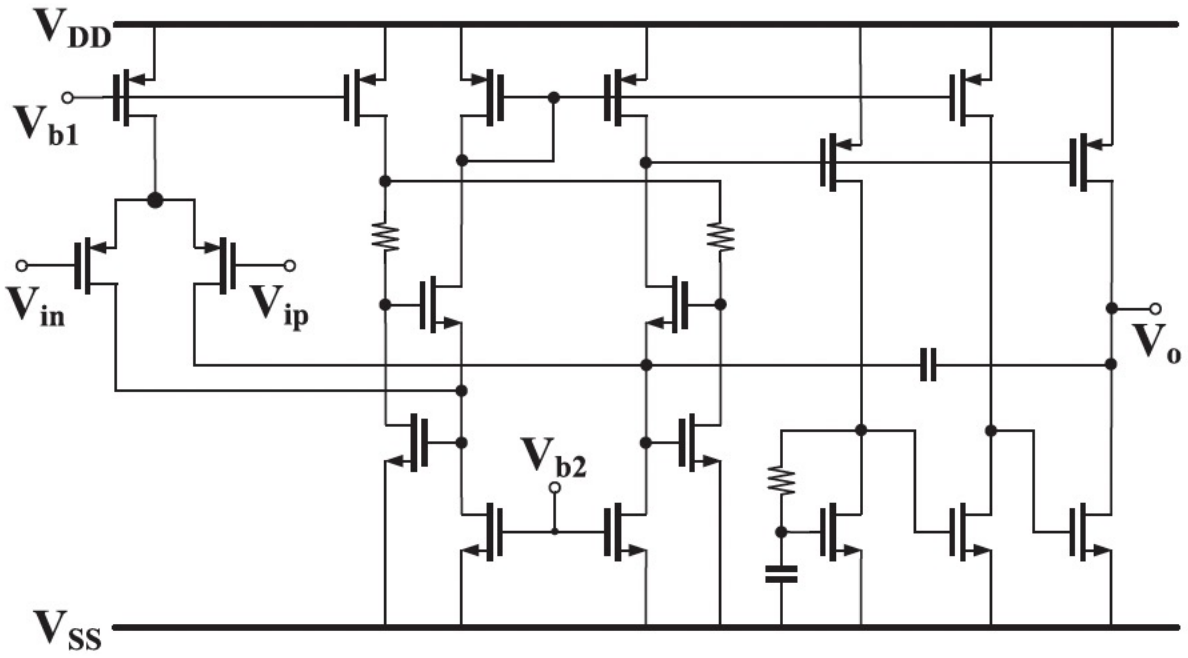
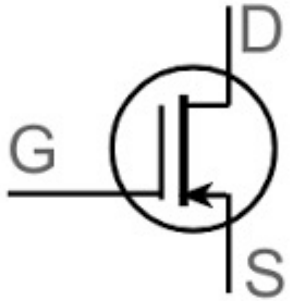
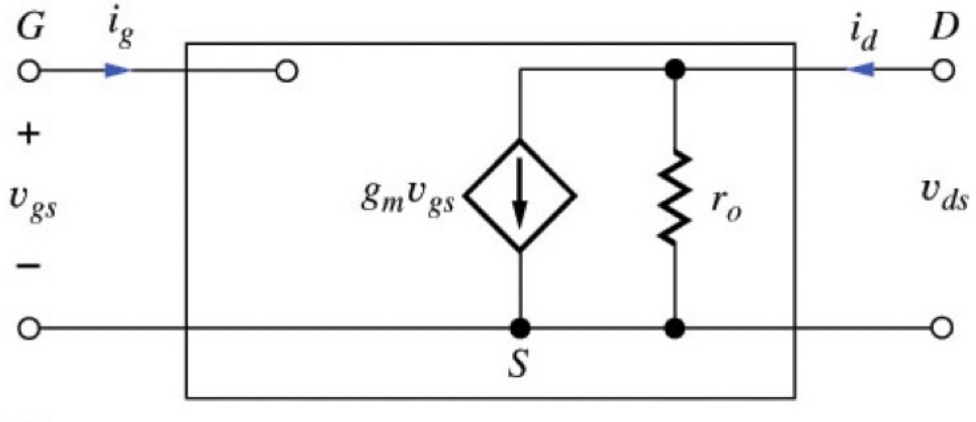
Analog design



Single transistor small-signal model





$$I_G = 0$$

$$I_D = \frac{K_n}{2}\left(V_{GS} - V_{TN}\right)^2\left(1 + \lambda V_{DS}\right)$$

Transconductance:

$$g_m = \frac{2I_D}{V_{GS} - V_{TN}} = \sqrt{2K_n I_D}$$

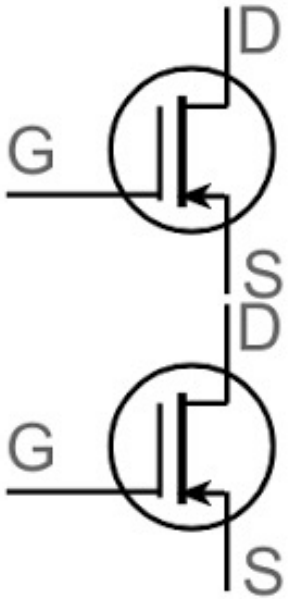
Output resistance:

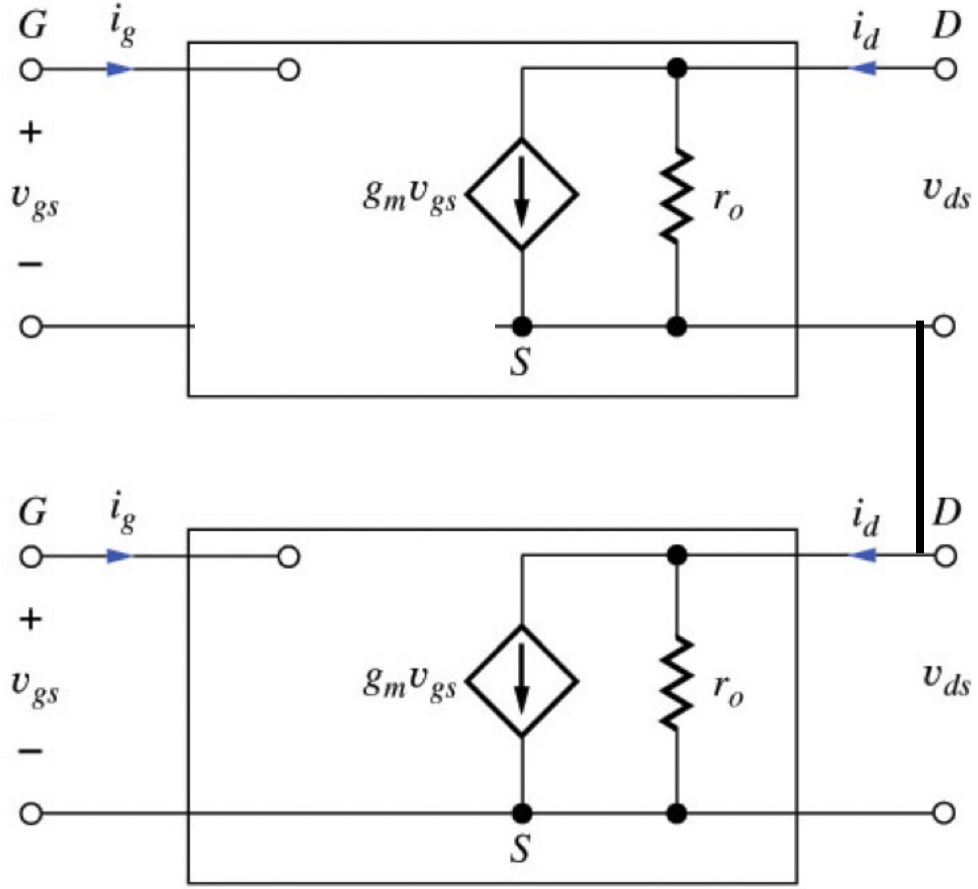
$$r_o = \frac{1}{g_o} = \frac{1 + \lambda V_{DS}}{\lambda I_D} \cong \frac{1}{\lambda I_D}$$

Amplification factor for $\lambda V_{DS} \ll 1$:

$$\mu_f = g_m r_o = \frac{1 + \lambda V_{DS}}{\lambda I_D} \cong \frac{1}{\lambda}\sqrt{\frac{2K_n}{I_D}}$$

Two transistors small-signal model





What about large circuits?

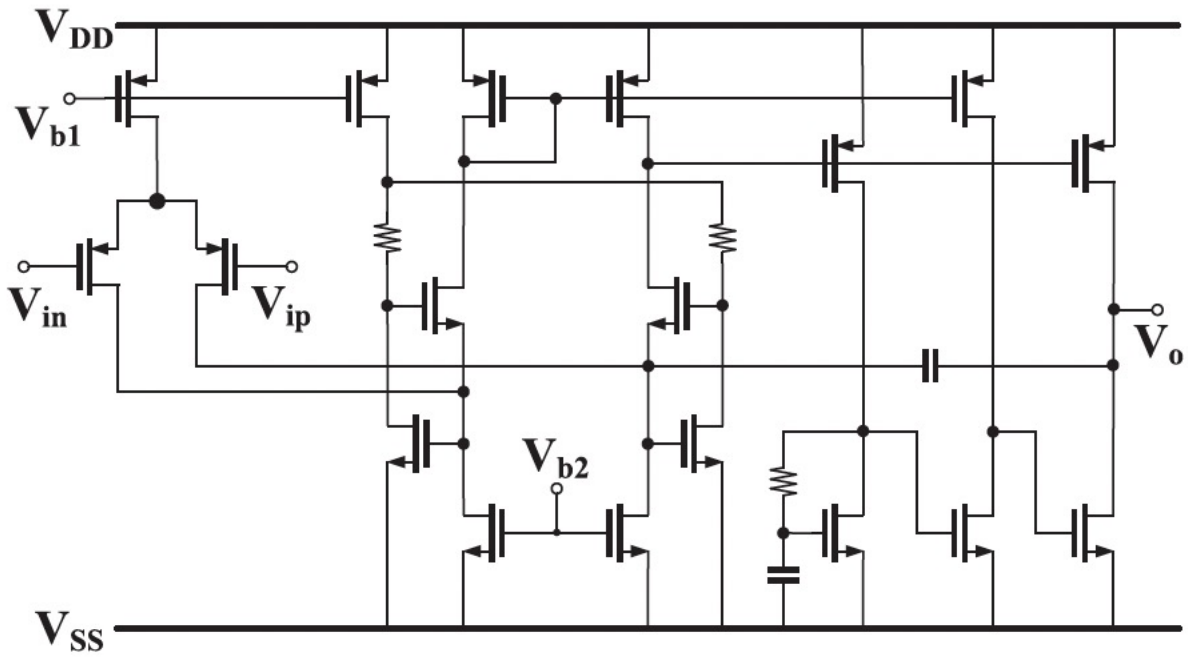
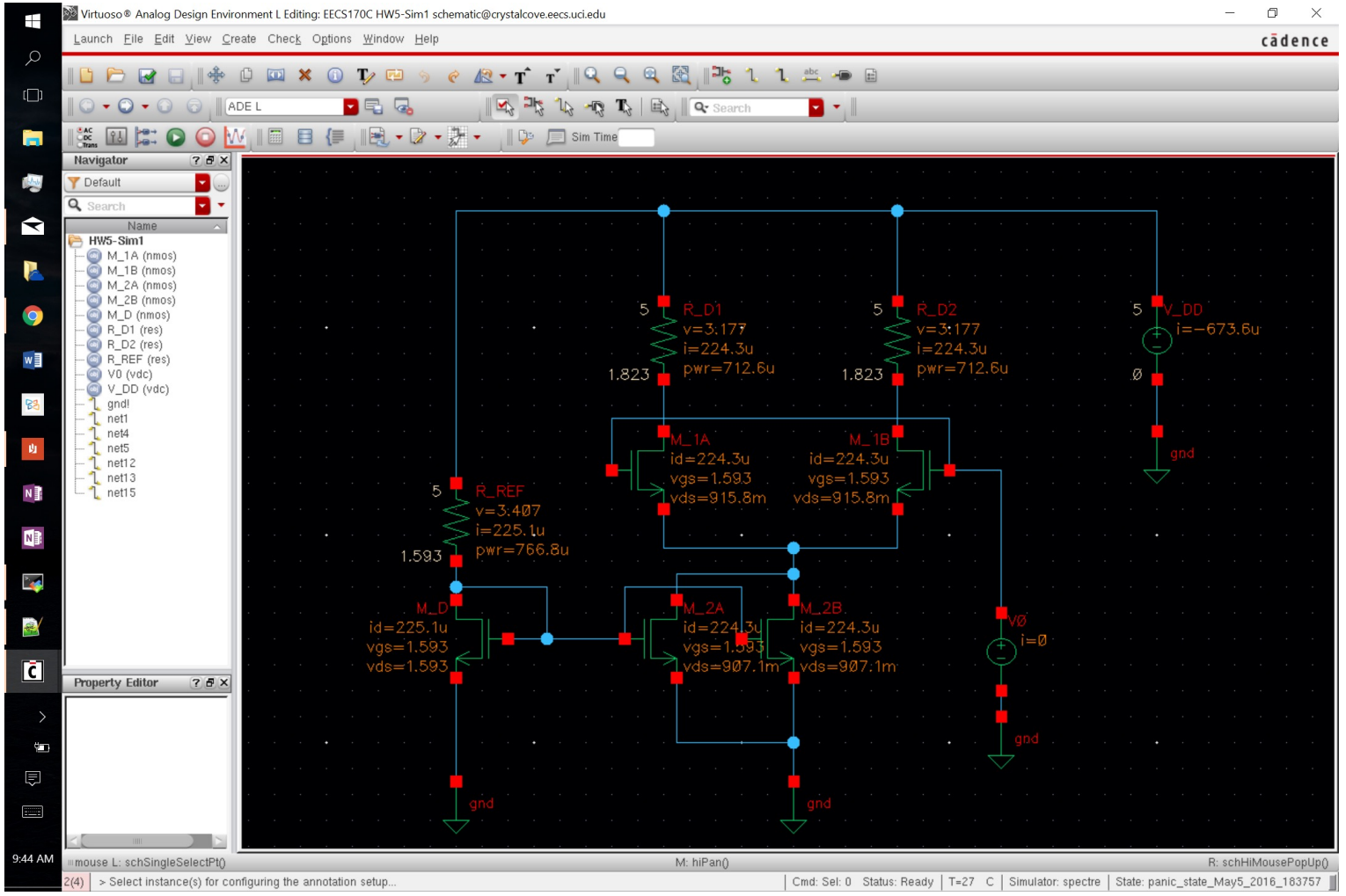


Figure 4: Schematic of the low power three-stage amplifier.

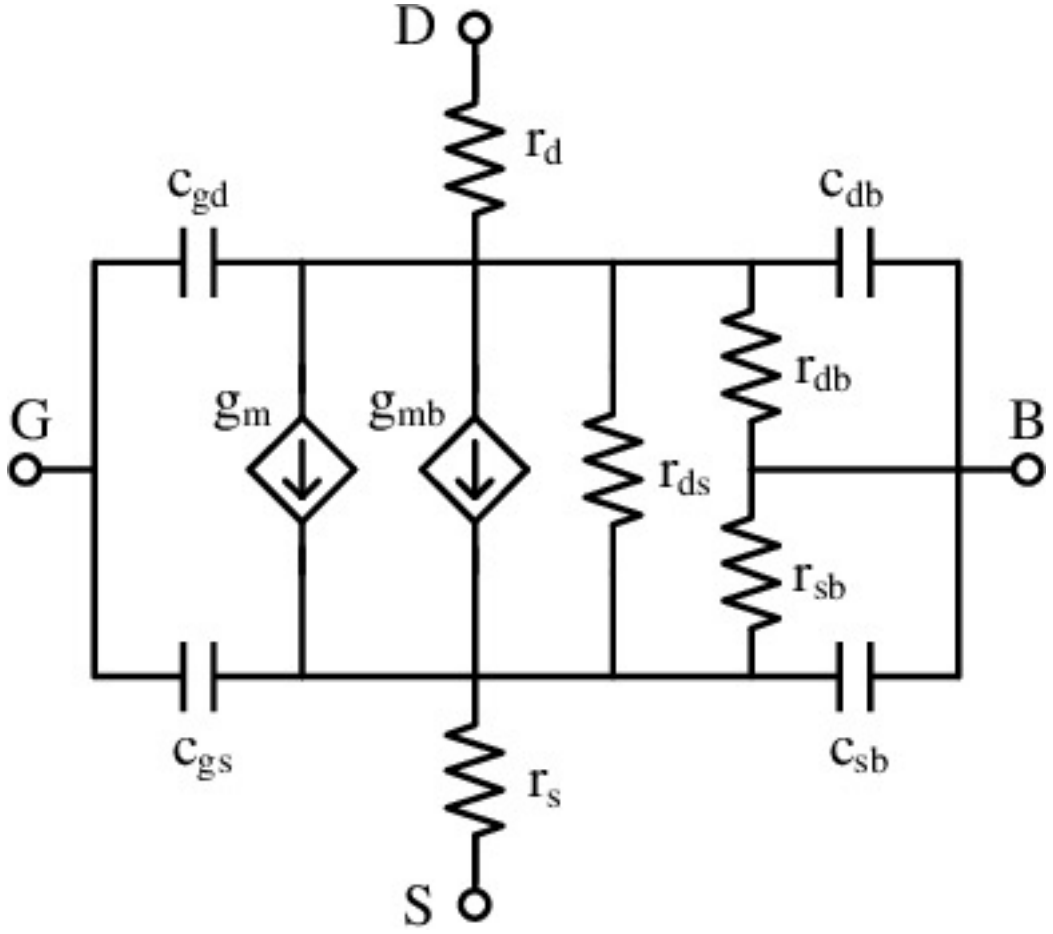
- **Requirements:**
  - **50dB gain**
  - **50dB SNR, etc.**

What about EDA tools?

- EDA tools can provide answer
- What should be FET size?
- Resistor value?
- Models give you ballpark
- Trial-and-error thereon

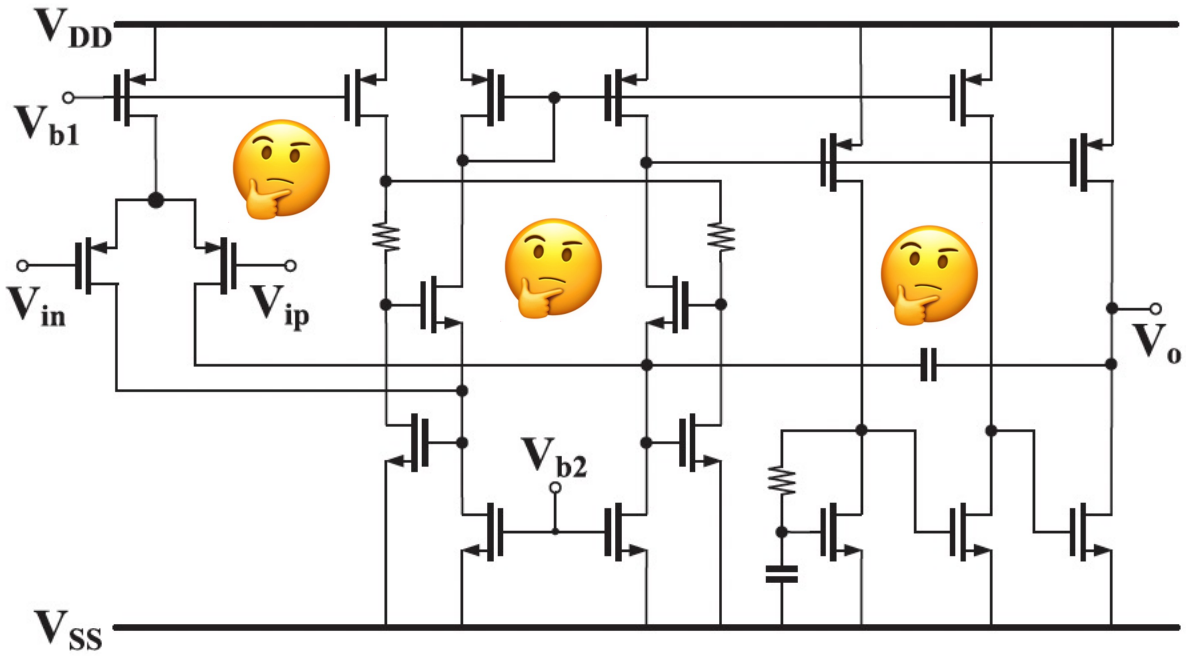


More accurate MOSFET model



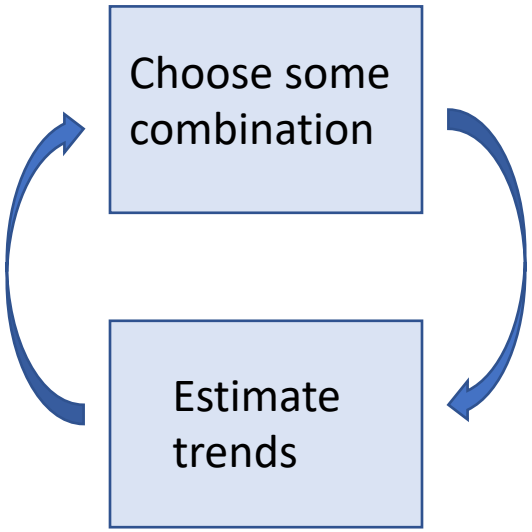
- Gives us closer to actual values
- More complex than earlier one
- Difficult to solve
- Does not scale well when #FET large

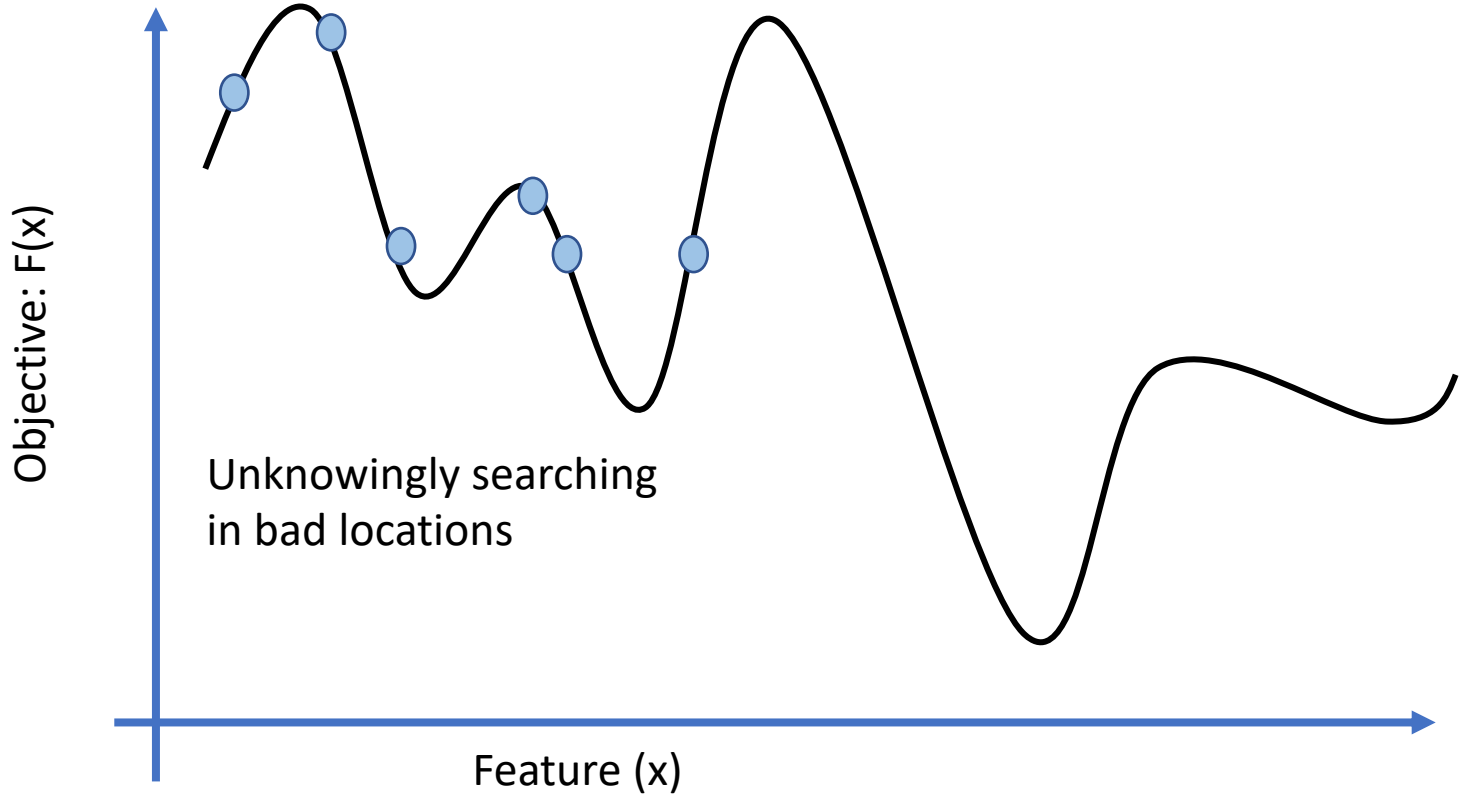
Our overall problem



- Vout = f(Vin, Size1, R1,…)
- Given some requirements (e.g., gain, SNR)
- Design the circuit that meets requirements

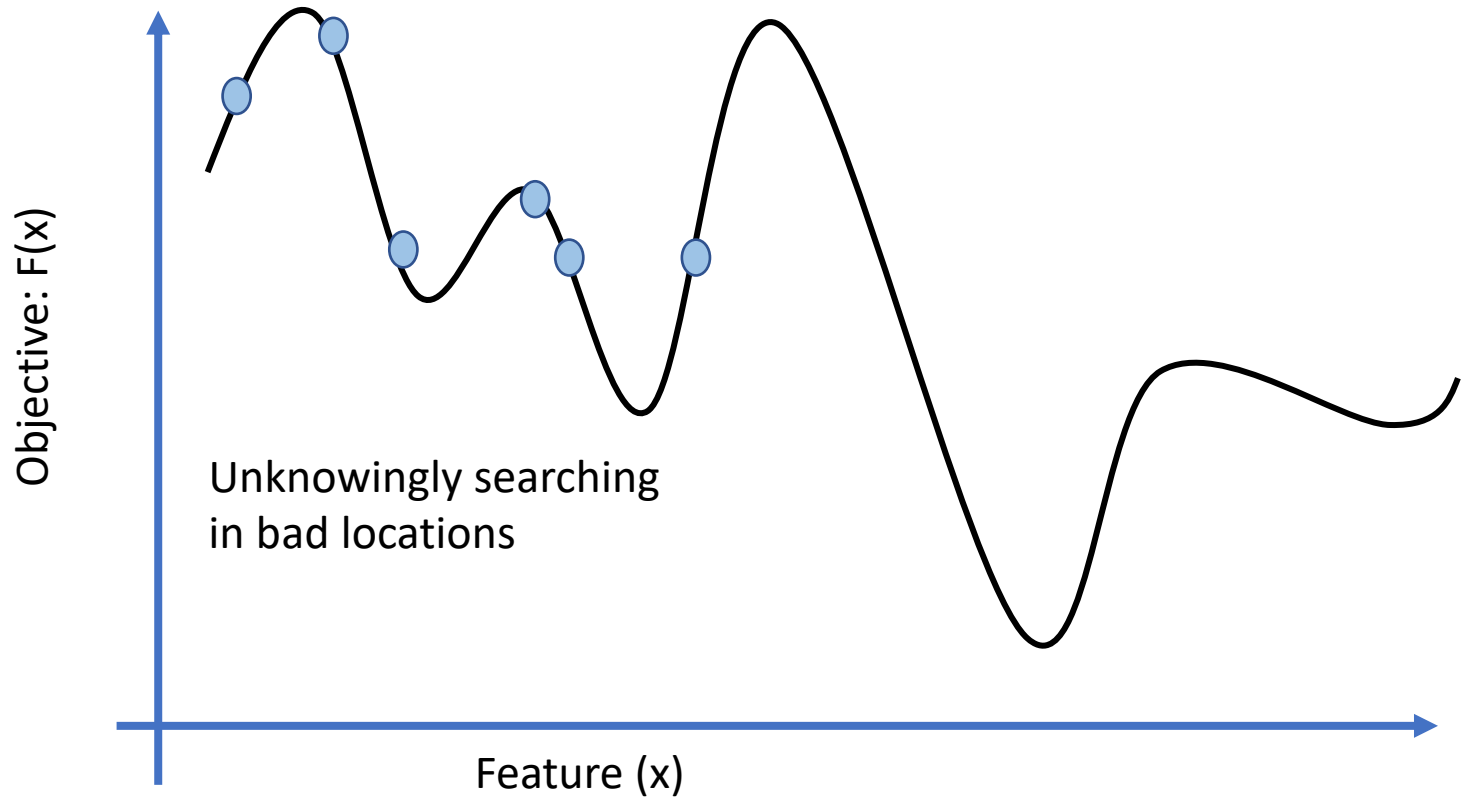
Problem with tuning variables/hyperparameters





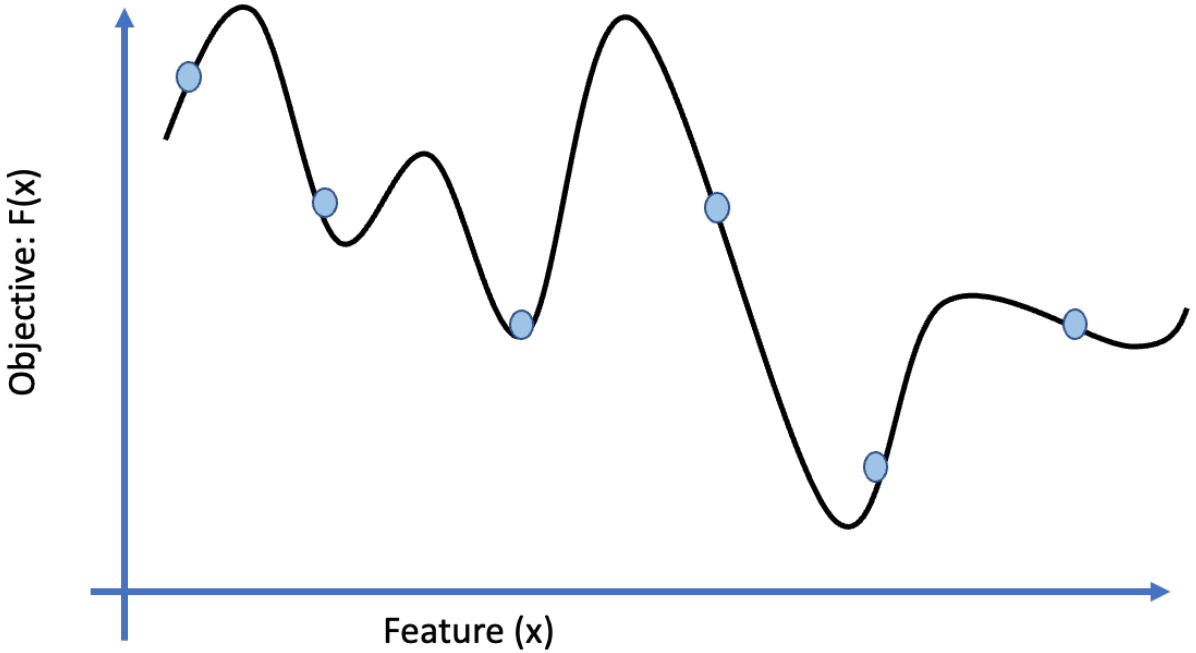
- **Easy to miss trends**
- **Difficult when #features high**

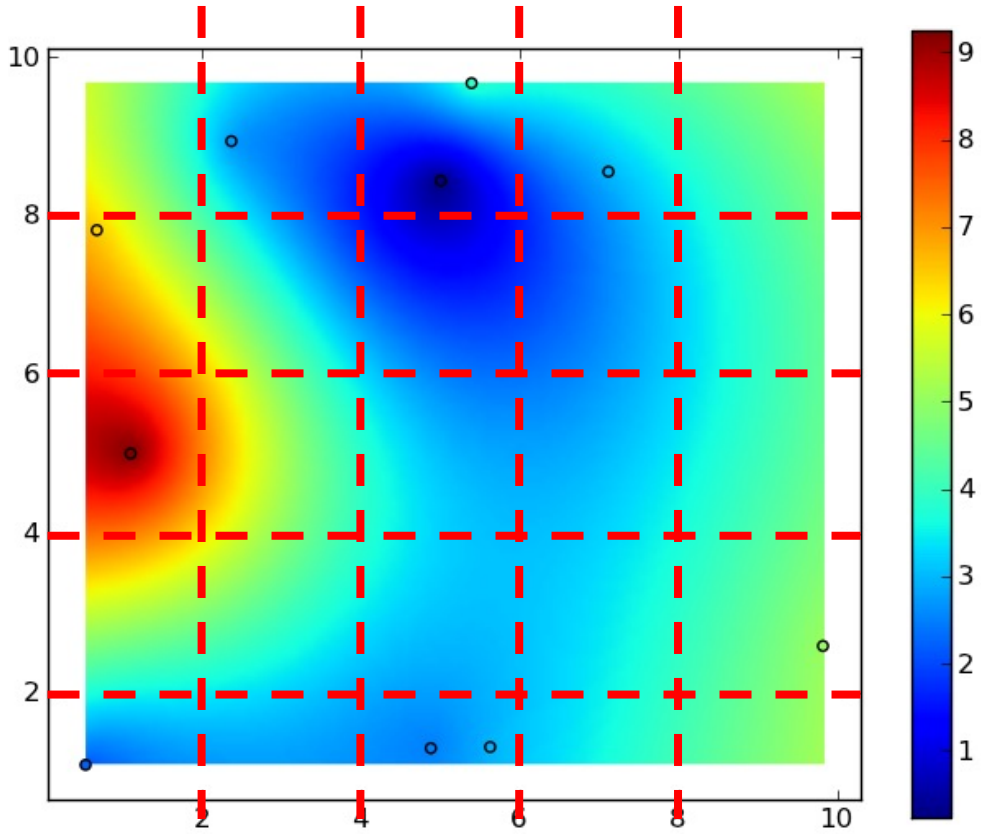
Random search



- **Not a good strategy for large design spaces or when # features is large**
- **Trial and error! We were already doing that!**

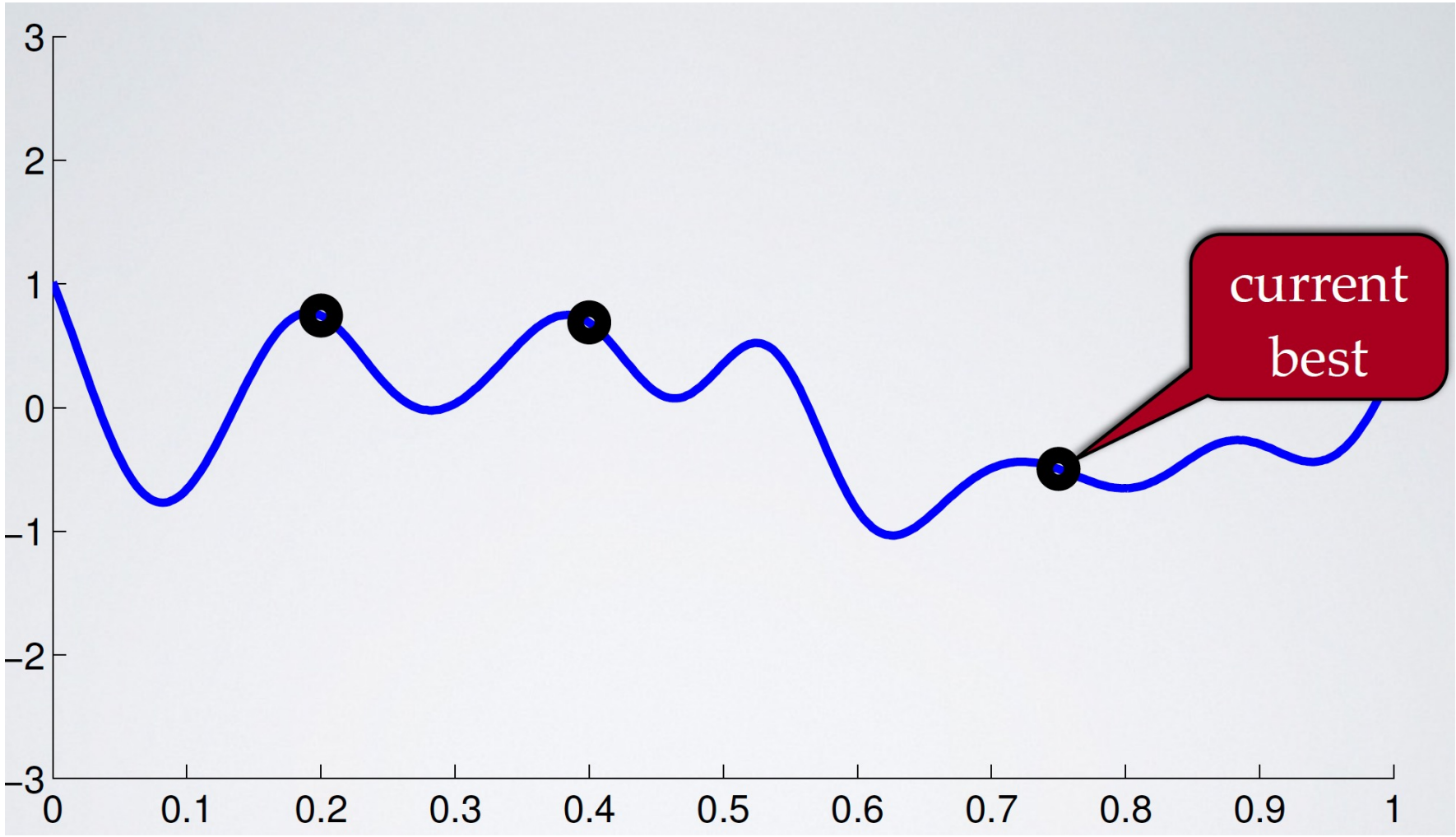
Grid search





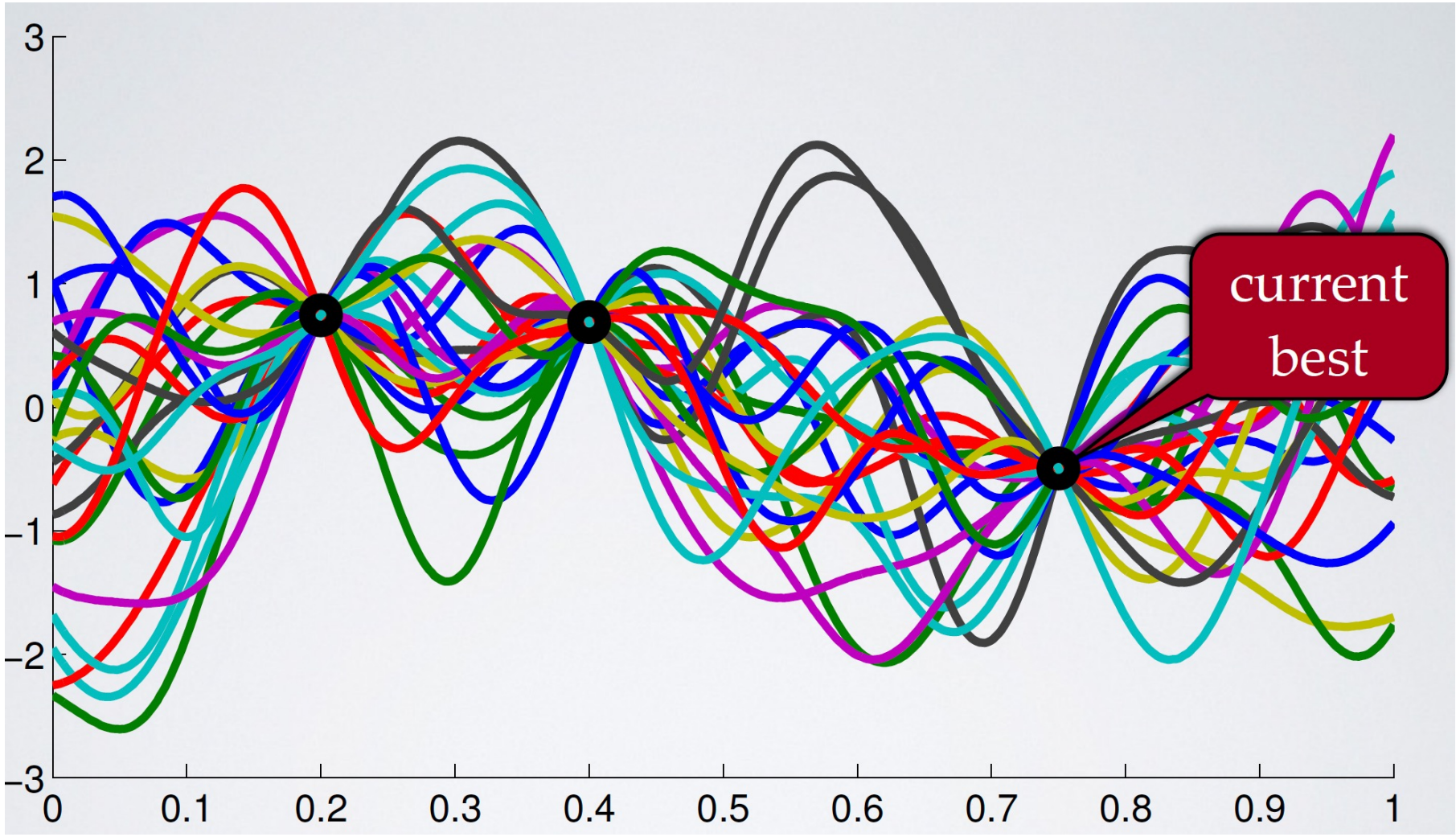
- Better than random search, more organized
- Assumes that every feature is equally important
- Can take a lot of time

Bayesian optimization



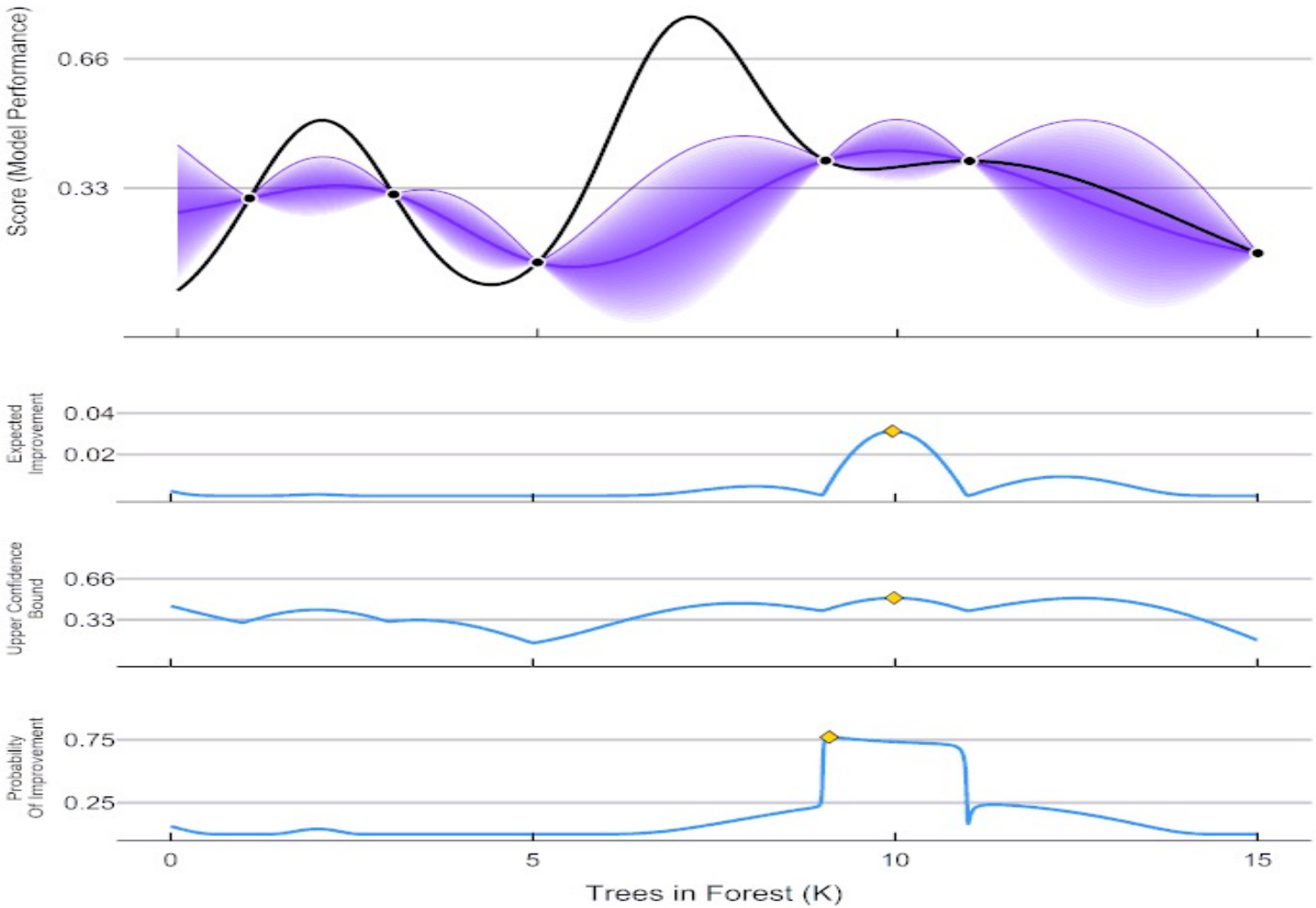
- **ML for function approximation**

Gaussian surrogate function

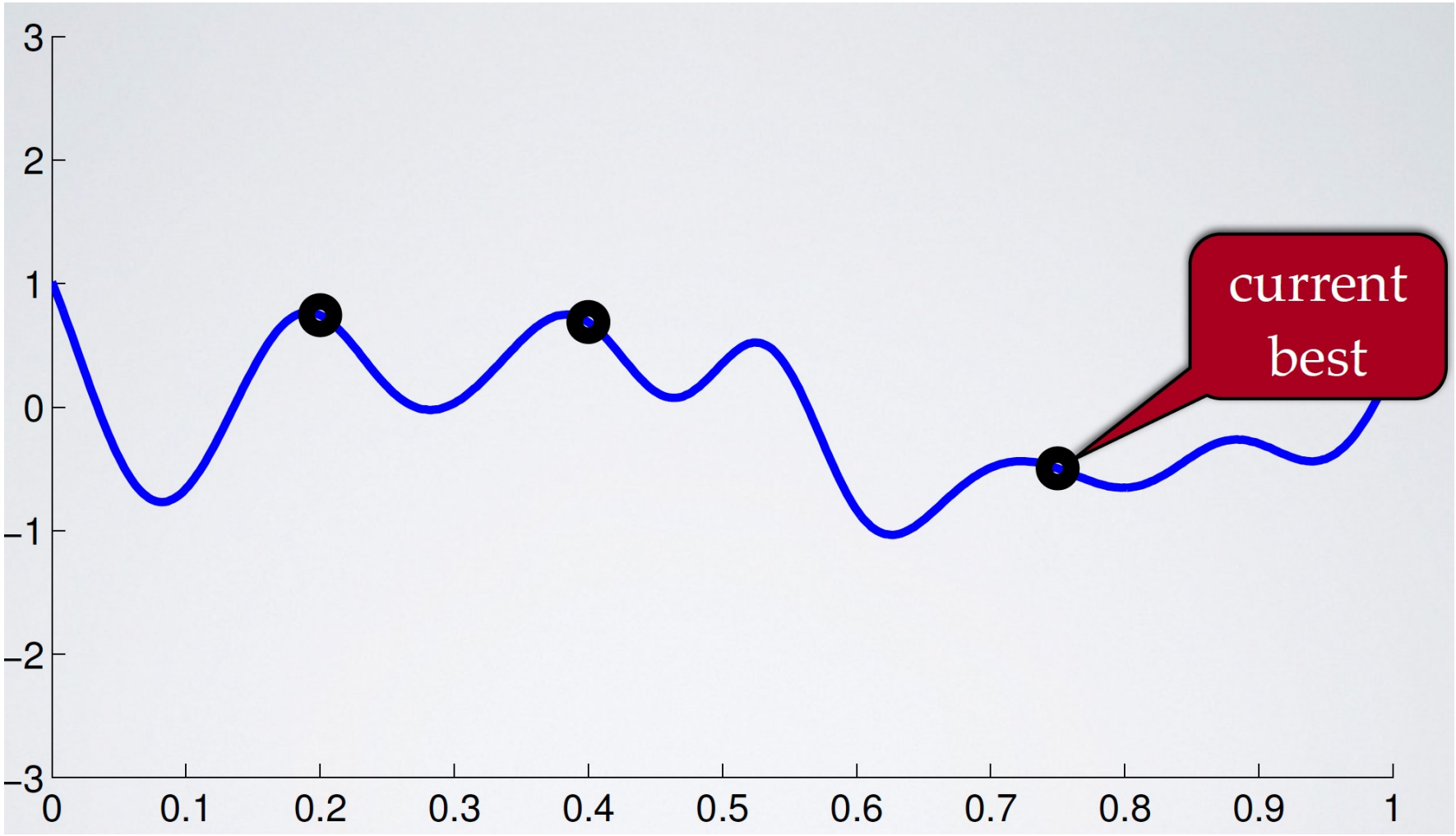


- $F(t) = a_1 t + a_2 t^2 + \cdots$
- **What are the values of a1, a2, etc?**

Bayesian optimization in action

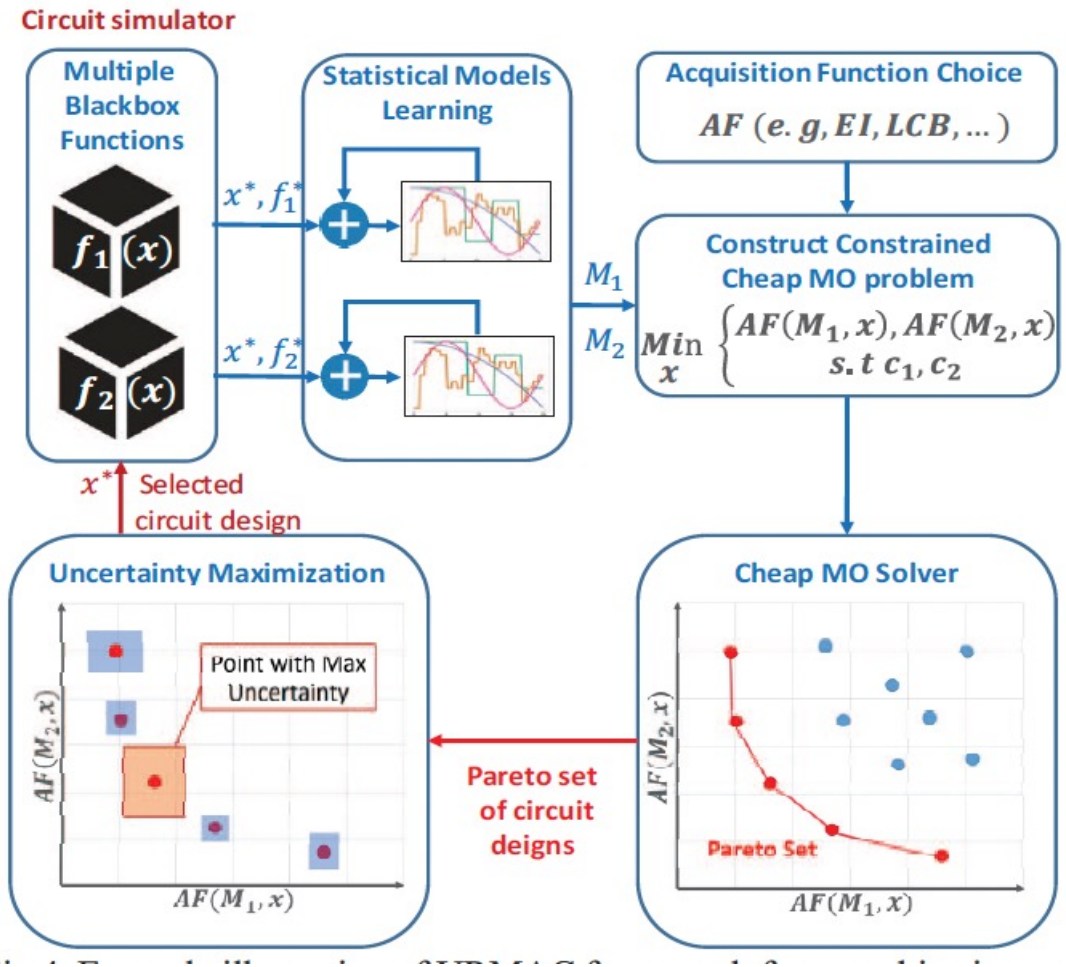


Bayesian optimization (overall)



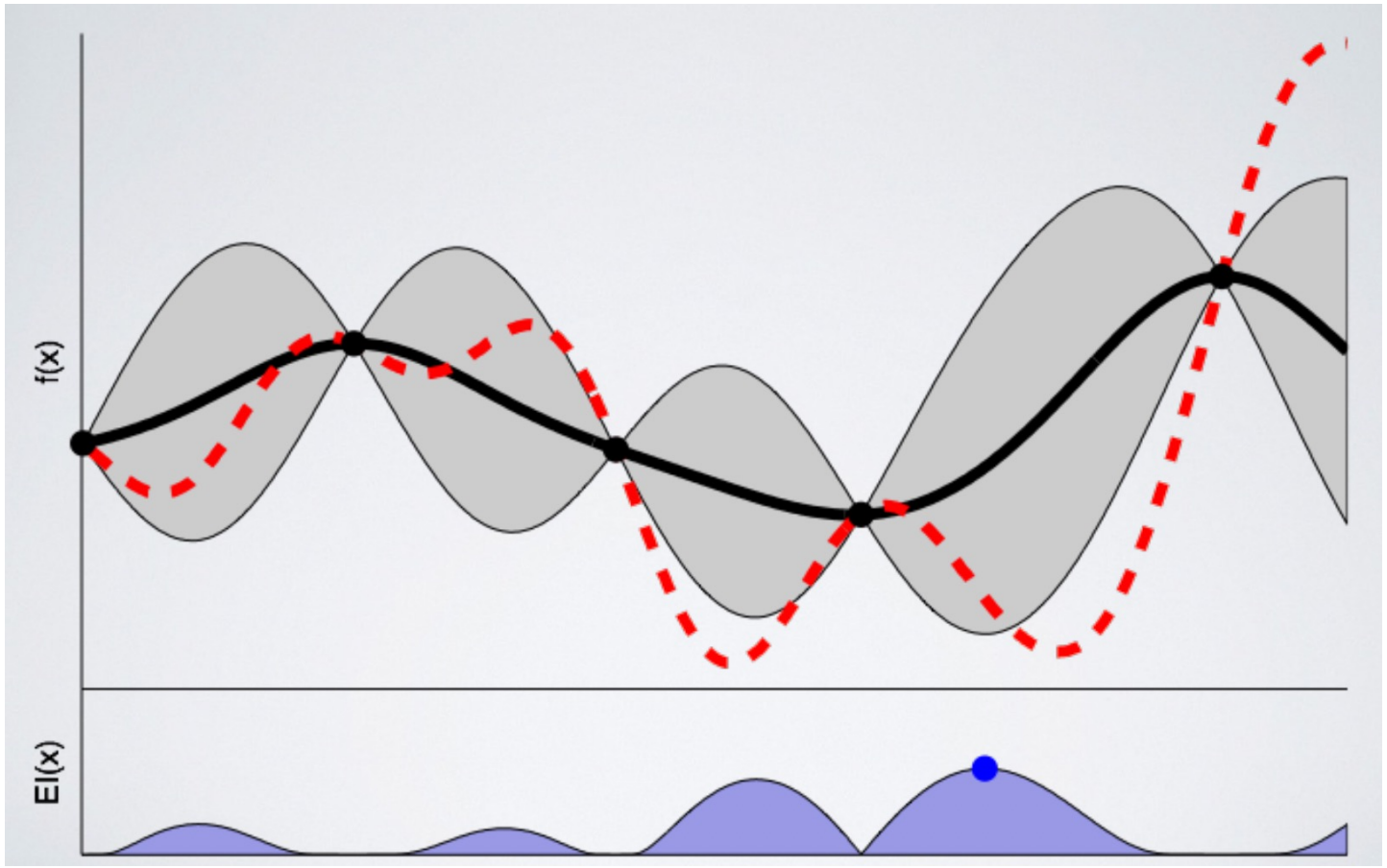
- **Get the best during optimization or use surrogate model to do further analysis**

Overall Analog design using BO



Z. Zhou et al., "Design of Multi-Output Switched-Capacitor Voltage Regulator via Machine Learning," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020, pp. 502-507, doi: 10.23919/DATE48585.2020.9116413.

Challenges of Bayesian optimization



- Too slow in practice
- 2 optimizations in one problem
- Time complexity high $O(n^4)$
- Not good when #hyperparameters too high

Reinforcement learning for analog design

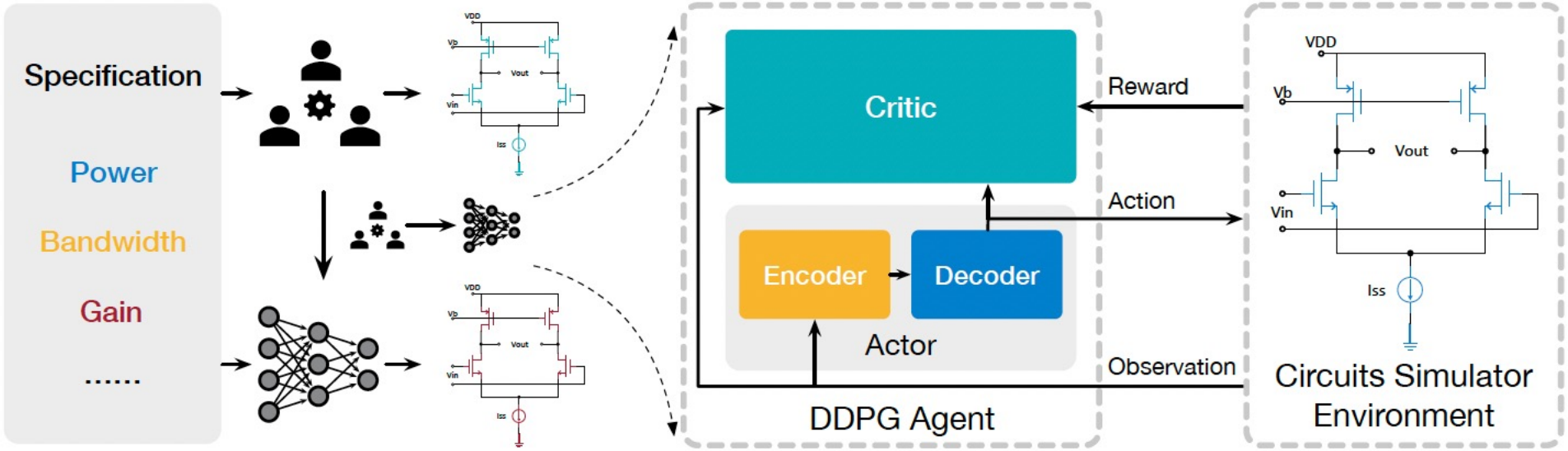
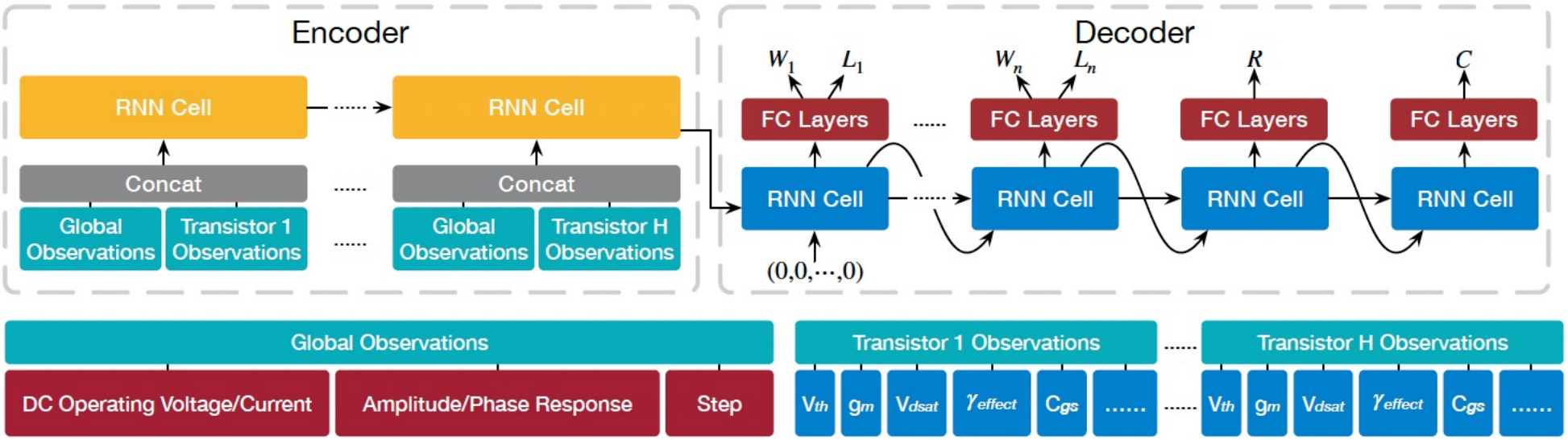


Figure 1: Learning to Design Circuits (L2DC) Method Overview.

- **Another option: RL**
- **Vary the parameters in an unsupervised manner**
- **Check if goal is met (reward)**

Wang, Hanrui & Yang, Jiacheng & Lee, Hae-Seung & Han, Song. (2018). Learning to Design Circuits. In arXiv:1812.02734v5

Encoder and Decoder for the RL method



- **Encoder/Decoder designed to convert parameters to intermediate values**
- **Chained as first transistor affects current on second transistor and so on**

Wang, Hanrui & Yang, Jiacheng & Lee, Hae-Seung & Han, Song. (2018). Learning to Design Circuits. In arXiv:1812.02734v5

Results of using ML for analog design (1)

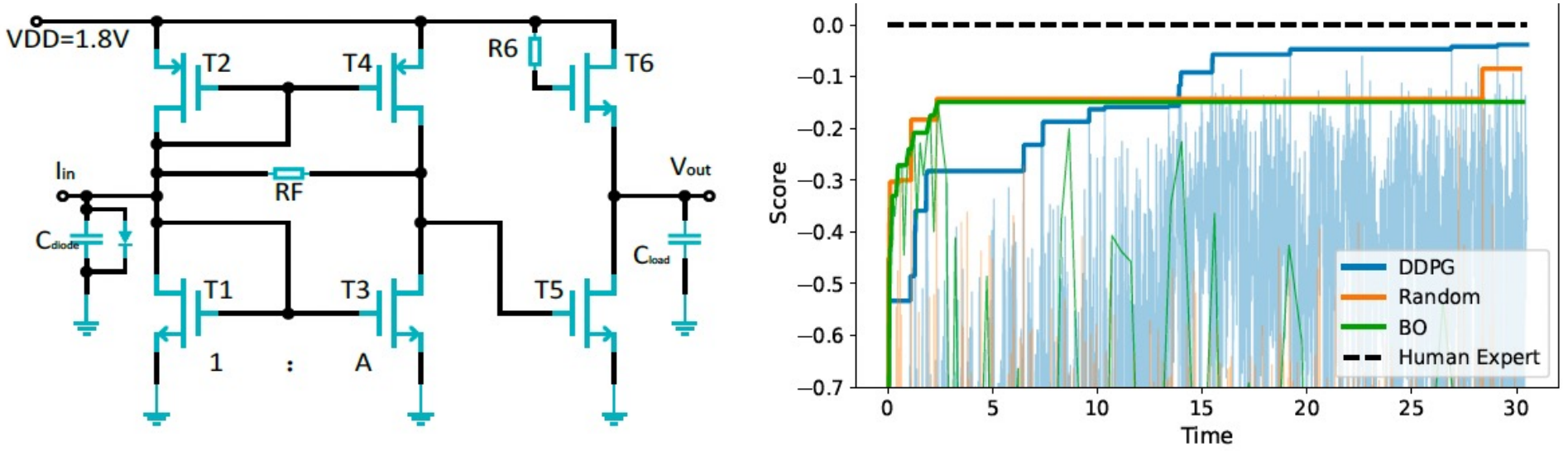


Figure 4: Left: Schematic of two-stage tranimpedence amplifier. Right: Learning curves of two-stage transimpedence amplifier.

- **2-stage transimpedance amplifier design problem**
- **RL achieves near human level performance**

Results of using ML for analog design (2)

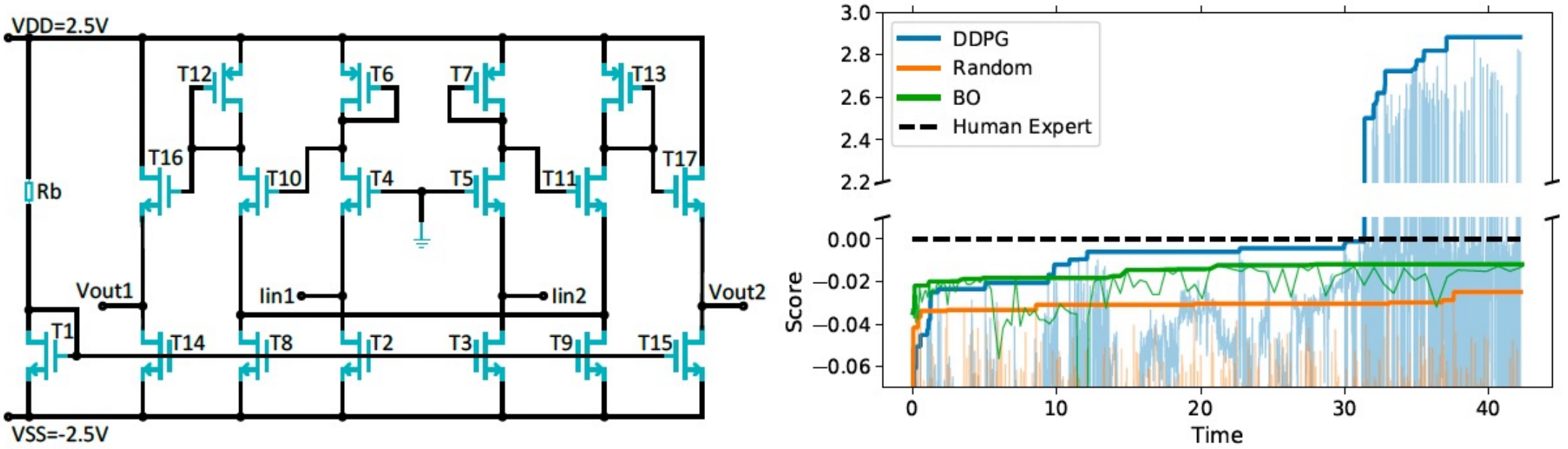
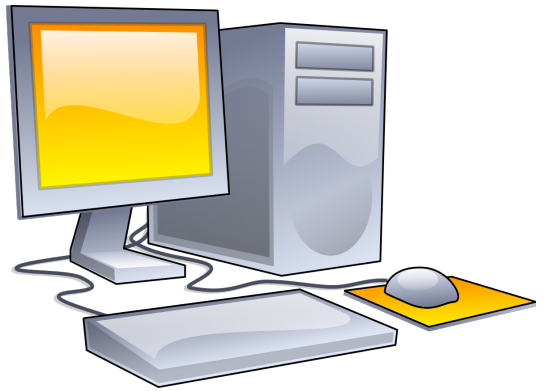


Figure 3: Left: Schematic of Three-stage transimpedence amplifier. Right: Learning curves of three-stage transimpedence amplifier.

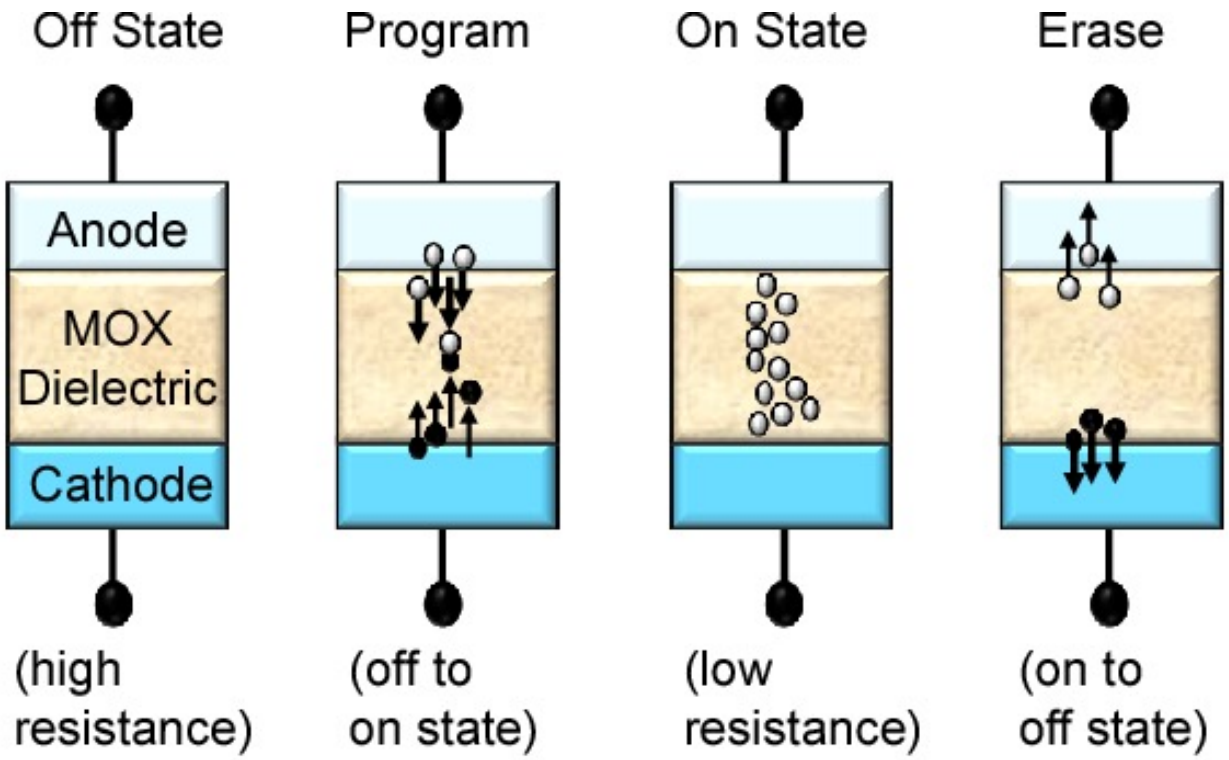
- **3-stage transimpedance amplifier design problem**
- **BO achieves better performance than random trial-and-error**
- **RL achieves better than human level performance (given sufficient time)**

Analog manycore using ReRAM

- Analog accelerators can lead to 100X speed-up
- Problems:
  - Hardware defects, BN layers
- Can we use ML to solve a problem of ML
- BO for Neural Architecture Search



Resistive Random-Access Memory (ReRAM)



- **Dielectric with two metal electrodes at the end**
- **Ion position determine the resistance**

Analog computing: ReRAMs

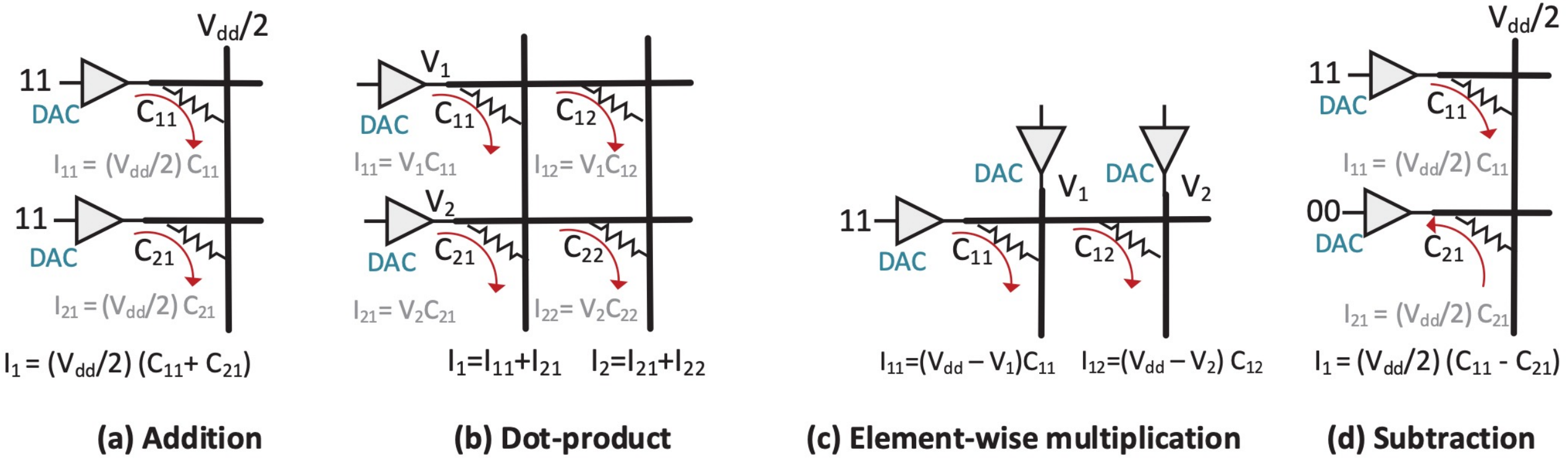


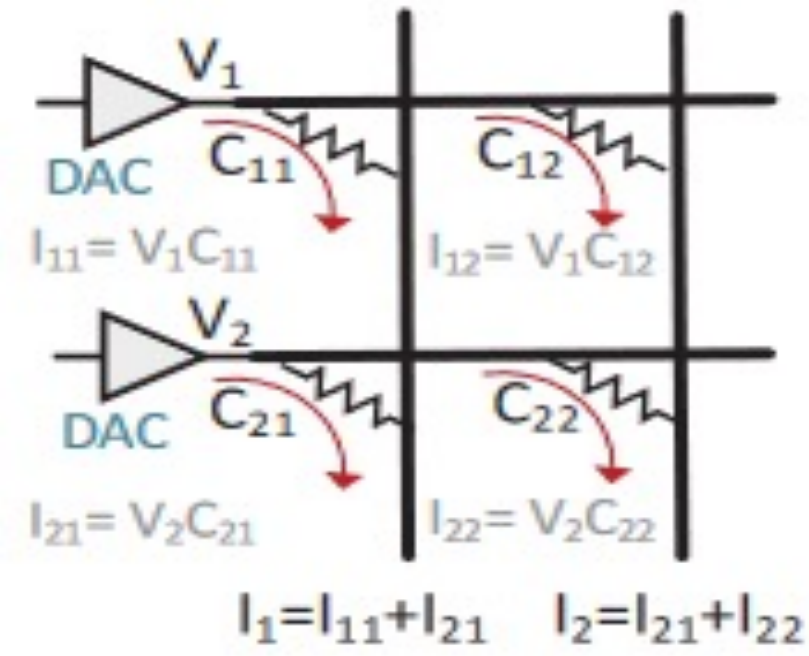
Figure 2. In-situ ReRAM array operations.

- **Ohm's law and Kirchoff's current law**

Daichi Fujiki, Scott Mahlke, and Reetuparna Das. 2018. In-Memory Data Parallel Processor. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18). Association for Computing Machinery, New York, NY, USA, 1–14

ReRAMs for Deep Learning

- **Most Deep learning algorithms are MVMs**
  - **CNNs, GNNs, etc.**
- **ReRAMs are natural MVMs**
  - **N^2 multiplications in O(1) time**
  - **Energy efficient**



$$\begin{bmatrix} C11 & C21 \\ C12 & C22 \end{bmatrix} \times \begin{bmatrix} V1 \\ V2 \end{bmatrix} = \begin{bmatrix} I1 \\ I2 \end{bmatrix}$$

The problem with GPUs

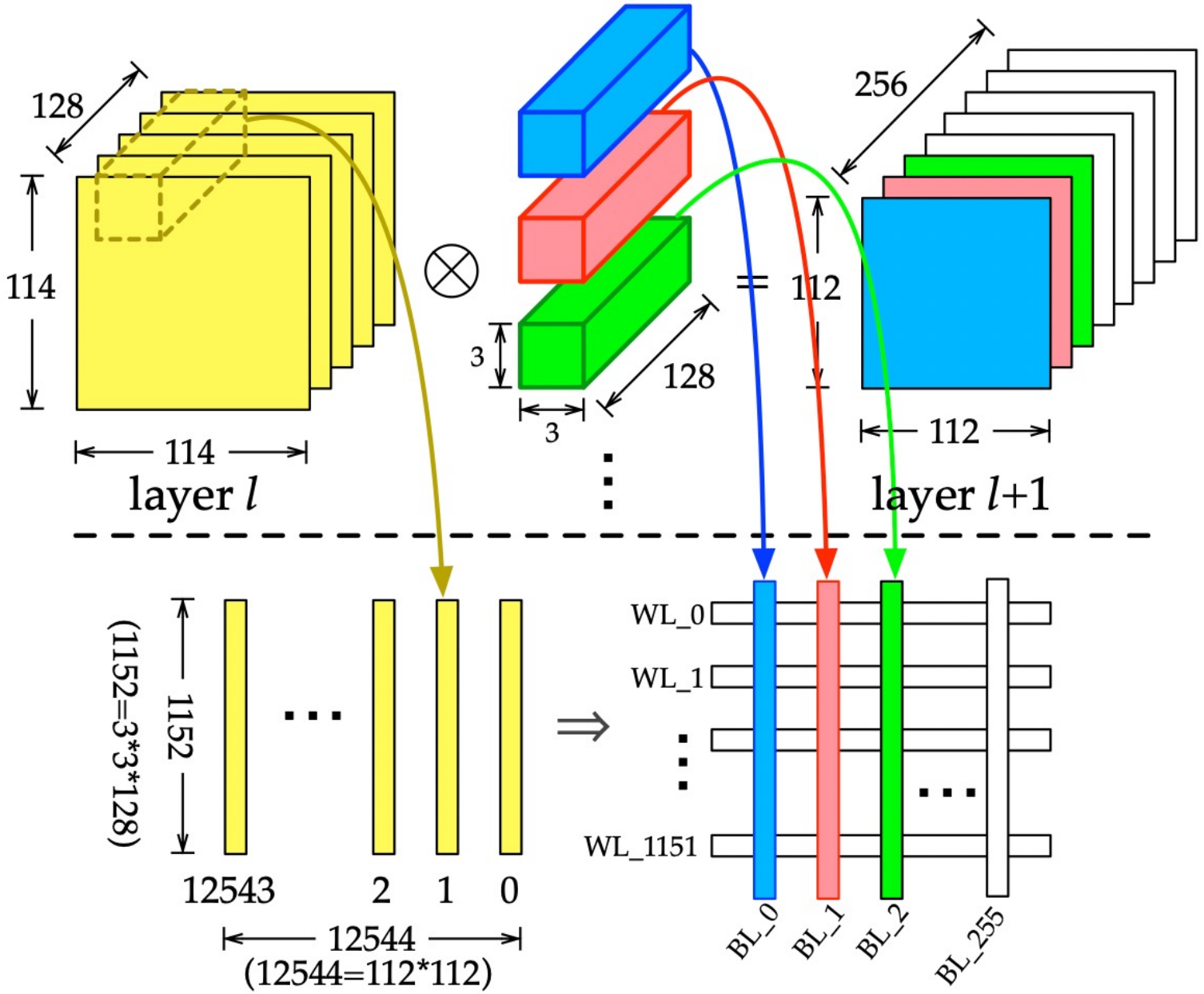
Common Hardware platforms for Deep learning applications:

- **CPU + *External* GPU**
- **Cloud Services: Amazon EC2 or Microsoft Azure**

GPUs are not optimized for Deep learning

- **Bandwidth bottleneck**
- **Sub-optimal performance, energy**

Convolution/FC on ReRAMs



ReRAM vs GPU

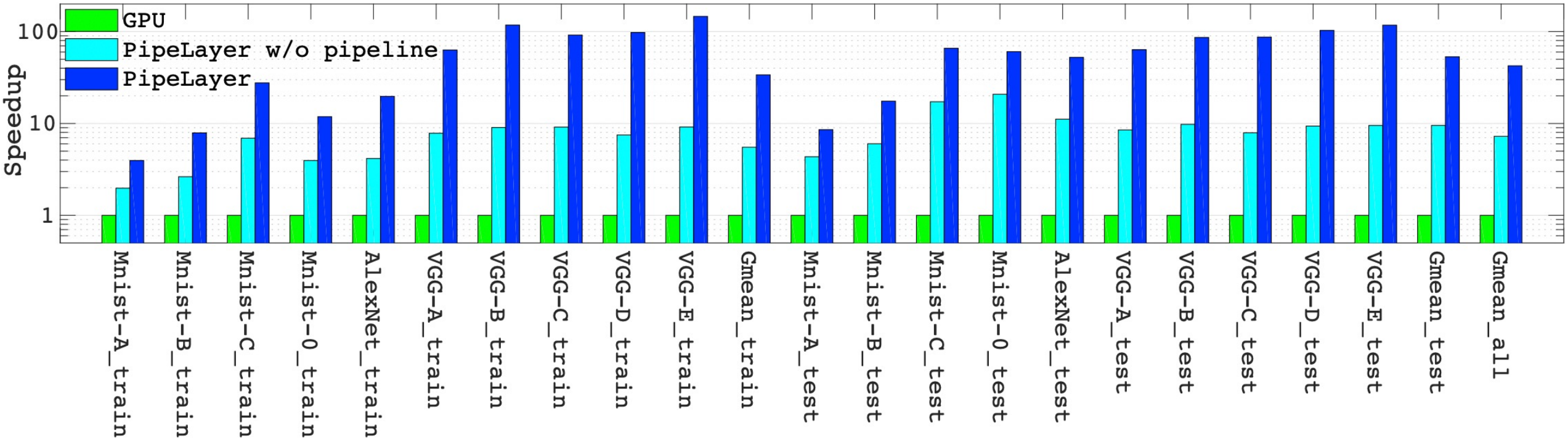


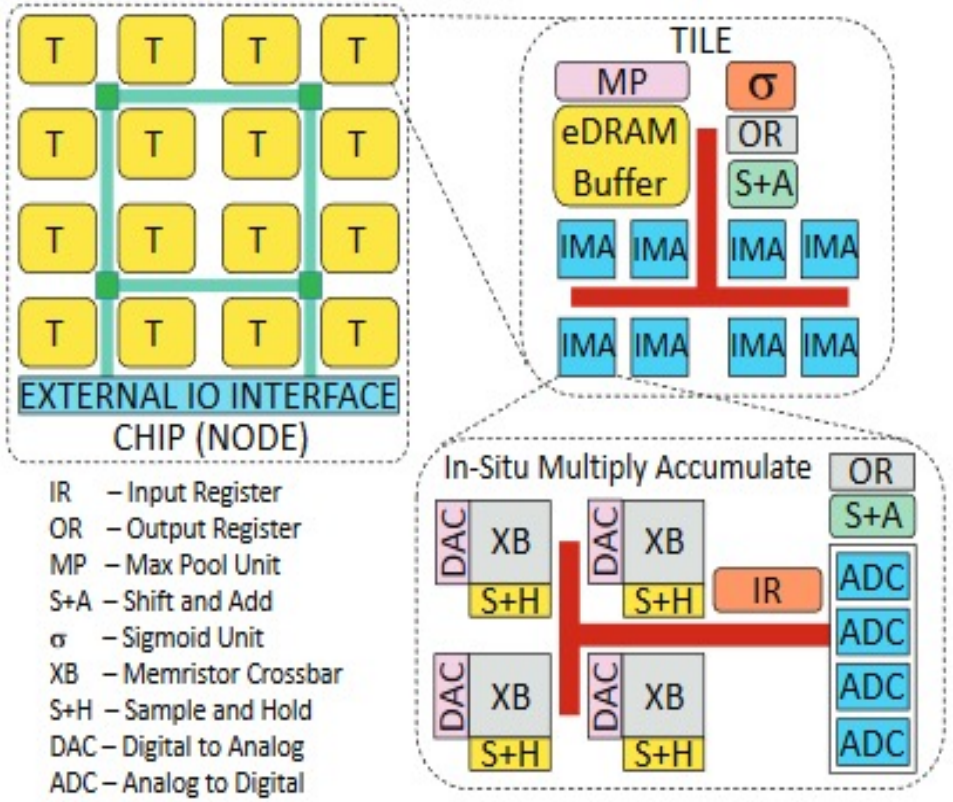
Figure 15: Speedups of Networks in Both Training and Testing

- **Up to 100X speed-up possible**
- **High throughput MAC on ReRAMs**

L. Song, X. Qian, H. Li and Y. Chen, "PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning," 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2017, pp. 541-552

Existing ReRAM-based architectures

- **Costly, Immature fabrication**
- **Low Precision**
  - **Accuracy Loss**
  - **Unstable training**
- **Lack of Normalization**
  - **Requires full-precision**



[1] A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," 2016 ACM/IEEE 43rd ISCA, Seoul, 2016, pp. 14-26.

BN layers

- **Scales data to have zero mean and unit variance**
- **Solves the internal covariate shift problem**
- **Prevents exploding/vanishing gradients**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

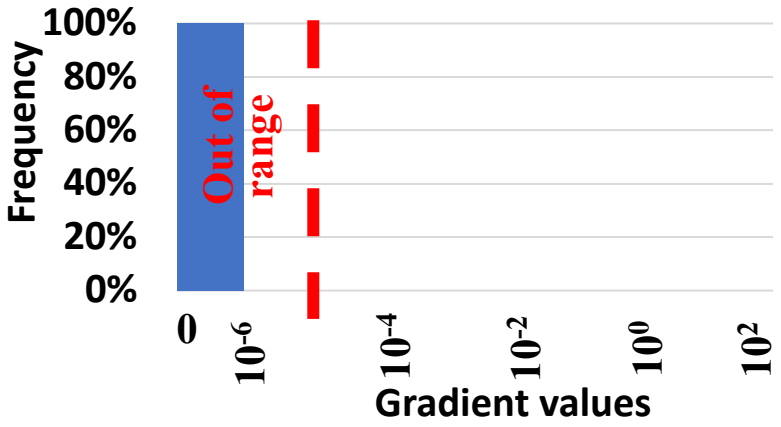
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

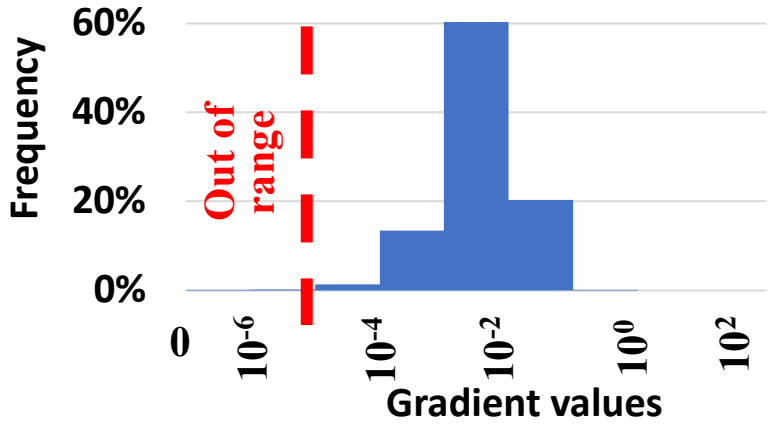
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Low-precision and lack of Normalization



No Normalization support

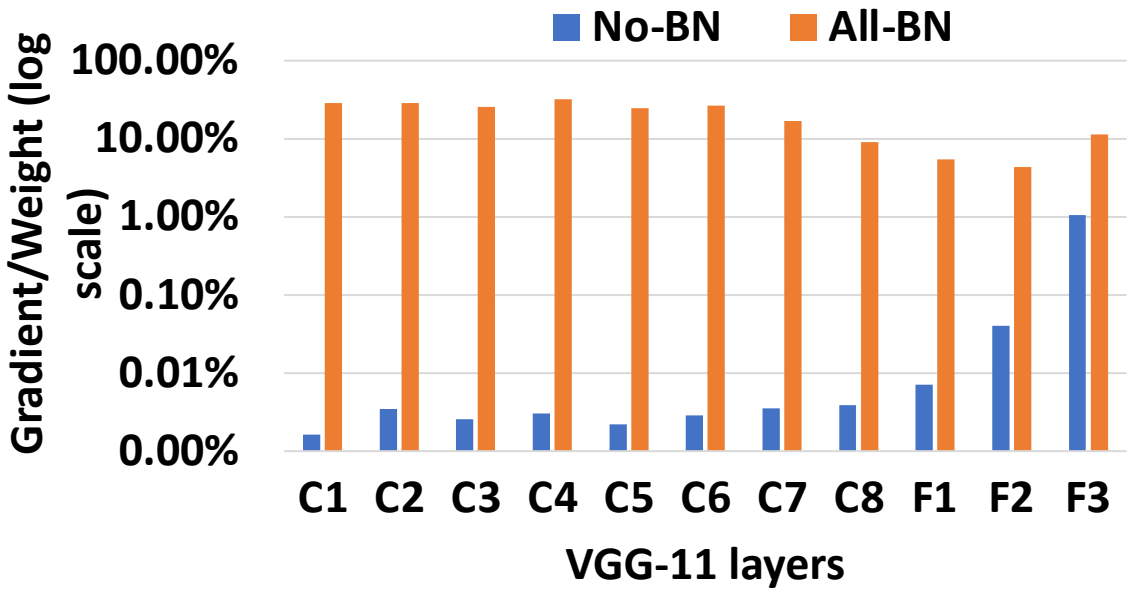


With Normalization support

$$w_{new} = w_{old} - \alpha * \Delta w$$

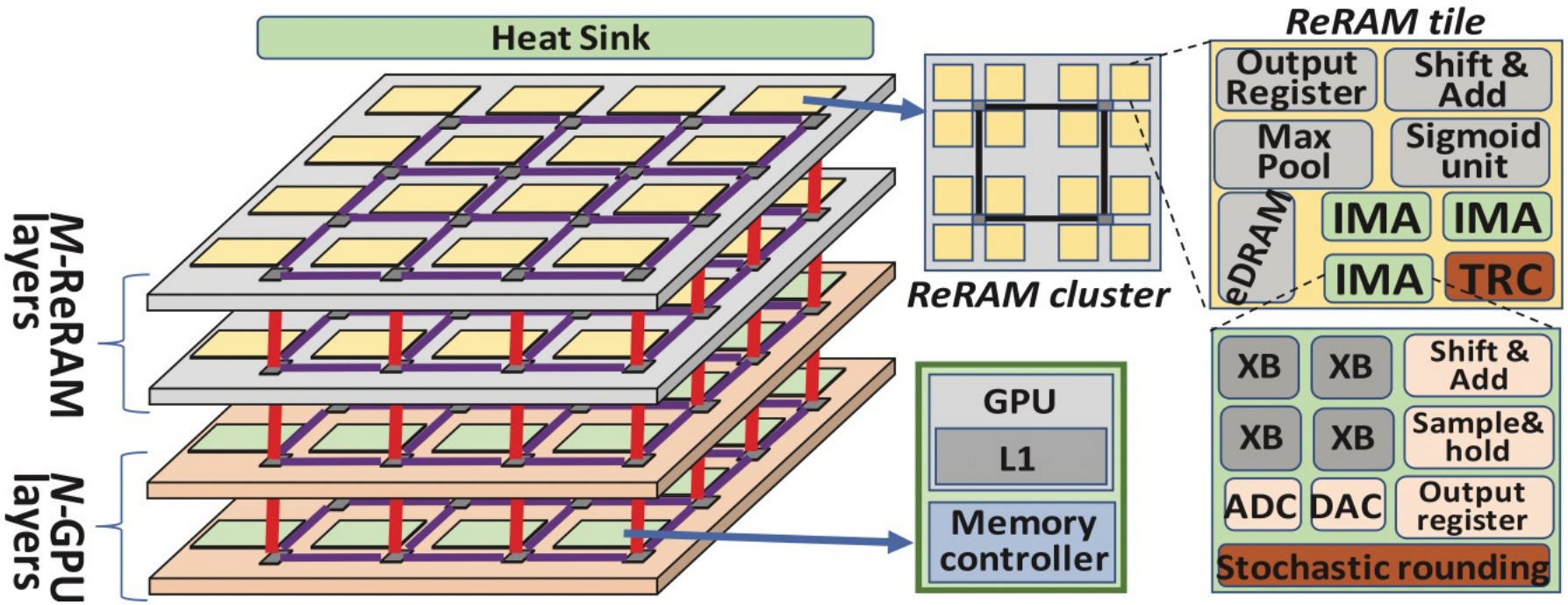
- Gradients too small without normalization
- Low precision cannot represent too small/large values
  - Gradients rounded to zero
  - No (or minimal) weight update
    - No meaningful learning

Results on absence/presence of BN



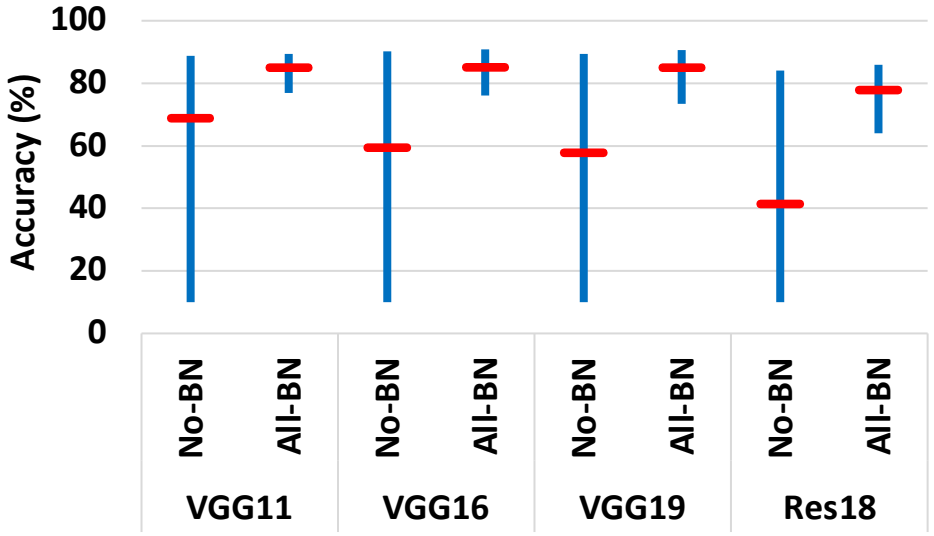
- Gradients too small without normalization
- CNN does not train successfully
- How to introduce BNs to ReRAMs?

New architecture?



- Heterogeneous architecture with GPUs
- GPUs support full-precision and hence BN
- Conv/FC→ReRAMs and BN→GPUs
- GPUs = Area, Energy overhead

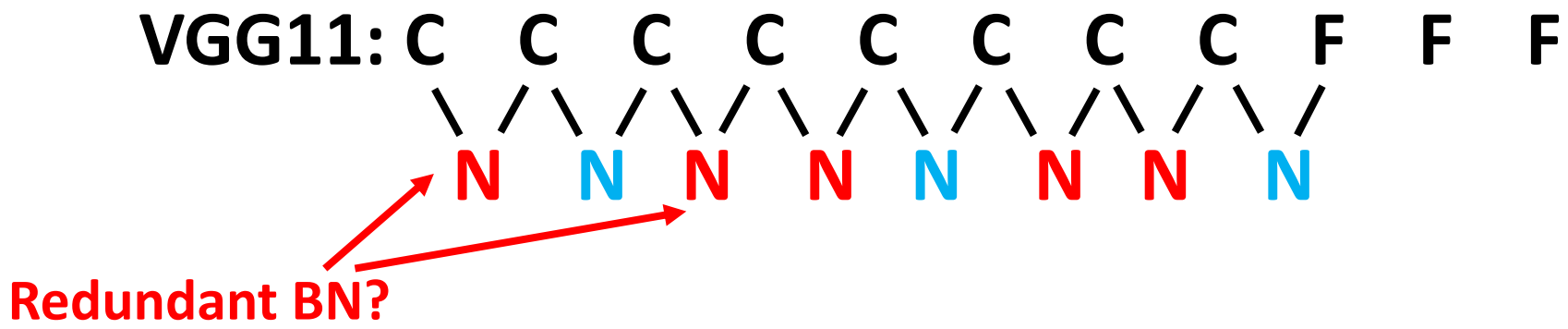
Can we remove the Normalization layers?



- **Batch Normalization (BN)**
  - **Prevents Vanishing/Exploding gradients problem**
- **Xavier/Kaiming initialization?**
  - **Sensitive to hyper-parameters**

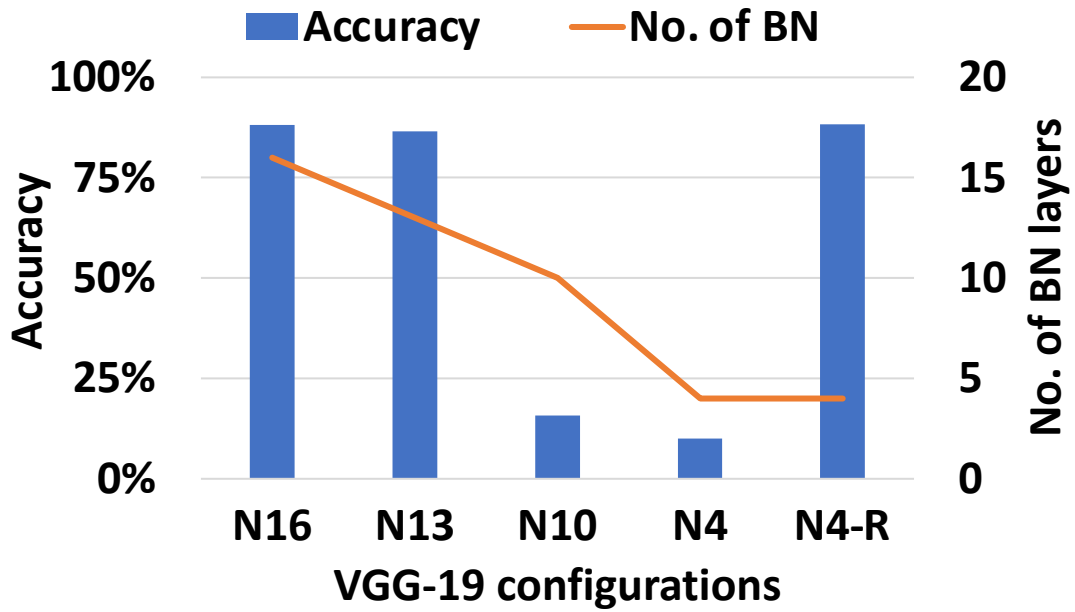
[1] B. K. Joardar, A. Deshwal, J. R. Doppa, P. P. Pande and K. Chakrabarty, "High-Throughput Training of Deep CNNs on ReRAM-based Heterogeneous Architectures via Optimized Normalization Layers," in IEEE TCAD, 2021
[2] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks,” In AISTATS, 2010

Are all Normalization layers necessary?



- How to know which layers are redundant?
  - No mathematical model/formulation or prior works
- CNN training is expensive
  - 2^P possible solutions
  - Trial-and-error or Exhaustive exploration not feasible

Removing BN



- How to know which layers are redundant?
  - No mathematical model/formulation or prior works
- CNN training is expensive
  - 2^P possible solutions
  - Trial-and-error or Exhaustive exploration not feasible

Naïve solution to find redundant BNs

VGG11: C C C C C C C C F F F

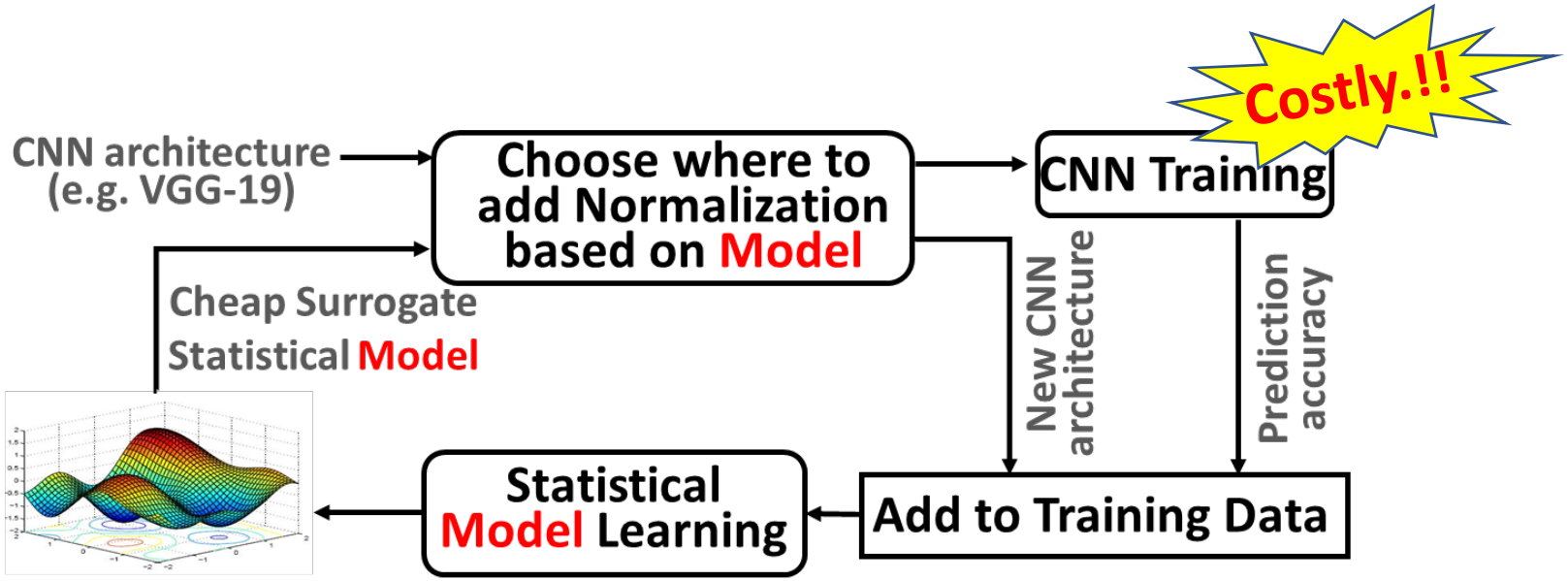
N N N N N N N N

....

N N N N N N N N

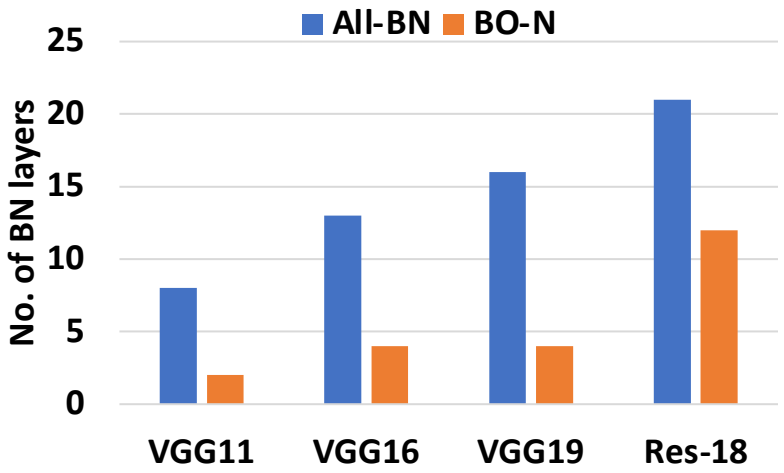
- **CNN training is expensive**
  - **2^P possible solutions**
  - **Only way to know is to train for N epochs**
  - **Trial-and-error or Exhaustive exploration not feasible**

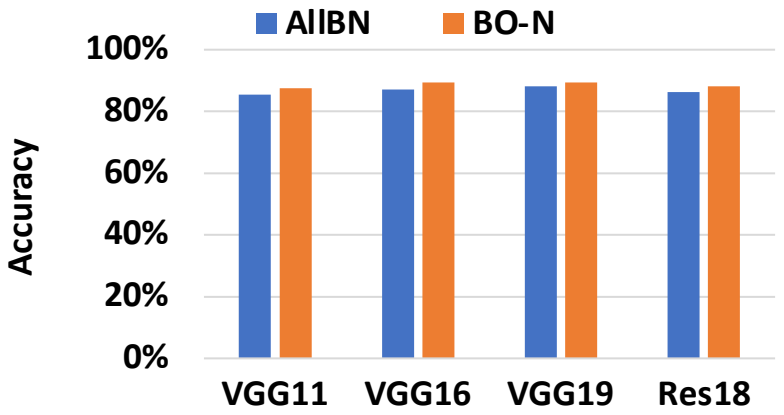
ML for ML: Bayesian Optimization to find important BN layers



- Bayesian optimization used when evaluation is costly
- Unlike conventional Bayesian Optimization:
  - Discrete design space
  - Sparsity-aware (few BNs)

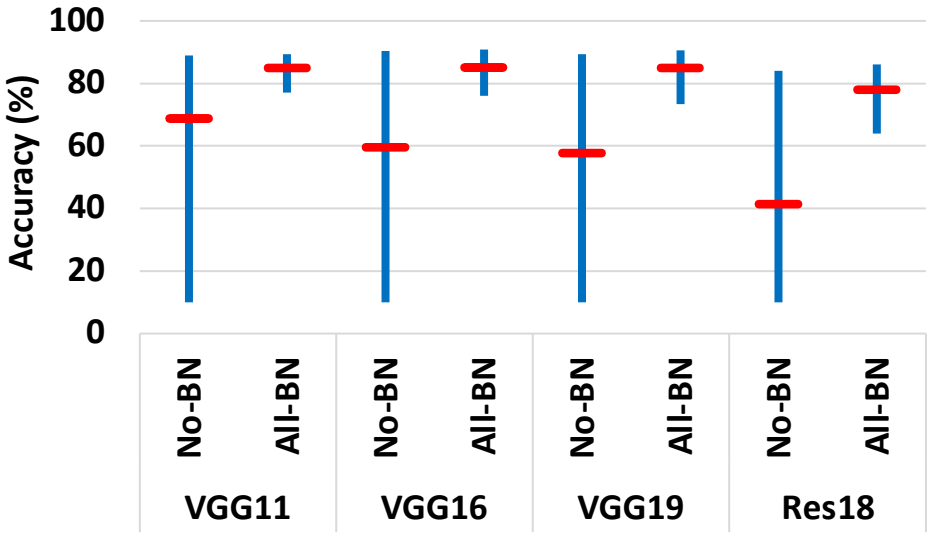
Optimized CNNs

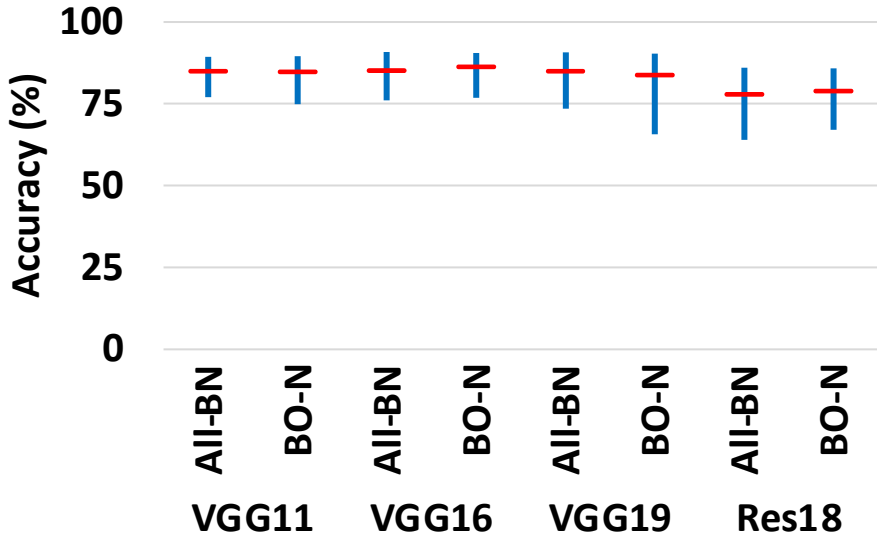




- Most BN layers are unnecessary
- No accuracy drop compared to All-BN

Robustness to hyper-parameters





- **Agnostic to the choice of hyper-parameters as All-BN**

Effect of BO

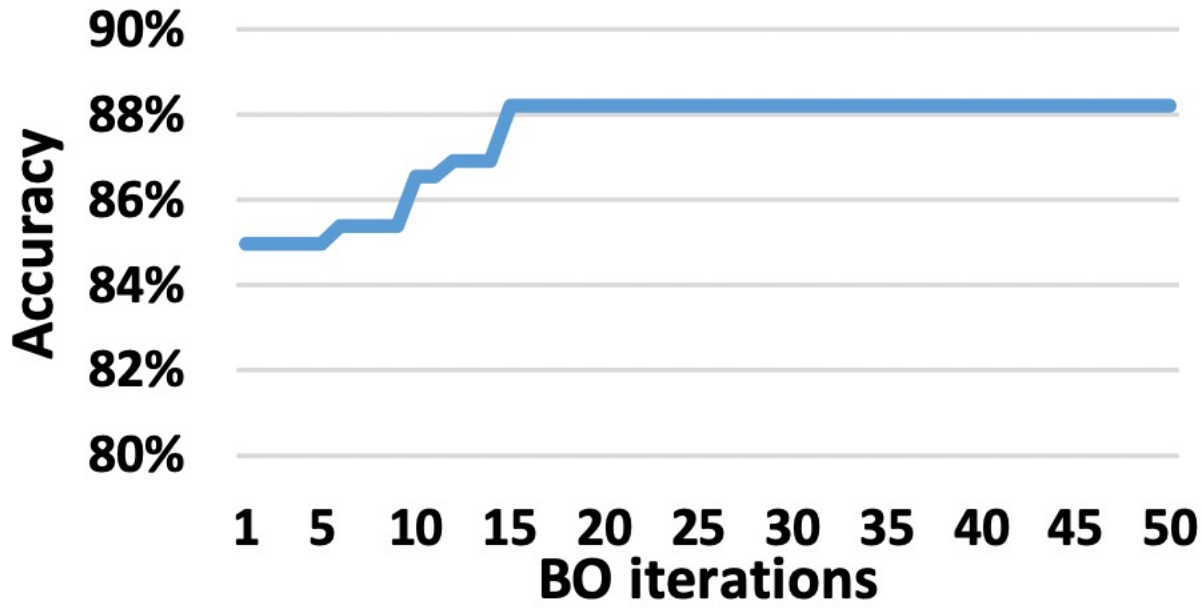


Fig. 8: Best model accuracy achieved after different BO iterations

- **BO converges very fast in practice**
- **~10-20 iterations necessary**
- **Significantly less than 2^n**

Is this good for GPUs?

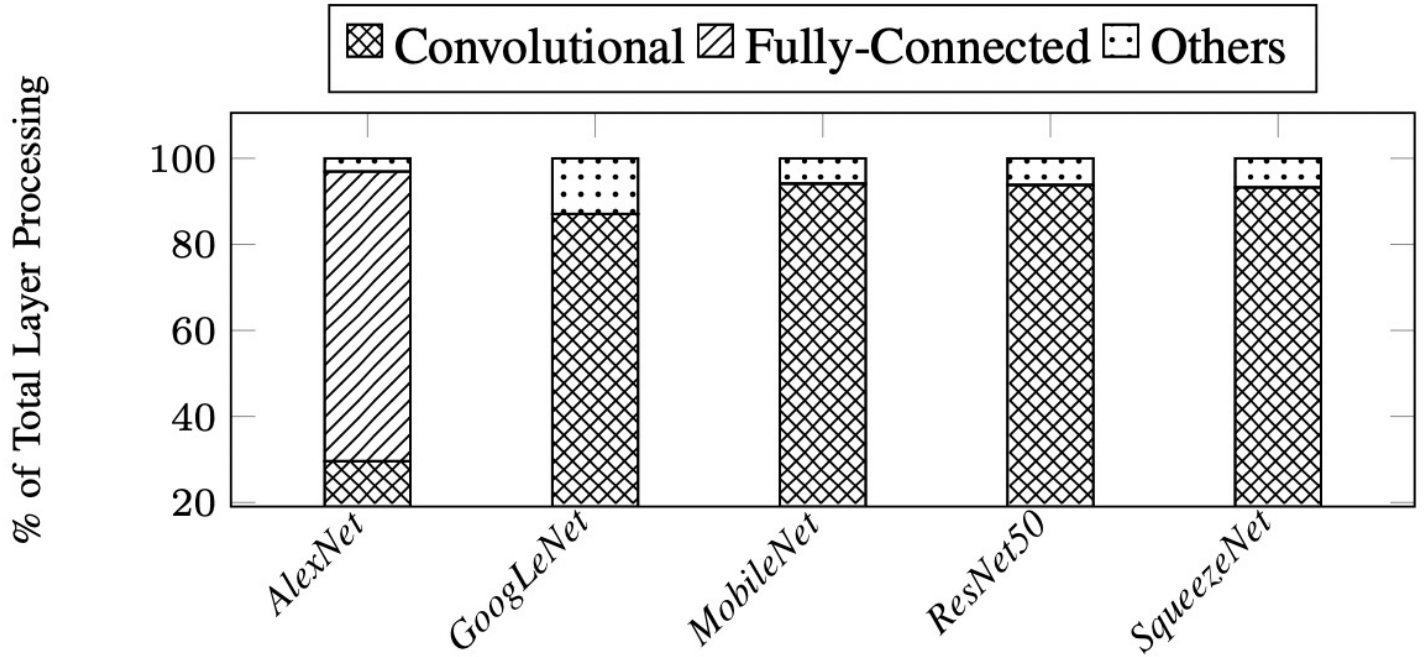


Fig. 6: The breakdown of CNN processing time between different layer types.

- **Conv/FC layers are the most time-consuming layers on GPUs**
- **Little performance benefit**
- **No area benefit**

S. Wang, G. Ananthanarayanan, Y. Zeng, N. Goel, A. Pathania and T. Mitra, "High-Throughput CNN Inference on Embedded ARM Big.LITTLE Multicore Processors," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 10, pp. 2254-2267, Oct. 2020, doi: 10.1109/TCAD.2019.2944584.