Robust Technology-Transferable Static IR Drop Prediction Based on Image-to-Image Machine Learning

C.-C. Lan, C.-C. Su, Y.-H. Lu, and Yao-Wen Chang

The Electronic Design Automation Lab
National Taiwan University

Outline

- Introduction
- Methodology
- Experimental Results
- Conclusion

Static IR Drop

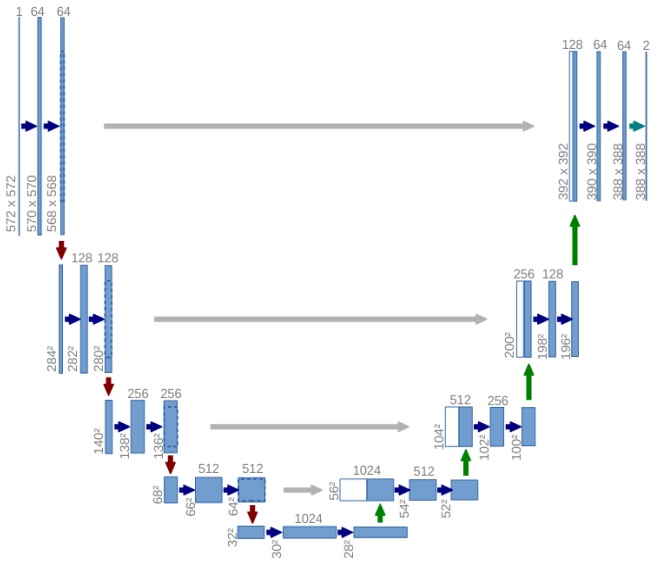
- Voltage drop between the power supply and circuit instances under steady-state conditions
  - Caused by the resistance of metals and vias in the power delivery network (PDN)

- Calculation of static IR drop invloves solving linear equations, where each equation corresponds to a node inside the PDN
  - Become extremely time-consuming for large designs

Machine Learning-based Prediction Works

- Cell-based prediction [Pao et al., DATE'20], [Ho et al., ICCAD'19], and [Kundu et al., VLSID'22]
  - Predict IR drop for a single cell per inference
  - Lack design transferability

- Image-to-image prediction [Chhabria et al., ASPDAC'21]
  - Predict IR drop for the whole circuit per inference with U-Net [Ronneberger et al., MICCAI'15]
  - Achieve design-transferability
  - Lack technology-transferability



The U-Net architecture

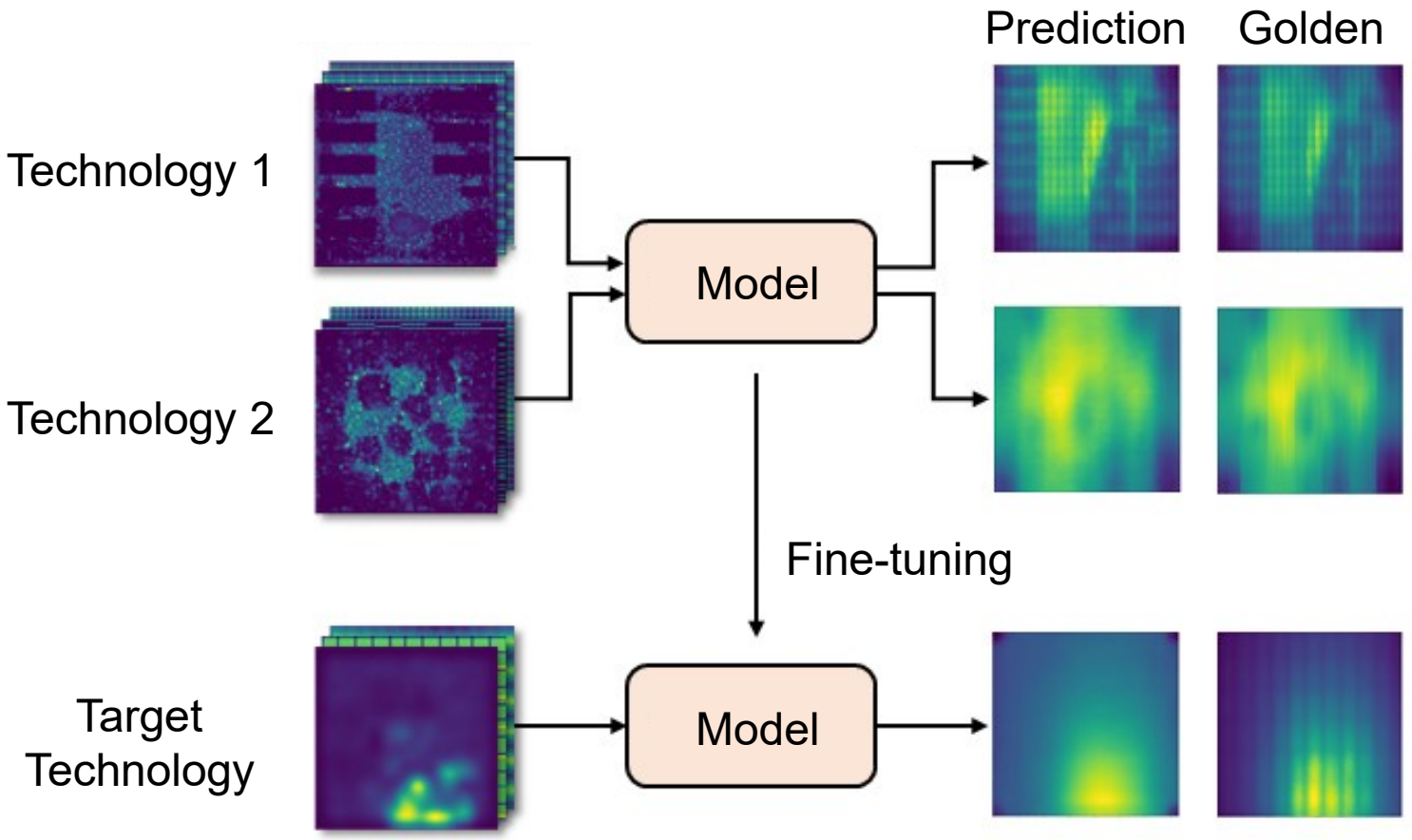
Motivation

- Previous works face challenges in real world application due to
  - Lack of technology-transferability
  - Imbalance training data across different technologies

- A prediction methodology that can transfer knowledge from one technology to another is needed

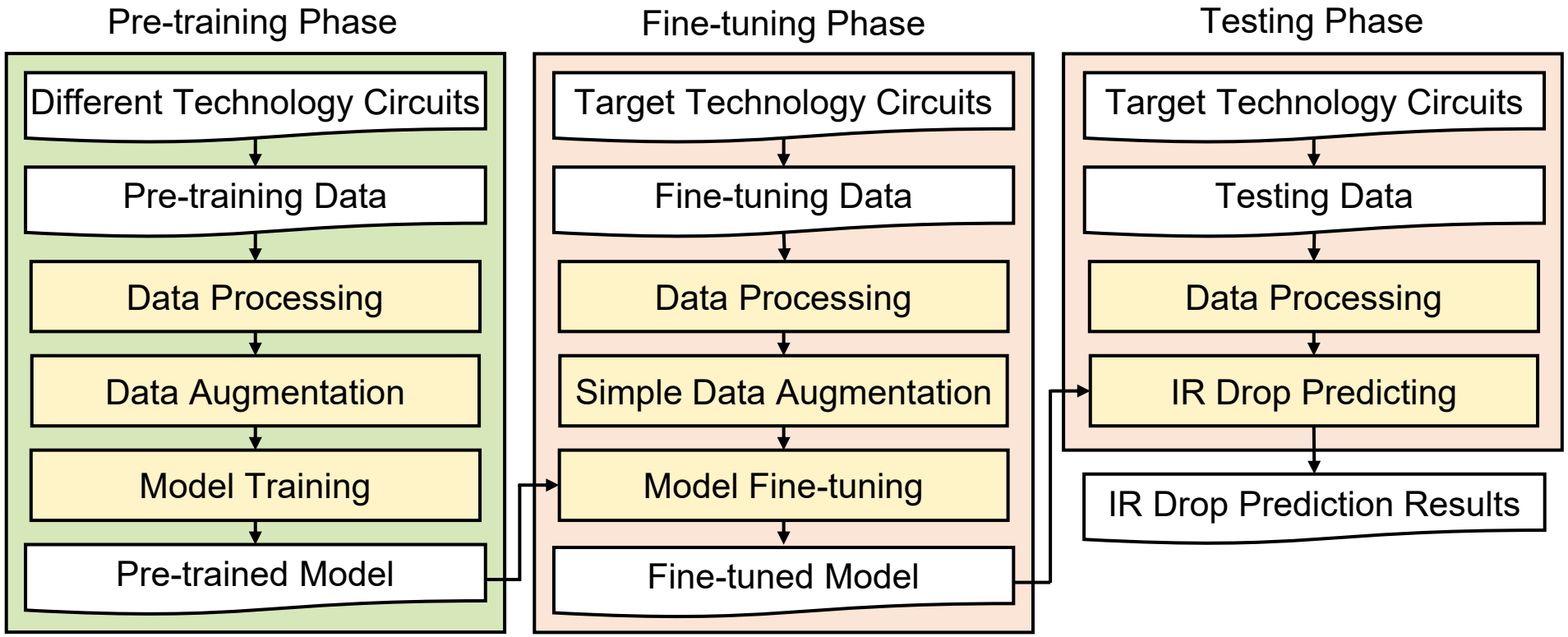
Overview

- We proposed the first technology-transferable prediction methodology for static IR drop



Pre-training and Fine-tuning

- The model can be pre-trained on different technologies and fine-tuned on target technology to predict IR drop for unseen design



IR Drop Prediction Problem

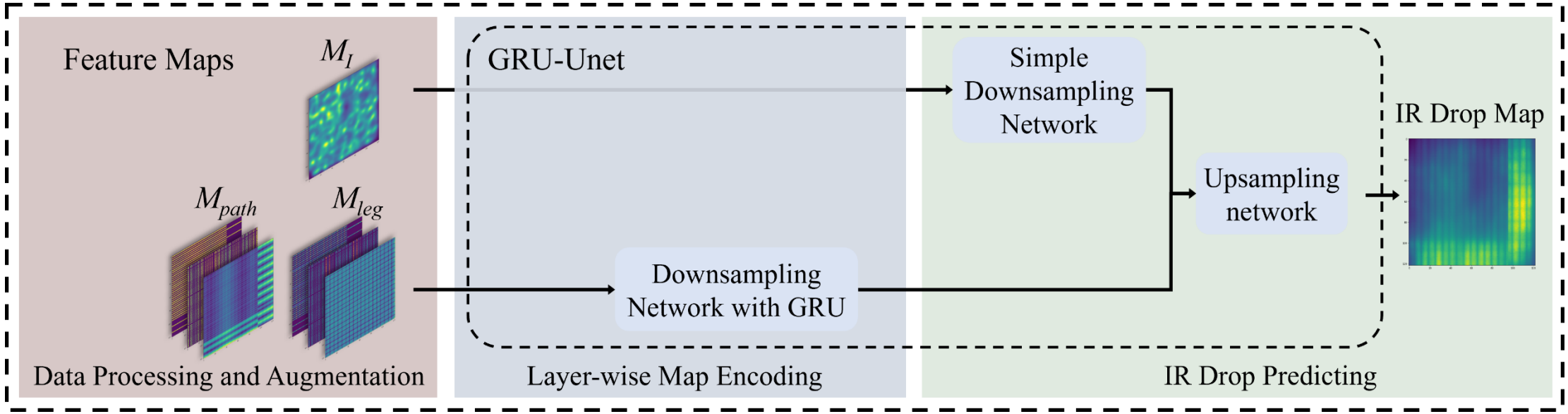
- Given
  - A netlist (SPICE file) that describes the PDN topology
  - A current distribution map
- Output
  - A predicted static IR drop map
- Objective
  - Minimize the mean absolute error (MAE)

Outline

- Introduction
- Methodology
- Experimental Results
- Conclusion

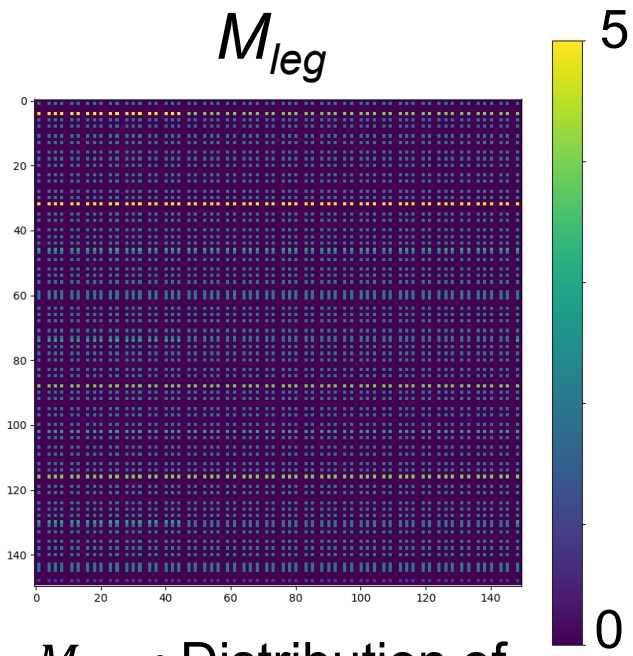
Prediction Methodology

- Our prediction flow consists of three main stages
  - Data Processing and Augmentation
  - Layer-wise Map Encoding
  - IR Drop Predicting

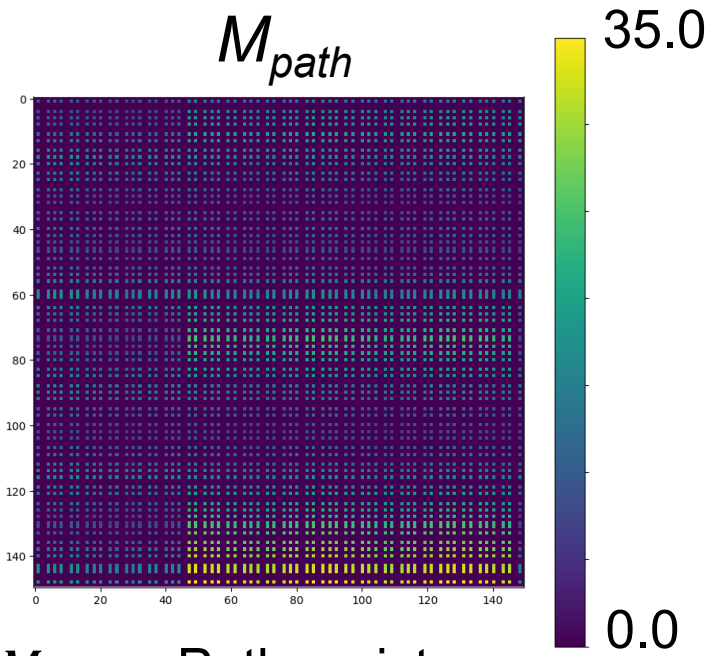


Data Processing and Augmentation (1/2)

- Layer-wise maps
  - Generate two layer-wise maps for each metal layer in the PDN



M_{leg} : Distribution of resistance pins, including metals and vias



M_{path} : Path resistance from a node to the nearest voltage source

Data Processing and Augmentation (2/2)

- Custom scaling augmentation
  - Scale the values of two of the followings: voltage, current, and resistance
  - Follow the Kirchhoff's Current Law (KCL) for each node, which is

$$I_i = \frac{V_1 - V_i}{R_{i1}} + \frac{V_2 - V_i}{R_{i2}} + \cdots + \frac{V_j - V_i}{R_{ij}} + \cdots$$

I_i : independent current source connected to node i

V_i : voltage of node i

R_{ij} : resistance between node i and node j

  - Performed in the pre-training phase

Data Processing and Augmentation (2/2)

- Custom scaling augmentation
  - Scale the values of two of the followings: voltage, current, and resistance
  - Follow the Kirchhoff's Current Law (KCL) for each node, which is

$$I_i = \frac{kV_1 - kV_i}{kR_{i1}} + \frac{kV_2 - kV_i}{kR_{i2}} + \cdots + \frac{kV_j - kV_i}{kR_{ij}} + \cdots$$

I_i : independent current source connected to node i

V_i : voltage of node i

R_{ij} : resistance between node i and node j

k : random scaling constant

  - Performed in the pre-training phase

Data Processing and Augmentation (2/2)

- Custom scaling augmentation
  - Scale the values of two of the followings: voltage, current, and resistance
  - Follow the Kirchhoff's Current Law (KCL) for each node, which is

$$\frac{1}{k} I_i = \frac{V_1 - V_i}{kR_{i1}} + \frac{V_2 - V_i}{kR_{i2}} + \cdots + \frac{V_j - V_i}{kR_{ij}} + \cdots$$

I_i : independent current source connected to node i

V_i : voltage of node i

R_{ij} : resistance between node i and node j

k : random scaling constant

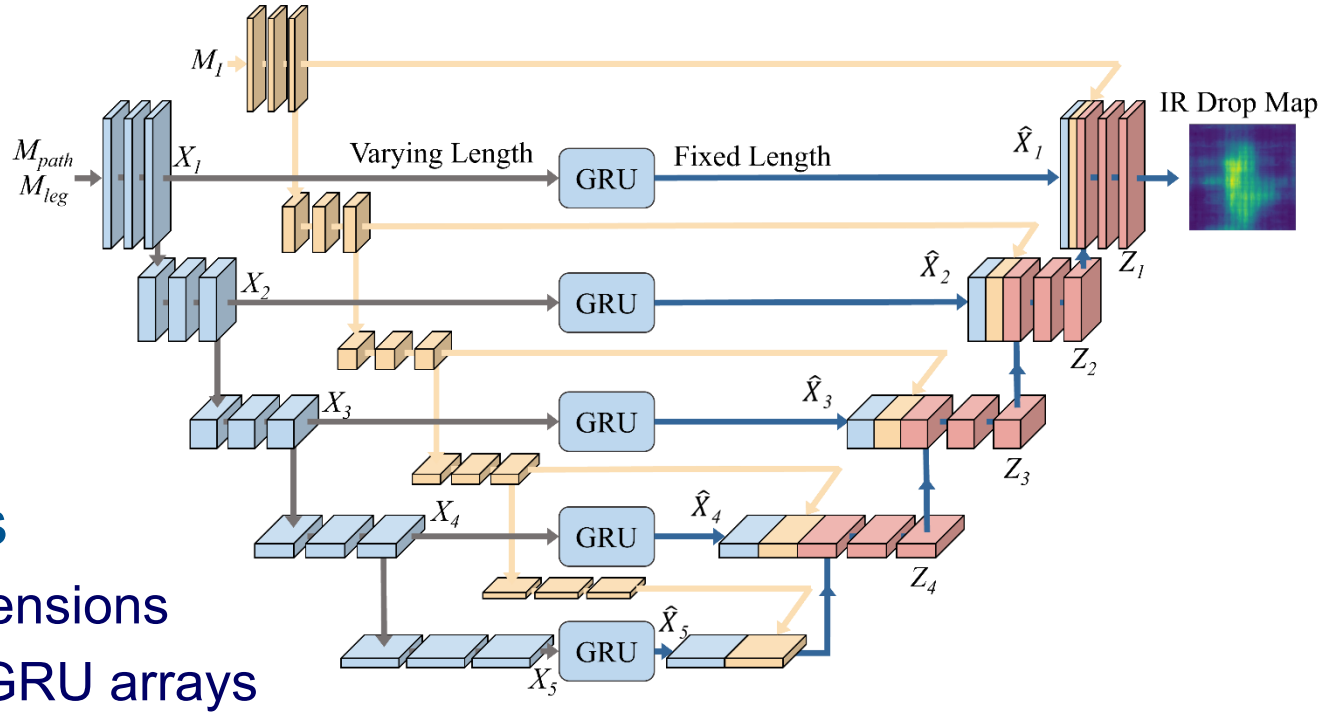
  - Performed in the pre-training phase

Layer-wise Map Encoding

- The number of layer-wise maps can vary
  - $2L$ layer-wise maps, where L is metal layer count in the PDN

- GRU-Unet handles PDNs with different number of layers
  - Downsampling network with GRU
  - Simple downsampling network
  - Upsampling network

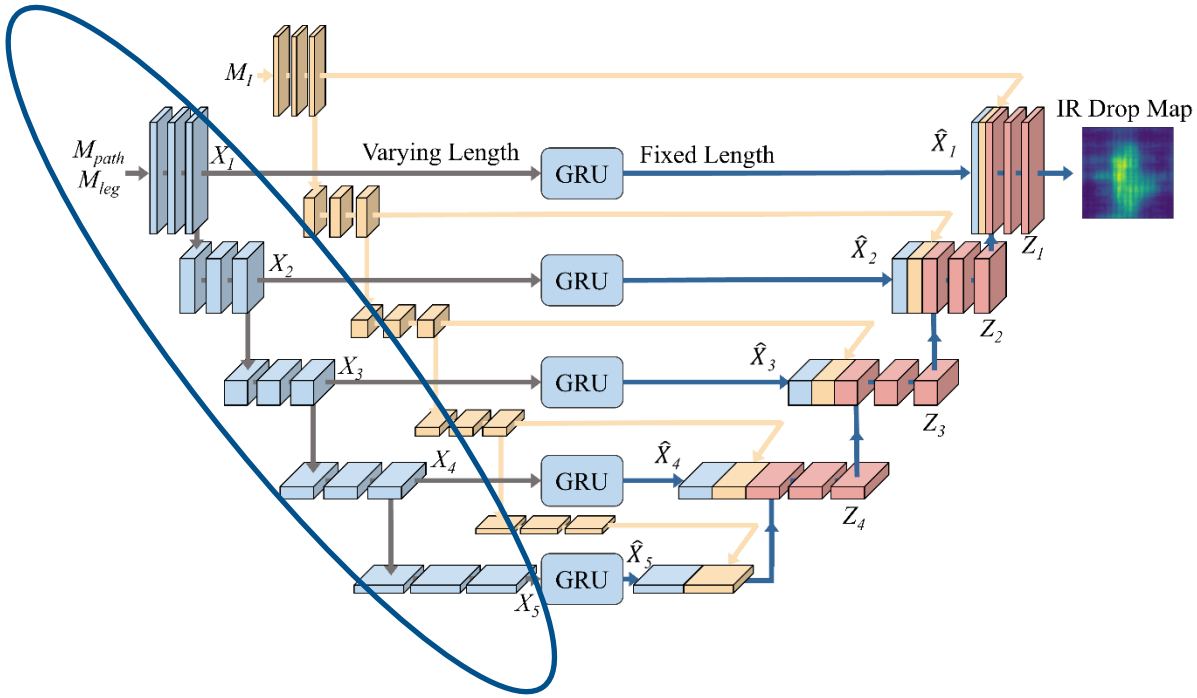
- In this stage, we encode the layer-wise maps with the downsampling network with GRUs
  - Aim to encode input with varying dimensions
  - Consist of downsampling network & GRU arrays



The GRU-Unet architecture

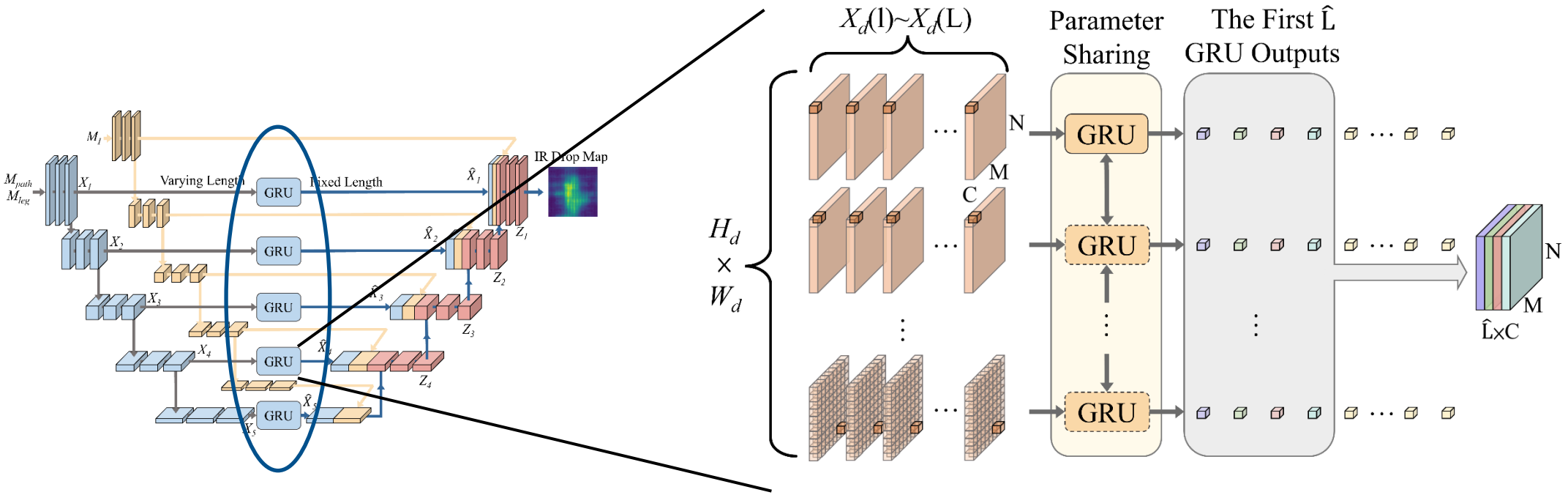
Layer-wise Map Encoding (1/2): Downsampling network

- Group the layer-wise maps
  - Each group has two feature maps (M_{leg},M_{path})
  - Total L groups
- Process each group with the downsampling network independently



Layer-wise Map Encoding (2/2): GRU Arrays

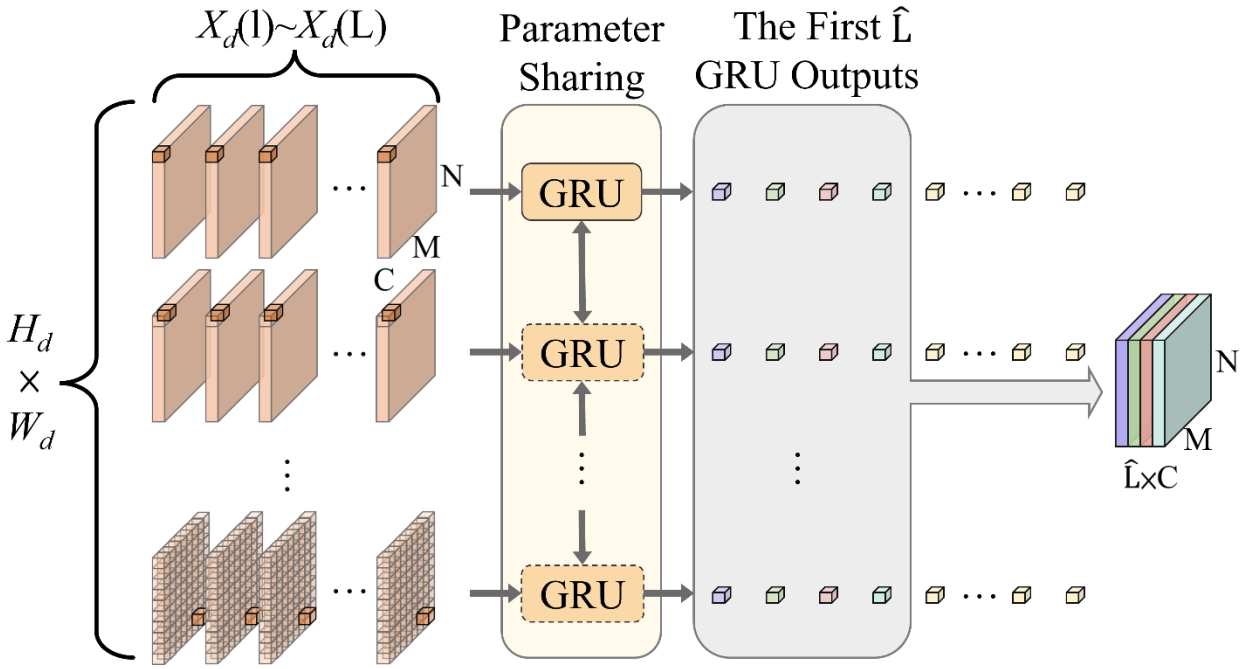
- Encode sequences of varying lengths with GRU units
  - Each sequence corresponds to a position



Encoding feature maps with GRU units

Parameter Sharing

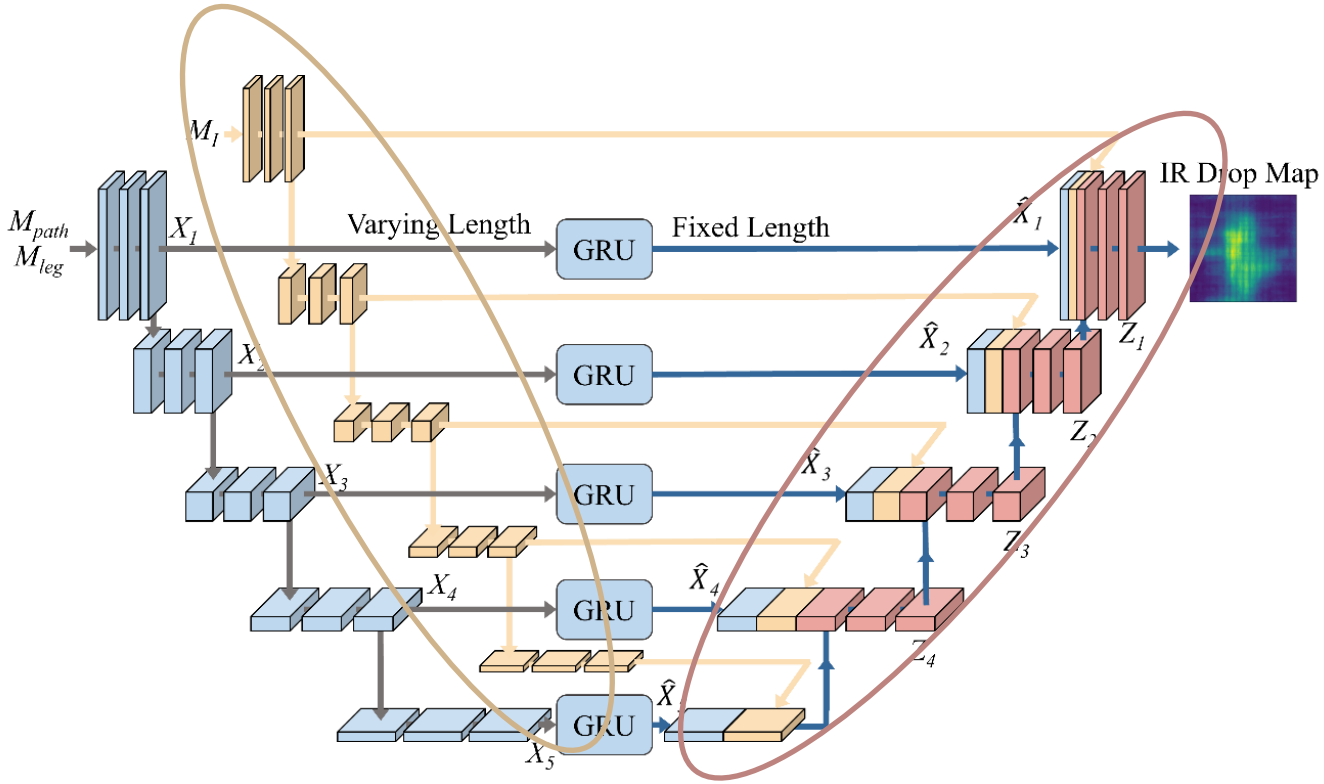
- The feature map size can vary due to different dimensions across designs
- Duplicate GRU units with shared parameters to process the sequences
- A constant length is sliced from each sequence to produce fixed-length embeddings



GRU units with shared parameter

IR Drop Predicting

- Predict the final IR drop map
  - Simple downsampling network generates embeddings for the current map M_I
  - Upsampling network combines the fixed-length layer-wise map embeddings and current map embeddings to generate the final IR drop prediction



Outline

- Introduction
- Methodology
- Experimental Results
- Conclusion

Experimental Results

- Setup
  - Platform: A single V100 GPU
  - Implemented in Python (PyTorch)
- Dataset: BeGAN [Chhabria et at., ICCAD'21]
  - Three technologies: Nangate 45nm, ASAP 7nm, and SkyWater 130nm
  - 100 circuits from each technology
  - PDNs generated and represented in SPICE netlists by OpenROAD and PDNSim
- We evaluate our methodology with four experiments

	#V (avg/max)	#I (avg/max)	#R (avg/max)	Area (mm^2) (avg/max)	#PDN layers	Vdd (V)	Rmin	Rmax
Nangate 45nm	19.3/49	74431/171054	99411/229600	0.332/0.859	5	1.1	0.017143	15
ASAP 7nm	2.6/4	8864/16129	42410/80722	0.008/0.016	5	0.7	0.001755	10
SkyWater 130nm	18.1/36	99895/199809	151784/331367	0.521/1.147	3	1.8	0.003525	4.5

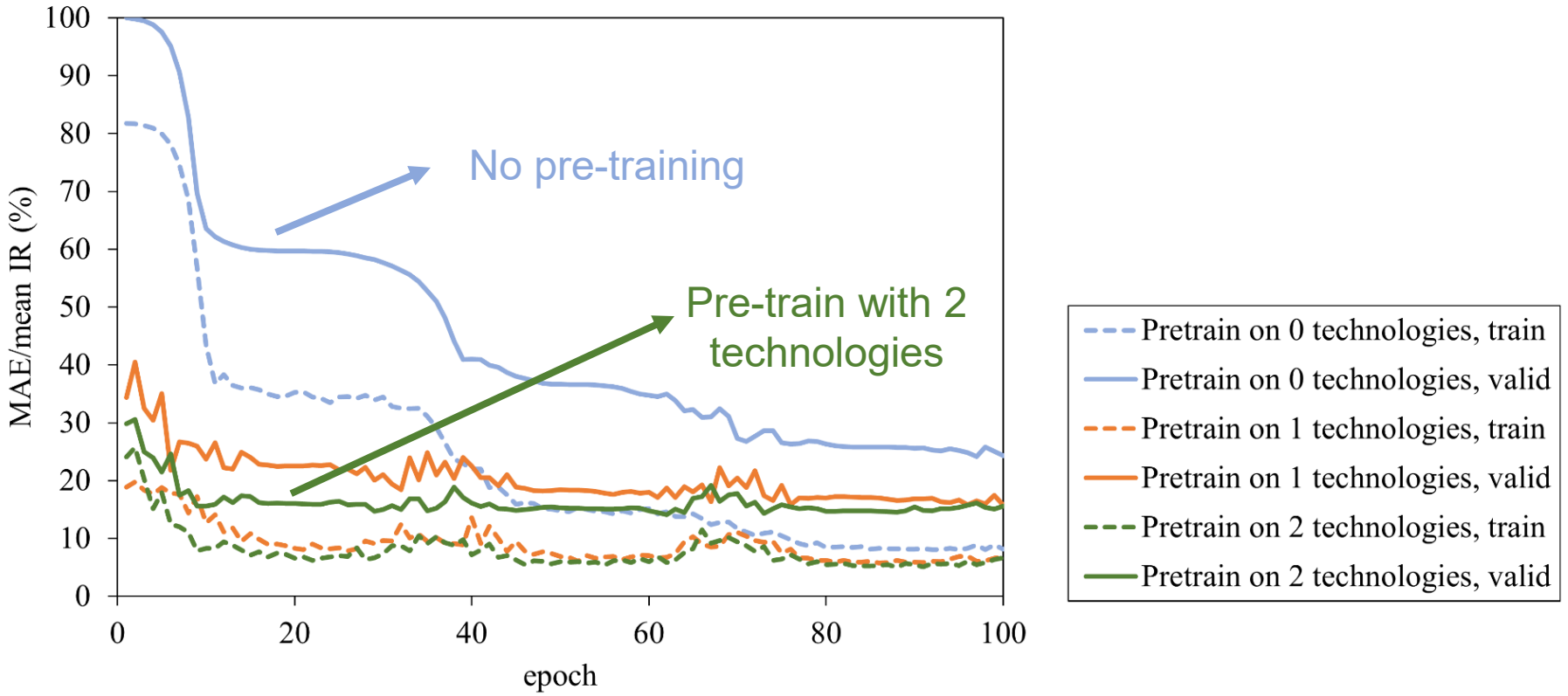
Prediction Quality

- Comparison with the winner of 2023 ICCAD CAD Contest Problem C
  - Achieve comparable average error and improved worst case error
  - Limitation: All designs in the contest share the same technology and number of PDN layers

	1^{st} place's MAE (10^{-3}mV)	Ours MAE (10^{-3}mV)	CC	NRMSE	Total time(s)
testcase 7	0.0656	0.0952	0.973	4.40%	1.151
testcase 8	0.0815	0.1477	0.958	5.75%	1.099
testcase 9	0.0406	0.0853	0.973	4.11%	2.612
testcase 10	0.0659	0.1463	0.954	6.29%	2.494
testcase 13	0.2067	0.1906	0.915	9.20%	0.215
testcase 14	0.4215	0.2962	0.928	9.69%	0.207
testcase 15	0.0968	0.1776	0.942	6.94%	0.797
testcase 16	0.1601	0.2435	0.943	6.02%	0.769
testcase 19	0.0905	0.0624	0.965	4.96%	2.795
testcase 20	0.1180	0.0734	0.974	4.37%	2.745
avg	0.1347	0.1518	0.953	6.17%	1.488
worst case	0.4215	0.2962	0.915	9.69%	2.795

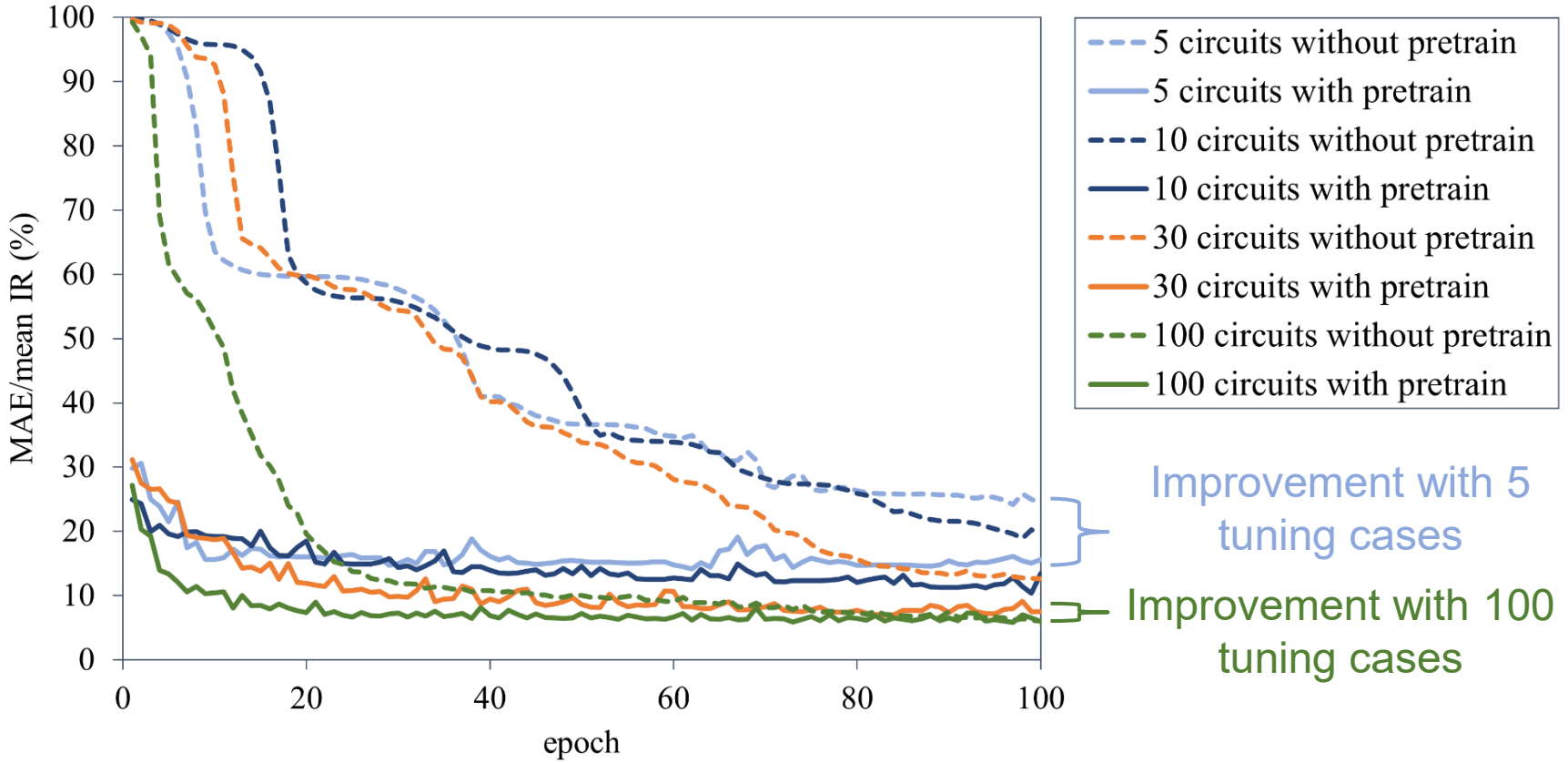
Technology Transferability and Robustness

- Evaluation on the effect of pre-training
  - Pre-train on 0/1/2 technologies and fine-tune on target technology
  - Improve the validation error with pre-training
  - Achieve technology-transferability



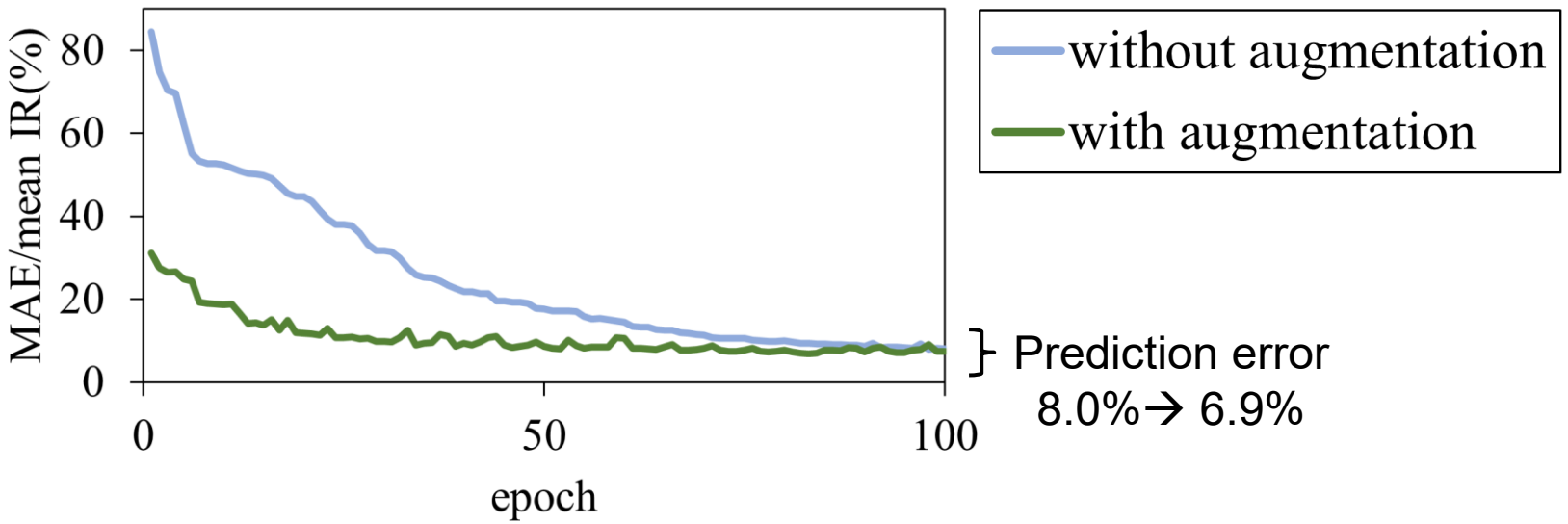
Effectiveness of Pre-training

- Evaluation of the generalization ability with different fine-tuning case amount
  - Pre-train on 2 technologies and fine-tune with 5/10/30/100 circuits
  - Improve significantly with less fine-tuning data



Effectiveness of Custom Augmentation

- Evaluation on the effect of proposed custom scaling augmentation
  - Pre-train w & w/o custom random scaling and fine-tuned on target technology
  - Enhance the convergence during fine-tuning
  - Slightly improve prediction quality



Outline

- Introduction
- Methodology
- Experimental Results
- Conclusion

Conclusion

- Proposed the first technology-transferable static IR drop-predicting methodology
- Developed layer-wise maps, effective and efficient input feature maps that encapsulate the PDN information
- Derived a generic ML model, GRU-Unet, to handle PDNs with different height, width, and layer counts
- Experimental results have shown that the proposed methodology achieved technology-transferability and made high-quality prediction comparable to the state-of-the-art predictors in their specialized cases