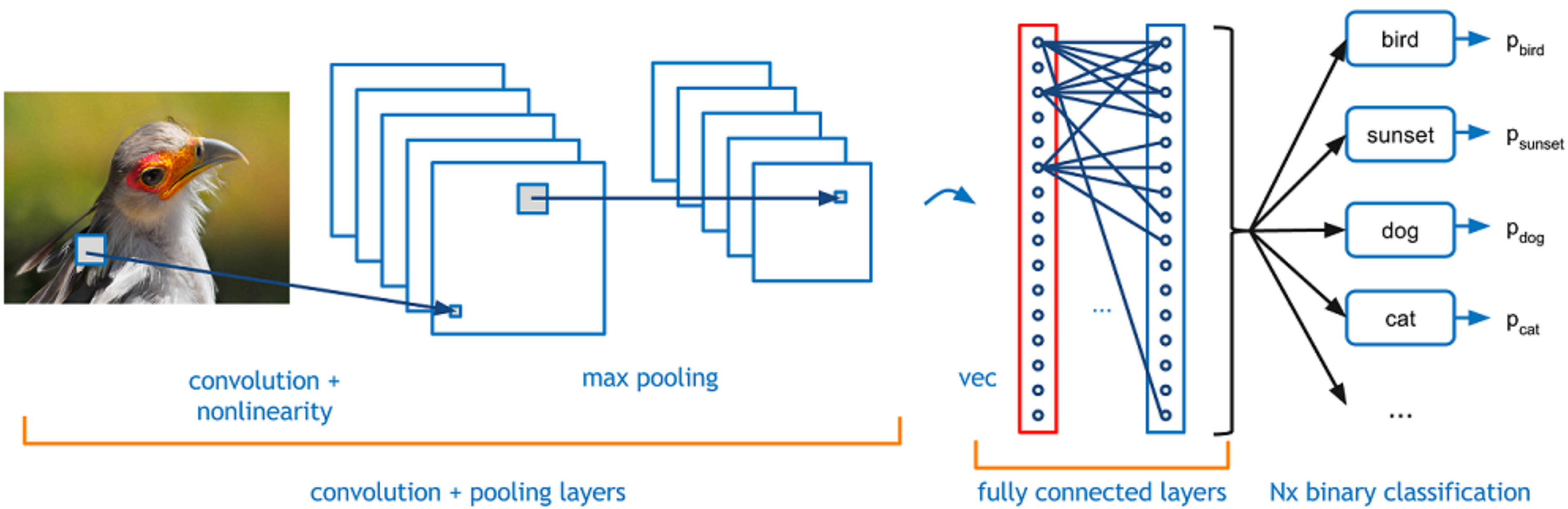


ResNet	Top-5 err.
34	7.4
101	6.0
152	5.7

More Flavors…
Convolutional Neural Networks

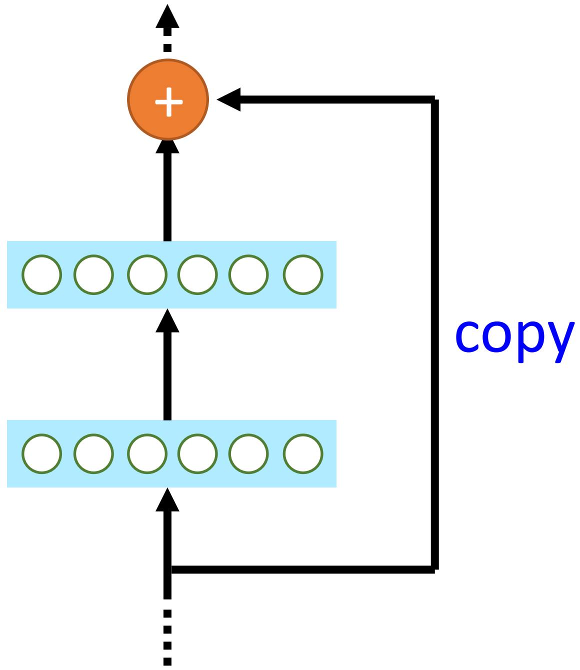
Convolutional layer

- Suitable to images
- Correlation with neighboring pixels



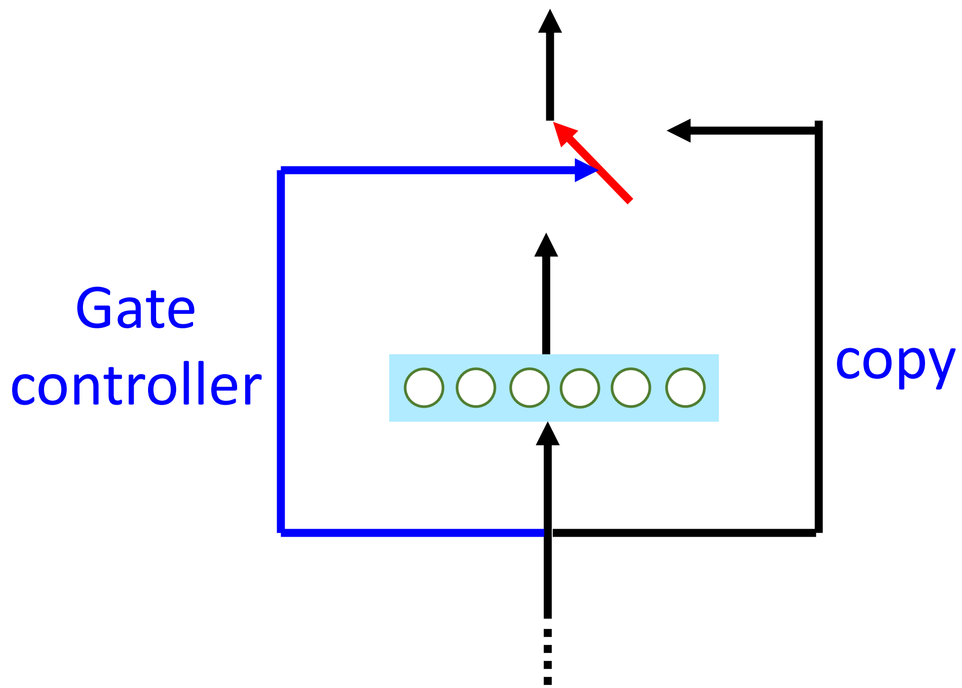
More Flavors…
Highway Networks

Residual Network



Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

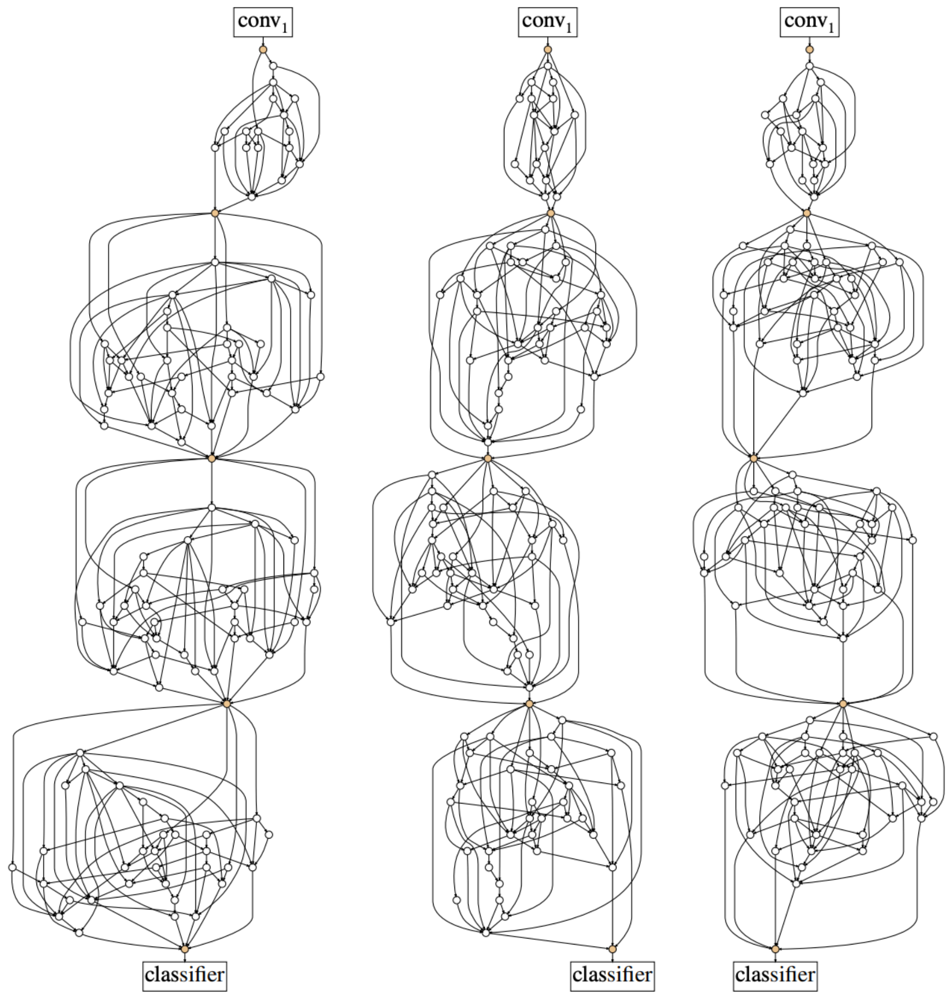
Highway Network



Training Very Deep Networks
https://arxiv.org/pdf/1507.06228v2.pdf

More Flavors…
Random Networks

Exploring Randomly Wired Neural Networks for Image Recognition
https://arxiv.org/pdf/1904.01569.pdf

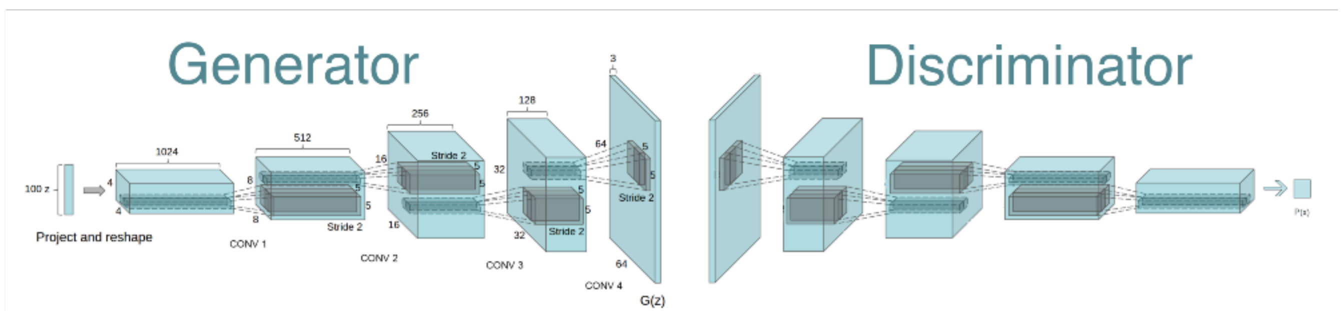


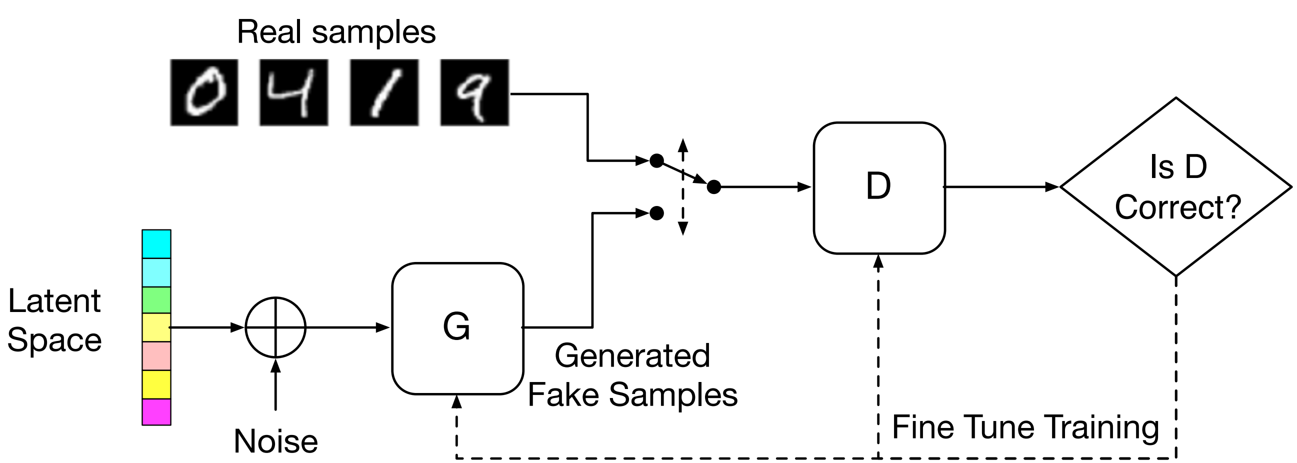
Generative Adversarial Networks

- GAN [Goodfellow et al, 2014] [Radford et al, 2015]
  - Two networks contest (generator and discriminator)
  - Produces images similar to those in the training data

Generative Adversarial Networks

- GAN [Goodfellow et al, 2014] [Radford et al, 2015]
  - Two networks contest (generator and discriminator)
  - Produces images similar to those in the training data

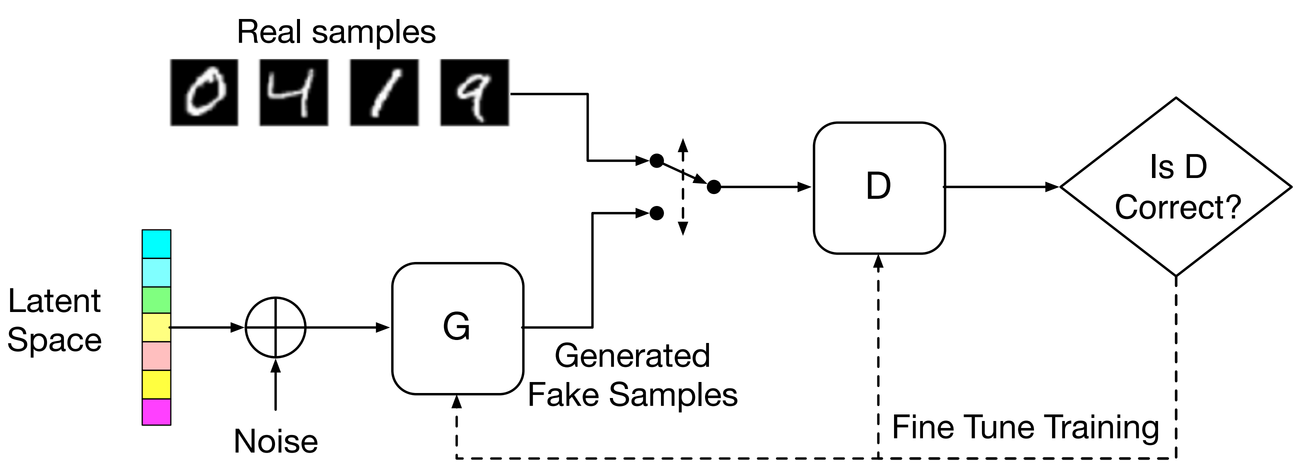




Generative Adversarial Networks

- GAN [Goodfellow et al, 2014] [Radford et al, 2015]
  - Two networks contest (generator and discriminator)
  - Produces images similar to those in the training data

$$\min_G \max_D \underbrace{\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)]}_{\textbf{Loss for real samples}} + \underbrace{\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]}_{\textbf{Loss for generated samples}}$$

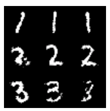


Generative Adversarial Networks

- GAN [Goodfellow et al, 2014] [Radford et al, 2015]
  - Two networks contest (generator and discriminator)
  - Produces images similar to those in the training data

DCGAN

Recent Development of GANs



Conditional GANs (CGANs)

What if I want my GAN to generate data with specific attributes

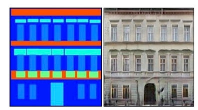
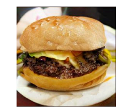


Image-to-Image Translation with CGAN (pix2pix)

Let's build a generic CGAN architecture where the condition is an image, and learn to translate this image into another domain



BigGAN

Google: "Hold my beer Nvidia, my interns can use thousands of TPUs to generate cheeseburgers"

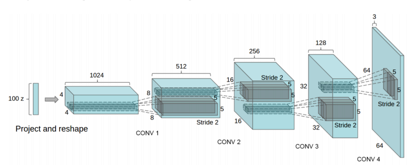
Jun. 2014 — **Nov. 2014** — **Nov. 2015** — **Nov. 2016** — **Nov. 2017** — **Sep. 2018**

Generative Adversarial Networks (GANs)

The architecture of a generator and a discriminator is first proposed by *Goodfellow et al.*

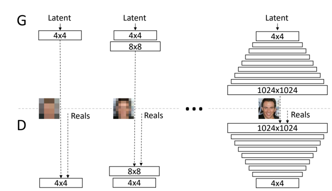
Deep Convolutional GANs (DCGANs)

As the first major improvement on the GAN architecture, DCGAN is more stable during training and generates higher quality samples



PG-GAN/pix2pixHD

Nvidia: "Let's make some high-resolution GANs with multi-scale approaches"



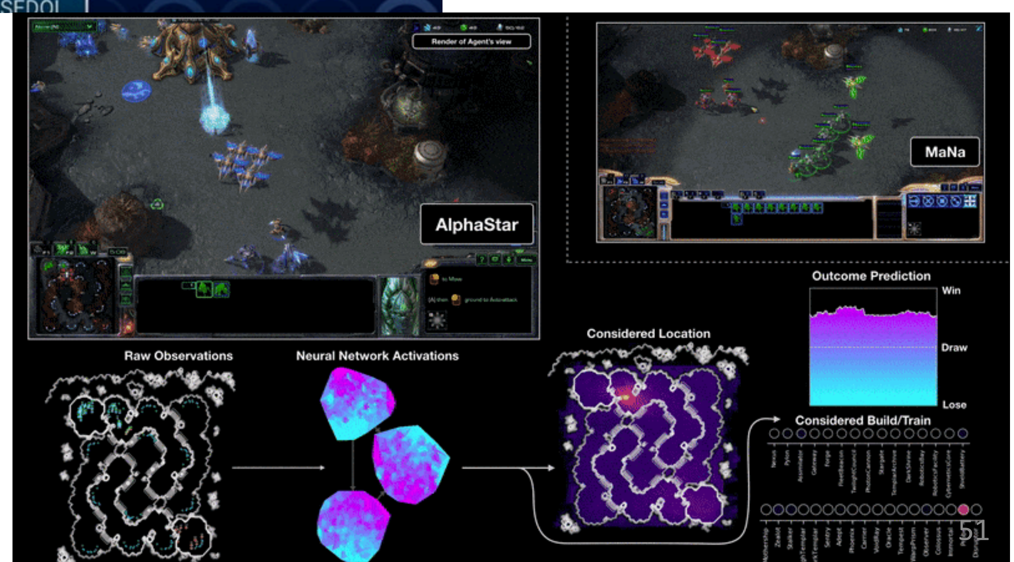
Recent Development of GANs

High quality images generated by BigGAN

Reinforcement Learning

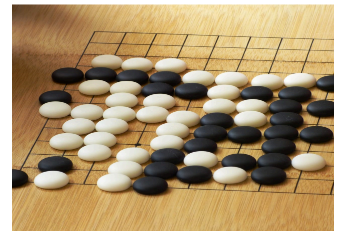
AlphaGO

AlphaStar

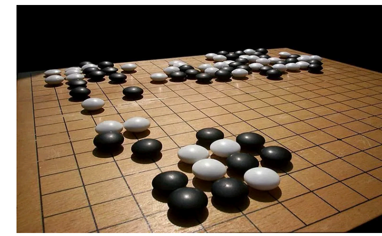


Supervised v.s. Reinforcement

- Supervised
  - Learning from teachers
  - Labeled data



Next move: “5-5”



Next move: “3-3”

- Reinforcement
  - Learn from critics

- AlphaGO
  - Supervised learning + reinforcement learning

Reference and More Resources

ML courses from Hung-Yi Lee at NTU

- http://speech.ee.ntu.edu.tw/~tlkagk/courses.html

Stanford ML courses on Coursera

- https://www.coursera.org/learn/machine-learning

Tutorials on GANs

- NIPS 2016: https://arxiv.org/pdf/1701.00160.pdf
- CVPR 2018: https://sites.google.com/view/cvpr2018tutorialongans/

Toolkit tutorials

- Tensorflow: https://www.tensorflow.org/tutorials
- PyTorch: https://pytorch.org/tutorials/

Outline

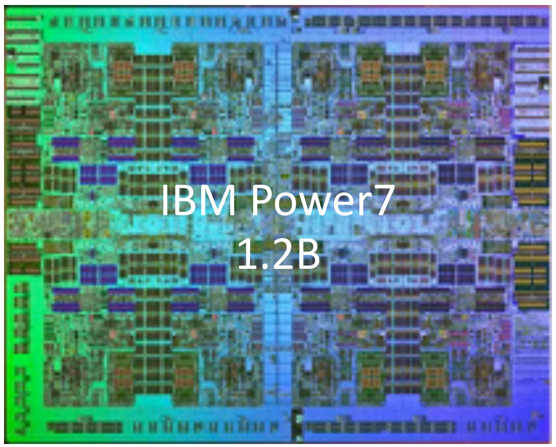
Part I: Introduction to Machine Learning

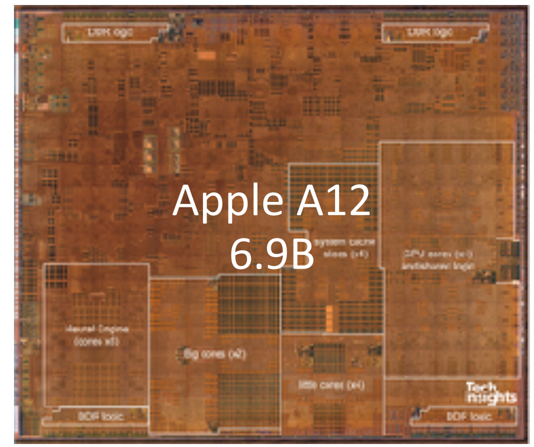
- What is Machine Learning
- Taxonomy of Machine Learning
- Machine Learning Techniques

Part II: Machine Learning for Physical Design

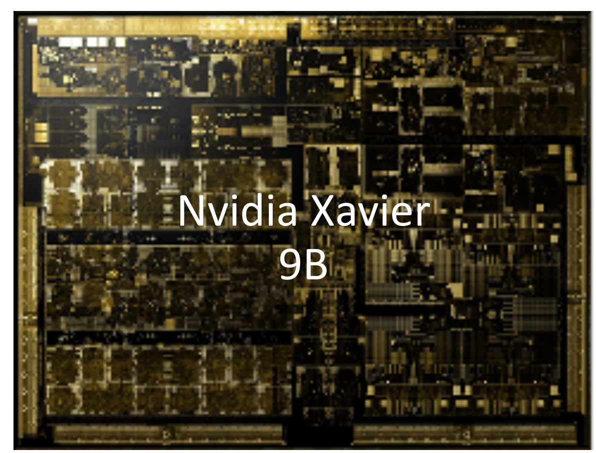
- Motivations
- Opportunities
- Challenges and Future Directions

Modern VLSI Layouts

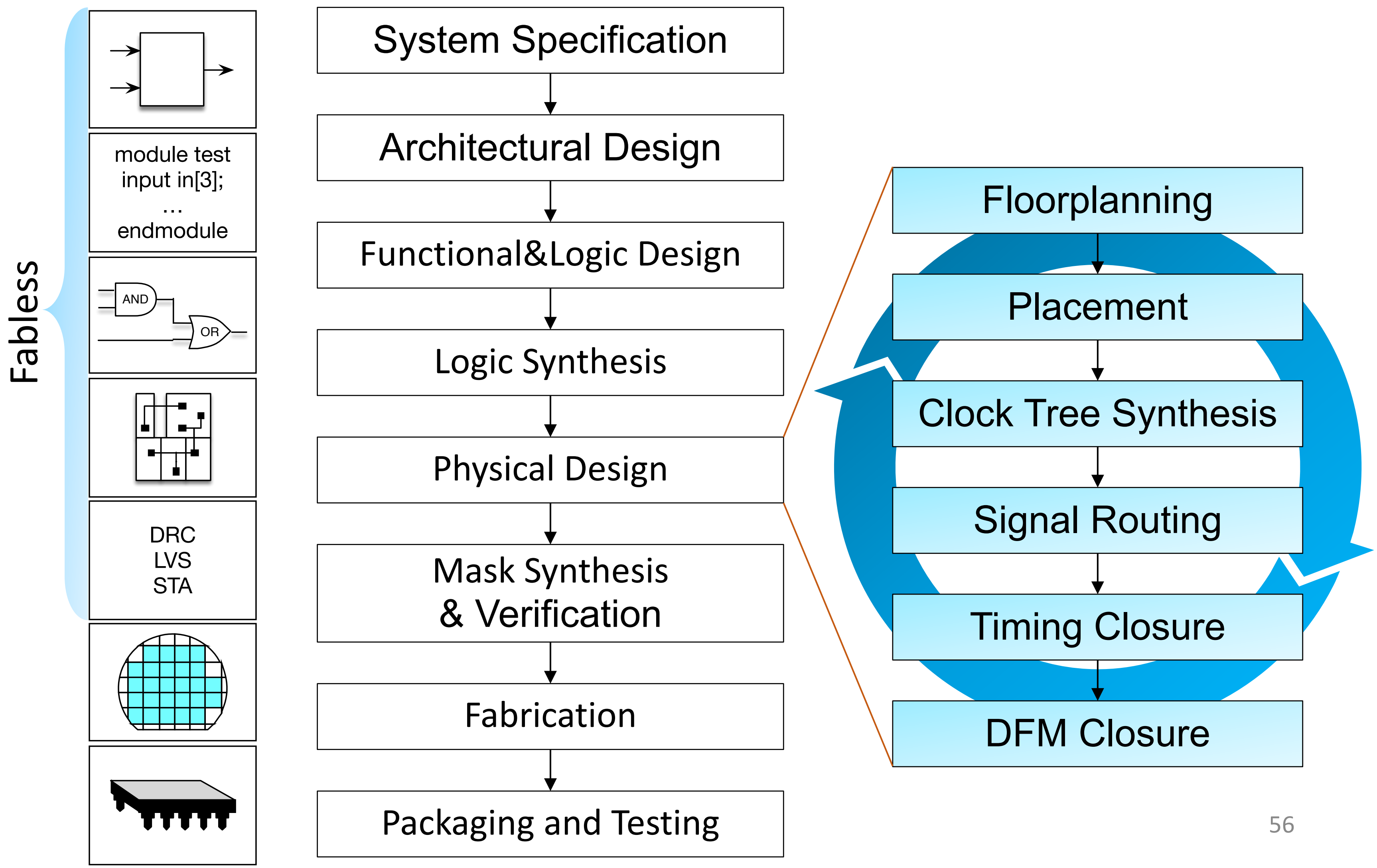




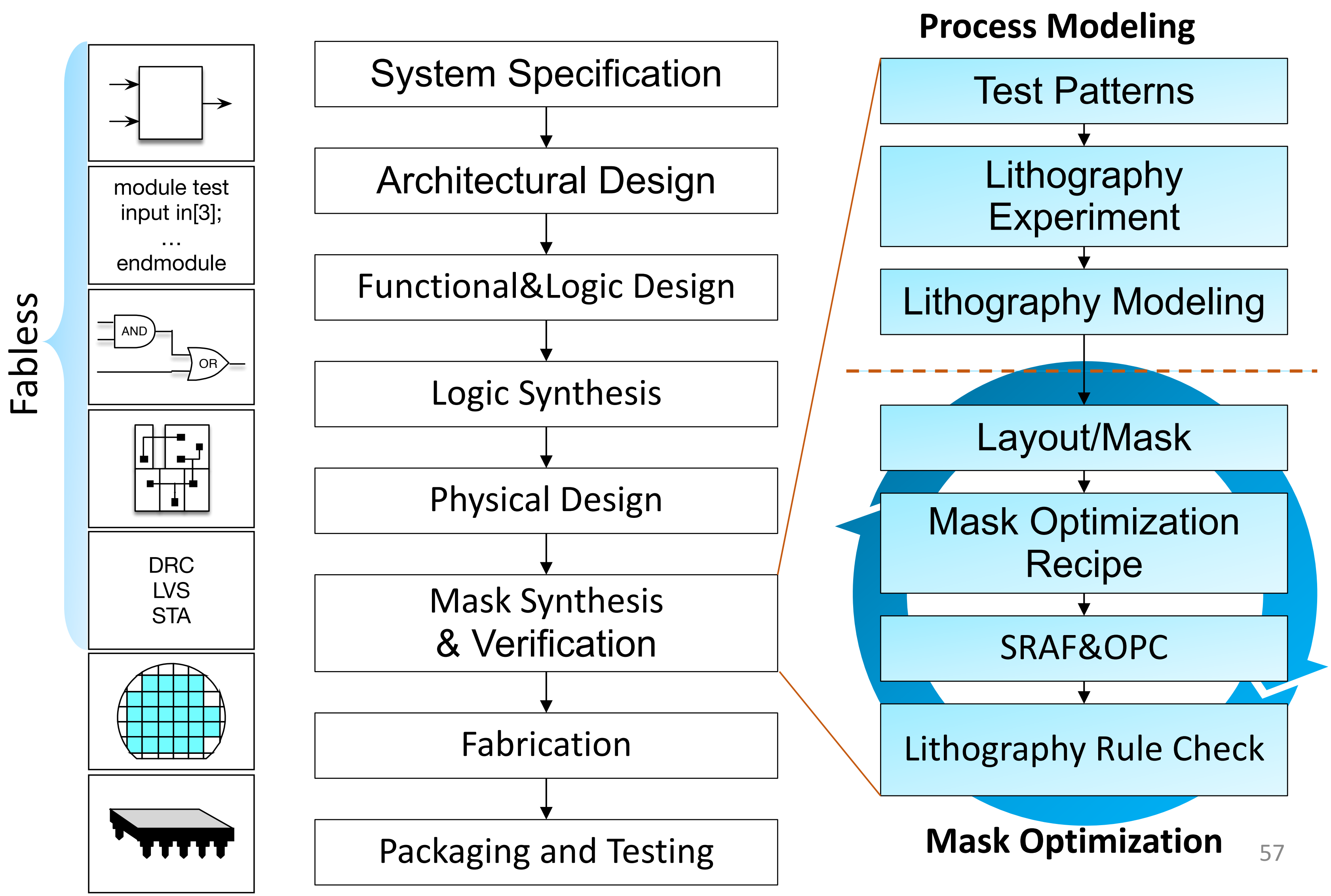
- Large scale: billions of transistors
- Complicated design flow
- Long design cycles



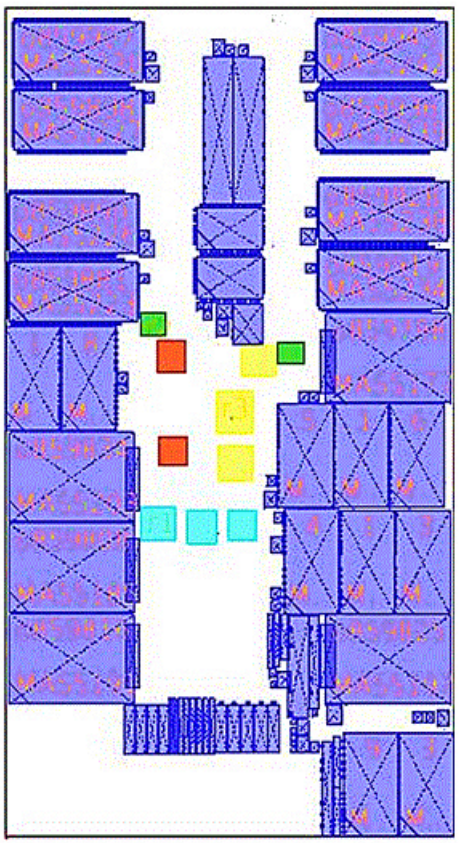
IC Design Flow – Silicon Compiler



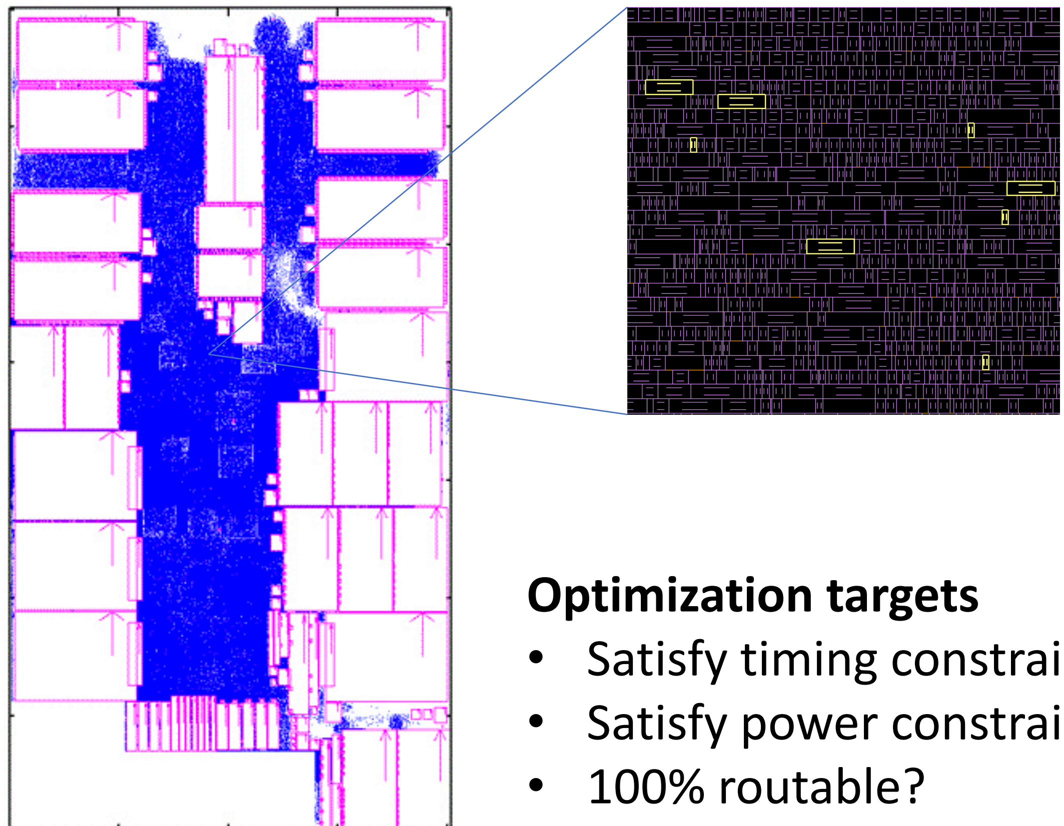
IC Design Flow – Silicon Compiler



Placement Example



Floorplan



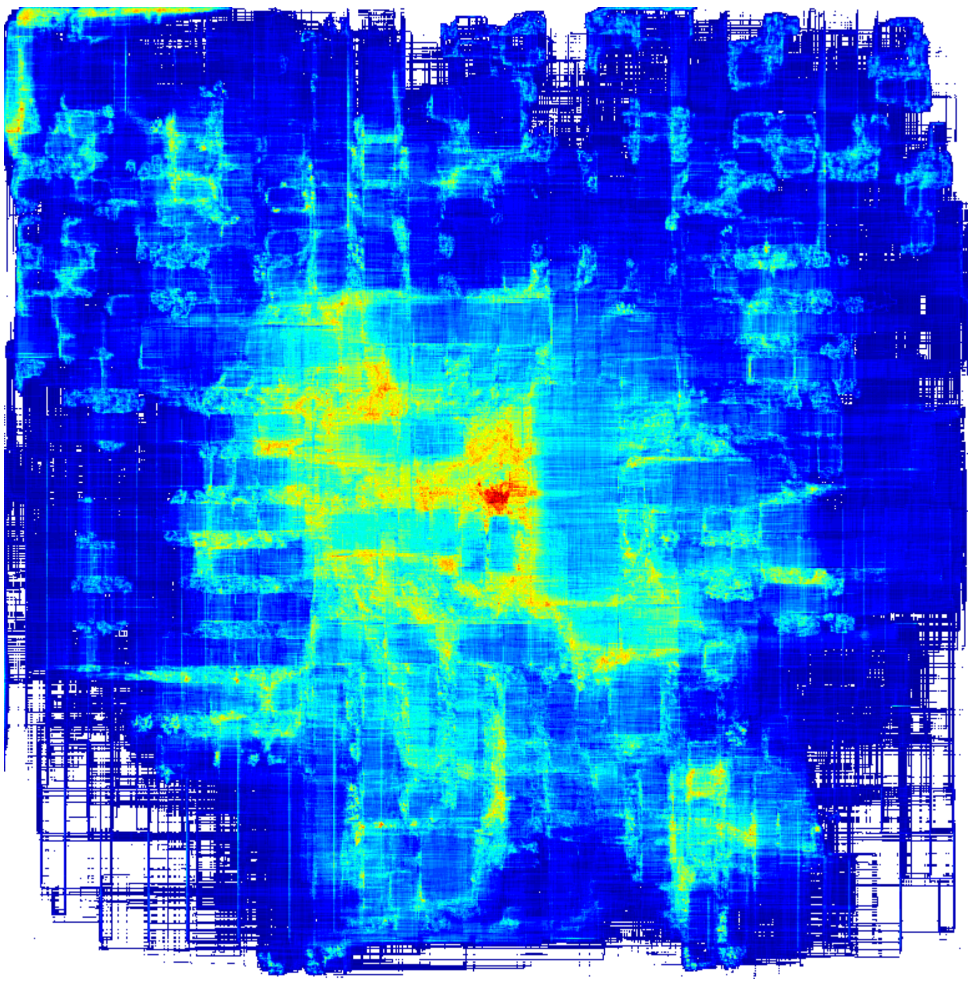
Placement

Optimization targets

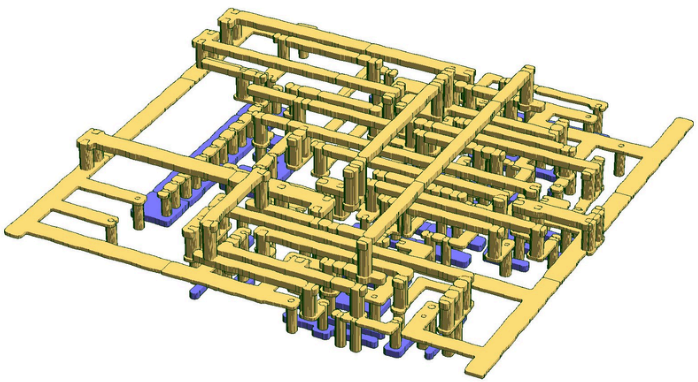
- Satisfy timing constraints?
- Satisfy power constraints?
- 100% routable?
- Wirelength minimized?
- …

[Curtesy NTU team]

Routing Example



[Curtesy Umich]

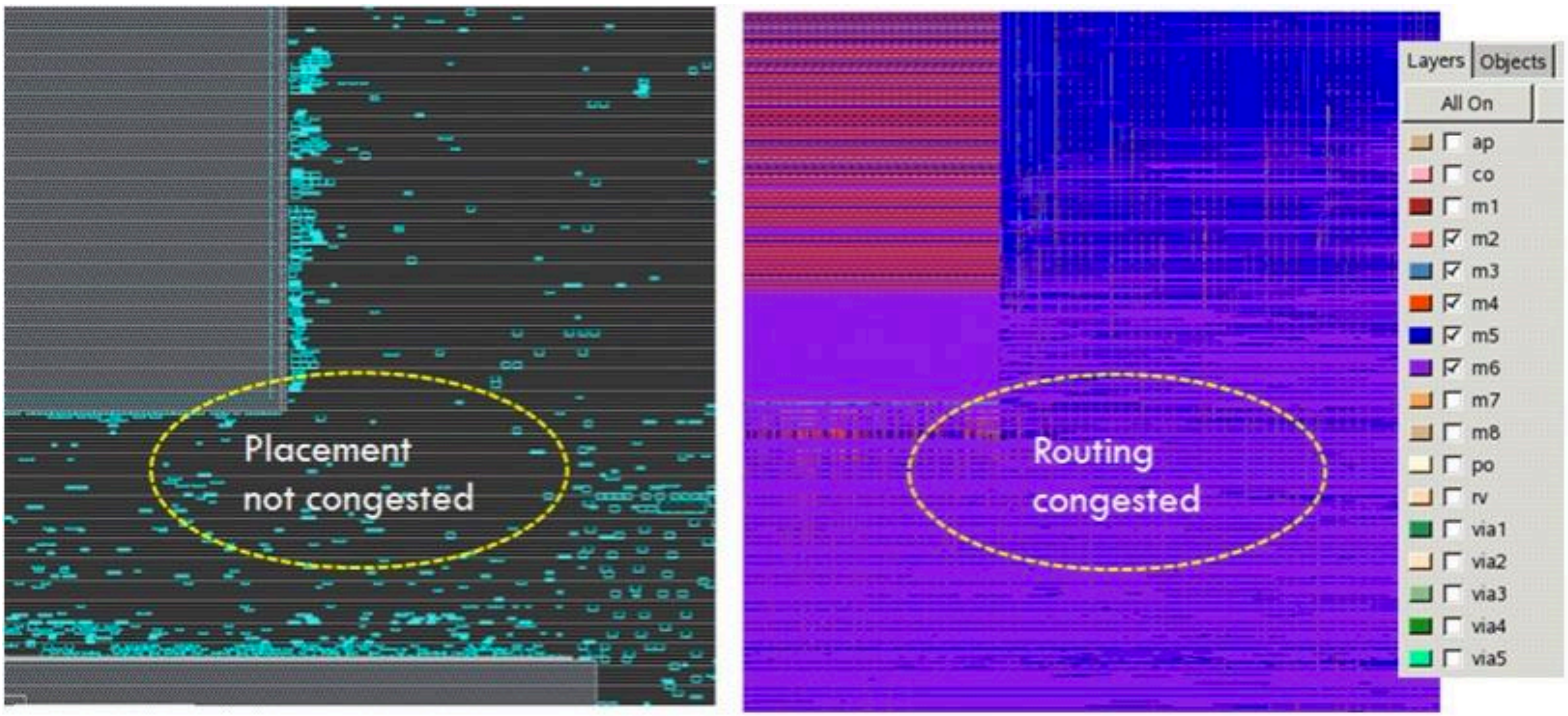


A zoom-in 3D view
[curtesy samyzaf]

Challenging problem

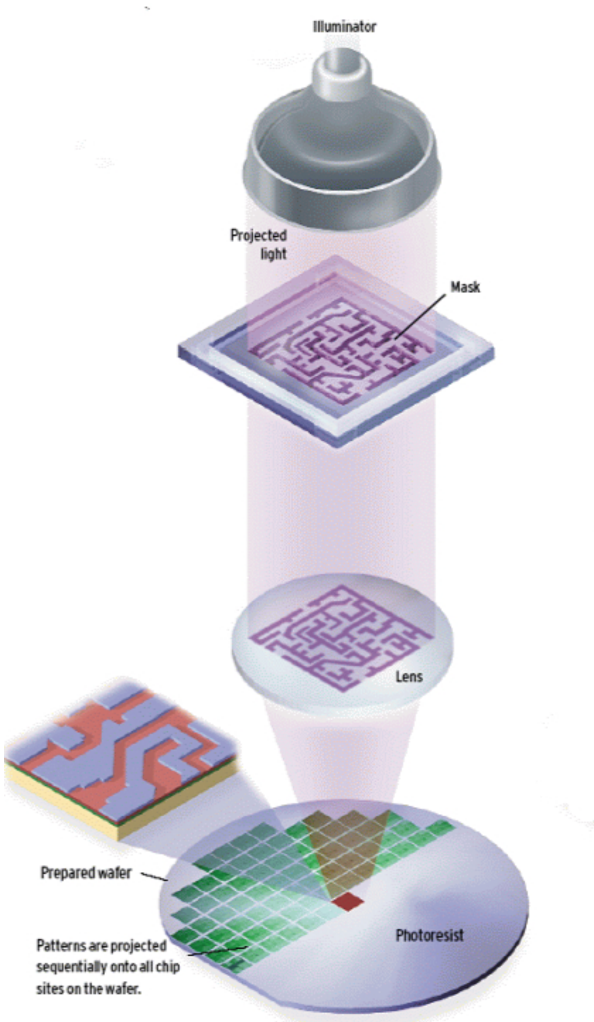
- 10+ metal layers
- Millions of nets
- May be highly congested
- Minimize wirelength

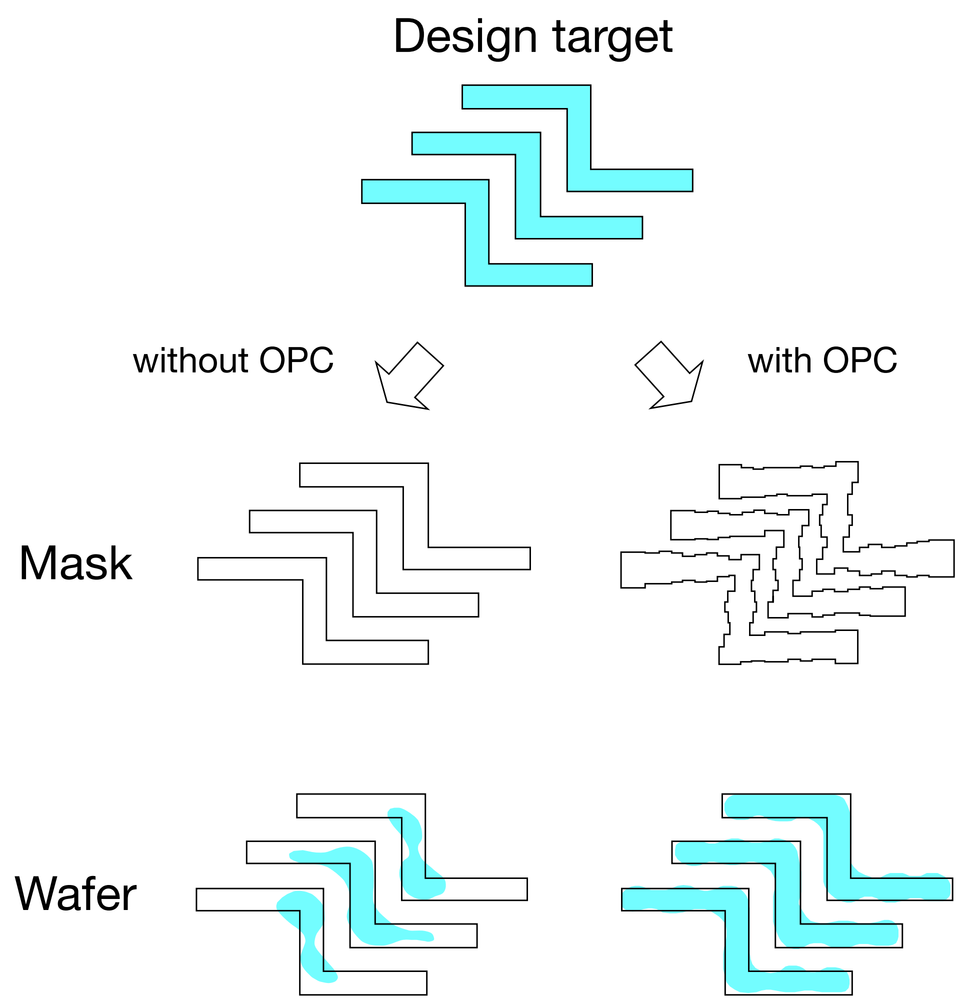
High Correlation between P&R



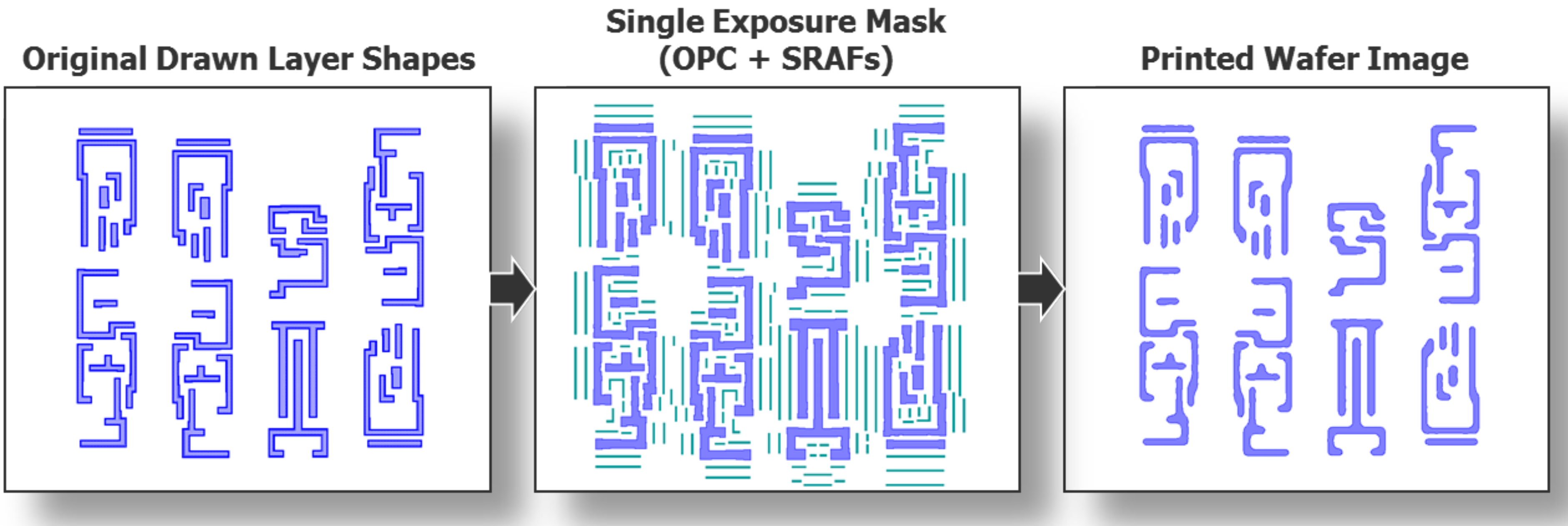
Curtesy semiwiki

Mask Synthesis



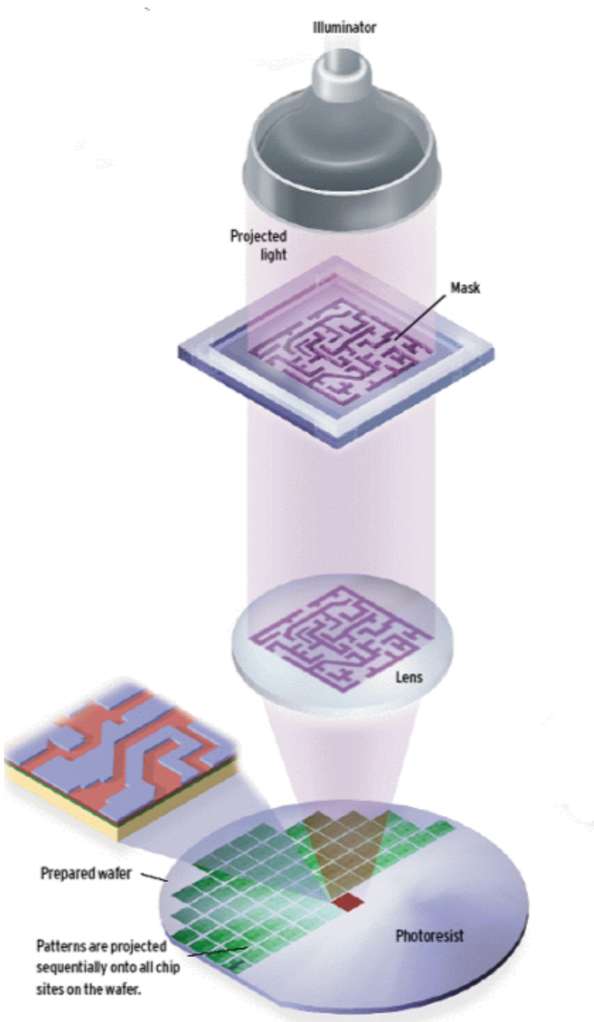


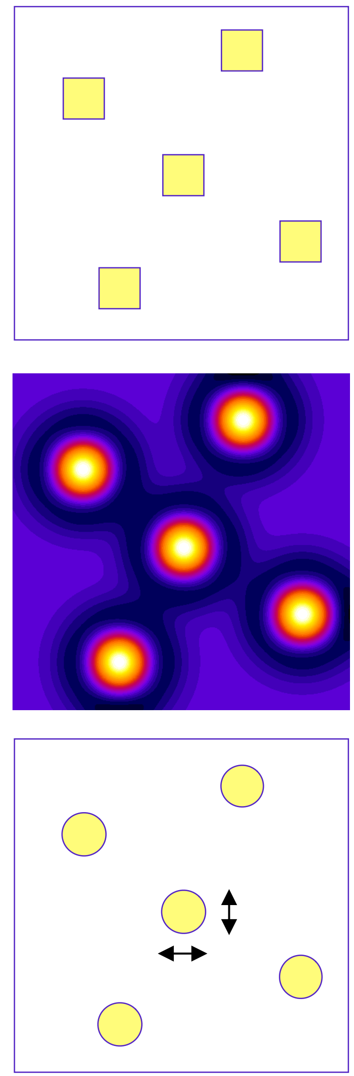
Mask Synthesis: OPC+SRAF



[Curtesy Semiengineering]

Mask Verification: Lithography Simulation



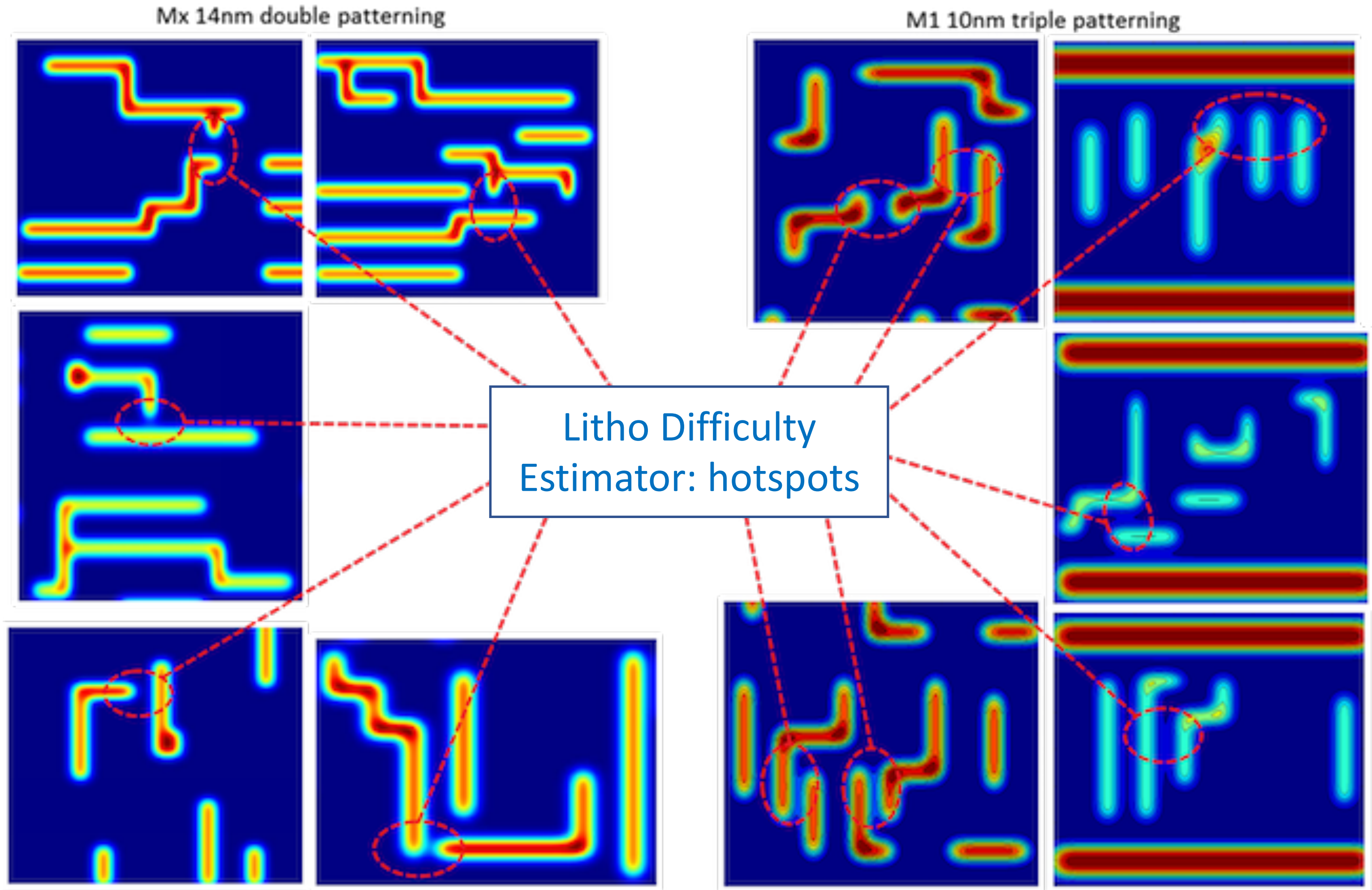


Contact Mask

Aerial Image
(Light intensity map)

Resist Pattern

Mask Verification: Lithography Hotspots

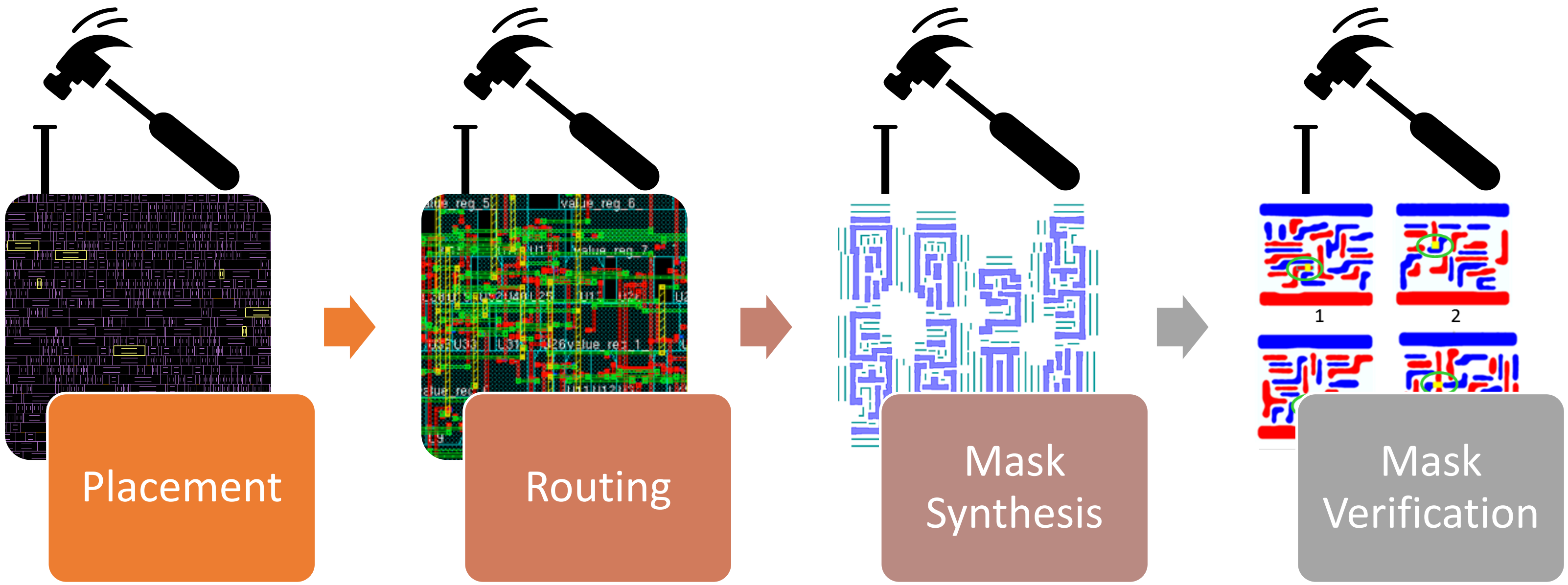


[Curtesy Tech Design Forums]

Challenges for VLSI Design

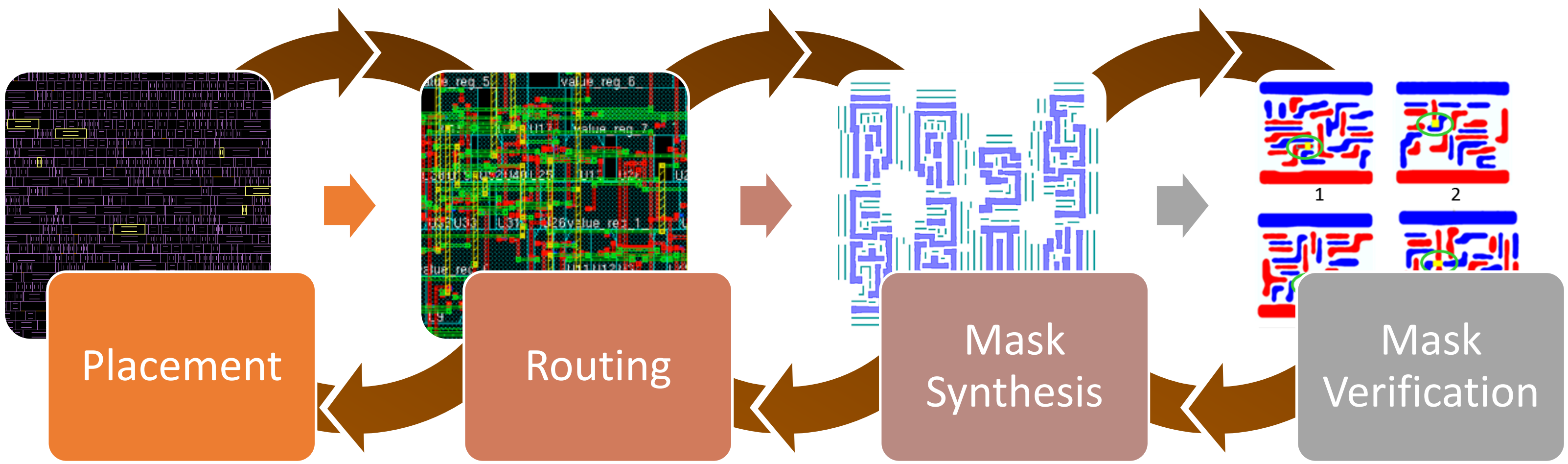
- Long and complicated design flow
- Nested chicken-egg loops
  - P&R, OPC&SRAF…
- Nearly all problems are NP-hard
- Stacking of metaheuristics
- High expectation to optimality
  - Shoot for even 1% improvement
- Single iteration is expensive
  - One iteration of backend flow may take days
- Require many iterations for convergence

How Machine Learning Can Help



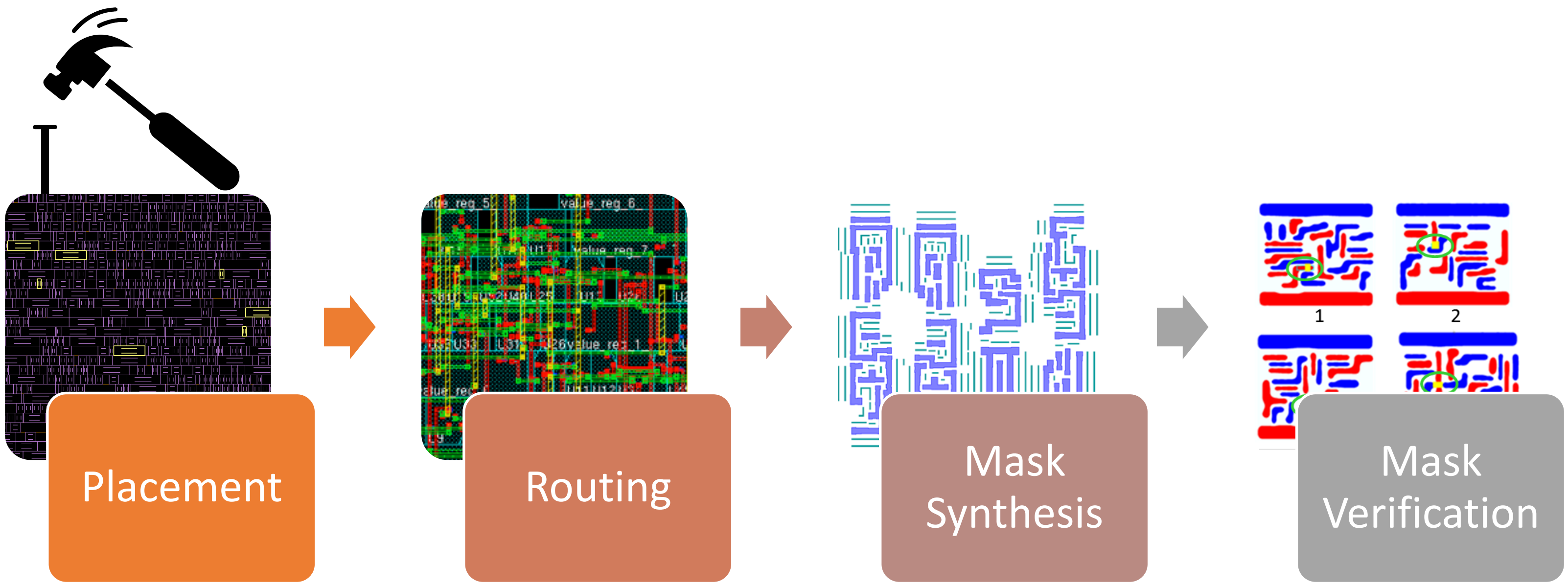
Hammers to tackle each step

How Machine Learning Can Help



Bridges to connect each step

Placement



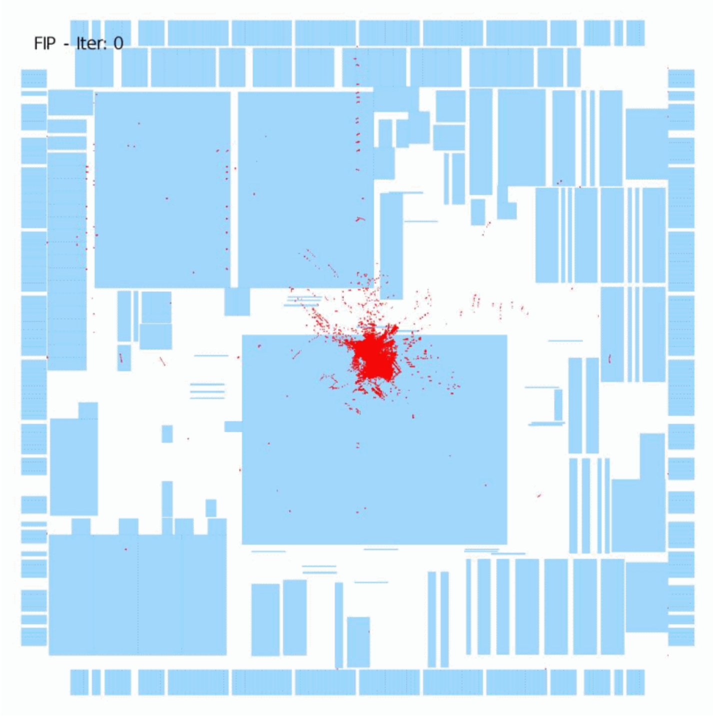
VLSI Placement

Placement is critical to VLSI design quality and design closure

Input
Gate-level netlist
Standard cell library

Output
Legal placement solution

Objective
Optimize wirelength, routability, etc.



RePlAce

Nonlinear Placement Algorithm

$$\min_{\mathbf{x},\mathbf{y}} \quad WL(\mathbf{x},\mathbf{y}),$$
$$\text{s.t.} \quad D(\mathbf{x},\mathbf{y}) \leq t_d$$

Objective of nonlinear placement

$$\min \quad \underbrace{\left(\sum\nolimits_{e \in E} WL(e;\mathbf{x},\mathbf{y})\right)}_{\text{Wirelength}} + \lambda \underbrace{D(\mathbf{x},\mathbf{y})}_{\text{Density}}$$

Challenges of Nonlinear Placement

Low efficiency

- > 3h for 10M-cell design

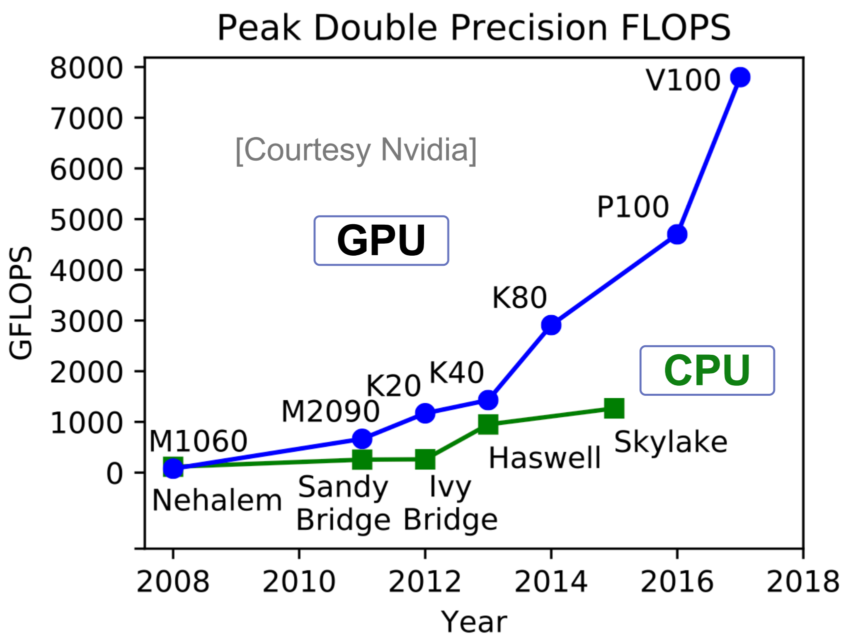
Limited acceleration

- Limited speedup, e.g., mPL, due to clustering

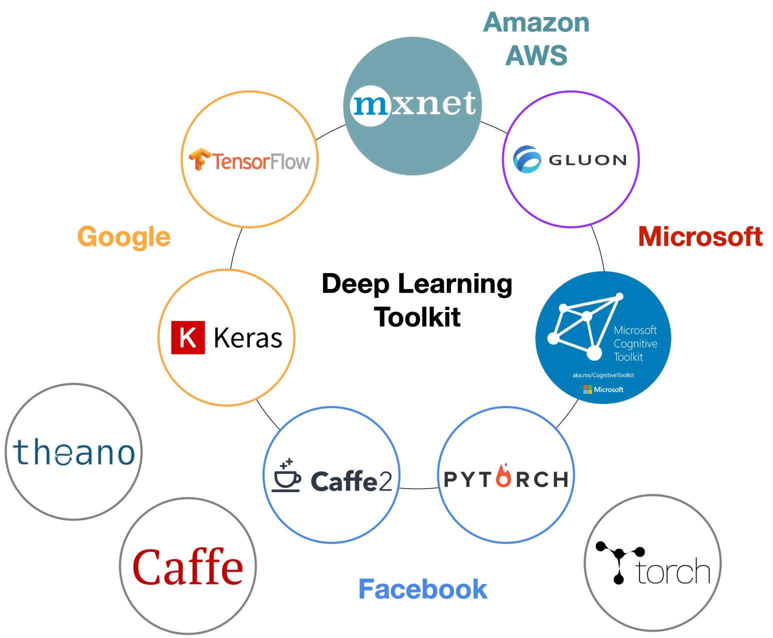
Huge development effort

- > 1 year for ePlace/RePlAce

Advances in Deep Learning Hardware/Software



Over **60x** speedup in neural network training since 2013



Deep learning toolkits

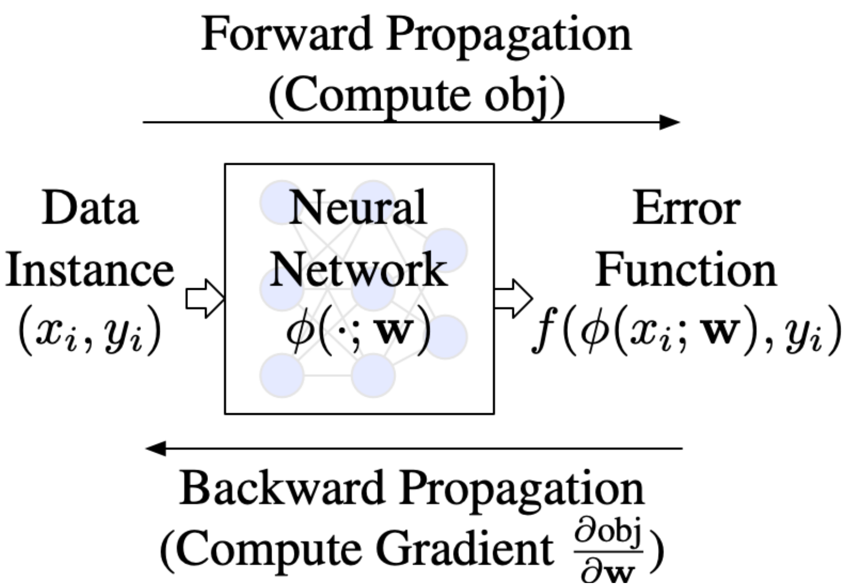
DREAMPlace Strategies

- We propose a novel analogy by casting the nonlinear placement optimization into a neural network training problem
- Greatly leverage deep learning hardware (GPU) and software toolkit (e.g., PyTorch)
- Enable ultra-high parallelism and acceleration while getting the state-of-the-art results

[Lin+,DAC'19]
Best paper award

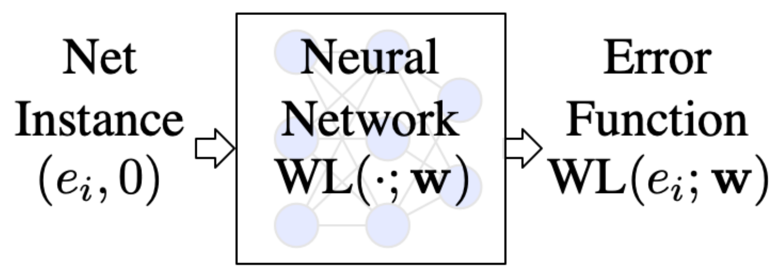
Analogy between NN Training and Placement

$$\min_{\mathbf{w}} \sum_{i}^{n} f(\phi(x_i; \mathbf{w}), y_i) + \lambda R(\mathbf{w})$$



Train a neural network

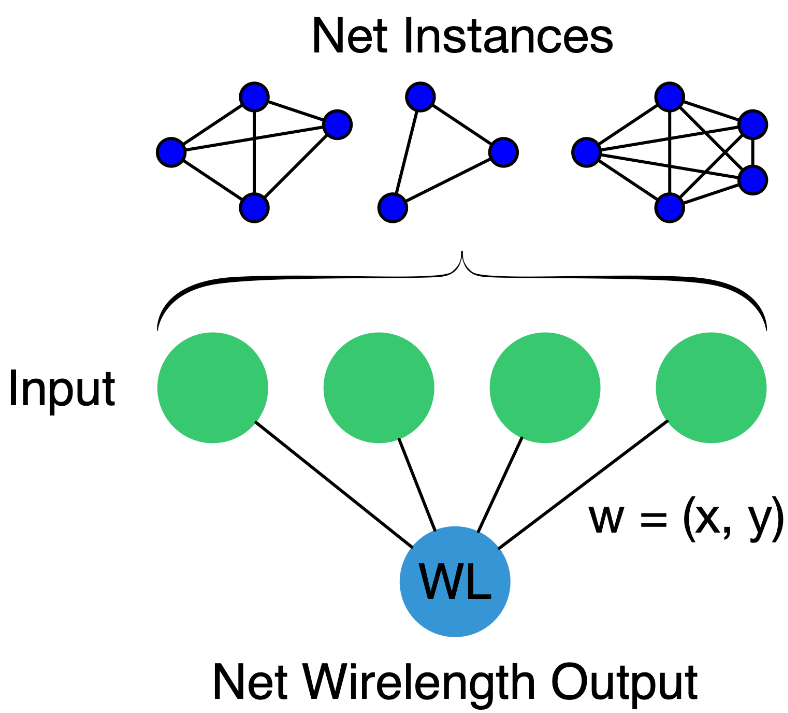
$$\min_{\mathbf{w}} \sum_{i}^{n} \mathrm{WL}(e_i; \mathbf{w}) + \lambda D(\mathbf{w})$$



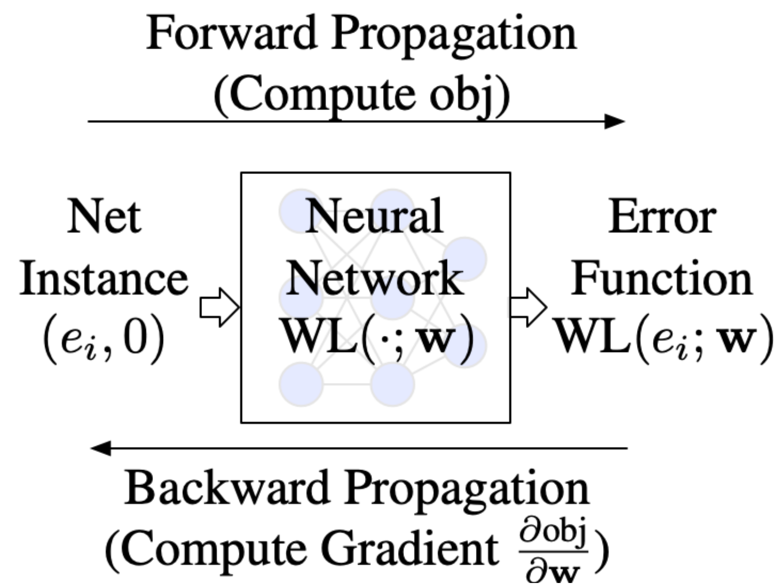
Solve a placement

Analogy between NN Training and Placement

Casting the placement problem into neural network training



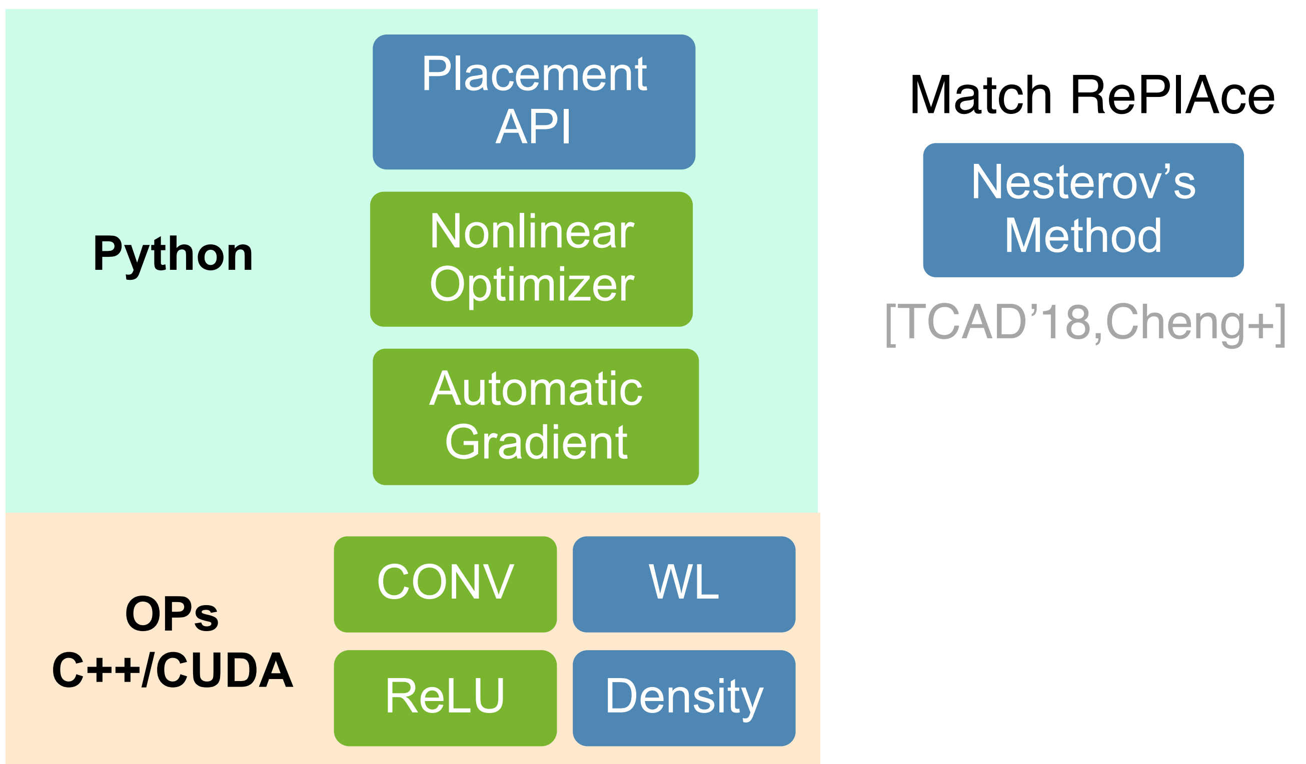
$$\min_{\mathbf{w}} \sum_{i}^{n} \mathrm{WL}(e_i; \mathbf{w}) + \lambda D(\mathbf{w})$$



Train a neural network ⟷ **Solve a placement**

DREAMPlace Architecture

Leverage highly optimized deep learning toolkit PyTorch



Experimental Results

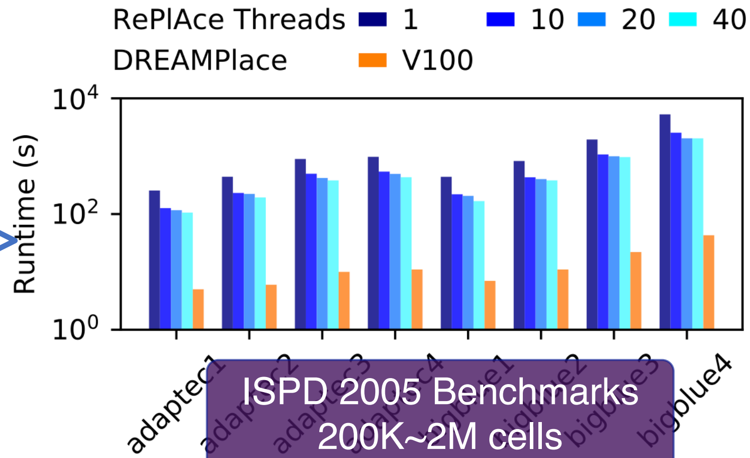
DREAMPlace
- CPU: Intel E5-2698 v4 @2.20GHz
- GPU: 1 NVIDIA Tesla V100
- Single CPU thread was used

RePlAce [TCAD'18,Cheng+]
- CPU: 24-core 3.0 GHz Intel Xeon
- 64GB memory allocated

Same quality of results!

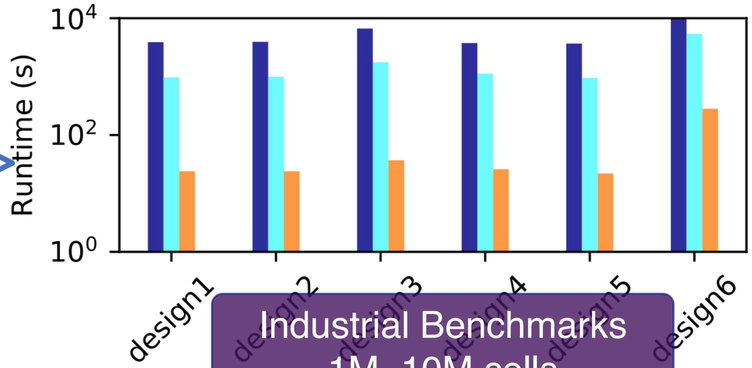
10M-cell design
finishes within **5min c.f. 3h**

34x speedup



ISPD 2005 Benchmarks
200K~2M cells

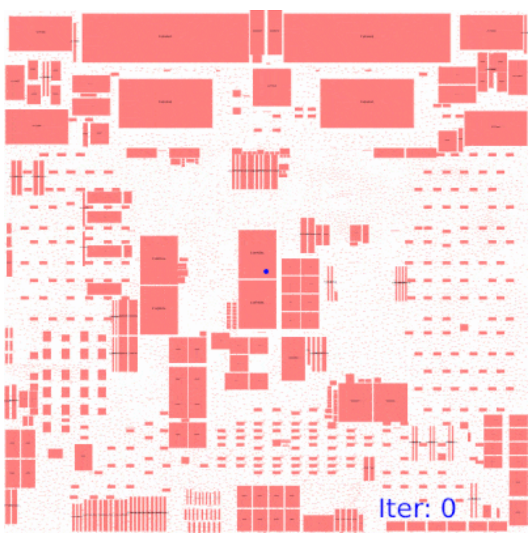
43x speedup

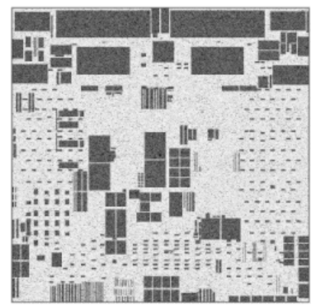


Industrial Benchmarks
1M~10M cells

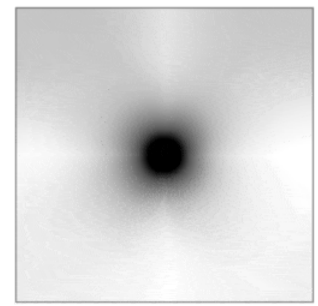
[Lin+,DAC'19]

Bigblue4 (2M-Cell Design)

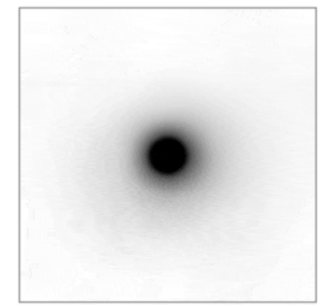




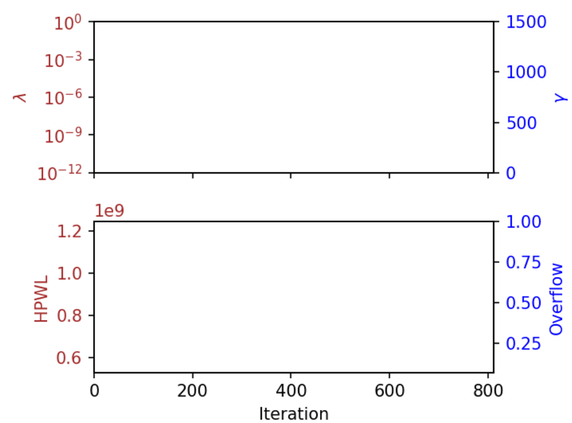
Density Map



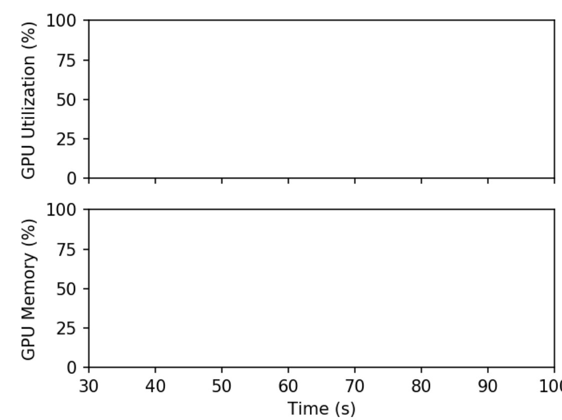
Potential Map



Field Map



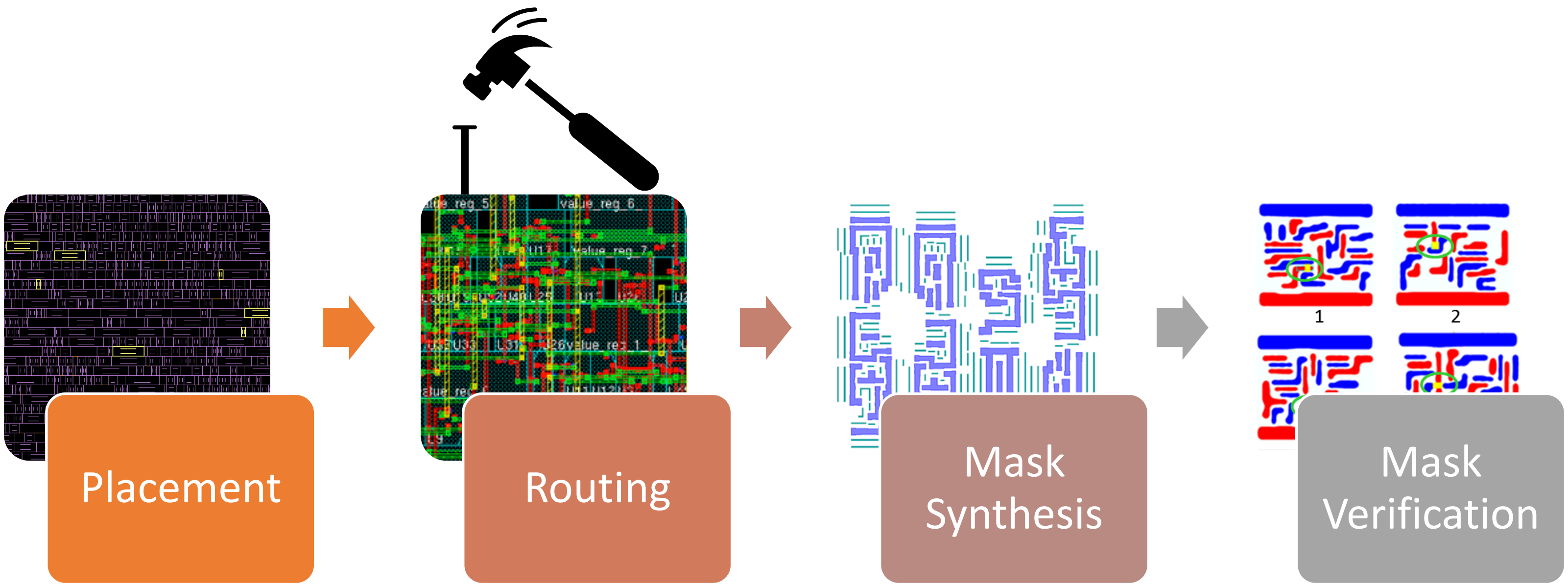
Placement Metrics



GPU Usage on Titan Xp

Code release: https://github.com/limbo018/DREAMPlace

Routing



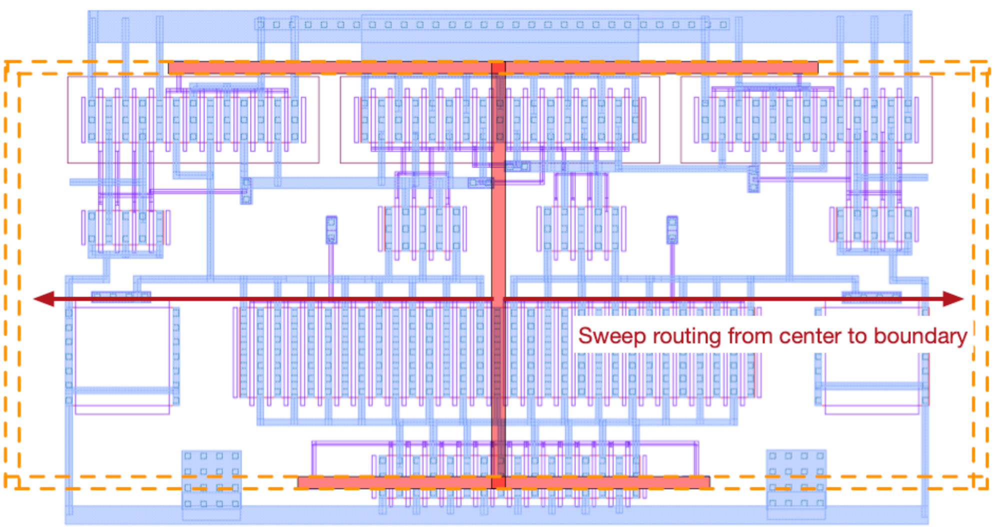
Routing Guidance: GeniusRoute

Routing for analog circuits, e.g., comparator

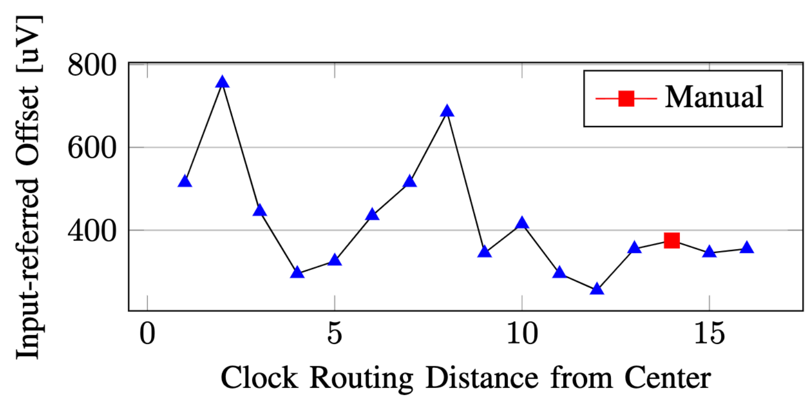
- Sensitive performance to clock routing

Existing manual layouts

- Hard to encode designer expertise into rules



Sweep the routing of the clock net



[Zhu+,ICCAD'19]

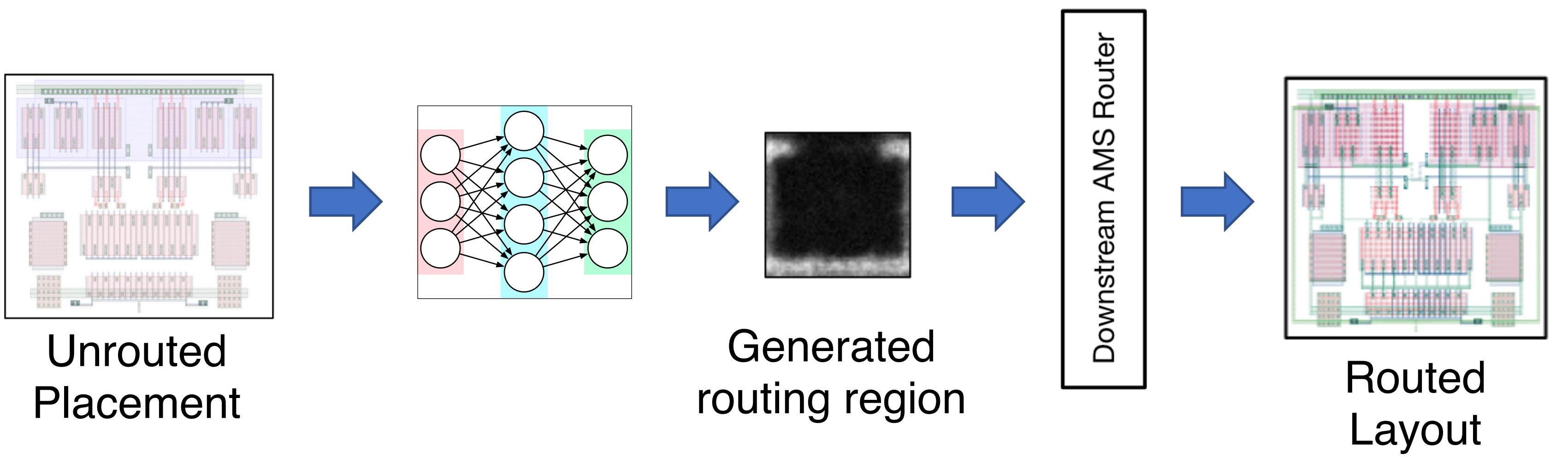
Routing Guidance: GeniusRoute

Learn from manual layouts

- Encode designer expertise into neural networks!

Generate routing guidance for critical nets

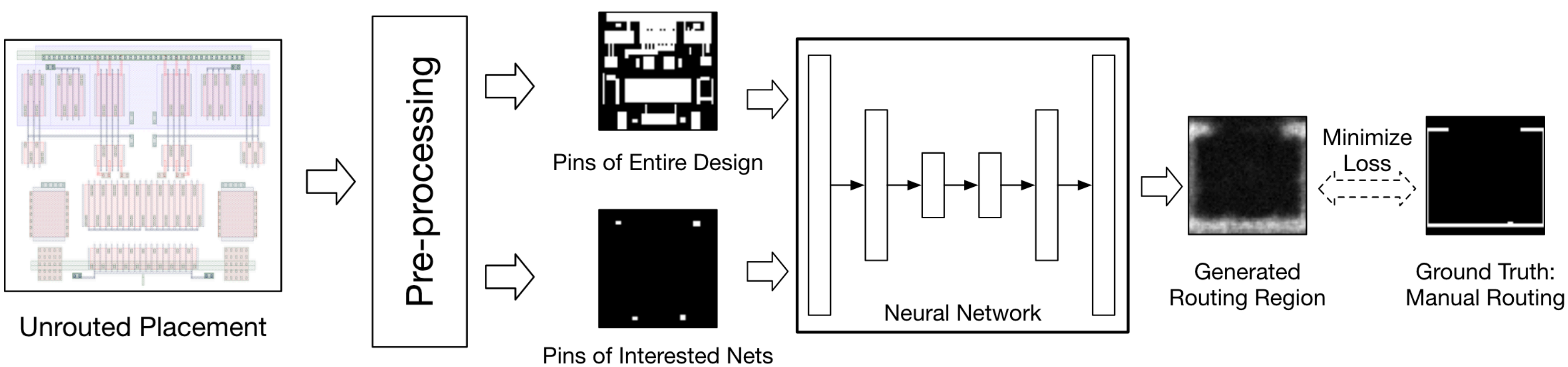
- Clock, power/ground, critical signal nets
- Probability map of routing



[Zhu+,ICCAD’19]

Routing Guidance: GeniusRoute

- Adopt autoencoder for routing region generation
- Compare with routing from manual layouts



[Zhu+,ICCAD'19]

Routing Guidance: GeniusRoute

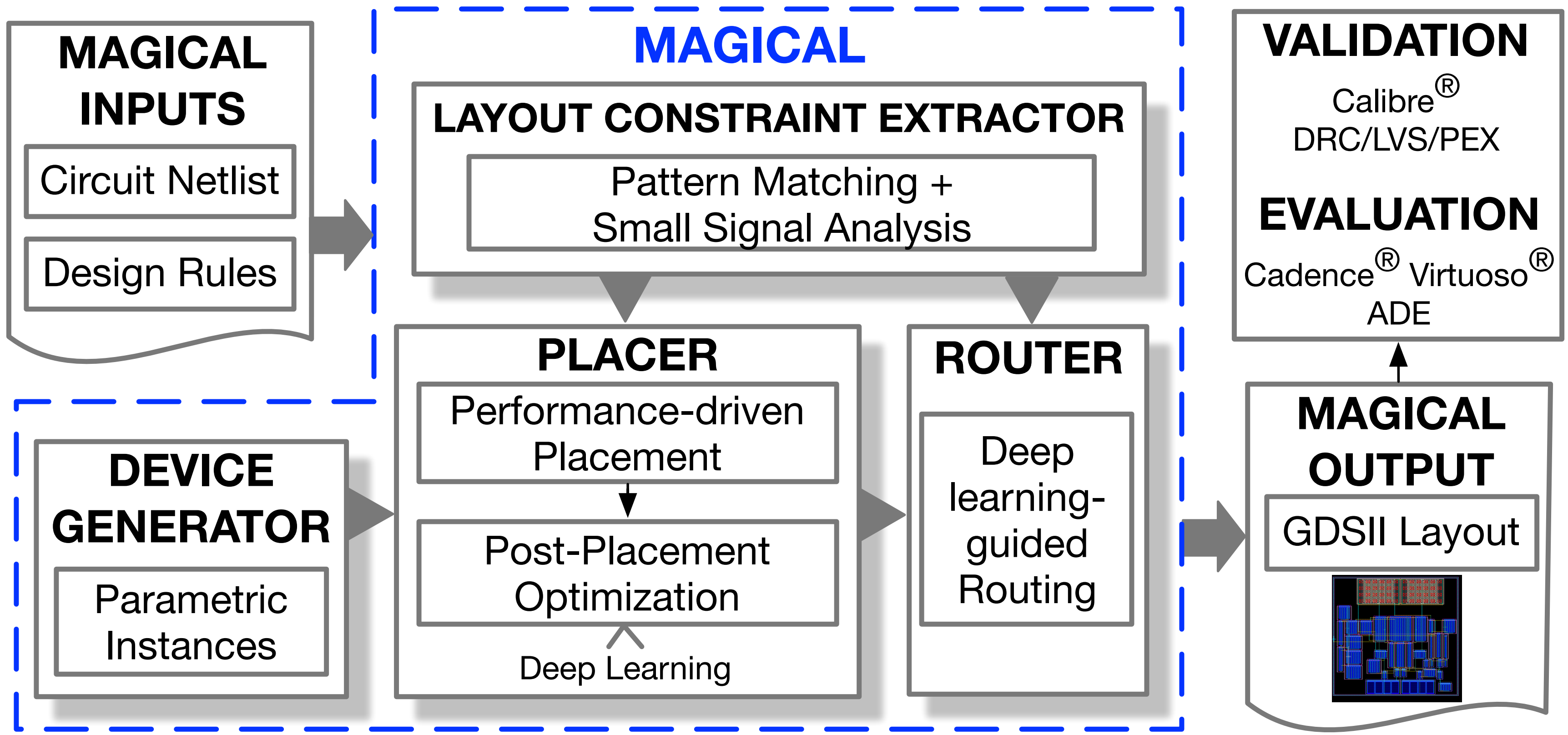
- Test on comparators and OTAs
- Evaluate with post layout simulation
- Compare with manual layout and previous methods

	Schematic	Manual	ICCAD'10	W/o guide	GeniusRoute
Offset (μV)	/	480	1230	2530	**830**
Delay (ps)	102	170	180	164	**163**
Noise (μV_{rms})	439.8	406.6	437.7	439.7	**420.7**
Power (μW)	13.45	16.98	17.19	16.82	**16.80**

Closest results to the manual layout

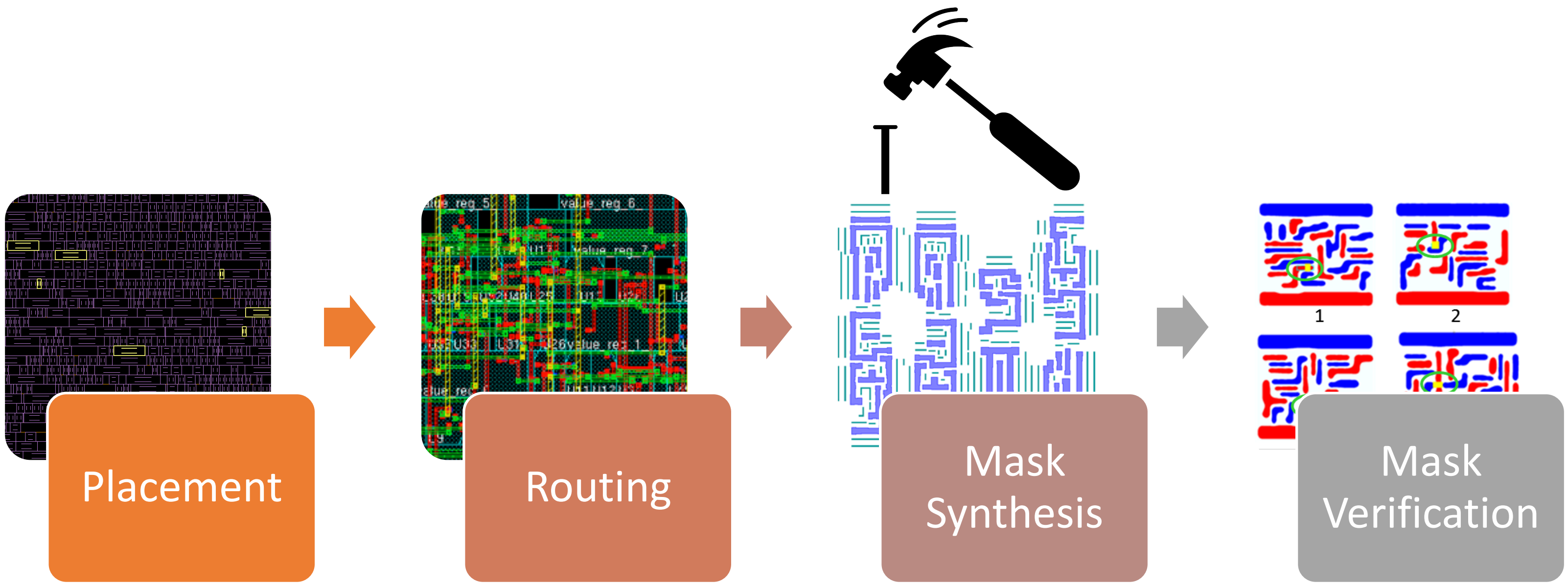
[Zhu+,ICCAD'19]

MAGICAL (Machine Generated Analog Layout) Overview

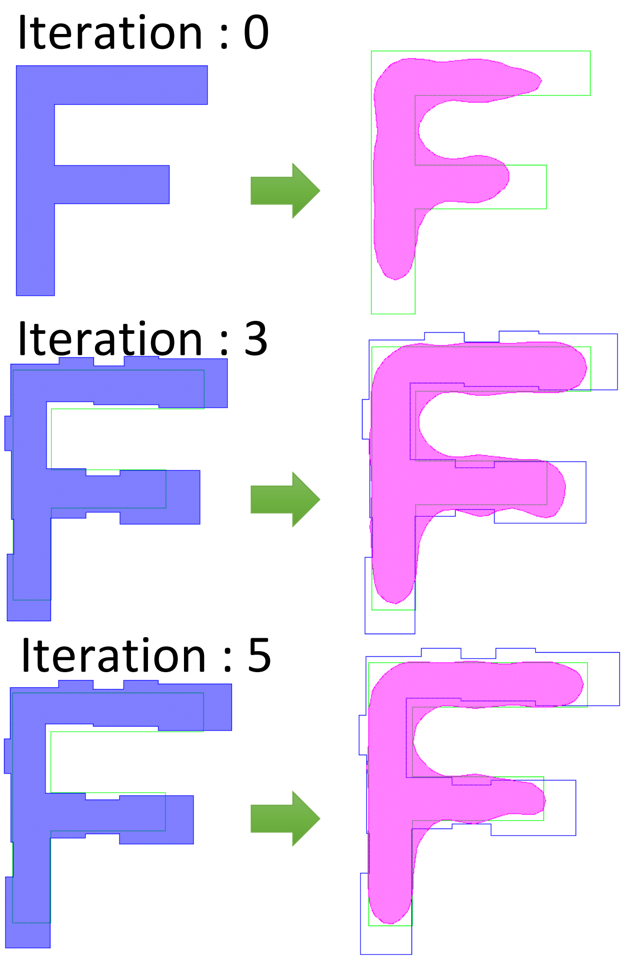


- MAGICAL 0.2 Version Open-source released in May 2019
- GitHub: https://github.com/magical-eda/MAGICAL
  - **End-to-end** analog layout generation from netlist to GDSII
  - **No dependency** on commercial tools
  - **Push-button, no-human-in-the-loop**

Mask Synthesis

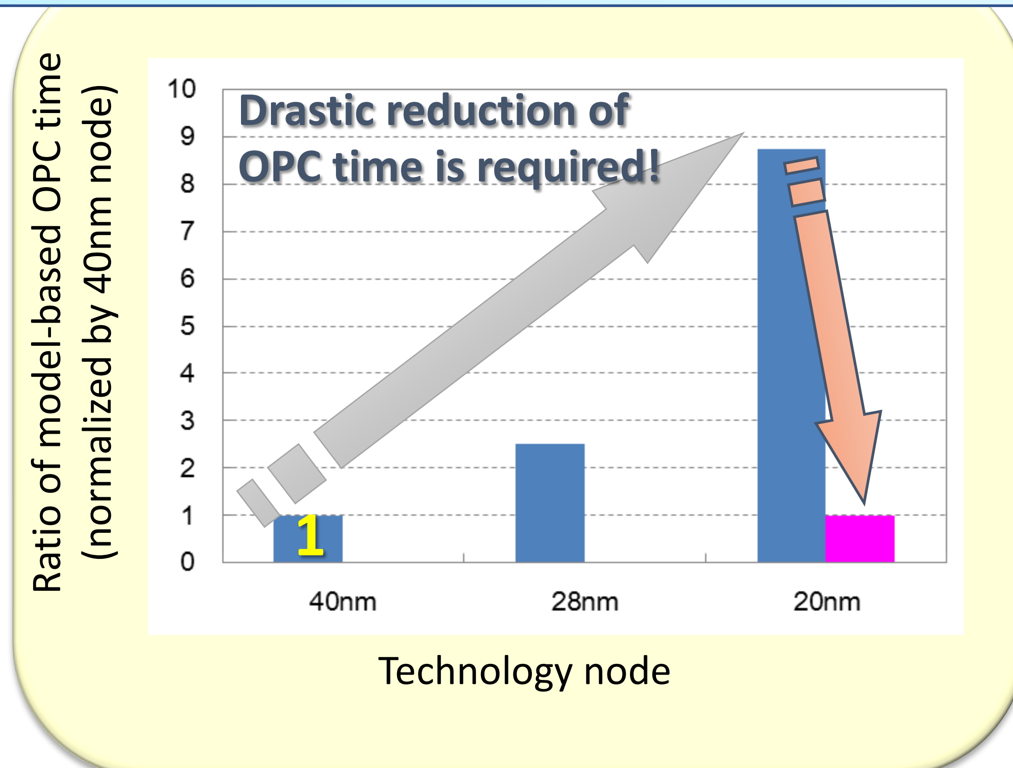


Optical Proximity Correction



Mask image **Printed image**

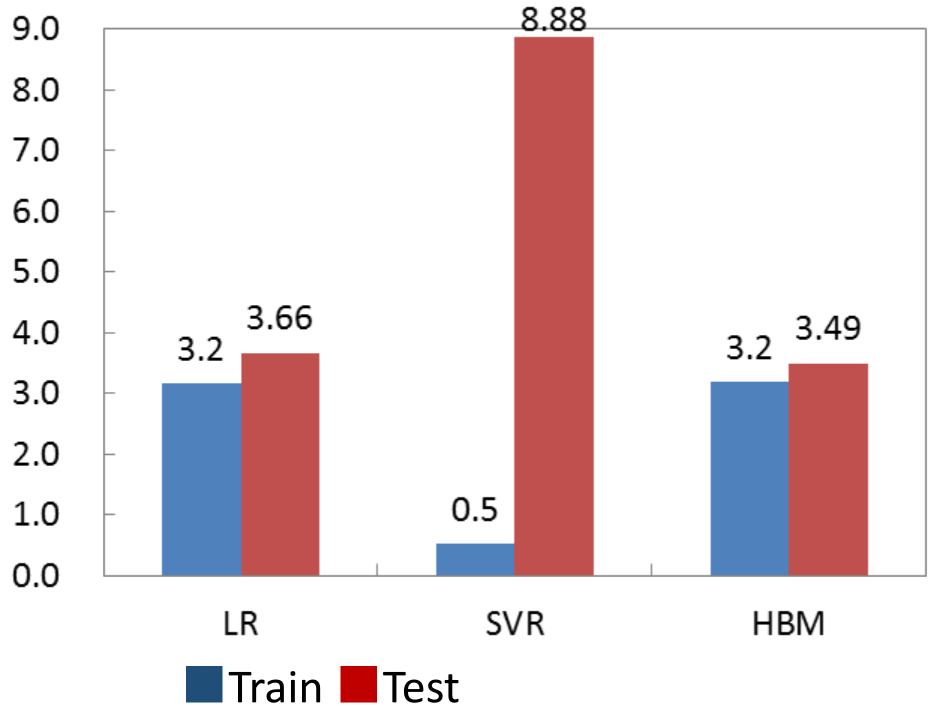
Issue: Conventional OPC consumes very long time
Goal: High accurate correction in shorter runtime



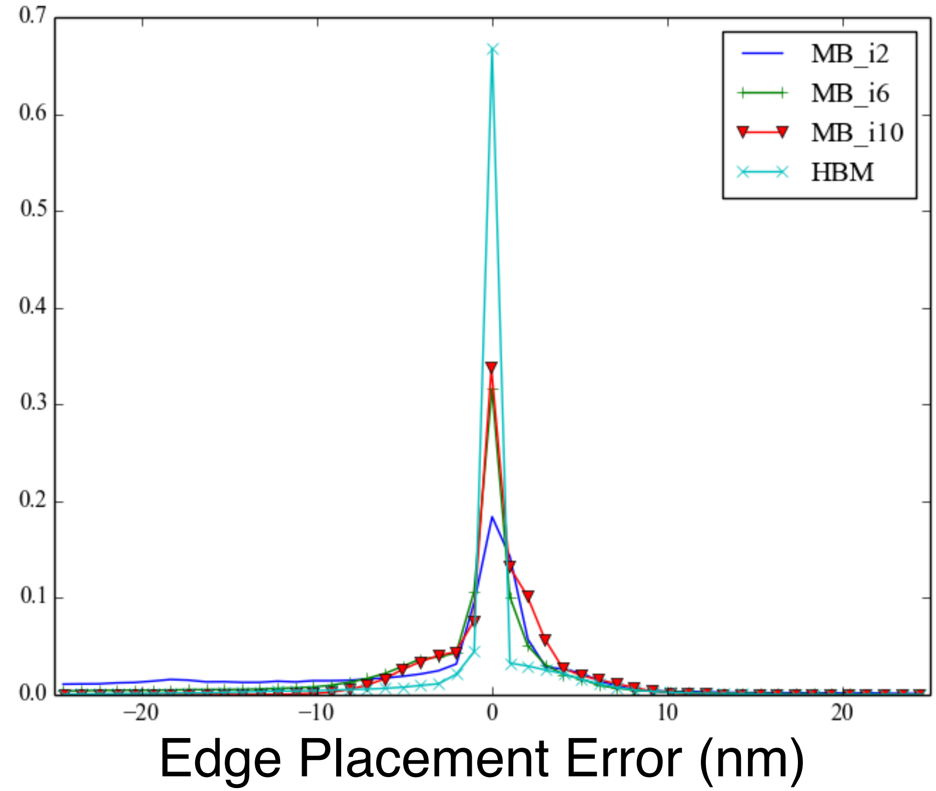
Model-based OPC is the most
widely used technique

Results on HBM-based OPC

RMSE (nm)
(Root mean square error)



Compare with model-based OPC @iteration 2, 6, 10



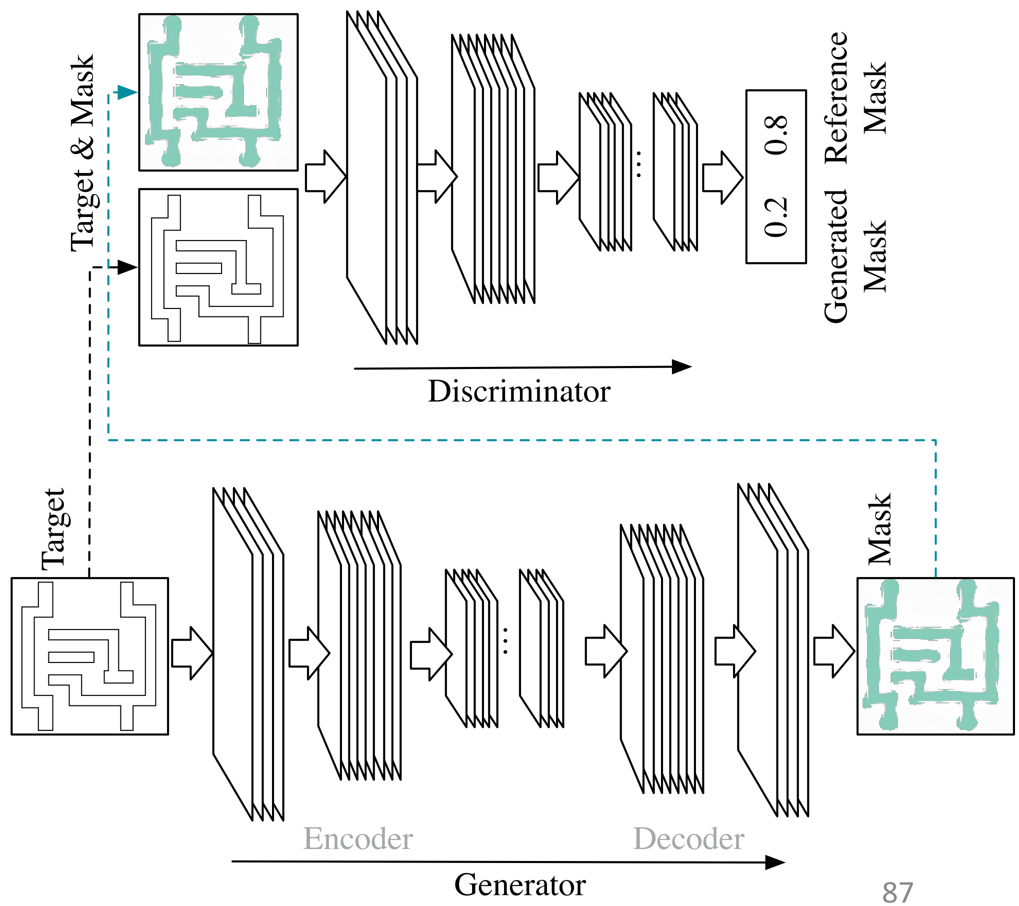
The number of iterations of model-based OPC can be reduced by 2x

[Matsunawa+, SPIE'15]

GAN for OPC

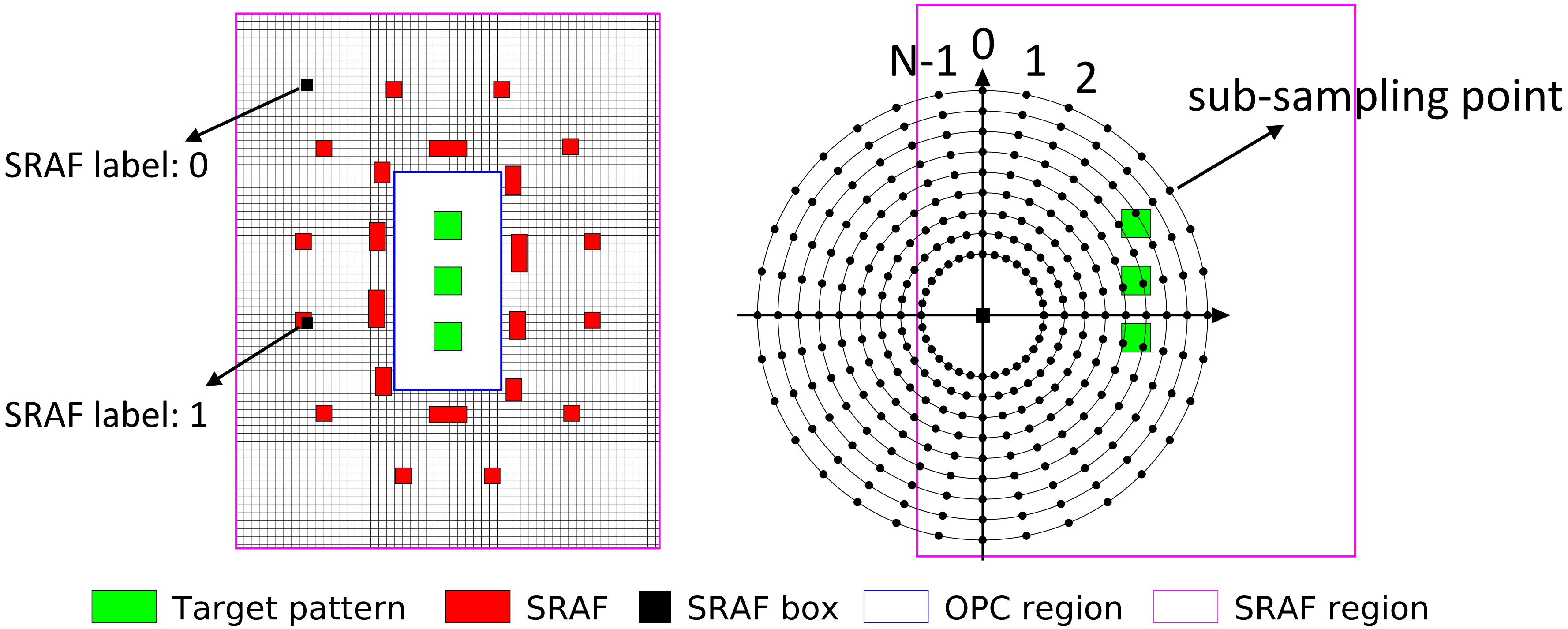
- Generative adversarial networks for OPC
  - [Yang+, DAC'18]

Compared with ILT flow
EPE error: 9% reduction
PV-band: 1% reduction
Overall runtime: 2x less



SRAF Insertion

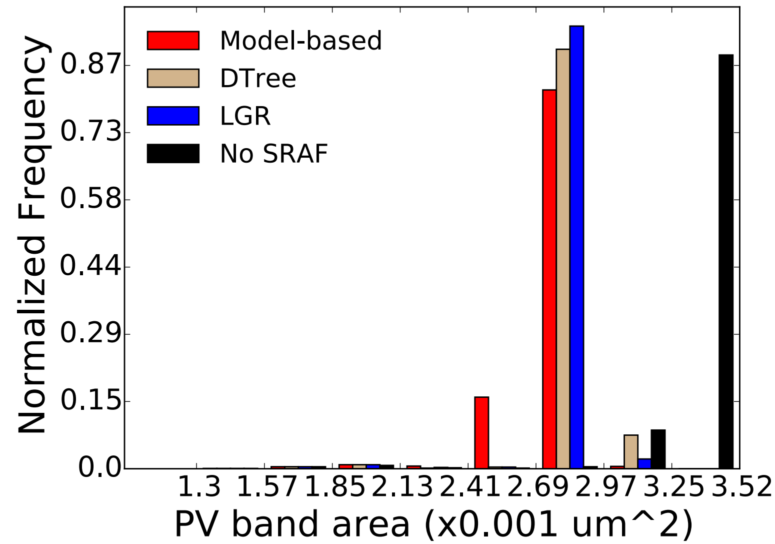
- SRAF (sub-resolution assist feature) insertion
- ML-based approach achieved similar result but 10x faster



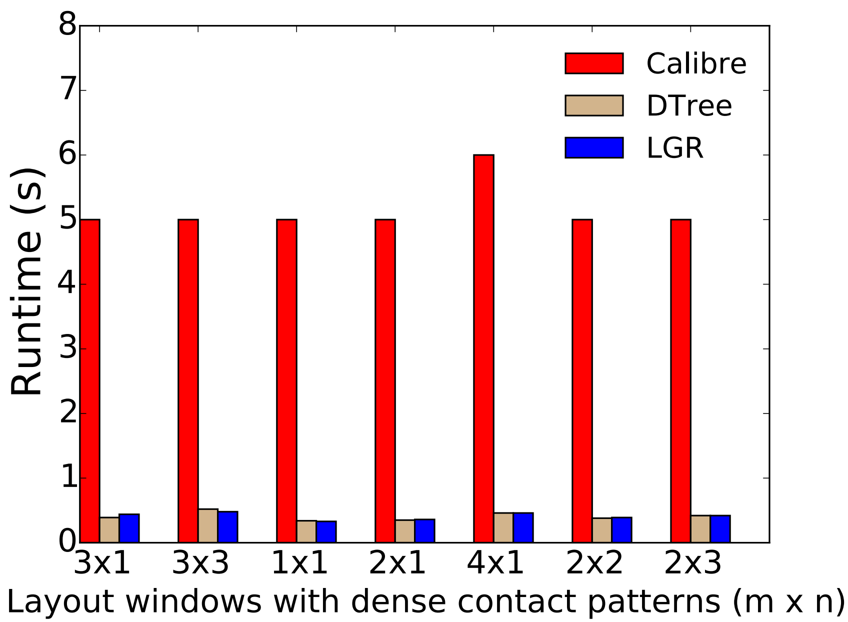
[Xu+, ISPD'16, TCAD'17]

Comparison with Model-based SRAF

- Similar PV band and EPE
- But 10x faster



<u>Can we do better?!</u>



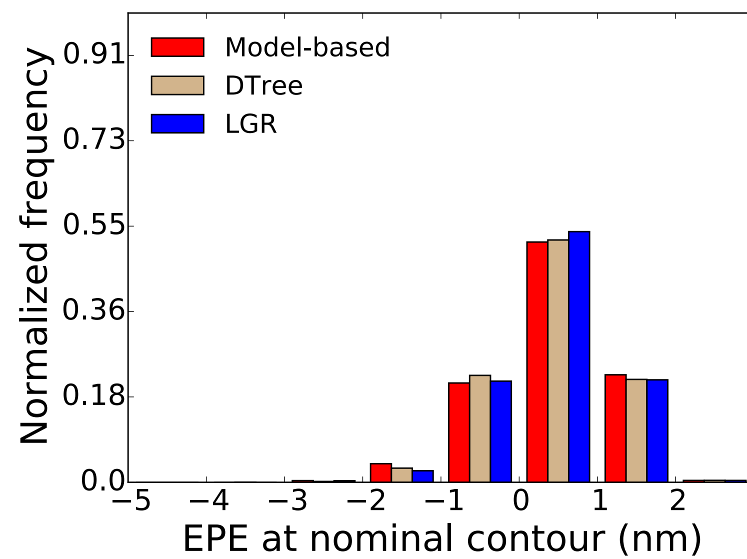


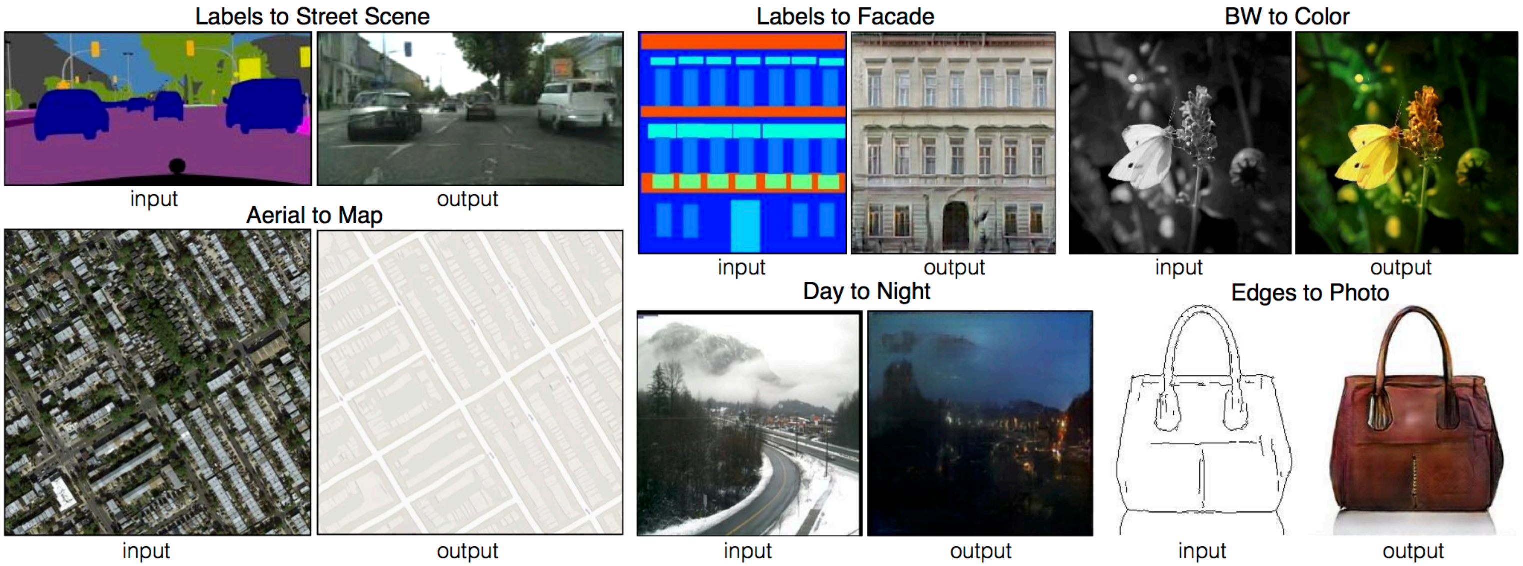
Image Translation with Generative Adversarial Networks

Generative Adversarial Network (GAN) [Goodfellow+, 2014]

- Two neural networks contest (generator and discriminator)
- Produces images similar to those in the training data set

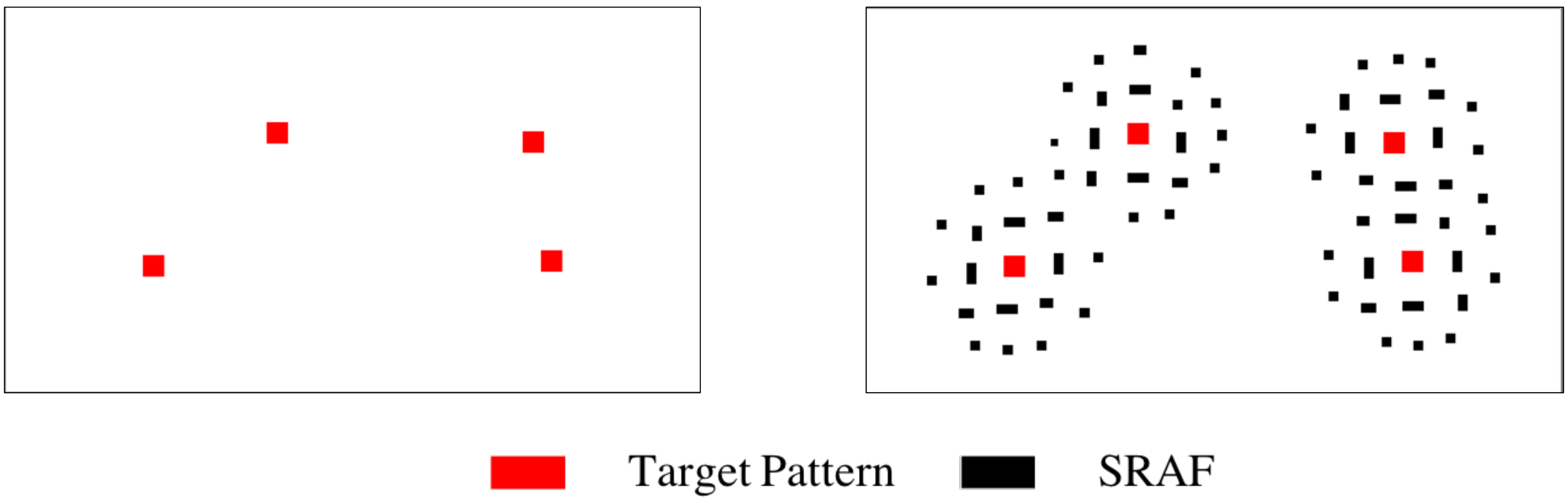
Conditional GAN (CGAN) for Image Translation [Isola+, CVPR'17]

- Takes an image in one domain and translate it to another one



SRAF Insertion & Image Translation

- What does SRAF generation have to do with Image translation?!



- Can we define the problem as translating images from the Target Domain (D_T) to the SRAF Domain (D_S)?

[Alawieh+,DAC'19]

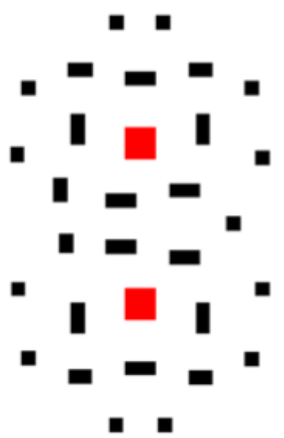
Challenges for SRAF Insertion

- Layout images have sharp edges which pose a challenge to GANs
  - No guaranteed to generate polygon SRAF shapes
  - Sharp edges can complicate gradient propagation
- Generated images need ultimately be changed to layout format
  - Images cannot be directly mapped to 'GDS' format
  - Post-processing step should not be time consuming
- *Hence, a proper* ***encoding*** *is needed to address these challenges!*

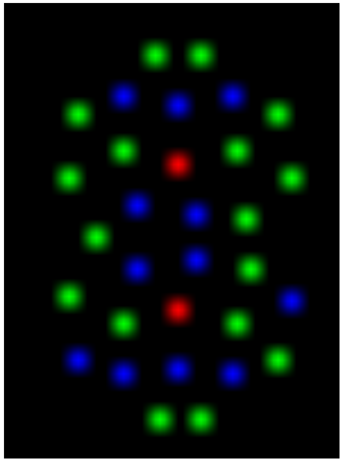
Multi-Channel Heatmap Encoding

Key Idea: encode each type of object on a separate channel in the image

- Channel index carries object description (type, size,...)
- Excitations on the channel carry objects location



Original Layout



Encoded Layout

CGAN Approach

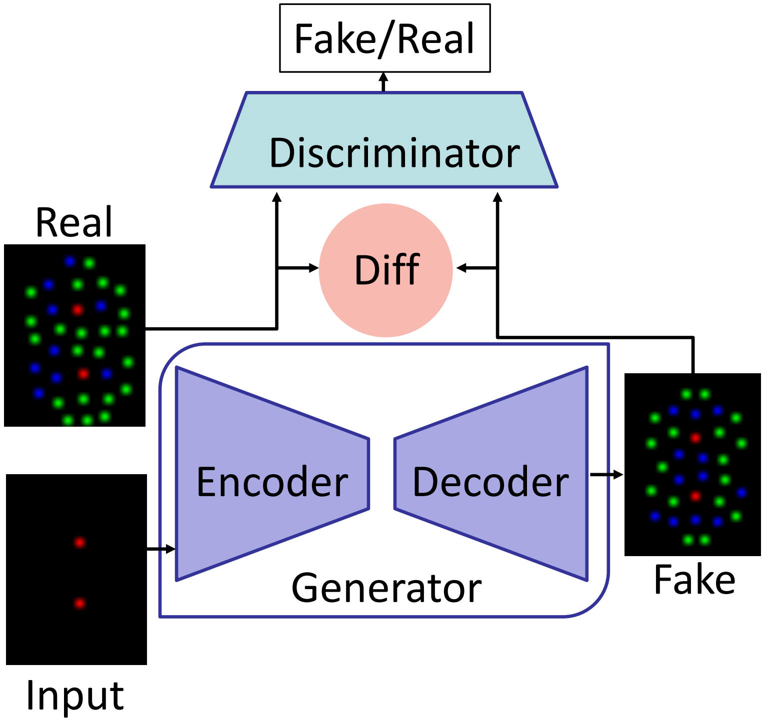
Generator:

- Trained to produce images in D_s based on input from D_T
- Tries to fool the Discriminator

Discriminator:

- Trained to detect 'fakes' generated by the Generator

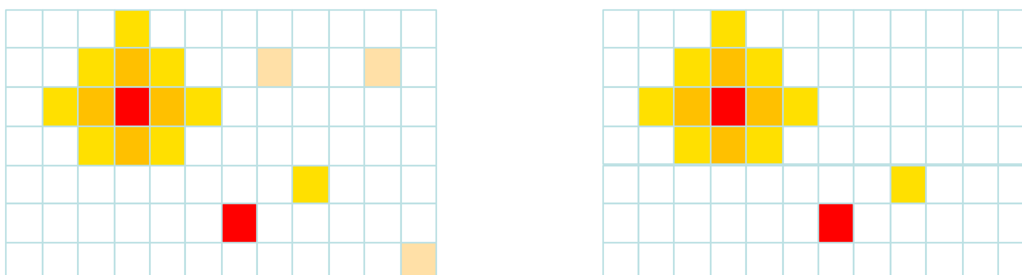
The two networks are jointly trained until convergence



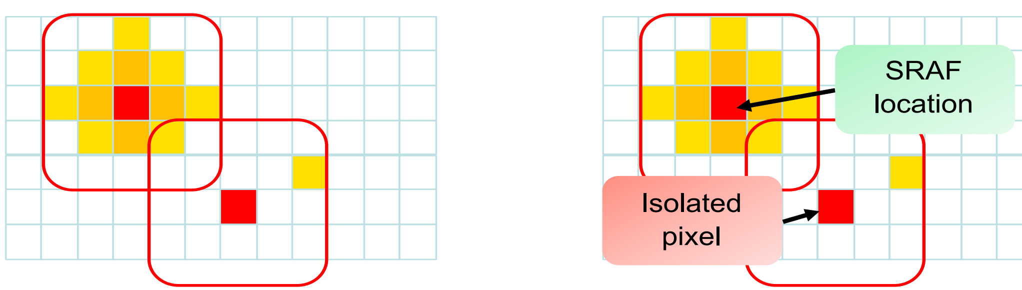
Results Decoding

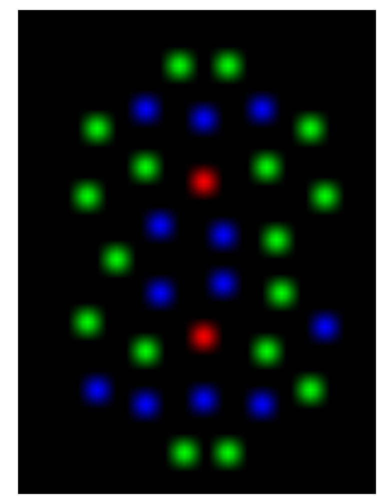
Decoding the generated layout images in two steps

- Thresholding
- Excitation detection



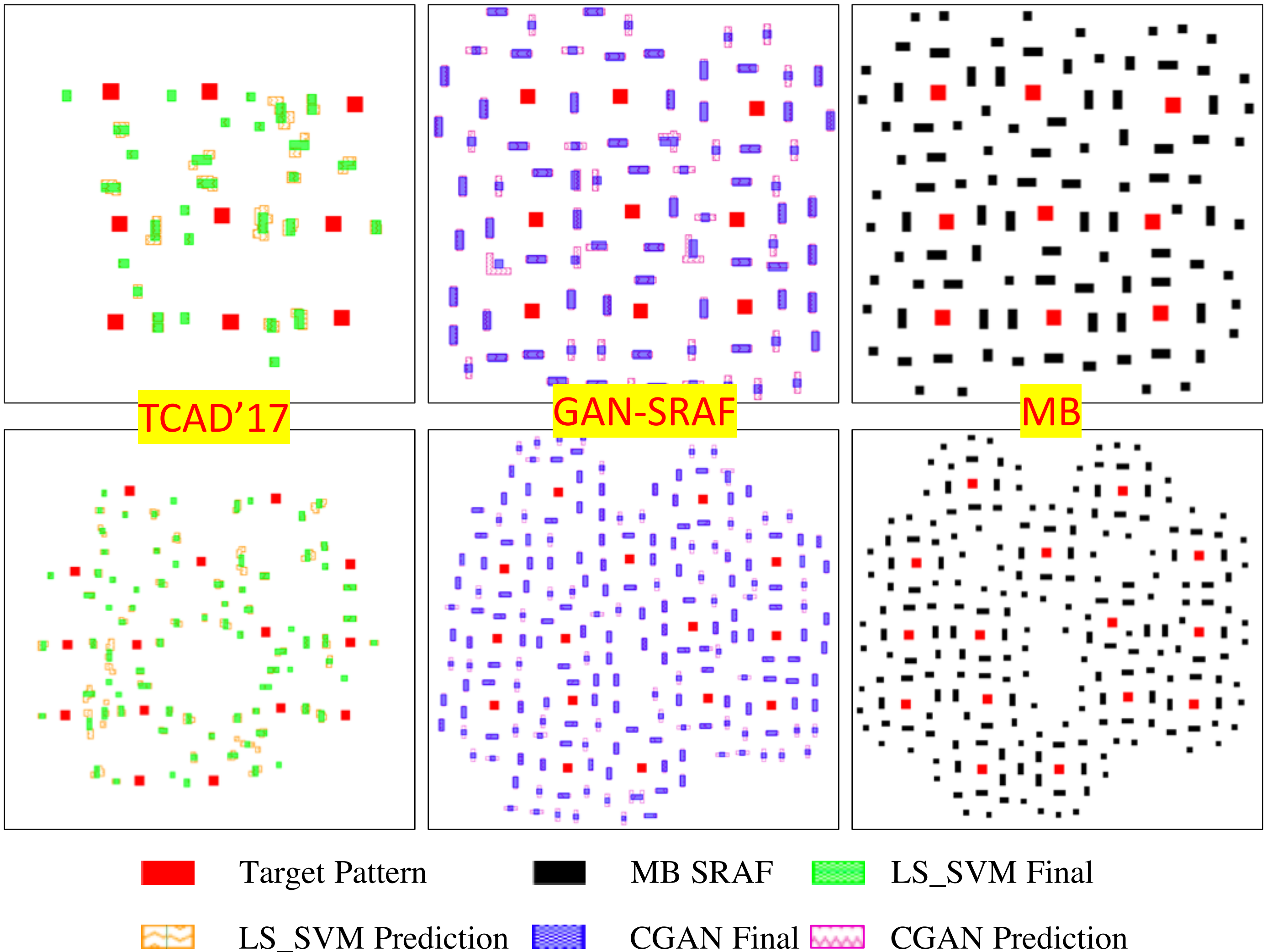
Decoding scheme is fast ➔ GPU accelerated





Sample Results

TCAD'17: [Xu+, ISPD'16, TCAD'17], SVM-based
MB: Model-Based Approach - Calibre



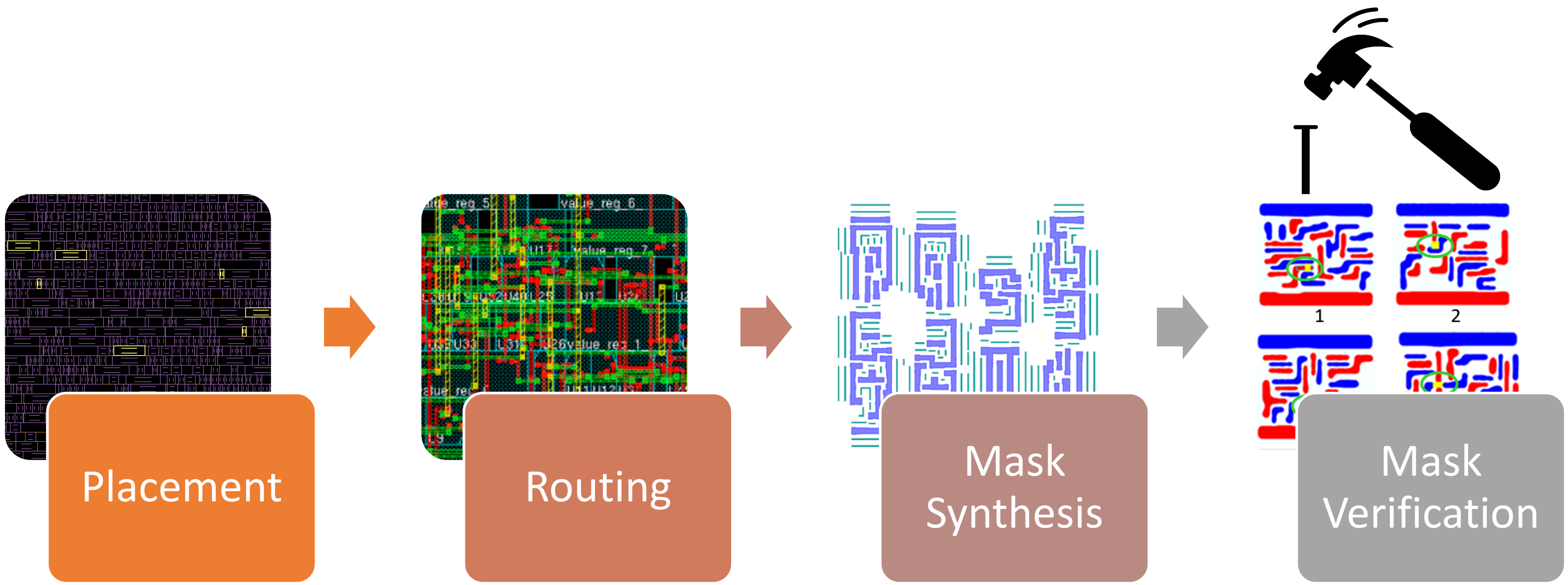
Comparison Summary

	No SRAF	MB	TCAD’17	CGAN
PV Band (um^2)	0.00335	0.002845	0.00301	0.00291
EPE (nm)	3.9287	0.5270	0.5066	0.541
Run time (s)	-	6910	700	48

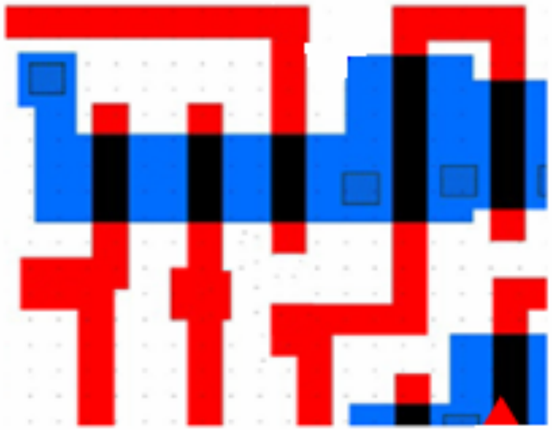
- The proposed CGAN based approach can achieve comparable results with TCAD’17 and MB with **14.6X** and **144X** reduction in runtime

[Alawieh+,DAC’19]

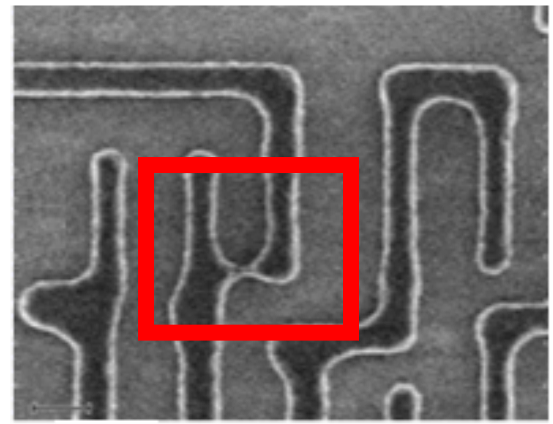
Mask Verification



Bottleneck in Semiconductor and IC Manufacturing: Lithography



Layout

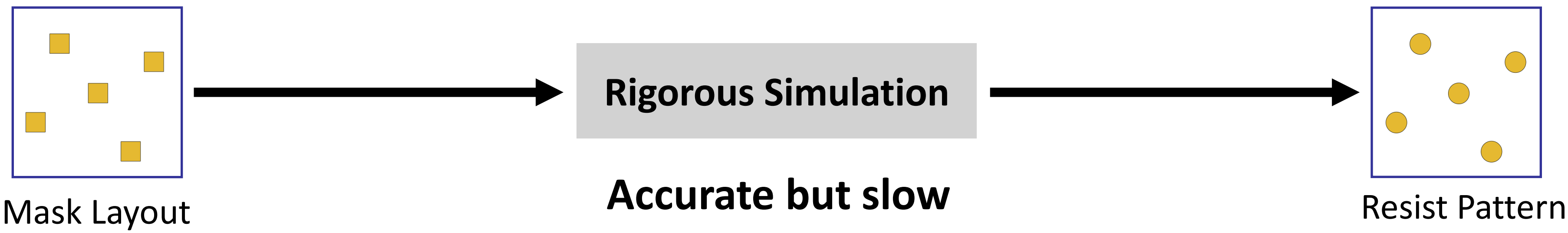


Chip

- Lithographic modeling and hotspot detection
  - What you see (at design) is NOT what you get (at fab)
  - Hotspot ➔ poor performance and yield
- Litho-simulations are extremely CPU intensive
  - Need much faster algorithms to guide IC design, and pinpoint possible mask/wafer inspection spots

Challenges in Lithography Modeling

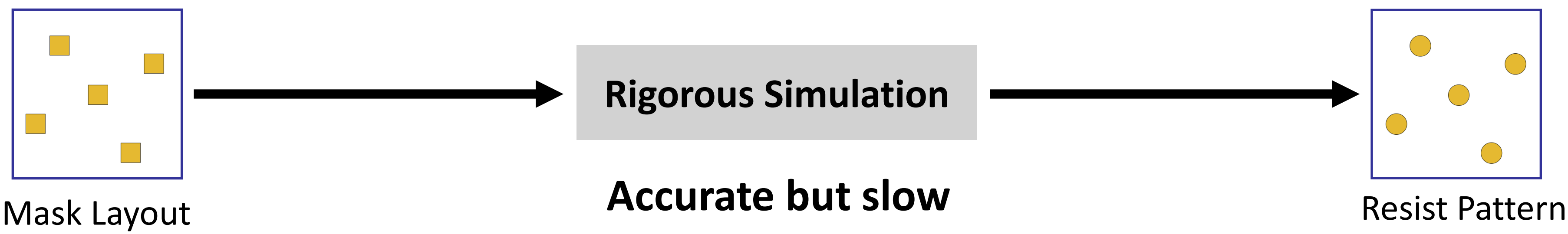
Rigorous simulation: physics-based simulation, e.g., Synopsys S-Litho



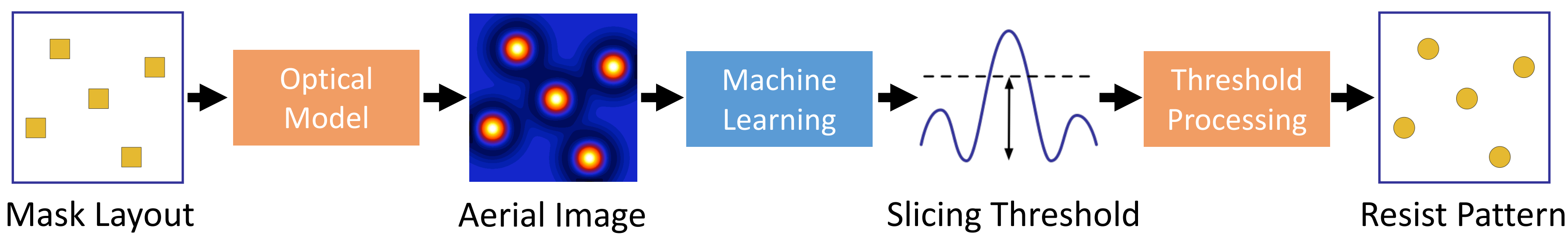
- Simulating 2 µm × 2 µm using Synopsys S-Litho ⟹ ~1 minute
- A 2 mm × 2 mm chip contains 1M such clips ⟹ 1.9 years!
- Intel Ivy Bridge 4C: 160 mm^2

ML for Lithography Modeling

Rigorous simulation: physics-based simulation, e.g., Synopsys S-Litho

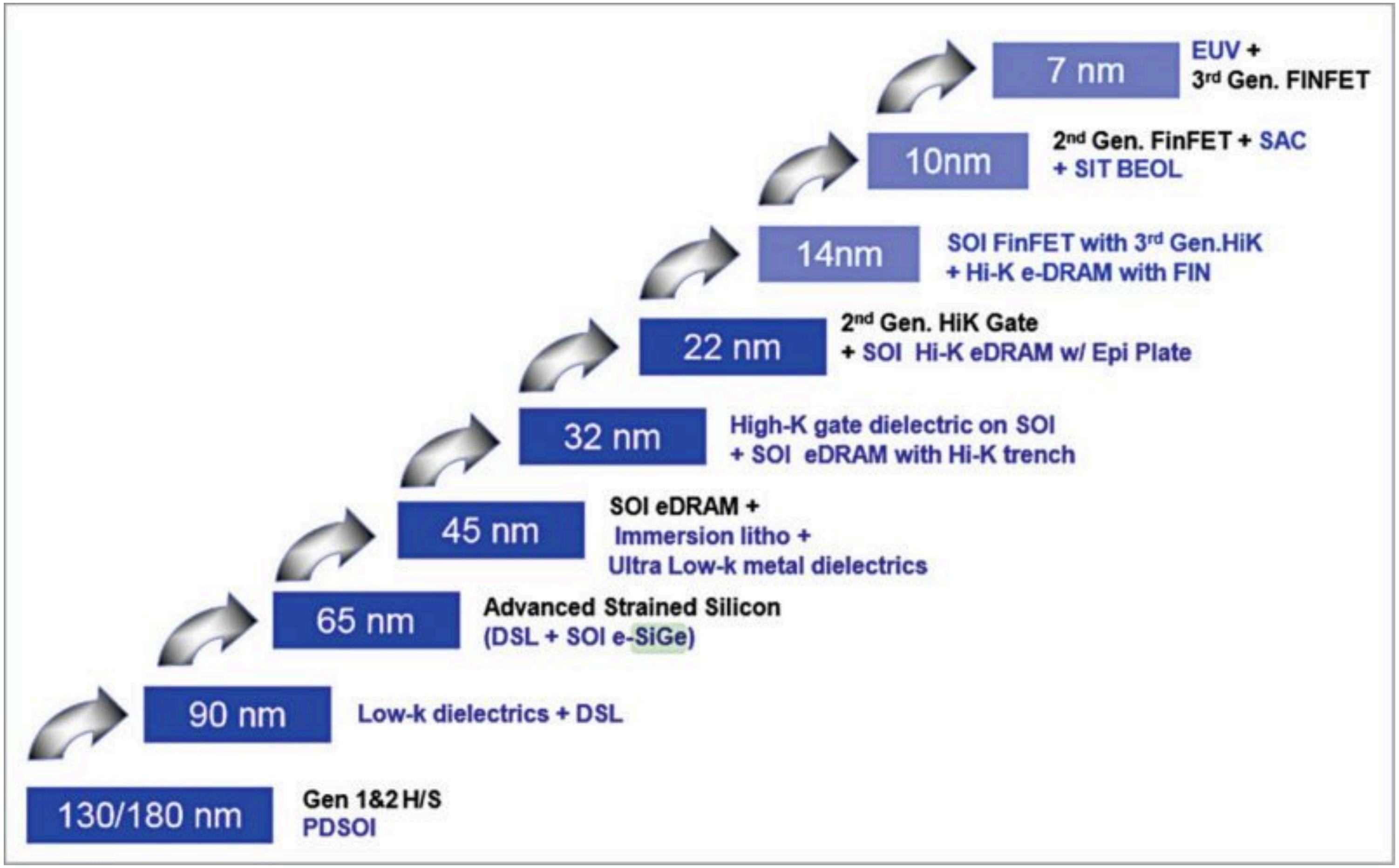


Machine learning for resist modeling [Watanabe+, SPIE’17] [Shim+, SPIE’17]



Speeds up the resist modeling stage

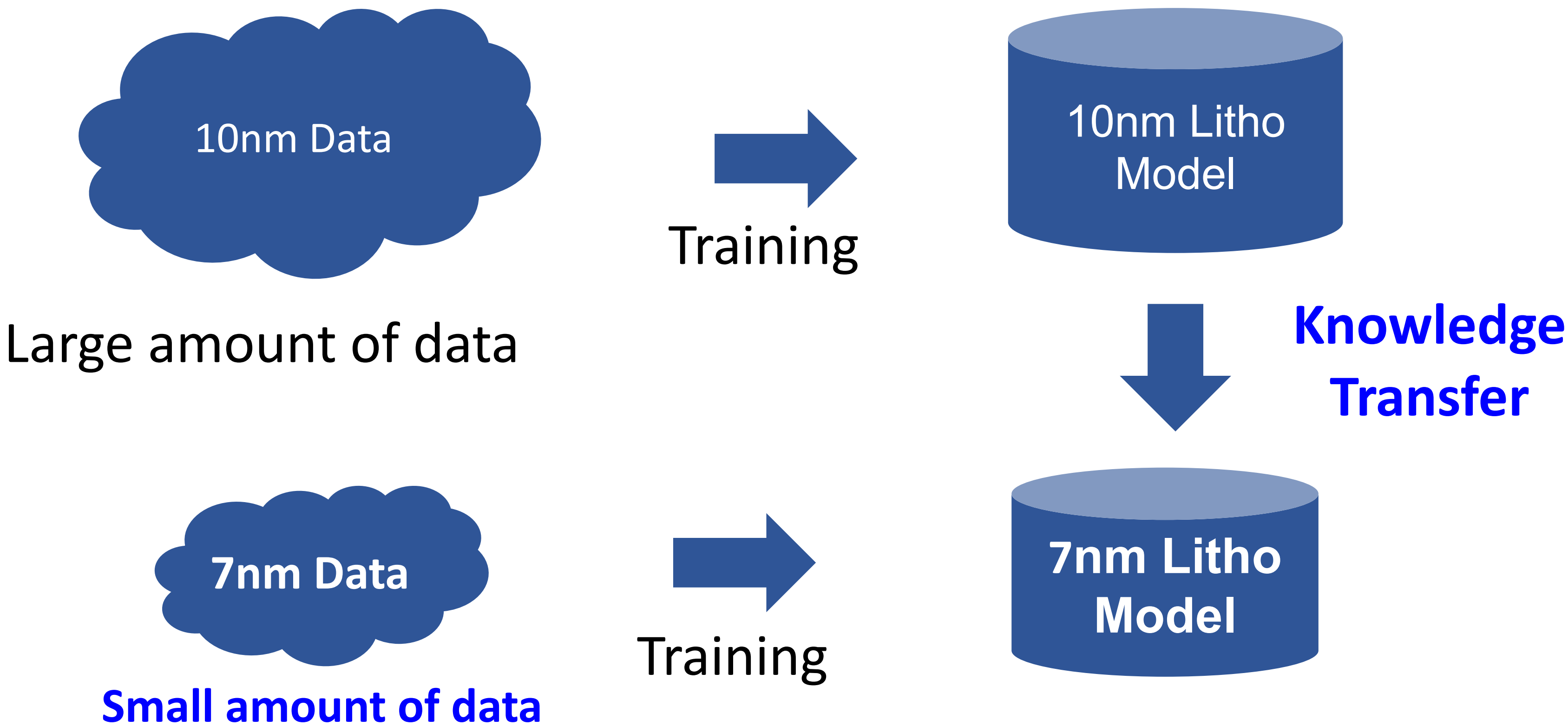
VLSI Technology Nodes



[electronicsforu, 2017]

Transfer Learning for Lithography Modeling

Training with limited new tech. data + older tech.

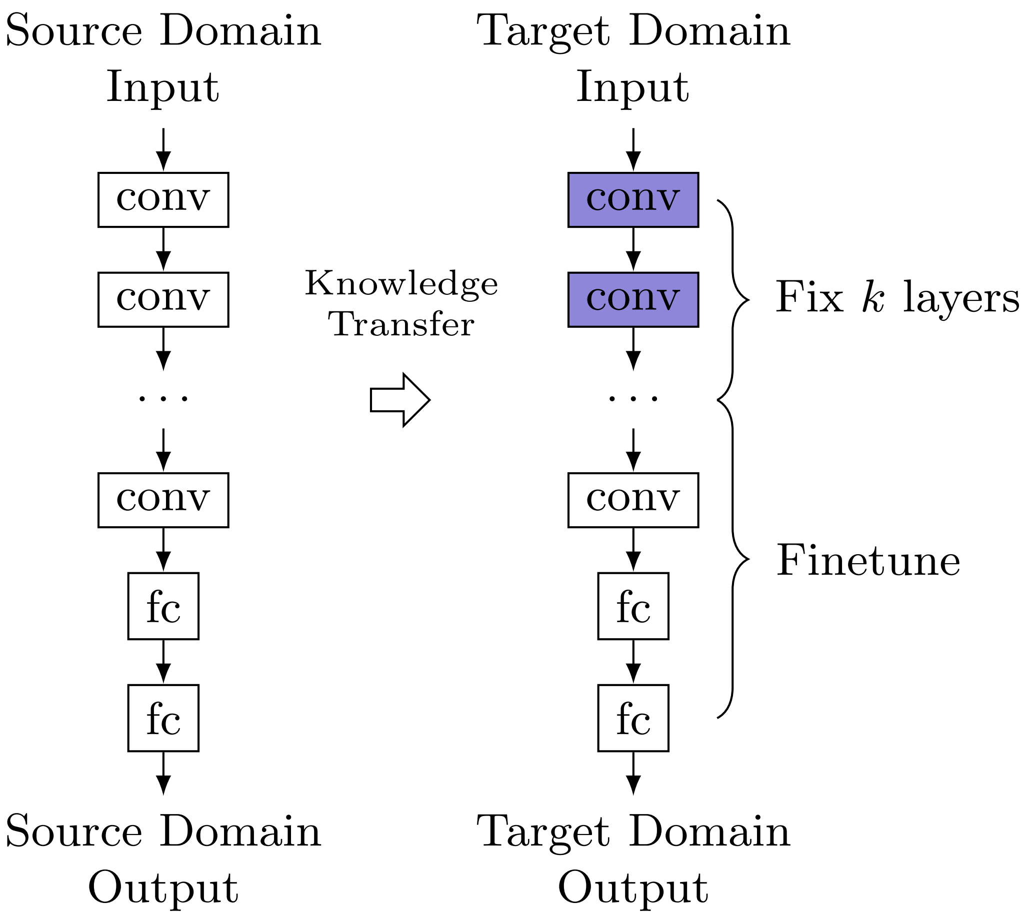


[Lin+, TCAD'19]

Good accuracy for very small data, CNN

Transfer Learning with ResNet

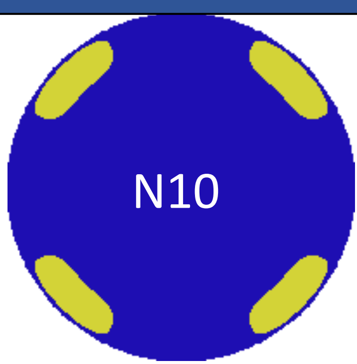
Transfer and Fix TF_k Scheme

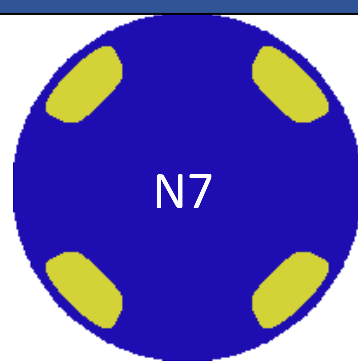


Technology Transition from N10 to N7

Contact Layer Design Rules [Liebmann, SPIE'15]		
	N10	N7
Patterning	LELE	LELELE

	N10	**$N7_a$**	**$N7_b$**
Design Rule	A	B	B
Optical Source	A	B	B
Resist Material	A	A	B





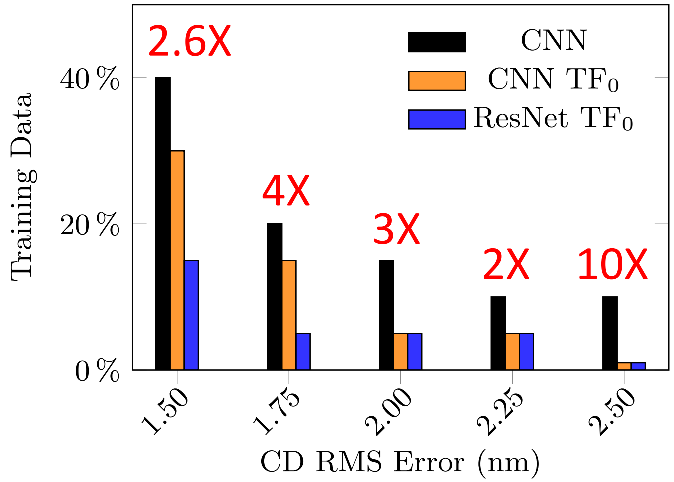
Resist A

Resist B

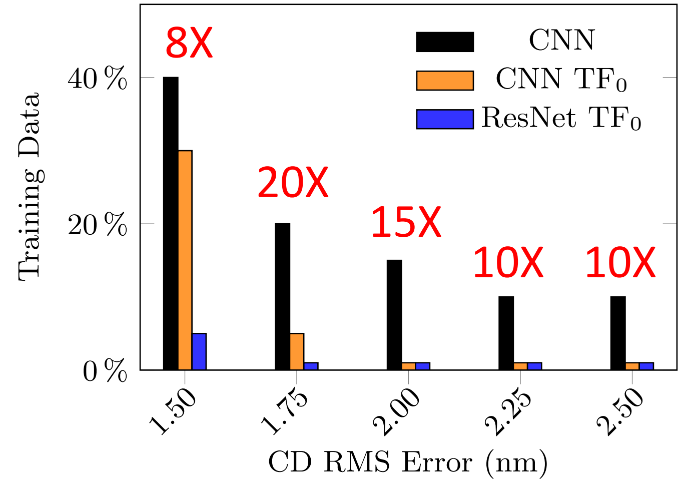
Different dissolution slopes

Data Reduction from Knowledge Transfer

From N10 to $N7_b$	From $N7_a$ to $N7_b$



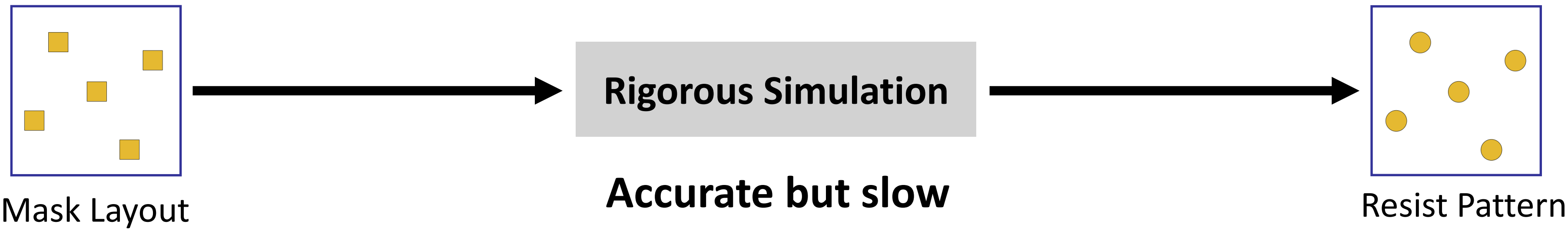
2~10X reduction of training data



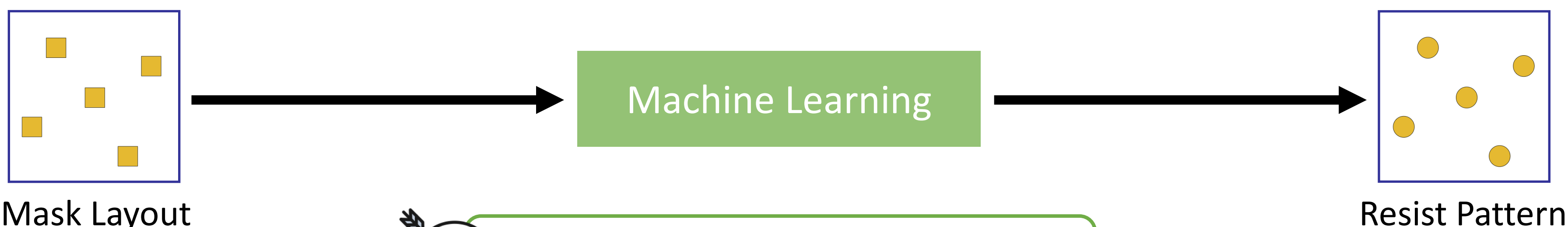
8~20X reduction of training data

End-to-End Lithography Modeling

Rigorous simulation: physics-based simulation, e.g., Synopsys S-Litho



Machine learning for end-to-end lithography modeling

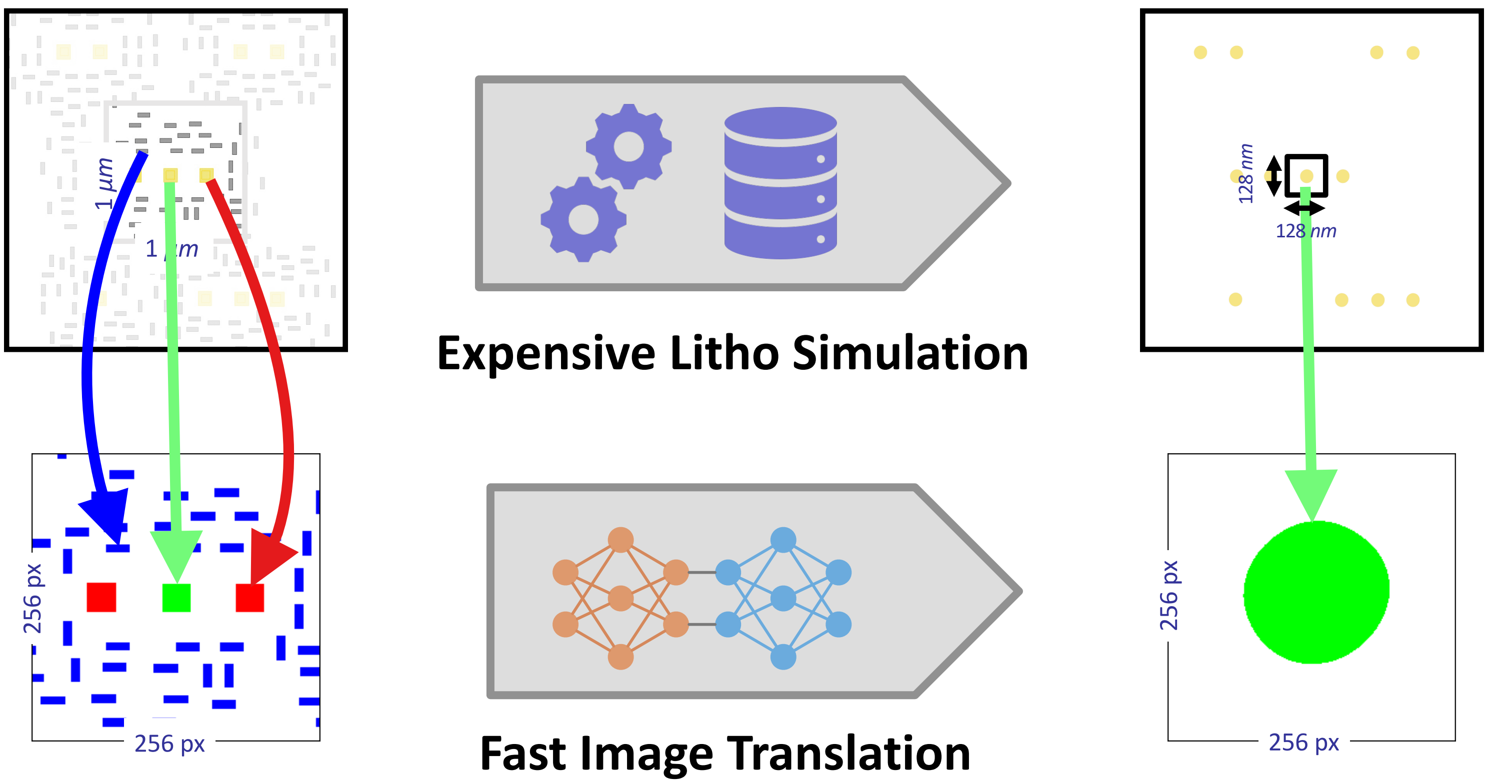


Goal: ultimately fast modeling

[Ye+,DAC'19]

Best paper nomination

Image Translation for Lithography Modeling

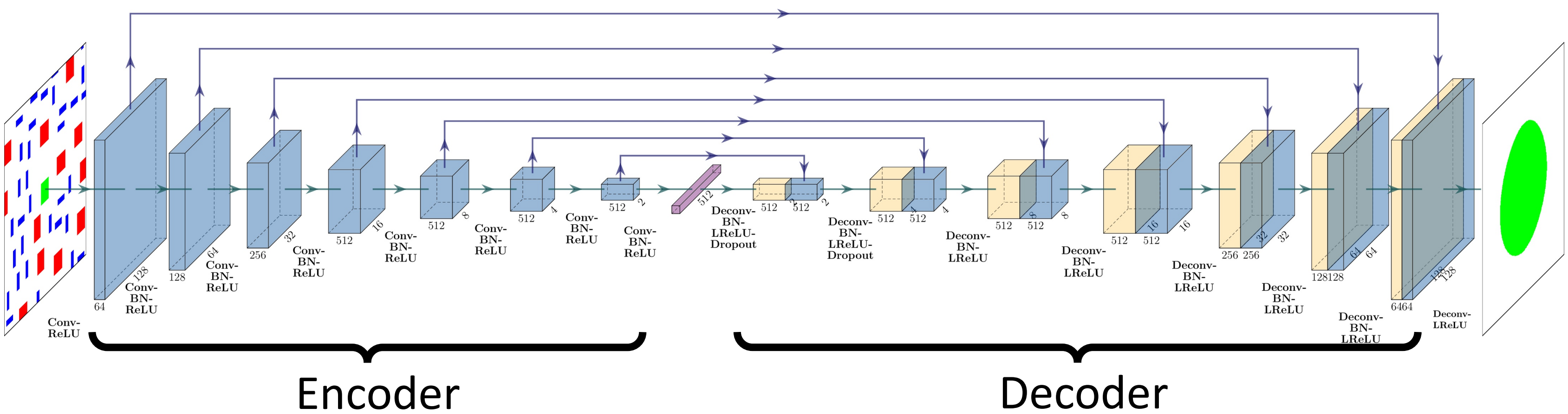


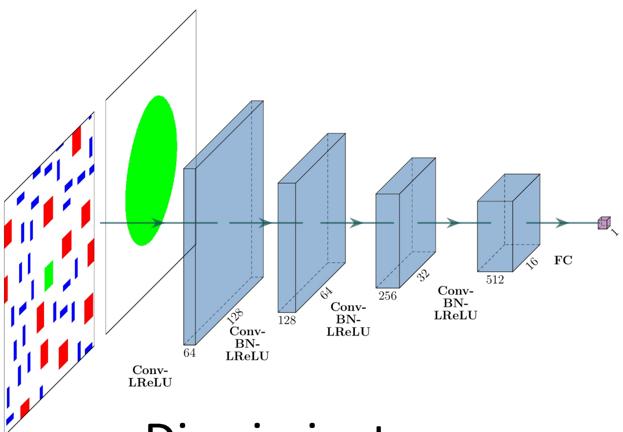
Different elements encoded on different image channels

Resist pattern zoomed in for high-resolution/accuracy

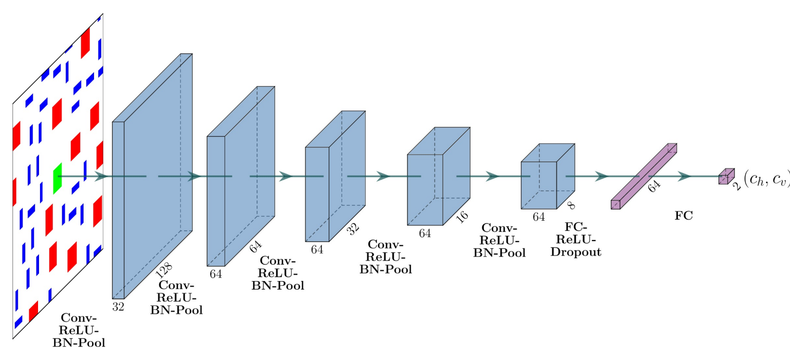
LithoGAN Architecture

Generator



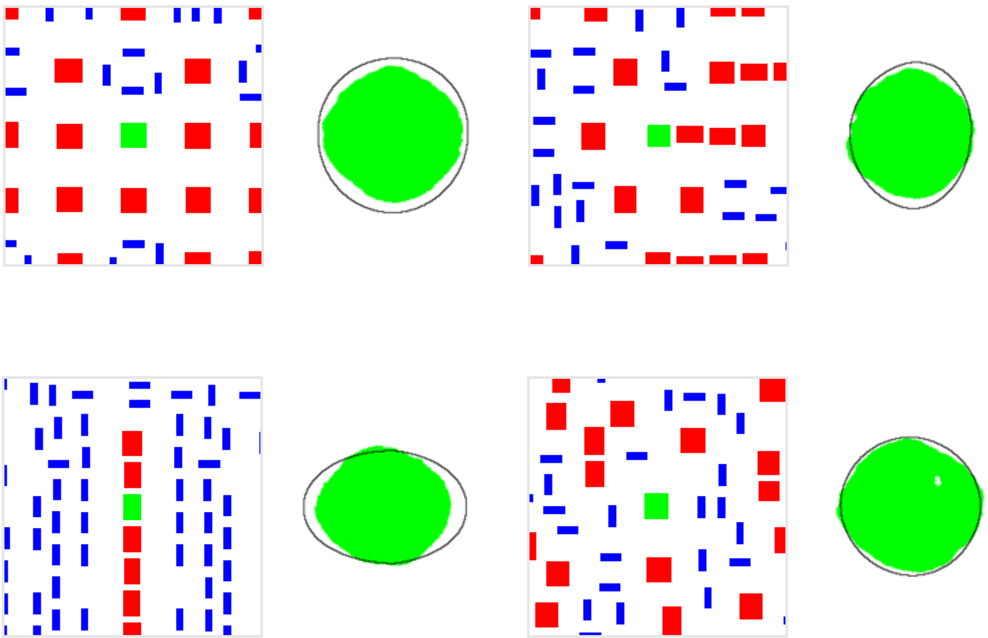


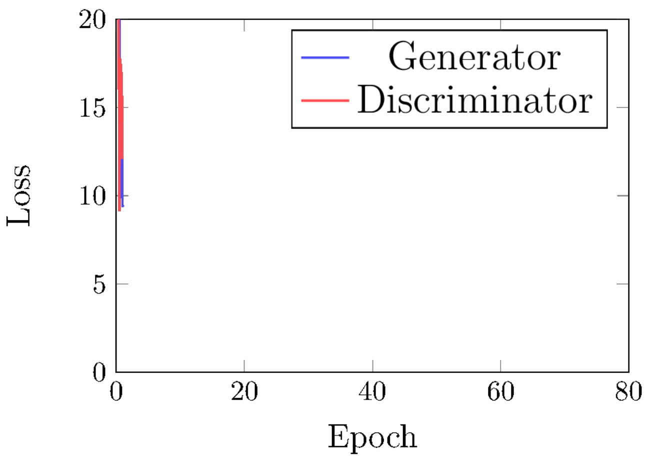
Discriminator



CNN for center prediction

LithoGAN Visualization





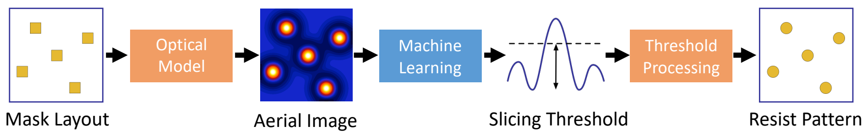
Model advancement progress

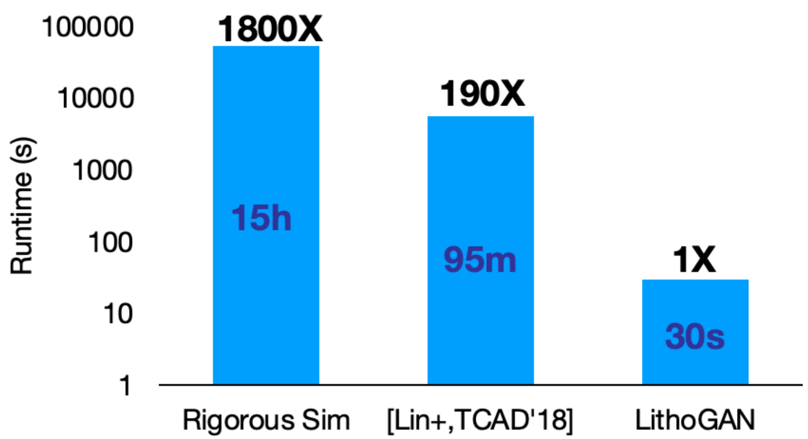
Experimental Results

Setup
- Python w/ TensorFlow
- 3.3GHz Intel i9 CPU & Nvidia TITAN Xp GPU

Datasets
- Different types of contact arrays [Lin+, TCAD'18]
  - 982 mask clips at 10nm node (N10)
  - 979 mask clips at 7nm node (N7)
- 75-25 rule for train/test split

Methods
- Rigorous sim using S-Litho: golden resist patterns
- [Lin+, TCAD'18]: Optical sim using Calibre + threshold prediction using CNN + post processing

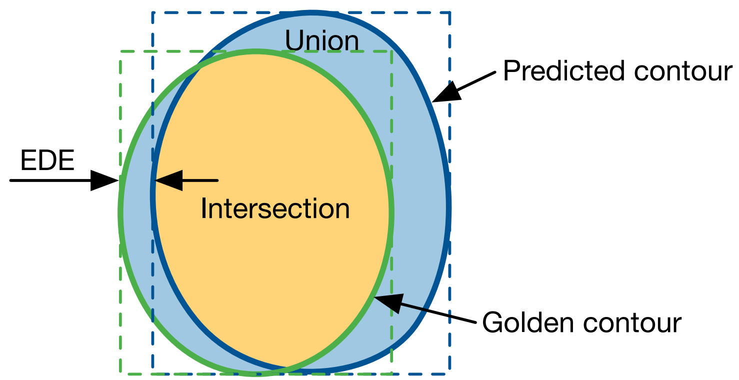




Compelling runtime speedup for early technology exploration

[Ye+,DAC'19]

Experimental Results



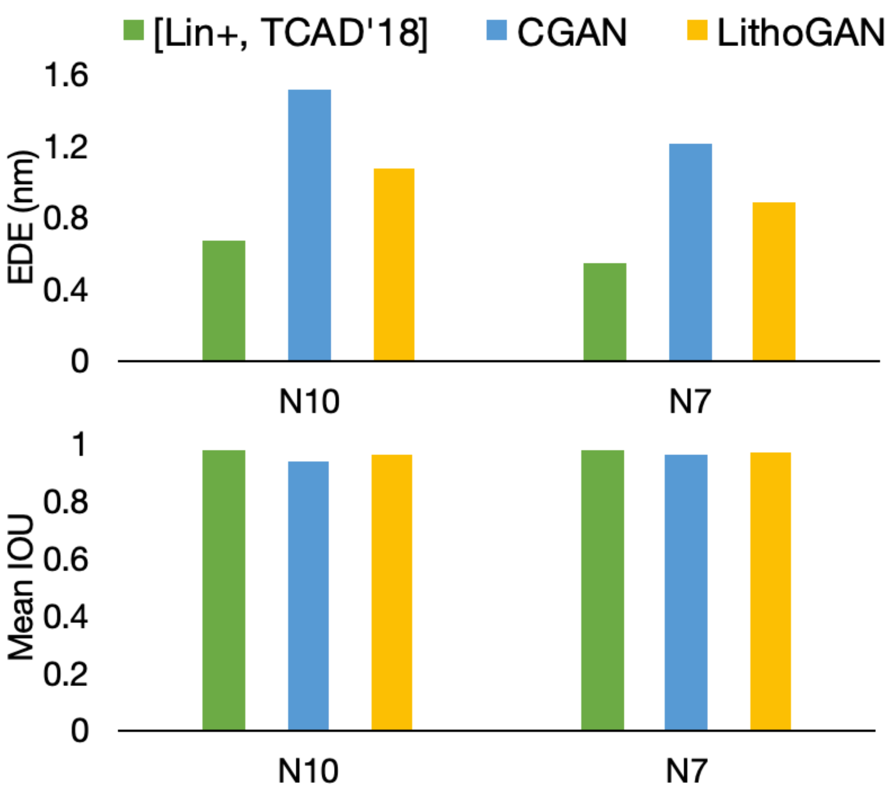
Accuracy measures

<u>Edge Displacement Error (EDE)</u>

- Distance between the golden edge and the predicted one of the bounding boxes
- The **smaller**, the **better**
- Captures bounding box mismatch

<u>IOU = Intersection/Union</u>

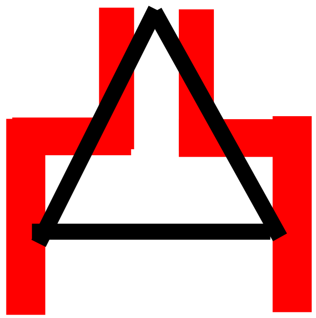
- The **larger**, the **better**
- Captures contour mismatch



Competent accuracy for lithography usage (in consultation with industry)

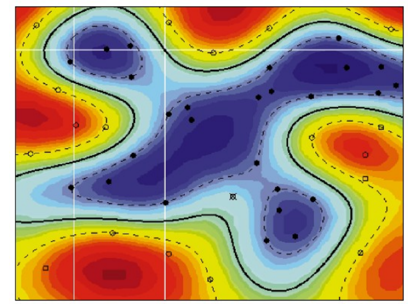
[Ye+,DAC'19]

Lithography Hotspot Detection



Pattern/Graph Matching

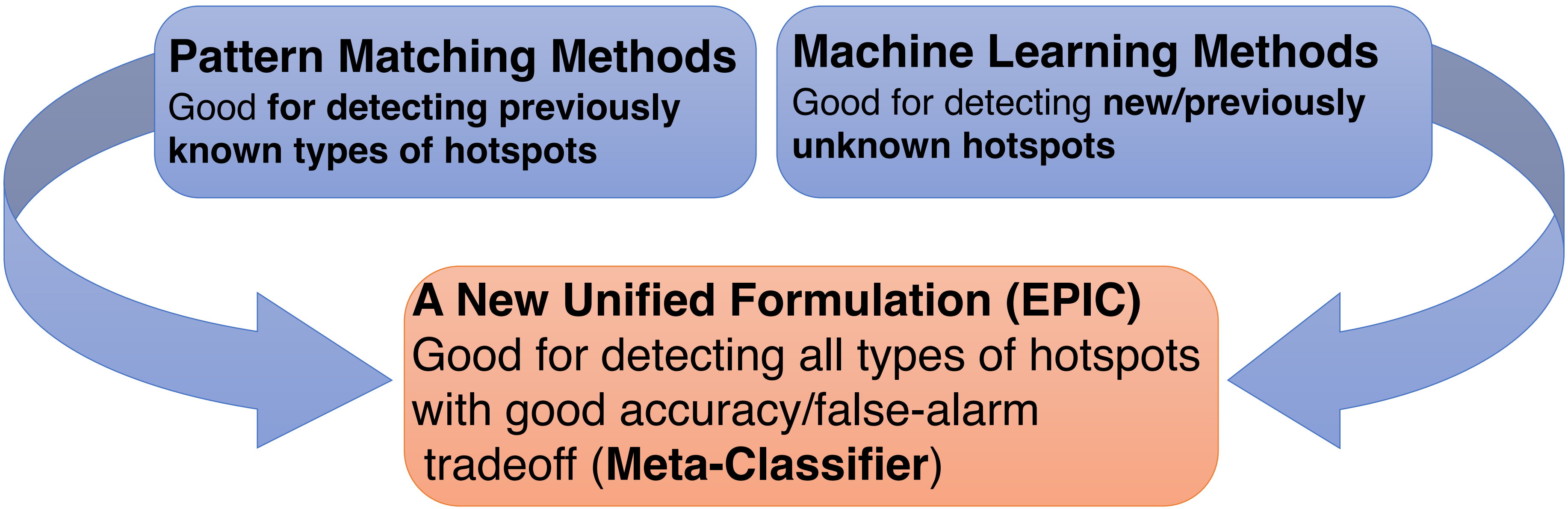
- Pros/cons
  - Accurate and fast for known patterns
  - But too many possible patterns to enumerate
  - Sensitive to changing manufacturing conditions



Data Mining/ML

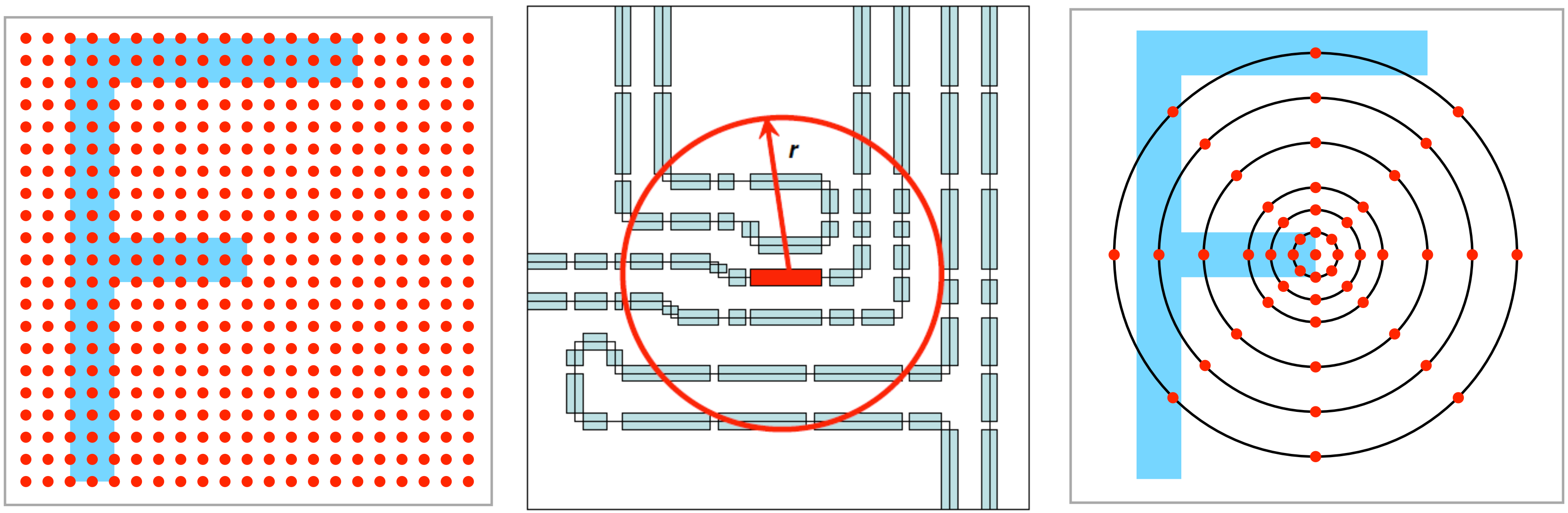
- Pros/cons
  - Good to detect unknown or unseen hotspots
  - Trade-off accuracy and false alarms
  - Accuracy may not be good for “seen” patterns

PM & ML for Hotspot Detection

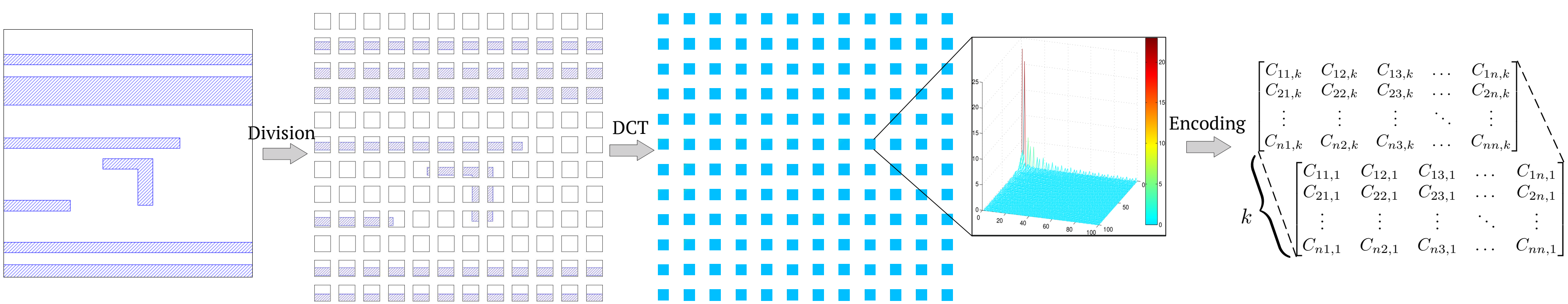


- Meta-classification combines the strength of different hotspot detection techniques [Ding+, ASPDAC'12]
- Balance accuracy and false-alarm

Application-Specific ML



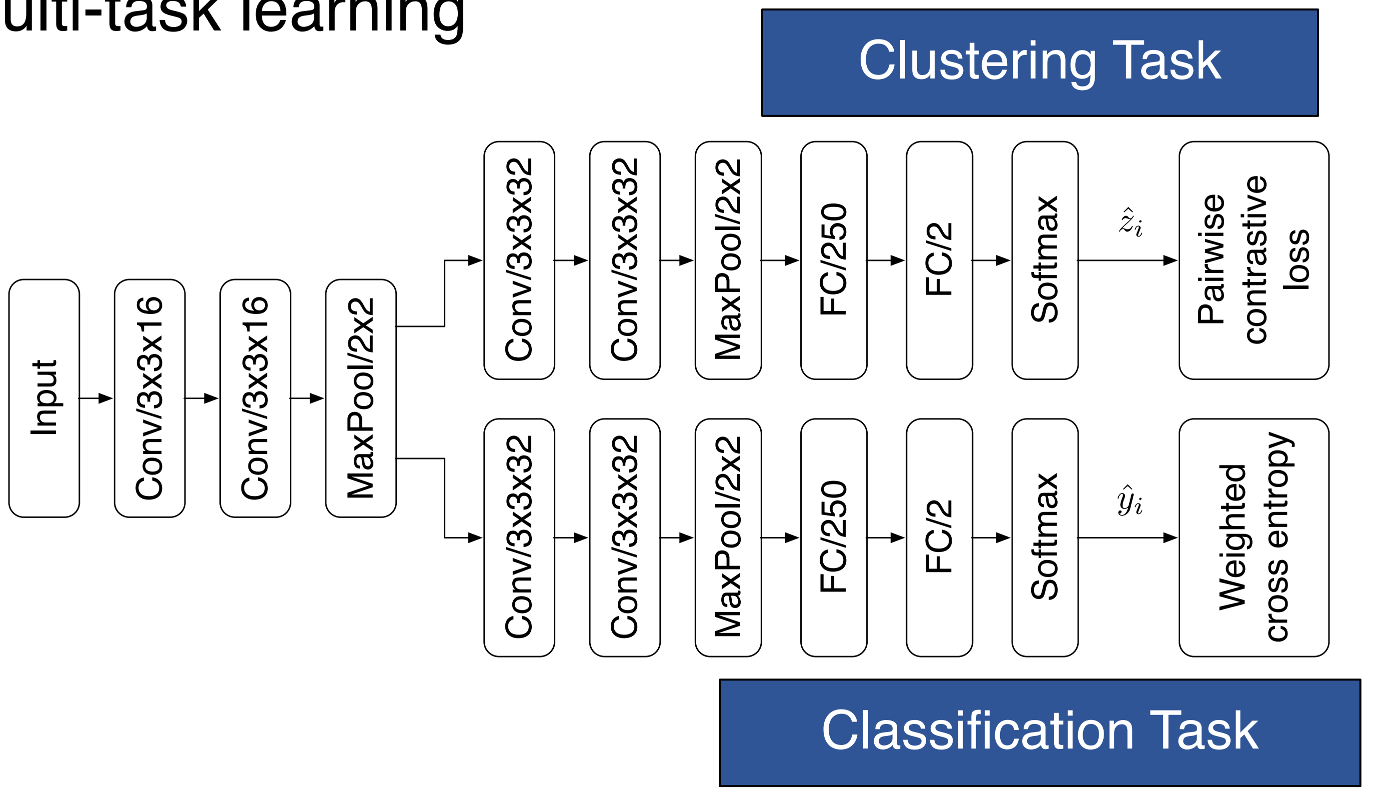
Different ways of feature extractions/representations



Deep learning: CNN, … [Yang+, TCAD'19]

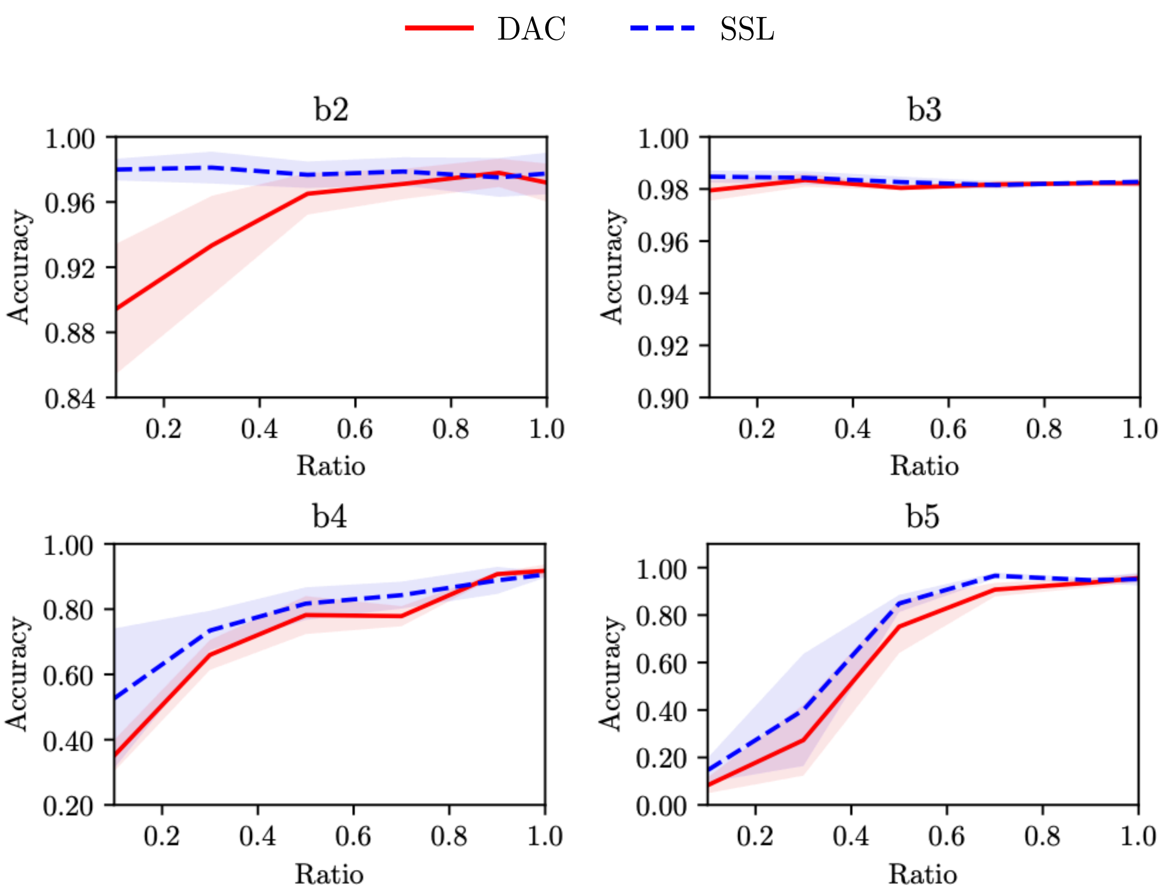
Recent Progress – Lack of Data

- Self-paced semi-supervised learning
  - [Chen+, ASPDAC'19]
  - Leverage unlabeled data
  - Multi-task learning



Preliminary Results [Chen+, ASPDAC'19]

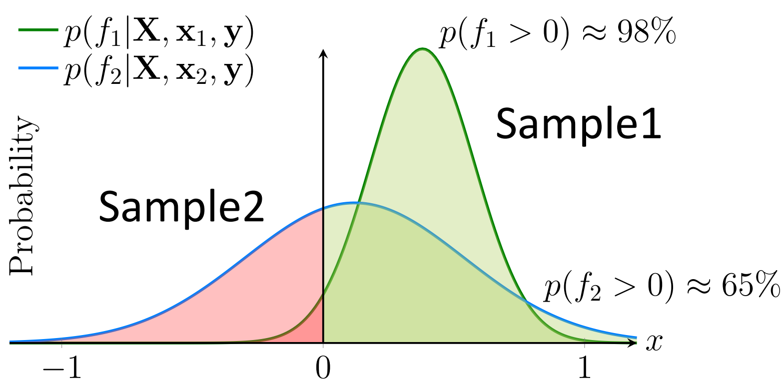
Better accuracy with small amount of training data

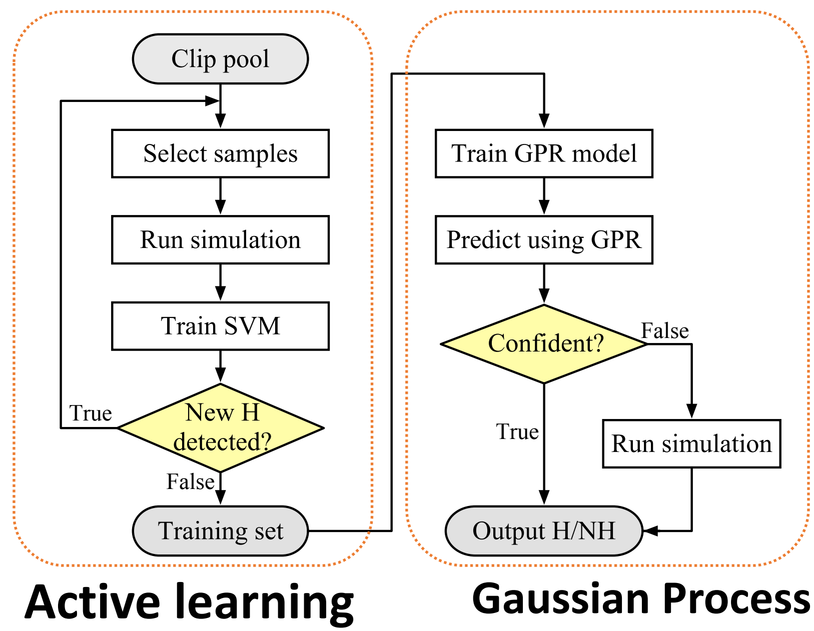


Recent Progress: Model Confidence

Examine confidence of model predictions

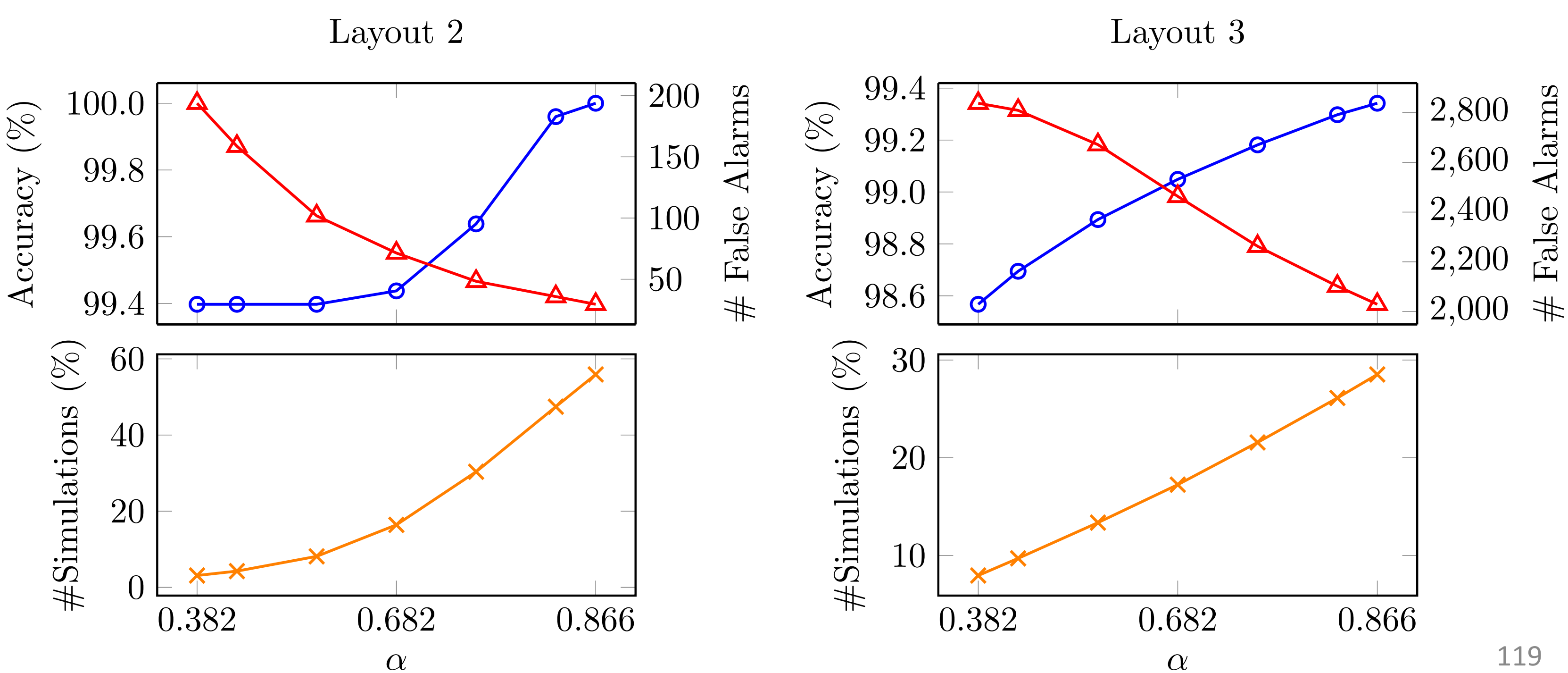
- [Ye+, DATE'19]
- Gaussian Process provides confidence level
- Active learning: build accurate GP model with little data
- Use GP model: simulate unconfident samples



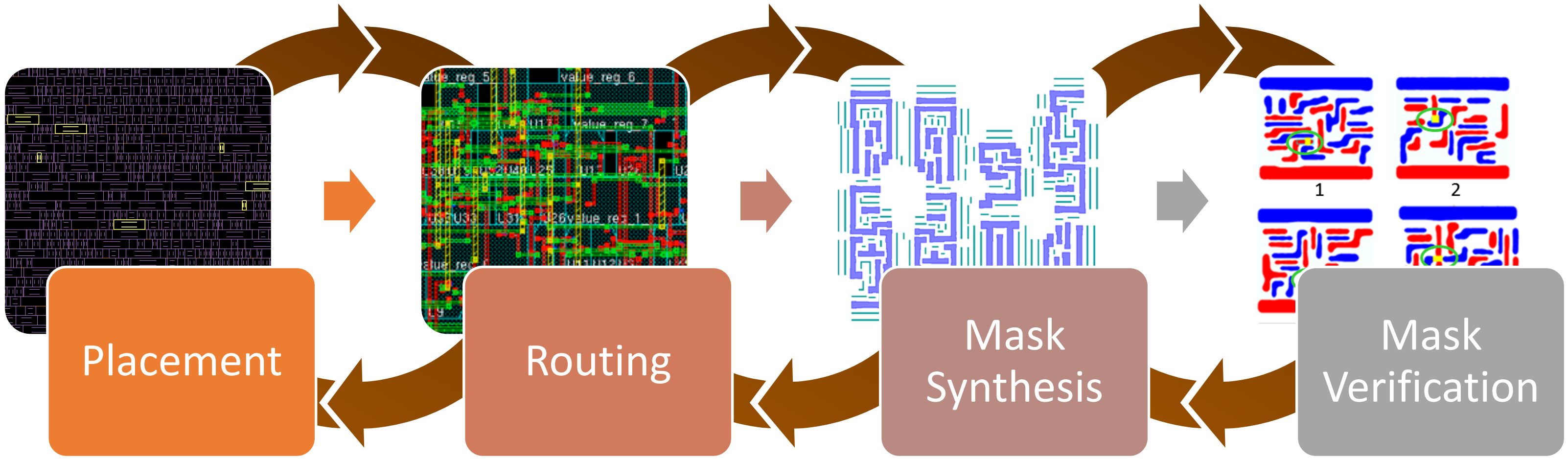


Preliminary Results [Ye+, DATE'19]

- Larger α implies a large confidence interval
  - $[\mu - \alpha\sigma, \mu - \alpha\sigma]$
- Trade-off: quality and #simulations required

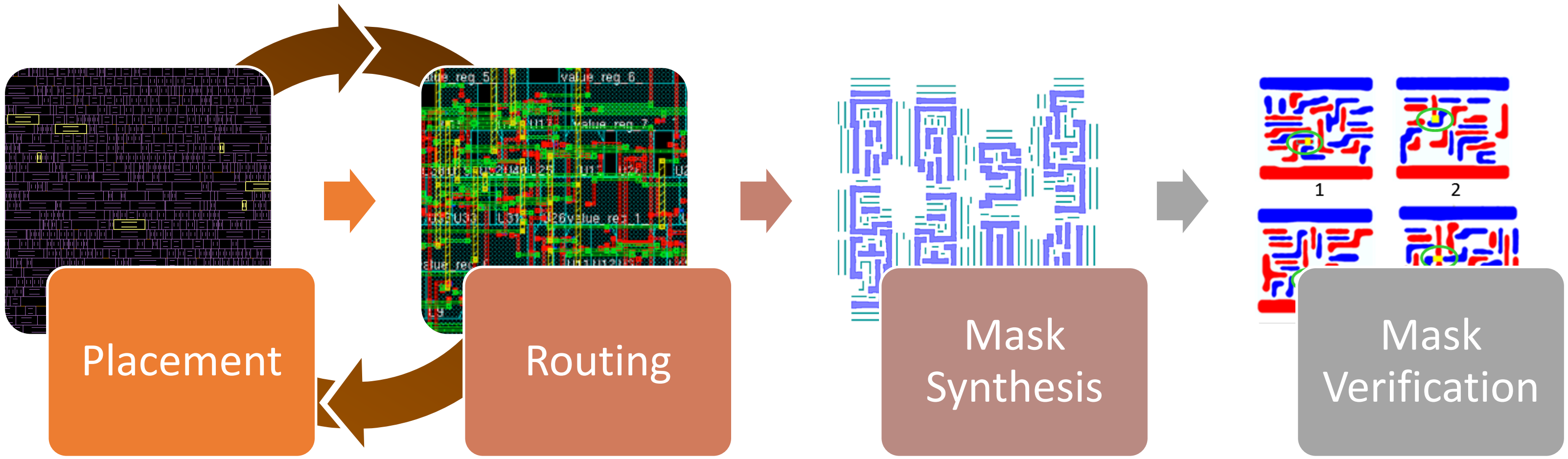


How Machine Learning Can Help



Bridges to connect each step

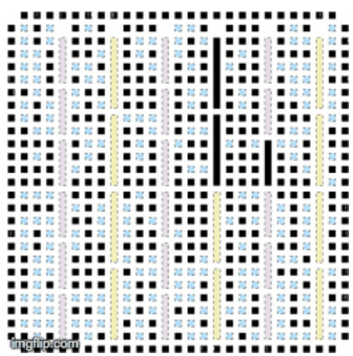
Bridge Placement and Routing



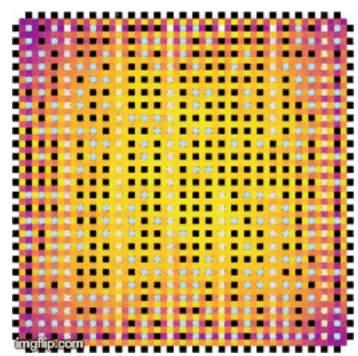
Predicting Routing Congestion

Given placement, predict

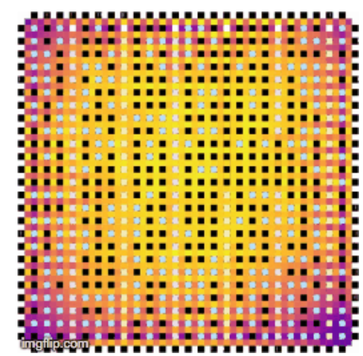
- Routing utilization map [Yu+,DAC'19] for FPGA
- DRC hotspots [Chan+,ISPD'17] [Xie+,ICCAD'18] for ASIC



Input FPGA Placement



Output Routing Util



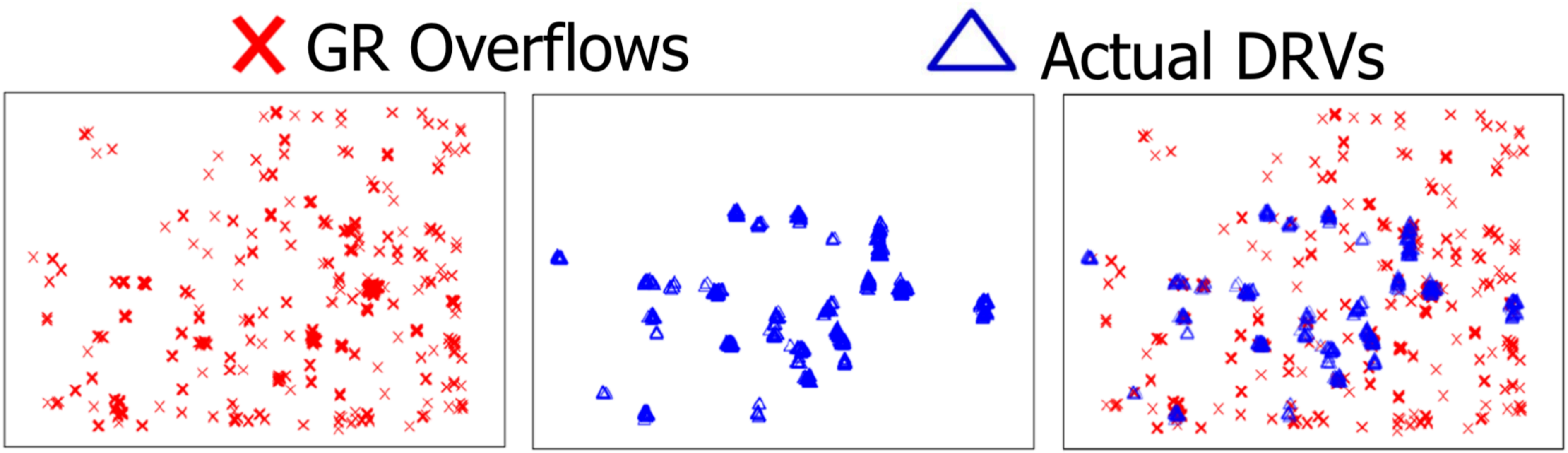
Ground Truth

https://ycunxi.github.io/cunxiyu/dac19_demo.html

Predicting Routing Congestion

Given placement, predict

- Routing utilization map [Yu+,DAC’19] for FPGA
- DRC hotspots [Chan+,ISPD’17] [Xie+,ICCAD’18] for ASIC



[Chan+,ISPD’17]

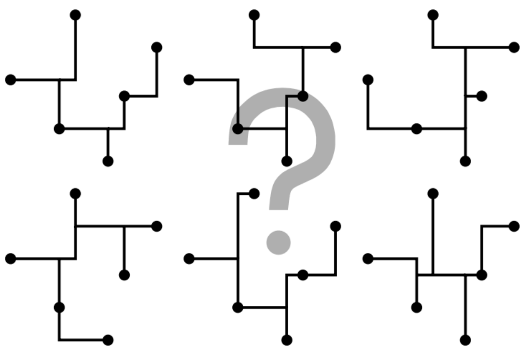
Challenges in Predicting Congestion

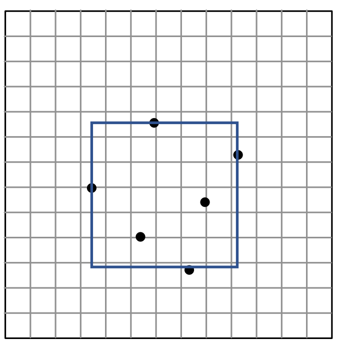
Feature representation

- How to encode interconnect information
- Scalability of features

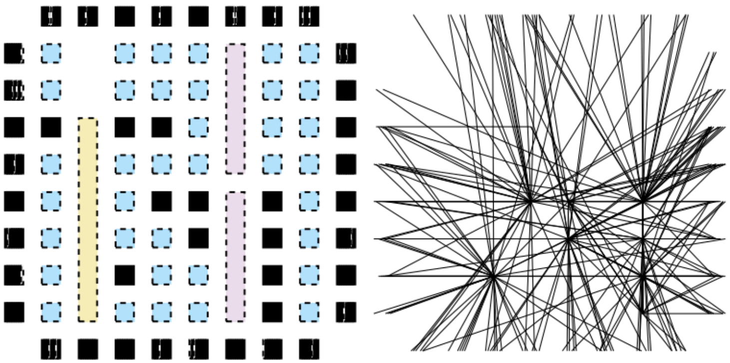
Model selection

- SVM, ANN, FCN, GAN, etc.





Different possibility of routing trees [Spindler+,DATE'07]
Evenly distribute routing demands (RUDY map)



Connectivity images
[Yu+,DAC'19]

Integrate Models into Placement

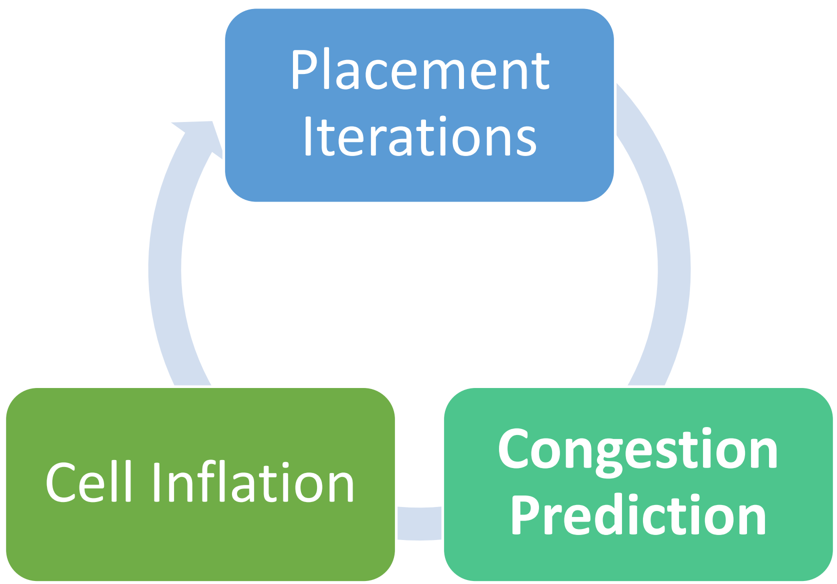
- Up to 76.8% DRC reduction
- Minor WL and timing impacts

[Chan+,ISPD'17]

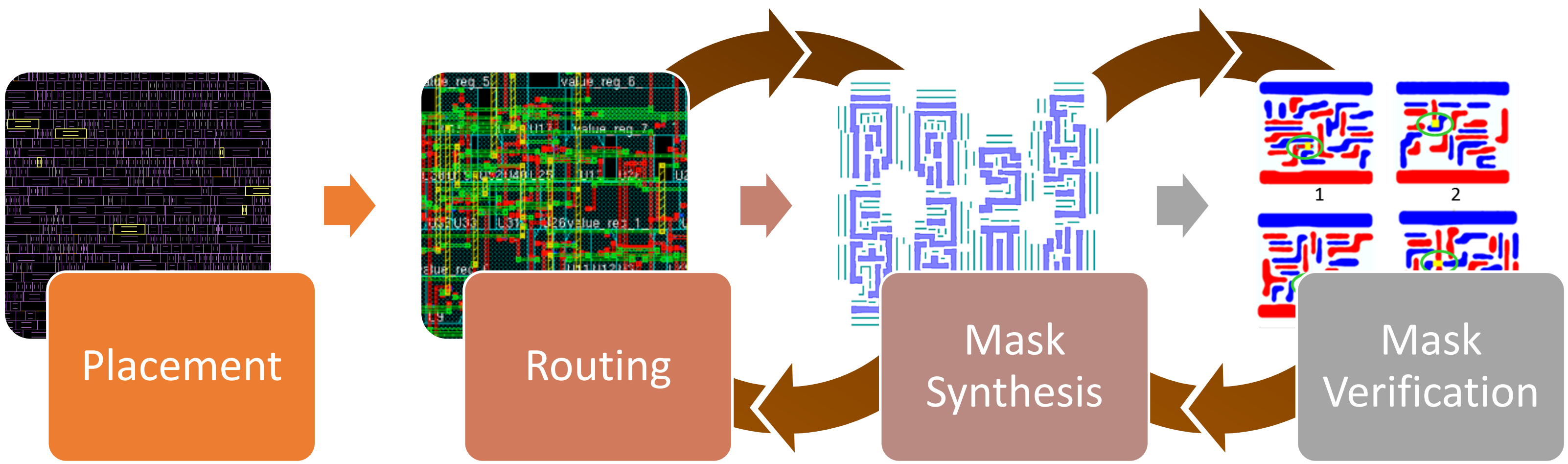
	#DRCs			Wirelength			TNS (ns)		
eg1	8478	1964	-76.83%	1742804	1747685	0.3%	-153.43	-158.4	3.2%
eg2	1502	927	-38.28%	1750698	1753047	0.1%	-168.23	-163.5	-2.8%
eg3	2017	1819	-9.82%	1772889	1773701	0.0%	-215.75	-213.6	-1.0%
eg4	2026	1780	-12.14%	1735185	1735227	0.0%	-151.36	-149.6	-1.2%
eg5	4252	4255	0.07%	1831492	1836060	0.2%	-264.34	-275.6	4.3%
eg6	3440	3891	13.11%	1790059	1794184	0.2%	-195.65	-203.5	4.0%
		avg	-20.6%		avg	0.2%		avg	1.1%
		max	13.1%		max	**0.3%**		max	**4.3%**
		min	**-76.8%**		min	0.0%		min	-2.8%

Integrate Models into Placement

- Recent study from UTDA
- On 2016 ISPD FPGA Placement contest benchmarks
- Up to **7%** reduction in routed wirelength

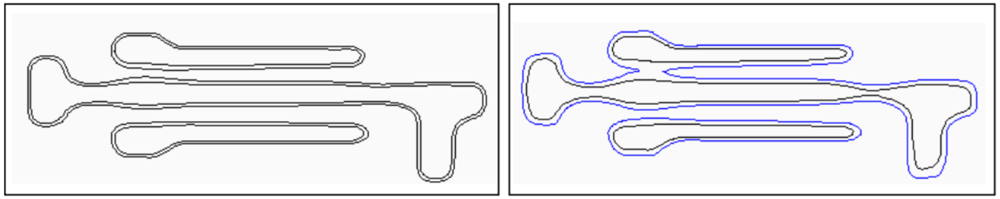


Bridge Design and Manufacturing

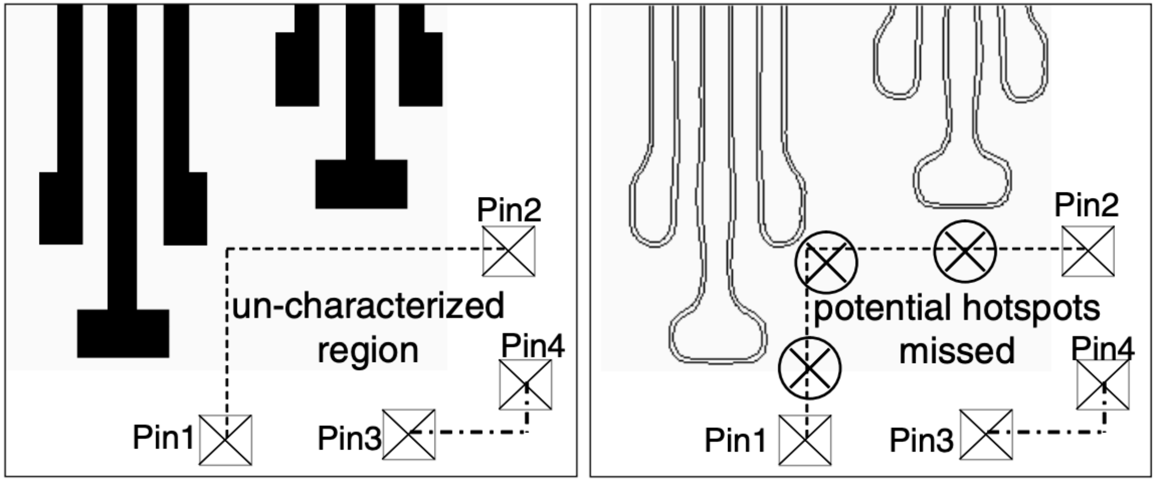


Lithography-Friendly Detailed Router: AENEID

- Motivation: avoid RET dependent layout printability



- Challenges: the lithography hotspot detection dilemma in the detailed routing stage



[Ding+,DAC'11]

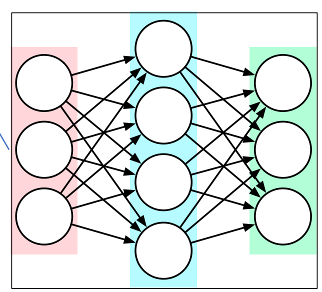
Lithography-Friendly Detailed Router: AENEID

- Objective: minimize total wirelength
- Subject to: keep lithography cost on each routing grid within a threshold

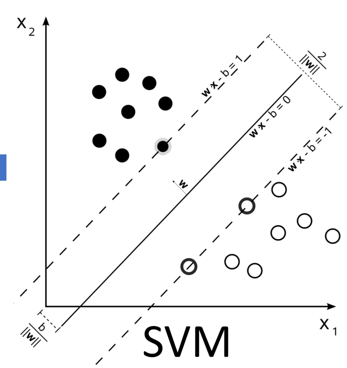
$$\min_{P} \sum_{e \in P} 1, \qquad \text{s.t.} \quad \underline{litho(e)} \le L, \quad \forall e \in P$$

Lagrangian relaxation

$$\max_{\lambda} \min_{P} \sum_{e \in P} 1 + \lambda_e(\underline{litho(e)} - L), \qquad \text{s.t.} \quad \lambda_e \ge 0$$



ANN



SVM

50% reduction in hotspots

[Ding+,DAC'11]

Conclusion

- Machine learning brings
  - New modeling opportunities
  - New optimization techniques
  - New hardware acceleration, e.g., GPU acceleration
  - New software platforms, e.g., Tensorflow, PyTorch
- Hammers and bridges for conventional EDA flow
  - Reformulate the problems, e.g., neural network training, image-to-image translation
  - Accurate and efficient information feedback from late stages to early stages

Open Problems

- Connectivity feature representation
  - How to encode hypergraph as input to models
  - Or develop models that are friendly to ultra-large graphs
- Optimization-friendly ML models and ML-friendly optimization techniques
  - Target at integrating ML models into optimization
  - Need to consider fidelity, smoothness, accuracy, convergence rate
- Generalization guarantee
  - How far can ML models generalize
  - How to know whether a model is applicable to new data

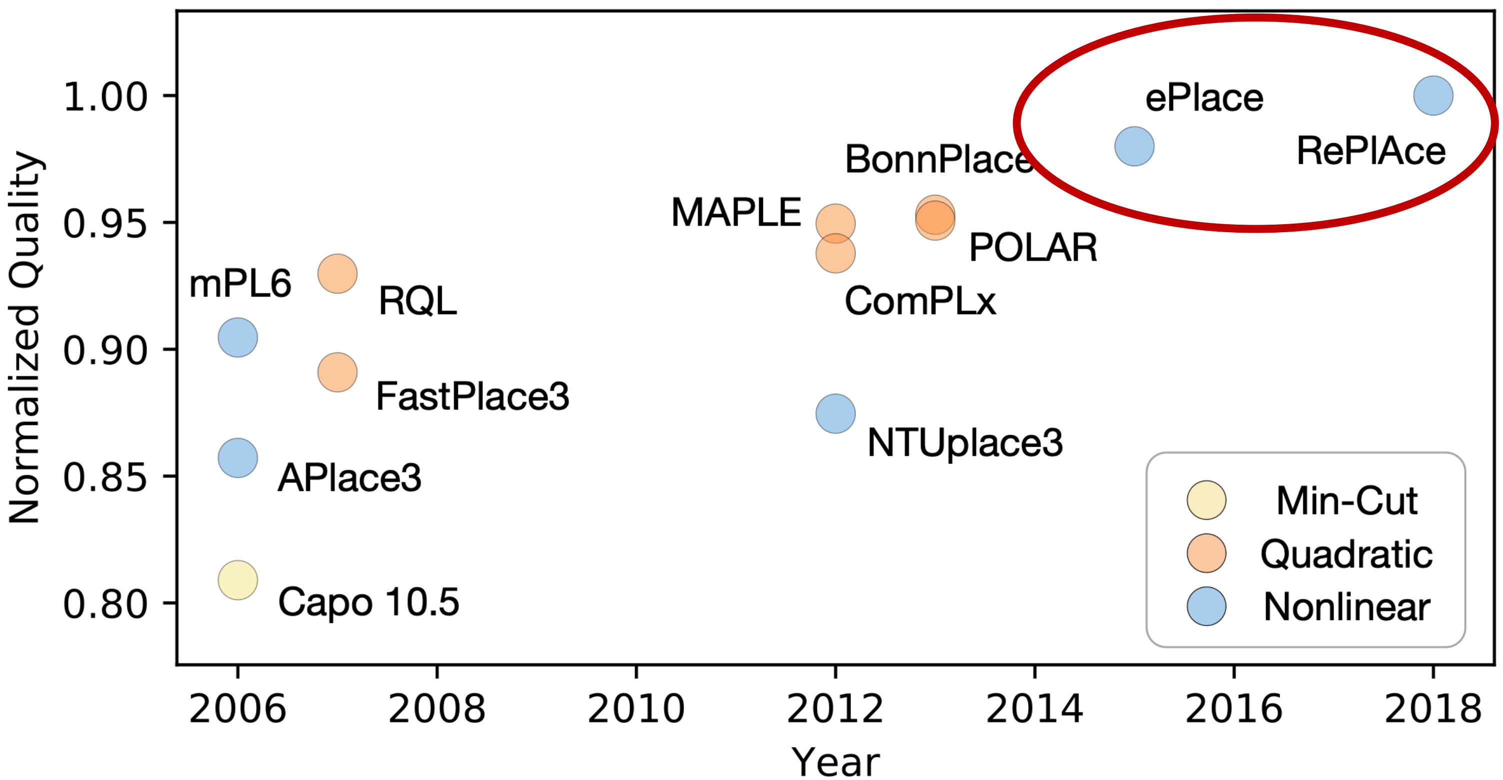
Future Directions

- Routability & Timing-driven DREAMPlace with deep learning toolkits and GPU acceleration
  - Integrate ML models for routability and timing prediction
- Reinforcement learning for routing strategies
  - Rip-up and re-route policy
  - Tree topology generation
- End-to-end mask synthesis and verification
  - Use LithoGAN to guide SRAF and OPC
  - No need to generate golden SRAF and OPC solutions
- Open discussions…

Q&A

Thanks

Recent Development of Placement



*Data collected from RePlAce [TCAD'18, Cheng+] and http://vlsi-cuda.ucsd.edu/~ljw/ePlace/ on ISPD 2005 benchmarks

Future Directions

New Solvers

SGD, ADAM, etc.

New Objectives

Routability, timing, etc.

DREAM

BIGGER

Gate sizing,
floorplanning,
…

Applicable to Other CAD Problems

Multi-GPU,
distributed computing,
mixed precision,
…

New Accelerations